Project no. FP6-507752

MUSCLE

Network of Excellence
Multimedia **U**nderstanding through **S**emantics, **C**omputation and **LE**arning

A Review of Machine Learning Techniques for Processing Multimedia Content

Due date of deliverable: 31.09.2004
Actual submission date: 10.10.2004

Start date of project: 1 March 2004 Duration: 48 Months

Workpackage: 8
Deliverable: D8.1

Editors:
Pàdraig Cunningham
Rozenn Dahyot
Computer Science Department
Trinity College Dublin
Dublin 2, Ireland

Revision 1.0

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Keyword List:

Contributing Authors

Pangos Apostolis	TUC	apagos@ced.tuc.gr
Csaba Benedek	SZTAKI	bcsaba@sztaki.hu
Horst Bischof	TU Graz	bischof@icg.tu-graz.ac.at
Nadia Bolshakova	TCD	Nadia.Bolshakova@cs.tcd.ie
Sabri Boughorbel	INRIA- IMEDIA	
Patrick Bouthemy	INRIA-Vista	Patrick.Bouthemy@irisa.fr
Nozha Boujemaa	INRIA- IMEDIA	nozha.boujemaa@inria.fr
Enis Cetin	Bilkent	aeniscetin@yahoo.com
Sara Colantonio	CNR	Sara.Colantonio@isti.cnr.it
Matthieu Cord	ENSEA	cord@ensea.fr
Pádraig Cunningham	TCD	Padraig.Cunningham@cs.tcd.ie
Michel Crucianu	INRIA- IMEDIA	michel.crucianu@inria.fr
Rozenn Dahyot	TCD	dahyot@mee.tcd.ie
Valerie Gouet	INRIA- IMEDIA	Valerie.Gouet@inria.fr
Nizar Grira	INRIA- IMEDIA	
Michal Haindl	UTIA	haindl@utia.cas.cz
Ian Jermyn	INRIA- ARIANA	Ian.Jermyn@sophia.inria.fr
Christophe Laurent	France Télécom R&D	Christophe2.Laurent@francetelecom.com
Nathalie Laurent	France Télécom R&D	Nathalie.Laurent@rd.francetelecom.com
Mariette Maurizot	France Télécom R&D	Mariette.Maurizot@francetelecom.com
Horst Perner	IBAI	ibai@aol.com
Petra Perner	IBAI	ibaiperner@aol.com
David Patterson	UU	wd.patterson@ulster.ac.uk
Alexandros Potamianos	TUC	potam@telecom.tuc.gr
Naill Rooney	UU	NF.Rooney@ulster.ac.uk
Ovidio Salvetti	CNR	Ovidio.Salvetti@isti.cnr.it
Chen Sagiv	Tel Aviv University	chensagi@post.tau.ac.il
Vincent Samson	INRIA-Vista	Vincent.Samson@irisa.fr
Nir Sochen	Tel Aviv University	sochen@post.tau.ac.il
Petr Somol	UTIA	somol@utia.cas.cz
Fred Stentiford	UCL	f.stentiford@ee.ucl.ac.uk
Zoltan Szlavik	SZTAKI	szlavik@sztaki.hu
Tamás Szirányi	SZTAKI	sziranyi@lutra.sztaki.hu
Simon Wilson	TCD	swilson@tcd.ie

Contents

Chapter 1

Introduction

Machine Learning (ML)techniques can find application in situations where data is available in electronic format and ML algorithms can 'add value' by analysing this data. This is the situation with the processing of multimedia content. The 'added value' from ML can take a number of forms:

- by providing insight into the domain from which the data is drawn,
- by improving the performance of another process that is manipulating the data or
- by organising the data in some way.

This report presents a comprehensive review of the different ML techniques that have been applied to the processing of multimedia content. The three main areas covered in the report are; supervised learning, unsupervised learning, and dimensions reduction. A complete outline of the chapters in the report is as follows:

- *Chapter 3:* Supervised learning accounts for a lot of the research activity in ML and many supervised learning techniques have found application in the processing of multimedia content. This chapter describes the operation of the popular techniques such as Support Vector Machines, Bayesian Networks, Nearest Neighbour Classifiers, etc.
- *Chapter 4:* Unsupervised learning is also very important in the processing of multimedia content as clustering or partitioning of data in the absence of class labels is often a requirement. This chapter covers, Clustering, Unsupervised Bayesian Techniques, Self-Organising Feature Maps. It also contains a sub-section on Cluster Validity Analysis, a difficult issue in unsupervised learning.
- *Chapter 5:* The distinction between supervised and unsupervised learning is not black and white. Chapter 5 reviews semi-supervised learning, the grey area in between. Learning may be 'semi-supervised' in a number of different ways. For instance, the training data may be all (or almost all) of one class. Or all data is available for training but only some of it is labelled.

- *Chapter 6:* In addition to the supervised-unsupervised dimension, learning can also be considered along an *active-passive* dimension. In these terms, standard supervised learning is *passive*, whereas *active learning* refers to a situation where the learner has some control over the training it experiences.

- *Chapter 7:*A defining characteristic of multimedia data is that it is of high dimensions. This creates problems for machine learning algorithms so dimensions reduction is an important issue in the application of ML to the processing of multimedia data. There are two general approaches, *feature transformation* converts the data to a lower dimensions representation and the original features are discarded and *feature selection* selects a subset of the original features. Both approaches are reviewed in detail in Chapter 7.

Chapter 2

Task Types

The emergence of multimedia technology coupled with the rapidly expanding data collection, for private, industrial and public uses (e.g. self-made photos and videos repositories, Web resources, etc.) has attracted significant research efforts in providing tools for effective retrieval and management of the multimedia information.

At first, retrieval techniques were mainly performed using an index of keywords. This indexing technique, mainly done manually, has since shown its limitation, both for its accuracy and for its applicability on huge database such as the world-wide web where the daily growing data cannot be handled manually.

In the last two decades, several publications [15] and softwares (e.g. QBIC http://wwwqbic.-almaden.ibm.com) have been proposed to tackle the indexing issue, providing automatic tools to extract semantics from multimedia database. Most of them has focused on the analysis on one modality, visual or audio. Only recently, some interests has been shown to address the multimodal/multimedia aspect of the data in itself, in defining semantics using multi-sources data [3, 16, 1]. Another essential aspect of retrieval systems is their ability to interact with human users (cf. fig. 2.1). This involves as well multi-modal signal analysis to improve the human-machine communication. By gathering expertise in different signal processing research

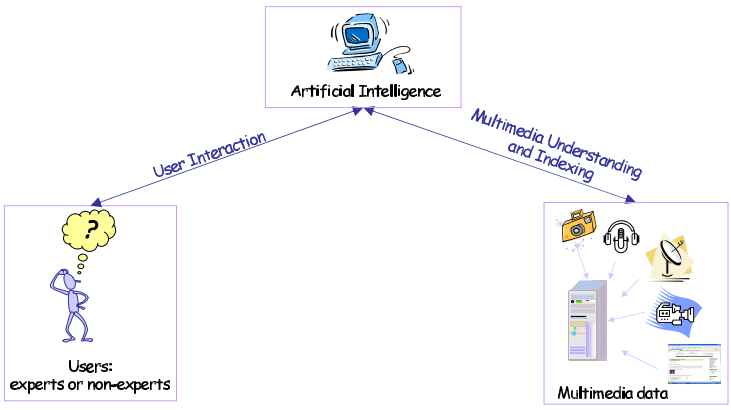


Figure 2.1: MUSCLE tasks: (1) Computer-aided understanding and indexing of Multimedia data, and (2) Human-Machine interaction for retrieval.

domains, the MUSCLE project aims at proposing new solutions to the difficult problem of the understanding of multimedia data at high semantic levels for retrieval.

Multimedia Understanding and Indexing. Multimedia data have a complex structure in terms of inter- and intra-document references (e.g. text refers to images, audio punctuates video) and are potentially an extremely valuable source of information. By its nature, the type of multimedia repository, for instance sport video databases or satellite image databases, constraints the set of information available. Moreover, the needs of the browsing application, usually defined by users, has to be defined a priori in order to link efficiently high-level concepts with the data. Some common objects like *people* or *faces* are already widely used as an elementary concepts used for browsing. Depending of the multimedia database, some specific objects can be defined if they are seen relevant for semantic extraction. The understanding of the context where the data have been recorded, and the application or information relevant in the recordings are essential priors to design efficient automatic semantic extracting tools. By their learning abilities, Machine Learning methods have therefore an essential role to play in the process of multimedia understanding, indexing and retrieval. Those are essential both to learn objects, events and semantics of interest, as for learning human needs and queries in the user interation process.

Human-Machine Interaction. One goal of the NoE MUSCLE is to automatically extract semantic information from multimedia data about people and their interactions with complex and natural environments. Human behaviour understanding has a wide range of future applications ranging from intelligent surveillance systems to advanced user interface systems [5, 7, 6, 10, 18, 9, 12, 2].

Development of digital libraries will require advanced user interface systems, which can interpret multimedia data in an automatic manner. Advanced user interface systems using speech and human motion understanding may provide control and command in high-level interaction with computerized systems and multimedia databases.

Early work on image retrieval systems is based on text input, in which the images are annotated by text and retrieval is performed on text [4]. However, manual annotation is labor-intensive and becomes impractical when the collection is large. Another problem is the subjective nature of keywords. The same image or video may be annotated differently with different annotators. Therefore, it is desirable to have a video database management system, including a Web-based graphical query interface, which can handle queries involving spatio-temporal and semantic properties of video data [8]. A spatio-temporal query may contain any combination of directional, topological, 3D-relation, object-appearance, trajectory-projection and similarity-based object-trajectory conditions. In this way, videos containing human activity and images containing humans in a database can be retrieved in an efficient manner. Such a system can interact with the user over the Internet through a graphical query interface as well.

It is also desirable to have a multimodal natural language interfaces for web search engines. The user will be able to perform web queries using natural language or speech input to the system. Such a system can engage the user in multimodal dialogue trying to further constrain or relax the query and better match the user's search requirements and it will be important to summarize the web query results and create visualizations effects.

Natural language processing and speech understanding has already been used in early human-machine interfaces. Vision is very useful to complement speech recognition and natural language understanding for more natural and intelligent communication between human and machines. More detailed cues can be obtained by gestures, body poses, facial expressions, etc [11, 14, 19, 17, 13]. Hence, future systems must be able to independently sense the surrounding environment, e.g., detecting human presence and interpreting human behavior.

In order to extract semantic information humans in the video have to be detected as a first step. Humans can be stationary or moving. Detection of stationary persons in the video is equivalent to detection of persons in still images. In order to reach a robust decision, one can take advantage of the associated audio and text (if available) as well. If there is human speech in the audio portion of the video then this may be used as an additional clue to reach a final decision. Therefore, following problems should to be solved in a semantic information system before extracting useful knowledge from multimedia data:

- Stationary human and human body parts detection in image and video,
- moving object detection and classification in video,
- tracking moving objects in video and gait analysis,
- speech detection in audio, and annotation of video using associated speech data, and
- analysis of the available text for human behaviour understanding in video.

Bibliography

[1] W. H. Adams, G. Iyengar, C.-Y. Lin, M. R. Naphade, C. Neti, H. J. Nock, and J. R. Smith. Semantic indexing of multimedia content using visual, audio and text cues. Technical report, IBM Thomas J. Watson Research Center, USA, november 2002.

[2] B.A. Boghossian and S.A. Velastin. Image processing system for pedestrian monitoring using neural classification of normal motion patterns. *Measurement and Control*, 32(9):261–264, 1999.

[3] R. Brunelli, O. Mich, and C. M. Modena. A survey on video indexing. *Visual Communication and Image Representation*, 2(10):78–112, 1999.

[4] S. Chang and A. Hsu. Image information systems: Where do we go from here? *IEEE Trans. on Knowledge and Data Enginnering*, 4(5):431–442, October 1992.

[5] R.T. Collins, A. Lipton, and T. Kanade et al. A system for video surveillance and monitoring. Technical report, CMU-RI-TR-00-12, 2000.

[6] R.T. Collins, A.J. Lipton, and T. Kanade. A system for video surveillance and monitoring. In *American Nuclear Society (ANS) Eighth International Topical Meeting on Robotics and Remote Systems*, Pittsburgh, PA, April 25-29 1999.

[7] R.T. Collins, A.J. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):745–746, 2000.

[8] M.E. Dönderler, E. Saykol, O. Ulusoy, and U. Güdükbay. Bilvideo: A video database management system. *IEEE Multimedia*, 10(1):66–70, January/March 2003.

[9] D.M. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[10] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: real-time surveillance of people and their activities. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[11] Yi Li, Songde Ma, and Hanqing Lu. Human posture recognition using multi-scale morphological method and kalman motion estimation. In *IEEE International Conference on Pattern Recognition*, pages 175–177, 1998.

[12] S. Maybank and T. Tan. Introduction to special section on visual surveillance. *International Journal of Computer Vision*, 37(2), 2000.

[13] O.F. Ozer, O. Ozun, C.O. Tuzel, V. Atalay, and A.E. Cetin. Vision-based single-stroke character recognition for wearable computing. *IEEE Intelligent Systems*, 16(3):33–37, May-June 2001.

[14] J. Segen and S. Kumar. Shadow gestures: 3d hand pose estimation using a single camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 479–485, 1999.

[15] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[16] C. G. M. Snoek and M. Worring. Multimodal video indexing: A review of the state-of-the-art. Technical report, Intelligent Sensory Information Systems, University of Amsterdam, The Netherlands, december 2001.

[17] M. Turk. Visual interaction with lifelike characters. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 368–373, Killington, 1996.

[18] Liang Wang, Weiming Hu, and Tieniu Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, 2003.

[19] M-H. Yang and N. Ahuja. Recognizing hand gesture using motion trajectories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 468–472, 1999.

Chapter 3

Supervised Learning

The defining characteristic of supervised learning is the availability of annotated training data. The name invokes the idea of a 'supervisor' that instructs the learning system on the labels to associate with training examples. Typically these labels are class labels in classification problems. Supervised learning algorithms *induce* models from these training data and these models can be used to classify other unlabelled data. In the language of statistics, the label is the *dependent* variable while the features that describe the examples are the *independent* variables.

Rather than be a class label, the dependent variable might also be a numeric variable, in which case the learning task is a regression rather than a classification problem. Some of the supervised learning techniques described here are naturally suited to classification tasks and will not work readily for regression problems , i.e. the learning algorithm depends on the availability of class labels. This is true of support vector machines which find hyperplanes that maximally separate two classes of labelled data. It is also true for decision trees that partition a feature space in order to separate labelled examples. Nevertheless, variants of these techniques have been developed in order to handle regression problems.

In the following sections we provide in-depth descriptions of the operation of the important supervised learning techniques together with examples of their application on multi-media data. The techniques that are covered are; Support Vector Machines, Decision Trees, Nearest Neighbour Classification, Bayesian Techniques and Neural Networks. There is also a subsection on Ensembles since almost all supervised learning techniques benefit from the ensemble approach, i.e. the formation of a *committee* of classifiers to address a problem. The section concludes with a brief discussion of the merits and demerits of these techniques for processing multimedia data.

3.1 Support Vector Machines

Support Vector Machines (SVM) are a new type of learning algorithms developed in the 1990s. They are based on results from statistical learning theory introduced by Vapnik [254]. These learning machines use kernels, which are a central concept for a number of learning tasks. The kernel framework is used now in a variety of fields, including multimedia information retrieval (see for instance [249, 261] for CBIR applications), bioinformatics, pattern recognition.

We focus here on the introduction of SVM for binary classification. Complete introduction

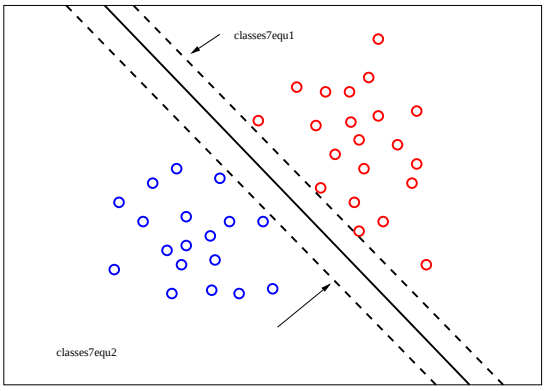


Figure 3.1: SVM margin

to SVM and kernel theory can be find in [39] and [235]. Starting with linear SV approach for classification, generalization producing nonlinear boundaries is then introduced, with practical implementation considerations, and applications to CBIR are discussed.

3.1.1 Linear classification

We assume here that both classes are linearly separable. Let $(\mathbf{x}_i)_{i\in[1,N]}$, $\mathbf{x}_i \in \mathbb{R}^p$ be the feature vectors representing the training data, and $(y_i)_{i\in[1,N]}$, $y_i \in \{-1,1\}$ be their respective class labels. Let define a hyperplane by $<\mathbf{w},\mathbf{x}>+b=0$ where $\mathbf{w}\in I\!R^p$ and $b\in I\!R$. Since the classes are linearly separable, we can find a function f, $f(\mathbf{x})=<\mathbf{w},\mathbf{x}>+b$ with

$$y_i f(\mathbf{x}_i) = y_i(<\mathbf{w},\mathbf{x}_i>+b) > 0,\ \forall i\in[1,N] \tag{3.1}$$

The decision function may be expressed as $f_d(\mathbf{x}) = sign(<\mathbf{w},\mathbf{x}>+b)$ with

$$f_d(\mathbf{x}_i) = sign(y_i),\ \forall i\in[1,N]$$

Since many functions realize the correct separation between training data, additional constraints are used. The SVM classification method aims at finding the *optimal* hyperplane based on the maximization of the *margin*[1] between the training data for both classes (see Figure 3.1).

Because the distance between a point $\mathbf{x}$ and the hyperplane is $\frac{yf(\mathbf{x})}{||\mathbf{w}||}$, it is easy to show [39] that the optimization problem may be expressed as the following minimization:

$$\min\frac{1}{2}||\mathbf{w}||^2 \text{ subject to } y_i(<\mathbf{w},\mathbf{x}_i>+b)\geq 1,\ \forall i\in[1,N] \tag{3.2}$$

The *support vectors* are the training points for which we have an equality in Eq. 3.2. They are all equally close to the optimal hyperplane. One can prove that they are enough to compute the separating hyperplane (whence their name).

This is a convex optimization problem (quadratic criterion, linear inequality constraints). Usually, the dual formulation is favored for its easy solving with standard techniques. Using

[1]The margin is defined as the distance from the hyperplane of the closest points, on either side.

Lagrange multipliers, the problem may be re-expressed in the equivalent maximization on α (dual form):

$$\alpha^\star = \underset{\alpha}{\operatorname{argmax}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j > \tag{3.3}$$

with:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \forall i \in [1,N] \quad \alpha_i \geq 0$$

The hyperplane decision function can be written as:

$$f_d(\mathbf{x}) = sign(\sum_{i=1}^{N} y_i \alpha_i^\star < \mathbf{x}, \mathbf{x}_i > + b) \tag{3.4}$$

3.1.2 Soft margin

The previous method is applicable to linearly separable data. When data can not be linearly separated, the linear SVM method may be adapted. A soft margin may be used too in order to get better efficiency in a noisy situation.

The idea is still to build a hyperplane, but the objective now is to *minimize* the classification error. In order to carry out the optimization, the constraints of Eq; 3.1 are relaxed using slak variables ξ_i:

$$y_i f(\mathbf{x}_i) = y_i(< \mathbf{w}, \mathbf{x}_i > + b) > 1 - \xi_i, \text{ with } \xi_i \geq 0 \ \forall i \in [1,N] \tag{3.5}$$

The minimization may be then expressed as follows:

$$\min \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{N} \xi_i \tag{3.6}$$

with a constant $C > 0$, subject to the constraints (3.5) .
The dual representation is obtained in a similar way:

$$\alpha^\star = \underset{\alpha}{\operatorname{argmax}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j < \mathbf{x}_i, \mathbf{x}_j > \tag{3.7}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{and} \quad \forall i \in [1,N] \quad 0 \leq \alpha_i \leq C$$

It consists in a very simple adaptation by introducing a bound C in the initial equations [256]. The constant C is used to tune the tradeoff between having a large margin and few classification errors.

3.1.3 Kernel-based classification

The linear SVM classifier previously described finds linear boundaries in the input feature space. To get much more general decision surfaces, the feature space may be mapped into a larger space before achieving linear classification. Linear boundaries in the enlarged space translate to nonlinear boundaries in the original space.

Let us note the induced space $\mathcal{H}$ via a map Φ:

$$\begin{array}{rcl} \Phi & : & \mathbb{R}^p \rightarrow \mathcal{H} \\ & & \mathbf{x} \mapsto \Phi(\mathbf{x}) \end{array}$$

In the SVM framework, there is no assumption on the dimensionality of $\mathcal{H}$ which can be very large, and sometimes infinite. We just suppose that $\mathcal{H}$ is equiped with a dot product. Maximizing the Eq. 3.7 now requires the computation of dot products $< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$. In practice, $\Phi(\mathbf{x})$ is never computed due to the *kernel trick*: kernel functions $k(.,.)$, respecting some conditions (positive definite, Mercer's conditions [2]), are introduced such that:

$$k(\mathbf{x},\mathbf{y}) =< \Phi(\mathbf{x}), \Phi(\mathbf{y}) >$$

Using these functions allows us to avoid to explicitly compute $\Phi(\mathbf{x})$. Everything about the linear case also applies to nonlinear cases, using a suitable kernel k instead of the Euclidean dot product. The resulting optimization problem may be expressed as follows:

$$\begin{array}{l} \alpha^\star = \underset{\alpha}{\text{argmax}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{with} \begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \forall i \in [1, N] \quad 0 \leq \alpha_i \leq C \end{cases} \end{array} \tag{3.8}$$

Thanks to the optimal $\alpha^\star$ value, the decision function is:

$$f_d(\mathbf{x}) = sign(\sum_{i=1}^{N} y_i \alpha_i^\star k(\mathbf{x}, \mathbf{x}_i) + b) \tag{3.9}$$

The most popular choices for k in the SVM literature are:

- Gaussian radial basis function kernel:

$$k(\mathbf{x},\mathbf{y}) = e^{-\frac{1}{2}\left(\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right)^2} \tag{3.10}$$

- Sigmoid kernel: $k(\mathbf{x},\mathbf{y}) = tanh(\eta_1 < \mathbf{x},\mathbf{y} > +\eta_2)$
- Polynomial kernel: $k(\mathbf{x},\mathbf{y}) =< \mathbf{x},\mathbf{y} >^d$

[2] We are dealing with the class of kernels k that correspond to dot product in the enlarged space.

For multicategory problems, the linear SVM approach may be extended as any linear machines to create boundaries consisting of sections of hyperplanes. When linear discrimination is not efficient enough, an appropriate nonlinear mapping can be found.

For density estimation problems, adaptation of binary SVM machines is also proposed. The One-Class SVM method estimates the density support of a vector set $(\mathbf{x}_i)_{i\in[1,N]}$. With a kernel $k(\mathbf{x},\mathbf{x}) = 1$, this lead to the following optimization problem:

$$\begin{aligned} \alpha^\star &= \underset{\alpha}{\operatorname{argmax}} \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{with} &\begin{cases} \sum_{i=1}^{N} \alpha_i = 1 \\ \forall i \in [1,N] \quad 0 \leq \alpha_i \leq C \end{cases} \end{aligned} \tag{3.11}$$

3.1.4 Alternative Kernels

Kernel-based algorithms are enjoying great popularity in machine learning community. One of the main success of this algorithms is that the similarity between features is expressed in term of kernels which maps features to a high dimension space. This mapping allows the svm classifier to deal with highly non-linear features. kernels can be viewed also as inner products in the mapped space. The mapping is implicit, which means that kernel can be written without a need to the explicit expression of the mapping. The crucial condition that kernels should satisfy to be suitable for SVM is the symmetry and the positive definess known as the mercer condition. Conditionally positive definite kernels [237] is a more general than positive definite kernels. They are suitable SVM since the equilibrium constraint is verified. Non-positive definite kernels [182] are also used for SVL. Although no warranty of the convergence of the SVM dual problem to a unique optimum, good performances are obtained with such kernels. Tools for designing kernels can be found in [14, 45]. We present in the following a non exhaustive review on kernels for SVM.

3.1.4.1 Basic Kernels

The simplest kernel is the linear kernel which is the euclidean inner product. One of the most popular kernel is the RBF kernel [121]. scale factor is a tunnable hyperparameter. It can be viewed as the kernel bandwidth. Using the RBF Kernel, the SVM can be viewed as radial basis function network with gaussian kernels centered at the support vectors. The polynomial kernel is directional, i.e. the output depends on the direction of the two vectors in low-dimensional space. This is due to the inner product in the kernel. Thus the polynomial kernel is suited for problems where all the training data are normalized. In [77], different classes of kernels are studied such as stationary, anisotropic, reducible kernels, etc.

3.1.4.2 Compactly supported kernels

Compactly Supported kernels are defined as kernels that vanish whenever the distance between the two features exceeds a certain distance [80]. Their main advantage is that the obtained kernel

matrix is sparse, therefore linear algebra algorithms can be applied to enhance computation. In [77], compactly supported kernels are studied. The Mattern kernel is multiplied by a know compactly supported kernel to obtain a new family of kernels. In [228] a generalization of the spherical kernels to high dimension is introduced.

3.1.4.3 Kernels for structured data

- Convolution kernels [93]: They are suitable for feature space that are not attribute-value tuples .

  The basic idea behind convolution kernels is that the similarity between composite objects can be captured by a relation between the object and its parts. The kernel on the object is then made up from kernels defined on different parts. Convolution kernels are very general and can be applied in a various problems.

- Graph kernels: A graph is defined as a set of vertices and a set of edges. Graph kernels compare the structure of graphs such as how many subgraphs in common [74, 73, 119, 120].

- Diffusion kernels: The main idea behind diffusion kernel is that it is easier to describe the local neighborhood of an instance than describing the whole instance space [128, 258, 257]. The neighborhood might be defined as instances that differs only by one property.

- Generative model kernels: The first prominent kernel based on generative model is the fisher kernel [108, 109]. It is based on the gradient of log-likelihood of the generative model (a posteriori probability) with respect to the parameter. The fisher kernel is defined from the information fisher matrix deduce from the generative model. A general framework of defining generative-model kernels has been presented in [251]. The fisher kernel comes as a particular case of the "marginalized kernels".

3.1.4.4 Kernels for set of vectors

The set of vectors are features with no structure is defined between vectors . Many applications use set of vectors as a feature representation [8]. Image representation with salient points is an example of the set vector feature representation. The Kernel principal angles [270] compares the two set of local features by computing principal angles. This consists of finding maximum angles between local features under orthogonality constraint. In [9] haussdorf kernel is used for SVM object recognition, it is proved to be definite positive when salient point are characterized with image patch. In [127], the set of vectors is fitted with a gaussian distribution and a battacharryaa affinity is defined to measure the similarity between the two gaussian distributions.

3.1.4.5 String kernels

The traditional kernel for text classification is simply the inner product of two words into the text space representation [115, 265, 155, 45, 236]. A recent approach consists of comparing common subsequences in the two words [186]. The gap between the subsequences are penalized. This can be done using the total length in the two strings. The p-spectrum kernel [145]

counts how many contiguous sub-strings of length p the strings have in common. A recent kernel [281] based on local correlations has reported better classification accuracy.

3.1.5 SVM application to CBIR

Contrary to the early systems, focused on "full-automatic" strategies, recent approaches introduce human-computer interaction into CBIR [255]. Starting with a coarse query, the interactive process allows the user to refine his request as long as it is necessary. Many kind of interaction between the user and the system have been proposed, but most of the time, user information is binary annotations, *i.e.* relevant for images belonging to the searched category, and otherwise irrelevant.

Statistical learning approaches have been introduced in the CBIR context and have been very successful. As well the techniques modeling the searched category as a density probability function, as the discrimination methods significantly improve the effectiveness of the visual information retrieval task. However, CBIR has a very specific classification context. There are very few training data during the retrieval process, the input space dimension is usually very high, unlabeled data are available, *etc*. Thus classical learning schemes have to be adapted.

Support Vector Machines (SVM) are used in such a context because they are dedicated for binary classification and are well adapted to these specificities. They have good classification performances with few training data and high input space.

In CBIR, systems have to classify image databases with few training data. When very few labels are available, inductive SVM classification may have unexpected results. Bad configuration may happen when learning samples do not represent accurately the structure of data. Meanwhile, all unlabeled data (images from database) are available. If data are structured, unlabeled data should be useful for classification. LeSaux[233] proposes to adapt the SVM scheme using unlabeled data. Only one parameter (threshold b in Eq. 3.4) is modified for all the data. Joachims proposes a method to use unlabeled data: Transductive SVM [114] in a text retrieval context. This method computes labels for unlabeled data such as hyperplane separates data with maximum margin. It has been used in CBIR (see the SVM_{light} implementation proposed by Joachims). Unfortunately, no trend becomes apparent with the use of TSVM. It is very data-dependent, and, of course, time consuming [37].

Besides, active learning [40] may be helpful to carry out an efficient online retrieval strategy. The performance of inductive classification depends on the training data set. In interactive CBIR, all the images labeled during the retrieval session are added to the training set used for classification. As a result, the choice of the labeled images to add will change system performance. For instance, labeling an image which is very close to one already labeled will not change the current classification. Active learning strategies offer a natural framework for interactive image retrieval. Usual active learning strategies in statistical learning propose to choose elements with the less classification accuracy. Some researchers for instance Cohn [40], propose to train several classifiers with the same training data, and choose data where classifiers disagree the most. Other ones, Zhu for instance[280], propose to minimize a cost function (the risk) to determine images with the less classification accuracy. Combining both SVM classification and Active learning has provided efficient strategies in CBIR [248]. The SVM_{active} learning method [248] tries to focus the user on images whose classification is difficult. It asks

user to label *m* images closest to the SVM boundary. The closer to the margin an image is, the less reliable its classification is.

3.1.6 Face processing by using SVMs

In the last years face and facial expression recognition have attracted much attention though they have been studied for more than 20 years by psychophysicists, neuroscientists and engineers. Many research demonstrations and commercial applications have been developed from these efforts. Face processing systems mainly developed for human computer interaction, in security applications etc. Examples include access control to buildings, surveillance and intrusion detection. Automatic face processing systems has the appealing characteristic of not being an invasive tool, as compared with fingerprint or iris biometric systems. A first step of any face processing system is detecting the locations in images where faces are present and than the recognition task can be performed.

3.1.6.1 Detection of Faces

Face detection from a single image is a difficult task because of changes in scale, orientation, location and pose. Facial expression, (self)occlusion and lighting conditions also change the overall appearance of faces.
Support Vector Machines (SVMs) were first applied to face detection by Osuna [183]. The SVM is a learning technique for training polynomial or Radial Basis Functions classifiers. The decision surfaces in this technique are found by solving a linearly constrained quadratic programming problem. This optimization is challenging because the quadratic form is completely dense and the memory requirements grow with the square of the number of data points. Osuna et al. developed an efficient decomposition algorithm that guarantees global optimality, and can be used to train SVM's over very large data sets (50000 data points). The training time of their algorithm versus number of samples and SVs can be seen in 3.2. This system detects faces by exhaustively scanning an image for face-like patterns at many possible scales, by dividing the original image into overlapping sub-images and classifying them using an SVM with a 2nd order polynomial as kernel function and an upper bound C=200. The input to the system is a 19*19 sub-image. The proposed method was tested on two standard test sets. Test set A, contained 313 high-quality images with one face per image. Test set B, contained 23 images of mixed quality, with a total of 155 faces. The detection rate on test set A is 97%, on test set B is 74%. This algorithm seems to be a promising method to solve the problem of face detection. Instead of estimating the densities of face and non-face classes, it seeks to model the boundary of the two classes. However the algorithm handle only frontal views.
In the work of [207] a comparison of SVM-based methods for face detection is given. Two different SVMs were trained, one with a polynomial kernel and another one with RBF kernel. As all the faces share the same structure, there must be an underlying model that generates all instances of the face class. A PCA is applied to model the set of available faces and the SVMs were trained in eigenfaces spaces. The detection results vs. kernel's parameters can be seen in 3.3. The best classification rates are summarized in 3.1. In the paper [123] a multiple face detection algorithm was proposed for video surveillance applications. The ICA-SVM based pattern detection is performed on the candidate region extracted by motion, color and

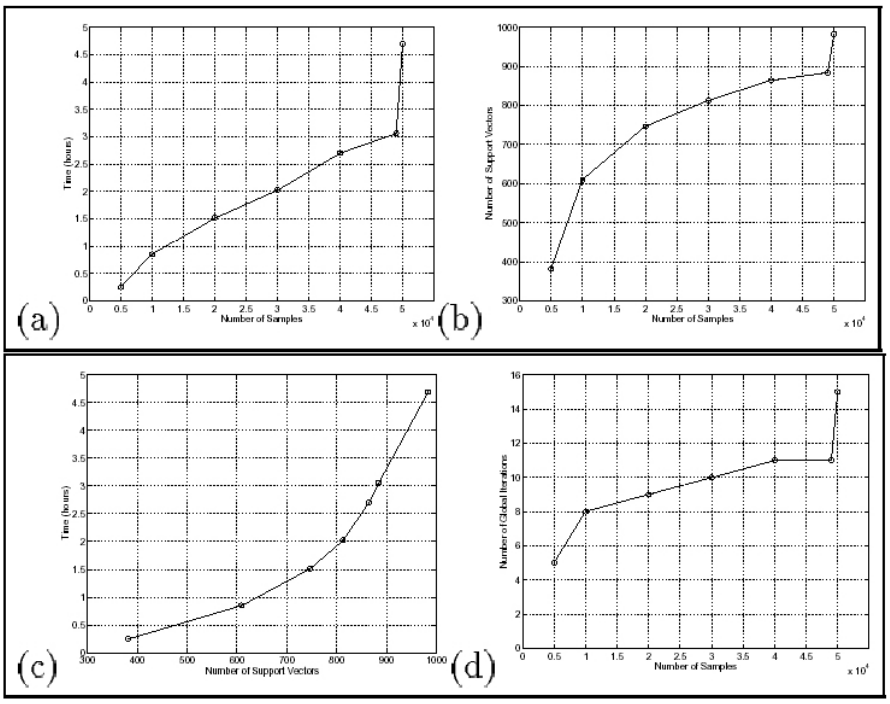
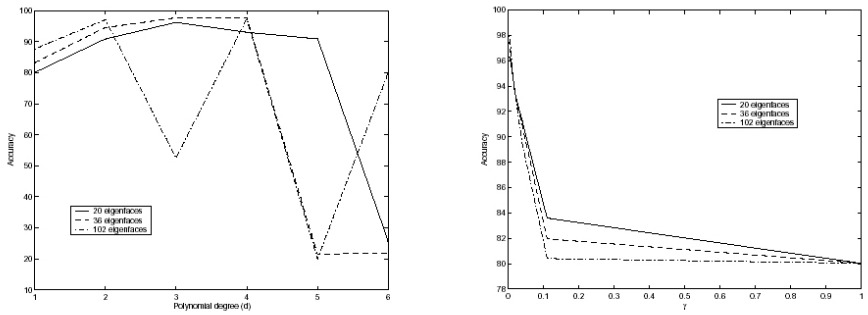


Figure 3.2: (a) Training Time on a SPARC station-20 vs. number of samples; (b) Number of Support Vectors vs. number of samples; (c) Training Time on a SPARC station-20 vs. number Support Vectors. Notice how the number of support vectors is a better indicator of the increase in training in training time than the number of samples alone; (d) Number of global iterations performed by the algorithm.



(a) Polynomial SVM (b) RBF SVM

Figure 3.3: Accuracy of two SVMs on the test set. The horizontal axis represents the values of the kernel parameter.

Classifier	Number of eigenfaces		
	20	36	102
Polynomial SVM	96.21%	97,86%	97.35%
RBF SVM	96.30%	97.41%	97.93%

Table 3.1: Top performances on the test set. The algorithms were tested on the BANCA test set.

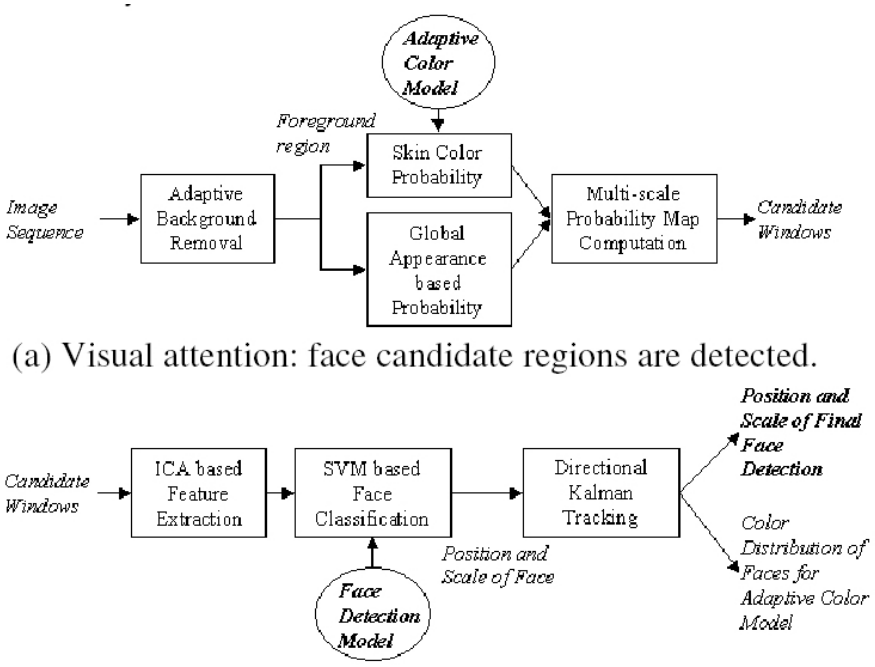


Figure 3.4: Architecture of the proposed algorithm.

global appearance information. The architecture of the proposed system can be seen in 3.4. A hybrid-learning scheme that combines ICA and SVM is used in the proposed algorithm. In low-level feature extraction, ICA produces statistically independent image bases. In high-level pattern classification, SVM classifies the ICA features as a face or non-faces. ICA features are expanded from not only face but also face-like class and also used the residual error in each class to dramatically enhance detection performance. Various face, face-like and non-face images were collected and normalized to 20*20 to train ICA and SVM. Normalized images were histogram equalized and dimensions were reduced to be 300 by oval masking. The face images included artificially rotated, shifted and resized faces. The face-like images were initially selected as the non-face images, which have small Euclidean distance to the average face image. 50 coefficients are used of the ICA bases, which span the face class and 50 coefficients of the ICA bases, which span the face-like class. Two more features are added, reconstruction errors in two classes. Euclidean distance measure is used for errors, assuming isotropic Gaussian data distribution. After extracting features of a given pattern, it is classified as a face or not using the trained polynomial SVM with a polynomial kernel with degree 2 and the number of support vectors is 467.

The proposed method was tested on two test sets. Set A contained 400 high-quality images

Classifier	Set A		Set B	
	Detect Rate	False Alarm	Detect Rate	False Alarm
Conventional*	97.2%	3	84.8%	127
Proposed**	98.5%	4%	90.1%	62

Table 3.2: Test result of the pattern detection step. * - 50 ICA features from face class; ** - 100 ICA features from face and face-like class and 2 residual errors.

	Detect Rate	False Alarms
Face candidate	97.66%	-
Face pattern	93.42	7

Table 3.3: Test results of the integrated algorithm.

with one face per image from Olivetti DB. Set B contained 36 images of mixed quality, with a total of 172 faces from CMU DB. Set A involved 1684800 pattern windows, while Set B involved 6178110. 3.2 shows that the proposed ICA feature expansion largely increases the detection performance. The proposed integrated algorithm was also quantitatively evaluated using 7000 640*480 stills from color video sequences. The video sequences were collected under various and changing conditions. The average execution time integrated algorithm is 250ms on a Pentium IV 1GHz PC. 3.3 shows two results of sequential steps. The first is face candidate detection through the background removal, color and global appearance. The latter is face detection through the pattern matching and tracking. Final face detection rate is 91.2%. In the paper [148] a combined Eigenface and SVM based method was proposed for multi-view face detection. Detecting faces across multiple views is more challenging than in a fixed view, e.g. frontal view, owing to the significant non-linear variation caused by rotation in depth, self-occlusion and self-shadowing. To address this problem, a novel approach is presented here. The view sphere is separated into several small segments. On each segment, a face detector is constructed. The pose of an image is explicitly estimated regardless of whether or not it is a face. A pose estimator is constructed using Support Vector Regression. The pose information is used to choose the appropriate face detector to determine if it is a face. With this pose-estimation based method, considerable computational efficiency is achieved. Meanwhile, the detection accuracy is also improved since each detector is constructed on a small range of views. Two tasks need to be performed for head pose estimation: constructing the pose estimators from face images with known pose information, and applying the estimators to a new face image. The method of SVM regression was adopted to construct two pose estimators, one for tilt (elevation) and the other for yaw (azimuth). The input to the pose estimators is the PCA vectors of face images; the dimensionality of PCA vectors can be reasonably small in the experiments (20, for example). The output is the pose angles in tilt and yaw. Compared with other learning methods, the SVM-based method has distinguishing properties such as:

- No model structure design is needed. The final decision function can be expressed by a set of 'important examples' (SVs).
- By introducing a kernel function, the decision function is implicitly defined by a linear combination of training examples in a high-dimensional feature space.

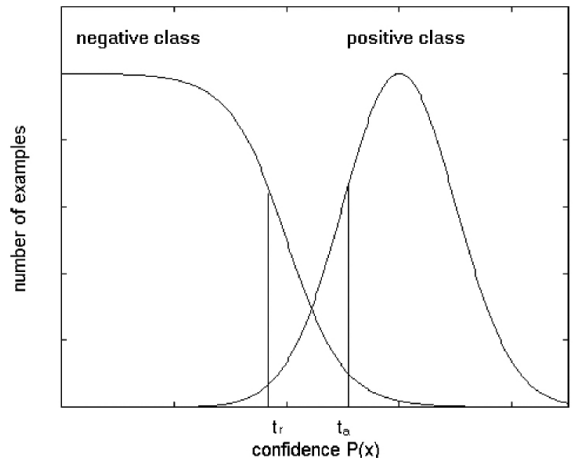


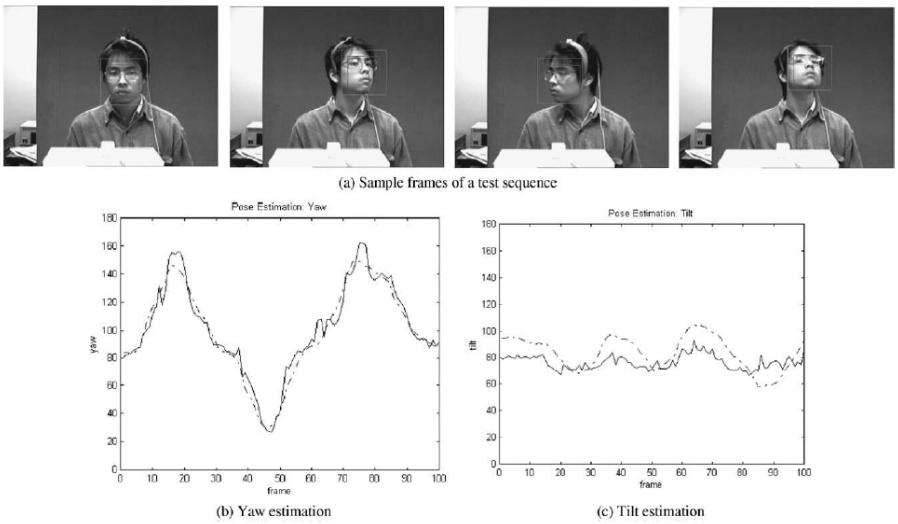
Figure 3.5: The hybrid method of Eigenface and SVM.

- The problem can be solved as a Quadratic Programming problem, which is guaranteed to converge to the global optimum of the given training set.

The detection process consists of a coarse detection phase by the Eigenface method followed by a fine SVM phase. A schematic illustration of the classification criterion of the hybrid method is given in 3.5. In the first phase, the probability density of each class is estimated as simply as possible. Two thresholds, a rejection threshold and an acceptance threshold are defined. For a test sample x; if the value of the probability of a pattern x being a face, returned by Eigenface method, is less than rejection threshold; it is rejected as a negative example. If the value is larger than acceptance threshold; it is accepted as positive. Otherwise, if the value falls between rejection and acceptance thresholds; it is considered as ambiguous and left to the SVM classifier in the next phase. The values of the two thresholds should be determined by the acceptable false positive and false negative rates which are usually application dependent.
An SVM-based classifier is trained using the examples in the middle region of 3.5 The classifier is only activated when an ambiguous pattern emerges. Usually the SVM-based classifier is computationally more expensive than the Eigenface method, but more accurate. However, since the proportion of the examples in the ambiguous region is relatively small, a significant improvement of the classification speed can be achieved. Furthermore, owing to the fact that the SVM classifier is trained only on the examples in the ambiguous region and not on the whole training set, the SVM classification problem is simplified to some degree. A more precise and compact set of SVs are obtained.
In the experiments the faces captured in the images are about 50*50 pixels. The range of pose of these face images is [-90,+90] degrees in yaw and [-30,+30] degrees in tilt. 3.6 shows estimated pose from a test sequence. The experimental results indicate that 3.7:

- The SVM method is the most accurate in terms of error in detection scale and location, but also the slowest;

- The Eigenface method is the fastest, but less accurate in certain frames;



(a) Sample frames of a test sequence

(b) Yaw estimation

(c) Tilt estimation

Figure 3.6: Pose estimation on a test sequence. In (b) and (c), the solid curves are the estimated pose in yaw and tilt and the dotted curves are the ground-truth pose which is measured by the data acquisition system.

- The hybrid method demonstrates the best balance between accuracy and speed; it is almost as accurate as the SVM method and not significantly slower than the Eigenface method in most frames.

3.1.6.2 Discussion

SVMs are proved to be excellent tool for classification very large-scale data. This review attempts to show how SVMs are used for face detection and recognition. As it can be seen from the results reported above SVMs outperforms all classical techniques, which have been introduced during last years, for face processing. Unfortunately, comparison of SVMs with all latest algorithms (neural networks, naive bayes classifier etc.) is not available in literature. However some concluding remarks can be made and questions for future research can be formulated:

- Classification with SVMs is computationally more demanding than other methods but more accurate. With respect to speeding up the classification (in most cases training can be made off line) the following questions arising:
  - How the number of SVs can be reduced or how to find a minimal subset of SVs that the performance of the classifier is still satisfactory?
  - How the process of training and testing can be parallelized?
- Most of the systems were tested on academic datasets with frontal faces. The systems should be tested in real circumstances.

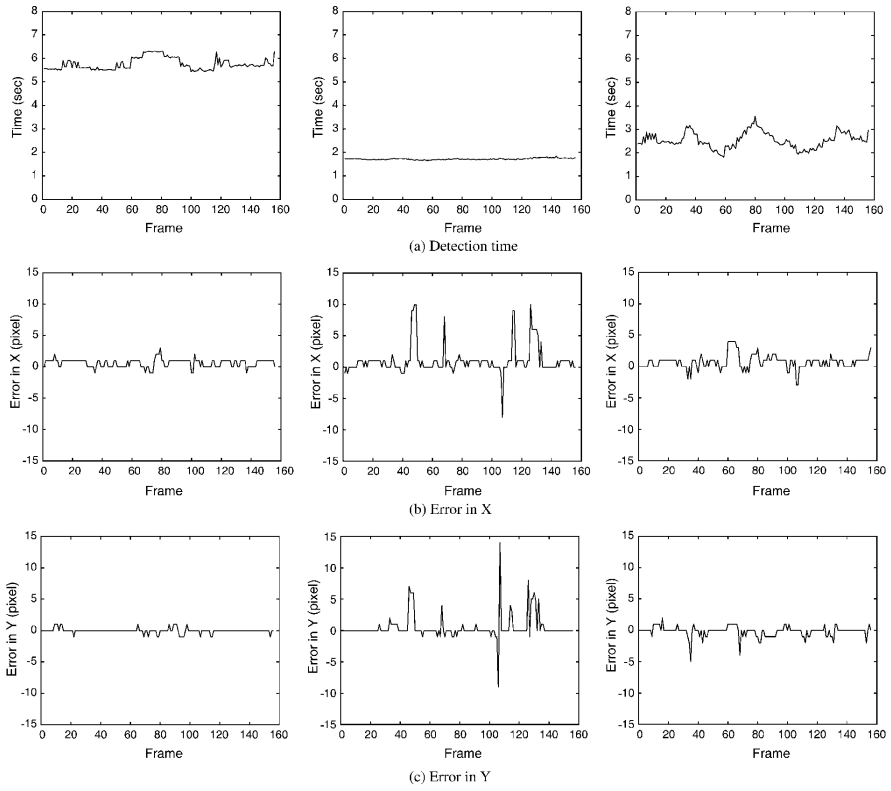


Figure 3.7: Comparison results of, from left to right, the SVM, Eigenface and hybrid methods for multi-view face detection on a test sequence: (a) shows the detection time in seconds on each frame; (b) and (c) are the position errors in pixels from the ground-truth position in horizontal and vertical direction, respectively.

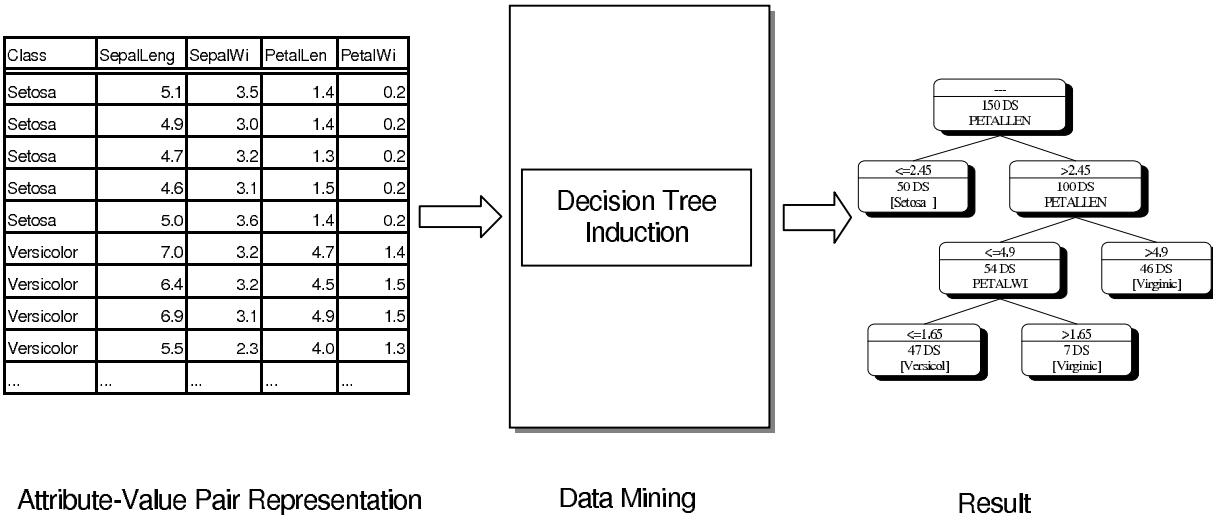


Figure 3.8: Basic principles in decision tree induction.

3.2 Decision Tree Induction

Decision tree induction allows one to learn a set of rules and basic features necessary for decision-making in a diagnostic task. The induction process does not only act as a knowledge discovery process, it also works as a feature selector, discovering a subset of features that is the most relevant to the problem solution. A decision tree partitions the decision space recursively into sub-regions based on the sample set (see Figure 3.8). In this way the decision tree recursively breaks down the complexity of the decision space. The outcome has a format, which naturally presents the cognitive strategy of human decision-making process. IThis learning methods is also called supervised learning since samples in the data collection have to be labelled by the class. Most decision tree induction algorithms allow the use of numerical attributes as well as categorical attributes. Therefore, the resulting classifier can make the decision based on both types of attributes.

3.2.1 Subtasks and Design Criteria for Decision Tree Induction

The overall procedure of the decision tree building process is summarized in Figure 3.9. Decision trees recursively split the decision space into subspaces based on the decision rules in the nodes until the final stopping criteria is reached or the remaining sample set does not suggest further splitting. For this recursive splitting the tree building process must always pick among all attributes that attribute which shows the best result on the attribute selection criteria for the remaining sample set. Whereas for categorical attributes the partition of the attributes values is given a-priori. The partition (also called attribute discretization) of the attribute values for numerical attributes must be determined.

It can be done before or during the tree building process [58]. We will consider the case where the attribute discretization will be done during the tree building process. The discretization must be carried out before the attribute selection process since the selected partition on the attribute values of a numerical attribute highly influences the prediction power of that attribute.

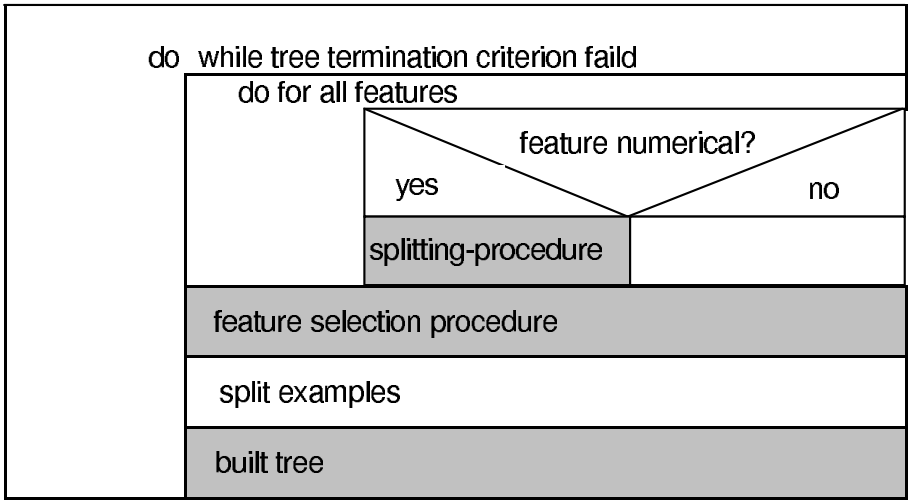


Figure 3.9: Basic principles in overall procedure for decision tree induction.

After the attribute selection criteria was calculated for all attributes based on the remaining sample set, the resulting values are evaluated and the attribute with the best value for the attribute selection criteria is selected for further splitting of the sample set. Then, the tree is extended by a two or more further nodes. To each node is assigned the subset created by splitting on the attribute values and the tree building process repeats. Attribute splits can be done:

- univariate on numerically or ordinal ordered attributes X such as $X \leq a$,
- multivariat on categorical or discretized numerical attributes such as $X \in A$, or
- linear combination split on numerically attributes $\sum_i a_i X_i \leq c$.

The influence of the kind of attribute splits on the resulting decision surface for two attributes is shown in Figure 3.10) The axis-parallel decision surface results in a rule such as:

$$IF \quad F3 \quad \geq \quad 4.9 \quad THEN \quad CLASS \leftarrow Virginica$$

while the linear decision surface results in a rule such as

$$IF \quad -3.272 + 0.3254 \times F3 + F4 \quad \geq \quad 0 \quad THEN \quad CLASS \leftarrow Virginica$$

The later decision surface better discriminates between the two classes than the axis-parallel one, see Figure 3.11). However, by looking at the rules we can see that the explanation capability of the tree will decrease in case of the linear decision surface.

The induced decision tree tends to overfit to the data. This is typically caused due to noise in the attribute values and class information present in the training set. The tree building process will produce subtrees that fit to this noise. This causes an increased error rate when classifying unseen cases. Pruning the tree which means replacing subtrees with leaves will help to avoid this problem. Now, we can summarize the main subtasks of decision tree induction as follows:

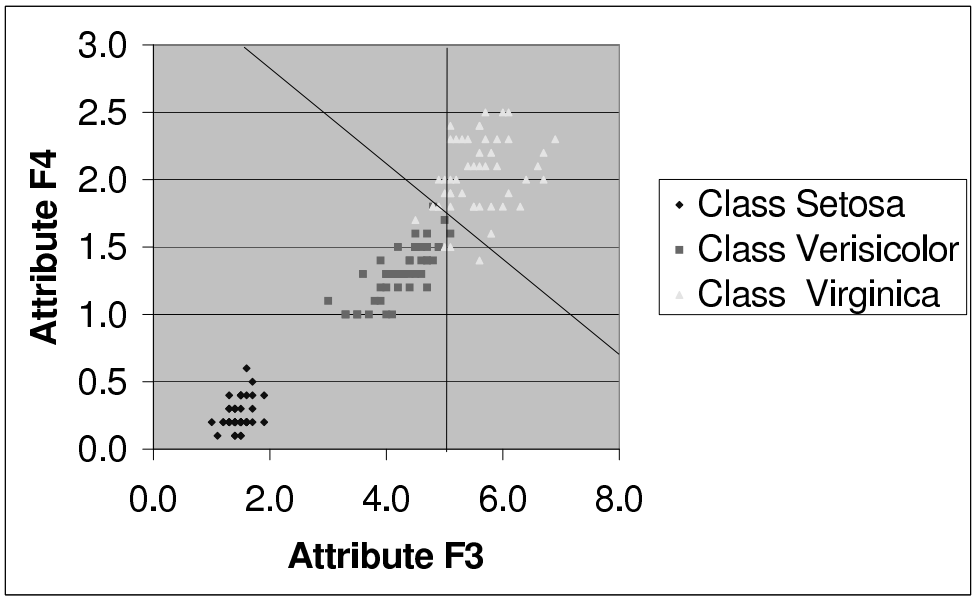


Figure 3.10: Axis-Parallel and linear attribute splits graphically viewed in decision space.

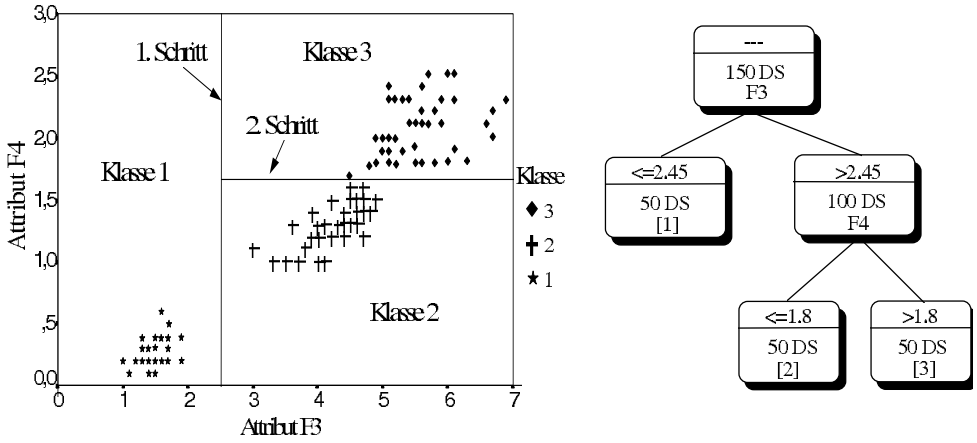


Figure 3.11: Demonstration of recursively splitting of decision space based on two attributes of the IRIS data set.

- attribute selection (Information Gain [210], χ^2 Statistic [122] ,Gini-Index [29] , Gain Ratio [212]),
- attribute discretization (Cut-Point [210] , Chi-Merge [122], LVQ [204], MLDP [66], Histogram [204] , Hybrid Methods [204]), and
- pruning (Cost-Complexity [29] , Error Reduction [210] , Confidence In-terval Method [211] , Minimal Error [178]).

Beyond that, decision tree induction algorithm can be distinguished in the way they access the data and in non-incremental and incremental algorithms. Some algorithms access the whole data set in the main memory of the com-puter. This is insufficient if the data set is very large. Large data sets of millions of data do not fit in the main memory of the computer. They must be assessed from disk or other storage device so that all these data can be mined. Accessing the data from external storage devices will cause long execution time. However, the user likes to get results fast and even for exploration purposes he likes to carry out quickly various experiments and compare them to each other. Therefore, special algorithm have been developed that can work efficiently although using external storage devices. Incremental algorithm can update the tree according to the new data while non-incremental algorithm go trough the whole tree building process again based on the combined old data set and the new data. Some standard algorithm are: CART, ID3, C4.5, C5.0, Fuzzy C4.5, OC1, QUEST, CAL 5.

3.2.2 Discretization of Attribute Values

A numerical attribute may take any value on a continuous scale between its mini-mal value x_1 and its maximal value x_2. Branching on all these distinct attribute values does not lead to any generalization and would make the tree very sensitive to noise. Rather we should find meaningful partitions on the numerical values into intervals. The intervals should abstract the data in such a way that they cover the range of attribute values belonging to one class and that they separate them from those belonging to other classes. Then, we can treat the attribute as a discrete vari-able with $k+1$ intervals. This process is called discretization of attributes. The points that split our attribute values into intervals are called cut-points. The cut-points k lies always on the border between the distribution of two classes. Discretization can be done before the decision tree building process or during decision tree learning [58]. Here we want to consider discretization during the tree building process. We call them dynamic and local discretization methods. They are dynamic since they work during the tree building process on the created subsample sets and they are local since they work on the recursively created sub-spaces. If we use the class label of each example we consider the method as su-pervised discretization methods. If we do not use the class label of the samples we call them unsupervised discretization methods. We can partition the attribute val-ues into two (k=1) or more intervals ($k>1$). Therefore, we distinguish between bi-nary and multiinterval discretization methods. In Figure 3.12, we see the conditional histogram of the values of the attribute petal_length of the IRIS data set. In the binary case (k=1), the attribute values would be splitted at the cut-point 2.35 into an interval from 0 to 2.35 and a second interval from 2.36 to 7. If we do multi-interval discretization, we will find another cut-point at 4.8. That groups the values

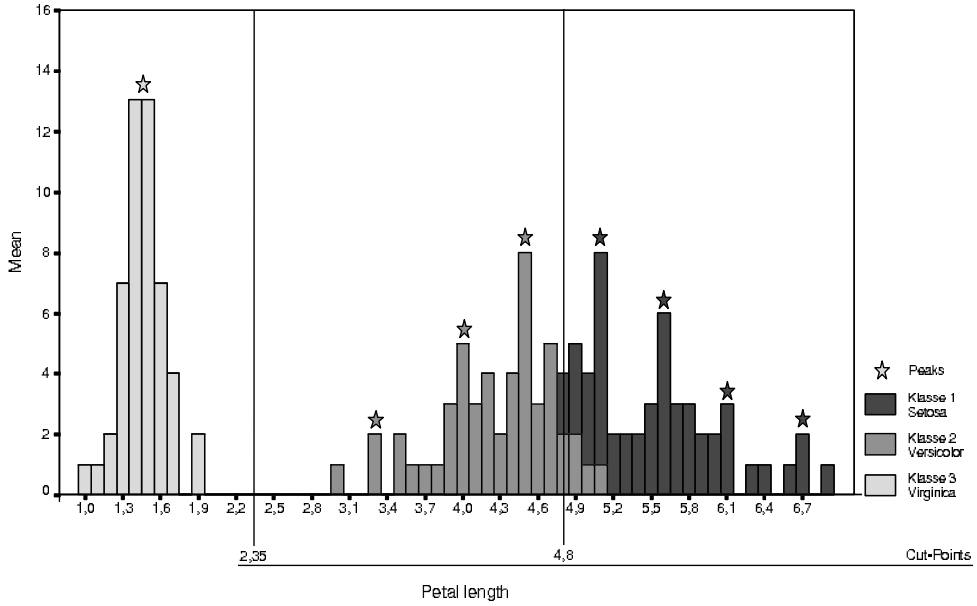


Figure 3.12: Histogram of cut points for the Petal-Length attribute

into 3 intervals (k=2): interval_1 from 0 to 2.35, interval_2 from 2.36 to 4.8, and interval_3 from 4.9 to 7. We will also consider attribute discretization on categorical attributes. Many at-tribute values of a categorical attribute will lead to a partition of the sample set into many small subsample sets. This again will result into a quick stop of the tree building process. To avoid this problem, it might be wise to combine attribute val-ues into a more abstract attribute value. We will call this process attribute aggrega-tion. It is also possible to allow the user to combine attribute interactively during the tree building process. We call this process manual abstraction of attribute values.

