Clustering and Prototype Based Classification

vorgelegt von
Diplom-Informatikerin

Sambu Seo

Falkultät IV-Elektrotechnik und Informatik
der Technischen Universität Berlin
Institut für Softwaretechnik und Theoretische Informatik

zum Erlangen des akademischen Grades
Doktor der Naturwissenschaft
(Dr. rer. nat.)

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. O. Hellwich
Berichter:	Prof. Dr. K. Obermayer
Berichter:	Prof. Dr. T. Scheffer

Tag der wissenschaftlichen Aussprache: 7. November 2005

Berlin 2005
D 83

.

Acknowledgments

This thesis could not have been written without the support of many people.

First of all, I would like to express my deep gratefulness to my mentor, Prof. Klaus Obermayer. His lecture on 'neural information processing' inspired me to enter the field of machine learning. I am very grateful to him for letting me join his creative and very international research group. He has been a driving force behind my research by inspiring me, providing me with critical advice and always having an open ear for problems. During our almost daily jogging through the park of Schloss Charlottenburg we had many interesting talks and discussions. I also deeply appreciate the support of Prof. Mathias Bode during my time at the company Cortologic. His critical and detailed questions guided my way in thinking scientifically. I wish to thank Prof. Tobias Scheffer for agreeing to be a reviewer of my thesis and for interesting discussions. I am very grateful to Prof. Zhang Byung-Tak for inviting me to present my research at Seoul National University and for interesting discussions during his stay in Berlin. I also owe thanks to Prof. Joachim Buhmann for providing the protein data set. Prof. Thomas Hofmann I wish to thank for letting me use the document data.

I want to thank all my collegues at the NI research group. Especially, I thank for their friendship Dr. Hendrik Purwins, Lars Schwabe, Dr. Holger Schöner, Dr. Gregor Wenning, Susanne Schönknecht, Prof. Xie Song-Yun and my room-mate Roland Vollgraf. Many thanks go to Dr. Sepp Hochreiter for providing his PSVM method for feature selection, for fruitful discussions and his many funny jokes. I am indebted to Dr. Hauke Bartsch for proof-reading one of my papers and to Dr. Christian Klose for our joint work of applying machine learning to the field of geology. For interesting conversation during lunch breaks I thank Thomas Hoch, André Paus, Stephan Schmitt, Claudia Sannelli, Steffen Grünewälder, Dr. Michael Sibila, Peter Wiesing, Oliver Beck, Robert Anniés, Kamil Adiloglu and Klaus Wimmer. I appreciate the help of Gabriele Rösler and Camilla Bruns for dealing with all the bureaucracy.

In particular, I am most grateful to my parents for their support and love. I will never forget they gave me the opportunity for my studies in Korea and Germany, their trust and never-ending support. I am also most grateful to my husband, Johannes Mohr, for his support as colleague and as best friend. I thank him for reading this work critically, correcting my English and making suggestions for last-minute improvements. I wish to thank him most of all for his love and understanding in difficult times during my thesis.

II

아버지 어머니께 바칩니다.

Contents

List of Figures

Chapter 1

Introduction

1.1 Cogito Ergo Sum

In the 17th century the founder of modern philosophy, René Descartes (1596-1650), inferred his existence from the fact that he was thinking, which includes doubting, understanding, conceiving, affirming, denying, willing, imaging and feeling. This