3.2.3 Pruning

If the tree is allowed to grow up to its maximum size it is likely that it becomes overfitted to the training data. Noise in the attribute values and class information will amplify this problem. The tree building process will produce subtrees that fit to noise. This unwarranted complexity causes an increased error rate when classi-fying unseen cases. This problem can be avoided by pruning the tree. Pruning means replacing subtrees by leaves based on some statistical criterion. This idea is illustrated in (see Figure 3.13) on the IRIS data set. The unpruned tree is a large and bushy tree with an estimated error rate of 6.67%. Subtrees up to the second level of the tree get replaced by leaves. The resulting pruned tree is smaller and the error rate becomes 4.67% calculated with cross validation (see Figure 3.14). Pruning methods can be categorized either in pre- or post-pruning methods. In pre-pruning, the tree growing process is stopped according to a stopping criteria before the tree reaches its maximal size. In contrast to that, in post-pruning, the tree is first developed to its maximum size and afterwards, pruned back according to a pruning procedure.

An overview about all methods can be found in [133].

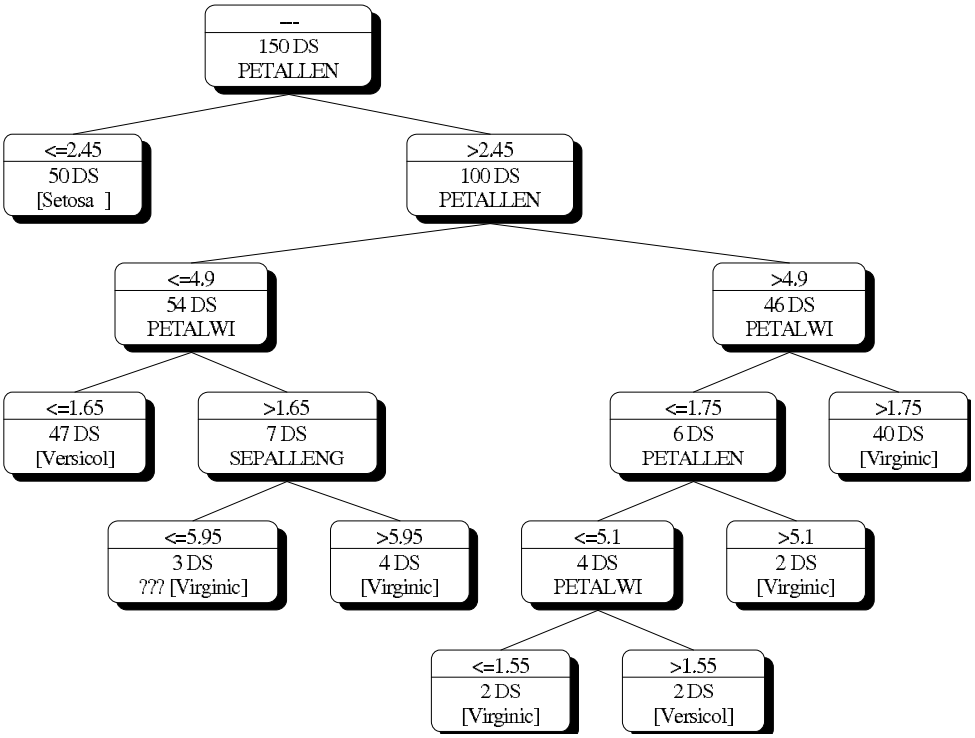


Figure 3.13: Unpruned decision tree for the IRIS dataset.

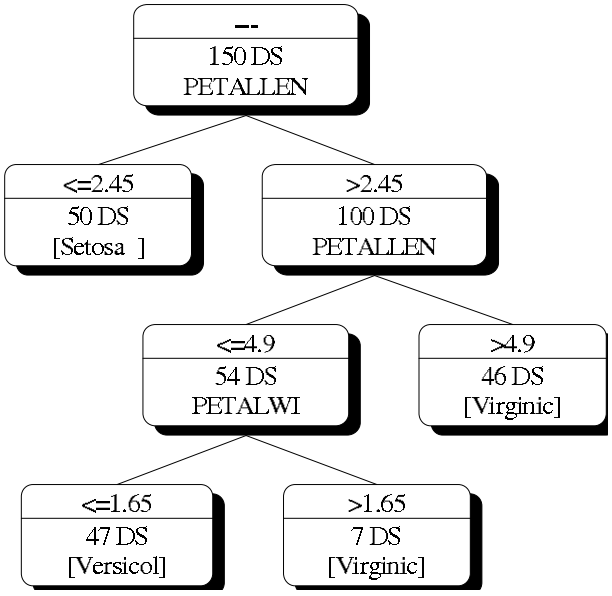


Figure 3.14: Pruned tree for the IRIS dataset based on Minimal Error Pruning.

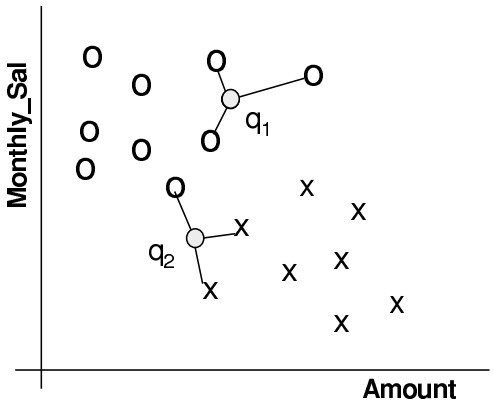


Figure 3.15: A simple example of 3-Nearest Neighbour Classification

3.3 Nearest Neighbour Classification

The intuition underlying Nearest Neighbour Classification is quite straightforward, examples are classified based on the class of their nearest neighbours. It is often useful to take more than one neighbour into account so the technique is more commonly referred to as k-Nearest Neighbour (k-NN) Classification where k nearest neighbours are used in determining the class. Since the training examples are needed at run-time, i.e. they need to be in memory at run-time, it is sometimes also called Memory-Based Classification. Because induction is delayed to run time, it is considered a *Lazy Learning* technique. Because classification is based directly on the training examples it is also called Example-Based Classification or Case-Based Classification.

The basic idea is as shown in Figure 3.15) which depicts a 3-Nearest Neighbour Classifier on a two-class problem in a two-dimensional feature space. In this example the decision for q_1 is straightforward - all three of it's nearest neighbours are of class O so it is classified as an O. The situation for q_2 is a bit more complicated at it has two neighbours of class X and one of class O. This can be resolved by simple majority voting or by distance weighted voting (see below).

So $k-$NN classification has two stages; the first is the determination of the nearest neighbours and the second is the determination of the class using those neighbours.

Let us assume that we have a training dataset D made up of $(\mathbf{x}_i)_{i\in[1,|D|]}$ training samples. The examples are described by a set of features F and numeric features have been normalised to the range [0,1]. Each training example is labelled with a class label $y_j \in Y$. Our objective is to classify an unknown example $\mathbf{q}$. For each $\mathbf{x}_i \in D$:

$$Sim(\mathbf{q},\mathbf{x}_i) = \sum_{f\in F} w_f \delta(\mathbf{q}_f,\mathbf{x}_{if}) \tag{3.12}$$

There are a large range of possibilities for this similarity metric; a basic version for continuous and discrete attributes would be:

$$\delta(\mathbf{q}_f,\mathbf{x}_{if}) = \begin{cases} 1 & f \text{ discrete and } \mathbf{q}_f = \mathbf{x}_{if} \\ 0 & f \text{ discrete and } \mathbf{q}_f \neq \mathbf{x}_{if} \\ 1-|\mathbf{q}_f - \mathbf{x}_{if}| & f \text{ continuous} \end{cases} \tag{3.13}$$

The k nearest neighbours are selected based on this similarity metric. Then there are a variety of ways in which the k nearest neighbours can be used to determine the class of $\mathbf{q}$.

A fairly general technique is distance weighted voting where the neighbours get to vote on the class of the query case with votes weighted by their similarity to the query.

$$Vote(y_j) = \sum_{c=1}^{k} w_c 1(y_j, y_c) \tag{3.14}$$

Where the weight $w_c = Sim(\mathbf{q}, \mathbf{x}_c)$, i.e. the similarity calculated above. Then the query case is assigned the class with the highest score. A simpler version still would be to assign all weights to 1 so that all neighbours get the same vote.

3.3.1 *k*-NN: Advantages and Disadvantages

k-NN is very simple to understand and easy to implement. So it should be considered in seeking a solution to any classification problem. Some advantages of *k*-NN are as follows (many of these derive from it's simplicity and interpretability):

- Because the process is transparent, it is easy to implement and debug.
- In situations where an explanation of the output of the classifier is useful, *k*-NN can be very effective if an analysis of the neighbours is useful as explanation.
- There are some noise reduction techniques that work only for *k*-NN that can be effective in improving the accuracy of the classifier [243].
- Case-Retrieval Nets [144] are an elaboration of the Memory-Based Classifier idea that can greatly improve run-time performance on large case-bases.

These advantages of *k*-NN, particularly those that derive from it's interpretability, should not be underestimated. On the other hand, some significant disadvantages are as follows:

- Because all the work is done at run-time, *k*-NN can have poor run-time performance is the training set is large.
- *k*-NN is very sensitive to irrelevant or redundant features because all features contribute to the similarity (see Eq. 3.12)and thus to the classification. This can be ameliorated by careful feature selection or feature weighting.
- On very difficult classification tasks,*k*-NN may be outperformed by more *exotic* techniques such as Support Vector Machines or Neural Networks.

3.3.2 CBR in Image Processing

Image processing is a challenging field. The unique data (images) and the necessary computation techniques require extraordinary case-representations, similarity measures and CBR strategies to be utilised.

Perner [194] proposes a system that uses CBR to optimize image segmentation at the low level unit according to changing image acquisition conditions and image quality. The intermediate-level unit extracts the case representation used by the high-level unit employed to dynamically adapt image interpretation. The system works on different case representations such as a graph-based representation for the cases of the high-level image description and the raw image matrix for the low-level image representation. Therefore the system uses different CBR strategies for the reasoning and learning part one is based on structural similarity and the other is based on the digital image distance.

Grimnes and Aamodt [86] present a system that integrates CBR into a task-oriented model-based system for the interpretation of abdominal CT images. A case-based reasoner working on a segment case-base contains the individual image segments. These cases with labels are considered indexes for another case-based reasoner working on an organ interpretation case-base. Their system is based on a propose-critique-modify learning cycle.

In [111] a system for ultra-sonic B-scans is presented that are one-dimensional signals. He presents a tree-based retrieval strategy. Micarelli *et al.* [164] applies CBR to scene recognition. They calculated image properties from images and stored them into a case base. They used the Wavelet transform because it is scale-independent, but this limits their similarity measure to consider only object rotation. The application of CBR to image segmentation for CT images from the brain is described in [195]. Different learning strategies in a hierarchy of structural cases are presented in [194] and [198]. Learning case representation and improving the system performance by controlling the similarity measure is described in [203]. The application of CBR image interpretation to health monitoring and biotechnology is described in [201].

A new challenging application field are geographical information systems. This kind of application requires special spatial problem solving techniques and spatial similarity measures. The first application of CBR to geographic information systems was reported by Holt and Benwell [103]. Their approach consists of combining case-based reasoning with geographical information systems to form a hybrid system to solve spatial problems in soil classification. Carswell *et al.* described in their paper [34] a spatial similarity measure for comparing the location objects in different images.

A completely different application of CBR to image processing has been described by Ficet-Cauchard *et al.* [68]. They apply CBR for the development of the image processing steps of formerly unknown image processing problems by using past experiences and plan adaption.

Case-based object recognition for fungi identification and the developed similarity measures are described in [200]. A system for case acquisition of visual forms and case mining has been developed in [202]. Similarity-based Learning of case description of visual objects is described in [203]. Finally, in Perner [196] is built a bridge between the work in CBR and the one in dissimilarity classification which became recently important in pattern recognition.

3.3.2.1 Case Representations for Images

Usually the main types of information concerned with image interpretation are image-related and non-image-related information. Image-related information can be the 1D, 2D, or 3D images of the desired application, while non-image-related information can include information about image acquisition (e.g., the type and parameters of the sensor, information about the objects, or the illumination of the scene). The type of application determines what type of information

should be considered for image interpretation. For medical CT image segmentation [195], patient-specific parameters such as age, sex, slice thickness, and number of slices were used. In [111] is considered the type of sensor for a railway inspection application and his system used it to control the type of case base that the system used during reasoning. How the 2D or 3D image matrix is represented depends on the application and the developer's point of view. An image may be described by the pixel matrix itself or by parts of this matrix (a pixel representation). It may be described by the objects contained in the image and their features (a feature-based representation). Furthermore, it can be described by a more complex model of the image scene comprising objects and their features as well as the object's spatial relationships (an attributed graph representation or semantic network).

Processing the image through multiple components and describing it by higher-level representations can reduce the number unnecessary details in its representation. This allows more noise tolerance and may speed up the retrieval process but may require additional modeling of the image content, which is difficult and time-consuming. Also, it requires processing steps that are often computationally intensive. Thus, the necessary abstraction level of the image information should be carefully chosen.

The abstraction problem is solved in [111] by using a four-level case hierarchy and different case bases for different sensor types. Stored at the lowest level of the hierarchy are the objects described by features such as their location, orientation, and type (line, parabola, or noise) parameters. The next level consists of objects of the same channel within the same subcluster. In the following level the subcluster is stored and the highest level stores the entire image scene. This representation allows cases to be matched on different granularity levels. Because the entire scene may have noise distortions and imprecise measurements, the influence of noise can be reduced by retrieving cases on these different levels.

Grimnes and Aamodt [86] developed a model-based system for the interpretation of abdominal CT images. The image's content was represented by a semantic network where concepts can be a general concept, a case, or a heuristic rule. Poorly understood parts of the model are expressed by cases and can be revised during system usage by the learning component. The combination of a partial well-understood model with cases helps to overcome the usual burden of modeling. The learning component is based on failure-driven learning and case integration. Non-image information is also stored such as sex, age, earlier diagnosis, and social condition.

In both of these systems, CBR is used only for the high-level component. We have studied different approaches for the different processing stages of an image interpretation system. For image segmentation [111], we studied a pixel-based approach and also a feature-based approach that described an image's statistical properties. Our results show that the pixel-based approach can yield better image segmentation [195]. For the high-level approach in an ultra sonic image interpretation system, a graph representation [194] was used.

Representing images at multiple levels of abstraction presents some technical challenges. When representing an image with a high-level abstraction rather than the image matrix itself, some information will be lost. Abstraction requires deciding which details of an image are necessary. If only some objects are seen at one time, then we might think that one detail is not of interest since our decision is based on a limited number of objects. This can cause problems. Therefore, storing the images themselves is always preferable but requires high storage capacity. Also, the different representations at each abstraction level require different similarity measures.

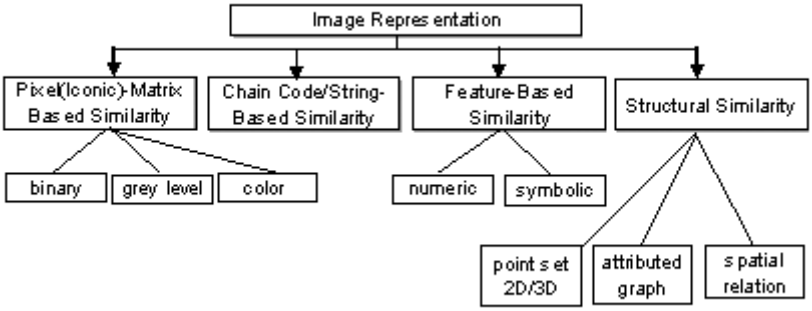


Figure 3.16: Image Representation and Similarity Measures

3.3.2.2 Similarity Measures for Image Interpretation

Images can be rotated, translated, different in scale, or may have different contrast and energy yet still considered to be similar. In contrast, two images may be dissimilar because the object in one image is rotated by 180 degrees. The concept of *invariance* in image interpretation is closely related to that of similarity. A good similarity measure should take this into consideration.

Classical similarity measures do not consider invariance. Usually, the images or the features have to be pre-processed in order to be adapted to the scale, orientation, or shift. This process is an additional and expensive processing step that needs some a priori information, which is not always given. Matched, linear, Fourier, and Wavelet filters are especially useful for invariance under translation and rotation [164]. There has been a lot of work done to develop such filters for image interpretation. The best way to achieve scale invariance from an image is by means of invariant moments, which can also be invariant under rotation and other distortions. Some additional invariance can be obtained by normalization (to reduce the influence of energy).

Depending on the image representation (see Fig. 3.16) we can divide similarity measures into:

- pixel (Iconic)-matrix similarity measures;
- similarity measures for comparing strings;
- feature-based similarity measures (numeric, symbolic, or mixed type); and,
- structural similarity measures.

Because a CBR image interpretation system must also account for non-image information (e.g., about the environment or the objects), similarity measures are needed that can combine non-image with image information. In [195] is described a first approach for doing this.
Systematic studies on image similarity have been conducted by Zamperoni and Starovoitov [275]. They studied how pixel-matrix similarity measures behave under different real-world influences such as translation, noise (spikes, salt and pepper noise), and different contrast. Image feature-based similarity measures have been studied from a broader perspective by Santini and

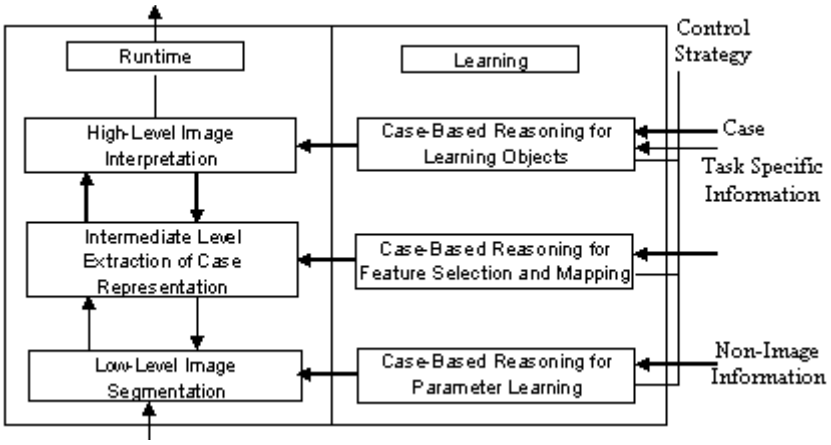


Figure 3.17: Architecture of a Case-Based Image Interpretation System

Jain [232]. To our knowledge, these are the only comprehensive studies on image similarity. Otherwise, every new conference on pattern recognition contains proposals for new similarity measures for specific purposes and different kinds of image representation [143], [192], [167], [253], [163]. While there was some simultaneous research on image similarity in the CBR community (e.g., [259]), this work has also not achieved new insight. In our view, images are a special type of information source that require special similarity measures, and these measures require more rigorous analysis.

3.3.2.3 The Architecture of the Case-Based Image Interpretation System

The architecture (Fig. 3.17) that uses CBR on all levels of an image interpretation system is proposed in [194]. The system subdivides into a run-time part and a maintenance and learning part. During run-time, the system uses CBR strategies to reason over images while the maintenance and learning part attempt to improve system performance off-line.

3.3.2.4 Image Segmentation

Most CBR image interpretation systems (e.g., [86], [68]) select among different image processing chains but they do not control the algorithm itself. This in accordance with most knowledge-based image interpretation systems described in the computer vision literature, which select a processing chain that best fits the current image analysis problem. This approach requires a large enough library of image processing procedures and special image processing knowledge.

However, modern segmentation techniques contain numerous control parameters, which can be adjusted to obtain optimal performance. Parameter selection should be done using a sufficiently large test data set that represents the entire domain well enough to support a general segmentation model. Obtaining a suitable test set is often impossible, which means that the segmentation model does not fit the data well and must be adjusted to new data. Also, a general model does not guarantee the best segmentation for each image, but instead it guarantees an average best fit over the entire set of images. Finally, differing image quality (e.g., caused

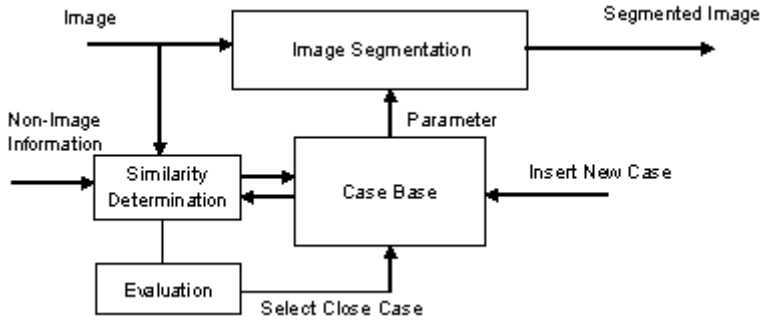


Figure 3.18: Case-Based Image Segmentation Component

by variations in environmental conditions, image devices) requires adapting the segmentation process accordingly. This necessitates equipping the segmentation component with learning capabilities, which can incrementally acquire segmentation model knowledge.

The system uses a case-based approach for parameter learning, in which formerly processed cases contain their original images, their non-image information (e.g., image acquisition parameters, object characteristics), and their image segmentation parameters. Finding the best segmentation for the current image is done by retrieving similar cases from the case base. Similarity is computed using non-image and image information. The evaluation component will use the most similar case for further processing. If two or more cases have the same highest similarity score then the first of these cases is used. The image segmentation parameter associated with the selected case will then be given to the image segmentation component, which will segment the current image (see Fig. 3.18). Images with similar image characteristics are assumed to yield similar good segmentation results when the same segmentation parameters were applied to these images. Superior performance for this approach has been demonstrated for CT image segmentation [195]. This approach is sufficiently flexible to be used for other applications and will therefore be used for Hep-2 cell image analysis [203].

3.3.2.5 Feature Selection

Feature selection is concerned with learning the most important (symbolic) features, while feature extraction is responsible for locating those features in the image and finding their values. From the preprocessed, segmented, and labeled 1-D, 2-D, or 3-D image matrix we can extract low-level or primitive image features that are corners, extended edges, textured regions, ribbons, the 2 1/2-D sketch, and semantic clusters of edges and regions. The number of primitive features that can be extracted from the image content is limited (e.g., color, gray level, spatial relations, motion). Understanding the image's content requires mapping those primitives to the desired symbolic features. In current approaches to image interpretation, performance degrades when new objects are encountered that may require the extraction of "shape primitives" not known to the system. To overcome the bottleneck of predetermined and static object features, automatic acquisition of new features using a learning approach is necessary, particularly for flexible image interpretation processes.

Therefore, one may introduce for the system a library of feature extractors that can calculate

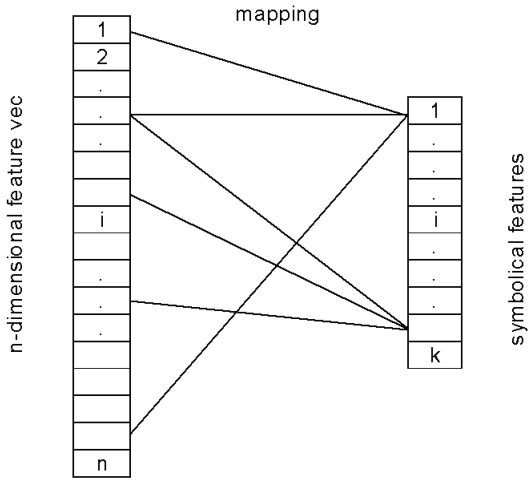


Figure 3.19: Low-Level Feature Selection and Mapping to Symbolic Features

all possible features. In the next step, the system selects from these features the necessary features describing the desired symbolic feature.

3.3.2.6 Signal-to-Symbol Mapping and Feature Selection

It is seldom the case that one low-level feature describes the right meaning of one quality of an object. Often a combination of a few low-level features is necessary to express a symbolic feature like *fine speckled*, which is a combination of low-level features such as number of small objects, object sizes, and their gray-level. In these situations, a mapping of (n) low-level features to the symbolic feature is needed. This problem is concerned with the selection of the right features (feature selection), their parameters, and the creation of a mapping function (classification function).

The problem here is to select this subset of features from a large/complex feature set that represent best the symbolic feature by means of classification accuracy or intra/inter class distance, see Fig. 3.19. To solve this problem, we use an induced decision tree [205]. This approach acts as feature filter for the image interpretation process. Once a new feature is discovered the low-level features are calculated from the image and labeled by the symbolic feature. The prototypes of the other features are taken and applied together with data from the new feature to the induction algorithm. The resulting set of rules is used as a feature selector.

3.3.2.7 High-Level Unit

The case representation of an image's high-level information can differ among images. This ranges among semantic networks [86], graphs [194], and decision trees [203]. Image interpretation problems always have some hidden taxonomy that, if discovered, can be used to help model the problem. An ultrasonic image showing a defect type *crack* might show a crack of a specific subclass such as *crack_under_pressure_x*. To classify this type of crack as a specific

subtype might prevent the class *crack* from having large variations which can help to improve the classification results.

To discover these concepts we have found decision tree induction [205] and incremental conceptual clustering [197], [193] very suitable. Based on the available cases we used C4.5 [193] to induce a tree for indexing the case base. This approach differs from the one presented in [111], where also a tree for case indexing was induced, in that we will incrementally update the tree structure based on newly discovered cases. Leaves in the tree where no class overlap occurs will remain as terminal leaves, while a leaf with class overlap will be pruned back until a predefined number of samples remain in the group covered by this leaf.

The query case may be clustered through the tree until it reaches a leaf node. If the leaf node is labeled with its class, then that class is assigned to the query. If it is not a final node then similarity will be calculated between all cases belonging to this node. We do not divide these cases into clusters but instead incrementally update the index structure when entering a new case.

3.3.2.8 Maintenance and Learning

An important focus of recent CBR research is on how to develop strategies for obtaining compact, competent case-bases, as a way to improve the performance of CBR systems [50]. Although maintenance approaches have not yet been extensively studied for image interpretation systems, they will play an important. Grimnes and Aamodt [86] mention that maintaining the case base in ImageCreek is complex, and that knowledge base maintenance methods are crucial for making the architecture scale. The main problem is handling the different types of knowledge. Jarmulak [111] takes into account case addition, splits the clusters into groups of fixed size, and represents them using a prototype to speed up the matching process. Perner [194], [195] takes into account case addition, learning of case classes and prototypes, and higher order constructs. We focus here on topics that, until now, have only been addressed as more specific problems.

3.3.2.9 Learning in a CBR System

CBR management is closely related to learning. It aims to improve the performance of the system. Let X be a set of cases collected in a case base CB. The relation between each case in the case base can be expressed by the similarity value sim. The case base can be partitioned into n case classes:

$$C : CB = \bigcup_{i=1}^{n} C_i \tag{3.15}$$

such that the intra-case class similarity is high and the inter-case class similarity is low.

The set of cases in each class C can be represented by a representative who generally describes the cluster. This representative can be the prototype, the mediod, or an *a priori* selected case. Whereas the prototype implies that the representative is the mean of the cluster which can easily be calculated from numerical data. The mediod is the case whose sum of all distances to all other cases in a cluster is minimal. The relation between the different case classes

C can be expressed by higher-order constructs expressed e.g. as super classes which give us a hierarchical structure over the case base.

There are different learning strategies that can take place in a CBR system:

- Learning takes place when a new case x has to be stored into the case base such that: $CB_{n+1} = CB_n \cup \{x\}$. That means that the case base is incrementally updated according to the new case.
- It may incrementally learn the case classes and/or the prototypes representing the class [198].
- The relationship between the different cases or case classes may be updated according the new case classes [194].
- The system may learn the similarity measure [203], [201].

3.3.2.10 Learning New Cases and Forgetting Old Cases

Learning new cases means just adding cases into the case base upon some notification. Closely related to case adding is case deletion or forgetting cases which have shown low utility. This should control the size of the case base. There are approaches that keep the size of the case base constant and delete cases that have not shown good utility within a fixed time window [134]. The failure rate is used as utility criterion. Given a period of observation of N cases, if the CBR component exhibits M failures in such a period, we define the failure rate as $f_r = M/N$. Other approaches try to estimate the "coverage" of each case in memory and by using this estimate to guide the case memory revision process [245].

The adaptability to the dynamic of the changing environment that requires storing new cases in spite of the case base limit is addressed in [246]. Based on intra class similarity it is decided whether a case is to be removed from or to be stored in a cluster. Case deletion in a pre-determined time window based on failure results [134] might not be appropriate for image interpretation because a failure might mean that, instead of the retrieved case being erroneous, there is some relevant knowledge that we could not describe using features. Also, cases that occur infrequently (*i.e.*, that have not been used recently) should be recognized by the system.

The causes for case deletion or addition might differ from other CBR applications - since images may be distorted and very noisy it might not be useful to store distorted representations. Determining which representation is distorted is sometimes not easy even if you have seen only a few images, and it is usually necessary to have domain knowledge that must also be built up over time.

Imprecise or noisy measurements can be caused by some defects of illumination, the image acquisition device, or the object itself. If the image analysis cannot adapt to these measurements, or the reasoning process cannot handle it, then this might cause failure results. However, if this is a systematic event then it might be worthwhile to store the recent case in the case base.

The last fact comes from the real world environment. It is not possible to determine all real world influences *a priori*. Thus, developers prefer to incorporate cases into a case base instead of forgetting them. Although case bases can grow very large, instead of forgetting cases, we would rather subdivide the case base into frequently *vs.* rarely used cases. This

requires addressing the issue of how should the addition of cases into one of these two case bases be controlled, as well as their respective reasoning processes.

3.3.2.11 Learning of Prototypes

Learning of prototypes has been described in [198] for a flat organization of a case base and for a hierarchical representation of a case base in [194]. The prototype or the representative of a case class is the most general representation of a case class. A class of cases is a set of cases sharing similar properties. The set of cases does not exceed a boundary for the intra class dissimilarity. Cases that are on the boundary of this hyper-ball have a maximal dissimilarity value. A prototype can be selected *a priori* by the domain user. This approach is preferable when the domain expert knows for sure the properties of the prototype. The prototype can be calculated by averaging over all cases in a case class or the median of the cases is chosen. If only a few cases are available in a class and subsequently new cases are stored in the class, then it is preferable to incrementally update the prototype according to the new cases.

3.3.2.12 Learning of Higher-Order Constructs

The ordering of the different case classes gives an understanding of how these case classes are related to each other. For two case classes which are connected by an edge similarity relation holds. Case classes that are located at a higher position in the hierarchy apply to a wider range of problems than those located near the leaves of the hierarchy. By learning how these case classes are related to each other, higher-order constructs are learnt [198].

3.3.2.13 Learning of Similarity

By introducing feature weights we can put special emphasis on some features for the similarity calculation. It is possible to introduce local and global feature weights. A feature weight for a specific attribute is called *local feature weight*. A feature weight that averages over all local feature weights for a case is called *global feature weight*. This can improve the accuracy of the CBR system. By updating these feature weights we can learn similarity [267].

3.3.2.14 Case Acquisition and Case Mining

A CBR system for image classification needs to have some particular features with respect to images. These features result from:

- special requirements of visual knowledge acquisition (image-language problem) and
- the need to transform an image's numerical data into a symbolic description.

The main problem with images and their translation into a language is that the knowledge about an image is usually tacit. To make this knowledge explicit is often hard. Sometimes the meaning of a word does not correspond to the correct meaning of the image. Therefore, it is necessary to support the operator in an efficient way.

Most case-based image interpretation systems do not pay attention to this problem. The only functionality these systems provide is visualization of the image or the processed image.

Usually, new case knowledge is obtained via manual acquisition with the expert. This is a time-consuming and sometimes boring process for both the system developer and the expert.

A CBR system for image interpretation should have a special case acquisition tool, such as the one detailed in [191]. By using a questioning strategy and evaluating the answers given by the expert, the expert or operator is forced to specify the right knowledge for image interpretation. The questioning strategy is designed to force an expert to explain what distinguishes one object from another and to specify the right property for the object. Recently, this problem has received attention for e-commerce applications. Automatic questioning strategies are important for acquiring customer requirements in e-commerce applications [229] because the customer acts alone on the net. A special case acquisition tool for image segmentation was described in [195]. With the help of this tool the user can control the parameters of the image segmentation algorithm. Simultaneously, he can view the segmented image and, if he is satisfied with the segmentation quality, he can store the parameters of the image segmentation algorithm together with the case description in the case base. Acquiring cases by tracing objects in images and learning more compact cases from these examples by case mining is described in [202].

3.3.2.15 Competence of Case Bases

An important problem in image interpretation concerns system competence. We follow the definition in [245] and define the competence of a system as the range of target problems the system can solve. It is often not clear to the computer vision community what problems the desired algorithm can solve. So we have to find a way to describe the competence of a system. This differs from what is usually understood about this problem in CBR. Competence is described based on statistical properties such as case-base size, density and distribution, or group coverage and group density. But what if some groups overlap? Smyth and McKenna [245] argue that these groups have shared competence and can be linked together in some way.

However, we can also view it as having a poor description of the target problem. Based on this description we may retrieve a similar case but its solution application to the query image may be low in quality. By investigating the failure we may learn that we did not consider a property of the environment or maybe we could not specify it because it was not contained in the description of the target problem. Therefore, the system performance decreases. The measures described in [50] and [245] only view competence based on the coverage of the problem space. How do we know that cases in group 1 and group 2 belong to the same target problem group? Proximity in problem space does not imply that they belong to the same problem group; misclassifications can occur because the patterns overlap. We argue that system competence must also account for the misclassification of the target problem based on the problem description.

3.3.2.16 Control Strategies and Monitoring System Performance

An important issue in maintaining an image interpretation system involves the controlling and monitoring of system performance. The system is a complex system comprising different processing components (e.g., image analysis, feature extraction and high-level image interpretation). The quality of the results of one component strongly depends on the quality of a preceding component. Several possible strategies exist for improving system performance.

Control without Feedback (Local Optimization)

Table 3.4: Logical Scheme of Performance Control

Segmentation (S)	Feature Extraction (FE)	Interpretation (I)	Action
Good	Good	Good	No Action
Good	Good	Poor	Optimize I
Good	Poor	Good	Impossible
Good	Poor	Poor	Optimize FE and examine effects on I
Poor	Good	Good	Impossible
Poor	Good	Poor	Impossible
Poor	Poor	Good	Impossible
Poor	Poor	Poor	Optimize S, then re-examine the performance of the other components

The simplest approach is to adjust the performance of each component without considering the others. Each component - segmentation, feature extraction and selection, and interpretation - acts alone. No interaction between them is allowed. Image segmentation performance may be determined by subjective evaluation of the segmentation result as done by an expert, by calculating the similarity between the original and segmented images, by interclass distances for feature extraction, or by classification error. This strategy has the advantage that the control of the system is simple. Nevertheless, it cannot optimize system performance because only local optimums can be achieved for each single component.

Control with Feedback (Global Optimization)

If after local optimization the performance of a component could not be improved or is not satisfactory, the control algorithm will lead the learning process to the preceding processing component in an attempt to further improve its performance. This process stops if the first processing component is reached and if no improvement could be established after local optimization.

The logical scheme in Table 1 shows us how control is guided. If the performance of all components is good, no action has to be taken. If the interpretation component's performance is poor, then its performance needs to be optimized. We assume that it is impossible for a preceding component to perform poorly while its successor components perform well.

3.3.2.17 Conclusions

We have given an overview about case-based reasoning for image processing and image understanding. A more detailed description of this topic can be found in Perner [199].

3.4 Bayesian Techniques

Bayesian analysis [15, 16, 31, 43, 52, 55, 69, 75, 152, 141, 170, 176, 208, 220, 266] uses explicit probability models to incorporate general as well as scene-specific prior knowledge into the data

(hypothesis, parameters) processing and provides a consistent framework for solving many of processing tasks. It consists of four successive stages ([17, 43, 31]):

- The underlying (true, hidden) data prior probability $p(X)$ distribution construction, which should capture general and scene-specific prior knowledge about X. It is seldom possible to model the global features, so usually this prior describes the local characteristics of X.
- The joint probability density (likelihood function) $p(Y|X)$ specification, usually by standard physical considerations. This likelihood function which describes an interaction between hidden and observed data may depend on unknown parameters that need to be estimated from the data.
- The prior probability and the likelihood are combined applying Bayesian theorem to form the posterior density $p(X|Y) \propto p(Y|X)\,p(X)$ (the normalization constant $p(Y) = \int p(Y|X)\,p(X)\,dX \neq 0$).
- An inference about X is based on $p(X|Y)$ given Y. If we ignore the existence of unknown parameter values, then the usual method is to find the maximum posterior (MAP) estimate of X and then to extract from it the features of primary interest.

A major strength of the Bayesian approach is that the resulting estimates are not point estimates but the complete posterior probabilities acquired either analytically, using numerical integration or more often using sampling-based methods (variety of Markov chain Monte Carlo (MCMC) methods). Provided with large sample from a density the mathematical form of that density can be approximated using curve estimation (kernel density) methods. From these posterior probabilities it is possible not only select some point estimates if required but also interval estimates, standard deviation, probabilities of error classification, etc. Bayesian inference is identical for different solved problems, what stands in contrast to other inference approaches that involve special techniques and principles for different tasks to be solved. On the other hand its criticism stems from the necessity to specify sometimes rather artificially the prior probability $p(X)$. If such a prior probability about the estimated quantity is not available but the data distribution itself is known, the maximum likelihood criterion can be used. In the opposite case, when the prior information is the only available the maximum entropy criterion emulates nature preference for higher entropy solutions. Bayesian analysis often requires numerical integration however recent computer intensive sampling methods of probability density function (pdf) estimation enables their wide use in many different application areas.

3.4.1 Several Sets of Data

For several sets of measurements generated independently of each other, the resulting posterior probability density function is identical whether it is constructed sequentially or from n combined samples, i.e.,

$$p(X|Y_1,\ldots,Y_n) \propto p(X)\prod_{i=1}^{n} p(Y_i|X) \quad . \tag{3.16}$$

3.4.2 Marginal Posterior

If X is partioned into subsets $X = (X_a, X_b)$, the corresponding marginal posterior can be obtained as follows:

$$p(X_a \,|\, Y) = \int_{\mathcal{X}_b} p(X \,|\, Y) dX_b = \int_{\mathcal{X}_b} p(X_a \,|\, X_b, Y)\, p(X_b \,|\, Y) dX_b \quad . \tag{3.17}$$

Marginal pdf for the observation can be obtained from the averaged conditional pdf (cpdf) as follows:

$$p(Y) = \int_{\mathcal{X}} p(Y \,|\, X)\, p(X) dX \quad . \tag{3.18}$$

3.4.3 Bayesian Estimation

Bayesian estimation of a set of model parameters X (assuming known cpdf $p(X|Y)$ and prior pdf $p(X)$) is based on the Bayes theorem

$$p(X \,|\, Y) = \frac{p(Y \,|\, X)\, p(X)}{\int p(Y \,|\, X)\, p(X)\, dX} \quad . \tag{3.19}$$

If the measurements are conditionally independent the general estimation formula (3.19) can be written using conditional marginals

$$p(X \,|\, Y^{(r)}) = \frac{\prod_{k=1}^{r} p(Y_k \,|\, Y^{(k-1)}, X)\, p(X)}{\int \prod_{k=1}^{r} p(Y_k \,|\, Y^{(k-1)}, X)\, p(X)\, dX} \quad . \tag{3.20}$$

Alternatively the estimation formula can be reshaped to the recursive form:

$$\begin{aligned} p(X \,|\, Y^{(r)}) &= \frac{p(Y_r \,|\, Y^{(r-1)}, X)\, p(X \,|\, Y^{(r-1)})}{\int p(Y_r \,|\, Y^{(r-1)}, X)\, p(X \,|\, Y^{(r-1)})\, dX} \\ &= \frac{p(Y_r \,|\, Y^{(r-1)}, X)}{p(Y_r \,|\, Y^{(r-1)})}\, p(X \,|\, Y^{(r-1)}) \quad , \end{aligned} \tag{3.21}$$

where $p(Y_r \,|\, Y^{(r-1)})$ is the one-step-ahead prediction for the next random variable using old parameter estimation $p(X \,|\, Y^{(r-1)})$.
Point estimators characterizing the posterior pdf can be based on different pdf measures such as dispersion, skewness, median, etc., or values that minimize the mathematical expectation of some loss function or the average risk. The optimal estimate for the quadratic loss function is the conditional expectation $E\{X \,|\, Y\}$, for the absolute error loss function it is the median. If the probability $p(X \in \mathcal{R} \,|\, Y) = \alpha$ is fixed to some α value, it is possible to find Bayesian region (not necessarily unique) $\mathcal{R}$.

3.4.4 Bayesian Decision Criteria

The maximal a posteriori probability (MAP) decision minimizes the probability that any data element in the lattice will be misclassified:

$$\omega^* = \arg \max_{\omega \in \{\hat{\Omega}\}^n} p(\omega|Y) \tag{3.22}$$

for the loss function

$$\mathcal{L}(\omega, \omega_{opt}) = 1 - \prod_{i \in I} \delta(\omega_i - \omega_{i,opt})$$

where ω is an interpretation index or an unobservable data element and I is a data index set. In the case of a multiresolution MAP the loss function is

$$\mathcal{L}(\omega, \omega_{opt}) = 1 - \prod_{\forall k} \prod_{i \in I} \delta(\omega_i^{(k)} - \omega_{i,opt}^{(k)}) \quad .$$

The MAP estimate can be too conservative because the loss function is maximal whenever any data element is incorrectly classified. The Bayes risk

$$R = \sum_{Y} \sum_{\omega} \mathcal{L}(\omega, \omega_{opt})\, p(\omega, Y)$$

is minimal if and only if all data elements are correctly classified, i.e., the MAP estimator is the Bayes estimator for the zero-one loss function. Computing the MAP estimate requires minimization of a discrete functional with many local minima. Exact minimization is intractable so some approximation like the simulated annealing has to be used.

The compound maximal marginal a posteriori probability MMAP (MPM) criterion is

$$\omega^* = \{\omega_i^* : i \in I;\ \omega_i^* = \arg \max_{\omega_i \in \hat{\Omega}} p(\omega_i|Y)\} \tag{3.23}$$

for the compound loss function

$$\mathcal{L}(\omega, \omega_{opt}) = 1 - \frac{1}{n} \sum_{i \in I} \delta(\omega_i - \omega_{i,opt}) \quad .$$

The compound a posteriori mean or a posteriori marginal expectation (AM, AME) is as follows:

$$\omega^* = \sum_{\omega} \omega\, p(\omega\,|\,Y) = \{\omega_i^* : i \in I; \omega_i^* = \sum \omega_i p(\omega_i\,|\,Y)\} \tag{3.24}$$

for the compound square loss function

$$\mathcal{L}(\omega, \omega_{opt}) = \frac{1}{n} \sum_{i \in I} (\omega_i - \omega_{i,opt})^2 \quad .$$

Specific applications of Bayesian techniques depend on the X, Y exposition. For example X are unobservable (unspoiled) data while Y are corresponding measurements in the data restoration

or reconstruction tasks, X can be a hypothesis or a data model in hypothesis testing or optimal model selection applications. X is a class label or class label set in context-free, contextual or discontinuity (edge) detection tasks, etc.

3.4.5 Bayesian Networks

3.4.5.1 Bayesian Network: Basics

Graphical models have been used in different research fields to visually represent interactions between multiple variables and parameters involved in the processes to be modelled. For instance, interactions between pixels in image and video data have been represented using graphs for applications such as image restoration or segmentation [28]. For probabilistic graphical models, the graph and its links between the variables are interpreted as probabilities. Probabilistic Graphical Models or PGM, by its visual representation, help then to infere statistical decision in a system [117, 116]. Similitudes between probabilistic bayesian networks and neural networks based on non-probabilistic graph however exist and have been underlined in [33]. Bayesian networks [97, 33] represent a particular class of probabilistic graphical models that is restricted to Directed and Acyclic Graphs. This specific type of graph is defined hereafter.

Definitions. In a graph, an *edge* links two random variables X_1 and X_2 (cf. fig. 3.20 (a)). A *directed edge* defines the connection in only one direction. In fig. 3.20 (b), the direction of the edge defines the variable X_1 as being *parent* of X_2. In this case, X_2 is seen as a *child* of X_1. When considering a set of variables linked by directed edges showing no cycle (i.e. no feedback cycles appear in the graph cf. example fig. 3.20 (c)), the graph is then called a *directed acyclic graph* (DAG). A more exhaustive set of definitions and properties related to Bayesian network

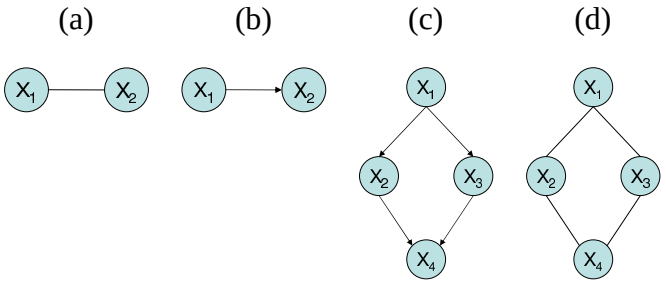


Figure 3.20: Examples of graphs showing the interactions between variables X_i. (a) Edge, (b) directed edge, (c) directed Acyclic graph (DAG) and (d) undirected graph.

can be found in the following book and tutorial [112, 97].

Joint Probability. Let consider a DAG with a set of variables $\{X_i\}$ and their corresponding parents $\{Pa_i\}$. The joint probability of $\{X_i\}$ is then defined as:

$$\mathcal{P}(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} \mathcal{P}(X_i | Pa_i)$$

For example in fig. 3.20 (c), the variable X_1 has no parent:

$$\mathcal{P}(X_1|Pa_1) = \mathcal{P}(X_1)$$

The variables X_2 and X_3 have one parent X_1 then:

$$\mathcal{P}(X_2|Pa_2) = \mathcal{P}(X_2|X_1)$$

$$\mathcal{P}(X_3|Pa_3) = \mathcal{P}(X_3|X_1)$$

The last variable X_4 of the graph has two parents X_2 and X_3 hence:

$$\mathcal{P}(X_4|Pa_4) = \mathcal{P}(X_4|X_2,X_3)$$

The joint probability encoded in the DAG fig. 3.20 (c) can then be written as:

$$\mathcal{P}(X_1,X_2,X_3,X_4) = \mathcal{P}(X_1)\cdot\mathcal{P}(X_2|X_1)\cdot\mathcal{P}(X_3|X_1)\cdot\mathcal{P}(X_4|X_2,X_3)$$

From the joint probability any other density can be inferred using Bayes law and integration. For instance, from the previous example, we can compute:

$$\mathcal{P}(X_1) = \int\int\int \mathcal{P}(X_1,X_2,X_3,X_4)\, dX_2\, dX_3\, dX_4$$

or

$$\mathcal{P}(X_1|X_2,X_3,X_4) = \frac{\mathcal{P}(X_1,X_2,X_3,X_4)}{\int \mathcal{P}(X_1,X_2,X_3,X_4)\, dX_1}$$

However solving the exact inference in an arbitrary Bayesian network can be difficult. As an alternative, approximations of the inference can be computed using simulation methods such as Monte Carlo, or variational methods [117, 116].

3.4.5.2 Using Bayesian Network in Multimedia applications

This section proposes a few references where bayesian networks are used to infer semantics from mutimedia data.

Inferring Semantics from multimedia data. Naphade and Huang [175] have proposed to map low level multimedia features to high-level semantics or *multijects* (for multimedia objects) such as Explosion, Mountain, Beach, Outdoor, etc. High-level semantic concepts such as Outdoor are difficult to infer directly from features. By modelling probabilistic dependences between multijects, high level semantics can however be modelled. For instance, an image can more easily classified as Outdoor if some of its regions are detected as Sky or Beach [175]. Bayesian networks, renamed in this context as *multinets*, are then used to model such dependencies between the multijects (cf. fig. 3.21 (a)). Similarily, in [48], shots of rallies in tennis video broadcasts are inferred from two other events: the detection of the racket hits in the audio data, and the detection of large view over the court in the image data (cf. fig. 3.21 (b)). More complex Bayesian networks are used in [240, 180] to infer semantics from sport videos.

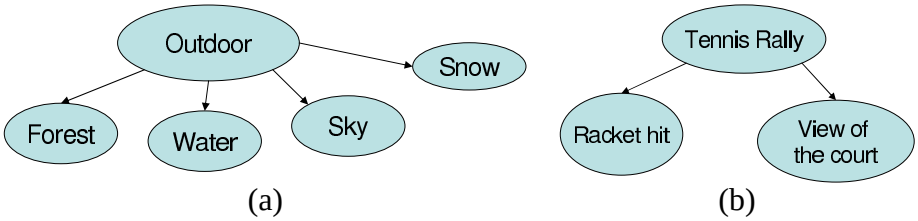


Figure 3.21: (a) High level semantic concept *Outdoor* is detected using Bayesian Network [175]. (b) Tennis rallies are inferred from the detection of racket hits in the audio data, and classification of shots showing views over the court from visual data [48].

Object-based video segmentation. In [264], a bayesian network is used to model the interactions between the displacement field, the intensity-based segmentation field, the spatio-temporal segmentation field and the observed images in the sequence. The proposed approach aims at improving hierarchical approaches where regions lost from an initial intensity segmentation cannot be recovered by a second process using the motion field for the grouping of regions.

Even/Object Classification. In [106], a Bayesian Network is used to model each human motion activities such as Walking Standing Up, Sitting Down and Others. Visual data from a static camera are used to infer each class. In [131], objects such as Pedestrian, Motorbike, Cars or Truck are detected from video data using a bayesian network .

3.4.6 Naïve Bayes

If we return to equation 3.16, we can see that with certain assumptions this can give us a classifier for a class label $y_j \in Y$ for examples described by features $\{a_1, ..., a_n\}$:

$$y_{NB} = \underset{y_j \in Y}{\operatorname{argmax}} P(y_j) \prod_{i=1}^{n} P(a_i|y_j) \tag{3.25}$$

This is the Naïve Bayes or Simple Bayes classifier and the key restrictive assumption is that the attributes a_i are independent of each other, i.e. $P(a_i|a_k) = P(a_i)$. This will almost never be true in practice so it is perhaps surprising that the Naïve Bayes classifier is often very effective on real data. One reason for this is that we don't need to know the precise values for the probabilities $P(y_j)$, we simply require the classifier to be able to *rank* them correctly. Since Naïve Bayes scales well to high dimension data it is used frequently in multimedia applications, particularly in text processing where it has been shown to be quite accurate [147].

In text classification, the conditional probabilities can be estimated by $P(a_i|y_j) = n_{ij}/n_j$ where n_{ij} is the number of times that attributes a_i occurs in those documents with classification y_j and n_j is the number of documents with classification y_j. This provides a good estimate of the probability in many situations but in situations where n_{ij} is very small or even equal to zero this probability will dominate, resulting in an overall zero probability. A solution to this is to incorporate a small-sample correction into all probabilities called the Laplace correction [177]. The corrected probability estimate is $P(a_i|y_j) = (n_{ij} + f)/(n_j + f \times n_{ki})$, where n_{ki} is the

number of values for attribute a_i. Kohavi et al. [124] suggest a value of $f = 1/m$ where m is equal to the number of training documents.

3.5 Feed-forward Neural Networks

Feed-forward Neural Networks (FNNs) represent, by far, the most widely known, analysed and used type of Artificial Neural Network (ANN). Historically, FNNs were introduced as models of biological neuronal networks in which nodes corresponded to neurons and connections between them to synapses [162]. However, further developments have proceeded largely independently of any biological modelling considerations. Actually, FNN are general-purpose, flexible, non-linear, mathematical models successfully used, within a supervised training strategy, in a large variety of application fields.

In the following sub-sections, we introduce the fundamental concepts of FNN, outlining the former architectures and learning rules that have been introduced (*Perceptron* and *Adaptive Linear Networks*), and subsequently focus on the more evolved FNN (*Error Back-Propagation Network* and *Radial Basis Function Networks*) and discussing their application to CBIR.

3.5.1 Basic Concepts

A *Feed-Forward Neural Network* (FFNN) is an adaptive processing system, structured into a series of successive layers of parallel, non-linear units (also called *nodes*, *neurons* or *processing elements*), in which the data flow is strictly feed-forward. There are no connections (the so-called *synapses*) from any unit to other units of the same layer, nor to units of the previous layers, nor to units more than one layer ahead: the output of each node is sent as input only to the next layer in the network structure.

An example of FFNN with three layers is shown in Figure 3.22: the network consists of r input units[3], each of which receives the input vector of n elements and is connected with each of the s hidden units of the middle layer. The output of these units, which have no contact with the external world, is sent to the m output units of the third and last layer.

When there is only one layer of nodes that act as input and output units, the network is said to be a *single-layer* FNN. Otherwise, if there are at least two layers (input and output layers - more than two, when one or more hidden layers are present), the network is said to be a *multi-layer* FNN (MLFNN or MLF network).

The basic element of a FNN (just like any Artificial Neural Network) is the artificial neuron which is a simple processing unit, u_i, and whose computation can be illustrated as shown in Figure 3.23. It involves calculating the so-called *net input* as the sum of the unit inputs x_j, weighted with the values w_{ij} associated to the corresponding synapses. A certain threshold θ_i is subtracted to this sum[4] and a function f, called *activation function*, is applied to the obtained

[3] In the literature, the input layer is identified, in some cases, with the input vector, i.e. supposed consisting of one unit for each input element. However, no processing would be done by such units: each of them would propagate its input to the next layer without any modification. For this reason, in the following, units of this kind are not considered as element of the network and the next layer is defined as input.

[4] The threshold value can be considered as the weight of an additional synapse along which a constant signal of value -1 or $+1$ travels. In the latter case, we talk about bias.

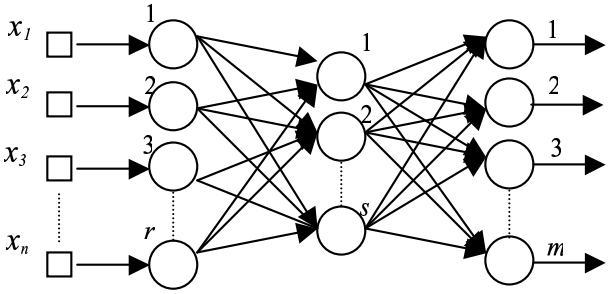


Figure 3.22: Example of FNN architecture with three layers: input, hidden and output layer.

value:

$$a_i = f_i(net_i) = f_i(\sum_{j=1}^{n} w_{ij}x_j - \theta_i) \tag{3.26}$$

An output function F can be applied to the activation value a_i from (3.26), which represents the excitement degree of the unit, in order to obtain the final output of the node:

$$o_i = F_i(a_i) \tag{3.27}$$

The function F is generally the identity function, while the activation function can be linear, sigmoid or hyperbolic. The net input constitutes the internal activity level of the node and applying the threshold θ_i to this value produces an affine transformation that can be seen, for instance, in the 2D Euclidean space for a two-input neuron as shown in Figure 3.24.

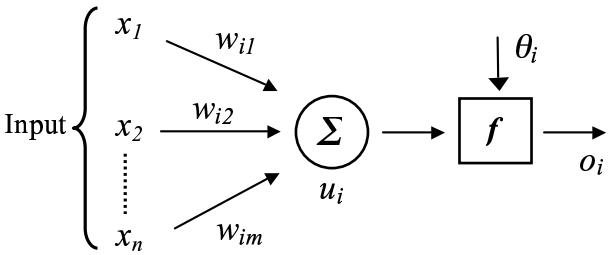


Figure 3.23: Illustration of an artificial neuron computation.

3.5.2 Perceptron

The Perceptron is the earlier and simpler FNN which was introduced by Rosenblatt in the 1958 [221]. Starting from the work of McCullough and Pitts, Rosenblatt proposed the first supervised training algorithm or *learning rule* capable of finding the optimal weight vector for a simple classification task.

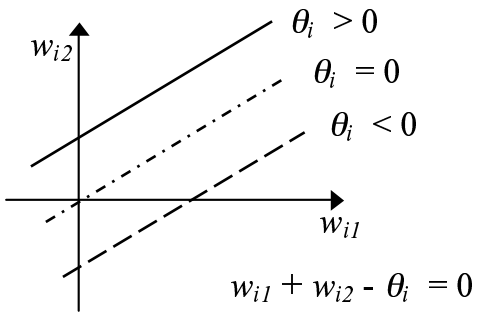


Figure 3.24: Illustration of the affine transformation produce by the threshold θ_i.

The Perceptron architecture consists of a single processing element with n real-valued inputs, the hard-threshold function as the activation function and a binary output o. Each of the two output values, $+1$ and 0, corresponds to one pattern class and is associated to one of the two regions, in the input space IR^n, delimited by the hyperplane the processing unit is trained to look for.

The training algorithm, proposed by Rosenblatt and known as the *Perceptron learning rule*, consists of an iterative correction of the input weights based on the evaluation of the error made by the network. Considering a training set composed of p examples of proper network behaviour, i.e. $\{(x_p, d_p)\}_p, p = 1, \ldots, P$, where x_p is the input vector and d_p is the desired binary output, for each training example, the weight adjustment is performed according to the following rule:

$$w_i = w_i + \Delta_p w_i \tag{3.28}$$

$$\Delta_p w_i = \eta(d_p - o_p)x_{pi} \tag{3.29}$$

where $i = 0, \ldots, n$ considering the bias as w_0, x_{pi} is the i-th component of the p-th input vector, w_i is the weight along the i-th synapsis, o_p is the actual output of the network for the p-th input vector and η is the so-called *learning rate*, i.e. a value that regulates the amount of the weight update.

According to the Perceptron Convergence Theorem [221], the above described algorithm is guaranteed to find a solution, i.e. the optimal weight vector, to the learning classification task whenever a solution exists. However, this convergence is valid only for linearly separable classes: in this case, the algorithm succeeds in finding the separating hyperplane. Moreover, evaluating the number of iterations of the algorithm is not possible, while it has been proved that, for some training sets, this number can be exponential in n and p [13]. If the classes are not linearly separable, the Perceptron algorithm moves cyclically from one weight vector to another, without supplying any information about the correctly classified examples. In cases of this kind, a single threshold neuron is not enough and networks of neurons are to be considered.

3.5.3 Adaptive Linear Elements and Delta Rule

While Rosenblatt was developing the Perceptron, Widrow and Hoff were studying another kind of neuron, called ADAptive LInear Element (ADALINE), which is similar to the Perceptron ex-

cept for the activation function: the ADALINE output depends linearly from the net input, i.e. the activation function is the identity function [268]. This condition changes the inner meaning of the weight update procedure, used for training, which is quite similar for the two models. Considering the training set $\{(x_h, d_h)\}_h, h = 1, \ldots, p$, for each sample pattern, the network output is compared to the target and the weights are adjusted in order to move it to the desired output. However, while the Perceptron algorithm tries to eliminate the wrong classifications, the ADALINE output approximates the correct output at every step of the training algorithm. In this way, the procedure achieves a better separating hyperplane in terms of the minimum distance along a parabolic surface.

The iterative technique for distance minimization consists in the gradient descent along the cost function surface. This cost function, E, is defined as the sum of the square errors, $E_p = (d_p - o_p)$, on the entire training set:

$$E = \sum_{p=1}^{P} (d_p - o_p)^2 \tag{3.30}$$

and its minimization is performed applying, at each step, the expression (3.28) but changing the weight vector w by following the opposite direction of the error gradient, i.e.:

$$\Delta w = -\eta \nabla E \tag{3.31}$$

where η is the learning rate and is chosen in order to maintain weight changes small. This means that the single weight is updated according to the following expression:

$$\Delta_p w_i = -\frac{\eta \delta E}{\delta w_i} \tag{3.32}$$

which can be explicitly expressed as the expression (3.29). In particular, indicating the difference between desired and actual output for p-th training pattern as δ_p, i.e.

$$\delta_p = d_p - o_p \tag{3.33}$$

the equation (3.29) can be written as

$$\Delta_p w_i = \eta \delta_p x_{pi} \tag{3.34}$$

This notation gives the name to the learning rule above defined, actually it is also called *delta rule* in addition to *Widrow-Hoff rule* or *Least Mean Square (LMS) rule*

The error function E can be represented in the n-dimensional space, whose single direction constitutes the variation space of each synaptic weight. If the training set is linearly separable, this function has a surface like a paraboloid with only one minimum point at null gradient. Then, the learning procedure consists in descending the surface toward this point with a trajectory defined by the initially chosen weight vector. If the pattern is not linearly separable, the delta rule yields a descent along the error surface without reaching the point with null gradient, but supplying only a partial solution to the classification task, i.e. only some classifications are

correct. Some problems, for instance wrong solutions even when the classes are linearly separable, can be avoided using a more general neuron with a non-linear activation function which is derivable. In this case, the weight update expression (3.34) becomes:

$$\Delta_p w_i = \eta \delta_p x_{pi} f'(net_p) \tag{3.35}$$

where f is the activation function and the apostrophe stands for derivative. Moreover, networks of one output layer of linear or non-linear neurons can be obtained for more complex problems (MADALINE I, MADALINE II) [269].

However, there exist some functions that cannot be approximated by a linear nor by a non-linear neuron nor by a network of only one layer of neurons. The typical problem used as an example of this conclusion is the approximation of the logic function exclusive OR (XOR) [166]: it requires at least two separating hyperplanes for correctly discriminating the input patterns and this situation can be achieved by considering a multilayer network with two linear input units and one output unit. Although the importance of the presence of at least two layers was known since the 1960s, a training algorithm for the training of the internal neurons was still not available. This lack was filled with the introduction of the Error Back Propagation algorithm which is discussed in the next sub-section.

3.5.4 Error Back-Propagation Networks

The Error Back-Propagation Networks (EBP) [225] are, undoubtedly, the most popular and used neural model. The EBP denomination refers to the learning rule used for supervised learning, while the network architecture corresponds to a MLF, with an input layer, an output layer and any hidden layers. The EBP algorithm is considered an extension to the MLF of the delta rule and for this reason is also called *generalized delta rule*. In particular, the EBP learning rule is defined for neurons with non-linear, differentiable activation function and allows the adaptation of an arbitrary number of internal layers in addition to the output one. This feature, together with the working simplicity, the application generality and the computing power made the EBP algorithm the cardinal model of the connectionist approach.

A brief description of the learning rule is given here. For a complete derivation see [94]. The main idea of the algorithm, as the name says, is back-propagating the error computed at the output layer to the inner units by introducing ideal feed-back connections. Through a recursive process, the inner synaptic weights are adjusted according to the amount of their influence to the network global output.

Let us consider the simpler case of a MLF network with only two layers: an input layer with r units and an output layer with s units. Extending the above introduced notation to the MLF network with multiple outputs, $\{(x_p, d_p)\}_p, p = 1, \ldots, P$ is the training set, x_{pj} represents the j-th element of the input vector x_p, d_{pi} is the i-th element of the desired output, w_{ij} is the weight along the synapse from the j-th input unit to the output unit i or from the j-th element of the input vector to the i-th input unit, o_{pi} the output of the i-th unit for the p-th training pattern, the total error function E is evaluated by considering only the results of the output units as follows:

$$E = \sum_{p=1}^{P} E_p \tag{3.36}$$

where E_p is the error committed on the single p-th pattern, i.e.

$$E_p = \frac{1}{2}\sum_{i=1}^{s}(d_{pi} - o_{pi})^2 \tag{3.37}$$

The EBP algorithm performs the following weight update after the processing of the entire training set:

$$\Delta_p w_{ij} = \eta \delta_{pi} o_{pj} \tag{3.38}$$

where

$$\begin{array}{ll} \delta_{pi} = (d_{pi} - o_{pi}) f_i'(net_{pi}) & \text{if i is an output unit} \\ \delta_{pi} = f_i'(net_{pi}) \sum_k \delta_{pk} \cdot w_{ki} & \text{if i is an input unit} \end{array} \tag{3.39}$$

k runs along the output units and η is the learning rate.

The weight change is performed as shown in Figure 3.25 which highlights the ideal feedback connections.

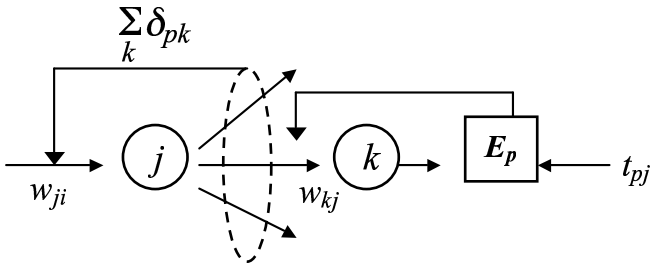


Figure 3.25: Illustration of the weight update for the j-th input unit.

The more general case, with more than two layers, can be easily obtained by extending the input layer rule to the additional layers.

To summarise, application of the EBP learning rule consists in two sequential passes for each training vector:

1. forward pass - first, the input is applied to the network and the output of each layer is forward propagated to compute the global response of the network, i.e. the value o_{pk}, $k = 1, \ldots, s$, for each output unit. This output vector is then compared to the target and the δ_{pk}, $k = 1, .., s$, values for the output units are computed.

2. backward pass - the delta values computed for the output units are recursively propagated backward to the internal layers until the input layer and the weight vectors are adjusted according to the rule (3.38).

The basic version of the algorithm performs the weight update operation after processing the entire training set. This behaviour corresponds to the so-called *batch* or *per-epochs* updating. The algorithm proceeds until a stop criterion is met, e.g. the error falls under a predefined threshold, changes in the gradient value are small or a predefined maximum number of epochs is reached. An *online* or *per-pattern* version of the algorithm is also available, with the same

kind of stopping criteria. In particular, weight updating, performed using only the information provided by the current pattern, produces an error minimization procedure which does not correspond exactly with a gradient descent. Although, for small values of the learning rate, batch and online version coincide, the noise added by instantaneous correction of the weights might help the algorithm convergence to the global minimum.

Actually, contrary to the delta and Perceptron rules, EBP does not assure the convergence to the point of minimum error and might get stuck in local minima. In fact, for a MLF network, the error function yields a surface that is more complex than that for a single-layer network. There can be many local minima due to weight vectors permutation that have the same error value or to an annulment of weights of opposite sign or, moreover, to the sum of many non-linear components in the input space. In this points, the error value is very small and the weight update is not enough to let the network get out of them. Another critical situation is provided by plateau where the first derivative is close to zero and the weight update is minimal. There have been various attempts to establish straightforward sufficient conditions for guaranteeing the absence of local minima [81, 250, 274]. In practise, the conditions found lead to networks with many input or hidden units. However, the resulting architectures do not seem to generalize very well because of the large number of free parameters to be adjusted [18]. Rules of thumb, used when the network get stuck in local minima and the error is still too large, consist in changing the learning parameters, in restarting the training algorithm with new initial weights or in adding some noise during the search for minimum. In particular, the latter solution might be helpful for the network generalization capability, but might cause the network to diverge if the error surface considerably changes.

Another expedient used to improve network convergence is the addition of the so-called *momentum*, i.e. a factor used to take into account the previous weight changes. The update rule becomes:

$$\Delta w_{ij}(t+1) = \eta d_i o_j + \alpha \Delta w_{ij}(t) \tag{3.40}$$

where α, called *inertia coefficient*, weights the influence of the additional factor and t stands for the iteration step. Using the momentum allows the choice of larger values for the learning rate and, even if there are no proofs that it actually improves the network convergence, various authors report a great increase of the convergence speed. In fact, besides the convergence, another critical aspect of EBP training is that it is very slow. Many variants of the basic version of the algorithm have been proposed aiming to improve the convergence speed or the error surface exploration. Generally, however, the efficacy of a certain variant depends on the shape of the same surface, hence estimating the best choice *a priori* is very difficult.

The algorithms proposed by various authors can be divided in the following categories:

- *Algorithmic changes*: methods that try to optimise the weight update procedure, without changing the network architecture. Examples are the Resilient Back-Propagation [218], the Delta-Bar-Delta algorithm [110], Quasi-Newton methods [56, 215], the Levenberg-Marquardt algorithm [91] and so on.

- *Heuristics applied to the algorithm*: techniques that change the way of applying the algorithm, without trying to optimise it. They are generally based on experimental results instead than theoretical considerations. Examples are the methodology of the Concurrent Gradient [96] or the Selective Update algorithm [98].

- *Regularization algorithms*: algorithms that change the error function, adding a cost factor to the weights in order to minimize the network architecture. Examples are Weight Decay [90] or the Rumelhart Regularization [226].

- *Techniques applied to the network*: changes of the network computation that do not modify the problem definition, for instance modification of the activation function.

Peculiar properties of the MLFNs are the universal approximation and the learning and generalization capabilities [94].

The presence of an input layer and, in case, of any hidden layers is the main features of these networks. The inner layers, in fact, assure an internal encoding of input vectors that allows the transformation of any n-dimensional input vector in a new m-dimensional vector, where m is the number of output units [10]. The Universal Approximation Theorem belongs [94] to this observation: it is an important principle that motivates mathematically the use of neural networks. However, it states only the existence of a network for any function approximation, without supplying any suggestions regarding the network architecture. For this reason, the theorem has limited practical implications.

On the other hand, the universal approximation does not imply the learning capability of the network: it is true that a network can approximate any function with the desired accuracy, but there are no evidence that the network successes in learning this approximation. In the field of Machine Learning, a theory regarding the so-called MLFN *learnability* has been developed [188]. It states that the learning capability consists in the learning algorithm ability to find correct values of the free parameters in order to assure the desired approximation accuracy. This ability seems to be related to various network features, such as network architecture, the training algorithm itself and the training set. In particular, while the generalization capability is linked to the amount of information contained in the training set, the network learnability is more likely related to the network typology and to the problem complexity. Unluckily, the only theoretical results in this field are upper bounds for the involved variables [12]: considering a two layers network with r input units and W synaptic weights, P patterns in the training set and a desired approximation error of ε, the following condition needs to hold:

$$P \geq O(\frac{W}{\varepsilon} \log_2 \frac{r}{\varepsilon})$$

As a consequence, a network with the above mentioned features cannot learn and generalize well if less than $\frac{W}{\varepsilon}$ patterns are used during the training phase.

Another problem regarding training is the network adaptability which affects its stability: these two properties, in fact, can be inversely proportional, i.e. high plasticity of the network can correspond to low stability of the same. Actually, a neural network is asked to adapt to significant external events while remaining stable after processing not meaningful events (*stability-to-plasticity dilemma*) [173]. On the contrary, it is possible that high flexible network, which has already learned a certain knowledge during the training phase, loses some of this information when a new pattern is presented. This phenomenon is known as *catastrophic interference* and has been investigated by various authors [161, 87, 88] which have outlined that determining factors might be the pattern presentation order, an intrinsic overlap of the patterns and the typology of the training procedure.

Dealing with the network generalization capability, it consists in the ability of extrapolating significant information even from a limited number of training patterns and in the correct behaviour on pattern never seen before. This is also a feature of other adaptive models used, for instance, in the Pattern Recognition field, however, the larger number of free parameters is a distinguishing characteristic of MLFNs. From a Pattern Recognition point of view, EBP networks can be compared to the *statistical* or *Decision Theoretic* approach: they are *feature-based*, i.e. they learn a mapping between input and output vectors, the hyperplanes yielded by each units can be thought as *decision boundaries* and the output units perform a non-linear discriminant function that distinguishes a class from the others. Moreover, it can be proved that an EBP network, trained by minimizing the square error, gives a direct estimation of Bayes a-posteriori probability function [260, 241, 79, 217, 277]. Various studies have shown the equivalence between EBP networks and the statistical approach to pattern classification and, for this reason, these networks has been seen as just alternatives to classic statistical methods. On the contrary, neural models offer many advantages with respect to above mentioned methods as they are non-linear, non-parametric and model-free systems able to deal with non-stationary data and non-Gaussian distributions [102]. Many studies, performed to compare the two approaches, have outlined the superiority of the ANN, in term of both theoretic [71, 6, 227] and applicative power [126, 47, 149, 165]

3.5.5 Radial Basis Function Networks

Radial Basis Function Networks (RBFNs) emerged as a variant of MLF network in the late 80's, even though their fundamentals involve much older pattern recognition techniques such as clustering, function approximation and spline interpolation [169]. Their use in pattern classification is based on the Cover theorem on the separability of patterns, which states that nonlinearly separable patterns can be separated linearly if the pattern is turned nonlinearly into a higher dimensional space [44]. Applying this theorem to an ANN means to find a network that converts the input patterns to a higher dimension after that each of them can be classified using only one layer of neurons with linear activation functions. Actually, RBFN topology is very simple, i.e. a two layers architecture of different typology neurons: the input layer consists of non-linear units whose activation function corresponds to a *radial basis function* (RBF), while the output layer consists of linear units (just like the ADALINE).

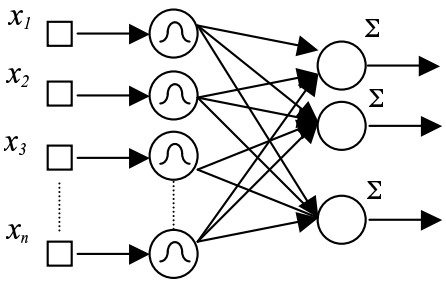


Figure 3.26: Example of a RBFN architecture.

RBFs are a special class of functions whose characteristic features are the radially symmetric

shape and the monotonic decrease of their response according to the distance from a central point. Mathematically, a RBF can be defined as follows:

$$\phi(\|x-c\|,\sigma) \tag{3.41}$$

where x is an input vector, c is the RBF central point, σ is a real positive scalar determining the RBF width, i.e. how spread the curve is, and $\|\cdot\|$ represents a distance measure, usually the Euclidean norm. The most common form of RBF used is the Gaussian function:

$$\phi(x) = exp(-\frac{\|x-c\|^2}{\sigma^2}) \tag{3.42}$$

while other possible choices are:

$$\phi(x) = \|x-c\|^2 \log\sigma^2$$

$$\phi(x) = (\|x-c\|^2+\sigma^2)^{-\sigma}$$

The use of RBF for the input layer of the network serves to perform a fuzzy or soft template matching between the input vector and a stored pattern, i.e. the weight vector: the output of each input unit has maximum value when the input exactly coincides with the weight vector, otherwise, as the difference between the two vectors increases, the neuron output approaches zero. The rate of this decrease is governed by σ. For this reason, RBFN are to be considered a hybrid kind of network for classification, with a first *prototype-based* layer, i.e. formed by neurons that extract, from the training set, a set of prototypes used as terms of comparison during the classification, and a second feature-based layer, like the EBP networks. In particular, in the literature, RBFN are generally considered as a smooth transition between Fuzzy Inference Systems and ANN, because each input neuron can be seen as supplying a degree of membership of the input pattern to the class of the corresponding prototype [60]. This interpretation confers to RBFNs the advantages of ANNs for mathematical tractability, especially for the back-propagation of the error and of the Fuzzy Systems for the incorporation of expert knowledge into the training procedure, especially in the assigning the initial value to the network parameters.

Many training algorithms have been tested for RBFNs within the supervised training strategy based on the minimization of the error function in (3.38). In the initial approaches, each data sample was assigned a RBF. This solution proved to be expensive in terms of memory requirement and in the number of parameters. Moreover, exact fit to the training data often causes bad generalization performance. Other approaches choose randomly or assumed known the hidden unit weights and calculate the output weights by solving a system of equation whose solution is given in the training set [32]. However, the matrix inversion required in this method is computationally expensive and could cause numerical problems in some situation. In [160, 231], the RBF centres are uniformly distributed in the data space and the function to be modelled with the network is obtained by interpolation. In [179], less basis functions then given data samples are used and a least squares solution that minimizes the interpolation error is defined. Orthogonal least squares using Gram-Schimdt is proposed in [38, 174]. Expectation-Maximization algorithm using a gradient descent algorithm for modelling the input-output distributions is employed in [35]. A two-stage training procedure is proposed in [168] and often used in practice. In the first stage, the input data is used alone to determine the parameters of the RBF using

unsupervised methods, e.g. clustering algorithms such as the *k-means*. In the second stage, the RBF are kept fixed and the second layer weights are optimised.

Dealing with the number of input units, various procedures have been employed for finding a suitable network topology. Usually, topology adaptive approaches use an additional regularization term to the cost function, depending on the number of input units [38]. Other methods employ cluster merging [174] or splitting [26].

RBFNs are capable of universal function approximation, like EBP networks [187]. However, they can offer advantages over the EBP in some applications, e.g. an RBFN can be easier to train than an EBP network. In particular, the possibility of choosing suitable parameters for the input units without having to perform a full non-linear optimisation of the network is one of the main advantages of RBFNs. There are many potential applications for ANN where unlabelled input data is plentiful while labelled training patterns are in short supply. In these cases, the two-stage training process of RBFN can be particularly useful, since the determination of the non-linear representation, given by the first layer, can be done using a large quantity of data. In particular, in order to obtain good generalization performance, the number of these data points should be large compared to the number of parameters to be determined.

3.5.6 Feed-Forward Neural Networks in Multimedia Analysis and Recognition

The computing strength and simplicity together with the application flexibility make the FFNN one of the neural models most used in practice. The usually implemented architectures are multi-layer with only two layers of input and output units.

The two typologies more largely used are EBP and RBF networks. However, FFNN are, often, combined in modular or hierarchical way to improve the classification or recognition capabilities of the neural system obtained. The hierarchical combination can involve ANN of different topologies obtaining in this way a higher degree of flexibility and reliability. These features are very important in solving highly complex problems which cannot be faced with a well-defined scheme. Optimised solutions are, on the contrary, required in these cases and can be achieved with neural architectures able of adapting to changes in problem representation model and able to supply adequate performance thanks to a hierarchical data analysis, reduction and understanding [23, 25, 24].

FFNN applications in multimedia processing range from restoration and compression to object detection and recognition, from visual data segmentation to image understanding. For accurate review, see [132, 61].

Restoration is a highly complex task because of conflicting criteria that are to be satisfied (e.g. resolution versus smoothness). FFNNs are primarily used in this field as filters for removing noise and can have relatively simple or highly complex architectures. Examples are the regression FFNN used in [85] and the modular FFNN used in [51]to mimic the behaviour of Kuwahara filter. FFNNs, trained for edge detecting, are also used for image enhancement [36, 209].

In **image compression**, FFNN approaches, consisting, for instance, in vector prediction [219, 262] or in wavelet coefficients identification [49], have to compete with well-established techniques such as JPEG, which could be used for comparison. The major advantage of ANNs

is that their parameters are adjustable and this feature may give better compression rates when training is performed on specific image material [61].

Good results have been obtained in **image segmentation**, which can be actually seen as a classification task: some studies [185, 157] have shown FFNN application validity in this field and the better performance obtained in comparison to other pattern recognition methods. Actually, the advantage of neural network approaches over classical statistical methods is the relative insensitivity to selection of the training sets, which is reinforced by generalization. Applications go from medical imaging [184, 206] to character recognition [84, 263].

As powerful pattern classifiers FFNNs are highly used and give best results in **multimedia detection and recognition** [132]. In this field, several different tasks can be identified, some of which are discussed in more details in the following.

The conversion of paper-based document information to electronic format is important for automated document delivery, document preservation and many other applications. FFNN have shown good capabilities in performing both handwritten and typed **character recognition**: the EBP model has been applied in several studies [82, 83, 4, 158]. In [137], four different typologies of ANNs, including EBPN, RBFN, Probabilistic Networks and Self-Organizing Maps, are compared: experimental results have shows that the RBF network has the highest classification accuracy and requires intermediate amounts of memory and training time as compared to the other neural models.

Fingerprint classification is one of the matured biometric techniques, used to people verification. It consists in a coarse-level matching of fingerprints, categorized according to the flow-like ridges, which form a special pattern in the central region of the fingerprint [63]. A complete evaluation of FFNN application to this task is given in [20], in which EBP and RBF models are compared with other statistical methods (e.g. Euclidean Minimum Distance and k-Nearest Neighbor). A simple EBP model has been adopted in several other studies [146, 105, 113], while in [118] it is introduced a pyramidal architecture constituted by several EBP network, each of which is trained to recognize fingerprint belonging to different classes.

FFNN is often used in order to facilitate detection or recognition of high-level features such as human faces. In **face detection**, the task is finding out if an image contains or not a face: it is an essential step for face recognition and can also be used alone in security and interface applications. The solutions proposed in literature usually work on sub-images or sliding windows of limited dimensions (e.g. 20x20 or 30x30) and try to improve the neural system performance using some optimisations methods in data pre-processing or in the network architecture. In [104], a *Polynomial Neural Network*(PNN) is combined with the Principal Component Analysis (PCA) and applied to cluttered images. A PNN [238] can be viewed as a generalized linear classifier, consisting of only one layer of neurons, which uses as stimuli not only the measurements of the input pattern but also the polynomials of these measurements. In order to reduce the number of parameters of the network, PCA is used for dimensionality reduction, but assuring also the improvement of classification efficiency thanks to feature extraction. A PNN is also used in [247], in combination, in this case, with *Hidden Markov Model* (HMMs) for first detect and then recognise faces in the images. A neural detector of frontal view faces is proposed in [5]: an EBP network is applied to sub-images 30x30 after a normalization process whose target is eliminating illumination variability for all possible face positions and adjusting the dimensionality of the processed data. For frontal face detection, 4x4 multi-resolution images (*quartet*

images) are scanned in [11], both in horizontal and vertical profile with a running window: first a SVM is trained to separate face from non-face patterns, then a first ensemble of EBP networks is trained for detecting patterns that fall between the cheeks and a second ensemble is trained for separating patterns that fall between eyebrows and the chin. A FFN network set is also adopted in [223]: first NN-based filters are applied to examine each 20x20 location in an image at several scale, looking for locations that might contain a face; an arbitrator is then used for merging detections from each filter and eliminating overlapping detections. A modified RBF neural network is used in [135]: modification consists in adoption of PCA coefficients, more exactly the distance of input pattern from the feature space detected with PCA.

In **face recognition**, the problem is to find a person in a database of faces or recognise people in real time. There are two main aspects to face in this task: determining whether someone is known (determining database membership) and identifying the person (determining identity). In [53], both question are solved using the discrete cosine transform to extract features from high dimensional facial data, an EBP network for determining identity and a *Counterpropagation* neural network [95] for determining database membership. A couple of EBP networks and a fuzzy system are combined in [65] in order to improve the decision making process. One EBP network is trained to separate authorized face images from non-face and non-authorized face images. Another network of the same typology is used as eye detector to determine the possible location of eye for a given image. Finally, a fuzzy system is implemented as the final decision making stage by utilizing the recognition rate provided by the neural networks. A Probabilistic RBF (RBFP) is introduced in [89]: it is derived from the combination of RBF and *Probabilistic Neural Networks*, trying to incorporate all the advantages of these two network typology while lowering their demerits. A RBFP model consists of 3 layers: the first layer is a nonlinear processing layer, generally consisting of selected centres from training examples; the second layer selectively sums the first layer output (generally the corresponding weight values are 1's); the last layer is just the output layer.

A significant body of work, both theoretical and experimental, has established the viability of ANNs as a useful technology for **speech recognition**: the neural models mostly used in this field are dynamic, i.e. able to take care of temporal sequences of patterns (*Recurrent and Time Delay Neural Networks*), however FFNNs are also used, particularly to augment speech recognisers whose underlying structure is essentially that of Hidden Markov Models. In [272], a MLF network is introduced to control an adaptive search technique, using observable features (e.g. each hypothesis score), for a HMM-based continuous speech recognition system. In [54], a three-layered FFNN is used to implement speech-frame prediction together with a Markov chain used to modulate the network weight parameters. A hybrid HMM-NN model is proposed in [27] in order to exploit at the same time the ability of dealing with temporal patterns, typical of HMM, and the pattern classification power of ANNs. A HMM-NN looks at a temporal context and uses an intrinsic discriminative training method, the EBP method, potentially achieving, in so doing, a better separation of similar acoustic classes. Recently, to exploit ANN capability to mix input features coming from different input sources without any particular assumption on their statistical distribution, *Multi-Source* NN have been introduced for speech recognition [76], aiming to use the synergy of different and partially complementary speech parameterisations in order to improve accuracy of the acoustic matching.

In **the medical field**, FFNN applications involve clinical functions of diagnosis, progno-

sis and survival analysis, the medical domains of oncology, critical care and cardiovascular medicine. An accurate review of the evidence of healthcare benefits assured by the use of ANNs is made in [153]. Several examples of FFNN applications can be drawn from the medical imaging field [189]. A modified EBP neural model is used in [279] for the detection of micro-calcification clusters in digitised mammograms. Different features, extracted from the spatial and spectral domain are provided as input to the network trained by using the EBP algorithm modified with Kalmar filtering. EBP networks have been applied in several other studies for the same diagnosis problem [57, 244, 156]. In [190], a two-level FFNN network is used to detect lung cancer nodules found on digitised chest radiographs: a first level EBP network is trained for detecting suspected nodule areas on low-definition images, then these zones are analysed by a second level EBP network that acts as classifier, in order to confirm or refuge the presence of nodules. Also in this case, there are many other examples in literature of FFNN applications for the resolution of the same problem [150, 154, 151]. A more complicated architecture than that introduced in the abovementioned study is the one adopted in [22] for the characterization of brain volume density in *Computer Tomography* or *Magnetic Resonance Imaging*. Two different typologies of ANNs, namely SOM and EBP network, are hierarchical combined in two levels and separately trained to perform the basic recognition task. Aim of this approach is exploiting the differences among the extracted features to improve the classification capability, the ability of easily change the number of features, the possibility to implement a full 3D approach, in terms of spatial geometric relations among the neuro-functional structures. In the first level, a set of SOM modules, one for each feature, are used for clustering the feature values in crisp classes. The second level is composed by a single EBP module that receives the outputs of the lower level, refining the classification and giving the final response. The validity of this approach has been confirmed by another application consisting in the categorization of cerebral microemboli in ultrasound images: a similar architecture is used to recognize solid or gaseous microemboli on the basis of a set of opportunely selected morphological and statistical features [41]. Other hierarchical combinations of EBP network are widespread in the medical imaging field [64, 230, 216].

In **CBIR** context, MLF networks are mainly used to estimate the similarity measure, based on extracted image features, for ranking the relevance of images in the database to a query image. Various approaches have been proposed in the literature which differ from each other in the network topology, which can be EBP or RBF, in the granularity of feature extraction, which can interest image regions or the image as a whole, and in the typology of retrieval, which can be non-adaptive or adaptive, i.e. with user's feed-back. Moreover, the network training phase can be executed before system application or embedded in the retrieval process. In [62], the validity of FNN application to CBIR is explored: using low-level visual features of the interesting regions contained in the image (e.g. eccentricity, compactness, elongation), different neural models, EBP, RBF, *General Regression* networks and SVM (see section 3.4) are trained with the aim of approximating a similarity function $g(u,v)$, where u and v are the feature vectors of the two images used as input to the networks. Each trained network is, then, employed for non-adaptive image retrieval. Experimental results, obtained on two different image typologies, seem to validate the adoption of neural models for predicting the user's notion of similarity. An EBP model together with *B-splines* for affine object representation is employed in [271] for non-adaptive content-based image and video retrieval, developed locating an initial number of

candidate objects and then refining the selection with curve matching. B-splines are used for shape representation of the object contours, obtained from image segments; a three-layers EBP network is trained to construct a prototype object database. The representation of curve prototypes is used as input to the network and its output is assigned to each primary object class, in order to constrain the search procedure into a small subset of classes. A two-layers EBP network is used in [107] to estimate the similarity measure from a set of image features, regarding colour and location information of image regions. A regional division method is performed by quantization, boundary processing and labelling. In [276], a neural network approach to adaptive region-based CBIR is proposed. *Multiple Instance Learning* [278] is integrated with the query refining process to learn two regions of interest from the user's feedback. Both local colour and local texture features are extracted from candidate image regions and two EBP networks (one for each region) are used to estimate, from the feature vector, how much the regions meet the user's concept. Given a query image, the most similar images in the database are first retrieved using a distance-based metric, labelled according to the user's feed-back and then used as training set for the networks. The feed-back and learning are executed iteratively until user satisfies. A similar approach is proposed in [172]: a cascade of non-homogeneous networks is used to progressively model image similarity through continual relevance user feed-back, in this way, the user directly modifies the query characteristics by specifying his desired image attributes in the form of training examples. A Self-Organizing Tree Map [129] and a Learning Vector Quantization [125] are adopted to generate the prototypes in the form of local data clustering: the former to create the relevant image prototypes and the latter to modify the same taking into account the negative samples. The prototype vectors obtained in this way are passed to a RBFN for the evaluation of the similarity between each prototype and the input vector corresponding to image in the database and, in so doing, ranking the database. The RBFN is a single-pass network, since each prototype vector is assigned as the centre of the corresponding Gaussian unit in the network. A RBF network is also used in [140] for ranking retrieved images in an adaptive CBIR. Images are represented by means of feature vectors consisting in low-level visual feature (colour, texture and shape) and coefficients obtained with the biorthogonal wavelet decomposition. During retrieval, the query feature vector is compared with the vectors of the database images and compared metric values are obtained. These values are combined using the RBFN for the ranked list of retrieved images. User's judgement of each retrieved image is then received and used to incrementally train the neural network.

Other application field include **remote sensing** [239, 19] [70], **motion tracking** [139, 159], **multisensor signals processing** [78].

In recent years, a great attention has moved to the challenging task of generating, extracting and retrieving information simultaneously from different medias, including text, audio, image and video. FFNN application in this field is promising thanks to the capacity of ANNs to deal with stimuli coming from different sources without assuming any hypothesis on their distribution. Some examples of FFNN application to the processing of signals of different types fall in the **audio-video conversion and synchronization**, which include joint audio-video coding, speech recognition and speech-assisted synchronization [21]. The goal of speech-driven facial animation is to synthesize realistic video sequences from acoustic speech. In [214], a three-layered EBP network is used for voice to image conversion. *Linear Perdition Coding Cepstrum* parameters [213] are given in input to the network and the mapping is learned by backpropaga-

tion training, with both voices and image parameters which are synchronized to each other. For speech recognition, an EBP network is adopted in [171] to fuse information from the acoustic channel and information from the visual one. The pixel values of the mouth image are fed to the network directly, without extracting any kind of features. The network is trained to estimate the voice spectrum from the image. The spectrum obtained as a result is then combined with the true spectrum and the couple is fed to a recognition system.

3.6 Convolutional Neural Networks

The term *Convolutional Neural Networks* (CNN) is used to describe a neural architecture well-suited to two dimensional input data (e.g. digital images), based on spatially localized neural input. Lecun and Bengio [2] have highlighted three architectural ideas common to CNNs : local receptive fields, shared weights (weight averaging), and often spatial down sampling. Figure 3.27 shows an architecture of a convolutional neural network.

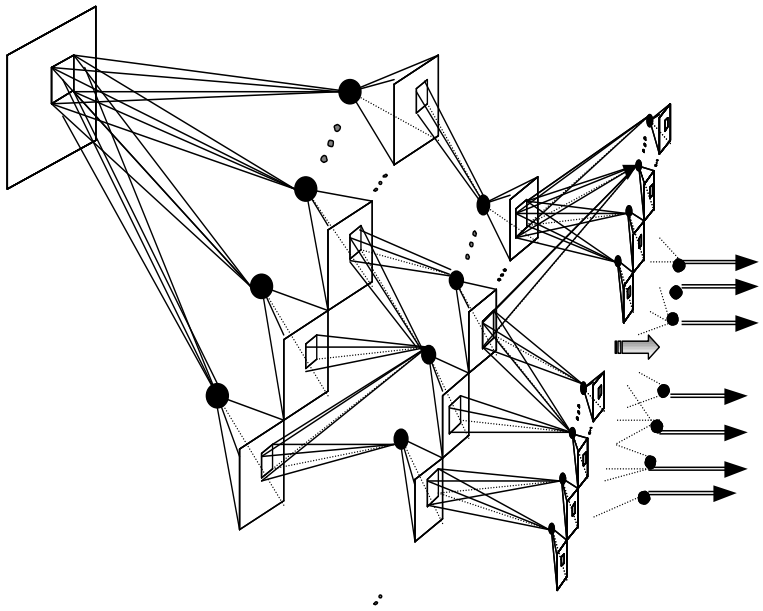


Figure 3.27: An architecture of a 5 layer convolutional neural network

A typical CNN consists of a set of layers, each one containing one or more planes. The input image I is first approximately centered and normalized before entering at the input layer. Thus, each unit in a plane receives input from a small neighborhood in the planes of the previous layer. In fact, the layers alternate between convolution layers with feature maps C^i_{kl}:

$$C^i_{kl} = g(I^i_{kl} \bigotimes W_{kl} + B_{kl}) \tag{3.43}$$

and non overlapping sub-sampling layers with feature maps S^i_{kl}:

$$S^i_{kl} = g(I \downarrow^i_{kl} w_{kl} + \Theta b_{kl}) \tag{3.44}$$

where g denotes an activation function, B , respectively b the biases, W and w the weights, I^i_{kl} the i^{th} input and $I\downarrow^i_{kl}$ the down sampled i^{th} input of the neuron of group k of layer l , Θ a matrix composed by 1 and $\bigotimes$ denotes a two dimensional convolution.
Processing units with identical weight vectors and local receptive fields are arranged in a spatial array, creating an architecture with parallels to models of biological systems cite**??**. A CNN image mapping is characterized by the strong constraint of requiring that each neural connection implements the same local transformation at all spatial translations. In other words, the weights forming the receptive field for a plane are forced to be equal at all points in the plane. This dramatically improves the ratio between the number of degrees of freedom in the system and number of cases, increasing the chances of generalization [142]. Thus, each plane can be considered as a feature map implementing a fixed feature detector that is convolved with a local window scanned over the planes in the previous layer. The number of planes will denote the number of features that had to be learnt. These layers represent the convolutional layers.The first layer implements non linear template matching at a relatively fine spatial resolution, extracting basic features of the data. Subsequent layers learn to recognize particular spatial combinations of previous features, generating "patterns of patterns" in a hierarchical manner. If down-sampling is introduced in the model, then subsequent layers perform pattern recognition at progressively larger spatial scales, with lower resolution. A CNN with several down-sampling layers enables processing of large spatial arrays, with relatively few free weights.
Training a CNN is achieved in a supervised manner by using the standard back-propagation algorithm adapted for CNN. Given input pattern μ, the hidden unit jq (neuron q of the j^{th} filter) receives net input (cf. Figure 3.28) :

$$V^{\mu}_{jq} = g(h^{\mu}_{j,q}) = g(\sum_{k} \sum_{t\in\{T_{11},\ldots,0,\ldots,T_{n_1,n_1}\}} w^{t}_{jk}\xi^{\mu}_{k,q+t} + b_j) \tag{3.45}$$

where :

- g denotes an activation function;
- V^{μ}_{jq} denotes the neural output of the hidden unit jq associated to the pattern μ;
- w^{t}_{jk} denotes the weight of the hidden unit jq; depending of the filter size with $t \in \{T_{11},\ldots,0,\ldots,T_{n_1,n_1}\}$, with n_1 denoting the maximal number of elements of the filter;
- b_j denotes the bias of the filter j.

The cost function E of the CNN is defined by:

$$E = \frac{1}{2}\sum_{\mu ip}[\zeta^{\mu}_{ip} - 0^{\mu}_{ip}]^2 \tag{3.46}$$

where ζ^{μ}_{ip} denotes the i^{th} output associated to the pattern μ.
The derivative of E with respect to the s^{th} weight of the filter connecting the j^{th} feature array to the i^{th} output array is defined by:

$$\frac{\partial E}{\partial w^{s}_{ij}} = -sum_{\mu p}[\zeta^{\mu}_{ip} - g(h^{\mu}_{ip]})]g'(h^{\mu}_{ip]})V^{\mu}_{j,p+s} \tag{3.47}$$

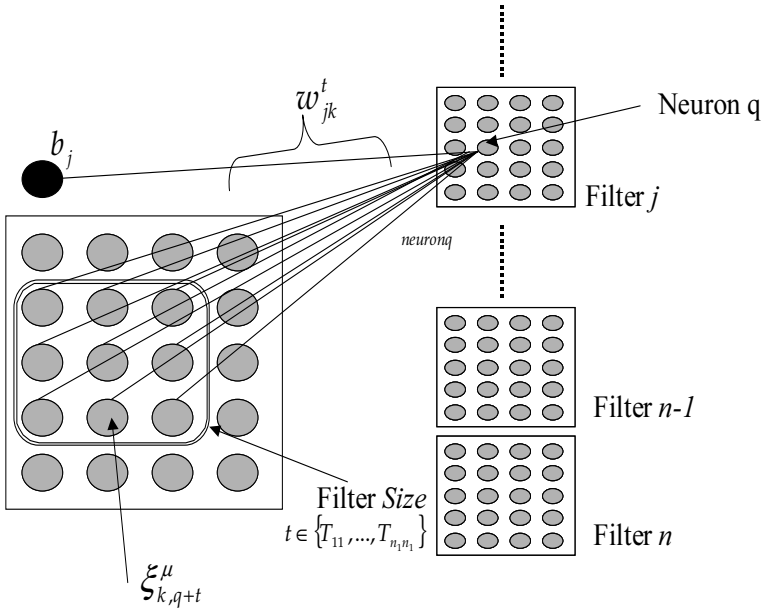


Figure 3.28: An example of shared weight

which, when used in combination with the familiar gradient descent weight update rule, yields:

$$\Delta w_{ij}^{s} = \eta \sum_{\mu p} \delta_{ip}^{\mu} V_{ip}^{\mu} \tag{3.48}$$

where :

$$\delta_{ip}^{\mu} = [\zeta_{ip}^{\mu} - g(h_{ip]}^{\mu})]g'(h_{ip]}^{\mu}). \tag{3.49}$$

After describing the generalized delta rule associated to convolutional neural network, we propose to expose, in the next section, how CNNs can be used for a face processing task.

3.6.1 CNN Applications

In this section, we present various applications proposed in the literature that make use of CNN. These applications include face detection, face recognition and handwritten recognition.

3.6.1.1 Face Detection

Human face detection is today a very hot research topic, due to its wide range of possible applications, such as security access control, model-based video coding, content-based video indexing or advanced human and computer interaction. It is also required as a preliminary step to face recognition and expression analysis. Numerous approaches for face detection have been proposed in the last years, many of them being described and compared in two interesting recent

surveys by Yang *et al.* [273] and Hjelmas *et al.* [99].

In fact, the approaches that have exhibited the best results are based on the use of neural networks. The first advanced neural network-based face detector has been proposed by Rowley *et al.* [224]. In their approach, the author are able to detect faces at any degree of rotation in the image plane. Their system employs multiple networks: a router network first processes each input window to determine its orientation and then uses this information to prepare the window for one or more detector networks. In fact, the network assumes that the input window contains a face and is trained to estimate its orientation. The inputs to the network are the intensity values in a 20×20 pixel window of the image (which have been preprocessed by a standard histogram equalization algorithm). After the router network, the window is aligned to make any face that may be present upright. The neural network based upright frontal face detection system denotes a retinally connected neural network that examines small windows of an image and decide whether the window contains a face or not. The system arbitrates between multiple networks to improve performance over a single network. It first applies a set of neural network based filters to an image, and then uses an arbitrator to combine the outputs. The filters examine each location in the image at several scales, looking for locations that might contain face. The arbitrator then merges detection results from individual filters and eliminates overlapping detections.

Various papers on face detection based on neural networks have followed the path of Rowley such as [67, 138, 222].

Finally, Garcia and Delakis [72] have shown that an efficient face detection system does not require any costly local illumination correction before classifying image areas as face or non face, and that the input image could be processed as a whole, with a fast pipeline of simple convolutions and subsampling modules implemented by a CNN, thus greatly speeding up the detection process. Very high detection rate are obtained with a particularly low level of false positives, as demonstrated on difficult test data sets, while running at approximately 12 frames per second on a conventional desktop.

3.6.1.2 Face Recognition

Various techniques for CNN-based face recognition have been proposed in the literature. In this document, we focus our interest on the two main ones. The first one has been elaborated by Lawrence [136] and in this approach, the author is interested in robust face recognition with varying facial detail, expression, pose, etc. However, the author does not consider invariance to high degrees of rotation or scaling, assuming that a minimal preprocessing stage is available if required. The system explores a local image sampling technique to reach a partial lighting invariance. Then, a Self Organizing Map (SOM) is used to project the image sample representation into a quantized lower space (the author specifically uses a three dimensional SOM corresponding to three output features). In fact, the local image samples are passed through the SOM at each step, thereby creating new training and test sets in the output space created by the SOM (each input image is now represented by three maps, each of which corresponds to a dimension in the SOM). The size of these maps is equal to the size of the input image divided

by the step size. Then, a CNN is trained on the newly created training set, before classification. The author shows that this method is able to efficiently perform classification, and consistently exhibits better classification performances than the eigenfaces approach [252].
In the second CNN-based approach, the author describes how exploiting CNN to realize robust face analysis such as the one presented by Fasel [1]. The author proposes a data driven face analysis approach able to extract features relevant to a given face analysis task and that is robust with regard to face location changes and scale variations. This is achieved by deploying CNNs, which are either trained for facial expression recognition or face identity recognition. In fact, two CNNs are combined, so that they may complement each other by delivering context information. This is for example the case with facial subject, when attempting facial expression recognition, as each individual has not only a different facial physiognomy leading to a specific facial action display, but furthermore also performs facial expressions with individual intensities. In fact, the approach combines two CNNs for the task of face recognition and facial expression by using a two layer MLP in order to merge the extracted data for improved and personalized facial expression recognition. By this way, the authors prove that, by combining facial expression with facial recognition, the results of facial expression recognition have been improved by about 20% when using the synergy that stems from processing the output of the facial expression and face recognition networks a fusion network where the extracted information is combined.

3.6.1.3 Visual Document analysis

It is well known that CNNs are a powerful technology for classification of visual inputs arising from documents because spatial topology is well captured by CNN [3]. Effectively, the author reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. CNNs, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques. To be able to reach handwritten digit recognition, a similar conception as the ones described on the previous sections is needed. However, although CNNs have been proposed for visual task for many years, they are still not popular in the engineering community. That is why some papers have proposed new methods for such networks that are much easier that previous techniques ad allow easy debugging. More information can be found in [242].

3.7 Ensemble Techniques

The key idea in ensemble research is; if a classifier or predictor is unstable then an ensemble of such classifiers voting on the outcome will produce better results - better in terms of stability and accuracy. While the use of ensembles in Machine Learning research is fairly new, the idea that aggregating the opinions of a committee of experts will increase accuracy is not new. The Condorcet Jury Theorem states that:

If each voter has a probability p of being correct and the probability of a majority of voters being correct is M, then p > 0.5 implies M > p. In the limit, M approaches 1, for all p > 0.5, as the number of voters approaches infinity.

This theorem was proposed by the Marquis of Condorcet in 1784 [42] - a more accessible reference is by Nitzan and Paroush [181]. We know now that M will be greater than p only if there is diversity in the pool of voters. And we know that the probability of the ensemble being correct will only increase as the ensemble grows if the diversity in the ensemble continues to grow as well. Typically the diversity of the ensemble will plateau as will the accuracy of the ensemble at some size between 10 and 50 members.

In ML research it is well known that ensembling will improve the performance of unstable learners. Unstable learners are learners where small changes in the training data can produce quite different models and thus different predictions. Thus, a ready source of diversity is to train models on different subsets of the training data. This approach has been applied with great success in eager learning systems such as Neural Networks [92] or Decision Trees [30]. This research shows that, for difficult classification and regression tasks, ensembling will improve the performance of unstable learning techniques such as Neural Networks and Decision Trees. Ensembling will also improve the accuracy of more stable learners such as k-NN or Naïve Bayes classifiers, however these techniques are relatively stable in the face of changes in training data so other sources of diversity must be employed. Perhaps the most popular choice for stable classifiers is to achieve diversity by training different classifiers on using different feature subsets [101, 100, 46].

Krogh and Vedelsby [130] have shown that the reduction in error due to an ensemble is directly proportionate to the diversity or ambiguity in the predictions of the components of the ensemble as measured by variance. It is difficult to show such a direct relationship for classification tasks but it is clear that the uplift due to the ensemble depends on the diversity in the ensemble members.

Colloquially, we can say that; if the ensemble members are more likely on average to be right, and when they are wrong they are wrong at different points, then their decisions by majority voting are more likely to be right than that of individual members. But they must be more likely on average to be right and when they are wrong they must be wrong in different ways.

3.7.1 Bagging

The simplest way to generate an ensemble of unstable classifiers such as Neural Nets or Decision Trees is to is to use Bootstrap Aggregation, more commonly known as Bagging [30]. The basic idea for a bagging ensemble is shown in Figure 3.29; given a set of training data T and a query sample $\mathbf{x}_q$ the key steps are as follows:

1. For an ensemble of n members, generate n training sets T_i $(i = 1,n)$ from T by bootstrap sampling, i.e. sampling with replacement. Often $|T_i| = |T|$.

2. For each T_i; let Tv_i be the set of training examples not selected in T_i (this set can be used as a validation set to control the overfitting of the ensemble member trained with T_i).

3. Train n classifiers $f_i(,T_i)$ using the T_i training sets. The validations sets Tv_i can be used to control overfitting.

4. Generate n predictions for $\mathbf{x}_q$ using the n classifiers $f_i(,T_i)$.

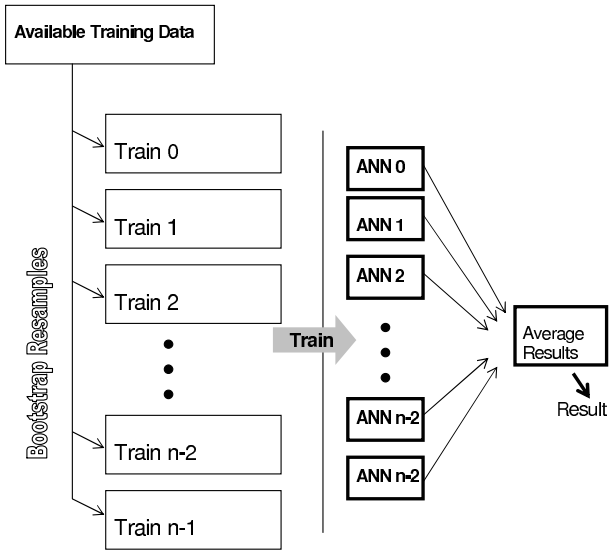


Figure 3.29: An overview of a bagging ensemble

5. Aggregate these n predictions $f_i(\mathbf{x}_q, T_i)$ to get a single prediction for $\mathbf{x}_q$ using some aggregation function.

The formula for this ensemble prediction would be:

$$f_E(x_q, T) = F(f_1(\mathbf{x}_q, T_1), f_2(\mathbf{x}_q, T_2), ..., f_n(\mathbf{x}_q, T_n)) \tag{3.50}$$

The simplest approach to aggregate a regression ensemble is by averaging and the simplest for a classification ensemble is by averaging or by *weighted* averaging:

$$f_E(x_q, T) = \sum_{i=1}^{n} w_i \times (f_i(x_q, T_i) \tag{3.51}$$

where $\sum_{i=1}^{n} w_i = 1$.

Provided there is diversity in the ensemble (and the bootstrap resampling should deliver this), the predictions of the ensemble $f_E(x_q, T)$ will be more accurate that the predictions from the ensemble members $f_i(x_q, T_i)$.

3.7.2 Boosting

Boosting is a more deliberate approach to ensemble building. Instead of building the component classifiers all at once as in Bagging; in Boosting, the classifiers are built sequentially. The principle can be explained with reference to Figure 3.30. This figure shows a true decision surface and a decision surface that has been learned by the classifier. The errors due to the discrepancy between the true decision surface and the learned one are highlighted. The idea with boosting is to focus on these errors in building subsequent classifiers. In boosting, classifiers are built sequentially with subsequent classifiers focusing on training examples that have not been

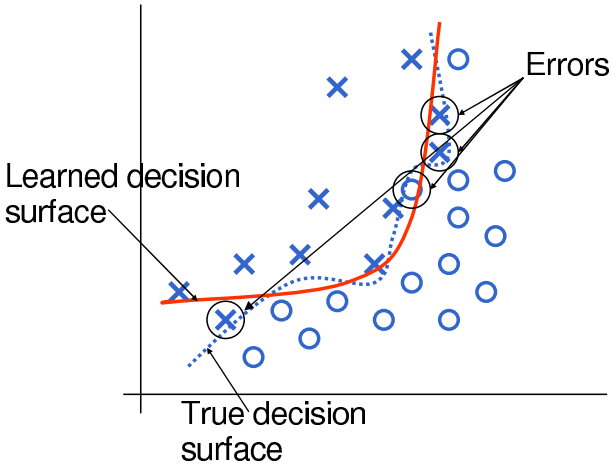


Figure 3.30: Difficult to classify examples near the decision surface

learned well by earlier classifiers. This is achieved by adjusting the sampling distribution of the training data. The details as outlined by Schapire[234] are as follows (where $(x_1,y_1),\ldots,(x_m,y_m)$ with $x_i \in \mathbf{X}, y_i \in Y = \{-1,+1\}$ is the available training data):

1. Initialise the sampling distribution $D_1(i) = 1/m$
2. For $t = 1,\ldots,T$:
   (a) Train classifier using distribution D_t.
   (b) Hypothesis this classifier represents is $h_t : \mathbf{X} \to \{-1,+1\}$.
   (c) Estimate error of this classifier as $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$.
   (d) Let $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$.
   (e) Update:
   $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha t} & \text{if } h_t(x_i) \neq y_i \end{cases} \tag{3.52}$$
   where Z_t is a normalisation factor set to ensure that D_{t+1} will be a distribution.
   (f) Continue training new classifiers while $\varepsilon_t < 0.5$.
3. This ensemble of classifiers can be used to produce a classification as follows:
   $$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)). \tag{3.53}$$

Clearly, the role that classification error plays in this algorithm means that this formulation only works for classification problems. However, extensions to the boosting idea for regression problems exist; two representative examples are ADABoost.R2 by Drucker [59] and BEM by Avnimelech and Intrator [7].

3.8 Strengths and Weaknesses

A summary of the strengths and weaknesses of these techniques from the point of view of Muscle participants are as follows:

- Multimedia data is inclined to be of very high dimension. In some circumstances, this may exclude some techniques such as decision trees or neural networks that work best with data of lower dimension. This may not always be the case however and we have presented examples of both decision trees and neural nets in use on multimedia data.
- Support Vector Machines work very well on high dimension data and are the *fashionable* ML technique of the moment. A significant drawback of SVMs is the fact that they are quite difficult to implement. However some good quality public-domain implementations are available.
- Nearest Neighbour classifiers *are* easy to implement and should not be overlooked. The advantages that derive from the transparency of *k*-NN classifiers should not be underestimated.
- It will often be the case that the accuracy of a classifier will be improved by aggregation into an ensemble. With the processing power available today this can be quite a practical course of action.

Bibliography

[1] Fasel B. Robust Face Analysis Using Convolutional Neural Networks. In *ICPR 2002*, Quebec, Canada, 2002.

[2] Lecun Y. and Bengio Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks, MIT Press*, pages 255–258, Cambridge MA, 1995.

[3] Bengio Y. Lecun Y., Bottou L. and Haffner P. Gradient-based learning applied to document recognition . In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[4] A. Amin. Off-line arabic character recognition: the state of the art. *Pattern Recognition*, 31(5):517–530, 1998.

[5] D. Anifantis, E. Dermatas, and G. Kokkinakis. A neural network method for accurate face detection on arbitrary images. In *Proc. of ICECS, 6th IEEE International Conference on Electronics, Circuits and Systems*, volume 1, pages 109–112, 5-8 Sept 1999.

[6] H. Asoh and N. Otsu. An approximation of nonlinear discriminant analysis by multilayer neural networks. In *In Proc. Int. Joint Conf. Neural Networks, San Diego, CA*, pages III–211–III–216, 1990.

[7] Ran Avnimelech and Nathan Intrator. Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, 11(2):483–497, 1999.

[8] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. of the 8th IWFHR*, pages 49–54, 2002.

[9] A. Barla, F. Odone, and A. Verri. Hausdorff kernel for 3d object acquisition and detection. In *In Proceedings of the European Conference on Computer Vision, LNCS 2353*, page p. 20 ff, 2002.

[10] A.R. Barron. Universal approximation bounds for superposition of sigmoid functions. *IEEE Trans. Information Theory*, 39:930–945, 1993.

[11] N. Bassiou, C. Kotropoulos, T. Kosmidis, and I. Pitas. Frontal face detection using support vector machines and back-propagation neural networks. In *Proc. of IEEE International Conference on Image Processing*, volume 1, pages 1026–1029, 7-10 Oct. 2001.

[12] E. B. Baum and D. Hausler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[13] Eric B. Baum. The perceptron algorithm is fast for nonmalicious distributions. *Neural Computation*, 2:248–260, 1990.

[14] C. Berg, J. P. R., Christensen, and P. Ressel. *Harmonic Analysis on Semigroups.* New York: Springer-Verlag, 1984. 18.

[15] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, New York, 1988.

[16] José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. John Wiley and Sons, 1997.

[17] J. Besag. Towards bayesian image analysis. *J. of Appl. Stat.*, 16:395–407, 1989.

[18] M. Bianchini, P. Frasconi, and M. Gori. Learning without local minima in radial basis function networks. *IEEE Trans. On Neural Networks*, 6(3):749–756, 1995.

[19] H. Bischof and A. Leonardis. Finding optimal neural networks for land use classification. *IEEE Transactions on Geoscience and Remote Sensing*, 36(1):337–341, 1998.

[20] J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson. Evaluation of pattern classifiers for fingerprint and ocr application. *Pattern Recognition*, 27:485–501, 1994.

[21] Z. Bojkovic and D. Milovanovic. Recent trends in neural networks for multimedia processing. In *NEUREL '02 6th Seminar on Neural Network Applications in Electrical Engineering*, pages 41–45, 26-28 Sept. 2002.

[22] S. Di Bona, H. Niemann, G. Pierri, and O. Salvetti. Brain volumes characterisation using hierarchical neural networks. *Artificial Intelligence in Medicine*, 28:307–322, 2003.

[23] S. Di Bona and O. Salvetti. Automatic monitoring of states evolution in dynamic scene supervision. *Pattern Recognition Image and Analysis*, 13(3):495–504, 2003.

[24] S. Di Bona and O. Salvetti. A multilevel neural approach to dynamic scene analysis. *Pattern Recognition and Image Analysis*, 13(1):86–89, 2003.

[25] M.G. Di Bono, G. Pieri, and O. Salvetti. A tool for system monitoring based on artificial neural networks. *WSEAS Transaction on System*, 3(2):746–751, 2004.

[26] A.G. Bors and G. Gabbouj. Minimal topology for a radial basis function neural network for pattern classification. *Digital Signal Processing: a review journal*, 4(3):173–188, 1994.

[27] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1993.

[28] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

[29] L. Breiman, J.H. Friedman, and R.A. Olshen. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series, 1984.

[30] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[31] L.D. Broemeling. *Bayesian analysis of linear models*. Dekker, 1985.

[32] D.S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, pages 321–355, 1988.

[33] W. Buntine. A guide to the literature on learning probabilistics networks from data. *IEEE Transactions on knowledge and data Engineering*, 8:195–210, 1996.

[34] J.D. Carswell, D.C. Wilson, and M. Bertolotto. Digital image similarity for geo-spatial knowledge management. In *6th ECCBR*, Berlin, 2002. Springer Verlag.

[35] I. Chan and S.A. Kassam. Rbfn resytoration of nonlinearly degraded images. *IEEE Trans. on Image Processing*, 5(6):964–975, 1996.

[36] V. Chandrasekaran, M. Palaniswami, and T.M. Caeli. Range image segmentation by dynami neural network architecture. *Pattern Recognition*, 29(2):315–329, 1996.

[37] E. Chang, B. T. Li, G. Wu, and K.S. Goh. Statistical learning for effective visual information retrieval. In *IEEE International Conference on Image Processing*, Barcelona, September 2003.

[38] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. On Neural Networks*, 2(2):302–309, 1991.

[39] N. Christianini and J. Shawe-Taylor. *Introduction to support vector machines*. Cambridge University Press, 2000.

[40] D. Cohn. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[41] S. Colantonio, O. Salvetti, and F. Sartucci. Automatic recognition and classification of cerebral microemboli in ultrasound images. In *Accepted for 7-th International Conference on Pattern Recognition and Image Analysis: New Information Technologies, St. Petersburg, Russian Federation, October*, pages 18–23, 2004.

[42] Marquis J. A. Condorcet. Sur les elections par scrutiny. *Histoire de l'Academie Royale des Sciences*, 31-34, 1781.

[43] Peter Congdon. *Bayesian Statistical Modelling*, volume Wiley Series in Probability and Statistics. John Wiley & Sons, Chichester, 2001.

[44] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. on Electronic Computers*, pages 326–334, 1965.

[45] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[46] Padraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In Ramon López de Mántaras and Enric Plaza, editors, *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 31 - June 2, 2000, Proceedings*, pages 109–116. Springer, 2000.

[47] S. P. Curram and J. Mingers. Neural networks, decision tree induction and discriminant analysis: An empirical comparison. *J. Oper. Res. Soc.*, 45(4):440–450, 1994.

[48] R. Dahyot, A. C. Kokaram, N. Rea, and H. Denman. Joint audio visual retrieval for tennis broadcasts. In *IEEE proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Hong Kong, April 2003.

[49] J.G. Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *EEE Trans. Acoustics, Speech Signal Processing*, 36(7):1169–1179, 1988.

[50] Leake D.B. and Wilson D.C. Remembering why to remember: performance-guided case-base maintenance. In *Advances in Case-Based Reasoning*, pages 161–172. Springer Verlag, Berlin, 2000.

[51] D. de Ridder, R.P.W. Duin, P.W. Verbeek, and L.J. Vliet. The applicability of neural networks to non-linear image processing. *Pattern Analysis Applications*, 2(2):111–128, 1999.

[52] M. DeGroot. *Optimal Statistical Decision*. McGraw-Hill, 1970.

[53] S.C. Delipersad and A.D. Broadhurst. Face recognition using neural networks. In *COMSIG, Proc. of IEEE 1997 South African Symposium on Communications and Signal Processing*, 33-36, 9-10 Sept 1997.

[54] L. Deng, K. Hassanein, and M. Elmasry. Neural-network architecture for linear and nonlinear predictive hidden markov models: application to speech recognition. In *Proc. of the IEEE Workshop on Neural Networks for Signal Processing*, pages 411–421, 30 Sept.-1 Oct 1991.

[55] David G. T. Denison, Christopher C. Holmes, Bani K. Mallick, and Adrian F. M. Smith. *Bayesian Methods for Nonlinear Classification and Regression*, volume Wiley Series in Probability and Statistics. John Wiley & Sons, Chichester, 2002.

[56] J.E. Dennis and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Englewood Cliffs, NJ, Prentice Hall, 1983.

[57] A.P. Dhawan, Y. Chitre, and C. Kaiser-Bonasso. Analysis of mammographic microcalcifications using gray-level image structure features. *IEEE Transactions on Medical Imaging*, 15(3):246–259, 1996.

[58] J. Dougherty, R. Kohavi, and M. Sahamin. Supervised and unsupervised discretization of continuous features. In *14th IJCAI*, pages 194–202, 1995.

[59] Harris Drucker. Improving regressors using boosting techniques. In Douglas H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pages 107–115. Morgan Kaufmann, 1997.

[60] M.O. Efe, O. Kaynak, B.M. Wilamowski, and X. Yu. Radial basis function neural network in variable structure control of a class of biochemical processes. In *In IECON: the 27th Annual Conference of the IEE Industrial Electronics Society*, pages 13–18, 2001.

[61] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks-a review. *Pattern Recognition*, 35:2279–2301, 2002.

[62] I. El-Naqa, M.N. Wernick, Yongyi Yang, and N.P Galatsanos. Image retrieval based on similarity learning. In *Proc. of the International Conference on Image Processing*, pages 722–725, 2000.

[63] L.C. Ern and G. Sulong. Fingerprint classification approaches: an overview. In *Sixth International, Symposium on Signal Processing and its Applications*, volume 1, pages 347–350, 13-16 Aug. 2001.

[64] O.K Ersoy and S.W. Deng. Parallel, self-organizing, hierarchical neural networks with continuous inputs and outputs. *IEEE Trans. on Neural Networks*, 6(5):1037–1044, 1995.

[65] A.M.H. Fadzil and Lim Cheah Choon. Face recognition system based on neural networks and fuzzy logic. In *IEEE International Conference on Neural Networks*, volume 3, pages 1638–1643, 9-12 June 1997.

[66] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *13th IJCAI*, volume 2, pages 1022–1027, 1993.

[67] Bernier O. Feraud R. and Viallet J.E. A Fast and Accurate Face Detector Based on Neural Networks . *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(1):42–83, 2002.

[68] V. Ficet-Cauchard, C. Porquet, and M. Revenu. Cbr for the reuse of image processing knowledge: A recursive retrieval/adaption strategy. In *Case-Based Reasoning Research and Development*, pages 438–453. Springer Verlag, Berlin, 1999.

[69] Jean Pierre Florens, Michel Mouchart, and Jean-Marie Rolin. *Elements of Bayesian Statistics*. Marcel Dekker, New York, 1990.

[70] G. Foody. A relative evaluation of multiclass image classification by support vector machines. *IEEE Transactions on Geoscience and Remote Sensing*, 42(6):1335–1346, 2004.

[71] P. Gallinari, S. Thiria, R. Badran, and F. Fogelman-Soulie. On the relationships between discriminant analysis and multilayer perceptrons. *Neural Networks*, 4:349–360, 1991.

[72] Garcia C.and Delakis M. Convolutional Face Finder : A Neural Architecture for Fast and Robust FAce Detection. *To appear in IEEE Trans. Pattern Anal. Mach. Intell.*, 2004.

[73] T. Gartner. Exponential and geometric kernels for graphs. In *In NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data*, 2002.

[74] T. Gartner, P. A. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. In *In Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*, 2003.

[75] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*, volume Texts in Statistical Science. Chapman & Hall, London, 1997.

[76] R. Gemello, D. Albesano, F. Mana, and L. Moisa. Multi-source neural networks for speech recognition: a review of recent results. In *Proc. of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 5, pages 265–270, 24-27 July 2000.

[77] M. Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

[78] G. Giacinto, F. Roli, and G. Fumera. Unsupervised learning of neural network ensembles for image classification. In 3, editor, *International Joint conference on Neural Networks*, pages 155–162, 2000.

[79] H. Gish. A probabilistic approach to the understanding and training of neural network classifiers. In *In Proc. IEEE Int. Conf. Acoustic, Speech, Signal Processing*, pages 1361–1364, 1990.

[80] T. Gneiting. Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508, 2002.

[81] Marco Gori and Alberto Tesi. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

[82] B. Gosselin. Neural networks combination for improved handwritten characters recognition. In *Proc. Of Int. Conf. on Signal and Image Processing*, pages 144–146, Las Vegas, Nevada, 1995.

[83] B. Gosselin. Multilayer perceptrons combination applied to handwritten character recognition. *Neural Processing Letters*, 3:3–10, 1996.

[84] P. Gouin, C.L. Scofield, C. Gareyte, and H-N. Pham. Neural network segmentation and recognition of text data on engineering documents. In *IJCNN International Joint Conference on Neural Networks*, volume 3, pages 730–735, 7-11 June 1992.

[85] D. Greenhil and E.R. Davies. Relative electiveness of neural network for image noise suppression. In *Proceedings of the Pattern Recognition in Practise IV*, 367-378, 1994.

[86] M. Grimnes and A. Aamodt. A two layer case-based reasoning architecture for medical image understanding. In *Advances in Case-Based Reasoning*, pages 164–178. Springer Verlag, Berlin, 1996.

[87] S. Grossberg. *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control.* Boston, MA: Reidel Press, 1982.

[88] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987.

[89] Lin Guo and De-Shuang Huang. Human face recognition based on radial basis probabilistic neural network. In *IEEE Proc. of the International Joint Conference on Neural Networks*, volume 3, pages 2208–2211, 20-24 July 2003.

[90] M.T. Hagan, H.B. Demuth, and M.H. Beale. *Neural Network design.* Boston, MA, PWS Publishing, 1996.

[91] M.T. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Trans. On Neural Networks*, 5(6):989–993, 1994.

[92] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[93] D. Haussler. Convolution kernels on discrete structures. In *Technical Report UCS-CRL-99-10*, 1999.

[94] S. Haykin. *Neural networks a comprehensive foundation.* Prentice hall, second edition edition, 1999.

[95] R. Hecht-Nielsen. *Neurocomputing*. Addison Wesley Publishirig Company, 1987.

[96] R. Hecht-Nielsen. *Neurocomputing.* Addison/Wesley, 1989.

[97] D. Heckerman. A tutorial on learning with bayesian network. Technical report, Microsoft MSR-TR-95-06, 1995.

[98] S.C. Hg and Y.F. Huang. Learning algorithms for perceptrons using back propagation with seleptive updates. *IEEE Control Systems Magazine*, pages 56–61, 1990.

[99] Hjelmas E.and Low B. K. Face Detection : A survey. *Computer Vision and Image Understanding*, 83:236–275, 2001.

[100] Tin Kam Ho. Nearest neighbors in random subspaces. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR '98 and SPR '98, Sydney, NSW, Australia, August 11-13, 1998, Proceedings*, pages 640–648. Springer, 1998.

[101] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[102] L. Holmstrom, P. Koistinen, J. Laaksonen, and E. Oja. Neural network and statistical perspectives of classification. In *IEEE Proceeding of ICPR*, pages 286–290, 1996.

[103] A. Holt and G.L. Benwell. Applying case-based reasoning techniques in gis. *Journal of Geograpical Information Science*, 13(1):9–25, 1999.

[104] Lin-Lin Huang, A. Shimizu, Y. Hagihara, and H. Kobatake. Face detection from cluttered images using a polynomial neural network. In *Proc. of IEEE International Conference on Image Processing*, volume 2, pages 669–672, 7-10 Oct. 2001.

[105] P.A. Hughes and A.D.P. Green. The use of neural networks for fingerprint classification. In *IEEE Second International Conference on Artificial Neural Networks*, pages 79–81, 18-20 Nov. 1991.

[106] N. S. Imennov, S. L. Dockstader, and A. M. Tekalp. A robust bayesian network for articulated motion classification. In *IEEE International Conference on Image Processing*, volume 3, pages 305–308, September,14-17 2003.

[107] M. Inoue, Y. Mitsukura, M. Fukumi, and N. Akamatsu. Neural net based image retrieval by using color and location information. In *In IEEE International Conference on Systems, Man, and Cybernetics*, pages 2575 – 2579, 2000.

[108] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems*, volume 10, 1999.

[109] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *In Proceedings of the 1999 Conference on AI and Statistics*, 1999.

[110] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, pages 295–307, 1988.

[111] J. Jarmulak. Case-based classification of ultrasonic b-scans: Case-base organisation and case retrieval. In *Advances in Case-Based Reasoning*, pages 100–111. Springer Verlag, Berlin, 1998.

[112] F. V. Jensen. *An Introduction to Bayesian Network*. UCL Press Limited, 1996.

[113] A.L.H. Jin, A. Chekima, J.A. Dargham, and Liau Chung Fan. Fingerprint identification and recognition using backpropagation neural network. In *SCOReD, Student Conference on Research and Development*, pages 98–101, 16-17 July 2002.

[114] T. Joachims. A statistical learning model of text classification with support vector machines. In *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR), ACM*, 2001.

[115] T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer Academic Publishers, 2002.

[116] M. I. Jordan. Graphical models. *Statistical Sciences (Special issue on Bayesian Statistics)*, 19:140–155, 2004.

[117] M. I. Jordan and Y. Weiss. Graphical models: Probabilistic inference. *In The Handbook of Brain Theory and Neural Networks*, 2002.

[118] M. Kamijo. Classifying fingerprint images using neural network: Deriving the classification state. In *IEEE International Conference on Neural Network*, volume 3, pages 932–1937, 1993.

[119] H. Kashima and A. Inokuchi. Kernels for graph classification. In *In ICDM Workshop on Active Mining*, 2002.

[120] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *In Proceedings of the 20th International Conference on Machine Learning*, 2003.

[121] S.S. Keerthi and C.J. Lin. Asymptotic behaviours of support vector machines with gaussian kernel. *Neural Computation*, 15, 2003.

[122] R. Kerber. Chimerge: Discretization of numeric attributes. In *AAAI 92*, pages 123–128, 1992.

[123] T.K. Kim, S.U. Lee, J.H. Lee, S.C. Kee, and S.R. Kim. Integrated approach of multiple face detection for video surveillance. In *ICPR02*, pages II: 394–397, 2002.

[124] R. Kohavi, B. Becker, and D. Sommerfield. Improving simple bayes. In *Proceedings of the European Conference on Machine Learning (ECML-87)*, pages 78–97, 1997.

[125] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 9, pages 1464–1480, 1990.

[126] T. Kohoonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks bench marking studies. In *IEEE Annual Int. I. Conf. on Neural Networks, San Diego*, 1988.

[127] R. Kondor and T. Jebara. A kernel between sets of vectors. In *Proceedings of the ICML*, 2003.

[128] R. I. Kondor and J. Laerty. Diffusion kernels on graphs and other discrete input space. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[129] H. Kong and L. Guan. Self-organizing tree map for eliminating impulse noise with random intensity distributions. *Journal of Electronic Imaging*, 1(7):36–44, 1998.

[130] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pages 231–238. MIT Press, 1994.

[131] P. Kumar, S. Ranganath, and H. Weimin. Bayesian network based computer vision algorithm for traffic monitoring using video. In *IEEE Intelligent Transportation Systems*, volume 1, pages 897–902, 2003.

[132] S. Kung and J. Hwang. Neural networks for intelligence multimedia processing. *Proceeding of the IEEE*, 86(6):485–501, 1998.

[133] S. Kuusisto. *Application of the PMDL Principle to the Induction of Classifi-cation Trees*. PhD thesis, Tampere Finland, 1998.

[134] Portinale L., Torasso P., and Tavano P. Speed-up, quality, and competence in multimodal reasoning. In *Case-Based Reasoning Research and Development*, pages 303–317. Springer Verlag, Berlin, 1999.

[135] A. Shimizu L. Huang and F. Kobatake. Face detection using a modified radial basis function neural network. In *Proc. of IEEE 16th International Conference on Pattern Recognition*, volume 2, pages 342–345, 11-15 Aug. 2002.

[136] Ah Chung T. Lawrence S., Lee Giles C. and Back A. D. Face Recognition : A convolutional Neural Network Approach. *IEEE Trans. on Neural Networks, Special Issue on Neural Networks and Pattern Recognition*, 8(1):98–113, 1997.

[137] D.x. Le, G.R. Thoma, and H. Wechsler. Document classification using connectionist models. In *IEEE International Conference on Neural Networks*, volume 5, pages 3009–3014, 27 June-2 July 1994.

[138] Lecun Y. Efficient BackProp. *Orr, G. and Müller, K. Springer, "Neural Networks: Tricks of the Trade,"*, 1998.

[139] D. Lee and H.S. Seung. A neural network based head tracking system. *Advances in Neural Information Processing Systems*, 10:908–14, 1998.

[140] H.K. Lee and S.I.Yoo. A neural network-based image retrieval using nonlinear combination of heterogeneous features. In *Proc. of the 2000 Congress on Evolutionary Computation*, volume 1, pages 667–674, 2000.

[141] Peter M. Lee. *Bayesian Statistics*. Adler, London, 1997.

[142] Lawrence S. Lee Giles C.Chung Tsoi A. and Back A. D. Face recognition : A Convolutional Neural Network Approach. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition*, 8:98–113, 1997.

[143] F. Leitao. A study of string dissimilarity measures in structural clustering. In *Advances in Pattern Recognition*, pages 385–394. Springer Verlag, Berlin, 1999.

[144] Mario Lenz and Hans-Dieter Burkhard. Case retrieval nets: Basic ideas and extensions. In *KI - Kunstliche Intelligenz*, pages 227–239, 1996.

[145] C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: A string kernel for svm protein classification. In *In Proceedings of the Pacific Symposium on Biocomputing*, 2002.

[146] W.F. Leung, S.H. Leung, W.H. Lau, and A. Luk. Fingerprint recognition using neural network. In *Proc. of the IEEE Workshop on Neural Networks for Signal Processing*, pages 226–235, 30 Sept.-1 Oct. 1991.

[147] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.

[148] Y. Li, S.G. Gong, J. Sherrah, and H. Liddell. Support vector machine based multi-view face detection and recognition. *Image and Vision Computing*, 22(5):413–427, May 2004.

[149] G. S. Lim, M. Alder, and P. Hadingham. Adaptive quadratic neural nets. *Pattern Recognition Letters*, 3:325–329, 1992.

[150] J. S. Lin, P. A. Ligomenides, Y. M. F. Lure, M. T. Freedman, and S. K. Mun. Aplication of neural networks for improvement of lung nodule detection in radiographic images. In *S/CAR'92, Proc. Symp. Comput. Assist. Radiol ()*, pages 108–115, 1992.

[151] J. S. Lin, S. B. Lo, A. Hasegawa, M. T. Freedman, and S. K. Mun. Reduction of false positives in lung nodule detection using a two-level neural classification. *IEEE Trans. on Medical Imaging*, 15:206–216, 1996.

[152] D. V. Lindley. *Introduction to Probability and Statistics from a Bayesian Viewpoint*. University Press, Cambridge, 1965.

[153] P.J.G. Lisboa. A review of evidence of health benefit from artificial neural networks in medical intervention. *Neural Networks*, 15(1):11–39, 2002.

[154] S. B. Lo, S. L. Lou, J. S. Lin, M. T. Freedman, and S. K. Mun. Artificial convolution neural network techniques and applications for lung nodule detection. *IEEE Trans. on Medical Imaging*, 14:711–718, 1995.

[155] H. Lodhi, J. Shawe-Taylor, N. Christianini, and C. Watkins. Text classification using string kernels. In *Advances in Neural Information Processing Systems*, 2001.

[156] R.E.A. Lucht, M.V. Knopp, and G. Brix. Classification of signal-time curves from dynamic mr mammography by neural networks. *Magnetic Resonance Imaging*, 19:51–57, 2001.

[157] W.P.J. Mackeown, P. Greenway, B.T. Thomas, and W.A. Wright. Labelling images with a neural network. In *Third International Conference on Artificial Neural Networks*, pages 16–20, 25-27 May 1993.

[158] N. Mani and B. Srinivasan. Application of artificial neural network model for optical character recognition. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2517–2520, 12-15 Oct 1997.

[159] D.W. Massaro, J. Besko, M.M. Cohen, C. Fry, and T. Rodriguez. Picture my voice: Audio to visual speech synthesis using artificial neural networks. In *AVSP'99*, 133-138, 1999.

[160] S. Matej and R.M. Lewitt. Practical considerations for 3-d image reconstruction using spherically symmetric volume elements. *IEEE Trans. On Medical Imaging*, 15(1):68–78, 1996.

[161] M. McCloskey and N.J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, 24:109–165, 1989.

[162] W. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[163] B. Messmer and H. Bunke. Efficient subgraph isomorphism detection: A decomposition approach. *IEEE Trans. on Knowledge and Data Engineering*, 12(2):307–323, 2000.

[164] A. Micarelli, A. Neri, and G. Sansonetti. A case-based approach to image recognition. In *Advances in Case-Based Reasoning*, pages 443–454. Springer Verlag, Berlin, 2000.

[165] Donald Michie, David J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[166] M. Minsky and S. Papert. *Perceptrons: An introduction to computational geometry.* MIT Press, Cambridge, MA, 1969.

[167] Moghadda, Jebra, and Pentland. Efficient map/ml similarity matching for visual recognition. In *Internatinal Confernce on Pattern Recognition*, pages 876–881. IEEE Computer Society Press, 1998.

[168] J. Moody. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1996.

[169] J. Moody and C. J. Darken. Fast learning networks of locally-tuned processing units. *Neural Computation*, 2(1):281–294, 1989.

[170] B. W. Morgan. *An Introduction to Bayesian Statistical Desicion Processes*. Prentice-Hall, Englewood Cliffs, 1968.

[171] S. Morishima and H. Harashima. A media conversion from speech to facial image for intelligent man-machine interface. *IEEE Journal on Selected Areas in Communications*, 9(4):594–600, 1991.

[172] P. Muneesawang and L. Guan. A neural network approach for learning image similarity in adaptive cbir. In *IEEE Fourth Workshop on Multimedia Signal Processing*, pages 257–262, 2001.

[173] J.M.J. Murre, R.H. Phaf, and G. Wolters. Calm: Categorizing and learning module. *Neural Networks*, 5:55–82, 1992.

[174] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Faris, and D.M. Hummels. On the training of radial basis function classifiers. *Neural Networks*, 5:595–603, 1992.

[175] M. R. Naphade and T. S. Huang. A probabilitic framework for semantic video indexing, filtering, and retrieval. *IEEE Transactions on Multimedia*, 3(1), March 2001.

[176] R.M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

[177] T. Niblett. Constructing decision trees in noisy domains. In *2nd European Working Session on Learning*, pages 67–78, Bled, Yogoslavia, 1987.

[178] T. Niblett and I. Bratko. Construction decision trees in noisy domains. In *Progress in Machine Learning*, pages 67–78. Sigma Press, England, 1987.

[179] M. Niranjan and F. Fallside. Neural networks and radial basis function in classifying static speech patterns. *Computer Speech and Language*, 4:275–289, 1990.

[180] Naoko Nitta. *Semantic Content Analysis of Broadcasted Sport Videos with InterModal Collaboration*. PhD thesis, Department of Informatics and Mathematical Science, Osaka University, Osaka, Japan, January 2003.

[181] S.I. Nitzan and J. Paroush. *Collective Decision Making*. Cambridge University Press, 1985.

[182] Cheng Soon Ong, Xavier Mary, Stephane Canu, and Alexander Smola. Learning with non positive kernels. In *In proceedings of the ICML*, 2004.

[183] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 130. IEEE Computer Society, 1997.

[184] M. Ozkan, B.M. Dawant, and R.J. Maciunas. Neural-network-based segmentazion of multi-modal medical images: a comparative and prospective study. *IEEE Trans. On Medical Imaging*, 12(3):534–544, 1993.

[185] M. Ozkan, H.G. Sprenkels, and B.M. Dawant. Multispectral magnetic resonance image segmentation using neural networks. In *IJCNN International Joint Conference on Neural Networks*, volume 1, pages 429–434, 17-21 June 1990.

[186] G. Paass, E. Leopold, M. Larson, J. Kindermann, , and S. Eickeler. Svm classification using sequences of phonemes and syllables. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, 2002.

[187] J. Park and I. W. Sandberg. Universal approximation using radial-basis function networks. *Neural Computation*, 3:246–257, 1991.

[188] D.W. Patterson. *Artificial Neuronal Networks. Theory and Application.* Prentice hall, 1996.

[189] C.S. Pattichis and A.G. Constantinides. Medical imaging with neural networks. In *Proc. of the 1994 In IEEE Workshop on Neural Networks for Signal Processing*, pages 431–440, 6-8 Sept 1994.

[190] M.G. Penedo, M.J. Carreira, A. Mosquera, and D. Cabello. Computer-aided diagnosis: a neural-network based approach to lung nodule detection. *IEEE Trans. on Medical Imaging*, 17(6):872–880, 1998.

[191] P. Perner. *How to use Repertory Grid for Knowledge Acquisition in Image Interpretation*. PhD thesis, HTWK Leipzig, 1994.

[192] P. Perner. Content-based image indexing and retrieval in a image database for technical domains. In *Multimedia Information Analysis and Retrieval*, pages 207–224. Springer Verlag, Berlin, 1998.

[193] P. Perner. Different learning strategies in a case-based reasoning system for image interpretation. In *Advances in Case-Based Reasoning*, pages 251–261. Springer Verlag, Berlin, 1998.

[194] P. Perner. Using cbr learning for the low-level and high-level unit of a image interpretation system. In *Advances in Pattern Recognition*, pages 45–54. Springer Verlag, Berlin, 1998.

[195] P. Perner. An architecture for a cbr image segmentation system. *Journal of Engineering Application in Artificial Intelligence*, 12(6):749–759, 1999.

[196] P. Perner. Are case-based reasoning and dissimilarity-based classification two sides of the same coin? *Journal of Engineering Applications of Artificial Intelligence*, 15(3):205–216, 2001.

[197] P. Perner. *Data Mining on Multimedia Data*. lnai 2558. Springer Verlag, Berlin, 2002.

[198] P. Perner. Incremental learning of retrieval knowledge in a case-based reasoning system. In *Case-Based Reasoning - Research and Development*, lnai 2689, pages 422–436. Springer Verlag, Berlin, 2003.

[199] P. Perner. Case-based reasoning for image analysis and interpretation. In *Handbook of pattern Recognition and Computer Vision, 2nd Edition*. World Scientific Ltd., 2004.

[200] P. Perner and A. Bühring. Case-based object recognition. In *ECCBR 2004*, Berlin, 2004. Springer Verlag.

[201] P. Perner, Th. Guenther, and H. Perner. Airborne fungi identification by case-based reasoning. In *5th ICCBR; Workshop Case-Based Reasoning in Health Science*, pages 73–79, Berlin, 2003. Springer Verlag.

[202] P. Perner and S. Jänichen. Learning of form models from exemplars. In *SSPR 2004*, Berlin, 2004. Springer Verlag.

[203] P. Perner, H. Perner, and B. Müller. Similarity guided learning of the case description and improvement of the system performance in an image classification system. In *Advances in Case-Based Reasoning*, lnai 2416, pages 604–612. Springer Verlag, Berlin, 2002.

[204] P. Perner and S. Trautzsch. Multinterval discretization for decision tree learning. In A. Amin, D. Dori, P. Pudil, and H. Freeman, editors, *Advances in Pattern Recognition*, LNCS 1451, pages 475–482. Springer, Heidelberg, 1998.

[205] P. Perner, U. Zscherpel, and C. Jacobsen. Comparison between neural networks and decision trees application of machine learning in industrial radiographic testing. *Pattern Recognition Letters*, 22(1):47–54, 2000.

[206] L. Piccoli, A. Dahmer, J. Scharcanski, and P.O.A. Navaux. Fetal echocardiographic image segmentation using neural networks. In *7-th IEEE International Conference on Image Processing and Its Applications*, volume 2, pages 507–511, 13-15 July 1999.

[207] Vlad Popovici and Jean-Philippe Thiran. Face detection using svm trained in eigenfaces space. In *4th International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 190–198, Berlin, 2003. Springer-Verlag.

[208] S. James Press. *Bayesian Statistics: Principles, Models and Applications*, volume Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1989.

[209] R.H. Pugmire, R.M. Hodgson, and R.I. Chaplin. The properties and training of a neural network based universal window filter developed for image processing tasks. In S. Amari and N. Kasaboy Eds, editors, *Brain-like computing and intelligent information systems*, pages 49–77. Sprinter-Verlang, 1998.

[210] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[211] J.R. Quinlan. Simplifying decision trees. *Machine Learning*, 27:221–234, 1987.

[212] J.R. Quinlan. Decision trees and multivalued attributes. In *Machine Intelligence 11*. Oxford University Press, 1988.

[213] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[214] Tsuhan Chen; R.R Rao. Audio-visual integration in multimodal communication. *Proc. of the IEEE*, 86(5):837–852, 1998.

[215] R.Battiti. First and second order methods for learning: between steepest descent and newton's method. *Neural Computation*, 4(2):141–166, 1992.

[216] W.E. Reddick, J.O. Glass, E.N. Cook, T.D. Elkin, and R.J. Deaton. Automated segmentation and classification of multispectral magnetic resonance images of brain using artificial neural networks. *IEEE Trans. on Medical Imaging*, 16:911–918, 1997.

[217] M. D. Richard and R. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computing*, 3:461–483, 1991.

[218] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. *Proc. of the IEEE International Conference on Neural Networks (ICNN)*, pages 586–591, 1983.

[219] S.A. Rizvi, L.C. Wang, and N.M. Nasrabadi. Non linear vector prediction using feedforward neural networks. *IEEE Trans. Image Processing*, 6(10):1431–1436, 1997.

[220] Christian P. Robert. *The Bayesian Choice*. Springer Texts in Statistics, New York, 1994.

[221] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[222] Yang M. H. Roth D. and Ahuja N. A SNoW-based Face Detector. *Advances in Neural Information Processing Systems 12 (NIPS 12);, MIT Press*, pages 855–861, 2000.

[223] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[224] Baluja S. Rowley H. A. and Kanade T. Neural Network-based Face Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):23–38, 1998.

[225] D. Rumelhart, G. Hinton, and R. Williams. *Learning internal representations by error propagation.*, volume 1. MIT Press, Cambridge, MA, 1986.

[226] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[227] A. R.Webb and D. Lowe. The optimized internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks*, 3(4):367–375, 1990.

[228] Boughorbel S., Tarel J.-P., Fleuret F., and Boujemaa N. Gcs kernel for svm object recognition. In *INRIA Technical Report*, 2004.

[229] Schmitt S., Jerusalem D., and Nett T. Representation and execution of questioning strategies for the acquisition of customer requirements. In *Eighth German Workshop on Case-Based Reasoning*, pages 23–37. DaimlerChrysler Research and Technology FT3/KL, 2000.

[230] P. Sajda, C. Spence, and J. Pearson. A hierarchical neural network architecture that learns target context: applications to digital mammography. In *IEEE Proceedings of the 1995 International Conference on Image Processing*, volume 3, pages 3149–3151, Los Alamitos, CA, USA, 1995.

[231] R.M. Sanner and J.J.E. Slotine. Gaussian networks for direct adaptive control. *IEEE Trans. On Neural Networks*, 3(6):837–863, 1994.

[232] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[233] B. Le Saux. *Classification non exclusive et personnalisation par apprentissage : Application à la navigation dans les bases d'images*. PhD thesis, INRIA, 2003.

[234] Robert E. Schapire. A brief introduction to boosting. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 1401–1406. Morgan Kaufmann, 1999.

[235] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[236] B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[237] Bernhard Scholkopf. The kernel trick for distances. In *NIPS*, pages 301–307, 2000.

[238] J. Schurmann. *Pattern Classification: A Unified View of Statistical Pattern Recognition and Neural Networks*. Wiley Interscience, 1996.

[239] S.B. Serpico, L. Buzzone, and F. Roli. An experimental comparison of neural and statistical non-parametric algorithms for supervised classification of remote- sensing images. *Patter Recognition Letters*, 17:1331–1341, 1997.

[240] H.-C. Shih and C.-L. Huang. Image analysis and interpretation for semantics categorization in baseball video. In *IEEE Interantional Conference on Information Technology: Computers and Communications*, 2003.

[241] P. A. Shoemaker. A note on least-squares learning procedures and classification by neural network models. *IEEE Trans. Neural Networks,*, 2:158–160, 1991.

[242] Steinkraus D. Simard P. Y. and Platt J. C. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *International Conference on Document Analysis and Recognition*, pages 958–962, 2003.

[243] Delany S.J. and Cunningham P. An analysis of case-base editing in a spam filtering system. In *7th European Conference on Case-Based Reasoning*. Springer Verlag, 2004.

[244] J. Sklansky, Tao EY, M. Bazargan, CJ Ornes, RC. Murchison, and S. Teklehaimanot. Computer-aided, case-based diagnosis of mammographic regions of interest containing microcalcifications. *Acad. Radiol.*, 7:395–405, 2000.

[245] B. Smyth and E. McKenna. Modeling the competence of case-bases. In *Advances in Case-Based Reasoning*, pages 208–220. Springer Verlag, Berlin, 1998.

[246] J. Surma and J. Tyburcy. A study on competence-preserving case replacing strategies in case-based reasoning. In *Advances in Case-Based Reasoning*, lnai 1488, pages 233–238. Springer Verlag, Berlin, 1998.

[247] Ti-QiongXu, Bi-Cheng Li, and Bo Wang. Face detection and recognition using neural network and hidden markov models. In *Proc. of the International Conference on Neural Networks*, volume 1, pages 228–231, 14-17 Dec. 2003.

[248] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *ACM Multimedia*, 2001.

[249] Simon Tong. *Active Learning: Theory and Applications*. PhD thesis, Stanford University, 2001.

[250] T.Poston, C. Lee, Y. Choie, and Y. Kwon. Local minima and backpropagation. In *In Proc. Int. Joint Conference on Neural Networks, Seattle, WA*, pages 173–176, 1991.

[251] K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. In *Bioinformatics*, 2002.

[252] Turk M.and Pentland A. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[253] A. van der Heiden and A. Vossepoel. A landmark-based approach of shape dissimilarity. In *Internatinal Confernce on Pattern Recognition*, pages 120–124. IEEE Computer Society Press, 1999.

[254] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[255] R.C. Veltkamp. Content-based image retrieval system: A survey. Technical report, University of Utrecht, 2002.

[256] K. Veropoulos. Controlling the sensivity of support vector machines. In *International Joint Conference on Artificial Intelligence (IJCAI99)*, Stockholm, Sweden, 1999.

[257] J.-P. Vert. A tree kernel to analyze phylogenetic profiles. *Bioinformatics*, 2002.

[258] J.-P. Vert and M. Kanehisa. Graph driven features extraction from microarray data using diffusion kernels and kernel cca. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

[259] A. Voss. Similarity concepts and retrieval methods. *Fabel Report*, 13, 1993.

[260] E. Wan. Neural network classification: A bayesian interpretation. *IEEE Trans. Neural Networks,*, 4(1):303–305, 1990.

[261] L. Wang. Image retrieval with svm active learning embedding euclidian search. In *IEEE International Conference on Image Processing*, Barcelona, September 2003.

[262] L.C. Wang, S.A. Rizvi, and N.M. Nasrabadi. A modular neural network vector predictor for predictive image coding. *IEEE Trans. Image Processing*, 7(8):1198–1217, 1998.

[263] R. Wang and Z. Chi. Automatic segmentation of chinese chunks using a neural network. In *Proc. of the 2003 International Conference on Neural Networks and Signal Processing*, volume 1, pages 96–99, 14-17 Dec 2003.

[264] Y. Wang, T. Tan, and K.-F. Loe. Video segmentation based on graphical model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[265] C. Watkins. Kernels from matching operations. In *Technical report*, 1999.

[266] Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models*, volume Springer Series in Statistics. Springer, New York, 1997.

[267] D. Wettscherek, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artifical Intelligence Review*, 1995.

[268] B. Widrow and M.E. Hoff. Adaptive switching circuits. *In IRE WESCON Convention Record*, 78:96–104, 1960.

[269] Bernard Widrow and M. A. Lehr. 30 years of adaptive neural networks: Perceptron, madaline and back-propagation. In *Proc. of the IEEE*, pages 1415–1442, 1990.

[270] L. Wolf and A. Shashua. Kernel principal angles for classification machines with applications to image sequence interpretation. In *CVPR*, 2003.

[271] Y. Xirouhakis, Y. Avrithis, and S. Kollias. Image retrieval and classification using affine invariant b-spline representation and neural networks. In *IEE Colloquium on Neural Networks in Interactive Multimedia Systems*, pages 4/1 – 4/4, 1998.

[272] K. Yamaguchi. A neural network controlled adaptive search strategy for hmm-based speech recognition. In *ICASSP-93, IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 582–585, 27-30 April 1993.

[273] Ahuja N. Yang M.H. and Kriegman D. Detecting Faces in Images : A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 84:264–284, 2001.

[274] X. Yu. Can backpropagation error surface not have local minima? *IEEE Trans. On Neural Networks*, 3(6):1019–1020, 1992.

[275] P. Zamperoni and V. Starovoitov. How dissimilar are two gray-scale images? In *17. DAGM Symposium*, pages 448–455, Berlin, 1995. Springer Verlag.

[276] C. Zhang, S-C Chen, and M-L Shyu. Multiple object retrieval for image databases using multiple instance learning and relevance feedback. In *IEEE International Conference on Multimedia and Expo (ICME 2004) Taipei, Taiwan, R.O.C.*, 2004.

[277] G. P. Zhang. Neural networks for classification: A survey. *IEEE Transaction Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 30(4), 2000.

[278] Q. Zhang, S. A. Goldman, W. Yu, and J. Fritts. Content based image retrieval using multiple-instance learning. In *Proc. of the 19th Intl. Conf. on Machine Learning*, 2002.

[279] Baoyu Zheng, Wei Qian, and L.P Clarke. Digital mammography: mixed feature neural network with spectral entropy decision for detection of microcalcifications. *IEEE Transactions on Medical Imaging*, 15(5):589–597, 1996.

[280] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

[281] A. Zien, G. Ratsch, S. Mika, B. Scholkopf, T. Lengauer, and K.-R. Muller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16:799–807, 2000.

Chapter 4

Unsupervised Learning

4.1 Unsupervised Clustering

Clustering (or *cluster analysis*) aims to organize a collection of data items into clusters, such that items within a cluster are more "similar" to each other than they are to items in the other clusters. This notion of similarity can be expressed in very different ways, according to the purpose of the study, to domain-specific assumptions and to prior knowledge of the problem.

Clustering is usually performed when no information is available concerning the membership of data items to predefined classes. For this reason, clustering is traditionally seen as part of unsupervised learning. We nevertheless speak here of *unsupervised* clustering to distinguish it from a more recent and less common approach that makes use of a small amount of supervision to "guide" or "adjust" clustering (see section 5.1).

To support the extensive use of clustering in computer vision, pattern recognition, information retrieval, data mining, etc., very many different methods were developed in several communities. Detailed surveys of this domain can be found in [67], [77] or [68]. In the following, we attempt to briefly review a few core concepts of cluster analysis and describe categories of clustering methods that are best represented in the literature. We also take this opportunity to provide some pointers to more recent work on clustering.

4.1.1 A Typology of Methods

We start by mentioning some criteria that provide significant distinctions between clustering methods and can help selecting appropriate candidate methods for one's problem:

- *Objective of clustering.* Many methods aim at finding a single *partition* of the collection of items into clusters. However, obtaining a *hierarchy* of clusters can provide more flexibility and other methods rather focus on this. A partition of the data can be obtained from a hierarchy by cutting the tree of clusters at some level.

- *Nature of the data items.* Most clustering methods were developed for numerical data, but some can deal with categorical data or with both numerical and categorical data.

- *Nature of the available information.* Many methods rely on rich representations of the data (e.g. vectorial) that let one define prototypes, data distributions, multidimensional

intervals, etc., beside computing (dis)similarities. Other methods only require the evaluation of pairwise (dis)similarities between data items; while imposing less restrictions on the data, these methods usually have a higher computational complexity.

- *Nature of the clusters*. The degree of membership of a data item to a cluster is either in $[0,1]$ if the clusters are *fuzzy* or in $\{0,1\}$ if the clusters are *crisp*. For fuzzy clusters, data items can belong to some degree to several clusters that don't have hierarchical relations with each other. This distinction between fuzzy and crisp can concern both the clustering mechanisms and their results. Crisp clusters can always be obtained from fuzzy clusters.

- *Clustering criterion*. Clusters can be seen either as distant *compact* sets or as *dense* sets separated by low density regions. Unlike density, compactness usually has strong implications on the shape of the clusters, so methods that focus on compactness should be distinguished from methods that focus on the density.

Several taxonomies of clustering methods were suggested in [36], [77] or [68]. But given the high number and the strong diversity of the existing clustering methods, it is probably impossible to obtain a categorization that is both meaningful and complete. By focusing on some of the discriminating criteria just mentioned we put forward the simplified taxonomy shown below, inspired by the one suggested in [68].

- *Partitional clustering* aims to directly obtain a single *partition* of the collection of items into clusters. Many of these methods are based on the iterative optimization of a criterion function reflecting the "agreement" between the data and the partition. Here are some important categories of partitional clustering methods:

  - *Methods using the squared error* rely on the possibility to represent each cluster by a prototype and attempt to minimize a cost function that is the sum over all the data items of the squared distance between the item and the prototype of the cluster it is assigned to. In general, the prototypes are the cluster centroids, as in the popular k-means algorithm [107]. Several solutions were put forward for cases where a centroid cannot be defined, such as the k-medoid method [77], where the prototype of a cluster is an item that is "central" to the cluster, or the k-modes method [63] that is an extension to categorical data.

    By employing the squared error criterion with a Minkowski metric or a Mahalanobis metric, one makes the implicit assumption that clusters have elliptic shape. The use of multiple prototypes for each cluster or of more sophisticated distance measures (with respect to one or several cluster models, see e.g. [24]) can remove this restriction.

    Fuzzy versions of methods based on the squared error were defined, beginning with the Fuzzy C-Means [13]. When compared to their crisp counterparts, fuzzy methods are more successful in avoiding local minima of the cost function and can model situations where clusters actually overlap. To make the results of clustering less sensitive to outliers (isolated data items) several fuzzy solutions were put forward, based on robust statistics [45] or on the use of a "noise cluster" [25], [94].

Many early methods assumed that the number of clusters was known prior to clustering; since this is rarely the case, techniques for finding an "appropriate" number of clusters had to be devised. This is an important issue for partitional clustering in general. For methods based on the squared error, the problem is partly solved by adding a *regularization* term to the cost function. This is the case, for example, for the competitive agglomeration method introduced in [44], where clusters compete for membership of data items and the number of clusters is progressively reduced until an optimum is reached. With such solutions, instead of the number of clusters one has to control a regularization parameter, which is often more convenient. Another solution is to use a cluster validity index (see section 4.7) to select *a posteriori* the appropriate number of clusters.

– *Density-based methods* consider that clusters are dense sets of data items separated by less dense regions; clusters may have arbitrary shape and data items can be arbitrarily distributed. Many methods, such as DBSCAN [35] (further improved in [130]), rely on the study of the density of items in the neighbourhood of each item. Some interesting recent work on density-based clustering is using 1-class support vector machines [9].

  One can consider within the category of density-based methods the *grid-based* solutions, such as DenClue [61] or CLIQUE [1], mostly developed for spatial data mining. These methods quantize the space of the data items into a finite number of cells and only retain for further processing the cells having a high density of items; isolated data items are thus ignored. Quantization steps and density thresholds are common parameters for these methods.

  Many of the *graph-theoretic* clustering methods are also related to density-based clustering. The data items are represented as nodes in a graph and the dissimilarity between two items is the "length" of the edge between the corresponding nodes. In several methods, a cluster is a subgraph that remains connected after the removal of the longest edges of the graph [67]; for example, in [143] the minimal spanning tree of the original graph is built and then the longest edges are deleted. However, some other graph-theoretic methods rely on the extraction of *cliques* and are then more related to squared error methods. Based on graph-theoretic clustering, there has been significant interest recently in *spectral* clustering using kernel methods [113].

– *Mixture-resolving* methods assume that the data items in a cluster are drawn from one of several distributions (usually Gaussian) and attempt to estimate the parameters of all these distributions. The introduction of the expectation maximization (EM) algorithm in [28] was an important step in solving the parameter estimation problem. Mixture-resolving methods have a high computational complexity and make rather strong assumptions regarding the distribution of the data. The choice of the number of clusters for these methods is thoroughly studied in more recent work such as [5] or [17]. In some cases a model for the noise is explicitely considered.

  Most mixture-resolving methods view each cluster as a single simple distribution and thus strongly constrain the shape of the clusters; this explains why we did not include these methods in the category of density-based clustering.

- *Hierarchical clustering* aims to obtain a hierarchy of clusters, called *dendrogram*, that shows how the clusters are related to each other. These methods proceed either by iteratively merging small clusters into larger ones (agglomerative algorithms, by far the most common) or by splitting large clusters (divisive algorithms). A partition of the data items can be obtained by cutting the dendrogram at a desired level.

  Agglomerative algorithms need criteria for merging small clusters into larger ones. Most of the criteria concern the merging of pairs of clusters (thus producing binary trees) and are variants of the classical single-link [133], complete-link [81] or minimum-variance [139], [111] criteria. The use of the single-link criterion can be related to density-based methods but often produces upsetting effects: clusters that are "linked" by a "line" of items cannot be separated or most items are individually merged to one (or a few) cluster(s). The use of the complete-link or of the minimum-variance criterion relates more to squared error methods.

  Many recent hierarchical methods focus on the use of density-like information and don't constrain the shape of the clusters. They often reflect interest in the database community for dealing with huge datasets and for speeding-up access. CURE [57] employs multiple representatives per cluster in order to obtain clusters of arbitrary shape while avoiding the problems of the single-link criterion. OPTICS [3] does not build an explicit clustering of the collection of items, but rather an ordered representation of the data that reflects its clustering structure.

4.2 Hierarchical Clustering

As outlined above, the basic idea with hierarchical clustering is to organise the objects to be clustered into a tree structure or hierarchy called a dendograph (see 4.1). The process starts out with each object in a cluster of its own and these clusters are connected, selecting the most similar clusters to connect at each stage, until all clusters are linked. So the result is not a set of clusters but a hierarchical structure with the distance between objects reflected by the distance between them in the hierarchy. Another view on the hierarchy is that it describes several alternative clusterings of different levels of *severity*. In Fig. 4.1 the *cut* $c1$ describes the clustering $\{\{AB\}\{C\}\{D\}\{E\}\{FG\}\}$ which is more *strict* than the clustering described by $c2$, $\{\{ABC\}\{D\}\{EFG\}\}$, in the sense that objects need to be more similar before being clustered together.

The basic version of the hierarchical clustering algorithm as presented by Han and Kamber [59] is as follows. Given a set of N items or objects to be clustered, and an $N \times N$ distance (or similarity) matrix, the basic process of hierarchical clustering is this:

1. Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters equal the distances (similarities) between the items they contain.

2. Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.

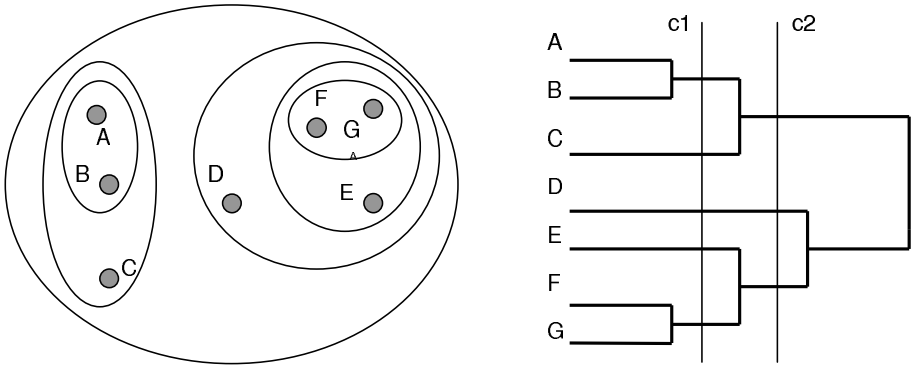


Figure 4.1: An example of 7 object that are organised into a hierarchical cluster, on the left a set-based version of this hierarchy is shown while the dendograph is shown on the right.

3. Compute distances (similarities) between the new cluster and each of the old clusters.

4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

The behaviour of this algorithm depends on the definition of the similarity between clusters. We can think of this as the *linkage* between clusters and there are three basic ways in which this can be measured.

- *single-link clustering*: We consider the distance between one cluster and another cluster to be equal to the shortest distance between any two objects from each cluster.
- *complete-link clustering*: In this case we consider the longest distance between objects in the two clusters.
- *average-link clustering*: In this case the average distance between objects in the two clusters is considered.

The appropriate choice of link criterion depends on the nature of the data and clusters that might exist. Single-link clustering will allow the development of *stringy* clusters, whereas the other two criteria require more compact structures.

4.3 k-Means and Variants

k-means is a *partitional* approach to clustering [99]. Heuristics are used to refine the clustering in order to find a good quality solution. The heuristic is to calculate the centroids of the clusters at each iteration and reallocate samples to clusters with closer centroids. For a given k, the *k*-means algorithm is implemented in 4 steps:

1. Partition objects into k non-empty subsets.

2. Compute the centroids of the clusters of the current partition. The centroid is the centre (mean point) of the cluster.

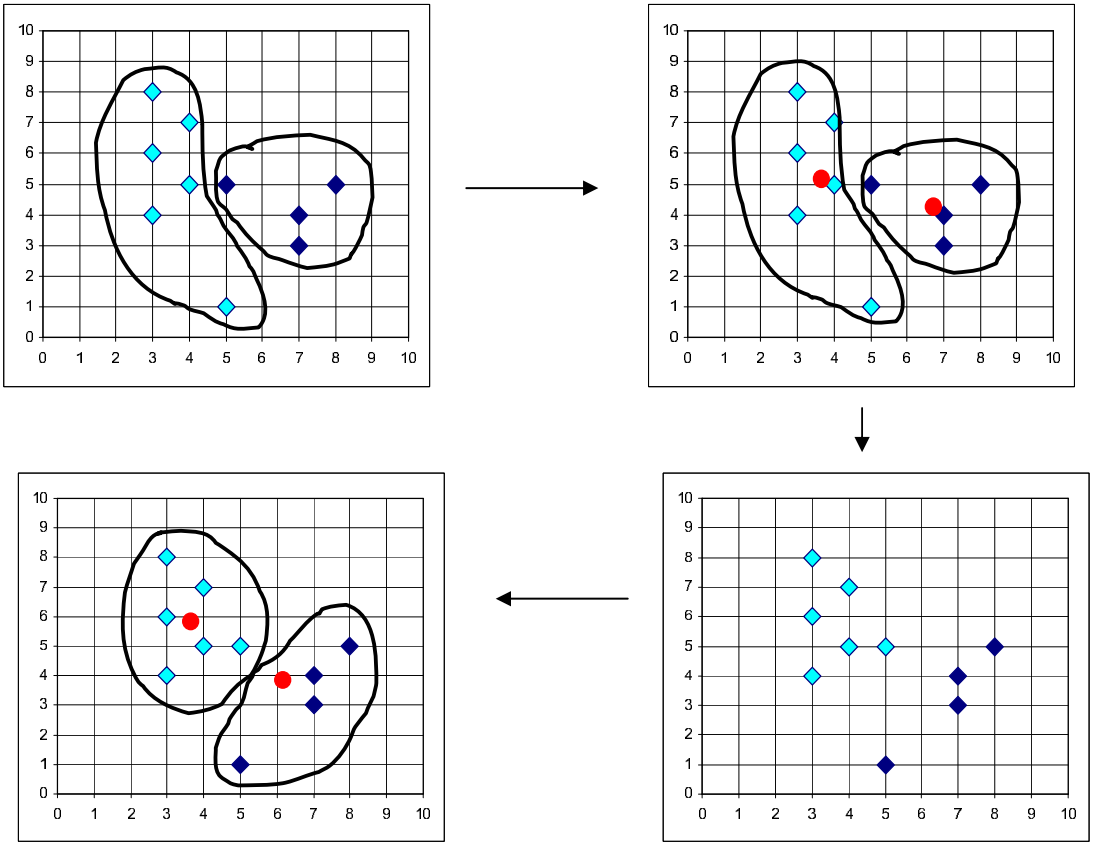


Figure 4.2: A simple example of k-means clustering in action

3. Assign each object to the cluster with the nearest centroid.

4. Go back to Step 2, stop when no reassignment occurs on a pass.

A simple example with $n = 10$ and $k = 2$ is shown in Figure 4.2. The initial clustering is shown in the top left-hand corner. The centroids of these clusters are shown in the second frame; then two samples are reallocated to clusters with closer centroids in the next frame. The centroids for these new clusters are shown in the fourth frame and the algorithm stops at this point as all samples are associated with their closest centroid.

The main strengths of *k*-means are that it is easy to implement and it is relatively efficient: $O(tkn)$ where t is the total number of iterations. Typically k and t will be very small compared to n.

It has a number of shortcomings:

- The basic version of the algorithm is applicable only when the mean is defined; this causes problems with data with discrete features.

- In many domains the need to specify k in advance is a drawback. One solution to this is to *hunt* through a range of k values looking for a partitioning that scores well on some validity metric (see section 4.7).

- It is poor at handling noisy data and outliers.
- Not suitable for discovering clusters with non-convex shapes.
- The solution found depends on the initial partitioning as different starting positions can lead to different local maxima. This is often addressed by running the algorithm several times and selecting the solution that scores well on some validity metric (see section 4.7).

Despite these weakness it remains a very popular algorithm.

The most popular variant on the *k*-means idea is *k*-medoids, also known as Partitioning Around Medoids (PAM)[78]. With *k*-medoids, the clustering is done around representative objects in clusters (the medoids) rather than around cluster means. Each cluster is represented by one of the objects located near the centre of the cluster. The algorithm begins by selecting an object as medoid for each of *k* clusters, then each of non-selected objects is grouped with the medoid to which it is the most similar. The algorithm swaps the medoids with other non-selected objects until all objects have been tried as medoids. Objects that work better as medoids (i.e. reduce the cost function) are retained as medoids. This extra layer of checking (all potential medoids v's a single mean) means that *k*-medoids is an order of magnitude slower than *k*-means. *k*-medoids works effectively for small data sets, but does not scale well for large data sets [78].

4.4 Overview of prototype-based clustering techniques

Generally speaking, data clustering consists in finding natural groups, or clusters, of similar patterns. Clustering techniques are classically divided into two broad categories: hierarchical and partitional algorithms [65, 66], even so we can also distinguish for example (probabilistic) model-based, grid-based or path-based clustering. We concentrate here on the partitional methods and especially on prototype-based clustering methods. Among the latter, the K-means algorithm is perhaps the most widely used for the sake of simplicity and efficiency.

Originally devised in [99] as an *on-line* clustering technique, most people refer to K-means as a *batch* algorithm. It provides a "hard" partition of the data as opposed to its "fuzzy" counterpart called fuzzy C-means [11]. The common purpose of these center-based clustering algorithms is to summarize multivariate data by a reduced set of central points. It is very close to vector quantization design which consists in encoding a signal source with codebook reference vectors. The best known technique in this domain, the LBG algorithm [97], can be interpreted as an adaptation of K-means for data compression. On the other hand, on-line versions are connected to the neural networks literature and the so-called competitive learning algorithms. We should note eventually that data mining applications have required specific clustering algorithms able to deal with large datasets and high-dimensional feature space [51].

K-means clustering consists in finding K clusters such that a global distortion error is minimized. Let $\mathcal{X} = \{\mathbf{x}_n\}_{n=1,\ldots,N}$ be a set of feature vectors and assume that each data instance contains d real-valued attributes:

FFFF IIII XXXXX The standard objective function is actually a sum of variances within each group. It involves the K centers of each cluster (often called prototypes) and the criterion

amounts to the minimization of the sum of distances to the nearest prototypes:

$$E(\mathcal{Y}_K) = \sum_{l=1}^{K} \sum_{\mathbf{x}_n \in \mathcal{C}_l} \mathcal{D}(\mathbf{x}_n, \mathbf{y}_l) \tag{4.1}$$

$$= \sum_{n=1}^{N} \min_{l \in \{1,\ldots,K\}} \mathcal{D}(\mathbf{x}_n, \mathbf{y}_l) \tag{4.2}$$

where $\mathcal{Y}_K = (\mathbf{y}_1, \ldots, \mathbf{y}_K)$ is the unknown K-tuple of prototypes and $(\mathcal{C}_1, \ldots, \mathcal{C}_K)$ the corresponding clusters which constitute a partition of the dataset. Conventional K-means algorithm uses the square Euclidean distance: $\mathcal{D}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2 = \sum_{i=1}^{d} (x_i - y_i)^2$, but other center-based clustering algorithms can be derived by using different measures of distortion or by assigning a weight to each data point.

Even in the standard case, minimizing the objective function requires an iterative procedure and getting the global minimum is not guaranteed. Optimization starts with some given prototypes, and alternates data allocation to the nearest prototype and update of the prototypes (as means of the sample vectors assigned to the corresponding cluster). As the EM algorithm for Gaussian mixture model estimation, such techniques only converge to one of numerous local minima and are known to be sensitive to outliers. Moreover, the number K of clusters must be known *a priori*.

Actually, the recent works on the subject have tried to overcome the major drawbacks of conventional K-means, namely influence of noise and outliers, choice of the number of clusters and sensitivity to initialization. To improve the initialization step, the basic idea is to replace random sample with refined starting points selected by a rough estimation of the modes in the data distribution. One alternative is to perform a stochastic optimization method like simulated annealing in order to avoid local minima of the given cost function.

We now briefly review partitional approaches to both *robust* clustering in the presence of noisy data and *unsupervised* clustering when the optimal number of clusters K is unknown.

The general technique to handling the problem of K-means sensitivity to outliers is to modify the objective function by considering an additional noise cluster or by incorporating concepts from robust statistics. The K-medoids algorithm is one of the earliest solution [79], but most algorithms have been derived in the framework of fuzzy clustering so that it can deal with overlapping cluster boundaries [23]. Note also that the notion of robust vector quantization developed in the context of data encoding and transmission does not refer to outliers but to channel noise and to random elimination of prototypes for codebook reduction [62].

The traditional method to determining K is to select the K-means partition that optimizes a certain validity measure over a range of different K values [12]. Nevertheless, the computational cost is quite high and the choice of appropriate validity indices evaluating the quality of the partition still remains a difficult question [58]. One alternative is to perform some progressive clustering by starting with an over-specified number of clusters and then adding a second phase of splitting and merging like in the popular ISODATA algorithm [4]. “Stepwise” clustering and “dynamic” local search have also been suggested [74] but all these solutions need to specify similarity criteria and to set thresholds. A non-parametric scale-space model order selection was presented in [125]. In the same way, recent methods try to make no additional assumption about the structure of the data distribution [93].

Very few techniques attempt to globally solve the unsupervised robust clustering problem. In [46], the authors propose a fuzzy clustering algorithm involving a robust "competitive agglomeration" process previously introduced in [43]. A regularized objective function is put forward with the use of fuzzy memberships and a robust loss function, while a separate virtual noise cluster is introduced in [95] to catch the outliers. Even so the number of clusters can be derived in that way, such a regularization approach requires in practice several heuristics to tune the parameters of the considered criterion. In [129], a novel extension of the standard K-means algorithm is suggested for learning the structure of each class of semantic events in video content analysis. It is based on the use of a robust distance involving a single scale parameter. The algorithm starts with a single cluster and progressively creates new clusters from the data considered as outliers with respect to the existing clusters. The number of prototypes K is determined by choosing the optimal value of the scale parameter for which the partition is the most stable.

4.5 Unsupervised Bayesian Learning in Image Segmentation

4.5.1 Introduction

Bayesian inference provides a well-founded mathematical framework in which to develop machine-learning methods. Recall that the premise of the Bayesian approach is that there are quantities of interest θ that we wish to learn about, and that we have data x available to us. In the multimedia setting, x is our image, audio or video signal, and θ is the output of our method, be it for example a restored image, the location and/or time of events in a video sequence, or transcribed speech.

The Bayesian paradigm says that what we have learnt about θ in light of x is described by the probability distribution $p(\theta|x)$, known as the posterior distribution. By Bayes' law, this distribution can be computed as

$$p(\theta|x) \propto p(x|\theta)\, p(\theta),$$

where the constant of proportionality in this relationship ensures that $p(\theta|x)$ sums or integrates to 1. On the left hand side, we have two distributions that we must specify:

- the distribution $p(x|\theta)$, known as the likelihood. This shows how the data arose from a particular value of θ. It is the model for our data.

- the distribution $p(\theta)$ is the prior distribution. This specifies what we know about θ prior to observing x.

There are many general texts on Bayesian learning. The foundations of the approach are described in [26]. In the statistical literature, [96] is a good introduction. A more comprehensive text is [49]. A good reference for the computational approaches to computing posterior distributions is [136].

4.5.2 The Prior Distribution in Multimedia Learning

Whereas the data model $p(x|\theta)$ is specified in all probabilistic learning methods, the prior distribution is a quantity unique to the Bayesian approach. It has both advantages and disadvantages. The main advantage is that it is a natural way to specify "structural" information about the quantities of interest. For example in image restoration one may specify a prior distribution that puts more weight on images that look like natural scenes. In object recognition we can specify whether occlusion can or cannot occur, and place zero probability on impossible locations of objects in a scene. The disadvantages are usually computational — Bayesian learning is not fast to compute — and a lack of objectivity, in the sense that different users may apply different priors to get different results. There is also a difficulty in specifying complete ignorance about a quantity, which is sometimes required in other statistical applications, although usually in multimedia data we are not in complete ignorance about θ.

4.5.3 Application: Unsupervised Segmentation of Satellite Images

In this example we use a prior that specifies structural information on the segmentation of an image, and use it to construct an unsupervised segmentation algorithm. The prior is on the field of labels that specify, for each pixel, to which segmentation class it belongs. A structural property of segmentations is that classes are clustered in space, as they represent continuous objects or activities. The Markov random field [141] is a natural way to specify a distribution on the labels that allows this clustering; they also specify a model for the appearance of each class, leading to the term double Markov random field. It has been extensively used in Bayesian approaches to image segmentation, see [108] for instance. Markov random fields have many applications to segmentation, see [29, 121] for example. This example also shows the considerable computational difficulties that can be encountered with the Bayesian approach, in particular in determining the number of classes in the segmentation proves to be computationally demanding.

Consider a rectangular lattice of pixel sites $\mathcal{S}$. An image consists of an array of grey values $(x_s)_{s\in\mathcal{S}}$ and labels $(y_s)_{s\in\mathcal{S}}$, identifying the texture type present. Let there be R textures in the image and each texture, defined on all of $\mathcal{S}$, is a Markov random field T^r, parameterised by θ_r, with neighbourhood system having set of cliques $\mathcal{C}_r$. The label process is another Markov random field Y, parameterised by β and with neighbourhood system having set of cliques $\mathcal{C}_Y$. All the fields are independent conditional on model parameters.

We define the $P(T^r|\theta_r)$ to be Gaussian Markov random fields (GMRFs), so that for a GMRF having K clique types in its neighbourhood system, $\theta_r = (\mu_r, \sigma_r^2, \theta_{r1}, \ldots, \theta_{rK})$, and $P(Y = y|R,\beta)$ to be a Potts model, so $\beta > 0$, which places more weight on labellings where neighbouring pixels are in the same class.

The *double Markov random field* is a probability distribution over X and Y of the form

$$P(X = x, Y = y|R,\theta_1,\ldots,\theta_R,\beta) = P(Y = y|R,\beta)\prod_{r=1}^{R} P(T^r_{S_r} = x_{S_r}|\theta_r), \quad (4.3)$$

where $S_r = \{s \in \mathcal{S}\,|\,y_s = r\}$ and $T^r_{S_r}$, x_{S_r} denote T^r and x restricted to S_r, and the distributions of T^r and Y are Gaussian and that of the Potts model respectively.

In what follows we assume that the classes are ordered by their mean value i.e. $\mu_r \leq \mu_{r+1}$. The ordering removes the non-identifiability of a segmentation, in the sense that, without the ordering, the method cannot distinguish between segmentations where the class labels are permuted. The non-identifiability would complicate the method that we propose for determining R.

4.5.3.1 Using the Model for Segmentation

In a Bayesian approach to unsupervised segmentation, the goal is to calculate the distribution of Y, R and the model parameters if unknown, given X, that is:

$$P(Y,R,\theta_1,\ldots,\theta_R,\beta\,|\,X) \propto P(X,Y\,|\,R,\theta_1,\ldots,\theta_R,\beta)P(R,\theta_1,\ldots,\theta_R,\beta), \tag{4.4}$$

where $P(R,\theta_1,\ldots,\theta_R,\beta)$ is a prior distribution on the parameters. The parameters are usually assumed independent *a priori*. In the GMRF case, these are assumed to be: uniform prior distribution for μ_r over the range $[0,255]$, inverse gamma prior for σ_r^2 with parameters a and b, uniform prior for $(\theta_{r1},\ldots,\theta_{rK})$ over the allowable range of values (see [92] for this range in the case of a second order neighbourhood system) and geometric prior on R with parameter p. The only available technique to evaluate this posterior distribution is Monte Carlo Markov chain (MCMC) simulation, usually the Gibbs sampler, where one simulates from the full conditional distributions of each Y_s, $s \in \mathcal{S}$, and those of β and θ_r, $r = 1,\ldots,R$. To simulate from R, other methods are needed, as described below.

For the double Markov random field, the full conditional distribution for the pixel labels is not easy to evaluate. We use an approximation to it; in [108] it was shown that the pseudo-likelihood performed well,

$$P(X,Y\,|\,R,\beta,\theta_1,\ldots,\theta_R) \approx P(Y\,|\,R,\beta)\prod_{s\in\mathcal{S}} P(X_s\,|\,\theta_{Y_s},X_t;t\in\eta_s), \tag{4.5}$$

where η_s is the neighbourhood of s. For each s, it is assumed that the texture Y_s holds in the entire neighbourhood of s. This approximation is developed in [10]. In the case of the Potts-Gaussian model, the full conditional of Y_s is:

$$P(y_s = r|x,y_j,j\neq s,\theta_r,\beta) \propto \frac{\exp\left[-\frac{\left\{x_s-\mu_r-\sum_{k=1}^{K}\sum_{j|<s,j>_k}\theta_{rk}(x_j-\mu_r)\right\}^2}{2\sigma_r^2}\right]}{\sqrt{2\pi\sigma_r^2}} \times \exp\left[\beta\sum_{j\in\eta_s} I(r=y_j)\right], \tag{4.6}$$

where $<s,j>_k$ means that pixels s and j form a clique of type k (horizontal, vertical neighbours, etc.), η_s is the neighbourhood of s in the Potts model and $I()$ is the indicator function.

The Gibbs sampler repeatedly samples from each full conditional in turn, forming a Markov chain whose stationary distribution is the posterior distribution. As a solution, we take the MAP or most likely posterior segmentation. This is found by running Gibbs sampling in tandem with simulated annealing to find the maximum.

4.5.3.2 The Reversible Jump Approach to Sampling from R

Reversible jump has been used for identifying R in image segmentation by Markov random fields [6, 76]. Reversible jump was proposed in [52], which we refer to for a full description of the approach. The idea is to sample R by a Metropolis move. This requires an acceptance probability, and in this application the only moves that are both practical and easy to propose with a non-zero acceptance probability are to increase or decrease R by 1. Increasing R by 1 requires that we generate parameter values for a new class, and a new segmentation that involves the new class. The new parameter values must be generated according to a 1-1 transformation between the new set of values and the old parameters together with random numbers necessary to generate the parameters for the new class. So a move from $\Theta_R = (\theta_1, \ldots, \theta_R)$ to $\Theta^*_{R+1} = (\theta^*_1, \ldots, \theta^*_R, \theta^*_{R+1})$ is achieved with a set random numbers u of the same dimension as θ^*_{R+1} such that there is a 1-1 function $(\Theta_R, u) \leftrightarrow \Theta^*_{R+1}$. The usual conditions of reversibility and irreducibility must be maintained. Decreasing R by 1 requires that we eliminate one of the classes, and propose a segmentation where the deleted class is not present. The reversibility means that this move must be done using the same 1-1 function between Θ^*_{R+1} and (Θ_R, u). Once either move has been done, an acceptance probability can be computed. For the move to increase R to $R+1$, with a change in parameters from Θ_R to Θ^*_{R+1}, and a change in segmentation from Y to Y^*, this acceptance probability is the minimum of 1 and

$$\frac{P(X,Y^* \,|\, R+1, \Theta^*_{R+1}, \beta)}{P(X,Y \,|\, R, \Theta_R, \beta)} \frac{P(\Theta^*_{R+1})}{P(\Theta_R)} \frac{P(R+1)}{P(R)} \frac{1}{P(u)} \frac{P(\text{decrease } R+1)}{P(\text{increase } R)} \left| \frac{\partial(\Theta_R, u)}{\partial \Theta^*_{R+1}} \right|, \tag{4.7}$$

where the final partial derivative is the Jacobian of the transformation $(\Theta_R, u) \rightarrow \Theta^*_{R+1}$, and $P(\text{decrease } R+1)$ and $P(\text{increase } R)$ refer to any other probabilities involved in decreasing or increasing the number of classes (for example, randomly choosing which class to delete).

The above is a strategy for sampling for R. Sampling from the other parameter values and Y is from the full conditionals, as described in Section 4.5.3.1. Repeated sampling like this in tandem with simulated annealing produces our solution.

4.5.3.3 Split and Merge Moves

Increasing R by 1 is done by choosing an existing class and splitting it into two. Several split moves for increasing R have been considered. Finally we propose the following, which is a compromise between computational complexity and arriving at a proposed new segmentation that has a reasonable acceptance probability. To increase the number of classes from R to $R+1$, we propose to:

1. Randomly pick a non-empty class c to split.
2. Generate new parameters θ^*_{c1} and θ^*_{c2} from θ_c and random numbers u by a 1-1 transformation such that $\dim(\theta_c, u) = \dim(\theta^*_{c1}, \theta^*_{c2})$. For GMRF texture models, we use the following, designed to be simple perturbations of the current parameters:

$$\begin{aligned} \mu^*_{c1} &= \mu_c - u_1 \sigma_c & \mu^*_{c2} &= \mu_c + u_1 \sigma_c \\ \sigma^{*2}_{c1} &= \sigma^2_c (1 + u_2) & \sigma^{*2}_{c2} &= \sigma^2_c (1 - u_2) \\ \theta^*_{c1,k} &= \theta_{ck} + u_{2+k} & \theta^*_{c2,k} &= \theta_{ck} - u_{2+k}, \end{aligned} \tag{4.8}$$

where the u_1 is uniform(0,1), u_2 is uniform(-1,1), u_{2+k} is uniform(-0.1,0.1) and $k = 1,\ldots,K$. The Jacobian of the transformation in Equation 4.8 is $4\sigma_c^3 2^K$.

3. Check to see that the new means are still ordered (that is, $\mu_{c-1} < \mu^*_{c1} < \mu^*_{c2} < \mu_{c+1}$) and lie in the prior range $[0,255]$. If not, reject move immediately. If so, continue.

4. Assign all labels currently assigned to class c to classes $c1$ or $c2$. This is done as follows. Labels in the set $\mathcal{S}_c$ are assigned randomly to either $c1$ or $c2$. Then several Gibbs sampling sweeps are taken over $\mathcal{S}_c$ only, using classes $c1$ and $c2$ and the full conditionals of Equation 4.6. The Gibbs sampling hopefully produces homogeneous regions in $\mathcal{S}_c$ of each new class.

5. Compute the acceptance probability (see below) and accept or reject the move. If accept, relabel parameters and classes to include $c1$ and $c2$.

The choices for the u at step 2 are a balance between allowing the possibility of reasonably large changes in the properties of old and new classes, and restricting changes so that the chance of accepting the move is not always 0.

Reducing R is done by choosing two existing classes and merging them into one. The merge move is to choose a class $c1$, select the class $c2$ that has closest mean to it (from the ordering on means property, this is either $c1+1$ or $c1-1$) and propose new parameters for the merged class c that are the inverse of the transformation in Equation 4.8. We use the following transformation of parameters for the reversal of the split transformation:

$$\begin{aligned} \mu^*_c &= \frac{n_1}{n_1+n_2}\mu_{c1} + \frac{n_2}{n_1+n_2}\mu_{c2}; \\ \sigma^{*2}_c &= \frac{\sigma^2_{c1}+\sigma^*_{c2}}{2}; \quad \theta^*_{ck} = \frac{\theta_{c1,k}+\theta_{c2,k}}{2}, \end{aligned} \tag{4.9}$$

where n_1 and n_2 are the number of pixels assigned to $c1$ and $c2$ respectively.

The accept probability is rather complicated and is omitted here; we refer you to [140] for its form, which is also accessible via the INRIA website .

4.5.3.4 Experiments

Figure 4.3 shows an aerial image, with the MAP solution from the reversible jump algorithm. The algorithm finally converged to 10 classes. This is an oversegmentation, in the sense of discriminating the main classes in the image: different fields, trees and the road. It has also segmented each field into different classes that correspond to different colourations in the image.

Figure 4.4 is an analysis of a satellite image of an area of Holland. In this case we end up with 24 classes. This represents another oversegmentation; for example, the reversible jump method of [76] finds 10 classes. The image consists of fields, which the algorithm classifies into different classes according to mean intensity, and urban areas, which the algorithm segments according to mean intensity, ignoring the similar texture over all the urban areas. We also find that 90% of pixels are assigned to only 4 classes. In spite of this, the algorithm did not merge the classes assigned to only a small number of pixels. We refer to [140] for more details.

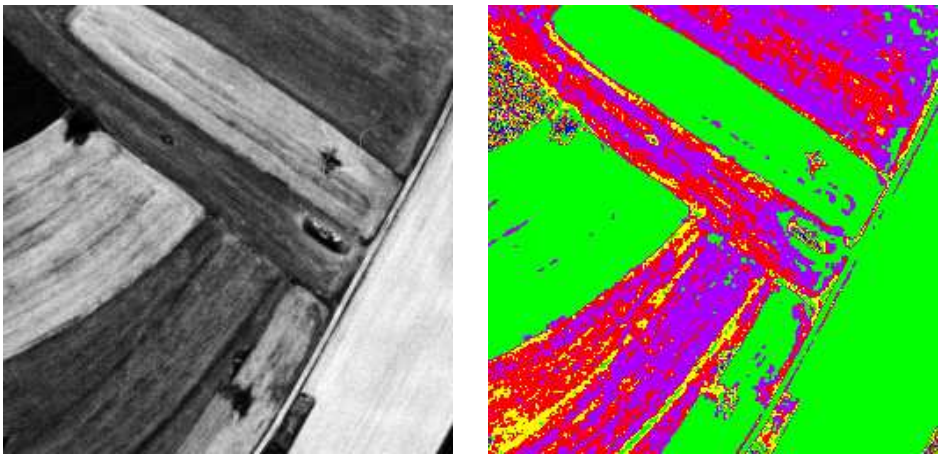


Figure 4.3: An aerial image on the left, with MAP segmentation, using splitting and merging of non-empty classes.

4.5.3.5 Conclusions

In this example we have developed a Bayesian segmentation algorithm based on a Markov random field prior for the labelling, implemented with MCMC. The split/merge moves proposed here seem to oversegment the images, with many classes being assigned to small proportions of the image and not merged. Possibilities for reducing the number of classes are: larger neighbourhood orders for the labels, fixing β to be reasonably large, running an MPM (marginal posterior mode) sampler or some form of post-processing. Other MCMC approaches that infer the number of classes may also provide ideas, particularly those using stochastic geometry (see [64] for work in Projet Ariana). These will be topics for future work.

4.6 Self-Organizing Maps: Principles and Algorithms

In many applications (data mining, pattern recognition, etc.), the dimensionality of input patterns is very large and it is thus extremely difficult for an analyst to extract the relevant features or to detect some useful structures in the data. When none *a priori* knowledge is available regarding the distribution of input data, unsupervised clustering is then an efficient tool to help the analyst to make a decision. Roughly speaking, the clustering allows to group a given collection of unlabeled patterns into meaningful clusters [70]. The obtained clusters are data driven in the sense that they are obtained solely from the input data. The main problems consists in defining the cluster boundaries as well as a meaningful representative pattern for each class. Many methods have been proposed for clustering data [69, 70] and generally, each class is represented by the centroid of the associated cluster. A particular class of clustering methods, called *partitional* methods, produce clusters by optimizing a criterion function defined either locally (on a subset of patterns) or globally (defined over all of the patterns). This kind of approach is closely related to vector quantization [19, 50]. Indeed, the design of a vector quantizer need a representation of the input data with a limited number of reproduction vectors (called *codewords*) and the defini-

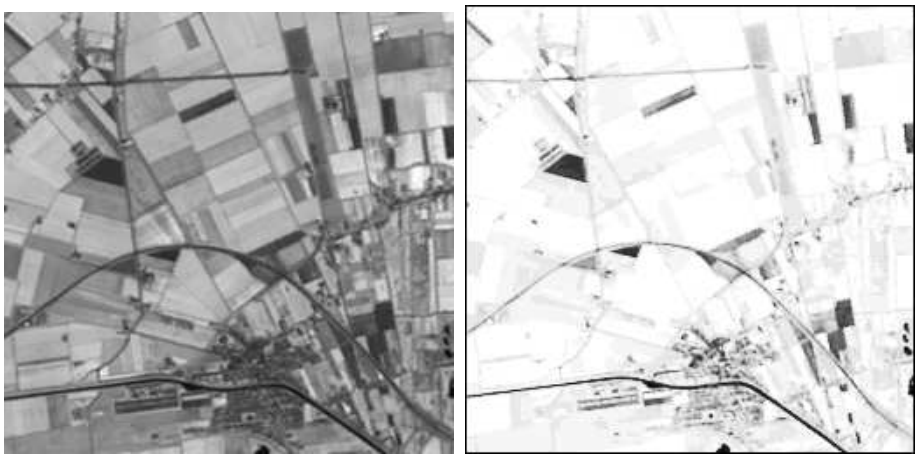


Figure 4.4: Image of an agricultural area (top), with MAP segmentation (bottom).

tion of a rule for mapping input vectors into the codewords. As mentioned in [50], an important special class of vector quantizers that is of particular interest, called *Voronoi* or *nearest neighbor* vector quantizers, has the feature that the partition is completely determined by the codebook (i.e. the set of the codewords) and a distortion measure. In fact, for a given codebook, such an encoder is optimal in the sense of minimizing average distortion and, without loss of generality, the term vector quantizer is commonly assumed to be synonymous with nearest neighbor vector quantizer. One of the most popular vector quantizer is the *Generalized Lloyd Algorithm*, also known as the *k-means* algorithm. This algorithm uses the nearest neighbor mapping rule and is thus optimal as underlined before. It is also largely used for clustering data in an unsupervised way when no *a priori* information is available regarding the probability density function of the data to be classified.
SOFM (*Self Organizing Feature Map*), also known as the Kohonen algorithm [86, 87], is a variant of the k-means algorithm in the sense that it also uses the nearest neighbor mapping rule but it also presents the great advantage to preserve the topology of input data. For this purpose, SOFM is built upon a lattice A of neural units (also called cells) with a given dimensionality and topology (generally rectangular or hexagonal). Each cell $i \in A$ is connected to a set of neighboring cells $\mathcal{N}_A(i)$ and the Kohonen algorithm defines a mapping Ψ from the input space $M \subset \mathbb{R}^d$ into A such that two similar input vectors in M are mapped onto neighboring neural units (see Figure 4.5).

To perform such mapping, SOFM is based on competitive learning [39] for which any input vector activates only one cell. This kind of learning is based on an adaptative process in which the cells of the network are progressively tuned to specific features of the input data set. In order to take place, each neural unit $i \in A$ is associated to a reference vector $m_i(t) \in \mathbb{R}^d$ where t indicates the time index. These reference vectors are progressively adapted to input features during an unsupervised learning process. As a result, a partitionning of the input space is built since the spatial location of a given cell $i \in A$ corresponds to a particular domain $R_i \subset M$ called the *receptive field* of i.
Usually, the dimension of A is lower than the dimension of M explaining why the Kohonen

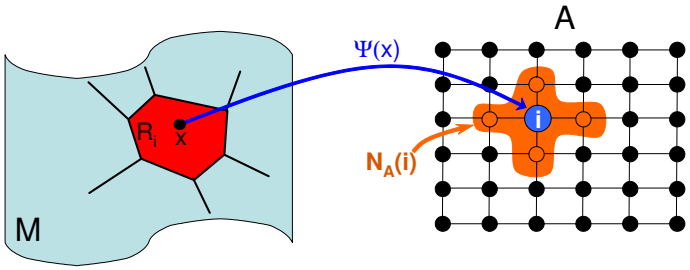


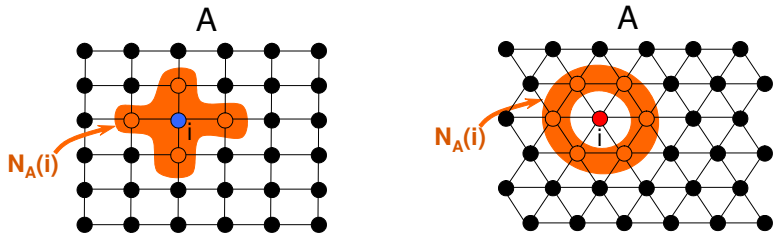
Figure 4.5: General principle of Kohonen networks

algorithm is often presented as an efficient tool to perform dimension reduction.
It is also worth noting that this kind of neural network is closely related to the human brain for which perceptually similar stimuli activates nearby neural zones.
This document is organized as follows: section 4.6.1 details the basic Kohonen algorithm and explains, from examples, how the lattice adapts its topology to reflect the underlying distribution of input data. Section 4.6.2 presents the supervised versions of the SOM learning rule called LVQ (*Learning Vector Quantization*) generally used to perform pattern classification. Section 4.6.3 details some quantitative measures that can be used to check the quality of the mapping. Section 4.6.4 presents some dynamic lattice structures that enhance the representation of input data. In particular, we will discuss about hierarchical structures that greatly improve the adaptation of the map topology with regard to the distribution of input data. Finally, section 4.6.5 concludes this document.

4.6.1 Basic Self-Organizing Map Algorithm

4.6.1.1 Neural Network Model

The Kohonen model is based on the construction of a neuron layer in which the neural units are arranged in a lattice *A*. Usually, the lattice is two-dimensional and the most popular lattices are rectangular or hexagonal (see Figure 4.6(a,b)).



(a) (b)

Figure 4.6: Rectangular (a) and hexagonal (b) two-dimensional lattices

The neural layer is innervated by *d* input fibers (i.e. as many fibers as the dimension of the input space *M*) called *axons* which carry the input signals and excite or inhibit the neurons via

synaptic connections (see Figure 4.7).

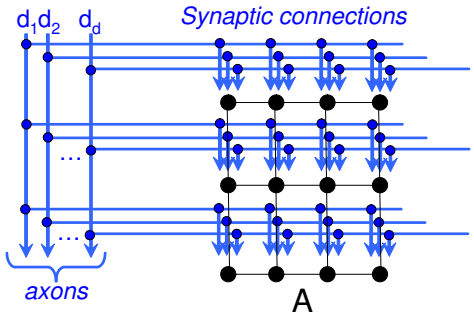


Figure 4.7: Neural network model

As underlined in the previous section, the Kohonen network aims at preserving the topology of the input space and at tuning each cell to a particular set of stimuli. To reach these goals, the excitation of neurons has to be restricted to a spatially localized region in A and the location of this region has to be determined by those neurons that respond most intensively to a given stimulus. Moreover, A acts as a topographic feature map if the location of the most strongly excited neurons is correlated in a regular and continuous fashion with a restricted number of signal features of interest [123]. Neighboring locations in A correspond thus to stimuli with similar features. To satisfy these properties, a neighboring function between cells must be added in the network model. For this purpose, each cell $i \in A$ is connected to a set $\mathcal{N}_A(i)$ of neighboring cells, defining thus a topological ordering of cells.
The goal of the Kohonen learning algorithm is then to adapt the "shape" of A to the distribution of the input vectors of $M \subset \mathbb{R}^d$.
In the following and without loss of generality, we assume that the Kohonen network is composed of n neural units arranged in a two-dimensional lattice A.

4.6.1.2 Competitive Learning

Let:

- $X = \{x(t),\ t = 1,2,\ldots\}$ be a set of observable samples with $x(t) \in M \subset \mathbb{R}^d$, t being the time index;
- $M = \{m_i(t),\ i = 1,2,\ldots,n \text{ and } t = 1,2,\ldots\}$ be a set of reference vectors with $m_i(t) \in \mathbb{R}^d\ \forall i,t$.

We suppose that $m_i(0)$ has been initialized in a proper way for all values of i. Generally, a random initialization would suffice.
If $x(t)$ can be compared simultaneously to all $m_i(t)$ by using a distance measure $d(x(t),m_i(t))$ in the input space, then the best candidate $m_c(t)$ is defined by:

$$m_c(t) = arg\ \min_i d(x(t),m_i(t))\ i = 1,2,\ldots,n.$$

By associating each reference vector $m_i(t)$ to a neural unit $i \in A$, the cell c attached to $m_c(t)$ is called the BMU (*Best Matching Unit*).
The idea behind the competitive learning paradigm consists in updating $m_c(t)$ in order to match even closely $x(t)$. By this way, the distance between $x(t)$ and $m_c(t)$ is decreased whereas the other reference vectors remain unchanged. This approach tends to tune each cell to a specific domain of the input space M.
Ideally, the updating rule should modify the reference vectors in such a way that they approximate the continuous density function $p(x)$ of the input vectors belonging to X. By analogy with vector quantization, the quality of the placement of reference vectors can be measured by using the r^{th} power of the reconstruction error E_r:

$$E_r = \int \|x - m_c\|^r p(x) dx$$

where m_c is the reference vector associated to the BMU corresponding to x:

$$\|x - m_c\| = \min_i \{\|x - m_i\|\}.$$

It can be shown that for $r = 2$, minimizing E_r conducts to the following stepwize delta rule used to update the reference vectors:

$$\begin{cases} m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)] \\ m_i(t+1) = m_i(t) \; \forall \; i \neq c \end{cases} \tag{4.10}$$

where $\alpha(t)$ designates the *learning rate* i.e. a monotonically decreasing sequence of scalar values with $0 < \alpha(t) < 1$.
As mentioned previously, this updating rule decreases the distance between $x(t)$ and $m_c(t)$, while the others remain unchanged (see Figure 4.8).

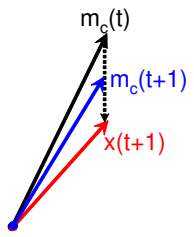


Figure 4.8: Illustration of the competitive delta rule

The rule (4.10) defines the basic rule of competitive learning and it is important to note that it allows to obtain the optimal placement of reference vectors with regard to the mean square reconstruction error. However, by using this simple rule, each cell acts independently and the order in which they are assigned to the different domains of X strongly depends on the initial values of $m_i(0)$ [86]. To reflect the topological ordering observed in the human visual cortex, this delta rule must be modified by incorporating a topological function.

4.6.1.3 Topological Ordering

A learning scheme that preserves the topology of input data is such that two vectors that are close to each others in the input space are mapped on nearby cells in the output space. By this way, the network is able to adapt its topology to the "shape" of the input data density function. To reach this goal, the lateral interactions between cells are reinforced by integrating the neighborhood $\mathcal{N}_A(i)$ of each cell i in the delta rule. By this way, at each learning step, the reference vectors associated to the cells that lie within the neighborhood $\mathcal{N}_A(c)$ of the BMU c are updated while the others remain unchanged.
In practice, the width of the neighborhood varies with time leading to a neighborhood function $\mathcal{N}_A(c,t)$: in the first learning steps, the width is chosen large allowing to quickly obtain a global ordering and then, the width decreases monotonically to obtain a fine tuning of the reference vectors to input features (see Figure 4.9).

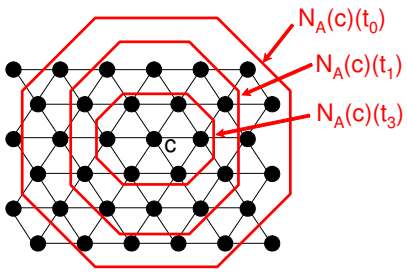


Figure 4.9: Variation of the neighborhood width with time

This new delta rule is expressed as follows:

$$\begin{cases} m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)] & \text{if } i \in \mathcal{N}_A(c,t) \\ m_i(t+1) = m_i(t) \ \forall \ i \neq c & \text{otherwise} \end{cases} \tag{4.11}$$

In practice, a kernel-based alternative to the delta rule (4.11) is used. This new updating scheme aims at performing a stronger weight adaptation at the BMU location than in its neighborhood. This kernel-based rule is defined by:

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t)[x(t) - m_i(t)] \tag{4.12}$$

where $h_{ci}(t)$ designates the neighborhood function that governs the strength of weight adaptation as well as the number of reference vectors to be updated. Typically, this neighborhood function is a unimodal function which is symmetric around the location of the BMU and monotically decreasing with increasing distance from the BMU. Classically, a Gaussian is used, leading to:

$$h_{ci}(t) = exp(-\frac{\|r_c - r_i\|^2}{2\delta(t)^2}) \tag{4.13}$$

where r_i denotes the 2D location vector of the cell i in the two-dimensional lattice A. The time dependent parameter $\delta(t)$ governs the width of the neighborhood function as underlined previously.

Intuitively, we can see by analyzing (4.12) that all cells lying in a close neighborhood of the BMU will be progressively tuned to similar features in the input space. Many methods have been proposed in the litterature to measure the degree of topology preservation as we will see later but it is very difficult to prove this assertion in the general case. Nevertheless, some attempts have been published for special cases and we invite the reader to refer to [7, 82, 83, 110, 138] for a deeper study on this subject.

4.6.1.4 Kohonen Basic Algorithm

In this section, we propose to show how to use the different notions introduced up to this point in order to obtain a complete algorithm performing competitive learning that preserves topology. A classical Kohonen algorithm is composed of four main steps: estimation of the lattice structure (i.e. number of neural units, dimension and topology), the initialization of the learning parameters (learning rate, neighborhood function, number of iterations), the selection of the BMU and the reference vectors updating.

Step 1: Estimation of the Lattice Structure Ideally, the lattice structure should be governed by a tradeoff between *resolution*, *quantization error* and *topology preservation*.
The two first notions are highly correlated: the resolution can be roughly defined as the mean volume of cell's receptive fields i.e. the volume of the domain associated to each cell in the input space. The quantization error qe is a useful measure that can be used to assess the resolution level:

$$qe = \frac{1}{p} \sum_{k=1}^{p} \|m_c - x_k\| \tag{4.14}$$

where p denotes the number of input vectors and m_c designates the reference vectors attached to the BMU associated to x_k. A high value of qe denotes a bad representation of input data by the set of reference vectors. Moreover, a high qe at the end of the learning step indicates generally an insufficient number of cells. In the basic version of the Kohonen algorithm, the number of cells as well as the size of each dimension of the network have to be predetermined and one often need to run several simulations to pick the optimal size. We will see later in this document (cf. section 4.6.4) how to iteratively adapt the size of the network to the distribution of input data.
An other important property to be satisfied by the network is the preservation of topology. However, the degree of preservation also depends on the lattice structure and in the form of the data manifold M. In fact, if a dimensional conflict occurs between the input and output spaces, the topology preservation property is lost. In [138], the authors introduce the notion of *topographic function* as a measure estimating the dimensional conflict.
In conclusion, defining an optimal lattice structure (i.e. a structure enabling an efficient representation of input vectors) is a tedious task that is generally performed in an empirical way. In fact, lattices of dimension two with an isotropic topology (rectangular or hexagonal) are generally used for two main reasons: the visualization of input data structure is facilitated by using 2D lattices and the access to the neighborhood of a cell is easy for isotropic structures. Many

adaptive methods have been proposed in the literature to obtain a 2D lattice structure enabling an efficient data representation [2, 47, 120].

Step 2: Initialization of the Learning Parameters The Kohonen Algorithm is controlled by four parameters: the learning rate $\alpha(t)$, the neighborhood function $h_{ci}(t)$, the number of learning iterations t_{max} and the initial reference vector values $m_i(0)\ \forall i = 1,2,\ldots,n$.
As underlined in [86], the convergence of the algorithm is ensured by imposing $\alpha(t)$ to be a monotically decreasing function of t but an accurate decreasing rule has no impact on the learning results. Consequently, the decreasing form can be linear, exponential of inversely proportional to t and we will discuss later about some useful rules that can be used to efficiently tuned this parameter during learning.
Regarding the neighborhood function, the classical used shape is the Gaussian one as defined in (4.13).
The stopping criterion of the learning steps can be expressed in two ways: either a maximum number t_{max} of iterations is defined before learning or the learning is performed until the quantization error is below a predefined threshold. This latter choice assumes a strong *a priori* knowledge about the distribution of input data and in practice, such a knowledge is not available. Therefore, we assume in the following that the learning is iterated during t_{max} steps. To define this number t_{max}, a "rule of thumb" [86] imposes that, for good statistical accuracy, the number of steps is at least 500 times the number of network cells.
Finally, without any *a priori* knowledge about input data distribution, the reference vectors are initialized with random values.

Step 3: Selection of the BMU As underlined in section 4.6.1.2, when presenting an input vector $x(t)$, the BMU is the cell c whose reference vector $m_c(t)$ is the closest one to $x(t)$ in the sense of a distance $d(x(t), m_c(t))$ computed in the input space. Two distances are used in practice: the Euclidian norm and the dot product. In the former case, the BMU is the cell whose reference vector has the smallest Euclidian distance from the input vector whereas in the latter case, the largest dot product designates the BMU.
For vectors of unit lengths, both methods are equivalent i.e. if the input vectors and reference vectors are of unit length, the same weight vector will be chosen as closest to the input vector, regardless of whether the Euclidian distance or the dot product is used. In practice, for consistency and to avoid the normalization step, the Euclidian distance is retained to choose the BMU.

Updating the Reference Vectors Once the BMU is chosen according to the previous step, the reference vector associated to the BMU as well as the ones associated to the cells lying in $h_{ci}(t)$ have to be updated according to the delta rule defined in (4.12).

4.6.1.5 Some Useful Rules

The rules presented in this section are extracted from [86, 88] and are intended to reach good learning results in terms of accurate input data representation.

- *Rule 1: number of iterations*
  Competitive learning can be interpreted as a stochastic process and thus, the final statistical accuracy of the mapping depends on the number of learning steps. As underlined before, a simple rule consists in performing a number of steps equivalent to 500 times the number of cells.

- *Rule 2: number of training samples*
  Generally, less input samples than the required number of steps are available. Consequently, each available input vector has to be used several times during training. Several alternatives exist: the available samples may be applied cyclically or in a randomly permuted order, or picked up at random from the basic set. However, experiments have shown that an ordered cyclic choice is often sufficient.

- *Rule 3: fine tuning of the learning rate*
  As mentioned before, the learning rate $\alpha(t)$ has to form a monotically decreasing sequence with regard to t. However, the decreasing duration has no need to coincide with the total learning duration. In fact, the first learning steps aim at obtaining a global ordering of the map while the objectives of the remaining steps are to reach a fine tuning of the reference vectors allowing to get an efficient representation of input data.
  A rule for tuning this parameter has been proposed in [86]: for approximately the first 1000 steps, $\alpha(t)$ should start with a value close to 1.0, thereafter decreasing monotonically. After this global ordering period, $\alpha(t)$ should attain small values (e.g. of the order of or less than 0.01) over a long period. Neither it is crucial whether the law for $\alpha(t)$ decreases linearly or exponentially during this final phase.

- *Rule 4: width of the neighborhood function*
  Special caution is required regarding the choice of the neighborhood width. If the neighborhood is to small during the first learning steps, the map is not able to reach a global topological ordering. To overcome this drawback, the rule generally agreed consists in choosing an initial neighborhood width of about half of the network width. This width can then decrease linearly with time to one unit. Finally, during the fine adjustment of the map, the neighborhood function can only cover the nearest neighbors of each cell in the lattice A.

4.6.1.6 Some Variants of the Basic SOM Algorithm

As we saw up to this point, the two main goals of SOM are vector quantization and topology preservation. However, these goals are in conflict when the dimension of the input space is larger than that of the lattice and in this case, the basic SOM algorithm tends to privilege the vector quantization goal at the cost of a poorer topology preservation. As mentioned earlier, the quantization quality can be measured by the quantization error qe (cf. (4.14)) and many measures have been proposed to assess the preservation of topology [7, 82, 83, 110, 138]. However, improving the quantization quality (by decreasing qe) often lead to decreasing the preservation of the topology. To find a good tradeoff between these two concepts, several variants have been proposed in the literature. In most cases, these variants modify the basic Kohonen algorithm by defining a new rule for choosing the BMU and/or a new rule for updating reference vectors.

In [82], the authors present three different strategies for choosing the BMU. These strategies are governed by two measures, namely *qe* and a topographic error TE that increases when two non adjacent cells are associated to nearby reference vectors in the input space. The first strategy consists in weighting *qe* and TE during the learning phase in order to choose the BMU. Consequently, the BMU is the cell whose reference vector is close to the input vector and such that the reference vector is somewhat close to the reference vectors associated to neighboring cells. In the second strategy, TE is improved by choosing the BMU by averaging a number of reference vectors that best match the input vector. Finally, the third strategy finds the minimal convex set (the simplex) in the input space with $\delta+1$ vertices adjacent in the lattice A where δ denotes the dimension of A. This last algorithm discourages the creation of knots by preventing the BMU from crossing the simplex bounded by other map nodes: it thereby lowers the TE. Regarding the experimental results, the first strategy produces interesting results but with a very high computational complexity, making it not usable. The second approach need a limited number of cells to be averaged but its efficiency is perceived late in the learning phase. Finally, the third strategy gives interesting results soon in the learning step.
In [83], the author presents a variant of the basic SOM algorithm called *AdSOM*. This variant is motivated by the following observations:

- if the natural dimension of the input space is larger than the dimension of the lattice, the map tries to approximate the higher dimension by folding itself in the input space;
- the degree of folding depends on the stiffness of the map that is governed by the width of the neighborhood function;
- excessive folding, that results in topographic error, manifests itself so that the BMU and the second BMU are no longer adjacent in the lattice.

The author proposes a solution for which the map preserves the topology while retaining as much flexibility as possible. To reach this goal, the idea is to make the radius of the neighborhood function dependent on the local degree of folding. In practice, the neighborhood function is still Gaussian (cf. (4.13)) but in this variant, each cell i has its own neighborhood width parameter $\sigma(i)$ determined by the local topographic error. The experimental results reported exhibit a negligible topographic error compared to the basic SOM algorithm.
In [73], the authors treat the case where the variances of input vector components are not of the same order of magnitude. As underlined before, the Euclidian norm is generally retained for choosing the BMU. When the variances between the input vector components are of the same order of magnitude, the choice of the Euclidian norm gives satisfying results. However, in the opposite case, an oblique orientation of the map is produced. To obtain a better orientation of the map, the authors propose to choose the BMU by using a weighted Euclidian distance in which the weight parameters are estimated recursively during the learning phase in order to balance the effects of the variance disparity. A geometric interpretation of this weighting scheme is that the equidistant surface around each reference vector in the input space becomes elliptic. Therefore, if we desire to force each cell to "win" the competitive learning as often as the others, one can add a constraint on the estimated weights in order to have a constant ellipsoid volume per reference vector. It is worth noting that the problem of disparity in variances of input vectors

components has been treated in a different way in [47] where the author constructs iteratively the lattice in order to efficiently match the distribution of input data.
In [30], the author proposes a variant called *competitive learning with conscience*. This algorithm is motivated by observing that all cells of the network are used but nothing ensure that each cell is chosen as the BMU as often as the others and in consequence, some cells can describe some domains in the input space with a very low density. Ideally, it should be a valuable characteristic of the network to force each cell to win the competition with a probability of $\frac{1}{n}$ where n denotes the number of cells in the map. However, it has been noted that the basic Kohonen algorithm does not achieve this result but is biased in favor of the regions of lower density of input vectors. The goal of the conscience mechanism proposed by the author is to bring all cells available into the solution quickly, and to bias the competition process so that each cell can win the competition with a probability close to $\frac{1}{n}$. To reach this goal, the conscience process is composed of two parts: the generation of the outputs of the competitive layer and the reference vector updating. In the conscience mechanism, the BMU is not necessarily the one whose reference vector is updated. A bias is introduced based on the number of times each cell won the competition. Therefore, the more a cell looses the competition and the more its reference vector is updated. Moreover, the more the probability of winning the competition is close to $\frac{1}{n}$ for each cell, and the more the conscience mechanism is analogous to the basic Kohonen algorithm. The experimental results reported have shown that the conscience approach is able to match the density function of input data better than the basic SOM algorithm by using a reduced number of learning steps. This approach has been successfully used in [8] for designing a model-based object recognition system based on geometric hashing.
An other variant discussed in this section concerns the choice of the neighborhood and has been discussed in [73]. In this paper, the authors remark that when the input vector distribution has a prominent shape, the choice of the BMU tends to be concentrated on a fraction of cells in the map. Moreover, due to the successive applications of the updating rule (4.12), it may easily happen that the reference vectors lying in zero-density areas are affected by input vectors from all surrounding parts of the nonzero distribution. Consequently, some cells usually remain outliers. To overcome this drawback, the authors propose a mechanism in which the neighborhood relationships between cells are defined adaptively during the learning phase. For this purpose, these relationships are defined by using the minimal spanning tree (MST) approach in which the arc lengths are simply defined by the Euclidian norm between reference vectors in the input space. The neighborhood of a cell is therefore the MST emanating from that cell.
Note that we have not discussed in this section about all variants aiming at building the lattice structure during the learning phase. This kind of approaches will be presented in detail in section 4.6.4.

4.6.1.7 Some Experimental Results

Figure 4.10 shows an example of application of SOM extracted from [123].

The experiments illustrated here assumes a curved region G containing a sound source moving in a random way within G (see Figure 4.10(a)). The sound is received by two microphones, each connected to an amplifier with logarithmic characteristics, producing thus two output signals v_1 and v_2 serving as inputs to a Kohonen network. This network is composed of 1600 neurons arranged in a 2D square lattice of size 40×40. As the input space is two-dimensional

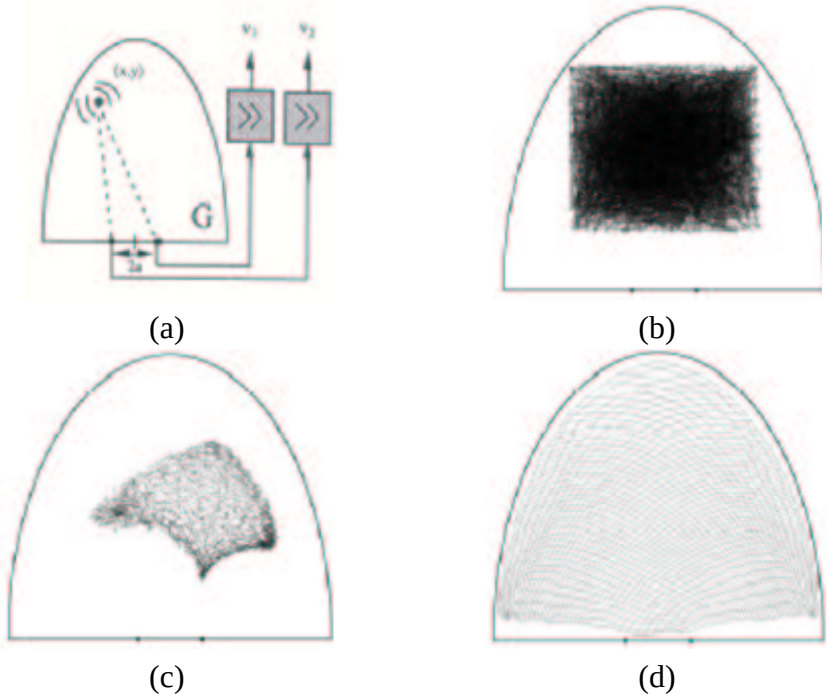


Figure 4.10: Application of the SOM algorithm. (a) Experimental conditions. (b)Initial values of reference vectors. (c)Values of reference vectors after 100 epochs. (d)Values of reference vectors after 40 000 epochs.

(i.e. the inputs are the amplifier responses), each network cell is associated to a reference vector with two components. Figure 4.10(b) illustrates the initial value of these reference vectors that have been randomly chosen inside the region G. After 100 learning steps, we can see (cf. Figure 4.10(c)) that the map starts to reach a global ordering. After 40,000 learning steps, a good correspondence between the reference vectors and the region G is obtained, as shown in Figure 4.10(d).

4.6.2 Learning Vector Quantization

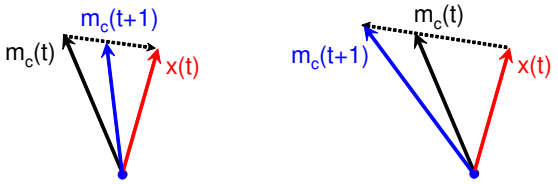
Up to this point, the SOM has been presented as an efficient unsupervised method to approximate the density function of input vectors from a limited number of codewords. However, if the problem to be solved is the classification of patterns, then the problem becomes a decision process and must be handled differently. Indeed, whereas the SOM tries to find an optimal placement of reference vectors in the sense of a distortion measure, a classification task has to affect a set of input patterns to different classes while obtaining a low misclassification rate. However, there is no guarantee that the basic SOM algorithm provides a placement of reference vectors that minimizes classification errors. The use of SOM for pattern classification is obvious: each reference vector represents a particular class and an input pattern p is categorized to the class whose the associated reference vector is the closest to p. In that case, the set of cell's receptive fields define the boundaries of each class. In a classification task, the classification

errors appear to be located at the class borders and thus the reference vectors have to be updated in order to shift the positions of these borders, resulting in improved classification rates. To allow this updating step, a training data set $T = \{(p,l)\}$ has to be built where p denotes a pattern to be classified ($p \in \mathbb{R}^d$) and l is the known label (or category) of p. *Learning Vector Quantization* (LVQ) methods [85] have been proposed to update in an efficient way the original reference vectors obtained by the basic SOM (or any other vector quantization method) in order to reduce the classification errors. It is thus possible to define effective values for the reference vectors such that they directly define near optimal decision borders between the classes, even in the sense of classical Bayesian theory which is known to be optimal [137].
LVQ methods can thus be considered as supervised alternatives to the basic SOM algorithm.
The starting point of each method presented below considers that a set of codewords defines a partition of the input space in such a way that the overall statistical density function of the input data is roughly described. The placement of these codewords could thus be performed by any vector quantization algorithm or even the basic SOM algorithm. The next phase is to determine the labels of the codewords, by presenting a number of input vectors with known classification, and assigning the cells to different classes by majority voting. It is important to note that this phase can conduct a single class to be represented by several reference vectors.
In the next section, we present different methods that can be used to finely tune the reference vectors for a classification task.

4.6.2.1 Type One Learning Vector Quantization (LVQ1)

The main idea of this first method consists in pulling the reference vectors away from the decision surface to demarcate the borders more accurately. Assume that m_c denotes the reference vector that is closest to the input vector x with a known classification. The reference vectors are updated as follows (see Figure 4.11(a,b)):

$$\begin{cases} m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)] & \text{if } x \text{ is classified correctly} \\ m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)] & \text{if the classification of } x \text{ is incorrect} \\ m_i(t+1) = m_i(t) & \text{for } i \neq c. \end{cases} \tag{4.15}$$



(a) (b)

Figure 4.11: LVQ1 adaptation rule. (a) corresponds to the case where x is correctly classified, whereas (b) corresponds to a misclassification

Let us recall that in these algorithms, the reference vectors are assumed to have a roughly correct value and need only a fine tuning. For this reason, the learning rate $\alpha(t)$ used in (4.15)

starts with very small values (e.g. ~ 0.01) and decreases monotonically to zero.
Intuitively, one can see that this algorithm tends to reduce the point density of the reference vectors around the Bayesian decision surfaces. Indeed, the difference between the density functions of the neighboring classes, by definition, drops to zero at the Bayes border and experimental results reported in [86] shows that LVQ1 allows to reach near optimal results compared to a pure Bayesian approach.

4.6.2.2 Type Two Learning Vector Quantization (LVQ2.1)

The LVQ2.1 algorithm is based on the assumption that two closest reference vectors m_i and m_j in the input space are initially in the wrong position. The incorrect discrimination surface is always defined as the midplane between m_i and m_j. To consider this fact, a symmetric window is defined around the midplane and the reference vectors m_i and m_j will be updated if and only if an input vector x with a known classification falls into the wrong side of the midplane (see Figure 4.12).

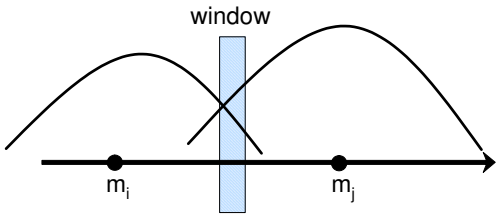


Figure 4.12: Window used for LVQ2 and LVQ 3 quantization methods

The updating rule is defined as follows:

$$\begin{cases} m_i(t+1) = m_i(t) - \alpha(t)[x(t) - m_i(t)] & \\ m_j(t+1) = m_j(t) + \alpha(t)[x(t) - m_j(t)] & \text{if } x \text{ falls into the window} \\ & \textbf{and } C_i \text{ is the nearest class} \\ & \textbf{but } x \text{ belongs to } C_j \neq C_i \\ & \text{where } C_j \text{ is the second nearest class} \\ m_k(t+1) = m_k(t) & \text{otherwise.} \end{cases} \tag{4.16}$$

As mentioned in [86], the updating rule (4.16) moves the reference vectors in such a way that the midplane moves towards the crossing surface of the class distributions, and thus asymptotically coincides with the Bayes decision border.
Regarding the window width, it must be determined experimentally from the number of available samples. With a small input data set, a width of 10%-20% of the difference between m_i and m_j has been judged as proper.
To avoid the drifting behavior of the m_i position, the updating rule (4.16) has to be applied during a limited number of steps (~ 10000) by using a small initial learning rate value (~ 0.02) that will decrease monotonically to zero.
The main problem with this algorithm concerns the path followed by each reference vector m_i that is not controlled and that can result finally in a poor representation of the input data density function. This problem is resolved by LVQ3.

4.6.2.3 Type Three Learning Vector Quantization (LVQ3)

Two kinds of bad effects can be reported when using LVQ2.1:

- because reference vectors correction in (4.16) are proportional to the difference of x and m_i, or x and m_j, the correction of m_j (i.e. the correct class) is of larger magnitude than that of m_i (i.e. the wrong class). Consequently, the distance $\|m_i - m_j\|$ is monotonically decreasing;
- it is not ensured that the reference vectors continue to approximate the density function of the input data.

In the LVQ3 variant, we suppose that a single class can be represented by several different reference vectors. The resulting new updating rule is defined as follows:

$$\begin{cases} m_i(t+1) = m_i(t) - \alpha(t)[x(t) - m_i(t)] & \\ m_j(t+1) = m_j(t) + \alpha(t)[x(t) - m_j(t)] & \text{if } x \text{ falls into the window} \\ & \textbf{and} \text{ where } m_i \text{ and } m_j \text{ are the two closest} \\ & \text{reference vectors to } x \\ & \textbf{and } x \text{ and } m_j \text{ belong to the same class} \\ & \textbf{while } x \text{ and } m_i \text{ belong to different classes} \\ & \text{where } C_j \text{ is the second nearest class} \\ m_k(t+1) = m_k(t) + \varepsilon\alpha(t)[x(t) - m_k(t)] & \text{for } k \in \{i, j\} \text{ if } x, m_i, m_j \text{ belong to the same class.} \end{cases} \tag{4.17}$$

Experiments have shown that $0.1 \leq \varepsilon \leq 0.5$ is applicable.The optimal value of ε seems to depend on the size of the window. However, the great advantage of this algorithm is the self-stabilization i.e. the placement of the reference vectors does not change by continuing learning. In [86], the author reports experimental results for a speech recognition application in which LVQ3 outperforms the two former approaches. Moreover, all LVQ methods have also exhibited a better recognition rate than a classical Bayes approach.
Note also that other variants of these classical LVQ methods have been discussed in [89].

4.6.2.4 Generalized Learning Vector Quantization (GLVQ)

More recently, the GLVQ method has been proposed in [131] as an alternative approach to classical LVQ methods to classify data in a supervised way. This new method has been motivated by observing that when using classical LVQ methods, the reference vectors diverge in some cases, degrading thus the classification ability.
The basic idea of GLVQ consists in defining a new learning rule based on the minimization of a cost function. Assume that m_1 is the nearest reference vector that belongs to the same class of the input vector x and likewise, let m_2 be the nearest reference vector that belongs to a different class from x. Let us now consider the relative difference $\mu(x)$ defined by (see Figure 4.13):

$$\mu(x) = \frac{d_1 - d_2}{d_1 + d_2}$$

where $d_i = \|x - w_i\|^2$, with $\|.\|$ denoting the Euclidian norm. $\mu(x)$ ranges from -1 to $+1$ and $\mu(x) < 0$ (resp. $\mu(x) > 0$) if x is classified correctly (resp. incorrectly).

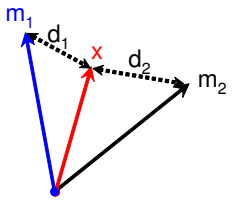


Figure 4.13: Relative differences used by GLVQ

In order to improve classification rates, $\mu(x)$ has thus to decrease for all input vectors x. A cost function S to be minimized can thus be defined by:

$$S = \sum_{i=1}^{p} f(\mu(x_i))$$

where p denotes the number of input vectors, and f is a monotonically increasing function. To minimize S, m_1 and m_2 are updated based on the steepest descent method with a small positive constant α:

$$w_i(t+1) = w_i(t) - \alpha \frac{\partial S}{\partial m_i} \tag{4.18}$$

with $i = 1, 2$. Computing the derivatives $\frac{\partial S}{\partial m_i}$ yields to the GLVQ learning rule:

$$\begin{aligned} m_1(t+1) &= m_1(t) + \alpha \frac{\partial f}{\partial \mu} \frac{d_2}{(d_1+d_2)^2}(x - m_1) \\ m_2(t+1) &= m_2(t) - \alpha \frac{\partial f}{\partial \mu} \frac{d_1}{(d_1+d_2)^2}(x - m_2). \end{aligned} \tag{4.19}$$

Three important observations must be made by observing (4.19):

- contrary to classical LVQ learning rules, the updating weight $|\triangle m_i|$ depends on $\frac{\partial f}{\partial \mu}$ which depends in turn of x;
- to decrease the error rates, the reference vectors must be updated by using input vectors around class boundaries in order to shift these boundaries toward the Bayes limit. Accordingly, $f(\mu)$ should be a non-linear monotonically increasing function and it is considered that classification accuracy strongly depends on the definition of $f(\mu)$;
- as just mentioned, the attractive and repulsive forces $|\triangle m_i|$ depends on the derivatives of μ. Consequently, the reference vectors will converge to an equilibrium state defined by these forces. Therefore, it can be considered that the convergence property depends on the definition of μ.

The GLVQ method is really a generalization of classical LVQ learning rules in the sense that a proper definition of μ and f can lead to (4.15),(4.16) or (4.17). Furthermore, it is also possible to obtain other learning algorithms by defining a different cost function, but as mentioned before, it is important to note that the convergence property depends on the definition of this cost function. Following this remark, it has been shown in [131] that the convergence property is not satisfied by LVQ2.1, contrary to the GLVQ algorithm.
Experimental results presented in [131] have shown that GLVQ outperforms classical LVQ methods for a character recognition application. Moreover, experiments have shown that the use a non-linear function $f(\mu)$ clearly improves the recognition rates compared to a linear version of f.

4.6.3 Quantitative Measures

Many mathematical studies have been reported in the literature for analyzing the properties of SOM such as convergence [20, 34, 87, 123, 124], reconstruction error, topographic error [7, 82, 83, 110, 138], etc. In this section, we propose to review some quantitative measures that can be helpful to measure the reconstruction and topographic error induced by the mapping computed by the Kohonen algorithm.

4.6.3.1 Reconstruction Error

As underlined before, the Kohonen algorithm is closely related to vector quantization and data compression. Compressing the data works by approximating each data to be compressed by a codeword with the same dimension. Of course, reaching an acceptable compression ratio leads to use less codewords than input generating thus a reconstruction error with an order of magnitude of $\|x - m_c\|$ if x designates the input vector and m_c the codeword yielding the most accurate approximation of x. The problem consists thus to find a good placement of the set M of codewords in the input space in order to minimize the overall reconstruction error E. A mathematical formulation of this yields to:

$$E(M) = \int \|x - m_x\|^2 P(x) dx \tag{4.20}$$

where $P(x)$ denotes the density function of the input data vectors x and m_x the codeword associated to x. Note that the minimization of $E(M)$ has to be reached with the constraint of a predefined limited number of codewords. The minimization of $E(M)$ with respect to the reference vectors m_i is a complicated, non-linear optimization problem, for which in most cases no closed solutions are known. For this kind of problem, iterative approximation methods are applicable. The simplest approach to minimize (4.20) with respect to m_i is to use gradient descent. Assuming that initial values $m_i(0)$ have been defined in a proper way, the reference vectors updating rule is defined by:

$$\begin{aligned} m_i(t+1) &= m_i(t) - \tfrac{\alpha}{2} \tfrac{\partial E}{\partial m_i} \\ &= m_i(t) + \alpha \textstyle\int_{s(x)=c} (x - m_c(t)) P(x) dx. \end{aligned} \tag{4.21}$$

The integration condition $s(x) = c$ restricts the region of integration to those x values for which m_c is the most suitable reference vector i.e. $\|x - m_c\| = \min_i \|x - m_i\|$.(4.21) has been firstly proposed in the LBG (or k-means) procedure [50]. Although this updating rule does not ensure the convergence toward a global minimum, the quality of the reference vectors localization is sufficiently good in many cases. However, updating a reference vector m_i requires the knowledge of the density $P(x)$ of input data, which is not available in most cases. To overcome this problem, (4.21) can be replaced by an incremental approach:

$$m_c(t+1) = m_c(t) + \alpha(x - m_c(t)) \tag{4.22}$$

where m_c denotes the codeword the most similar to x i.e. the one minimizing $\|x - m_i\|$ for all codewords m_i. By this way, for a sufficiently small constant α, the accumulation of many individual steps of (4.22) by using data vectors selected at random leads to an approximation of the integration in (4.21). At this point, it is interesting to note that (4.21) is closely related to the kernel-based learning rule (4.12) used by the Kohonen algorithm with a vanishing neighborhood function $h_{ci} = \delta_{ci}$ [123]. The SOM learning rule can thus be seen as a generalization of the vector quantization procedure classically used for data compression.
If we analyze (4.21), one can see that the shift of the reference vector m_i occurs in the direction of the center of gravity $\int_{F_c} xP(x)dx$ where F_c denotes the receptive field of the codeword m_c i.e. the set of input vectors that are well approximated by m_c. Therefore, (4.21) leads to a distribution of reference vectors in which each vector coincides with the center of gravity of the data in its receptive field. This procedure is well-known in the area of data clustering since it corresponds to the k-means algorithm. If we now make an analogy with the Kohonen algorithm, the distribution of reference vectors is somewhat different due to the use of the neighborhood function h_{ci} (see (4.12)). The average shift of a reference vector is then [123]:

$$\Delta m_i = \sum_i h_{ci} \int_{F_i} (x - m_i)P(x)dx. \tag{4.23}$$

In that case, the shift of m_i occurs in the direction of the mean center of gravity of all receptive fields F_i, the contribution of each field being weighted by the neighborhood function h_{ci}. Consequently, the learning rule induced by the use of the Kohonen algorithm does no longer lead to the minimization of the reconstruction error (4.20) but to the minimization of (4.23). Therefore, the Kohonen algorithm tries to find a tradeoff between a good representation of input data (by minimizing a modified version of the reconstruction error) and the preservation of topology (by the use of the neighborhood function).

4.6.3.2 Preservation of Topology

SOMs can be characterized by two main properties: a good representation of the density function of input data by a set of reference vectors, and the preservation of the input space topology. The quality of the placement of the reference vectors is classically measured by the reconstruction error as shown in the previous section. However, the preservation of the topology is more difficult to assess and is closely related to the dimensional conflict existing between the input space and the output space (i.e. the lattice). Roughly speaking, the topology of the input space is preserved if: (i) nearby reference vectors in the input space are mapped onto neighboring

locations in the output space and (ii) two neighboring cells i and j are associated to nearby reference vectors m_i and m_j in the input space. However, what is the meaning of "nearby reference vectors"?
Mathematically speaking, a SOM $\mathcal{S}_{A,M}$ can be defined by two mappings $\Psi_{A\longrightarrow M}$ and $\Psi_{M\longrightarrow A}$ defined by:

$$\mathcal{S}_{A,M} = \begin{cases} \Psi_{A\longrightarrow M} : A \longrightarrow M;\ i \in A \longmapsto m_i \in M \\ \Psi_{M\longrightarrow A} : M \longrightarrow A;\ x \in M \longmapsto i^*(x) \in A \end{cases} \tag{4.24}$$

where $i^*(x)$ denotes the BMU associated to the input vector x. Let us recall that this cell is defined by:

$$i^*(x) = arg\ \min_i \|m_i - x\|\ \forall i = 1,2,\ldots,n$$

where n denotes the number of cells in the lattice A.
Now, two reference vectors are said to be adjacent in M (what we called "nearby" previously) if their associated receptive fields R_i and R_j defined by the masked Voronoi polyhedra are adjacent in M i.e. $R_i \cap R_j \neq \emptyset$. These receptive fields are defined by:

$$R_i = \{x \in M,\ \|x - m_i\| \leq \|x - m_j\|\ \forall j \in A\}. \tag{4.25}$$

Regarding the neighborhood property within the lattice A, two cells i and j are said to be adjacent in A if and only if they are nearest neighbors in the lattice A in the sense of a norm defined in A (usually, the Euclidian norm is used).
Many measures have been defined in the literature to quantify the topology preservation [7, 82, 83, 110, 138] that can be used to exhibit a dimensional conflict between the input and the output spaces.
One of the earlier and most popular measure is the *topographic product* introduced in [7] that quantifies the preservation of neighborhood relationships in maps between spaces of possibly different dimensionality. To justify the topographic product idea, let $n_k^A(i)$ be the k^{th} nearest neighbor of the cell i in A, in the sense of a distance d^A expressed in the lattice A. In the same way, let $n_k^M(i)$ be the k^{th} nearest neighbor (among all reference vectors) of the vector m_i but with a distance measure d^M expressed in the input space M. From this quantities, we define the ratios:

$$\begin{aligned} Q_1(j,k) &= \frac{d^M(m_j, m_{n_k^A(j)})}{d^M(m_j, m_{n_k^M(j)})} \\ Q_2(j,k) &= \frac{d^A(j, n_k^A(j))}{d^A(j, n_k^M(j))} \end{aligned} \tag{4.26}$$

From (4.26), we have $Q_1(j,k) = Q_2(j,k) = 1$ if the nearest neighbors of order k in the input and output space coincide. Any deviation of Q_1 and Q_2 from 1 denotes a violation of the nearest neighbor ordering in the map. Q_1 and Q_2 are then combined yielding to:

$$P_3(j,k) = \left(\prod_{l=1}^{k} Q_1(j,l)Q_2(j,l)\right)^{\frac{1}{2k}}. \tag{4.27}$$

Thanks to the inverse nature of Q_1 and Q_2, neighborhood violations are detected by $P_3 \neq 1$. More precisely, the deviation of P_3 above or below 1 indicates whether the embedding dimension of A is too large or too small, respectively.
Finally, the topographic product P is defined by averaging P_3 for all values of j and k:

$$P = \frac{1}{n(n-1)} \sum_{j=1}^{n} \sum_{k=1}^{n-1} log(P_3(j,k)). \tag{4.28}$$

This last formulation is motivated by observing that one are mainly interested in deviations from 1.
More recently, Villman *et al.* [138] explained that the preservation of the topology depends on the lattice structure A but also on the form of the manifold M. However, this last parameter is often unavailable and the lattice structure A is chosen empirically. Without considering the shape of the data manifold M, the authors underline that none topology preservation measure is able to reach satisfying results in the case of non linear data manifolds. To overcome this problem, they propose a new measure called *topographic function* that explicitly considers the structure of the input data manifold. For a rectangular lattice A, the topographic function $\Phi_A^M(k)$ is defined by:

$$\Phi_A^M(k) = \begin{cases} \frac{1}{n}\sum_{i\in A} f_i(k) & \text{for } k > 0 \\ \Phi_A^M(-1) + \Phi_A^M(1) & \text{for } k = 0 \\ \frac{1}{n}\sum_{i\in A} f_i(k) & \text{for } k < 0 \end{cases} \tag{4.29}$$

where n denotes the number of cells in A, and the function f_i is defined by (for $k > 0$):

$$\begin{cases} f_i(k) = \sharp\{j, \|i-j\|_{max} > k,\ d^{\mathcal{D}_M}(i,j) = 1\} \\ f_i(-k) = \sharp\{j, \|i-j\| = 1,\ d^{\mathcal{D}_M}(i,j) > k\} \end{cases}$$

in which:

- $\|.\|_{max} = \overset{d_A}{\underset{j=1}{max}} |(.)_j|$ with d_A denoting the dimension of the lattice A;
- $\|.\|$ denotes the classical Euclidian norm;
- $\mathcal{D}_M$ is the Delaunay triangulation obtained from the set of reference vectors m_i. This triangulation induces the metric $d^{\mathcal{D}_M}$ in which the distance between two reference vectors is determined by the length of the shortest path in the Delaunay graph.

$f_i(k)$ (resp. $f_i(-k)$) measures the topology preservation obtained by $\Psi_{M\longrightarrow A}$ (resp. $\Psi_{A\longrightarrow M}$) and $\Phi_A^M(0) = 0$ if and only if the SOM mapping perfectly preserves the topology. Moreover, the largest values $k^+ > 0$ for which $\Phi_A^M(k^+) \neq 0$ holds yields the range of the largest fold if the effective dimension of the data manifold M is larger than the dimension d_A of the lattice A. Conversely, the smallest $k^- < 0$ for which $\Phi_A^M(k^-) \neq 0$ holds yields the range of violations of topology preservation if the effective dimension of the data manifolds M is smaller than the dimension d_A of the lattice A. As a result, the values of k^+ and k^- give information about the degree of the dimensional conflict. Small values indicates only a local dimensional conflict whereas large values indicate the global character of the conflict.

It is important to note that in the case of a linear data manifold M, the topographic product (4.28) and the topographic function (4.29) give the same results. However, when M becomes non linear, only the topographic function exhibits a correct behavior. In [138], the authors have also extended the definition of the topographic function to more general lattice structures.
In [83], the author observes that although the topographic function is a good measure, it does not result in a scalar value but in a function depending on k, increasing thus the comparison between two mappings. To overcome this problem, the author proposes a simple measure called the *topographic error* and that averages the number of times the BMU and the second BMU are not adjacent. This measure is defined by:

$$\varepsilon_t = \frac{1}{p}\sum_{k=1}^{p} u(x_k) \text{ where } u(x_k) = \begin{cases} 1 & \text{if BMU and second-BMU are not adjacent} \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, ε_t measures the local discontinuity in the sense that when two similar input vectors are mapped on distant cells, the mapping is no more continuous.
In [110], an other measure is proposed, also based on the masked Delaunay triangulation. Here, the topology is considered as preserved if the Delaunay triangulation coincides with the output network shape. In this new measure, the topology preservation at the cell i is defined by two ratios:

$$\Upsilon^{+}(i) = \frac{\sharp\{j,\ d^{(\mathcal{D}_M)}(m_i,m_j)=1 \text{ and } d^{(A)}(i,j)=1\}}{\sharp\{j,\ d^{(\mathcal{D}_M)}(m_i,m_j)=1}$$
$$\Upsilon^{-}(i) = \frac{\sharp\{j,\ d^{(\mathcal{D}_M)}(m_i,m_j)=1 \text{ and } d^{(A)}(i,j)=1\}}{\sharp\{j,\ d^{(A)}(m_i,m_j)=1}$$

where $d^{(\mathcal{D}_M)}$ (resp. $d^{(A)}$) denotes the distance computed in the Delaunay graph associated to the set of reference vectors (resp. in the output lattice). Both distances are graph-based in the sense that they represent the number of edges to follow in order to link two nodes. In the case where $\Upsilon^{+}(i)$ and $\Upsilon^{-}(i)$ becomes 1.0 for all neurons, then the masked Delaunay graph of the input space coincides with the lattice structure. An important result of this method is that the Kohonen algorithm using rectangular lattice is not able to reach the topology preservation for uniform distribution contrary to the use of an hexagonal lattice.

4.6.4 Dynamic Architectures

As we saw up to now, one of the limitation of SOM is that the structure of the lattice has to be predetermined prior to learning. The definition of the lattice structure includes the dimension of the network, the number of cells in each dimension (and thus the total number of cells) and the topology of the map (rectangular, hexagonal, etc.). This limitation conducts inevitably the user to run several simulations with different network structures in order to pick the optimal network. Moreover, the use of a fixed and regular topology (such as a rectangular one) cannot perfectly reflect the structure of input data or disjoint clusters that may exist in the input space. For example, discontinuities in the input space may appeared bridged in the map [15]: the map may have connections that span disjoint clusters or it may have reference vectors located within discontinuities where the probability density function of input data is zero.

Regarding the dimension of the lattice, 2D maps are generally retained for visualization purposed as well as ease in implementing SOM.
To overcome the problems discussed above, some dynamic map architectures have been proposed [2, 15, 47, 90, 120] in order to provide a better representation of input data. The main properties of these alternative structures are their ability to grow and change their structure in an incremental way.
In this section, we present some of the most popular dynamic architectures proposed in the literature. Section 4.6.4.1 reviews some dynamic flat 2D topologies, whereas section 4.6.4.2 present the hierarchical variants.

4.6.4.1 Flat Topologies

In [15], the authors propose a method called IGG (*Incremental Grid Growing*) that builds in a dynamic way a lattice structure in which non-convexities, disconuities and clusters present in the input data set are explicitely represented in the 2D output lattice. To reach these goals, the proposed method is incremental and starts with only four connected nodes with reference vectors randomly chosen among training data. The learning phase is performed on these four nodes according to the basic SOM algorithm and each cell i maintains the quantization error qe_i related to its own reference vectors. In the IGG approach, only nodes located at the map boundaries are allowed to grow and at each iteration of the algorithm, the boundary cell i_{err} with the highest quantization error is expanded toward all free neighboring locations (see Figure 4.14).

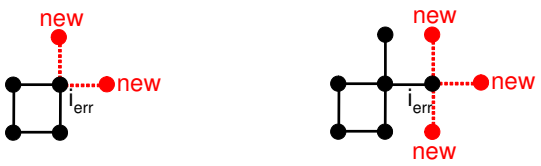


Figure 4.14: Growing rule used by the IGG method

The new added cells are connected to i_{err} and the reference vectors of these cells are interpolated from neighboring reference vectors. Initially, the new nodes are thus connected to the structure through i_{err}. Then, when the structure continues to organize, the new nodes may develop reference vectors that are close to their neighbors counterpart to which they are not connected. In this case, it is desirable to connect non connected cells with close reference vectors. Conversely, connected nodes with distant reference vectors must remove their connection. To reach these goals, IGG defines two thresholds Θ_1 and Θ_2 used as follows:

- if $\|m_i - m_j\| > \Theta_1$, the connection between i and j is removed;
- if $\|m_i - m_j\| < \Theta_2$, a connection between i and j is created.

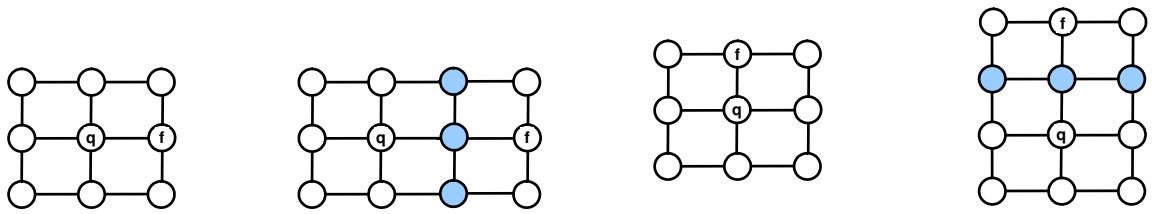
By adding nodes at the perimeter of the lattice, the map is able to develop an arbitrary topology. Moreover, adding nodes only in areas that inadequately represent the input data encourages the map to develop only those topological structures that are actually present in

the input space. Finally, disconnecting nodes that span discontinuities allows clusters to be separated before continuing to develop independently of each others.
In [2], the author present a method called GSOM (*Growing Self-Organizing Map*) inspired from IGG but that brings some improvements to the former method. The main improvements are the use of an adaptative learning rate, the consideration of non boundary nodes and the definition of a spread factor controlling the development of the map. In the GSOM method, the lattice starts with four connected nodes as in the basic IGG approach. However, the authors underline that using a small number of nodes has a direct impact on the learning rate $\alpha(t)$. Indeed, a same BMU is selected for very different input vectors, making the associated reference vector oscillate in the input space. To overcome this problem, an adaptative rule is proposed to set the learning rate value: the learning rate depends on the number of cells in the lattice. The higher is the number of cells, the higher the learning rate. The second improvement considers the non boundary cells holding a localized quantization error greater that the threshold GT used to decide if a node has to grow. In the basic IGG approach, such a node cannot grow because it is not located on the map perimeter. In GSOM, the idea consists in artificially reducing its associated quantization error while increasing the quantization error of the neighboring nodes. By this way, non boundary nodes have the ability to indirectly initiating node growth. The last improvement concerns the definition of a spread factor SF that can be used to control the spread of the map. As mentioned before, a threshold GT is used to decide when a node can initiate the growing process. Therefore, a large GT results in an abstract picture of input data with a small number of nodes whereas a small GT results in a more spread map. However, GT depends the dimension of input data as well as the number of cells, both parameters depending on the feature space and the time index since the map grows dynamically. The use of such a threshold is thus very difficult for an analyst who wants to control the map growth. Moreover, it is also very difficult to compare maps of several data sets since GT cannot be compared over different data sets. To tackle this problem, a spread factor SF is defined ($0 \leq SF \leq 1$):

$$GT = -D.ln(SF)$$

where D denotes the dimension of input data. Now, instead of having to provide the threshold GT, the analyst has just to provide a scalar value from which the associated GT will be deduced.
The *Growing Grid* method discussed in [47] is different of IGG and GSOM in the sense that the lattice structure remains rectangular during the organization process. This method defines an incremental approach that converges toward a ratio with/height well suited to input data. This model uses locally accumulated statistical measures to decide where to insert new rows or columns into the initially small lattice. The properties of the new inserted cells are then interpolated from information extracted in the neighboring nodes. This algorithm is based on the use of a resource variable τ_i for each cell i that maintain the number of times the cell i won the competition. After λ steps of the learning adaptation rule, the algorithm selects the cell q that won the most often the competition i.e. such that $\tau_q \geq \tau_i \ \forall i$. The cell q denotes a dense area in the input space that should be represented by more cells. To reach this goal, the neighbor f of q with the most different reference vector is selected and a row or column is inserted between q and f (see Figure 4.15).

The reference vectors of the new cells are then interpolated from the neighboring reference



(a) (b)

Figure 4.15: Growing rule used by the Growing Grid method. Column insertion rule (a) and row insertion rule (b).

vectors. The algorithm stops when the desired number of cell is reached or when each cell represents approximatively the same amount of input signals.

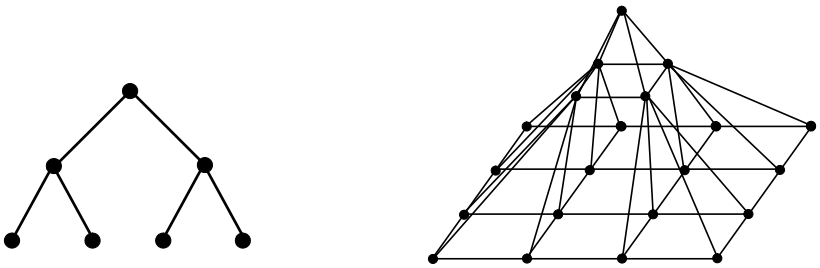
4.6.4.2 Hierarchical Topologies

The second kind of dynamic architectures are composed of several SOM layers arranged in a tree data structure. They provide thus a hierarchical ordering of cells and have been motivated by the following observations:

- in the basic SOM using flat topologies, the number of cells needed to provide an efficient representation of input data increases exponentially with the dimension of the input space, leading thus to a computational complexity of $O(n)$ where n is the number of cells. The use of a tree reduces this complexity to $O(log(n))$;
- by using flat topologies, hierarchical relations between input features are very hard to discern although these properties are of prime importance for some applications such as data mining, pattern recognition, etc.
- finally, a hierarchical representation of input features allows to tune the desired level of resolution: a coarse resolution is obtained at the higher levels (i.e. the input data are represented by a small number of cells) whereas the resolution is increased at the bottom levels. By this way, we obtain a multi-resolution scheme for feature selection in which at different hierarchical levels, a different level of generalization is obtained.

The first hierarchical structure proposed in the literature was the TSTFM (*Tree Structured Topological Feature Map*) algorithm discussed in [90]. This algorithm is an extension of the well-known TSVQ (*Tree Structured Vector Quantization*) algorithm [50] used for vector quantization. In TSVQ, the codewords are arranged in a tree data structure and the search of the best matching codeword is performed in stages. In each stage, a substantial subset of candidate codewords is eliminated from consideration. In TSTFM, lateral connections between cells are considered (as in the the basic Kohonen network) conducting to the construction of a pyramid data structure instead of a classical tree (see Figure 4.16).

As in TSVQ, the training takes place at the root node. When the organization of a cell at any hierarchical level is completed, the plasticity of that cell is frozen and the procedure is



(a) (b)

Figure 4.16: Tree data structure used by TSVQ and pyramidal data structure used by TSTFM

applied for the units on the subsequent layers. The BMU selection and the reference vectors updating rule are however different in TSTFM from those applied in TSVQ. In TSTFM, the BMU selection is performed by a tree search as in TSVQ but at each step, a lateral search is also done to find the BMU. Regarding the updating rule, the nearest neighbors of the BMU are updated similarly to the standard SOM algorithm. This kind of hierarchical architecture has, for example, been successfully applied in the design of the PicSOM [91]. PicSOM is a very interesting CBIR (*Content-Based Image Retrieval*) system based on MPEG-7 descriptors [104] and that makes use of SOM for classifying images into semantic categories.
In [120], the author propose an algorithm called GHSOM (*Growing Hierarchical Self Organizing Map*) that combines a hierarchical structure and the *Growing Grid* method [47] discussed before. In this approach, each independent SOM has a dynamic rectangular topology (as specified in the Growing Grid algorithm) but each cell can also grow in the vertical dimension, creating thus a new map in the next hierarchical level. As a result, the final map is composed of several independent growing SOMs arranged in a tree data structure (see Figure 4.17).

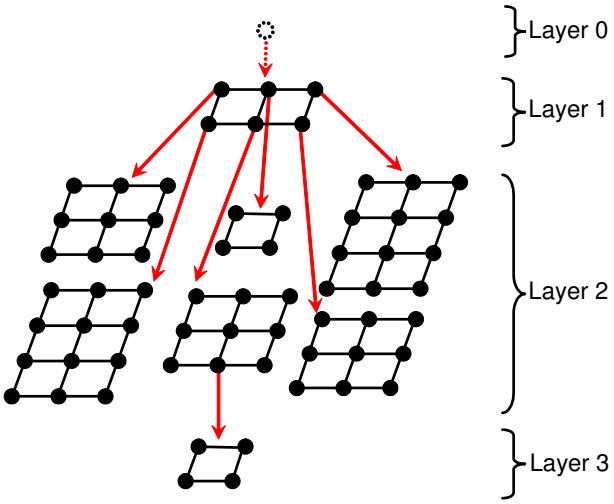


Figure 4.17: Hierarchical lattice structure obtained by the GHSOM method

GHSOM is composed of three main steps. In the first step, the network is built with 2×2 connected cells. At the root level, a virtual cell is created whose the reference vector is the mean of all input vectors. This cell maintains its associated quantization error qe_0 that will be used in the next level to govern the growing phase. The reference vectors of the first map composed of 2×2 connected cells are chosen in a random way. In the second step, the map organization is performed according the the standard growing grid method. This step is performed until the quantization error associated to the current map is below a certain fraction of the quantization error associated to the parent node. When the learning phase is completed at a given hierarchical level, each cell is analyzed to decide whether it must be splitted in a subsequent level. This decision is made on the basis of the error quantization associated to the analyzed cell. If this error is greater than the authorized threshold, the cell is splitted into a new small map composed of 2×2 connected cells. However, the initialization of reference vectors is no more random to avoid topological distortion between neighboring maps. In order to reach a global orientation of maps in the different hierarchical levels, the orientation of each map must refer to the orientation of input data falling in the receptive field of the parent node. As a result, GHSOM obtains a hierarchical architecture in which rectangular SOMs with possibly different sizes are present at each level.

4.6.5 Conclusion

In this section, we have proposed a first overview of the SOMs by discussing several aspects such as the basic Kohonen algorithm, its variants, the quantitative measures and the dynamic lattice structures. As we saw, SOMs are a very efficient tool to perform vector quantization while preserving the topology of the input space. In consequence, they are applied in many research fields that we cannot review in detail here due to the reduced space. However, we invite the interested reader to refer to [75, 114] to get an exhaustive lists of approaches related to SOMS. Of course, this document has not discussed all interesting variants of SOMs, such as the use of SOMs for non vectorial data or the learning of SOMs parameters, but it will evolve with time to resolve this drawback.

4.7 Cluster Validity Analysis

An unsupervised learning procedure is usually more difficult to assess than a supervised one. Several questions can be asked regarding the application of clustering methods:

- Are there clusters in the data?
- Are the identified clusters in agreement with the prior knowledge of the problem?
- Do the identified clusters fit the data well?
- Are the results obtained by a method better than those obtained by another?

The first question concerns the *cluster tendency* of the data and should in principle be answered before attempting to perform clustering, using specific statistical tests. Unfortunately, such tests are not always very helpful and require the formulation of specific test hypotheses.

The other questions concern the analysis of cluster validity and can only be answered after application of clustering methods to the data. According to [68], one can distinguish between three types of validation procedures:

- *External* validation consists in finding an answer to the second question above and can only be performed when prior knowledge of the problem is available. The prior knowledge may concern general characteristics of the clusters (e.g. expected compactness) or relations between specific items (e.g. items A and B should belong to a same cluster and item C to a different one). Sometimes this knowledge is confirmatory but not prescriptive.

- *Internal* validation concerns the third question above and is based on an evaluation of the "agreement" between the data and the partition. In the following we give a few examples of validity indices that were put forward in the literature for some of the above categories of clustering methods. Note that the definitions of these internal validity indices usually make their direct optimization intractable.

  In [14] several indices for crisp clustering are evaluated: the modified Hubert's statistic, related to the alignment between the dissimilarity matrix and the crisp partition matrix, the Davies-Bouldin index, roughly defined as the ratio between within-cluster scatter and between-cluster separation, and Dunn's index with several alternative definitions (some of which are introduced in [14]) for the diameter of a set and for the distance between sets.

  For fuzzy partitional methods, internal validity indices should take into account both the data items and the membership degrees resulting from clustering. The *average partition density* in [48] is obtained as the mean ratio between the "sum of central members" (sum of membership degrees for the items close to the prototype) of each cluster and the volume of the cluster. The Xie-Beni index put forward in [142] is the ratio between the average intra-cluster variation and the minimum distance between cluster centers, and is thus related to the ratio between intra-cluster and inter-cluster variance.

  Among the validity indices suggested for density-based clustering methods, we mention the two in [42]: the first one measures the variation of cluster labels in the neighbourhood of data items, the other evaluates the density on the path between data items.

- *Relative* comparisons attempt to provide an answer to the fourth question above and are usually the main application of the indices defined for the internal validation. Such comparisons are often employed for selecting good values for important parameters, such as the number of clusters.

The question of cluster vanlidity analysis is also discussed in some detail in section 7.8 where cluster validity is a key issue in determining the usefulness of feature subsets.

4.8 Conceptual Clustering

Classical clustering methods only create clusters but do not explain why a cluster has been established. Conceptual clustering methods built clusters and explain why a set of objects confirm

a cluster. Thus, conceptual clustering is a type of learning by observations and it is a way of summarizing data in an understandable manner [31]. In contrast to hierarchical clustering methods, conceptual clustering methods built the classification hierarchy not only based on merging two groups. The algorithmic properties are flexible enough in order to dynamically fit the hierarchy to the data. This allows incremental incorporation of new instances into the existing hierarchy and updating this hierarchy according to the new instance. Known conceptual clustering algorithms are Cluster/S [127], Cobweb [31], UNIMEM [98] and conceptual clustering of graphs [115].

4.8.1 Concept Hierarchy and Concept Description

A concept hierarchy is a directed graph in which the root node represents the set of all input instances and the terminal nodes represent individual instances. Internal nodes stand for sets of instances attached to that nodes and represent a superconcept. The super concept can be represented by a generalized representation of this set of instances such as the prototype, the mediod or a user selected instance. Therefore a concept C, called a class, in the concept hierarchy is represented by an abstract concept description and a list of pointers to each child concept $M(C)=\{C_1, C_2,..., C_i,..., C_n\}$, where C_i is the child concept, called subclass of concept C.

4.8.2 Category Utility Function

Category utility can be viewed as a function of the similarity of the objects within the same class (intra-class similarity) and the similarity of objects in different classes (inter-class similarity). The goal should be to achieve high intra-class similarity while low inter-class similarity. The category utility function can be based on:

- a probabilistic concept [31] or
- a similarity based concept [115].

The category utility function in COBWEB is defined based on the probabilistic concept. The category utility function is defined as the increase in the expected number of attribute values that can correctly guessed.

4.8.3 Application in Image Processing and Understanding

Conceptual Clustering has been used for differernt types of problems in image processing and understanding. For image segmentation it was used in [31]. For learning the case-base index for high-level image interpretation it was used in [115]. For learning general forms of objects it was used in [119].

4.9 Ontology Derivation

Ontologies are popular in a number of fields such as knowledge engineering and representation, qualitative modeling, database design, information modeling and integration, object-oriented

analysis, information retrieval and extraction, natural language processing, knowledge management, agent systems, and more [56]. In addition to those fields, research analyst companies report on the critical roles of ontologies in areas such as, browsing and searching for e-commerce, and for interoperability for facilitation of knowledge management and configuration [106]. Ontologies are becoming essential in many on-line applications including Yahoo!, Google, Amazon, and eBay.

However, the problem of their construction and engineering remains not to be completely solved and their development today is more craft than a science. Automated ontology construction tools provide little support to knowledge acquisition. Therefore, the ontology construction process becomes time consuming and this leads to the fact that their wide usage has been limited.

Several approaches to the ontology acquisition from documents exist, each one with a different perceptive and from a different point of view. Some of them are completely automated, some are semi-automated. This report has as main purpose to shed some light in the ontology acquisition from documents domain, by presenting the most important and most prominent approaches along with some details about them.

4.9.1 Ontology Definition

The Artificial-Intelligence literature contains many definitions of an ontology.

The term is borrowed from philosophy, where the ontology is a systematic account of Existence. According to The Free On-line Dictionary of Computing[41], an ontology is an explicit formal specification of how to represent the objects, concepts/classes and other entities that are assumed to exist in some area of interest and the relationships that are held among them.

The following definition[54] is widely used.

An ontology is the specification of conceptualizations, used to help programs and humans share knowledge. In this usage, an ontology is a set of concepts/classes - such as things, events, and relations - that are specified in some way, such as specific natural language, in order to create an agreed-upon vocabulary for exchanging information.

A more formal definition, is given by Maedche and Staab[103], an ontology can be described by a 5-tuple consisting of the core elements of an ontology such as concepts/classes, relations, an hierarchy, a function that relates classes non-taxonomically and a set of axioms. Then the ontology $O = \{C, R, H, f, A\}$ consists of:

- C, classes and R, relationships, are two disjoint sets.
- H, class hierarchy $H{:}H \subseteq C \times C$ also called taxonomy. $H(c_1, c_2)$ means that c_1 is a subconcept or a subclass of class c_2.
- f , a function that relates classes non-taxonomically: $f : R \rightarrow C \times C$
- A, a set of ontology axioms expressed in appropriate logical language

4.9.2 Ontology Acquisition From Text

The understanding of ontologies is different among different researchers, this therefore causes them to differ in their point of views and approaches. However, the basic idea of all of these approaches and views is quite similar and could be simply describe as:

Taking a set of textual documents, running a sophisticated tool and getting an ontology as the result.[18]

While information contained in textual documents could be understood as flat representation, knowledge represented in ontology is hierarchical. The way from flat to hierarchy, from information to knowledge is not so trivial and incorporates many scientific fields such as Natural Language Processing, Machine Learning, Information Extraction, Clustering, Classification and so on.

Two ontologies describing the same domain might be completely different and in addition, if two people are given the same set of documents and are asked to outline an ontology, their results will certainly be different. One of them might concentrate on one hierarchical aspect while the other on another one. The question here is what information, concepts and relationships we are mostly interested in from a given set of documents. Therefore, the tool should explicitly know or should be built for a specific task, otherwise it will not be able to determine what is important for us and what is not.

Splitting the basic idea down, many different problems have to be faced. Some of them are listed here:

- Recognize concepts/classes
- Define slots for each class
- Discover taxonomic relationships
- Discover non-taxonomic relationships
- Extracted ontology versus the truth
- Ontology population with instances
- Ontology refinement and maintenance

The vast number of tasks that need to be undertaken in order to fulfill the basic idea has lead the researchers to concentrate on one aspect at a time.

4.9.3 Towards Ontology Acquisition

There are many trends in ontology construction, each one concentrating in specific methods. The first one discovers non-taxonomic relations from text and enhances an already taxonomic hierarchy based on association rules. The second one discovers taxonomical relationships and places discovered concepts into hierarchy, based on clustering of sub-categorization frames. The third one, is based on modified self-organizing maps and the last two ones use statistical methods.

4.9.3.1 Text-To-Onto

The first approach [103, 101] deals with discovering non taxonomic relationships from text and enhancing already defined taxonomic hierarchy. The Text-to-Onto[102] system uses shallow parsing as a natural language module. This module consists of tokenizer, morphological and lexical processing and chunk parsing that uses lexical resources to produce mixed syntactic/semantic information. The output of this module is then XML-tagged text.

The learning component is for discovering non-taxonomic relationships. The system is based on discovering generalized association rules[135]. The mining generalized association rules is an extension to mining association rules technique - Apriori algorithm. While the original Apriori algorithm considers all items to be completely disjoint without any hierarchy (*milk* and *bread* are considered to be as similar as *Pepsi* and *Sprite* or *bicycle* and *tea*), the extension towards generalized association rules considers *Pepsi* and *Sprite* to be *soda*, and *soda* is considered to be a *beverage*; *milk* and *bread* is *food* while *bicycle* and *tea* share only one class - *item*.

Taking the hierarchy of items might result in getting association not just between instances (items at the lowest level - *"crunchy chips"* and *"diet coke"*) but also between classes (*snack* and *soda*) or between a class and an instance (*"crunchy chips"* and *"soda"*). Therefore, the result of their learning module is a set of couples/classes that is understood as a relationship between them. However, there are two issues that need to be solved. One, is that the number of irrelevant rules is very high among a small number of interesting ones. The other problem is, that relationships are unlabeled. The algorithm extracts some rules between them, i.e. *snack* and *soda* but one does not know what this relationship is, nor knows what direction it goes. Having only unlabeled relationships is not sufficient enough.

In addition to taxonomic relationship that is needed prior to relationship extraction, the system also requires a lexicon. The lexicon defines what class a particular item is part of. For example, instances such as *Pepsi, Coke, Sprite* are sodas; *Hilton, Marriott, Balagio* are hotels; *Bancha, Earl Gray, Chinese powder* are teas. A lexicon is needed in order for the learning module to understand that for example *Marriott* is a hotel.

Text & Processing Management Component The ontology engineer uses the Text & Processing Management Component to select domain texts exploited in the further discovery process. He/she chooses among a set of text (pre-)processing methods available on the Text Processing Server and among a set of algorithms available at the Learning & Discovering component. The former module returns text that is annotated by XML and this XML-tagged text is fed to the Learning & Discovering component.

Text Processing Server. The Text Processing Server may comprise a broad set of different methods. In Text-to-Onto project, it contains a shallow text processor based on the core system (Saarbrücken Message Extraction System)[112]. SMES is a system that performs syntactic analysis on natural language documents. In general, the Text Processing Server is organized in modules, such as a tokenizer, morphological and lexical processing,and chunk parsing that use lexical resources to produce mixed syntactic/semantic information. The results of text processing are stored in annotations using XML-tagged text.

Lexical DB & Domain Lexicon. Syntactic processing relies on lexical knowledge. In Text-to-Onto, SMES accesses a lexical database which is used for lexical analysis and chunk parsing. The domain-specific part of the lexicon (abbreviated "domain lexicon"; cf. Left lower

part) associates word stems with concepts available in the concept taxonomy. Hence, it links syntactic information with semantic knowledge that may be further refined in the ontology.

Learning & Discovering component. The Learning & Discovering component uses various discovering methods on the annotated texts, e.g. term extraction methods for concept acquisition. It uses the learning algorithm for discovering generalized association rules already described. Conceptual structures that exist at learning time (e.g. a concept taxonomy) may be incorporated into the learning algorithms as background knowledge. The evaluation of the applied algorithms is performed in a submodule based on the results of the learning algorithm.

Ontology Engineering Environment. The Ontology Engineering Environment ONTOEDIT, which is a submodule of the Ontology Learning Environment TEXT-TO-ONTO supports the ontology engineer in semi-automatically adding newly discovered conceptual structures to the ontology.

4.9.3.2 ASIUM

Asium[38, 37] is able to learn semantic knowledge from text. In this context it means extracting concepts/classes and putting them into taxonomic relationship. It is a semi-automatic system meaning that user's control is needed in the process. Asium learns semantic knowledge and ontologies in the form of subcategorization frames of verbs. A sub-categorization frame in this context is defined as:

<verb> <preposition or syntactic role: headword> <preposition or syntactic role: headword> . . .

For example, a sub-categorization frame of the sentence: *My father travels by car* is: *<to travel> <subject: father> <by: car>*. The system uses a stop list and it only takes headwords into consideration, So all articles, adjectives, etc. (*a, the, my, your, nice, beautiful, etc.*) are considered noise and are ignored, due to them still being believed to be preserved semantic information. Moreover, syntactic parser Sylex identifies whether headwords are expressions, i.e. *double decker, Ford Escort* or single words. The syntactic parser gives all possible frame interpretations of sentences and ASIUM uses all of them for this approach to avoid a very time consuming hand disambiguation step while still giving a good outcome.

Once each sentence has been instantiated into a frame the learning component takes them as input and learns an ontology. This step incorporates unsupervised clustering (bottom-up) and relies on the following assumption:

Headwords occurring after the same preposition or syntactic role, and with the same verbs represent the same concept.

For example from *<to travel> <subject: father> <by: car>* and *<to travel> <subject: father> <by: train>* one can conclude that *car* and *train* represent the same concept, i.e. *motorized vehicle.*

This assumption is implemented in two steps. The first step gathers headwords that occur in the same contexts such as with the same verb and the same preposition or syntactic role. The second one builds synthetic frames according to verbs of subcategorization frames and assigns number of occurrence in the given context. For example, from the following instantiated frames:

<to travel> <subject: father> <by: car> <to travel> <subject: mother> <by: train>
<to drive> <subject: friend> <object: car>
<to drive> <subject: colleague> <object: motorbike>

<to drive> <subject: friend> <object: motorbike>

Asium might create synthetic frames, one per verb:

<to travel> <subject: [father(1), mother(1)]> <by: [car (1), train(1)]>

<to drive> <subject: [friend(2), colleague(1)]> <object: [car(1), motorbike(2)]>

At this stage clustering comes into play. The clustering is based on the distance between two clusters. In this context a cluster is represented as a list of headwords, for example *[car(1), train(1)]*. Overlapping clusters are aggregated into a new cluster. Thus, clusters that contain the same headwords with the same frequencies are considered to be similar - their distance is zero. On the other hand, the distance of clusters that do not share any headword is the highest, equal to 1. The clustering algorithm is very simple and could be briefly described as an examination of each possible couple of clusters, aggregating those pairs that are the most similar(a threshold value is used for distance pruning) and repeating the same step over and over again until it is not able to aggregate any pair anymore. It is important to understand that aggregated are the headwords of two clusters (no frames). For example *[car(1), train(1)]* and *[car(1), motorbike(2)]* might be aggregated to form new cluster called *motorized vehicle8.* After the aggregation, the new cluster is propagated through all the synthetic frames, meaning that every occurrence of *[car(1), train(1)]* and *[car(1), motorbike(2)]* will be replaced with *motorized vehicle.* At this stage, a user is asked to accept or reject the aggregation to be propagated. For instance aggregation might yield new cluster *[car(1), train(1), bike(1), motorbike(2)]*. While this cluster is certainly good for a frame <*to travel*>, it is no good for <*to drive*> since everyone knows that a bike is not drivable because it is not a motorized vehicle.

In this description a concept/class of a building ontology is a cluster. Therefore at each level of clustering new classes are introduced. One can observe that clustering, which at each level creates only pairs, might lead to enormous number of useless classes. Asium has however a post-processing phase in which it removes all useless classes. This approach might be a big help in ontology construction in a narrow specific domain but it might not be very useful in a general one. In case of a frame such as <*to present*>, <*to give*> or <*to perform*> the set of headwords might differ resulting in the user being asked to accept or reject too many aggregations that he will be rejecting.

4.9.3.3 Automatic Ontology Derivation From Text (Khan, Luo and Yen, 2002)

Khan, Luo and Yen[80] describe a system which is based on simple clustering of documents based on modified self-organizing maps (SOM)[55] with the extension of topic tracking. The system is then capable of clustering documents and labeling clusters. It also uses Wordnet[109], a general ontology lexicon database, as a tool for labeling.

According to their method ontologies are constructed automatically in bottom up fashion. For this, a hierarchy is constructed using some clustering algorithms. Recall that if documents are similar to each other in content they will be associated with the same concept in ontology. In the next step, a concept for each node in the hierarchy is assigned. For this, two types of strategy are deployed and a bottom up concept assignment mechanism is adopted. First, for each cluster consisting of a set of documents, a topic is assigned based on a modified Roccchio algorithm for topic tracking[71]. However, if multiple concepts are candidate for a topic, an intelligent method for arbitration is proposed.

Next, to assign a concept to an interior node in the hierarchy, WordNet, the aforementioned linguist ontology is used. Descendant concepts of the internal node will also be identified in WordNet. From these identified concepts and their hypernyms a more generic concept can be identified that can be assigned as a concept for the interior node.

With regard to hierarchy construction, the automatic ontology construction is based on a self-organizing tree which builds a hierarchy from top to bottom. To achieve this a new hierarchical, growing self-organizing tree algorithm (HGSOT), and a modified self organizing tree (MSOT) algorithm are proposed. Both algorithms construct hierarchy with better precision and recall than a hierarchical agglomerative clustering algorithm.

4.9.3.4 ADMI 2001

This is a Bowie State University project[100] in accomplice with University of Maryland, Baltimore County, and the National Security Agency, who are the sponsors for this project. This system constructs a domain specific ontology from text documents, presented to it as a training set[134]. The documents are read, processed, and a graph-structured ontology is produced. This ontological structure can be viewed and modified with graphical user interface (GUI).

In the process of achieving this goal, contemporary statistical methods of information retrieval such as Boolean, extended Boolean and vector space approaches have been studied[53]. Of all these approaches vector space approach has been found to be the most efficient method. It describes each document as a set of terms. This set defines the document space such that each distinct term represents one dimension in that space[128]. Each document in document space is defined by the weights, or in other words frequencies, of the terms that represent it.

Another vector space approach called Latent Semantic Indexing (LSI)[27] attempts to catch term-term statistical references by replacing the document space with lower-dimensional concept space. LSI accomplishes this by using Singular Value Decomposition (SVD), a method of matrix decomposition. The effectiveness of SVD as compared to other techniques is described in[60].

The research team of this project is concentrated on statistical analysis methods, as compared to heuristic and rule based methods because of their simplicity and because these methods are based on fairly precise mathematical foundation.

4.9.3.5 Auto-Induced Semantic Classes

Another interesting statistical approach, although its final purpose is not an ontology derivation, was performed by Pargellis, Fosler-Lussier, Lee, Potamianos, Tsai[116]. They use an unsupervised training approach consisting of two complementary procedures. First, they use n-gram statistics to determine the similarity of words and more generally, phrases, by looking at their bigram lexical contexts within a single domain. According to their research, they compared four different similarity metrics[116, 132, 118, 117] in order to accomplish auto-inducement of semantic classes. These metrics are the Kullback-Leibler distance, the Information-Radius distance, the Manhattan-Norm distance, and the Vector-Product[118, 16, 22, 33, 105]. Phrases that are determined to be the most similar are grouped into the same semantic class (or, concept).

Next, they go a step further by ranking the degree of domain independence for various concept groups across pairs of domains in order to validate the process of using concepts across

domains. Those semantic groups that are considered domain independent will be ported across domains and used to bootstrap the understanding module for a new, untested, domain. The ability to automatically induce a concept in one domain and port it to a new domain for which little training data is available could be a powerful tool, aiding developers in building new speech services. Semantic classes, developed for well-studied domains, could be used for a new domain with little modification. By identifying task-independent versus task-dependent concepts with this methodology, a system developer can import data from other domains to fill out the set of task-independent phrases, while focusing efforts on completely specifying the task-dependent categories manually. They hypothesize that domain-independent semantic classes (concepts) should occur in similar syntactic (lexical) contexts across domains[16]. The methodology proposed ranks orders concepts by degree of domain independence which is achieved by using two metrics, the concept-comparison and concept-projection metric.

They propose an iterative procedure for automatically inducing semantic classes, consisting of three main components: a lexical phraser, a semantic generalizer, and a corpus reparser. First, the lexical phraser groups words in a single lexical unit. Next, a semantic generalizer generates rules that map words (and concepts) to concepts. Finally, a corpus parser re-parses the corpus using the rules generated from the semantic generalizer.

4.9.3.6 Related Work

There is a large body of literature in the area of ontology definition, ontology markup languages, manual and semi-automatic ontology creation. In the semantic web and web applications literature in general, there can be found a large number of papers on the computation of the semantic distance between words, phrases and concepts, e.g., [122], with application to improved web searches and web search result presentation. In [72], ontologies are constructed semi-automatically using techniques from [122] and applied to word-sense disambiguation. In [116], the authors provide techniques for finding associations between two taxonomies, i.e., mapping between semantic classes. Similar work on how to compute distance between concepts and how to merge ontologies (in the same or different languages) can be found in [32, 126]. Finally, efforts to manually or semi-automatically create ontologies to use in NLP applications can be found in [84, 21, 40].

4.9.3.7 Future Research Directions

The problem of automatic ontology constructions is a hard and very diverse one. The type of ontology that one needs to construct depends very much on the application. For example, the Sensus ontology [84] has been developed specifically for machine translation and merges information from various sources including WordNet [40], dictionaries, semantic categories and other ontologies. Creating complex ontologies from text is a very hard and futuristic goal at this point. Instead, we will focus our work in the simplest of ontologies a taxonomy, i.e., a tree ontology with only the 'is-a' relationship. Semantic taxonomies are simple yet powerful tools for many natural language processing (NLP) applications.

Traditional NLP techniques for determining semantic similarity between words and identifying 'is-a' relationships, include bag-of-word techniques and lexico-syntactic features as presented above. We will work on combining these techniques to automatically create taxonomies.

There are a lot of practical issues when building systems that automatically extract taxonomies from text. These include the design and stopping criteria for the chunk parser and the semantic generalizer, the semantic similarity measures (could be different at different levels of the taxonomy) and the evaluation of the results.

Automatic taxonomy creation using unsupervised machine learning algorithms is a challenging and important research area with many applications to web and NLP applications. Research in the area of automatic ontology creation can prove valuable especially for minority languages where few resources exists to manually create such ontologies.

Bibliography

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM-SIGMOD International Conference on the Management of Data*, pages 94–105, June 1998.

[2] Alahakoon D., Halgamuge S.K., and Srinivasan B. Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery. *IEEE Transactions on Neural Networks*, 11(3):601–614, May 2000.

[3] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 49–60. ACM Press, 1999.

[4] G.H. Ball and D.J. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, 1967.

[5] Jeffrey D. Banfield and Adrian E. Raftery. Model-based gaussian and non-gaussian clustering. *Biometrics*, 49:803–821, 1993.

[6] S. A. Barker and P. J. W. Rayner. Unsupervised image segmentation using Markov random field models. *Pattern Recognition*, 33:587–602, 200.

[7] Bauer H.U. and Pawelzik K.R. Quantifying the Neighborhood Preservation of Self-Organizing Feature Maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, July 1992.

[8] Bebis G., Georgiopoulos M., and Lobo N. Using Self-Organizing Maps to Learn Geometric Hash Functions for Model-Based Object Recognition. *IEEE Transactions on Neural Networks*, 9(3):560–570, May 1998.

[9] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2002.

[10] J. Besag. Spatial interaction and the statistical analysis of lattice systems (with discussion). *J. Roy. Statist. Soc. B*, 36:192–236, 1974.

[11] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[12] James C. Bezdek. Some new indexes of cluster validity. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 28(3):301–315, June 1998.

[13] James C. Bezdek, R. Ehrlich, and W. Full. FCM: Fuzzy c-means algorithm. *Computers and Geoscience*, 1984.

[14] James C. Bezdek and Nikhil R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics*, 28(3):301–315, 1998.

[15] Blackmore J. and Miikkulainen R. Incremental Grid Growing: Encoding High-Dimensional Structure into a Two-Dimensional Feature Map. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 450–455, 1993.

[16] Brown, P. F., et al. *Class-based n-gram models of natural language*. Computational Linguistics, vol. 18(4), pp. 467-479, 1992.

[17] Gilles Celeux and Gérard Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793, 1995.

[18] Čeljuska, D. and Vargas-Vera, M. *Semi-Automatic Population of Ontolgies from Text*. Master Thesis, Technical University Košice, 2004.

[19] Cosman P.C., Oehler K.L., Riskin E.A., and Gray R.M. Using Vector Quantization for Image Processing. *Proceedings of the IEEE*, 81(9):1326–1341, September 1993.

[20] Cottrel M., Fort J.C., and Pagès G. Two of Three Things that We Known about the Kohonen Algorithm. In *Proceedings of the 2nd European Symposium on Artificial Neural Networks*, pages 235–244, April 1994.

[21] CyC corporation. *The cyc merged ontology*. at http://www.cyc.com.

[22] Dagan, Lee and Pereira. *Similarity-Based Methods for Word-Sense Disambiguation*. Proc. of the 35th Annual Meeting of the ACL, with EACL 8, 1997.

[23] R. N. Davé and R. Krishnapuram. Robust clustering methods: A unified view. *IEEE Trans. on Fuzzy Systems*, 5(2):270–293, 1997.

[24] Rajesh N. Davé. Use of the adaptive fuzzy clustering algorithm to detect lines in digital images. In *Intelligent Robots and Computer Vision VIII*, volume 1, pages 600–611, 1989.

[25] Rajesh N. Davé and Raghu Krishnapuram. Robust clustering methods: A unified view. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, 1997.

[26] B. de Finetti. *Theory of probability*, volume 1. Wiley, New York, 1974.

[27] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. *Indexing by latent semantic analysis.*, 1990.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[29] H. Derin and H. Elliott. Modeling and segmentation of noisy and textured images using Gibb's random fields. *IEEE Trans. Patt. Anal. Machine Intell.*, PAMI-9:39–55, 1987.

[30] DeSieno D. Adding a Conscience to Competitive Learning. In *IEEE International Conference on Neural Networks*, volume 1, pages 117–124, San Diego (USA), 1988.

[31] Fisher DH. Knowledge acquisition via incremental clustering. *Machine Learning*, 2:139–172, 1987.

[32] Doan, A., Madhavan, J., Domingos, P. and Halevy, A. *Learning to map between ontologies on the semantic web*. In *In The Eleventh International WWW Conference*, Hawaii, US, 2002.

[33] Duda, R. O., Hart, P. E. and Stork, D. G. *Pattern Classification*. John Wiley & Sons, Inc.,New York, 2001.

[34] Erwin E., Obermayer K., and Schulten K. Self-Organizing Maps: Stationary States, Metastability and Convergence Rate. *Biological Cybernetics*, (67):35–45, 1992.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[36] Brian Everitt. *Cluster analysis*. Halsted Press, New York, 1974, 1993.

[37] Faure, D. and N'Edellec C. *A Corpus-based Conceptual Clustering Method for Verb Frames and Ontology Acquisition*. ASIUM System, 1998.

[38] Faure, D. and N'Edellec C. *ASIUM: Learning sub-categorization frames and restrictions of selection*. 10th Conference on Machine Learning (ECML 98), Workshop on Text Mining, Chemnitz, Germany, 1998.

[39] Fausset L. *Fundamentals of Neural Networks - Architecures, Algorithms and Applications*. Prentice Hall, 1994.

[40] Fellbaum, C. (editor). *WordNet An Electronic Lexical Database*. The MIT Press, 1998.

[41] FOLDOC. *The Free Online Dictionary of Computing*. at http://foldoc.doc.ic.ac.uk/foldoc/index.html.

[42] Greet Frederix and Eric Pauwels. Two general geometric cluster validity indices. In *Proceedings of the 4th Industrial Conference on Data Mining (ICDM'2004)*, July 2004.

[43] Hichem Frigui and Raghu Krishnapuram. A robust clustering algorithm based on competitive agglomeration and soft rejection of outliers. In *CVPR'96*, San Francisco, June 1996.

[44] Hichem Frigui and Raghu Krishnapuram. Clustering by competitive agglomeration. *Pattern Recognition*, 30(7):1109–1119, 1997.

[45] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, 1999.

[46] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Trans. on PAMI*, 21(5):450–465, May 1999.

[47] Fritzke B. Growing Grid - A Self-Organizing Network with Constant Neighborhood and Adaptation Strength. *Neural Processing Letters*, 2(5):9–13, 1995.

[48] I. Gath and A. B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on pattern Analysis and Machine Intelligence*, 11(7):773–781, 1989.

[49] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. Chapman and Hall, London, 1995.

[50] Gersho A. and Gray R.M. *Vector Quantization and Data Compression*. Kluwer Academic, 1997.

[51] J. Grabmeier and A. Rudolph. Techniques of cluster algorithms in data mining. *Data Mining and Knowledge Discovery*, 6(4):303–306, October 2002.

[52] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82:711–732, 1995.

[53] Greengrass, E. *Information retrieval: an overview*, 1997.

[54] Gruber, T. R. *A Translation Approach to Portable Ontology Specification*. Knowledge acquisition 5(2), pages 199-220, 1993.

[55] Gruber,T. R. *Toward Principles for the design of Ontology used for Knowledge Sharing*. In Proc. of International Workshop on Formal Ontology, 1993.

[56] Guarino, N. *Formal Ontology and Information Systems*. Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98, Trento, Italy, 1998.

[57] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 73–84. ACM Press, 1998.

[58] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, December 2001.

[59] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[60] Harman, D. *Overview of the Fourth Text REtrieval Conference (TREC-4)*. National Institute of Standards and Technology, 1995.

[61] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, 1998.

[62] Thomas Hofmann and Joachim M. Buhmann. Competitive learning algorithms for robust vector quantization. *IEEE Trans. on Signal Processing*, 46(6):1665–1675, June 1998.

[63] Zhexue Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (SIGMOD-DMKD'97)*, May 1997.

[64] M. Imberty and X. Descombes. Simulation de processus objets: etude de faisabilité pour une application à la segmentation d'image. Technical Report 3881, INRIA, Sophia Antipolis, 2000.

[65] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[66] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surv.*, 31(3):264–323, Sept. 1999.

[67] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[68] Anil K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[69] Jain A.K. and Dubes R.C. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[70] Jain A.K, Murty M.N., and Flynn P.J. Data Clustering: A Review. *ACM Computing Surveys*, 31(3):265–323, September 1999.

[71] Joachims, T. *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*. Logic J. of the IGPL, 1998.

[72] Kang, S.J., Li, H.F. and Lee,J.H. *Semi-automatic construction of language independent and practical ontology for word senses disambiguation*. Proceedings of the 8th Machine Translation Summit, Santiago de Compostela, Galicia, Spain, 2001.

[73] Kangas J.A., Kohonen T., and Laaksonen J.T. Variants of Self-Organizing Maps. *IEEE Transactions on Neural Networks*, 1(1):93–99, March 1990.

[74] Ismo Kärkkäinen and Pasi Fränti. Dynamic local search for clustering with unknown number of clusters. In *ICPR'02*, Quebec, August 2002.

[75] Kaski S., Kangas J., and Kohonen T. Bibliography of Self-Organizing Map (SOM) Papers: 1981-1997. *Neural Computing Surveys*, 1:102–350, 1998.

[76] Z. Kato. Bayesian color image segmentation using reversible jump Markov chain Monte Carlo. Technical Report 01/99-R055, European Research Consortium for Informatics and Mathematics, 1999.

[77] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990.

[78] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley, 1990.

[79] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley and Sons, New York, 1990.

[80] Khan, L., Luo, F. and Yen, I. *Automatic Ontology Derivation from Documents*. University of Texas at Dallas, 2002.

[81] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 69:86–101, 1967.

[82] Kirk J.S. and Zurada J.M. Algorithms for Improved Topology Preservation in Self-Organizing Maps. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 396–400, October 1999.

[83] Kiviluoto K. Topology Preservation in Self-Organizing Maps. In *IEEE International Conference on Neural Network*, volume 1, pages 294–299, June 1996.

[84] Knight, K. and Luk, S. *Building a largescale knowledge base for machine translation*. In em Proc. of the National Conference on Artificial Intelligence (AAAI), pages 773–778, Seattle, WA, 1994.

[85] Kohonen T. Learning Vector Quantization. *Neural Networks*, 1(1):303, 1988.

[86] Kohonen T. The Self-Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[87] Kohonen T. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, New York, third edition, 1997.

[88] Kohonen T., Hynninen J., Kangas J., and Laaksonen J. SOM PAK: The Self-Organizing Map Program Package. Technical report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[89] Kohonen T., Hynninen J., Kangas J., Laaksonen J., and Torkkola K. LVQ_PACK: The Learning Vector Quantization Package. Technical report, Helsinki University of Technology, April 1995.

[90] Koikkalainen P. and Oja E. Self-Organizing Hierarchical Feature Maps. In *Proceedings of the International Join Conference on Neural Networks*, volume 2, pages 279–285, 1990.

[91] Laaksonen J., Koskela M., and Oja E. PicSOM - Self-Organizing Image Retrieval With MPEG-7 Content Descriptors. *IEEE Transactions on Neural Networks*, 13(4):841–853, July 2002.

[92] S. Lakshmanan and H. Derin. Valid parameter space for 2-d Gaussian Markov random fields. *IEEE Trans. Inform. Theory*, 39:703–709, 1993.

[93] T. Lange, V. Roth, M. Braun, and J. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, June 2004.

[94] Bertrand Le Saux and Nozha Boujemaa. Unsupervised robust clustering for image database categorization. In *Proceedings of the IEEE-IAPR International Conference on Pattern Recognition (ICPR'2002)*, August 2002.

[95] Bertrand Le Saux and Nozha Boujemaa. Unsupervised robust clustering for image database categorization. In *ICPR'02*, Quebec, August 2002.

[96] P. M. Lee. *Bayesian statistics: an introduction*. Hodder Arnold H& S, London, third edition, 2004.

[97] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Comm.*, 28:84–95, 1980.

[98] Lebowitz M. Categorizing numeric information for generalization. *Cognitive Science*, 9:285–309, 1985.

[99] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on Math. Statistics and Probability*, 1967.

[100] Maddi, G. R. and Velvadapu, C. S. *Ontology Extraction from text documents by Singular Value Decomposition*. Bowie State University, 2001.

[101] Maedche, A. and Staab, S. *Discovering Conceptual Relations from Text*. Technical Report 399, Institute AIFB, Karlsruhe University, 2000.

[102] Maedche, A. and Staab, S. *The Text-to-Onto Ontology Learning Environment*. Institute AIFB, Karlsruhe University, 2000.

[103] Maedche, A. and Staab, S. *Ontology learning for the Semantic Web*. IEEE Intelligent Systems, 16(2), 2001.

[104] Manjunath B.S., Ohm J.R., Vasudevan V.V., and Yamada A. Color and Texture Descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):703–715, June 2001.

[105] Manning, C. D. and Schutze, H. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, 2000.

[106] McGuinness, D. L. *Ontologies Come of Age. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.

[107] J. McQueen. Some methods for classifcation and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathmatical Statistics and Probabilty*, pages 281–297, 1967.

[108] D. E. Melas and S. P. Wilson. Double Markov random fields and Bayesian image segmentation. *IEEE Trans. Sig. Proc.*, 50:357–365, 2002.

[109] Miller, G. *WordNet: A Lexical Database for English*. In Proc. of Communications of CACM, 1995.

[110] Murashima S., Kashima M., and Fuchida T. New Method for Measuring the Topology Preservation of Self-Organizing Feature Maps. In *Proceedings of the* 6^{th} *International Conference on Neural Information Processing*, volume 1, pages 273–278, November 1999.

[111] Fionn Murtagh. Structure of hierarchic clusterings: implications for information retrieval and for multivariate data analysis. *Inf. Process. Manage.*, 20(5-6):611–617, 1984.

[112] Neumann, G., Backofen, R., Baur, J., Becker, J. and Braun, C. *An information extraction core system for real world german text processing*. In ANLP'97 Proceedings of the Conference on Applied Natural Language Processing, pages 208-215, Washington, USA, 1997, 1997.

[113] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of 14th Advances in Neural Information Processing Systems*, 2002.

[114] Oja M., Kaski S., and Kohonen T. Bibliography of Self-Organizing Map (SOM) Papers: 1998-2001. *Neural Computing Surveys*, 3:1–156, 2003.

[115] Perner P. Different learning strategies in a case-based reasoning system for image interpretation. In B. Smyth and P. Cunningham, editors, *Advances in Case-Based Reasoning*, LNAI 1488, pages 251–261. Springer, Heidelberg, 1998.

[116] Pargellis, A., Fosler-Lussier, E., Lee, C, Potamianos, A. and Tsai, A. *Auto-Induced Semantic Classes*. Speech Communication, to appear, 2004, 2004.

[117] Pargellis, A. N., Fosler-Lussier, E. *A Comparison of Four Metrics for Auto-inducing Semantic Classes*. Proc. of the Automatic Speech Recognition and Understanding Workshop, Madonna di Campiglio, 2001.

[118] Pargellis, A. N., Fosler-Lussier, E., Potamianos, A. and Lee, C.-H. *Metrics for Measuring Domain Independence of Semantic Classes*. Proceedings of the 7th European Conference on Speech Communication and Technology, Aalborg, Denmark, 2001.

[119] P. Perner and S. Jänichen. Learning of form models from exemplars. In *SSPR 2004*, page to be published. Springer, Heidelberg, 2004.

[120] Rauber A., Merkl D., and Dittenbach M. The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, November 2002.

[121] C. S. Regazzoni and A. N. Venetsanopoulos. Group-membership reinforcement for straight edges based on Bayesian networks. *IEEE Trans. Image Proc.*, 7:1–19, 1998.

[122] Resnik, P. *Using taxonomy information content to evaluate semantic similarity in a taxonomy*. In Proc. of IJCAI-95, Montreal, Canada, 1995.

[123] Ritter H., Martinetz T., and Schulten K. *Neural Computation and Self-Organizing Maps - An Introduction*. Addison-Wesley, New York, 1992.

[124] Ritter H. and Schulten K. Convergence Properties of Kohonen's Topology Conserving Maps: Fluctuations, Stability, and Dimension Selection. *Biological Cybernetics*, (60):59–71, 1988.

[125] S. J. Roberts. Parametric and non-parametric unsupervised cluster analysis. *Pattern Recognition*, 30(2):261–272, February 1997.

[126] M. Rodriguez and M. Egenhofer. *Determining semantic similarity among entity classes from different ontologies*. IEEE Trans. on Knowledge and Data Eng., 15(2):442–456, 2003.

[127] Michalski RS. A theory and methodology of inductive learning. In R. S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: Artificial Intelligence Approach*. Morgan Kaufmann, 1983.

[128] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.

[129] V. Samson and P. Bouthemy. Learning classes for video interpretation with a robust parallel clustering method. In *ICPR'04*, Cambridge, UK, August 2004.

[130] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Min. Knowl. Discov.*, 2(2):169–194, 1998.

[131] Sato A. and Yamada K. Generalized Learning Vector Quantization. In *Advances in Neural Information Processing Systems*, volume 7, pages 423–429, 1995.

[132] Siu, K.-C., Meng, H. M. *Semi-automatic Acquisition of Domain-Specific Semantic Structures*. Proc. of the Sixth European Conf. on Speech Comm. and Tech., Budapest, vol. 5, pp. 2039-2042, 1999.

[133] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman, 1973.

[134] Srivastava,S., Lamadrid,J. G. and Karakashyan, Y. *CADIP 2000: Document Ontology Extractor*. Bowie State University, 2000.

[135] Strikant, R. and Agrawal, R. *Mining Generalized Association Rules*. In Proc. of VLDB'95, pages 407-419, 1995.

[136] M. A. Tanner. *Tools for statistical inference: methods for the exploration of posterior distributions and likelihood functions*. Springer-Verlag, New York, Second edition, 1993.

[137] Theodoridis S. and Koutroumbas K. *Pattern Recognition*. Elsevier Academic Press, second edition edition, 2003.

[138] Villmann T., Der R., Herrmann M., and Martinetz T.M. Topology Preservation in Self-Organizing Feature Maps: Exact Definition and Measurement. *IEEE Transactions on Neural Networks*, 8(2):256–266, March 1997.

[139] J. H. Ward. Hierachical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

[140] S. P. Wilson and J. Zerubia. Segmentation of textured satellite and aerial images by Bayesian inference and Markov random fields. Technical Report 4336, INRIA Sophia-Antipolis, 2001.

[141] G. Winkler. *Image analysis, random fields and dynamic Monte Carlo methods*. Springer-Verlag, Berlin, First edition, 1995.

[142] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):841–847, 1991.

[143] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computing*, C-20:68–86, 1971.

Chapter 5

Semi-supervised Learning

5.1 Semi-supervised Clustering

In addition to the similarity information used by unsupervised clustering, in many cases a small amount of knowledge is available concerning either pairwise (must-link or cannot-link) constraints between data items or class labels for some items. Instead of simply using this knowledge for the external validation of the results of clustering, one can imagine letting it "guide" or "adjust" the clustering process, i.e. provide a limited form of supervision. The resulting approach is called *semi-supervised clustering*. We also consider that the available knowledge is too far from being representative of a target classification of the items, so that supervised learning is not possible, even in a transductive form.

Note that class labels can always be translated into pairwise constraints for the labeled data items and, reciprocally, by using consistent pairwise constraints for some items one can obtain groups of items that should belong to a same cluster.

5.1.1 A Typology of Methods

Two sources of information are usually available to a semi-supervised clustering method: the similarity measure unsupervised clustering would employ and some pairwise constraints (must-link or cannot-link). For semi-supervised clustering to be profitable, these two sources of information should not completely contradict each other.

Unlike traditional clustering, the semi-supervised approach to clustering has a short history and few methods were published until now. The main distinction between these methods concerns the way the two sources of information are combined: either by adapting the similarity measure or by modifying the search for appropriate clusters.

- In *similarity-adapting* methods, an existing clustering algorithm using some similarity measure is employed, but the similarity measure is adapted so that the available constraints can be easier satisfied. Several similarity measures were employed for similarity-adapting semi-supervised clustering: the Jensen-Shannon divergence trained with gradient descent [3], the Euclidean distance modified by a shortest-path algorithm [7] or Mahalanobis distances adjusted by convex optimization [11], [2]. Among the clustering

algorithms using such adapted similarity measures we can mention hierarchical single-link [2] or complete-link [7] clustering and k-means [11], [2].

- In *search-based* methods, the clustering algorithm itself is modified so that user-provided constraints or labels can be used to bias the search for an appropriate clustering. This can be done in several ways, such as by performing a transitive closure of the constraints and using them to initialize clusters [1], by including in the cost function a penalty for lack of compliance with the specified constraints [4], or by requiring constraints to be satisfied during cluster assignment in the clustering process [10].

5.1.2 Introduction

Gaussian mixture model approach is a flexible statistical approach to model complex and inhomogeneous categories [6]. Let us consider that observations of class c have the following density:

$$f_c(\mathbf{x}) = P(X = \mathbf{x}|Y = c) = \sum_{r=1}^{R_c} \pi_{cr} G_{cr}(X; \mu_{cr}, \Sigma_{cr})$$

where the mixing proportions π_{cr} sum to one, G_{cr} is a Gaussian distribution, and μ_{cr} and Σ_{cr} are the class center and the covariance matrix respectively.

We are now give an example of these general Gaussian mixture models for CBIR application. Image indexes are supposed to be generated from a Gaussian mixture with R components. Interactive image retrieval is characterized by a small number of labeled observations (images). However, the mixture model approach does not seem adapted because it requires many parameter estimation. Using few observations to estimate many parameters leads at least to non robust estimation. Considering diagonal covariance matrix allows to reduce the number of parameters to be estimated and increases the estimation robustness. To handle with multi-modal categories of relevant images (class C_1) and a heterogeneous class C_2 of irrelevant images, both classes are represented by a mixture of R_1 and R_2 Gaussian components respectively. To simplify the problem formulation, we note $g_r()$ the Gaussian density of the mixture component r. Thus the probability density of vector $\mathbf{x}_i$ is:

$$g(\mathbf{x}_i|\Phi) = \sum_{r=1}^{R_1} \pi_r g_r(\mathbf{x}_i|\theta_r) + \sum_{r=R_1+1}^{R_1+R_2} \pi_r g_r(\mathbf{x}_i|\theta_r) \tag{5.1}$$

where π_r is the proportion of component c_r, ($0 < \pi_r < 1$ and $\sum_{r=1}^{R} \pi_r = 1$). $\Phi = (\pi_1, ..., \pi_R, \theta_1, ..., \theta_R)$ is the parameter vector to be estimated.

The mixture parameters can be estimated by maximizing the likelihood knowing the indexes and their labels. The classical and natural method for computing the maximum-likelihood estimates for mixture distributions is the EM algorithm [5].

5.1.3 Semi-supervised mixture model

In addition to mixture modeling, using all the data of the database (for instance, labeled and unlabeled images in CBIR) as training set is also helpful to obtain robust estimation [8]. This approach lies within the semi-supervised learning framework [9].

The adaptation of the previous approach is simple and is based on the introduction of annotation vectors $\mathbf{z}_i \in \{0,1\}^{R_1+R_2}$ which are coded as follows:

- $\mathbf{z}_i = (\underbrace{1,\ldots,1}_{R_1 \text{ times}},\underbrace{0,\ldots,0}_{R_2 \text{ times}})$, for $\mathbf{x}_i$ *labeled* relevant,
- $\mathbf{z}_i = (\underbrace{0,\ldots,0}_{R_1 \text{ times}},\underbrace{1,\ldots,1}_{R_2 \text{ times}})$, for $\mathbf{x}_i$ *labeled* irrelevant,
- $\mathbf{z}_i = (\underbrace{1,\ldots,1}_{R_1+R_2 \text{ times}})$ if $\mathbf{x}_i$ is *unlabeled*.

The posterior probability that $\mathbf{x}_i$ belongs to class C_1 is given by:

$$P(\mathbf{x}_i \in C_1|\mathbf{x}_i,\mathbf{z}_i;\Phi) \quad = \quad \sum_{r=1}^{R_1} p(r|\mathbf{x}_i,\mathbf{z}_i;\Phi) = \sum_{r=1}^{R_1} \frac{z_{ir}\pi_r g_r(\mathbf{x}_i|\theta_r)}{\sum_{s=1}^{R} z_{is}\pi_s g_s(\mathbf{x}_i|\theta_s)} \tag{5.2}$$

The mixture parameter estimation, at each iteration q, may be expressed in that case as follows:

- E-step: for each component r and each image $\mathbf{x}_i$, compute

$$c_{ir}^{(q)} = p(r|\mathbf{x}_i,\mathbf{z}_i;\Phi^{(q)}) = \frac{z_{ir}\pi_r^{(q)} g_r(\mathbf{x}_i|\theta_r^{(q)})}{\sum_{l=1}^{R} z_{il}\pi_l^{(q)} g_l(\mathbf{x}_i|\theta_l^{(q)})}$$

- M-step: Compute the parameters $\Phi^{(q+1)}$, which maximize

$$Q(\Phi|\Phi^{(q)}) = \sum_{i=1}^{N}\sum_{r=1}^{R} c_{ir}^{(q)} \log \pi_r g_r(\mathbf{x}_i|\theta_r)$$

Finally, the images are returned to the user by descending order of:

$$f(\mathbf{x}_i) = \frac{P(\mathbf{x}_i \in C_1|\mathbf{x}_i,\mathbf{z}_i;\Phi)}{P(\mathbf{x}_i \in C_2|\mathbf{x}_i,\mathbf{z}_i;\Phi)}$$

This kind of strategy has been successfully applied in CBIR for medical imagery applications [8], and may be compared to transductive SVM strategies.

Bibliography

[1] S. Basu, A. Banerjee, and R. J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of 19th International Conference on Machine Learning (ICML-2002)*, pages 19–26, 2002.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *International Conference on Knowledge Discovery and Data Mining*, pages 39—-48, Washington, DC, 2003.

[3] David Cohn, Rich Caruana, and Andrew McCallum. Semi-supervised clustering with user feedback, 2000.

[4] A. Demiriz, K. Bennett, and M. Embrechts. Semi-supervised clustering using genetic algorithms. In C. H. Dagli et al., editor, *Intelligent Engineering Systems Through Artificial Neural Networks 9*, pages 809–814. ASME Press, 1999.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likehood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[6] T. Hastie and R. Tibshirani. Discriminant analysis by gaussian mixtures. *Journal of the Royal Statistical Society B*, 58:155–176, 1996.

[7] Dan Klein, Sepandar D. Kamvar, and Christopher D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning*, pages 307–314. Morgan Kaufmann Publishers Inc., 2002.

[8] N. Najjar, J.P. Cocquerez, and C. Ambroise. Feature selection for semi supervised learning applied to image retrieval. In *IEEE ICIP*, Barcelena, Spain, Sept. 2003.

[9] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Classification from labeled and unlabeled documents using em. *Machine learning*, 39(2/3):135–167, 2000.

[10] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1103–1110, 2000.

[11] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512, Cambridge, MA, 2003. MIT Press.

Chapter 6

Active learning

6.1 Introduction

Learning is a process where an agent uses a set of experiences to improve its problem solving capabilities. In the *passive learning* model, the learner passively receives a srteam of experiences and processes them. In the *active learning* model the learner has some control over its training experiences. Markovitch and Scott [20] define the information filtering framework which specifies five types of selection mechanisms that a learning system can employ to increase the utility of the learning process. Active learning can be viewed in this model as *selective experience* filter.

Why should a learner be active? There are two major reasons for employing experience selection mechanisms:

1. If some experiences have negative utility – the quality of learning would have been better without them – then it is obviously desirable to filter them out. For example, noisy examples are usually harmful.

2. Even when all the experiences are of positive utility, there is a need to be selective or to have control over the order in which they are processed. This is because the learning agent has bounded training resources which should be managed carefully.

In *passive* learning systems, these problems are either ignored, leading to inefficient learning, or are solved by relying on a human teacher to select informative training experiences [36]. It may sometimes be advantageous for a system to select its own training experiences even when an external teacher is available. Scott & Markovitch [27] argue that a learning system has an advantage over an external teacher in selecting informative examples because, unlike a teacher, it can directly access its own knowledge base. This is important because the utility of a training experience depends upon the current state of the knowledge base.

In this summary we concentrate on active learning for supervised concept induction. Most of the works in this area assume that there are high costs associated with tagging examples. Consider, for example, training a classifier for a character recognition problem. In this case, the character images are easily available, while their classification is a costly process requiring a human operator.

In the following subsections we differentiate between pool-based selective sampling and example construction approaches, define a formal framework for selective sampling algorithms and present various approaches for solving the problem.

6.2 The Source of the Examples

One characteristic in which active learning algorithms vary is the source of input examples. Some algorithms *construct* input examples, while others *select* from a pool or a stream of unlabeled examples.

6.2.1 Example Construction

Active learning by example construction was formalized and theoretically studied by Angluin [2]. Angluin developed a model which allows the learner to ask two types of queries: a *membership* query, where the learner selects an instance x and is told its membership in the target concept, $f(x)$; and an *equivalence* query, where the learner presents a hypothesis h, and either told that $h \equiv f$, or is given a counterexample. Many polynomial time algorithms based on Anguin's model have been presented for learning target classes such as Horn sentences [3] and multivariate polynomials [7]. In each case, a domain-specific query construction algorithm was presented.

In practical experiments with example construction, there are algorithms that try to optimize an objective function like information gain or generalization error, and derive expressions for optimal queries in certain domains [30, 31]. Other algorithms find classification borders and regions in the input space and construct queries near the borders [6, 13].

A major drawback of query construction was shown by Baum and Ladner [15]. When trying to apply a query construction method to the domain of images of handwritten characters, many of the images constructed by the algorithm did not contain any recognizable characters. This is a problem in many real-world domains.

6.2.2 Selective Sampling

Selective sampling avoids the problem described above by assuming that a pool of unlabeled examples is available. Most algorithms work iteratively by evaluating the unlabeled examples and select one for labeling. Alternatively it is assumed that the algorithms receives a stream of unlabeled examples, and for each example a decision is made (according to some evaluation criteria) whether to query for a label or not.

The first approach can be formalized as follows. Let $\mathcal{X}$ be an *instance space*, a set of objects described by a finite collection of attributes (*features*). Let $f : \mathcal{X} \to \{0,1\}$ be a *teacher* (also called an expert) that labels instances as 0 or 1. A *(supervised) learning algorithm* takes a set of *labeled examples*, $\{\langle x_1, f(x_1)\rangle, \ldots, \langle x_n, f(x_n)\rangle\}$, and returns a *hypothesis* $h : \mathcal{X} \to \{0,1\}$. Let $X \subseteq \mathcal{X}$ be an *unlabeled training set*. Let $D = \{\langle x_i, f(x_i)\rangle : x_i \in X, i = 1, \ldots, n\}$ be the *training data* – a set of labeled examples from X. A selective sampling algorithm S_L, specified relative to a learning algorithm L, receives X and D as input and returns an unlabeled element of X.

Active Learner(X,f()):

1. $D \leftarrow \emptyset$.
2. $h \leftarrow L(\emptyset)$.
3. While stopping-criterion is not satisfied do:

a)	$x \leftarrow S_L(X, D)$	; Apply S_L and get the next example.
b)	$\omega \leftarrow f(x)$	; Ask the teacher to label x.
c)	$D \leftarrow D \bigcup \{\langle x, \omega \rangle\}$	; Update the labeled examples set.
d)	$h \leftarrow L(D)$	; Update the classifier.

4. Return classifier h.

Figure 6.1: Active learning with selective sampling. Active Learner is defined by specifying stopping criterion, learning algorithm L and selective sampling algorithm S_L. It then works with unlabeled data X and teacher $f()$ as input.

The process of learning with selective sampling can be described as an iterative procedure where at each iteration the selective sampling procedure is called to obtain an unlabeled example and the teacher is called to label that example. The labeled example is added to the set of currently available labeled examples and the updated set is given to the learning procedure, which induces a new classifier. This sequence repeats until some stopping criterion is satisfied. This criterion may be a resource bound, M, on the number of examples that the teacher is willing to label, or a lower bound on the desired class accuracy. Adopting the first stopping criterion, the goal of the selective sampling algorithm is to produce a sequence of length M which leads to a best classifier according to some given measure.

The pseudo code for an active learning system that uses selective sampling is shown in Figure 6.1.

6.3 Example Evaluation and Selection

Active learning algorithms that perform selective sampling differ from each other by the way they select the next example for labeling. In this section we present some common approaches to this problem.

6.3.1 Uncertainty-based Sampling

Most of the works in active learning select untagged examples that the learner is most uncertain about. Lewis and Gale [17] present a probabilistic classifier for text classification. They assume that a large pool of unclassified text documents is available. The probabilistic classifier assigns tag probabilities to text documents based on already labeled documents. At each iteration, several unlabeled examples that were assigned probabilities closest to 0.5 are selected for labeling. It is shown that the method significantly reduces that amount of labeled data needed to achieve

a certain level of accuracy, compared to random sampling. Later, Lewis and Catlett [16] used an efficient probabilistic classifier to select examples for training another (C4.5) classifier. Again, examples with probabilities closest to 0.5 were chosen.

Special uncertainty sampling methods were developed for specific classifiers. Park [24] uses sampling-at-the-boundary method to choose support vectors for SVM classifier. Orthogonal support vectors are chosen from the boundary hyperplane and queried one at a time. Tong and Koller [34] also perform pool-based active learning of SVMs. Each new query is supposed to split the current version space into two parts as equal as possible. The authors suggest three methods of choosing the next example based on this principle. Hasenjager & Ritter[13] and Lindenbaum et. al [19] present selective sampling algorithms for nearest neighbor classifiers.

6.3.2 Committee-based sampling

Query By Committee (QBC) is a general uncertainty-based method where a committee of hypotheses consistent with the labeled data is specified and an unlabeled example on which the committee members most disagree is chosen. Seung, Oper and Sompolinsky [28] use Gibbs algorithm to choose random hypotheses from the version space for the committee, and choose an example on which the disagreement between the committee members is the biggest. Application of the method is shown on the high-low game and perceptron learning. Later, Freund et. al. [10] presented a more complete and general analysis of query by committee, and show that the method guarantees rapid decrease in prediction error for the perceptron algorithm.

Cohn et. al. [8] present the concept of uncertainty regions - the regions on which hypotheses that are consistent with the labeled data might disagree. Although it is difficult to exactly calculate there regions, one can use approximations (larger or smaller regions). The authors show an application of the method to neural networks - two networks are trained on the labeled example set, one is the most general network, and the other is the most specific network. When examining an unlabeled example, if the two networks disagree on the example, the true label is queried.

Krogh and Vedelsby [14] show the connection between the generalization error of a committee of neural networks and its ambiguity. The larger is the ambiguity, the smaller is the error. The authors build an ambiguous network ensemble by cross validation and calculate optimal weights for ensemble members. The ensemble can be used for active learning purposes - the example chosen for labeling is the one that leads to the largest ambiguity. Raychandhuri et. al. [26] use the same principle and analyze the results with an emphasize on minimizing data gathering. Hasenjager and Ritter [12] compare the theoretical optimum (maximization of information gain) of the high-low game with the performance of QBC approach. It is shown that QBC results rapidly approach the optimal theoretical results.

In some cases, especially in the presence of noise, it is impossible to find hypotheses consistent with the labeled data. Abe and Mamitsuka [1] suggest to use QBC with combination of boosting or bagging. Liere and Tadepalli [18] use a small committee of 7 Winnow classifiers for text categorization. Engelson and Dagan [4] overcome the problem of finding consistent hypotheses by generating a set of classifiers according to the posterior distribution of the parameters. The authors show how to do that for binomial and multinomial parameters. This method is only applicable when it is possible to estimate a posterior distribution over the model

space given the training data. Muslea et. al. [22] present an active learning scheme for problems with several redundant views – each committee member is based on one of the problem views.

Park et. al. [25] combine active learning with using unlabeled examples for induction. They train a committee on an initial training set. At each iteration, the algorithm predicts the label of an unlabeled example. If prediction disagreement among committee members is below a threshold, the label is considered to be the true label and committee members weights are updated according to their prediction. If disagreement is high, the algorithm queries the oracle (expert) for the true label. Baram et. al. [5] use a committee of known-to-be-good active learning algorithms. The committee is then used to select the next query. At each stage, the best performing algorithm in the committee is traced, and committee members weights are updated accordingly.

Melville and Mooney [21] use artificially created examples to construct highly diverse committees. Artificial examples are created based on the training set feature distribution, and labeled with the least probable label. A new classifier is built using the new training set, and if it does not increase the committee error, it is added to the committee.

6.3.3 Clustering

Recently, several works suggest performing pre-clustering of the instance pool before selecting an example for labeling. The motivation is that grouping similar examples may help to decide which example will be useful when labeled. Engelbrecht et. al. [9] cluster the unlabeled data and then perform selective sampling in each of the clusters. Nguyen et. al. [23] pre-cluster the unlabeled data and give priority to examples that are close to classification boundary and are representatives of dense clusters.

Soderland [29] incorporates active learning in an information extraction rules learning system. The system maintains a group of learned rules. In order to choose the next example for label, all unlabeled examples are divided to three categories - examples covered by rules, near misses, and examples not covered by any rule. Examples for labele are sampled randomly from these groups (proportions are selected by the user).

6.3.4 Utility-Based Sampling

Fujii et. al. [11] have observed that considering only example uncertainty is not sufficient when selecting the next example for label, and that the impact of example labeling on other ulabeled examples should also be taken into account. Therefore, example utility is measured by the sum of interpretation certainty differences for all unlabeled examples, averaged with the examined example interpretation probabilities. The method was tested on the word sense disambiguation domain, and showed to perform better than other sampling methods.

Lindenbaum et. al. [19] present a very general utility-based sampling method. The sampling process is viewed as a game where the learner actions are the possible instances and the teacher reactions are the different labels. The sampling algorithm expands a “game” tree of depth k, where k depends on the resources available. This tree contains all the possible sampling sequences of length k. The utility of each leaf is computed by evaluating the classifier resulting from performing induction on the sequence leading to that leaf. The arcs associated with the

teacher's reactions are assigned probabilities using a random field model. The example selected is the one with the highest expected utility.

6.4 Conclusion

Active learning is a powerful tool. It has been shown to be efficient in reducing the number of labeled examples needed to perform learning, thus reducing the cost of creating training sets. It has been successfully applied to many problem domains, and is potentially applicable to any task involving learning. Active learning was applied to numerous domains such as function approximation [14], speech recognition [35], recommender system [32], text classification and categorization [17, 34, 18, 33, 37], part of speech tagging [4], image classification [22] and word sense disambiguation [11, 25].

Bibliography

[1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 1–9. Morgan Kaufmann Publishers Inc., 1998.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[3] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9(2-3):147–164, July 1992.

[4] S. Argamon-Engelson and I. Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, November 1999.

[5] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 19–26. AAAI Press, 2003.

[6] E.B. Baum. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2(1):5–19, January 1991.

[7] Nader H. Bshouty. On learning multivariate polynomials under the uniform distribution. *Information Processing Letters*, 61(6):303–309, March 1997.

[8] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, May 1994.

[9] A.P. Engelbrecht and R. Brits. Supervised training using an unsupervised approach to active learning. *Neural Processing Letters*, 15(3):247–260, June 2002.

[10] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, August-September 1997.

[11] A. Fujii, T. Tokunaga, K. Inui, and H. Tanaka. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597, December 1998.

[12] M. Hasenjager and H. Ritter. Active learning of the generalized high-low game. In *Proceedings of the 1996 International Conference on Artificial Neural Networks*, pages 501–506. Springer-Verlag, 1996.

[13] M. Hasenjager and H. Ritter. Active learning with local models. *Neural Processing Letters*, 7(2):107–117, April 1998.

[14] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, 7:231–238, 1995.

[15] K.J. Lang and E.B. Baum. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, Septebmer 1992.

[16] D.D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 148–56. Morgan Kaufmann, July 1994.

[17] D.D. Lewis and W.A. Gale. A sequential algorithm for training text classifiers. In W.B. Croft and C.J. Van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. ACM/Springer, 1994.

[18] R. Liere and P. Tadepalli. Active learning with committees for text categorization. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 591–596. AAAI Press, July 1997.

[19] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2):125–152, February 2004.

[20] Shaul Markovitch and Paul D. Scott. Information filtering: Selection mechanisms in learning systems. *Machine Learning*, 10(2):113–151, 1993.

[21] P. Melville and R. Mooney. Diverse ensembles for active learning. In *In Proceedings of the 21th International Conference on Machine Learning*, pages 584–591. IEEE, July 2003.

[22] I. Muslea, S. Minton, and C.A. Knoblock. Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 621–626. AAAI Press / The MIT Press, 2000.

[23] H.T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *In Proceedings of the 21th International Conference on Machine Learning*. IEEE, July 2003.

[24] J.M Park. On-line learning by active sampling using orthogonal decision support vectors. In *Neural Networks for Signal Processing - Proc. IEEE Workshop*. IEEE, December 2000.

[25] S.B. Park, B.T. Zhang, and Y.T. Kim. Word sense disambiguation by learning decision trees from unlabeled data. *Applied Intelligence*, 19(1-2):27–38, July-October 2003.

[26] T. Raychaudhuri and L.G.C. Hamey. Minimisation of data collection by active learning. In *IEEE International Conference on Neural Networks Proceedings*, pages 1338–1341. IEEE, 1995.

[27] Paul D. Scott and Shaul Markovitch. Experience selection and problem choice in an exploratory learning system. *Machine Learning*, 12:49–67, 1993.

[28] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294. ACM Press, 1992.

[29] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, February 1999.

[30] P. Sollich. Query construction, entropy, and generalization in neural network models. *Physical Review E*, 49(5):4637–4651, May 1994.

[31] Kah-Kay Sung and P. Niyogi. Active learning the weights of a rbf network. In *Neural Networks for Signal Processing*, Septebmer 1995.

[32] I. R. Teixeira, Francisco de A. T. de Carvalho, G. Ramalho, and V. Corruble. Activecp: A method for speeding up user preferences acquisition in collaborative filtering systems. In G. Bittencourt and G. Ramalho, editors, *16th Brazilian Symposium on Artificial Intelligence, SBIA 2002*, pages 237–247. Springer, November 2002.

[33] C.A. Thompson, M.E. Califf, and R.J. Mooney. Active learning for natural language parsing and information extraction. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 406–414. Morgan Kaufmann Publishers Inc., June 1999.

[34] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, November 2001.

[35] D. Tur, G. Riccardi, and A. L. Gorin. Active learning for automatic speech recognition. In *In Proceedings of IEEE ICASSP 2002*. IEEE, May 2002.

[36] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[37] C. Zhang and T. Chen. Annotating retrieval database with active learning. In *Proceedings. 2003 International Conference on Image Processing*, pages II: 595–598. IEEE, September 2003.

Chapter 7

Dimension Reduction

7.1 Introduction

There are two commonly used techniques in machine learning for reducing the dimensionality of a dataset:

- *Feature extraction:* Feature extraction (or Feature Transformation)is a pre-processing technique that transforms the original features of a data set to a smaller, more compact feature set, while retaining as much information as possible. The most popular approach to feature extraction involves the application of *Principal Component Analysis* (PCA), a multivariate analysis procedure that projects data onto a reduced set of dimensions by producing a new set of orthogonal axes, where each is a linear combination of the original features. A variation of this technique, *Latent Semantic Indexing* (LSI), has been commonly employed in information retrieval scenarios.

  While feature extraction methods can often be effective in reducing dimensionality, they are generally not appropriate for knowledge discovery tasks, as the transformed features have no intuitive meaning to the user, making clusters formed in the new space difficult to interpret.

- *Feature selection:* Feature selection is concerned with locating a minimum subset of the original features that optimises one or more criteria, rather than producing an entirely new set of dimensions for the data. For cluster analysis, this involves the identification of a subset of features that best reveals the “natural" groupings of instances in the data.

  As with feature extraction, the economy of representation afforded by the use of a reduced number of features can lead to improved computational efficiency. However, in contrast to transformation-based techniques, a subset of the original dimensions is chosen, thus ensuring that each feature has a real meaning for the user. Due to the importance of interpretability in knowledge discovery, this survey focuses on these techniques.

7.2 Feature Transformation Techniques

The most commonly used technique for compression of training images is based on *principal component analysis* (PCA) [63]. An image is considered to be a vector in high-dimensional space of all possible images. The basic idea of PCA is to efficiently map high-dimensional input data to a low-dimensional subspace by reducing the redundancy and preserving as much information as possible. To achieve this goal, the directions with the largest variance of input data are found in the high-dimensional input space. The dimension of the space can be reduced by discarding the directions with small variance of the input data. By projecting the input data into this subspace, which has the principal directions for the basis vectors, we obtain an approximation with an error, which is minimal (in the least squares sense) among all linear transformations to a subspace of the same dimension.

Thus, learning is performed by estimating the *principal directions* considering all training images. Since these directions are usually obtained using the eigendecomposition, they are commonly referred to also as *eigenvectors*. An object is represented with the projections of the training images into the *principal subspace* (*eigenspace*) determined by the principal directions. It turns out that the correlation between two images can be approximated by the distance between their projections in the principal subspace. Thus, the recognition can be carried out by projecting an image of an unknown object into the principal subspace and finding the nearest projected training image [1].

First of all, the representation is easy to build. Well known statistical and algebraic methods for principal component analysis can be employed. It is very effective in terms of compression, since it requires an amount of memory as small as possible to represent input images to a certain degree of accuracy. Next, by interpolating between the projected points in the principal subspace, training images can be generalized to unfamiliar views as well. And lastly, the recognition can be performed very quickly, since it is reduced to the search for the closest point in a low-dimensional principal subspace.

Nevertheless, PCA is not the only subspace method that can be used to map high-dimensional images to a low-dimensional subspace. There exist several other techniques, each with its own properties and goal applications. In the next subsection we will briefly outline some of them.

7.2.1 PCA - the Basic Principle

Principal component analysis is a linear transformation from a high-dimensional input space to a low-dimensional feature space, which among all linear transformations guarantees the best possible representation of the high-dimensional input vectors in the low-dimensional feature space. It rotates the coordinate frame in a data-driven way, such that the variability of the input data can be efficiently described using only a small number of basis vectors.

This principle is illustrated in a simple 2-D example in Fig. 7.1. Consider eight 2-D points depicted as black dots. The goal of PCA is to find a new coordinate frame in which these eight points can be represented using only one basis vector as well as possible. This coordinate system is depicted with blue lines in Fig. 7.1 and its axes are referred to as *principal axes* (also *principal vectors* or *principal directions*). Due to the correlation between the elements of the input vectors, the first principal axis encompasses most of the variability of the input

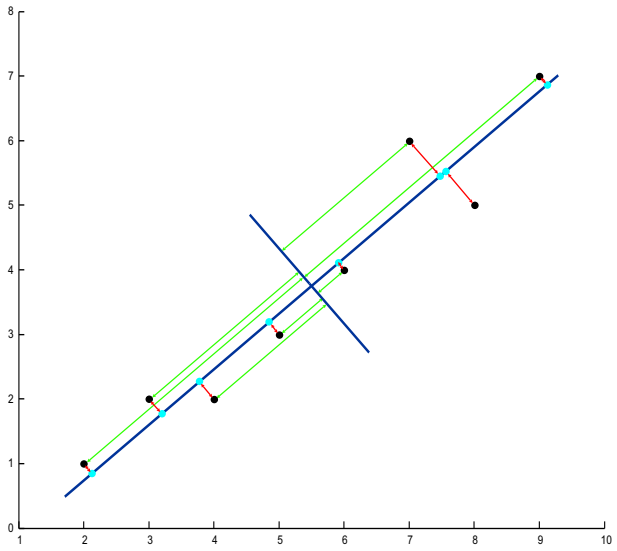


Figure 7.1: Principles of PCA.

points. Thus, by projecting the input points onto the first principal axis we obtain 1-D *principal components* which are the best possible 1-D representations of the input points. The principal components (also referred to as *coefficients*) are depicted as cyan dots in Fig. 7.1.

The principal axes are obtained in such a way that they minimize the squared reconstruction error between the input points and their representations and they maximize the variance of the principal components. If we consider Fig. 7.1, the goal is to rotate the long blue line in a such direction, which yields the red lines as short as possible and the green lines as long as possible. It turns out that these two critera, namely the minimization of the reconstruction error and the maximization of the variance, are equivalent and can be uniquely satisfied using PCA.

PCA produces very good results, if the high-dimensional input vectors are correlated. This means that they contain redundant information. PCA removes the redundancy by decorrelating the input vectors; the new coordinates of the input vectors (principal components) are uncorrelated. As a consequence, the correlated high-dimensional input vectors can be efficiently represented as the uncorrelated low-dimensional vectors of principal components making PCA a very powerful tool for data compression.

7.2.2 PCA - the Details

Principal Component Analysis (PCA), also known as the (discrete) Karhunen-Loeve or the Hotelling transform, is one of the most popular tools for dimensionality reduction of multivariate data points. It originated from Pearson [94] as a methodology for fitting planes in the least-squares sense (linear regression), and it was Hotelling (1933) [64] who proposed this technique for the purpose of analyzing the correlation structure between many random variables.

It is applied in many areas such as image analysis, pattern recognition, compression and for applications such as faces recognition, recognition of $3D$ objects under varying pose and tracking of deformable objects. These applications require a robust feature extraction method

that is capable of deriving low-dimensional features effective for preserving class separability. Such low dimensional features are also important when one considers the computational efficiency. Principal Component Analysis (PCA), a second-order method, is capable of deriving uncorrelated components and is commonly used for deriving low-dimensional representation of input images.

Why is dimensionality reduction an important issue in these fields ? First, the storage, transmission and processing of high dimensional data places great demands on computational systems. Second, with high dimensional data, it is difficult to understand the underlying structure. This is related to what is known as the "curse of dimensionality" [8]. Thus, not only that an increase in the number of features does not improve representation or segmentation results, it actually deteriorates them. High dimensional spaces are inherently sparse, thus a major portion of the observed data has a low variance estimator. Moreover, psychophysical findings indicate that "perceptual" tasks such as similarity judgment tend to be performed on a low-dimensional representation of the sensory data. Thus, the motivation for PCA is to reduce the dimensionality of the original data, while maintaining its original structure.

PCA assumes that important structure in the data actually lies in a much smaller dimensional space, and that the dimensionality reduction method loses as little relevant information as possible in the transformation from high-dimensional space to the low-dimensional one.

PCA can be understood as a fitting to a Gaussian model and choosing the uncorrelated variables with the large variance as the relevant features. Suppose that we measure d-dimensional vectors. Given N such vectors we assume a Gaussian probability distribution:

$$P(x|\Sigma,\mu) = \prod_{i=1}^{N} P(x_i|\Sigma,\mu) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{d}{2}}} \frac{1}{(\det\Sigma)^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\}$$

By the maximum likelihood procedure we find the well known results: μ is the arithmetic average and Σ is the covariance matrix.

The PCA procedure diagonalizes the covariance matrix and de-correlates the feature channels. Since the covariance matrix is a non-negative, symmetric matrix it can be diagonalized with an orthogonal transformation L. The multi-variate normal distribution reads

$$P(x|\Sigma,\mu) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{d}{2}}} \frac{1}{(\det L^T\Sigma L)^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\tilde{x}_i^T L^T)(L\Sigma^{-1}L^T)(L\tilde{x}_i)\}$$

where $\tilde{x}_i = x_i - \mu$. Let us denote the diagonalized covariance matrix by $\Lambda = L^t\Sigma L$ then

$$P(x|\Sigma,\mu) = \prod_{i=1}^{N} \frac{1}{(2\pi)^{\frac{d}{2}}} \frac{1}{(\det\Lambda)^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\hat{x}_i^T\Lambda^{-1}\hat{x}_i)\}$$

Here $\hat{x}_i = L\tilde{x}_I = L(x_i-\mu)$ are the decorrelated variables with variance which is given by the diagonal of Λ. We arrange Λ such that the eigenvalues are sorted from higher to lower values. If the lower eigenvalues are very small it means that the corresponding random variables $\hat{x}_i$ are all very close to the average and that this feature is not discriminative and doesn't carry useful information about the distribution and can be discarded in the analysis.

From this perspective the PCA can be viewed as some projection of the observations onto orthogonal axes in the space defined by the original variables. When the observed variables

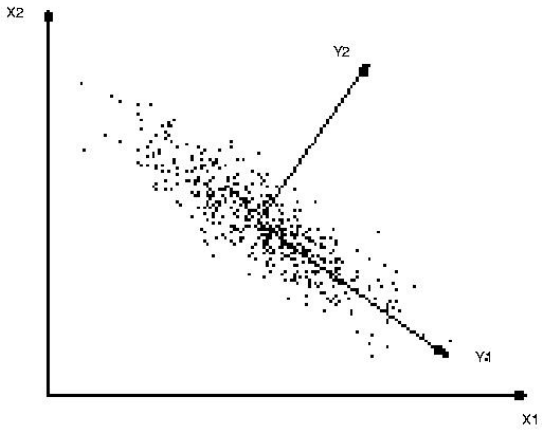


Figure 7.2: A two dimensional data set in axes (x1,x2) is transformed to its uncorrelated principal components (y1,y2). Note the the principal components are aligned with the principal axes of the ellipse shaped data set

have a non-zero correlation, the dimensionality of the data space does not represent the number of independent variables that are really needed to describe the data. The more correlated the observed variables, the smaller the number of independent variables that can adequately describe them. The criteria for obtaining the principal components is that the first axis "contains" the maximum amount of variation in the original variables. The second axis contains the maximum amount of variation orthogonal to the first. The third axis contains the maximum amount of variation orthogonal to both the first and second axis, and so on until one has the last new axis which is the amount of variation left. Figure 7.2 demonstrates a transformation from a two dimensional data set in axes (x1,x2), to its uncorrelated principal components (y1,y2).

Usually, following PCA, the data dimension reduction process involves considering for example only the first component which encompasses the largest variance of the original data. The original information is actually projected on a lower dimension space so that as little information as possible is lost in a mean-square sense.

PCA can also be described using matrix methods. Suppose that we have observations of N variables x, where $x^i = x^i_1, x^i_2, ..., x^i_d$, $i = 1, ..., N$. We would like to find new variables $y^i, i = 1, ..., N$ where $y^i = y^i_1, y^i_2, ..., y^i_d$ which are linear combinations of the x's but are uncorrelated, thus: $y = Lx$ and L is the matrix representing this linear transformation. The transformation is imposed to be self-orthogonal, $LL^T = I$, where I is the identity matrix. It follows that once the principal components are obtained, the original variables can be reproduced by: $X = L^T Y$. The constraint of de-correlation, $E(y_i y_j) = 0$ is equivalent to: $E(YY^T) = E(LX(LX)^T) = E(LXX^T L^T) = L\Lambda L^T$ where we require that the matrix $L\Lambda L^T$ is with zero off diagonal elements and with diagonal elements $\lambda_1 ... \lambda_N$ representing the variances of the Y's. Λ is the covariance matrix of the original variables X and the goal is to find the eigenvectors of the covariance matrix. These eigenvectors correspond to the directions of the principal components of the original data, and and their statistical significance is given by their corresponding eigenvalues. The transformation matrix L is then built using the eigenvectors as its columns. This

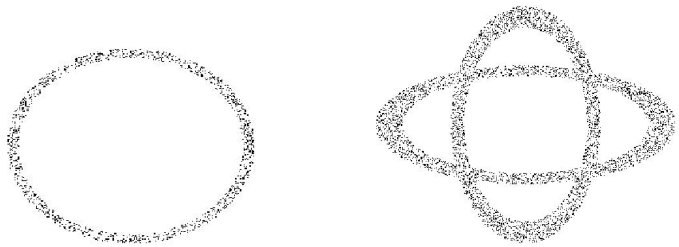


Figure 7.3: These examples cannot be modelled by a Gaussian distribution, therefore an improvement to the PCA method is required.

technique can be described as follows:

1. Collect x_i of a $d-dimensional$ data set X, $i = 1,2,...N$.
2. Calculate the mean $\bar{x}$ and subtract it from each data point: $x_i - \bar{x}$.
3. Calculate the variance-covariance matrix Λ:

$$\Lambda_{ij} = (x_i - \bar{x})(x_j - \bar{x})$$

4. Determine the eigenvectors and eigenvalues of the matrix Λ. Thus, solve: $\Lambda\alpha = \lambda\alpha$, where λ is the eigenvalue and α is an eigenvector.
5. The transformation matrix L is now constructed using the eigenvectors obtained, as its columns.
6. The next step involves sorting the eigenvectors and then select the first eigenvectors which correspond to the largest eigenvalues.

The features $y_1,...,y_N$, are called the principal components of y. They are statistically uncorrelated: $E(y_i y_j) = 0$, and their variances are equal to the eigenvalues calculated: $E(y_i^2) = \lambda_i$.

If the Gaussian model is valid, thus the higher order moments are not significant, then PCA is an excellent choice as it is easy to implement and provides good results. However, if the basic assumption that the data can be well described by a Gaussian model is violated, then PCA may yield poor results. See Fig. 7.3 for examples of data sets for which PCA cannot be used as the Gaussian model is not valid. In such cases other methods which account for the higher order moments are needed, such as non-linear PCA which extend the ability of PCA to incorporate non-linear relationships in the data [76] and Independent Component Analysis, which derives independent components by means of high order statistics [133].

7.2.2.1 Principal Component Analysis Applied to Image Analysis

Applying PCA technique to face recognition, Turk and Pentland [123] developed a well-known Eigenfaces method, where the eigenfaces correspond to the eigenvectors associated with the

largest eigenvalues of the face covariance matrix. The eigenfaces thus define a feature space, or "face space", which drastically reduces the dimensionality of the original space, and face recognition is then carried out in the reduced space.

In the work of Rousson and Deriche [102] a variational framework that extends the work of Chan, Sandberg and Vese [24] is presented where the energy functional to be minimized accounts for the deviation of the attributes from a mean value, and the covariance relationship between the attributes. The density function which describes the image data is a Gaussian distribution, so that the conditional probability with respect to the hypothesis h_i verified in a region Ω_i is:

$$P_{\Sigma_i,\mu_i} = \frac{1}{(2\pi)^{\frac{d}{2}}(det\Sigma_i)^{\frac{1}{2}}} e^{-\frac{1}{2}(I-\mu_i)^T\Sigma_i^{-1}(I-\mu_i)}, \tag{7.1}$$

where I is image data, μ_i is the mean value in the region Ω_i and Σ_i is the variance-covariance matrix between the image features involved in the region Ω_i. In such application, it seems natural to perform PCA using the variance-covariance matrix already presented to extract the most dominant features in each region.

The most straight forwards need for the application of PCA to the Gabor feature space is dimensionality reduction. Usually, Gabor filters are generated for several scales and orientations. Even a solid choice of 6 orientations and 5 scales results in values, which are entries in a $30-dimensional$ texture feature vector. Therefore, each pixel is characterized by a high dimensional vector. PCA can be applied to the features vectors to compute the eigenvectors and eigenvalues of the cross-correlation matrix. The eigenvectors are then ordered according to the value of the corresponding eigenvalues.

Some of the studies which apply PCA to Gabor filters output are as follows: Monadjemi and Mirmehdi [2] examine the covariance matrix of their feature set, and found that the relative correlation of the high frequency features was greater than the low frequency features suggesting a lack of variance in the high frequency features. Thus, an $8-dimensional$ feature space was transformed into a new $3-dimensional$ set using the new features.

To conclude, PCA is a powerful tool which enables data dimensionality reduction, to obtain the most important attributes of multidimensional data points. The PCA provides good results when the data can be well described by a a Gaussian model. When the Gaussian assumption is not valid, the use of PCA may not yield good results and other methods are required. However, as the Gaussian model is valid for several physical phenomena, PCA is a popular method. Applying PCA to the Gabor features space may provide a compact presentation of textural data, as well as generate the most relevant features for texture segmentation.

7.2.3 Applications of PCA

7.2.3.1 Electrical Impedance Tomography

[112] The recent industrial and medical interest is focused on real-time reconstruction. That is not possible in Electrical Impedance Tomography (EIT) using the classical approach, especially in 3D space. Therefore, in this paper 3D Boundary Element Method (BEM) for thin layers in EIT and the solution to the inverse problem using Neural Network is presented.
Thin layers like skull in EIT cause many problems to geometrical representation. Finite El-

ement Method, which is time-consuming in 3D space, is usually used to the solution of the forward problem. The BEM, which represents only a discretization of the surface, reduces the number of necessary elements as a consequence the computation time.
In order to solve the inverse problem, the Neural Network method was applied. For selection of neural network size, which is the one of more important and complex problems, the Principal Component Analysis (PCA) is used. PCA decomposes high-dimensional data into a low-dimensional subspace component. In this way, the size of input vector to train the neural networks is limited.

7.2.3.2 Application of recirculation neural network and principal component analysis for face recognition

[17] Bryliuk and Valery Starovoitov described their study of the recirculation neural network (RNN) for calculation principal components from a large set of face images. They developed a simple and fast algorithm was used to train RNN. Obtained principal components were used for face recognition. They also explored the ability of the RNN to reconstruct face images.

7.2.3.3 PCA with Missing Data and its Application to Polyhedral Object Modeling

[106] Observation-based object modeling often requires integration of shape descriptions from different views. To overcome the problems of errors and their accumulation, the authors have developed a weighted least-squares (WLS) approach which simultaneously recovers object shape and transformation among different views without recovering interframe motion. They showed that object modeling from a range image sequence is a problem of principal component analysis with missing data (PCAMD), which can be generalized as a WLS minimization problem. An efficient algorithm is devised. After they have segmented planar surface regions in each view and tracked them over the image sequence, they construct a normal measurement matrix of surface normals, and a distance measurement matrix of normal distances to the origin for all visible regions over the whole sequence of views, respectively. These two matrices, which have many missing elements due to noise, occlusion, and mismatching, enable them to formulate multiple view merging as a combination of two WLS problems. A two-step algorithm is presented. After surface equations are extracted, spatial connectivity among the surfaces is established to enable the polyhedral object model to be constructed. Experiments using synthetic data and real range images show that the approach is robust against noise and mismatching and generates accurate polyhedral object models.

7.2.3.4 Linear Subspace Technique for Pattern Classification

In [127] Vaswani presented a linear pattern classification algorithm, Principal Component Null Space Analysis (PCNSA) which uses only the first and second order statistics of data for classification and compared its performance with existing linear algorithms. PCNSA first projects data into the PCA space in order to maximize between class variance and then finds separate directions for each class in the PCA space along which the class has the least variance (in an ideal situation the null space of the within class covariance matrix) which is defined as the

”approximate null space” (ANS) of the class. To obtain the ANS, the authors calculate the covariance matrix of the class data in PCA space and find its eigenvectors with least eigenvalues. The method works on the assumption that an ANS of the within-class covariance matrix exists, which is true for many classification problems. A query is classified as belonging to the class for which its distance from the class mean projected along the ANS of the class is a minimum. Results for PCNSA's superior performance over LDA and PCA were shown for object recognition.
In [128] Vashwani and Chelappa discussed the PCNSA algorithm more precisely and derived tight upper bounds on its classification error probability. They used these expressions to compare classification performance of PCNSA with that of Subspace Linear Discriminant Analysis (SLDA).
In [129] the authors proposed a practical modification of PCNSA called progressive-PCNSA that also detects ”new” (untrained classes). Finally, they provided a brief experimental comparison of PCNSA, progressive-PCNSA and SLDA for three image classification problems - object recognition, facial feature matching and face recognition under large pose/expression variation. They also showed application of PCNSA to two classification problems in video - an abnormal activity detection problem and an action retrieval problem.

7.2.3.5 High-Resolution Infrared Measurement Compression and Retrieval

[65] A simulation study is used to demonstrate the application of principal component analysis to both the compression of, and meteorological parameter retrieval from, high-resolution infrared spectra. The study discusses the fundamental aspects of spectral correlation, distributions, and noise; the correlation between principal components (PCs) and atmospheric-level temperature and water vapor; and how an optimal subset of PCs is selected so a good compression ratio and high retrieval accuracy are obtained. Principal component analysis, principal component compression, and principal component regression under certain conditions are shown to provide 1) nearly full spectral information with little degradation, 2) noise reduction, 3) data compression with a compression ratio of approximately 15, and 4) tolerable loss of accuracy in temperature and water vapor retrieval. The techniques will therefore be valuable tools for data compression and the accurate retrieval of meteorological parameters from new-generation satellite instruments.

7.2.3.6 Multisensor classification

[28] The potential of high-resolution radar and optical imagery for synoptic and timely mapping in applications such as resource management, disaster delineation and weather mapping is well-known. Numerous methods have been developed to process and quantify useful information from remotely sensed images. Most image processing techniques use texture based statistics combined with spatial filtering to separate target classes or to infer geophysical parameters from pixel radiometric intensities. The use of spatial statistics to enhance the information content of images, thereby providing better characterization of the underlying geophysical phenomena is a relatively new technique in image processing.
The authors are currently exploring the relationship between spatial statistical parameters of various geophysical phenomena and those of the remotely sensed image by way of principal

component analysis of radar and optical images. Issues being here explored are the effects of incorporating PCA into land cover classification in an attempt to improve its accuracy. Preliminary results of using PCA in comparison with unsupervised land cover classiffcation are presented.

7.2.3.7 Microarray Experiments

[100] A series of microarray experiments produces observations of differential expression for thousands of genes across multiple conditions. It is often not clear whether a set of experiments are measuring fundamentally different gene expression states or are measuring similar states created through different mechanisms. It is useful, therefore, to define a core set of independent features for the expression states that allow them to be compared directly. Principal components analysis is a statistical technique for determining the key variables in a multidimensional data set that explain the differences in the observations, and can be used to simplify the analysis and visualization of multidimensional data sets. The authors showed that application of PCA to expression data (where the experimental conditions are the variables, and the gene expression measurements are the observations) allows to summarize the ways in which gene responses vary under different conditions. Examination of the components also provides insight into the underlying factors that are measured in the experiments. The authors applied PCA to the publicly released yeast sporulation data set (Chu et al. 1998). In that work, 7 different measurements of gene expression were made over time. PCA on the time-points suggests that much of the observed variability in the experiment can be summarized in just 2 components—i.e. 2 variables capture most of the information. These components appear to represent (1) overall induction level and (2) change in induction level over time. The authors also examined the clusters proposed in the original paper, and show how they are manifested in principal component space. The results are available on the internet at http://www.smi.stanford.edu/projects/helix/PCArray.

7.2.3.8 Blind Multiuser Detection

[51] SIPEX-G is a fast converging, robust, gradient-based PCA algorithm that has been recently proposed by the authors. Its superior performance in synthetic and real data compared with its benchmark counterparts makes it a viable alternative in applications where subspace methods are employed. Blind multiuser detection is one such area, where subspace methods, recently developed by researchers, have proven effective. In this paper, the SIPEX-G algorithm is presented in detail, convergence proofs are derived, and the performance is demonstrated in standard subspace problems. These subspace problems include direction of arrival estimation for incoming signals impinging on a linear array of sensors, non-stationary random process subspace tracking, and adaptive blind multiuser detection.

7.2.3.9 Material Science

[114] The relationship between apparently disparate sets of data is a critical component of interpreting materials' behavior, especially in terms of assessing the impact of the microscopic characteristics of materials on their macroscopic or engineering behavior. In this paper the authors demonstrated the value of principal component analysis of property data associated with

high temperature superconductivity to examine the statistical impact of the materials' intrinsic characteristics on high temperature superconducting behavior.

7.2.3.10 Statistics of Shape via PCA on Lie Groups

[91] Principal component analysis has proven to be useful for understanding geometric variability in populations of parameterized objects. The statistical framework is well understood when the parameters of the objects are elements of a Euclidean vector space. This is certainly the case when the objects are described via landmarks or as a dense collection of boundary points. The authors have been developing representations of geometry based on the medial axis description or m-rep. Although this description has proven to be effective, the medial parameters are not naturally elements of a Euclidean space. In this paper the authors showed that medial descriptions are in fact elements of a Lie group. They developed methodology based on Lie groups for the statistical analysis of medially-defined anatomical objects.

7.3 Feature Transformation in Image Analysis

In the area of visual processing (image analysis, object recognition, texture classification, texture segmentation and image retrieval, among others) there are still several fundamental questions which remain unanswered. One of the core components of any image analysis architecture is the feature transformation: a mapping from the space of image pixels to a feature space with better properties for representation and interpretation. An important key question in visual processing regards the selection of such transformation for the interpretation of the images' contents. In this manuscript we address this issue in the context of Gabor filtering. Application of Gabor filtering to image processing is widespread, and we only present here a small portion of the large amount of studies in this field.

The raw image data can be gray level values on a rectangular grid, which may describe the intensity of illumination, radiation and even heat. Naturally, image information is not restricted to a single value per pixel location, e.g. color images. In order to interpret image contents, one may generate some features which capture or enhance some property of the image. Thus, a typical image processing algorithm involves a selection of a suitable representation space. This can naturally be the image itself, but other common choices are windowed Fourier transforms, the Gabor representation [70], Wavelet transforms [25, 77, 124], local histograms [115], the local structure tensor [81] and the space of oscillating functions [130]. Once the representation space is selected, some features are extracted, e.g the magnitude of the response of the Gabor filters and particular moments which are calculated from local histograms [70],[20]. These features can be used for image representation, segmentation etc. For example, some measure, Kulback-Leibler, Mutual information etc., can be introduced on the features. Also, some objective function can be defined using the image features, and the segmentation, diffusion or other image processing task can be formulated as an optimization problem.

Image representation and modelling can be roughly divided into two classes: statistical based approaches and filtering based approaches. Statistical modelling is based on the assumption that each image has unique statistical attributes. Among them are: local statistical features [27], random field models [29, 57, 67], co-occurence matrices [50], second order statistics [26],

statistics of texton attributes [72], local linear transforms [124], and a gaussian distribution modelling of the structure tensor [81]. The filtering modelling is based on applying some filter bank to the image and considering the filters' responses as information about the local behavior of the image.

An important approach in computer vision is to learn as much as possible from biological visual systems. Physiological as well as psychophysical findings indicate that biological processing is based on local signatures of frequency and orientations. As Gabor functions are tuned to both orientation and frequency, it is believed that simple cells in the visual cortex can be modelled by Gabor functions [35, 85], and that the Gabor scheme provides a suitable representation for visual information in the combined frequency-position space [95]. Moreover, when considering the spatial and frequency spaces, Gabor filters offer the best trade off between spatial and frequency widths of uncertainty. Indeed, the Gabor representation has been shown to be optimal in the sense of minimizing the joint two-dimensional uncertainty in the combined spatial-frequency space [55, 35]. The analysis of Gabor filters was generalized to multi-window Gabor filters [134] and to Gabor-wavelets [79, 68, 95, 134], and studied both analytically and experimentally on various classes of images [15, 45, 134]. These attributes have made the use of Gabor filters widespread in image analysis: the use of Gabor filters for image representation was studied in the context of the frame theory to ensure the ability to use Gabor functions for image analysis and synthesis. Gabor filters are also applied to image classification, in areas such as texture representation and segmentation, affine invariant measures and recognition tasks such as face detection. It is important to note that for the second type of tasks, it is not required that the Gabor functions used are a complete set, in the sense that signals can be reproduced.

When applying Gabor filtering methods it is apparent that there is an infinite number of possible pavings of the spatial-frequency space. One should select the number of spectral components for spanning the global frequency bandwidth. The main issues involved in the Gabor filtering approach are the characterization and number of the channels to be used as well as the extraction of appropriate features from the filtered images. In appropriate we mean, a feature set which may lead to a successful representation or segmentation. The selection of the Gaborian features set can be divided into three phases:
1. The attributes of the Gabor filters used (number of scales, orientations etc.).
2. The generation of features from the Gabor responses.
3. Using these features to achieve the task in hand, representation, segmentation etc.

The first phase involves obtaining a set of filters with good coverage of the spatial-frequency domain. The guidelines for this selection are based on physiological findings and on the requirement that the selected filters constitute a frame. In several applications, these parameters are selected according to experience, and the common wisdom is that a window which is too large will make the Gabor feature less representative for the local attributes of the image, and a window which is too small will result in high sensitivity to noise.

Once the raw Gabor responses are generated, the texture features are calculated. If segmentation is what we are interested in, then the next phase is usually implemented by using clustering algorithms or PDE based mechanisms, such as active contours (snakes).

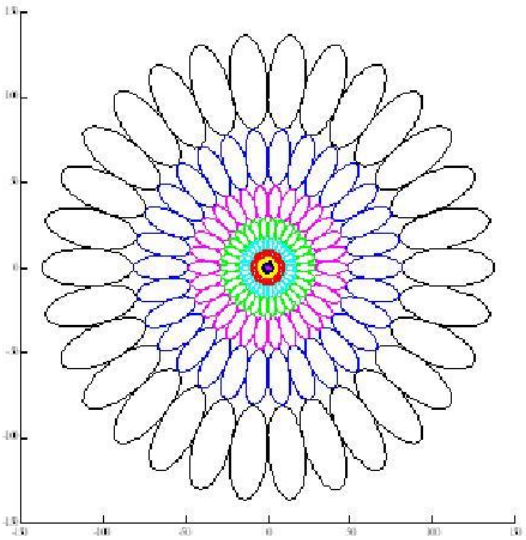


Figure 7.4: In this diagram the responses in the frequency domian of a possible set of Gabor wavelets is presented. A common design strategy for a Gabor filter bank is to ensure that the half-peak magnitude support of the filters' responses in the frequency domain touch each other.

7.4 Gabor Transform

A Gabor filter centered at the 2D frequency coordinates (U,V) has the general form of:

$$h(x,y) = g(x',y')\exp(2\pi i(Ux+Vy)) \tag{7.2}$$

where

$$(x',y') = (x\cos(\phi)+y\sin(\phi), -x\sin(\phi)+y\cos(\phi)), \tag{7.3}$$

$$g(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2}{2\lambda^2\sigma^2}-\frac{y^2}{2\sigma^2}\right), \tag{7.4}$$

and λ is the aspect ratio characterizing the elliptic Gaussian window, σ is the scale parameter, and the major axis of the Gaussian is oriented at angle ϕ relative to the x-axis and to the modulating sinewave gratings.

Accordingly, the Fourier transform of the Gabor function is:

$$H(u,v) = \exp\left(-2\pi^2\sigma^2((u'-U')^2\lambda^2+(v'-V')^2)\right) \tag{7.5}$$

where, (u',v') and (U',V') are rotated frequency coordinates.

Thus, $H(u',v')$ is a bandpass Gaussian with its minor axis oriented at angle ϕ from the u-axis, and the radial center frequency F is defined by : $F = (U^2+V^2)^{1/2}$, with orientation $\theta = \arctan(V/U)$. Since maximal resolution in orientation is desirable, the filters whose sinewave

gratings are co-oriented with the major axis of the modulating Gaussian are usually considered ($\phi = \theta$ and $\lambda > 1$), and the Gabor filter is reduced to:

$$h(x,y) = g(x',y')exp(2\pi iFx'). \tag{7.6}$$

It is possible to generate Gabor wavelets from a single mother-Gabor-wavelet by transformations such as: translations, rotations and dilations. We can generate, in this way, a set of filters for a known number of scales, S, and orientations K:

$$h_{mn}(x,y) = a^{-m}h(\frac{x'}{a^m}, \frac{y'}{a^m}), \tag{7.7}$$

where (x',y') are the spatial coordinates rotated by $\frac{\pi n}{K}$ and scaled by powers $m = 0,\ldots,S-1$. The responses of Gabor wavelets in the frequency spectrum can be seen in figure 7.4.

Alternatively, one can obtain Gabor wavelets by logarithmically distorting the frequency axis [95] or by incorporating multi-windows [134]. In the latter case one obtains a more general scheme wherein subsets of the functions constitute either wavelet sets or Gaborian sets.

There are several degrees of freedom in selecting the family of Gabor filters to be used: number and values of scales, frequencies and orientations. In order to obtain good segmentation results, the filters should be carefully selected, so that they represent the data and the differences in textures within the data in an accurate way. Although some techniques were suggested to obtain such selection [45, 121], they are complex to implement and in many applications the selection is based on physiological constraints, as well as guidelines offered by Lee [79] which are more related to the issue of completeness.

The feature space of an image is obtained by the inner product of this set of Gabor filters with the image:

$$\begin{aligned} W_{mn}(x,y) &= R_{mn}(x,y) + iJ_{mn}(x,y) \\ &= I(x,y) * h_{mn}(x,y). \end{aligned} \tag{7.8}$$

Once this representation is generated, one may use all channels, or use an appropriate subspace.

7.5 The Design of the Gabor Filter Bank

The issue of designing Gabor filter banks is important for image representations and segmentation. The guidelines for the filter bank design for each task are quite different. For image representation we consider selecting an adequate set of filters, which constitute a basis or a frame (see section 7.5.4). For image segmentation the issue of frame is not really important. Then, we would like to use the smallest number of filters needed for accomplishing discrimination between different textures.

In this section we refer to general issues concerning the Gabor filters and filter bank.

7.5.1 Gabor "coherent states" vs. Gabor wavelets

When Gabor published his work regarding the uncertainty principle, he referred to Gaussian functions that were modulated by an oscillatory function. Gabor [55] showed that there exists a "quantum principle" for information which is analogous to the Heisenberg's uncertainty principle in physics. This principle states that we cannot tell the exact frequency of a signal for a specific location. Thus, there is a trade-off between time resolution and frequency resolution, and there is a lower bound on their product. Gabor also discovered that Gaussian modulated complex exponentials provide the minimal uncertainty in the combined spatial-frequency spaces. Gabor dealt with functions which have a fixed Gaussian while the frequency of the modulated wave varies. These functions are known in quantum physics as the canonical coherent states generated by the Weyl-Heisenberg group. Decomposition of a signal to its projections onto these functions is known as the windowed Fourier transform. A $2D$ Gabor function was first given by Daugman [35]:

$$G(x,y) = \frac{1}{2\pi\sigma\beta} e^{-\pi[\frac{(x-x_0)^2}{\sigma^2}+\frac{(y-y_0)^2}{\beta^2}]} e^{i[\xi_0 x+\nu_0 y]}, \tag{7.9}$$

where (x_0,y_0) is the center of the filter in the spatial domain and (ξ_0,ν_0) is the center frequency of the filter in the frequency domain. σ and β are the standard deviations of the elliptical Gaussian along x and y.

However, results from vision research indicate that the central frequencies of localized frequency operators have octave relationships. This means that the effective spatial width of the Gabor filter becomes narrower as the frequency increases, and thus frequency and scale are correlated. Daugman suggested that $2D$ Gabor wavelets sampling the frequency domain in a log-polar manner. Thus, the window size is allowed to change according to the frequency. A particular Gabor elementary function is used as the mother-wavelet, and the whole family of Gabor wavelets is generated by scaling and rotation as follows:

$$\psi_\theta(a,x,y,x_0,y_0) = \|a\|^{-1}\psi_\theta(\frac{x-x_0}{a},\frac{y-y_0}{a}), \tag{7.10}$$

where

$$\psi_\theta(x,y) = \psi(x',y'), \tag{7.11}$$

and (x',y') were defined in eq. 7.3. The image processing community gives more attention to the Gabor wavelets rather than the classical Gabor functions that Gabor himself referred to. This is because it is easy to implement, and because the inverse relationships between scale and frequency were in synchronization with the wavelets paradigm.

7.5.2 Gabor Filter Bank Design for Image Representation

Gabor filters are characterized by their frequency and orientation as well as their frequency and orientation bandwidths. By varying these parameters a Gabor filter spanning any elliptical portion of the spatial frequency domain can be generated. Any function $f \in L^2$ can be expressed as a weighted sum of appropriately shifted Gabor functions, the Gabor expansion.

$$f(x,y) = \sum_{r,s,m,n=-\infty}^{\infty} \beta_{rsmn} g(x-x_r, y-ys) e^{2\pi i[U_m(x-x_r)+V_n(y-y_s)]} \tag{7.12}$$

where the sequence of shifts $\{x_r\}$ and $\{y_s\}$, and modulation frequencies $\{U_m\}$ and $\{U_n\}$ have constant spacings x, y, u, v which satisfy $xu = yv = 1$. This expansion is a complete representation as the original signal can be exactly reconstructed from the expansion coefficients. Although the original introduction of Gabor filters dates back to 1946, methods for analysis and synthesis of signals were suggested only 40 years later using complex methods such as the Zak transform. However, since the Gabor representation is redundant we may use a finite number of Gabor filters to obtain a good image representation.

When considering sampling of the Gabor filters, we should remember that Gabor filters are not band limited, and thus some aliasing will occur regardless of how well we sample these filters. The most obvious constraint for sampling a Gabor filter which is characterized by a frequency value F is that the sampling rate f_s satisfies $f_s > 2F$ in accordance with the Nyquist rule. Using Gabor filters for image representation is not an easy task since the Gabor elementary functions do not constitute an orthonormal basis. One solution to this problem, developed by Bastiaans [7], is to introduce an auxiliary function which is biorthogonal to the Gaussian window, so that it is used in the synthesis process. Porat and Zeevi proposed a scheme suitable for visual information representation through two dimensional Gabor elementary functions [95]. They also generalized this scheme to account for position dependent Gabor-sampling rate, oversampling, logarithmic frequency scaling and phase quantization characteristics of the visual system.

The next step is to create the Gabor filter bank to be used. The approaches to accomplish this task are divided to supervised and unsupervised methods. In supervised methods some prior knowledge about the contents of the image is used to create the filter bank, whereas in unsupervised methods the guidelines are more general, i.e. not specific to a certain image.

Supervised methods to create a Gabor filter bank were proposed by Bovik et al [15] and they include the characteristics of the power spectrum of predefined textures. Dunn and Higgins [45] have proposed a method to select the optimal parameters of a filter to obtain texture segmentation based on knowledge of the statistics of the textures present in the image.

Unsupervised methods to construct a filter bank are more attractive as no prior knowledge about the image in hand is required, and the framework is general in nature, not image specific. Instead of trying to identify the filters which are the most correlated with the textures within the image, a filter bank is created, containing filters that offer an almost uniform coverage of the spatial-frequency domain.

Daugman [35] referred to the constraints on degrees of freedom in the two dimensional Gabor filter family. He has established the relationship between the spatial-frequency bandwidth in octaves $\Delta\omega$, and the orientation half bandwidth $\Delta\theta_{\frac{1}{2}}$ (where the half-angle is measured relative to the x axis):

$$\Delta\theta_{\frac{1}{2}} = arcsin(\lambda\frac{2^{\Delta\omega}-1}{2^{\Delta\omega}+1}) \tag{7.13}$$

and λ is the spatial aspect ration between the width and the length of the filter. A small λ favors orientation selectivity at the expense of spatial-frequency selectivity, whereas a large λ favors spatial-frequency selectivity at the expense of orientation.

Filter size is also an important issue in filter design. The objective of applying a local Gabor function to an image, is to estimate the energy in the filter's output in that region. Each Gabor filter is a band-pass filter with frequency and orientation selective properties. It turns out

that accurate edge location and accurate features estimation are conflicting tasks. High spatial resolution is required for the accurate edge localization, while high spatial frequency resolution (and as an immediate result of the uncertainty principle, poor spatial resolution) is required for accurate features estimation. Thus, even after Gabor filters are selected to obtain the best trade-off in mutual space-frequency resolution, we still have an infinite number of possible filters. One option proposed by Jain and Farrokhnia use a gaussian window with a space constant σ which is proportional to the intensity variations in the image [70].

7.5.3 Gabor Wavelets Design for Image Representation

A central study in Gabor filters design for image representation is that of Lee [79]. In his study a derivation of a class of $2D$ Gabor wavelets is presented, with their parameters properly constrained by neurophysiological data on simple cells in the visual cortex and by the wavelet theory. The Daubechies completeness criteria for $1D$ wavelets is extended to $2D$. Lee showed in his work the constraints on number of scales, orientations and frequencies so that a tight frame is obtained.

First, Lee uses neurophysiological constraints which are widespread in the vision community. Following the discovery of Hubel and Wiesel [66] that the primary visual cortex in mammalian brains has a crystalline organization, Marcelja [85] and Daugman [35] suggested that the simple cells in the visual cortex can be modelled by Gabor functions. More recent neurophysiological studies suggests that a wavelet like behavior of the Gabor functions is more adequate for the presentation of the simple cells in the visual cortex. Lee starts his derivation of the Gabor wavelets using a general Gabor function:

$$\psi(x,y,\xi_0,\nu_0,x_0,y_0,\rho,\theta,\sigma,\beta) = \frac{1}{\sqrt{\pi\sigma\beta}} e^{-\frac{(x-x_0)'^2}{2\sigma^2}+\frac{(y-y_0)'^2}{2\beta^2}} e^{i(\xi_0(x-x_0)+\nu_0(y-y_0)+\rho)} \tag{7.14}$$

where the filter in centered at $x = x_0, y = y_0$ in the spatial domain and at $\xi = \xi_0, \nu = \nu_0$ in the frequency domain. σ and β are the standard deviations of an elliptical gaussian along the x and y axes. θ is the orientation of the filter, and ρ is the absolute phase which can be set to zero. $(x-x_0)'$ and $(y-y_0)'$ are defined according to eq. 7.3. Next, Lee uses physiological constraints to reduce the number of degrees of freedom:

Constraint1: The gaussian envelope us usually elliptical, with an aspect ratio of $1.5-2.0$.
Constraint 2: The plane wave with frequency (ξ_0,ν_0) tends to have its "propagating direction" along the short axis of the elliptical gaussian. This implies that the center frequency (ξ_0,ν_0) of the filter is related to the rotation angle θ of the modulating gaussian by $\xi_0 = \omega_0 cos\theta$ and $\nu_0 = \omega_0 sin\theta$. The orientation of the filter is thus aligned with the long axis of the elliptical gaussian.
Constraint 3: The half amplitude bandwidth of the frequency response is about 1 to 1.5 octaves along the optimal orientation. The relationship between σ and ω_0 can be derived to be:

$$\sigma = \frac{k}{\omega_0}, \tag{7.15}$$

where $k = \sqrt{2ln2}\frac{2^{\phi}+1}{2^{\phi}-1}$ where ϕ is the bandwidth in octaves. Imposing these constraints, the following Gabor filters are obtained (where for simplicity, $x_0 = 0, y_0 = 0$:

$$\psi(x,y,\omega_0,\theta) = \frac{\omega_0}{\sqrt{2\pi}k} e^{-\frac{\omega_0^2}{8k^2}(4(xcos\theta+ysin\theta)^2+(xsin\theta+ycos\theta)^2)} e^{i\omega_0(xcos\theta+ysin\theta)} \tag{7.16}$$

Further, he forces these filters to be admissible wavelets using Constraint 4: admissible wavelets are functions having zero mean. Finally, he got this family of Gabor filters:

$$\psi(x,y,\omega_0,\theta) = \frac{\omega_0}{\sqrt{2\pi}k} e^{-\frac{\omega_0^2}{8k^2}(4(xcos\theta+ysin\theta)^2+(xsin\theta+ycos\theta)^2)} (e^{i\omega_0(xcos\theta+ysin\theta)} - e^{\frac{k^2}{2}}) \tag{7.17}$$

This family of Gabor wavelets can be generated by the $R^+ \times R^2 \times SO(2)$ group of rotations, dilations and translations from the following mother Gabor wavelet:

$$\psi(x,y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{8}(4x^2+y^2)} (e^{ikx} - e^{-\frac{k^2}{2}}) \tag{7.18}$$

Next, Lee requires that the family of Gabor wavelets generated constitutes a tight frame. We will elaborate on frames in the next section.

7.5.4 The frame criterion for Gabor Wavelets

A set of vectors or functions $e_i (i = 1, ..., \infty)$ spans a Hilbert space **H** if any element of that space can be expressed as a linear combination of members of that set. Thus, each vector $v \in \mathbf{H}$ can be reconstructed by a linear superposition of the set members:

$$\mathbf{v} = \sum_i c_i \mathbf{e_i},$$

where c_i are scalars. The set is a basis for the vector space if the scalars c_i are unique for any $v \in \mathbf{H}$, or in other words if the set of vectors is independent. The set forms an orthonormal base if the following condition is met:

$$\langle \mathbf{e_i}, \mathbf{e_j} \rangle = \delta_{ij}.$$

In this case, the scalars c_i are easily calculated:

$$\langle \mathbf{v}, \mathbf{e_j} \rangle = \sum_i c_i \langle \mathbf{e_i}, \mathbf{e_j} \rangle = c_j.$$

If the set of functions or vectors is not an orthonormal basis, but still spans the vector space, another set of vectors e_i^*, called the dual basis, is required in order to determine the expansion coefficients. The orthonormality condition is extended to:

$$\langle \mathbf{e_i}, \mathbf{e_j^*} \rangle = \delta_{ij}.$$

A vector $v \in \mathbf{H}$ can be fully reconstructed using:

$$\mathbf{v} = \sum_i c_i \mathbf{e_i},$$

where the scalars c_i are now calculated using:

$$\langle \mathbf{v}, \mathbf{e_j^*} \rangle = \sum_i c_i \langle \mathbf{e_i}, \mathbf{e_j^*} \rangle = c_j.$$

The Gabor wavelets defined in the previous section are an example for a set of functions that are dependent, and thus cannot be an orthonormal basis. The Gabor family and its dual can be used to span the vector space of L^2 functions. However, it is over-complete: there is a large degree of redundancy in representing a function using the Gabor family. The Gabor family and similar sets of functions are studied in the context of frames. Frames were first introduced by Duffin and Schaeffer [44] in relation to non-harmonic Fourier series. The definition of frames is as follows: A family of functions ψ_{mn}, $m,n \in \mathbf{Z}$ in a Hilbert space $\mathbf{H}$ is called a **frame** if there exists $A > 0, B < \infty$ so that, for all $u \in \mathbf{H}$,

$$A\|u\|^2 \leq \sum_{mn} |\langle u, \psi_{mn} \rangle|^2 \leq B\|u\|^2. \tag{7.19}$$

A and B are called the frame bounds. The relationship between A and B provides information about the quality of the frame. The term $\frac{A+B}{2}$ is a measure of the redundancy of the frame and the ratio $\frac{B}{A}$ is a measure of the tightness of the frame. It was shown that if A and B are sufficiently close to each other, $A \approx B$, there is a simple formula for a numerically stable reconstruction of u without using the dual frame:

$$u \approx \frac{2}{A+B} \sum_{mn} \langle \psi_{mn}, u \rangle \psi_{mn}. \tag{7.20}$$

The frame is then called a tight frame. A tight frame can be treated as an orthonormal base for image analysis and synthesis.

Daubechies [34] has established a frame criterion for one-dimensional wavelets and an estimate for the frame bounds; Her conclusion was: "...if ψ is at all descent (reasonable decay in time and frequency, $\int dx \psi(x) = 0$), then there exists a whole range of scaling and translation factors so that the corresponding ψ_{mn} constitute a frame". Lee [79] has generalized the frame criterion to two dimensions for the Gabor wavelets and has provided the actual number of scales and orientations needed, given the scaling and translation factors, in order to obtain a frame. In his work, Lee considers the following wavelet family:

$$\psi(x,y,\omega_0,\theta) = \frac{\omega_0}{\sqrt{2\pi}k} \exp\left(-\frac{{\omega_0}^2}{8k^2}(4x^2+y^2)\right)\left(\exp\left(ix'\right) - \exp\left(\frac{-k^2}{2}\right)\right), \tag{7.21}$$

where ω_0 is the radial frequency in radians per unit length and θ is the wavelet orientation in radians. Lee has shown that if ψ is admissible there exists a range of scale, orientation and translation factors (a_0, θ_0 and b_0) for which the ψ family constitute a frame. He also provides the upper bounds for these parameters.

In his work Lee provides the frame bounds A, B for $2D$ Gabor wavelets. When $A > 0$ and $B < \infty$ the set of filters $\psi_{m,n,k,l}$ constitute a frame. The upper bound for the scale, orientation and translation parameters are simply a requirement that $A > 0$. Lee computed several frame bounds for several values of parameters to obtain tight frames. We refer the reader to his detailed derivation[79].

Another approach, grounded in practical considerations was proposed by Manjunath and Ma [84]. The Gabor filter bank is designed to cover the region of interest in the frequency spectrum. The Gabor filters are constructed so that their half peaks in the frequency domain touch each other. Thus, having the number of orientations and frequencies necessary to provide a tight frame, we can use their scheme generate a set of Gabor wavelets of the form:

$$g_{m,n}(x,y) = a^{-m} g(x'_n, y'_n), \tag{7.22}$$

where (x'_n, y'_n) are the rotated coordinates with angle θ_n (see equation 7.3) and

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2})+2\pi i w x}. \tag{7.23}$$

If U_l and U_h are the lower and upper center frequencies of interest respectively, K is the number of orientations and S is the number of scales then we can obtain the following constraints for building our filters bank:

$$\begin{aligned} a &= \frac{U_h}{U}^{\frac{1}{S-1}} \\ \sigma_u &= \frac{(a-1)U_h}{(a+1)\sqrt{2ln2}} \\ \sigma_v &= tan(\frac{pi}{2k}) * (Uh - 2ln(\frac{2\sigma_u^2}{Uh}) * (2ln2 - \frac{(2ln2)^2\sigma_u^2}{U_h^2})^{-\frac{1}{2}}, \end{aligned} \tag{7.24}$$

and $\sigma_u = \frac{1}{2\pi\sigma_x}$, $\sigma_v = \frac{1}{2\pi\sigma_y}$.

7.5.5 Gabor Filter Bank Design for Image Segmentation

In this case the input image is assumed to be composed of several textures. When Gabor filters are applied to texture images the filters parameters are usually predetermined ad hoc for the particular task. The design of a Gabor filter bank for segmentation is conceptually different from the same process for representation. While completeness is the main concern for representation, capturing the differences between image patches is the key issue for segmentation. It is even better if the filter selection is optimized. Thus, the process begins with a motivated selection of a subset of Gabor filters which are best tuned for the segmentation task.

Several approaches were suggested for this task: Bovik et al suggested tuning the filters to the dominant components in the FFT of the textures involved [15]. Teuner et al [118] partitioned the image into cells which are decomposed using a dyadic Gabor transfrom. Each cell is examined to find the most significant spectral component. This information is then used to create an appropriate filter bank. Zhu et al [104] applies the minimax entropy principle which suggests that an optimal set of filters should be chosen to minimize the Kullback-Leibler distance between the true and estimated probability densities. Weldon, Higgins and Dunn [121] proposed to minimize the image segmentation error by modelling the output statistics of a Gabor filtered texture with a Rician distribution. Another approach taken by Jain and Farrokhnia is to evaluate the least squares error obtained when the image is reconstructed using only a subset

of the original filters [70]. The choice of Gabor filters and their parameters is motivated by both constraints taken from the human visual system (e.g. frequency bandwidth of one octave) as well as a goal to construct an approximate basis for a wavelet-like transform. To make the computational burden easier, they suggest to discard as many filters as possible from the full set of filters so that we maintain at least 95% of their power spectrum energy. This approach was also taken by Guo et al [60]. Moreover, Guo et al suggest to keep only the Gabor channels output (considered as the filtered image) which are "far away" from each other. If the distance between two feature images is small, then there is a redundancy and using both channels is not necessary. The metric used for comparison is the regular L^2 norm, though other norms can be considered as well. Other applications use a filter bank with parameters that guarantee that we obtain a tight frame. Campbell and Thomas had applied a genetic algorithm to obtain the set of filter parameter[22].

As can be seen, although segmentation and image representation are quite different issues, filter bank design is heavily related to the ability to choose a set of filters which enables good image representation.

7.6 Gabor Features Generation

Assuming that the primitives of natural textures are indeed localized Gabor filters tuned to frequency and orientation, texture analysis takes the form of inner products or correlation of such primitives with textured images. Existing techniques may use either complex-valued Gabor filters [15] or real-valued, even symmetric Gabor filters [70]. The actual filter responses that result from application of the Gabor filter bank may be used directly, or some non-linear thresholding operation may be applied to these values. Other approaches involve calculating the norm of Gabor response, and use this norm (some L^p norm) as an indication for the correlation of a filter to the image patch. For example the magnitude of the output of the Gabor functions can be used [15] or full-wave rectification (summing the absolute value of the real and imaginary responses) which is a nonlinear method that is thought to be used in texture perception by the human visual system [10]. The real and complex moments features can also be used [11, 105]. Malik and Perona [83] provide some justification for using even-symmetric filters only. After the Gabor filters are applied to the image, we may generate the entities which constitute the Gabor features. The Gabor features are generated according to the application in hand. When image representation is considered, and the parameters are selected according to a frame guidelines, we actually use all the output channels of the Gabor filtered image. The magnitude of the output of the Gabor filters is a common selection as we assume that a larger magnitude indicates a good correlation of the filter to the image patch. If the textures involved are not narrow-band enough, there may be a large response to several filters, and Bovik [15] suggested to perform a Gaussian post filtering to avoid this problem. An alternative for the magnitude of the Gabor filters can be rectification of this output, i.e. summing the absolute value of the real and imaginary responses. This is in synchronization with several models of the human visual system. In the study of Jain and Farrokhnia [70], the real component of the Gabor response is only considered and each filtered channel is subjected to non linear threshold like transformation. This non linearity bears a similarity to the sigmodial activation function used in neural networks. Then some “texture energy" measure is calculated and a square error clustering algorithm is used to produce segmentation.

Lee, Mumford and Yuille [122] attempted to use the Gabor feature space for segmentation, by implementing a variant of the Mumford-Shah functional adapted to signature vectors in the Gabor space. Sochen, Kimmel and Malladi [99] apply the Beltrami flow to the Gabor channels of textured images. The filter bank is chosen so that it provides a tight frame using the guidelines of Lee. The Gabor transform is then viewed as a mapping $W : (x, y, \sigma, \theta) \rightarrow (x, y, \sigma, \theta, R, J)$ where R and J are the real and imaginary values of the transform. Thus, it can be viewed as a four-dimensional manifold embedded in a six dimensional space. The metric is this manifold is calculated and the area minimizing and features preserving Beltrami flow is then applied to the Gabor transform of the texture image. Gabor analysis was used by Sagiv, Sochen and Zeevi [18, 19, 20, 21]for texture segmentation. Following application of Gabor filters, the full Gabor information is considered, or alternatively a subspace containing the values of the scale, orientation and frequency per pixel, for which the maximal response was obtained. They view the Gabor space as a two-dimensional manifold embedded in the higher dimensional features space, using the Beltrami framework. The metric of this surface provides a good indicator of texture changes and is used, therefore, in a Beltrami-based diffusion mechanism and in a geodesic active contours algorithm for texture segmentation. Porat and Zeevi [96] proposed using localized features based on the Gabor transform of the image, and computed for this purpose the mean and variance of the localized frequency, orientation and intensity. Sagiv, Sochen and Zeevi [20], applied a Beltrami-based multi-valued snakes algorithm to this feature space. Hofmann et al [115] considered the homogeneity between pairs of texture patches by a non-parametric statistical test applied to the Gabor space. A pairwise data clustering algorithm was utilized to perform segmentation. In Paragios and Deriche [92], a supervised variational framework was developed, where the responses of isotropic, anisotropic and Gabor filters applied to the texture image were considered as multi component conditional probability density functions. This information served as the stopping term in a variation of the geodesic snakes mechanism. Zhu et al [104, 103] proposed an approach called region competition, unifying snakes, region growing and Bayes/MDL criterion by the application of a variational principle for multi- band image segmentation. This algorithm integrates the geometric benefits of the snakes/balloons mechanism with the benefits of the statistical modelling used in region growing. Sandberg, Chan and Vese [6] applied a vector-valued active contour without edges mechanism [24] to the Gabor filtered images. Vese and Osher [130] used a model which assumes that an image is a linear combination of some bounded variation function, a "cartoon" approximation of the image, and an oscillatory function which represents texture or noise, following a model proposed by Meyer [87]. The energy functional to be minimized in this approach assumes that the oscillating part is given as $div(g)$, while g itself is iteratively updated so that its norm is minimized, following a model proposed by Meyer [87]. Texture discrimination can then be accomplished by applying the active contours without edges model to the resultant g. Some approaches combine statistical modelling, structural modelling and the filter bank model. The FRAME theory proposed by Zhu et al [104, 103] combines the use of filters, random fields and maximum entropy as a unified approach for texture modelling. The features extraction in this model is based on application of a filter bank consisting of Gabor filters along with other filters to the image. The histograms of the filtered images provide estimates of the marginal probability distribution of the textured image. The maximum entropy principle is then applied to derive a distribution function which is restricted to have the same marginal distribution obtained in the first phase.

Galun et al [56] use a bottom-up aggregation framework that combines structural characteristics of texture elements with filter responses. Their process adaptively identifies the shape of texture elements and characterize them. Then, various statistics of these properties are used to distinguish between different textures. At the same time the statistics of filter responses are used to characterize textures. In their process the shape measures and the filter responses crosstalk extensively. In several applications such as face recognition or image retrieval from a database, a features vector is generated. For example, Manjunath and Ma [84] defined features vector whose components are the responses of the Gabor channels. They used the Euclidean distance between these vectors as a criterion for similarity between textures. A complementary feature space, which is less evident in previous studies involves the phase of the complex outputs of the Gabor filters. De Buff has considered this issue for simple examples [43] and Bovik et al [15] proposed a method to locate phase discontinuities. These are located by finding peaks in the gradient of the phase, or the zero crossings of the laplacian of the phase.

7.7 Feature Selection in Supervised Learning

Feature selection is concerned with locating a minimum subset of the original features that optimises one or more criteria, rather than producing an entirely new set of dimensions for the data. Most supervised learning algorithms perform poorly in the face of high dimensional data. Consequently, substantial work has been performed to develop feature selection methods to improve the results obtained on the data. Many reviews exist which comprehensively cover this area [12, 30, 80]. The topic is covered also in standard books [42, 54, 131, 119].

Supervised feature selection is usually posed as a search problem with three core components [78]:

- *Search strategy.* A strategy for exploring the space of feature subsets, including methods for determining a suitable starting point and generating successive candidate subsets.
- *Evaluation scheme.* A function to evaluate and compare candidate subsets, which serves to guide the search process.
- *Stopping criterion.* One or more requirements that must be satisfied before the selection process can be terminated.

7.7.1 Search Strategies

The search algorithms employed in supervised feature selection methods may be divided into two categories: optimal and sub-optimal approaches.

7.7.1.1 Optimal Search Strategies

An exhaustive search will guarantee an optimal solution that maximises a given criterion function by considering all potential feature subsets. However, this is computationally intractable for practical applications, as the number of posssible subsets grows exponentially with the number of dimensions in the data. The only alternative is the family of Branch & Bound algorithms.

They are able to find optimum considerably faster, but at a cost of additional requirement put on the criterion function which must fulfil the so-called *monotonicity condition*. For overview of the latest development in this field see [108].

Nevertheless, all optimal algorithms have exponential nature and as such are usually unusable for high-dimensional problems.

7.7.1.2 Optimal Search Strategies

In response, various search heurstics have been applied to the problem. Of these, greedy “hill-climbing" procedures have been mostly commonly used in the literature [see 23]. These methods provide no guarantee that an optimal solution will be produced, as a single iteration of the search procedure will not visit all regions of the search space. Forward Sequential Selection (FSS) begins with an empty subset and at each iteration generates subsets by adding the most relevant feature from those that have yet to be selected. Backward Sequential Selection (BSS) works in reverse, starting with a complete feature set and discarding the least relevant feature at each iteration. More computationally complex floating search heuristics have been shown to produce near-optimal solution by allowing back-tracking [98].

Further development of the idea of floating search led to Adaptive Floating Search [109] and the more flexible and promising Oscillatin Search [107]. Unlike other methods, the Oscillating Search repeatedly modifies the current subset of a given cardinality. Its concept makes it possible to define a number of algorithms with different properties – targeted at finding as good results as possible regardless time, or in contrary, at yielding reasonable results in short, restricted time. It may be also looked upon as a universal tuning tool which is able to improve solutions obtained in any other way.

Other feature selection approaches have made use of non-deterministic techniques such as Genetic Algorithms [e.g. 125, 132], which have been shown to efficiently search large spaces [59]. These procedures maintain a constant-size population of candidate solutions corresponding to different feature subsets. The solutions are represented by binary strings, where each bit indicates the presence or absence of a feature in the corresponding subset. By applying genetic operators such as crossover and mutation to population members, successive generations of solutions can be produced that show improvement according to an accuracy-based fitness function.

7.7.2 Evaluation Schemes

The evaluation schemes used in supervised feature selection can generally be divided into two broad categories:

7.7.2.1 Wrappers.

The *wrapper* approach [71] to feature selection makes use of the induction algorithm itself to choose a set of relevant features. The wrapper conducts a search through the feature space, evaluating candidate feature subsets by estimating the predictive accuracy of the classifier built on that subset. The goal of the search is to find the subset that maximises this criterion.

Wrapper approaches have proved effective in increasing predictor accuracy, as the bias of the learning algorithm is taken in account when choosing a subset. However, this results in a lack of generality, as their dependence on a given classifier usually requires the selection process to be repeated if a different learning algorithm is to be used. Moreover, they can be computationally unfeasible for large data sets, where the iterative application of an induction algorithm may be excessively time-consuming.

7.7.2.2 Filters.

Feature selection can also be performed using a *filter* approach [71], which attempts to remove irrelevant features from the feature set prior to the application of the learning algorithm. Initially, the data is analysed to identify those dimensions that are most relevant for describing its structure. The chosen feature subset is subsequently used to train the learning algorithm. Feedback regarding an algorithm's performance is not required during the selection process, though it may be useful when attempting to gauge the effectiveness of the filter.

Criteria that have been used to select relevant features include [80]:

- *Distance measures.* Metrics such as Euclidean or Manhattan distance have been employed to measure find a feature subset that maximises inter-class distance while minimising intra-class distances. For instance, the RELIEF [74] algorithm estimates feature relevance based on how well their values discriminate between instances in the same and different classes.

- *Information-theoretic and dependency measures.* These measures attempt to find a feature subset that maximises the correlation between the features and each class (i.e. choose features that are most predictive of a class), while minimising the correlation between pairs of features (i.e. eliminate redundant features). Information-theoretic methods are generally used to determine the correlations between features and classes, and between pairs of features. Popular measures include conditional entropy [62] and cross-entropy [75].

- *Consistency measures.* In this context, an inconsistency occurs when two instances agree on all values for a given subset of features, but belong to different classes. Thus, feature selection can be viewed as the search for a minimal feature subset that provides an acceptable level of consistency. A well-known example of this approach is the FOCUS algorithm [5], which performs an exhaustive breadth-first search to locate a consistent subset of features.

Since filter methods do not involve the repeated application of a predictor, they are less computationally expensive than wrappers. In addition, the evaluation of intrinsic properties of the data, independent of any classifier, allows filters to produce more general solutions that may be suitable for use with a broad range of classifiers. However, this generality precludes the ability to consider the interactions between a feature subset and a given learning algorithm, suggesting that they are less effective than wrapper approaches. It has also been observed that filters are also often only viable for large datasets with a high number of dimensions [71].

7.7.3 Stopping Criteria

For wrapper methods, the search process is often terminated when the removal of features no longer results in improved classifier accuracy. Alternatively, thresholding can be employed to end the search after a given number of iterations have been completed, or when a pre-defined level of accuracy is attained.

7.7.4 An Alternative Approach to Feature Selection

For the cases when we cannot even assume that class-conditional pdfs are unimodal and the only available source of information is the training data, an alternative approach to supervised feature selection has been developed based on approximating the unknown class conditional distributions by finite mixtures of parametrized densities of a special type. Two methods currently exist [97, 90].

7.8 Unsupervised Feature Selection

The task of identifying a minimal set of relevant features to represent data has received significant attention in machine learning. Previously, the majority of this work has been focused on supervised learning algorithms, with little attention given to the problem of feature selection in the absence of class information. However, the presence of irrelevant or redundant features also has a significant impact on the performance of unsupervised procedures, such as cluster analysis. This is particularly relevant in domains such as gene expression analysis and document categorisation, where the importance of extracting useful information from high-dimensional data has recently lead to the proposal of feature subset selection techniques designed to improve the output of unsupervised learning algorithms.

Much of what has been said in the earlier section on supervised feature selection applies also for unsupervised feature selection. The main difference is the question of how to evaluate the quality of a feature subset since there is no obvious metric such as the generalisation error of a classifier to guide us. This issue is discussed in detail in section 7.8.2.

7.8.1 Motivation

For unsupervised learning tasks, the potential benefits of reducing data dimensionality include:

7.8.1.1 Improved performance

Clustering algorithms are susceptible to the “curse of dimensionality" [9], as increased information in the form of additional features does not necessarily lead to an improved partition of the data. Rather, algorithm performance often deteriorates rapidly in the presence of many irrelevant features. In addition, the inherent sparsity of points in high dimensional data can further impair an algorithm’s ability to correctly uncover the natural groupings of instances in data. Therefore, it is highly desirable to reduce data dimensionality prior to the application of clustering algorithms.

7.8.1.2 Increased efficiency

Dimensionality reduction can lead to a considerable improvement in the efficiency of unsupervised learning tasks, as decreasing the number of dimensions in the data reduces the size of the hypothesis space. Pre-processing techniques such as feature selection can improve the scalability of many computationally complex clustering algorithms, reducing processing time and storage overhead.

7.8.1.3 Improved interpretability

High dimensionality in data often significantly impairs our ability to perform knowledge discovery tasks successfully. Clusters formed in these spaces can be difficult to interpret, with standard visualisation techniques performing poorly in these situations. By reducing the number of features in the data, it is possible to improve the comprehensibility of the output of a clustering procedure without adversely affecting the algorithm's performance.

For knowledge discovery purposes, we may also be interested in discovering hidden relationships between features, and in identifying those features that are effective in discriminating between groups of instances. The insight gained from determining relevant modeling variables is of particular relevance in areas such as gene expression analysis where, for instance, correlation between features could suggest the co-regulation of genes.

7.8.2 Feature Subset Evaluation

In supervised learning, the effectiveness of a feature selection algorithm can be assessed by applying the classifier built in the chosen feature subspace to a separate training set and comparing the results with external class information. Where no separate data is available, cross-validation may be employed to provide a similar evaluation. In both cases, the comparison of different selection approaches is based on predictive accuracy. However, the absence of predefined class labels makes it difficult to compare the effectiveness of feature selection algorithms in unsupervised learning. The situation is further exacerbated by the strong correlation between a clustering and the dimensions upon which it was generated.

In this section we discuss two broad approaches for the experimental evaluation of unsupervised feature selection results:

- *External measures.* Use a priori information, which provides additional knowledge concerning the data, to either directly evaluate the selected feature subset or to measure the prediction accuracy of the partition generated on those dimensions.
- *Internal measures.* Determine the quality of a partition without referring to any external knowledge. This approach is motivated by the assumption that the optimal feature subset should give the "best" partition as defined by one or more statistical indices. These indices can be further divided into parametric and non-parametric measures.

7.8.2.1 External Measures

In cases where the annotated class labels or generative models for a dataset are available, external evaluation criteria may provide an effective means of assessing the performance of a feature

selection algorithm. These measures determine how closely a clustering procedure's output corresponds to the known natural groupings in the data according to knowledge unavailable to the clustering algorithm itself.

7.8.2.2 Mis-assignment rate.

A simple approximation of accuracy for unsupervised learning that employs external class information was described by Topchy et al. [120]. By finding the optimal correspondence between a dataset's annotated class labels and the clusters in a given partition, a performance measure may be derived that reflects the proportion of instances that were correctly assigned. Note that this measure is only applicable when the number of clusters k is the same as the number of natural classes.

7.8.2.3 Jaccard index.

In the *Jaccard index* [69], which has been commonly applied to assess the similarity between different partitions of the same dataset, the level of agreement between a set of class labels C and a clustering result K is determined by the number of pairs of points assigned to the same cluster in both partitions:

$$J(C,K) = \frac{a}{a+b+c}$$

where a denotes the number of pairs of points with the same label in C and assigned to the same cluster in K, b denotes the number of pairs with the same label, but in different clusters and c denotes the number of pairs in the same cluster, but with different class labels. The index produces a result in the range $[0,1]$, where a value of 1.0 indicates that C and K are identical.

7.8.2.4 Conditional entropy.

Boley et al. [13] suggested an entropy-based measure for assessing the agreement between a partition and external class information. For a dataset of n instances, with m associated natural classes, the level of agreement is given by the weighted sum of the entropies for each of the k clusters:

$$CE(C,K) = \frac{1}{n}\sum_{j=1}^{k} n_j \sum_{i=1}^{m} -p_{ij}\log p_{ij}$$

where p_{ij} denotes the probability that an instance in cluster C_j belongs to class K_i. A minimal value for this index is desirable, with a value of 0.0 indicating that each cluster contains instances from a single class only.

7.8.2.5 Normalised mutual information.

Strehl and Ghosh [113] observed that external measures such as those described above are biased with respect to the number of clusters k, as the probability of each cluster containing only instances from a single class increases as k increases. As an alternative, the authors proposed

a criterion based on *normalised mutual information*, defining the accuracy of a partition by the expression

$$\phi^{(NMI)}(C,K) = \frac{\sum_{j=1}^{k}\sum_{i=1}^{m} p_{ij}\log\left(\frac{p_{ij}}{p_i p_j}\right)}{\sqrt{E(C)E(K)}}$$

where $E(C)$ denotes the entropy of the distribution of class labels and $E(K)$ denotes the entropy of the distribution of cluster labels. This criterion should be maximised, with a value of 1.0 indicating an exact correspondence between the assignment of instances in a given partition and the dataset's natural classes.

7.8.3 Parametric Internal Measures

As it is often necessary to analyse the performance of clustering procedures on data for which no class information exists, a variety of "relative" cluster validation indices have been suggested to measure the internal quality of a partition. These are often employed to identify the optimal parameters for a specific clustering algorithm on a given data set by comparing the relative index scores of different partitions on the data. It is important to note that many internal validation indices make assumptions about the parametric form of the underlying distribution of the natural classes in the data, favouring Gaussian-shaped clusters with high inter-cluster separability. A brief summary of these indices is provided here. For more detailed coverage of this topic, see Halkidi et al. [61] and Bolshakova and Azuaje [14].

7.8.3.1 Sum-of-squares error.

A simple validation index, the *sum-of-squares error* for a partition of k clusters measures the total amount of inter-cluster scatter, as defined by the expression

$$SSE = \sum_{i=1}^{k}\sum_{x\in C_i} \|x - M_i\|^2$$

where m_i denotes the mean of the points in the i-th cluster C_i, which is the best representative of those points. The optimal partition is defined to be that which minimises this value, indicating a collection of cohesive clusters with a minimal amount of variance between the instances assigned to the same cluster.

7.8.3.2 Trace criterion.

The *trace* criterion, originally used in discriminant analysis, can be used to measure the compactness and separability of clusters in a given partition.

Let m_i be the mean vector for the i-th cluster containing n_i points and let m be the total mean vector for the partition. The *within-cluster scatter matrix* for a partition of k clusters, measuring how far the points in each cluster are from their respective cluster means, can be expressed as:

$$S_W = \sum_{j=1}^{k}\sum_{x\in C_i} (x - m_i)(x - m_i)^T$$

The *between-cluster scatter matrix* for the k clusters, which measures how far the cluster means are from the total mean, is given by the expression:

$$S_B = \sum_{j=1}^{k} (x - m_i)(x - m_i)^T$$

The *trace* criterion $tr(S_W^{-1} S_B)$ measures the ratio of between-cluster to within-cluster scatter, where the trace of a matrix is the sum of its diagonal elements.

7.8.3.3 Dunn's index.

Dunn [46] suggested a validation measure for identifying "compact and well separated clusters". Dunn's index for a partition of k clusters is defined as:

$$D = \min_{1 \le i \le k} \left\{ \min_{1 \le j \le k, i \ne j} \left\{ \frac{\delta(C_i, C_j)}{\max_{1 \le l \le k} \{\Delta(C_l)\}} \right\} \right\}$$

where $\delta(C_i, C_j)$ is the inter-cluster distance between the i-th and j-th clusters, and $\Delta(C_i)$ is the average intra-cluster distance (dispersion) of the i-th cluster. The goal of highly compact clusters with low inter-cluster similarity suggests that better partitions are indicated by larger values for this index.

The main deficiencies of this index are its computational complexity as k and n increase, and its sensitivity to outliers [61].

7.8.3.4 Davies-Bouldin index (DB).

The DB-Index [36] is a measure of cluster quality that comprises the summation over all k clusters of the ratio of intra-cluster scatter to inter-cluster separability. For the i-th cluster, with centre Z_i, we calculate the intra-cluster scatter as the average Euclidean distance between all points in C_i:

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|X - Z_i\|$$

The inter-cluster separability between the i-th and j-th clusters is calculated by the expression:

$$d_{ij} = \|Z_i - Z_j\|$$

Consequently, the DB-Index is defined as the ratio:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \ne j} \left\{ \frac{S_i + S_j}{d_{ij}} \right\}$$

This value will decrease as clusters become more compact and more distinctly separated. Therefore, when using this index to assess cluster validity, minimal values are desirable.

7.8.3.5 Silhouette method.

In the silhouette method [101], a *silhouette width* is calculated for each point, measuring the strength with which it is perceived to belong to its assigned cluster. Let a_i denote the average dissimilarity of the i-th point to all other points in the same cluster. Let b_i denote the average dissimilarity of a point to all points in the nearest competing cluster. The silhouette width for x_i is given by:

$$S(i) = \frac{b_i - a_i}{\max\{a(i), b(i)\}}$$

A value close to 1.0 indicates that the point is likely to have been assigned to the correct cluster. A silhouette width of 0.0 suggests that the point could also have been assigned to the nearest cluster, while a value of -1.0 suggests that either the point was incorrectly assigned or is an outlier. By averaging across all clusters, an evaluation for an entire partition can be obtained.

7.8.4 Non-Parametric Internal Measures

The internal evaluation criteria described previously are parametric and implicitly favour bell-shaped distributions, making them inappropriate for some evaluation tasks. To address this issue, Pauwels and Frederix [93] proposed alternative non-parametric measures that do not require any assumptions to be made about the underlying distribution of partition, allowing them to handle arbitrarily shaped clusters.

7.8.4.1 Isolation.

This index is based on the assertion that neighbouring instances in feature space often occur in the same natural cluster. The *isolation* of each cluster is measured using the k-nearest neighbour norm, where the norm $\nu_k(x_i)$ of each instance x_i is defined as the proportion of its k nearest neighbours that have been assigned to the same cluster as x_i. By summing over all n instances in the data, the homogeneity of a partition may be computed as follows

$$I_k = \frac{1}{n}\sum_{i=1}^{n} \nu_k(x_i)$$

The authors acknowledge that, while this index rewards partitions that assign regions of well-connected instances to the same cluster, it fails to penalise partitions where well-separated clusters are merged, since only a limited locality is considered for each point.

7.8.4.2 Connectivity

A second evaluation criterion was proposed by Pauwels and Frederix [93], based on the assumption that, for any pair of instances assigned to the same cluster in a partition, the density of the data along the path connecting the pair should be consistently high. Additionally, the index penalises clusters containing two or more well-separated regions.

In the implementation described by Frossyniotis et al. [53], r pairs of instances are randomly selected such that both instances in each pair are assigned to the same cluster, where r is a user-defined parameter. A total evaluation for the partition is obtained using the expression

$$C_k = \frac{1}{r}\sum_{i=1}^{r} d(\mu_i)$$

where μ_i is the midpoint of the line joining the i-th pair of points, and $d(\mu_i)$ is the local density at that midpoint.

7.8.4.3 Combined isolation/connectivity.

Pauwels and Frederix [93] suggest that, by combining the output of both non-parametric indices, a superior measure may be obtained that compensates for the deficiences of the *isolation* criterion. This is achieved by computing the Z-scores for each of the indices, and summing the results to get an overall Z-score for the partition.

7.8.5 Models for Unsupervised Feature Selection

In Section 2 we discussed techniques used in supervised learning to deal with high dimensional data. Similar problems occur when applying learning algorithms in situations where class information is unavailable. This has motivated the development of unsupervised feature selection procedures to remove redundant and irrelevant features. In this section, we discuss the key issues that must be addressed by unsupervised feature selection algorithms, and we examine two fundamental models for feature subset evaluation.

In developing techniques for unsupervised feature selection, some key considerations are:

- *Lack of class information:* Supervised feature selection methods often make use of classifier accuracy to guide the search for a suitable feature subset. Conversly, there is no definitive measure of accuracy for unsupervised learning algorithms. As illustrated in Section 7.8.2, numerous approaches have been proposed to quantify clustering performance based on factors such as cluster compactness or inter-cluster separability. However, many of these measures are only appropriate for use with a particular clustering model or domain.

- *Number of clusters:* Typically the optimal number of clusters k for a data set is unknown. In addition, the number of clusters and the dimensions used to produce a partition are inter-related. Therefore, the use of a fixed number of clusters when evaluating feature subsets may not provide useful results, as data in a given feature subspace may only cluster well for certain values of k.

- *Algorithm instability:* Classifier instability has been shown to be an issue for wrappers in supervised feature selection, where small changes in the composition of the data set can significantly affect the resulting feature set. Ensemble techniques such as those proposed by Breiman [16] have been used to produce more robust classifiers. Dunne et al. [47] suggested the *wrapper-2* algorithm, where the selection process is stabilised by effectively

"wrapping the wrapper". By aggregating the output of multiple sequential search procedures, the authors observed a reduction in the diversity of subsets produced by different trials of the same selection process.

In the context of cluster analysis, the stability of an algorithm refers its tendency to generate similar partitions over multiple trials performed on the same data. Many commonly used clustering algorithms are unstable in that they can potentially partition the same data set in many different ways. Small changes in the initial parameters specified for an algorithm may lead to considerably different results. For instance, the standard *k*-means algorithm [82] will often converge on different local optima depending on the choice of initial centroids. Consequently, unsupervised feature selection procedures that rely on the output of a clustering algorithm may be unstable.

- *Local feature selection:* The majority of work in unsupervised dimensionality reduction has focused on *global feature selection* methods, which attempt to find a single set of relevant features for an entire dataset. However, it has been observed that, in high dimensional spaces, different groups of instances may cluster more successfully using different feature subsets [3]. Often, the sparsity of the data renders all but a few features irrelevant to each cluster. As a result, it may be difficult to reduce the size of a global feature set significantly without incurring the loss of information.

  To address this problem, several authors [3, 4] have proposed *local feature selection* methods, where each cluster can potentially be assigned a different subset of relevant features. For further discussion of this topic, see Section 7.8.6.4.

7.8.5.1 Wrapper Model

Wrapper schemes for unsupervised feature selection involve the search for a representative subset of the original dimensions, by using the output of a clustering algorithm to direct the search process. In general, this involves finding a feature subset that maximises algorithm performance as quantified by some predefined criterion. When a desirable feature subset has been selected, the same clustering algorithm is applied based on the chosen dimensions to generate the final partition.

While wrappers have been used extensively in supervised learning, several difficulties exist when implementing similar techniques for unsupervised learning tasks. Most problematic is the lack of any universally accepted method for comparing the quality of partitions generated in different feature subspaces. Other concerns include the instability of the clustering algorithm being wrapped, and the computational cost of repeatedly applying complex clustering procedures.

7.8.5.2 Filter Model

Rather than requiring the repeated application of a clustering algorithm, *filter* schemes for unsupervised feature selection attempt to choose the best feature subset based on intrinsic properties of the data. As with supervised filters, the data is first analysed to identify relevant features,

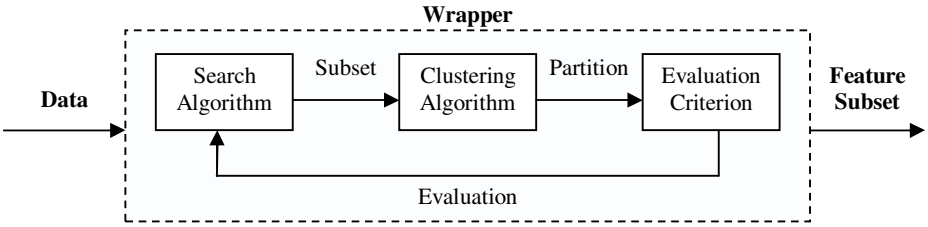


Figure 7.5: Wrapper Model for Unsupervised Feature Selection

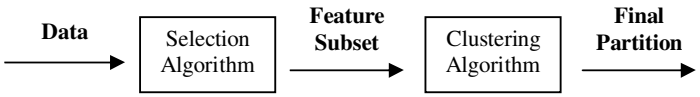


Figure 7.6: Filter Model for Unsupervised Feature Selection

typically using information-theoretic measures to examine characteristics such as feature correlation. Following the selection process, a given clustering algorithm is run on the subset of most relevant features to generate the final partition.

Since filters are independent of any given clustering algorithm, it is possible to avoid the difficulties posed by the lack of a definitive quality criterion and the need to choose algorithm parameters. These methods are also less computationally expensive, as there is for the repeated application of a clustering algorithm.

7.8.6 Unsupervised Wrapper Methods

7.8.6.1 Global Feature Selection

The simplest approach analogous to supervised wrappers involves selecting a single feature subset for an entire partition by wrapping a commonly used clustering algorithm. Dy and Brodley [48] proposed the FSSEM method, which wraps the search process around the expectation-maximization (EM) clustering algorithm [39]. A forward sequential search is performed to generate candidate subsets, which are evaluated using one of two proposed selection criteria. The first is the trace criterion, discussed in Section 7.8.3, which the wrapper attempts to maximise. The second measure is the *maximum likelihood* (ML) criterion, employed in the EM clustering itself . In this case the search involves choosing the subset that is most likely to fit the data to the chosen model and parameters.

An interactive visualisation framework for FSSEM was described by Dy and Brodley [49]. User input is employed to guide the choice of initial feature set, to instigate back-tracking during the search process and to identify an appropriate evaluation criterion. This effectively allows domain knowledge to be manually incorporated into the feature selection process.

7.8.6.2 Feature Selection in Conceptual Clustering

Conceptual clustering algorithms process instances to produce a generalisation of the data in the form of a set of concepts, which are often organised in a hierarchical manner. Generally concept learners rely on a fixed set of features for describing instances, with the choice of dimensions made prior to the construction of the concept hierarchy. This constraint renders the system incapable of adjusting to subsequent changes in the feature set used to describe objects, without requiring the concept model to be completely reconstructed.

To address this issue, Devaney and Ram [40] proposed AICC, an *attribute-incrementation* technique that extends the COBWEB framework [52] to allow the reuse of existing concepts when changes are made to the underlying feature set. The first phase of the algorithm involves the modification of the feature set used to describe the concepts in the existing hierarchy. The algorithm visits each node in the hierarchy, updating its description to reflect the addition or removal of features. Once the feature set of each node has been updated, the tree is likely to be unbalanced and will require reorganisation. In the second phase, each node is revisited, and the potential increase in *category utility* (CU) gained by splitting the node calculated. If the split is advantageous, the operation is applied and the algorithm examines the newly created child nodes in the same way. When all beneficial splits have occurred at the partition, a similar approach is applied to examine the possibility of merging pairs of nodes occurring on the same level in the tree. The second phase is completed when all partitions have been visited, resulting in an amended concept hierarchy that makes use of the updated feature set.

Devaney and Ram [41] employed AICC for the purposes of unsupervised feature selection, making use of its ability to incrementally adjust the current feature set with requiring the concept hierarchy to be reconstructed. The algorithm explores the feature space using a forward or backward sequential search algorithm. Every time a feature is added or removed, the hierarchy is updated and reorganised as described above. The utility score of the root partition is then calculated to evaluate the suitability of the current feature set. The process terminates when there is no longer an improvement in utility.

7.8.6.3 Evolutionary Multi-Objective Optimisation

In feature selection, it is generally desirable to reduce the dimensionality of data without adversely affecting the performance of the learning algorithm. There will typically be a conflict between these goals, with no feasible solution that optimises both of them. In addition, many of the performance indices for unsupervised learning described in Section 7.8.2 can produce contradictory results. However, the wrapper methods described previously select a feature subset based on a single criterion, without considering other relevant factors.

Alternatively, feature subset selection may be formulated as a multi-objective optimisation problem, where the goal is to obtain a set of *Pareto optimal* solutions that represents the best trade-off between objectives. These solutions are described as *non-dominated*, since there exist no other solutions that are better on all criteria. The curve formed by joining the set of *non-dominated* solutions is referred to as the *Pareto-optimal front*. To identify these solutions, Goldbreg [59] suggested the use of evolutionary algorithms due to their effectiveness at performing a rapid global search of large problem spaces. Evolutionary optimisation algorithms seek to iteratively improve their approximation of the optimal front, while simultaneously maintain-

ing a diverse population of solutions that prioritise objectives differently. For a comprehensive discussion of evolutionary multi-objective optimisation algorithms, see [38].

Morita et al. [89] applied the *Non-dominated Sorting Algorithm* (NSGA) [111] to the problem of unsupervised feature selection. The process begins with the random generation of a population of candidate feature subsets, each of which is represented by a bit string indicating the presence or absence of features. A partition is formed in each feature subspace by applying the standard k-means algorithm, which is subsequently evaluated based on the number of dimensions in the subspace and the quality of the clustering, measured using the Davies-Bouldin index. The results are analysed to identify the Pareto front of all non-dominated solutions. These individuals are assigned a high “dummy" fitness value and are temporarily removed. The process is repeated until all solutions have been assigned a fitness score, essentially ranking the population members based on the front in which they occur. On the basis of these fitness values, the next generation of solutions is produced using the crossover, mutation and roulette wheel selection genetic operators. The cycle is repeated for N generations, at which time the subset with the highest fitness score is deemed to be the final solution.

Kim et al. [73] performed dimensionality reduction by applying ELSA [86], an evolutionary optimisation algorithm that employs *local selection* to maximise population diversity. In this context, *local selection* refers to an optimisation approach that seeks to minimise the interaction between population members by allocating a restricted area of the search space to each individual. This has the effect of decreasing the likelihood that the search will converge to a single solution prematurely, while simultaneously affording improved efficiency by allowing different areas of the search space to be examined in parallel. Since the choice of feature subset is related to the number of clusters k, the authors do not suggest the use of a fixed value for k. Rather, a preferred value is determined as part of the evolutionary process. This is achieved by assigning a value of k to each population member, which is incorporated into the binary representation of the individual in the form of a block of trailing bits.

The selection process begins with the generation of population members, each of which is assigned a random solution and an initial reservoir of “energy". The specific level of energy assigned to an individual is determined by performing a k-means clustering in the associated feature subspace, and rating the fitness of the resulting partition using four quality criteria based on cluster compactness, inter-cluster separability, the number of clusters and the number of dimensions in the subspace. At each iteration of the algorithm, the candidate solution is mutated and the new solution’s energy level is adjusted based upon its fitness and the fitness of other individuals in the same locality. As a result, individuals occurring in dense regions in the objective space face greater competition for energy, thus ensuring that a diverse range of solutions is maintained. At the end of each iteration, those individuals whose energy level is above a fixed threshold θ are used as the basis for the next generation of solutions. The algorithm halts after N generations, at which time the best solution is used to determine the final feature subset and value for k.

7.8.6.4 Local Feature Selection

The wrapper techniques described previously attempt to choose a global feature subset common to all clusters in a given partition. However, several authors have proposed *local feature selection* techniques, based on the concept of *projected clustering* as introduced by Agrawal

et al. [4]. This clustering model allows each cluster to be potentially formed on a different subset of features. This approach to clustering can be viewed as an optimisation problem, involving the search for a set of clusters $\{C_1, C_2, \ldots, C_k\}$ with corresponding feature subspaces $\{S_1, S_2, \ldots, S_k\}$, that maximises intra-cluster similarity for each cluster when its points are projected to the associated feature subspace.

The authors proposed CLIQUE, a projected clustering algorithm, which attempts to address the problem of high dimensionality by employing a grid-based scheme to locate clusters formed from dense subspaces in the data. A three step approach is taken to choosing clusters and their associated subspaces. Initially, each dimension is divided into a number of equal intervals, so that the data space forms a grid. A set of dense “units" (grid cells) are found by examining the density of points in 1-dimensional space and proceeding level-by-level until all n dimensions have been examined or no more dense subspaces are found. The second phase involves the identification of clusters by performing a depth-first search to locate maximal sets of connected units. Finally, a minimum-length description is generated for each cluster by covering it with a set of maximal rectangles and greedily removing redundant (i.e. overlapping) rectangles. This final step ensures that each cluster is described by only those features that are relevant to the instances assigned to it.

The ProClus algorithm [3] is also designed to find clusters in small subspaces of high-dimensional data, but uses a wrapper-based extension to the k-medoids clustering algorithm. The procedure begins with a greedy search to find the “best" initial set M of $B \times k$ medoids, which is achieved by choosing each cluster centre so that it is well separated from those that have been previously selected. In the second phase of the algorithm, a random subset of medoids M_C is chosen. Dimensions are assigned to each of these medoids by examining the instances in its “locality" and choosing those features on which the points are most similar to that medoid. Clusters are then formed by assigning each point to the closest cluster centre. Intra-cluster scatter is measured on the resulting partition using a variation of the Manhattan distance metric. The subset M_C is iteratively improved by performing a hill-climbing search, where clusters are repeatedly formed and evaluated. The final clusters are refined by computing the appropriate dimensions based on the points in each cluster rather than the medoidï¿½ locality.

7.8.7 Unsupervised Filter Methods

7.8.7.1 Subset Search Methods

Many filter methods for unsupervised feature selection make use of search procedures such as those applied in wrappers. However, rather than clustering the data in each candidate subspace, measures are employed to evaluate intrinsic properties of the data. In these situations, information-theoretic measures, such as entropy, are commonly used to determine the amount of information gained by using a given feature. Entropy measures the uncertainty of a random variable. When the variable’s probability distribution is uniform, its value is unpredictable and the resulting entropy is high. Conversely, when the variable is likely to take one of a small number of values, the outcome is quite predictable and the entropy is low. In the context of cluster analysis, data with an orderly configuration of distinct clusters has low entropy, while disordered data with no obvious clusters has high entropy. This concept can be useful for feature selection, where the effect of removing a feature can be determined by the relative change

in entropy.

The first formal proposal for unsupervised feature selection, described by Dash et al. [31], was based on the assumption that irrelevant features should have no influence on the formation of distinct clusters, and consequently should result in no change in entropy. Therefore, if the projection of the data from N to $N-1$ dimensions preserves the separation between clusters, the excluded feature can be deemed to be irrelevant. A backward sequential search was performed, where the change in entropy due to the removal of each feature is calculated using:

$$E = -\sum_{i=1}^{N}\sum_{j=1}^{N}\left[D_{ij}\log D_{ij} + (1-D_{ij})\log(1-D_{ij})\right]$$

where D_{ij} is refers to the normalised point-to-point distance between the points x_i and x_j. At each iteration, the least relevant feature, with the lowest entropy score, is removed. Typically the procedure terminates when the subset has been reduced to a certain size or there is no significant change in entropy.

Dash et al. [32] proposed an enhanced entropy-based filter, noting that distance-based entropy criteria have several drawbacks. Firstly, these criteria attain optimal values at the mean distance of 0.5, which represents the meeting point separating inter-cluster and intra-cluster distances. As a result, a low inter-cluster distance of 0.5 could be assigned the highest entropy value. Secondly, such measures are susceptible to yielding significantly higher values as a result of relatively small increases in intra-cluster distances. The authors suggest the introduction of a flexible meeting point μ to address the first issue and a coefficient β to reduce the bias against slight increases in internal cluster scatter. This leads to a new entropy measure, given by the expression:

$$E = \sum_{i=1}^{N}\sum_{j=1}^{N} E_{ij}$$

$$\text{where } E_{ij} = \begin{cases} \frac{exp(\beta * D_{ij}) - exp(0)}{exp(\beta*\mu) - exp(0)} & \text{if } 0 \le D_{ij} \le \mu; \\ \frac{exp(\beta*(1.0-D_{ij})) - exp(0)}{exp(\beta*(1.0-\mu)) - exp(0)} & \text{if } \mu \le D_{ij} \le 1.0; \end{cases} \text{ and } 0 \le E_{ij} \le 1.$$

The proposed feature selection algorithm proceeds with a forward sequential search generating candidate subsets. The solution that produces the lowest entropy score is selected as the best subset of dimensions for clustering.

7.8.7.2 Feature Ranking Methods

Feature ranking selection algorithms avoid computationally expensive search procedures by assigning weights to individual features indicating their relevance, and ordering the complete set of features according to their respective scores.

A representative example of this ordering approach is the feature selection method for hierarchical clustering algorithms proposed by Talavera [116]. The author suggests two relevance measures for determining feature weightings. The first is *salience*, which was originally proposed in the CLASSIT clustering system [58]. In this context, the salience of a feature A_i is defined as its contribution to category utility, calculated by the expression

$$sal(A_i) = \frac{\sum_k P(C_k)\sum_j [P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2]}{n}$$

The second proposal for feature weighting is based on the distance measure for decision tree learning [37], where the relevance of the feature A_i is defined as:

$$rel(A_i) = \frac{-\sum_k P(C_k)\log_2 P(C_k) - \sum_j P(A_i = V_{ij})\log_2 P(A_i = V_{ij})}{-\sum_k \sum_j P(A_i = V_{ij}|C_k)\log_2 P(A_i = V_{ij}|C_k)} - 1$$

This method was further developed by Talavera [117], who suggests the use of a measure for feature dependency originally proposed by Fisher [52]. This measure asserts that a feature is relevant if knowing its value increases our ability to correctly predict the value of other features. Formally, the dependency of a feature A_x on another feature A_y is defined as:

$$\begin{aligned} dep(A_x, A_y) &= \sum_{j_y}\left[P(A_y = V_{yj_y})\sum_{j_x} P(A_x = V_{xj_x}|A_y = V_{yj_y})^2\right] \\ &- \sum_{j_x} P(A_x = V_{xj_x})^2 \end{aligned}$$

7.8.7.3 Other Filter Methods

Some unsupervised filter methods can perform feature selection without requiring a complex search or the selection of a threshold for choosing from a ranked list of features. Most notably, several authors have suggested the use of a model-based clustering approach, where the unsupervised learning problem is formulated in terms of *Bayesian networks* [see 110]. In this model, it is shown that a relevant feature must be dependent on the random cluster variable C, which represents the cluster to which a point is assigned. Consequently, all relevant features must be pair-wise dependent. However, irrelevant features will be weakly dependent on the other features of the data. Feature selection approaches in both supervised and unsupervised learning have identified relevant features by ordering the complete feature set based on cumulative pair-wise dependency scores.

Sondberg-Madsen et al. [110] suggested a naive-Bayesian model, where a probabilistic relevance score is calculated using either of two common dependency measures. Both are based on a probability distribution derived from the data using the maximum likelihood (ML) criterion. Let $E(Y_i)$ be the entropy of the feature A_i and let $E(A_i|Aj)$ be the conditional entropy of Y_i given Y_j. The *mutual information* between the features A_i and A_j is given by:

$$XMI(A_i, A_j) = E(A_i) - E(A_i|Aj)$$

Let $PA(A_i)$ be the probability of correctly predicting the value of A_i based on the most probable value for A_i. Let $A(A_i|A_j)$ be the probability of correctly predicting the value of A_i based on the most probable value for A_i, when the value of the feature A_j is known. The *mutual prediction* between a pair of features is given by:

$$MP(A_i, A_j) = 1 - \frac{1}{2}\left(\frac{PA(A_i)}{PA(A_i|A_j)} + \frac{PA(A_j)}{PA(A_j|A_i)}\right)$$

$$\text{where } PA(A_i) = \max_{y_i} p(y_i),\ PA(A_i|A_j) = \sum_{y_i} p(y_j)\max_{y_i} p(y_i|y_j)$$

By calculating average pair-wise dependency using one of these measures, a relevance score for each feature can be derived. The authors suggest performing a significance test under the null hypothesis on each score to determine whether a feature is relevant or superfluous.

Model selection for clustering was also employed for feature selection by Vaithyanathan and Dom [126], who proposed an objective function based on a Bayesian estimation scheme to evaluate the size and quality of candidate subsets. This technique was applied experimentally to the problem of identifying topical terms in document clustering.

A feature-correlation based method for removing redudant features without any search or ranking procedure was suggested by Mitra et al. [88]. The original feature set is partitioned into groups using a *k*-NN approach, where all features in a given group are highly similar. The authors proposed a new similarity measure, the *maximal information compression index*, to measure the degree of linear dependence between two features. The index corresponds to the smallest eigenvalue of the covariance matrix of random variables x and y, representing two features in the data. Formally this is expressed as:

$$2\lambda_2(x,y) = var(x) + var(y) - \sqrt{(var(x)+var(y))^2 - 4var(x)var(y)(1-\rho(x,y)^2)}$$

where lower values of λ_2 indicate increased correlation. A single representative feature is chosen for each group and its redundant neighbours are discarded. The active subset of features is then iteratively refined by altering the value of k and removing features until an acceptable level similarity between all remaining features is attained. These retained features are chosen as the final subset for clustering.

7.8.8 Hybrid Methods

Rather than choosing one of the models described in Section 7.8.5, some authors have attempted to handle the problem of dimensionality in unsupervised learning by combining aspects of both wrapper and filter schemes.

Dash and Liu [33] suggested a two-phase hybrid approach, referred to as the RANK algorithm. Initially, the complete set of features is ordered by assigning a weight to each feature based on a distance-based entropy measure, as used in various filter methods described previously. Specifically, the weights are calculated by removing each feature in turn and measuring the relative change in entropy. However, rather than using thresholding to select a subset of features from the top of the ranked list, a wrapper method is subsequently applied to examine the effect of the remaining features on clustering performance. Features are sequentially added to the current subset in the order that they were ranked, the *k*-means algorithm is applied to the subset and the resulting partition is evaluated using an arbitrary performance index.

Sondberg-Madsen [110] observed that dependency-based filters behave conservatively in their removal of irrelevant features. The authors suggested a two-phase approach to feature selection, where a wrapper procedure is used to detect irrelevant features missed by the application of a filter method. Initially, a search-based filter is run to identify a large set of potentially relevant features. Subsequently, a wrapper with BSS is applied only to those features chosen in the first phase. Redundant features are removed while there is no decrease in clustering performance.

7.8.9 Conclusions on Unspervised Feature Seleciton

While many studies have been conducted on the use of feature selection in supervised learning, the presence of irrelevant or redundant features also has a significant impact on the performance of unsupervised procedures, such as cluster analysis. However, feature subset selection in the absence of class information is essentially an ill-posed problem. In this paper, we discussed the key difficulties faced when performing this task, and provided a review of the general models and practical approaches for selection that have been proposed in the recent literature.

It has been observed that wrapper schemes have considerable potential to reduce dimensionality without adversely affecting clustering performance. However, inherent instability of many popular clustering algorithms makes the validity of comparing subsets based on the partitions generated in the respective subspaces questionable. Recent work in the area of ensemble clustering could potentially address this issue by improving the robustness of clustering generated when evaluating subsets of features. Alternatively, selection methods could examine the relationship between base-level features and the output of an ensemble procedure. It must be noted that, in both cases the selection of suitable ensemble parameters for a given dataset is non-trivial. Furthermore, the computational complexity of ensemble procedures may render such approaches infeasible for large datasets.

Evolutionary multi-objective optimisation algorithms have been applied effectively in wrapper feature selection schemes to identify solutions that balance conflicting goals such as minimising feature subset size and maximising partition quality. However, the repeated application of a clustering algorithm when evaluating possible solutions also makes this approach computationally expensive. We suggest that filter-based schemes employing these optimisation algorithms could provide a comparable level of success with improved efficiency.

An outstanding issue remains the difficulty in choosing an appropriate evaluation measure to compare unsupervised feature selection algorithms. The use of synthetic data has been prevalent in the literature, as relevant features can be identified from the data's generative model. However, for practical purposes it is desirable to examine the performance of a selection technique on unlabelled data in a domain of interest. In these situations, the use of non-parametric internal measures could prove useful. As with the issue of cluster quality, no definitive techniques has yet been suggested for validating the output of an unsupervised feature selection algorithm.

7.9 Conclusions

In this section we have presented an extensive treatment of Dimension Reduction organised into Feature Transformation approaches and Feature Subset Selection Approaches. In situations where interpretability is not an issue and irrelevant and correlated features are assumed to exit, the Dimension Reduction techniques presented here can be very effective. If interpretability is important, in order to gain some insight into the domain of study for instance, then the Feature Subset Selection techniques described in the latter part of this chapter should be considered. Finally, the benefits of transparency that come from working in the original feature space in feature selection should not be underestimated.

Bibliography

[1]

[2] M. Mirmehdi A. Monadjemi, B. T. Thomas. Experiments on high resolution images towards outdoor scene classification. In Horst Wildenauer and Walter Kropatsch, editors, *Proceedings of the Seventh Computer Vision Winter Workshop*, pages 325–334. Vienna University of Technology, 2002.

[3] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 61–72, 1999.

[4] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of ACM-SIGMOD Int. Conf. Management of Data*, pages 94–105, Seattle, Washington, June 1998.

[5] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of the 9th National Conference on Artificial Intelligence*, volume 2, pages 547–552, Anaheim, CA, 1991. AAAI Press.

[6] T. Chan B. Sandberg and L. Vese. A level-set and gabor based active contour algorithm for segmenting textured images. Technical Report 02-39, UCLA Computational and Applied Mathematics, 2002.

[7] M.J. Bastiaans. A sampling theorem for the complex spectrogram and Gabor's expansion of a signal in Gaussian elementary signals. *Optical Engineering*, 20(4):594–598, 1981.

[8] R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[9] Richard Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[10] J.R. Bergen and E.H. Adelson. Early vision and texture perception. *Nature*, 333(26):363–364, 1988.

[11] J. Bigun and J.M. du Buf. N-folded symmetries by complex moments in gabor space and their application to unsupervised texture segmentation. *IEEE Trans. Pattern Anal. Machine Intell*, 16(1):80–87, 1994.

[12] Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[13] Daniel Boley, Maria Gini, Robert Gross, Eui-Hong Han, George Karypis, Vipin Kumar, Bamshad Mobasher, Jerome Moore, and Kyle Hastings. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.

[14] N. Bolshakova and F. Azuaje. Cluster validation techniques for genome expression data. Technical Report TCD-CS-2002-33, Trinity College Dublin, September 2002.

[15] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):55–73, 1990.

[16] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[17] Dmitry Bryliuk and Valery Starovoitov. Application of recirculation neural network and principal component analysis for face recognition. In *The 2nd International Conference on Neural Networks and Artificial Intelligence, Minsk, Belarus,*, 2001.

[18] N.A. Sochen C. Sagiv and Y.Y. Zeevi. Gabor space geodesic active contours. In *Algebraic Frames for the Perception-Action Cycle*, Lecture Notes in Computer Science , *Vol. 1888*, 2000.

[19] N.A. Sochen C. Sagiv and Y.Y. Zeevi. Gabor feature space diffusion via the minimal weighted area method. In *Proceedings of Energy Minimization Methods in Computer Vision and Pattern Recognition*, Lecture Notes in Computer Science *Vol. 2134*, pages 621–635, 2001.

[20] N.A. Sochen C. Sagiv and Y.Y. Zeevi. Geodesic active contours applied to texture feature space. In *Proceedings of Scale-Space and Morphology in Computer Vision, Lecture Notes in Computer Science, Vol. 2106*, pages 344–352, 2001.

[21] N.A. Sochen C. Sagiv and Y.Y. Zeevi. Texture segmentation via a diffusion-segmentation scheme in the gabor feature space. In *Proceedings of Texture 2002, The 2nd international workshop on texture analysis and synthesis*, pages 123–128, 2002.

[22] N. W. Campbell and B. T. Thomas. Automatic selection of gabor filters for pixel classification. In *Sixth International Conference on Image Processing and its Applications*, pages 761–765. IEE, 1997.

[23] Rich Caruana and Dayne Freitag. Greedy attribute selection. In *International Conference on Machine Learning*, pages 28–36, 1994.

[24] T. Chan, B. Sandberg, and L. Vese. Active contours without edges for vector-valued images. *J. Visual Communication Image Representation*, 11(2):130–141, 2000.

[25] T. Chang and J. C.-C. Kuo. Texture analysis and classification with tree–structured wavelet transform. *IEEE Trans. IP*, 2(4):429–441, 1993.

[26] P.C. Chen and T. Pavlidis. Segmentation by texture using correlation. *IEEE Transactions on PAMI*, 5(1):64–69, 1983.

[27] R.W. Conners and C.A. Harlow. A theoretical comparison of texture algorithms. *IEEE Tran. on PAMI*, 2(3):204–222, 1980.

[28] B. R. Corner, R. M. Narayanan, and S. E. Reichenach. Application of principal component analysis to multisensor classification. *AIPR Workshop: Advances in Computer-Assisted Recognition*, 1999.

[29] G.R. Cross and A.K. Jain. Markov random field texture models. *IEEE Trans. on PAMI*, 5:25–39, 1983.

[30] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 15(3), 1997.

[31] M. Dash, H. Liu, and J. Yao. Dimensionality reduction for unsupervised data. In *Proceedings of 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '97)*, pages 532–539, Newport Beach, CA, November 1997.

[32] Manoranjan Dash, Kiseok Choi, Peter Scheuermann, and Huan Liu. Feature selection for clustering - a filter solution. In *IEEE International Conference on Data Mining (ICDM'02)*, page 115, Maebashi City, Japan, December 2002.

[33] Manoranjan Dash and Huan Liu. Feature selection for clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 110–121, 2000.

[34] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Reg. Conf. Series in Applied Math. SIAM, 1992.

[35] J.G. Daugman. Uncertainty relation for resolution in space, spatial frequency and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Am.*, 2(7):1160–1169, 1985.

[36] D. L. Davies and W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[37] R. López de Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning Journal*, 6(1):81–92, 1991.

[38] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[40] Mark Devaney and Ashwin Ram. Dynamically adjusting concepts to accommodate changing contexts. In *Proceedings of the ICML-96 Workshop on Learning in Context-Sensitive Domains*, pages 6–14, Bari, Italy, 1996.

[41] Mark Devaney and Ashwin Ram. Efficient feature selection in conceptual clustering. In *Proceedings of 14th International Conference on Machine Learning*, pages 92–97. Morgan Kaufmann, 1997.

[42] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, 1982.

[43] J.M.H. du Buf and P. Heitkämper. Texture features based on Gabor phase. *Signal Processing*, 23:227–244, 1991.

[44] R. Duffin and S. Schaeffer. A class of nonharmonic Fourier series. *Trans.Amer.Math.Soc.*, 72:341–366, 1952.

[45] D.F. Dunn and W.E. Higgins. Optimal Gabor filters for texture segmentation. *IEEE Transactions on Image Processing*, 4(7):947–964, 1995.

[46] J. C. Dunn. Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[47] K. Dunne, P. Cunningham, and F. Azuaje. Solutions to instability problems with sequential wrapper-based approaches to feature selection. Technical Report TCD-CS-2002-28, Trinity College Dublin, 2002.

[48] Jennifer G. Dy and Carla E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 247–254. Morgan Kaufmann, San Francisco, CA, 2000.

[49] Jennifer G. Dy and Carla E. Brodley. Visualization and interactive feature selection for unsupervised data. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 360–364, Boston, MA, August 2000.

[50] I. Elfadel and R. Picard. Gibbs random fields, cooccurrences, and texture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):24–37, 1994.

[51] Deniz Erdogmus, Yadunandana N. Rao, Kenneth E. Hild II, and Jose C. Principe. Simultaneous principal component extraction with application to adaptive blind multiuser detection. *EURASIP JASP*, 2002.

[52] Douglas Fisher. Knowledge acquisition via incremental conceptual clustering. *Journal of Machine Learning*, 2(2):139–172, September 1987.

[53] D. Frossyniotis, A. Likas, and A. Stafylopatis. A clustering method based on boosting. In *Pattern Recognition Letters*, 2004.

[54] K. Fukunaga. *Introduction to Statistical Pattern Recognition: 2nd edition*. Academic Press Inc., 1990.

[55] D. Gabor. Theory of communication. *The Journal of the Institution Of Electrical Engineers*, 93(3):429–457, 1946.

[56] Meirav Galun, Eitan Sharon, Ronen Basri, and Achi Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *iccv 03*, pages 716–723, Nice, France, 2003.

[57] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Machine Intell.*, 6(6):721–741, 1984.

[58] John H. Gennari, Pat Langley, and Doug Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–62, 1989.

[59] David E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[60] G. D. Guo, S. Z. Li, K. L. Chan, and H. Pan. Texture image segmentation using reduced gabor filter set and mean shift clustering. In *Fourth Asian Conference on Computer Vision, ACCV2000*, pages 198–203, 2000.

[61] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[62] Mark Hall and L. A. Smith. Feature subset selection: A correlation based filter approach. In *Proceedings of the Fourth International Conference on Neural Information Processing and Intelligent Information Systems*, pages 855–858. Dunedin, 1997.

[63] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[64] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

[65] Hung-Lung Huang. Application of principal component analysis to high-resolution infrared measurement compression and retrieval. *J. Appl. Meteor.*, 2001.

[66] D.H. Hubel and T.N. Wiesel. Functional architecture of macaque monkey visual cortex. *Proc. Royal Soc. B (London)*, 198:1–59, 1978.

[67] A. Jain J. Mao. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.

[68] E. Fourgeau J. Morlet, G. Arens and D. Giard. Wave propagation and sampling theory - part 2: sampling theory and complex waves. *Geophysics*, 47,2:222–236, 1982.

[69] P. Jaccard. The distribution of flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

[70] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Rcognition*, 24(12):1167–1186, 1991.

[71] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the 11th International Conference on Machine Learning*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.

[72] B. Julesz. Texton gradients: the texton theory revisited. *Biological Cybernetics*, 54:245–251, 1986.

[73] Y. S. Kim, W. N. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, Boston, MA, August 2000.

[74] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proc. of AAAI-92*, pages 129–134, San Jose, CA, 1992.

[75] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the International Conference on Machine Learning*, pages 284–292, 1996.

[76] M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChe Journal*, 37(2):233–243, 1991.

[77] A. Laine and J. Fan. Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1190, 1993.

[78] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance, New Orleans*, 1994.

[79] T. S. Lee. Image representation using 2d gabor wavelets. *PAMI*, 18(10):959–971, 1996.

[80] Huan Liu and L. Yu. Feature selection for data mining. Technical report, Arizona State University, 2002.

[81] T. Brox M. Rousson and R. Deriche. Active unsupervised texture segmentation on a diffusion based feature space. In *CVPR03*, pages II: 699–704, 2003.

[82] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[83] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of Optical Society of America A*, 5(7):923–932, 1990.

[84] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, 1996.

[85] S. Marcelja. Mathematical description of the responses of simple cortical cells. *J. Opt. Soc. Am.*, 70(11):1297–1300, 1980.

[86] Filippo Menczer and Richard K. Belew. Local selection. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, pages 703–712, Berlin, 1998. Springer.

[87] Y. Meyer. *Oscilating Patterns in image processing and nonlinear evolution equations*. University Lecture Series. American Mathematical Society, 2002.

[88] P. Mitra, C.A. Murthy, and S.K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):301–312, 2002.

[89] M. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, page 666, Edinburgh, Scotland, August 2003.

[90] J. Novovičová, P. Pudil, and J. Kittler. Divergence based feature selection for multimodal class densities. *IEEE Transactions on PAMI*, 18(2):218–223, 1996.

[91] Conglin Lu P. Thomas Fletchery and Sarang Joshi. Statistics of shape via principal component analysis on lie groups. In *CVPR*, 2003.

[92] N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 46(3):223–247, 2002.

[93] E. J. Pauwels and G. Frederix. Finding salient regions in images: nonparametric clustering for image segmentation and grouping. *Comput. Vis. Image Underst.*, 75(1-2):73–85, 1999.

[94] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

[95] M. Porat and Y.Y. Zeevi. The generalized Gabor scheme of image representation in biological and machine vision. *IEEE Trans. PAMI*, 10(4):452–468, 1988.

[96] M. Porat and Y.Y. Zeevi. Localized texture processing in vision: Analysis and synthesis in the Gaborian space. *IEEE Trans. on Biomedical Eng.*, BME-36/1:115–129, 1989.

[97] P. Pudil, J. Novovičová, N. Choakjarerwanit, and J. Kittler. Feature selection based on the approximation of class densities by finite mixtures of special type. *Pattern Recognition*, 28(9):1389–1397, 1995.

[98] P. Pudil, J. Novovičová, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, November 1994.

[99] R. Malladi R. Kimmel and N. Sochen. Images as embedded maps and minimal surfaces: Movies, color, texture, and volumetric medical images. *International Journal of Computer Vision*, 39(2):111–129, 2000.

[100] Soumya Raychaudhuri and Joshua M. Stuart andand Russ B. Altmany. Principal components analysis to summarize microarray experiments: Application to sporulation time series. *Pacific Symposium on Biocomputing, Hawai*, 2000.

[101] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[102] M. Rousson and R. Deriche. A variational framework for active and adaptative segmentation of vector valued images. In *Proc. IEEE Worshop on Motion and Video Computing*, 2002.

[103] A. Yuille S.C. Zhu. Region competition: Unifying snakes, region growing and bayes/mdl for multiband image segmentation. *IEEE Transactions on PAMI*, 18:884–900, 1996.

[104] Y.N. Wu S.C. Zhu and D.B. Mumford. Equivalence of julesz ensembles and frame models. *International Journal of Computer Vision*, 38(3):247–265, 2000.

[105] N. Petkov S.E. Grigorescu and P. Kruizinga. Comparison of texture features based on gabor filters. *IEEE Trans. on Image Processing*, 11(10):1160–1167, 2002.

[106] Heung-Yeung Shum, Katsushi Ikeuchi, and Raj Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):854–867, September 1995. note that this paper has been previously published in CVPR'94.

[107] P. Somol and P. Pudil. Oscillating search algorithms for feature selection. In *Proc. 15th International Conference on Pattern Recognition*, pages 406–409, Los Alamitos, 2000. IEEE Computer Society.

[108] P. Somol, P. Pudil, and J. Kittler. Fast branch & bound algorithms for optimal feature selection. *IEEE Transactions on PAMI*, 26(7):900–912, 2004.

[109] P. Somol, P. Pudil, J. Novovičová, and P. Paclík. Adaptive floating search methods in feature selection. *Pattern Recognition Letters*, 20(11/13):1157–1163, 1999.

[110] Nicolaj Sondberg-Madsen, Casper Thomsen, and Jose M. Pena. Unsupervised feature subset selection. In *Proceedings on the Workshop on Probabilistic Graphical Models for Classification (in ECML/PKDD-03)*, pages 71–82, 2003.

[111] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[112] Magdalena Stasiak, Krzysztof Siwek, Ryszard Sikora, Stefan F Filipowicz, and Jan Sikora. The combination of principal component analysis with neural network as a highly efficient 3d eit solution. In *3rd World Congress on Industrial Process Tomography, Banff, Canada*, 2003.

[113] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)*, 3:583–617, December 2002.

[114] Changwon Suh, Arun Rajagopalan, Xiang Li, and Krishna Rajan. The application of principal component analysis to material science data. *Data Science Journal*, 2002.

[115] J. Puzicha T. Hofmann and J.M. Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):803–818, 1998.

[116] Luis Talavera. Dynamic local feature selection in incremental clustering. In *Proceedings of the First Catalan Conference on Artificial Intelligence*, pages 282–290, Tarragona, Spain, 1998.

[117] Luis Talavera. Feature selection as a preprocessing step for hierarchical clustering. In I. Bratko and S. Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pages 389–397, Bled, Slovenia, 1999. Morgan Kaufmann.

[118] A. Teuner, O. Pichler, and B.J. Hosticka. Unsupervised texture segmentation of images using tuned matched Gabor filters. *IEEE Tran. on Image Processing*, 4(6):863–870, 1996.

[119] S. Theodoridis and K. Koutroumbas. *Pattern Recognition: 2nd edition*. Academic Press Inc., 2003.

[120] A. Topchy, A.K. Jain, and W. Punch. Combining multiple weak clusterings. In *Proc. Third IEEE International Conference on Data Mining (ICDM'03)*, November 2003.

[121] W.E. Higgins T.P. Weldon and D.F. Dunn. Efficient gabor filter design for texture segmentation. *Pattern Recognition*, 29(12):2005–2015, 1996.

[122] D. Mumford T.S. Lee and A. Yuille. Texture segmentation by minimizing vector-valued energy functionals: the coupled-membrane model. In *Lecture Notes in Computer Science, Computer Vision ECCV 92*, pages 165–173, Santa Margherita Ligure, Italy, 1992.

[123] M.A. Turk and A.P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 1(3):71–86, 1991.

[124] M. Unser. Local linear transfermations for texture measurements. *SP*, 11:61–79, 1986.

[125] H. Vafaie and K. De Jong. Robust feature selection algorithms. In *Proc. 5th Intl. Conf. on Tools with Artificial Intelligence*, pages 356–363, Rockville, MD, 1993.

[126] Shivakumar Vaithyanathan and Byron Dom. Model selection in unsupervised learning with applications to document clustering. In I. Brakto and S. Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pages 433–443, Bled, Slovenia, June 1999. Morgan Kaufmann.

[127] Namrata Vaswani. A linear classifier for gaussian class conditional distributions with unequal covariance matrices. In *ICPR*, 2002.

[128] Namrata Vaswani and Rama Chellappa. Classification probability analysis of principal null space analysis. *IEEE Transactions on Image Processing*, 2003.

[129] Namrata Vaswani and Rama Chellappa. Principal component null space analysis for image/video classification. *IEEE Transactions on Image Processing*, Submitted.

[130] L.A. Vese and S.J. Osher. Modeling textures with total variation minimization and oscilating patterens in image processing. Technical Report 02-19, UCLA Computational and Applied Mathematics, 2002.

[131] A.R. Webb. *Statistical Pattern Recognition: 2nd edition*. John Wiley and Sons Ltd., 2002.

[132] Jihoon Yang and Vasant Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems (Special Issue on Feature Transformation and Subset Selection)*, 13(2):44–49, 1998.

[133] X.Y. Zeng Y.W. Chen and H. Lu. Edge detection and texture segmentation based on independent component analysis. In *16 th International Conference on Pattern Recognition (ICPR'02), volume 3*, page 30351, Quebec City,QC, Canada, 2002.

[134] M. Zibulski and Y.Y. Zeevi. Analysis of multiwindow gabor-type schemes by frame methods. *Applied and Computational Harmonic Analysis*, 4:188–221, 1997.

Chapter 8

Conclusion

This review provides a comprehensive coverage of the most important Machine Learning techniques in use in the processing of Multimedia data. We have covered supervised learning techniques in detail in chapter 3. This is the longest section of the review reflecting the importance of this area. Chapter 4 covers unsupervised learning techniques with a particular focus on basic clustering techniques and on unsupervised Bayesian techniques. Reflecting the fact, that the distinction between supervised and unsupervised learning is not completely black and white, the next chapter (6) cover the emerging areas of semi-supervised learning, a machine learning methodology that lies somewhere between the supervised and unsupervised extremes. Chapter 6 covers active learning which is presented here as a variant of classical supervised learning which is *passive* in these terms. The final core chapter is chapter 4 on dimension reduction. Reflecting the importance of this issue in multimedia data processing, this is one of the longest sections of the report.

The contents of these core chapters reflect the expertise in the Muscle consortium. Appendix B presents a brief summary of the topics we propose to cover in the coming months.

Appendix A

Areas of Expertise within Muscle

- ENSEA: Sup, Un-sup, Semi-supervised
- SZTAKI: learning in motion tracking and background filtering, supervise/unsupervised learning in texture segmentation, adaptive intelligent image sensors
- TCD
  - Simon Wilson: Bayesian Learning
  - Pádraig Cunningham: Case-Based Learning, Neural Networks, Ensembles, Feature Selection
- Bilkent - Enis Cetin: Feature Extraction techniques
- INRIA-Ariana: Bayesian learning of parameters and models for image processing.
- LIC2M: indexing and retrieval of multilinual and multimedia data
- Technion: Active Learning
- IBaI: decision tree induction, CBR, conceptual clustering, feature subset selection

Appendix B

Next Steps within Muscle

- *Cluster Validity Metrics* Evaluation of existing cluster validity metrics show that available metrics still do not capture the notion of cluster quality well. This area will be a topic within Muscle over the coming year.
- *Adaboost*Improving the robustness of Adaboost for use in person detection systems. This work will involve a comparative evaluation of the performance of this improved Adaboost against other classifiers.
- *Unsupervised Bayesian Techniques* Bayesian CBIR and parallel processing for CBIR with large databases.
- *Unsupervised Learning for Visual Attention* We will be investigating unsupervised learning in the context of measuring visual attention and similarity in images. This work will build upon existing algorithms for visual attention to develop measures of similarity that do not depend upon sets of pre-selected features or labeled data.
- *Learning from User Interaction in CBIR* The objective of this work is to exploit the information provided by user's interaction in CBIR. All these semantic annotations could be integrated in order to improve the search engine. The work will use similarity matrices of the database images. The similarity matrix is analyzed in the kernel matrix framework. We will deal with efficient kernel adaptation techniques, taking care to preserve the properties of kernels.
- *Scene Event Analysis* We will work on scene event analysis and indexing, motion tracking and gesture recognition by using Bayes, PCA and SVM.
- *Ontology Discovery* Traditional NLP techniques for determining semantic similarity between words and identifying 'is-a' relationships, include bag-of-word techniques and lexico-syntactic features as presented above. We will work on combining these techniques to automatically create taxonomies. There are a lot of practical issues when building systems that automatically extract taxonomies from text. These include the design and stopping criteria for the chunk parser and the semantic generalizer, the semantic similarity measures (could be different at different levels of the taxonomy) and the evaluation of the results.

- *Learning shape probability distributions for regions.* Prior information about the geometry of the regions in an image that correspond to some entity is critical for extraction of many types of semantics, and yet existing models are very limited in their ability to describe this information. Learning a model (a.k.a. probability distribution) for region geometry from examples is a crucial part of capturing this prior information because it is frequently impossible to construct the interactions necessary to describe a particular class of regions from *a priori* considerations. Over the coming year, INRIA-Ariana will tackle the problem of learning probability distributions over geometries by applying a Bayesian approach to the higher-order active contours developed in the Ariana group, using a reformulation of these models as statistical field theories.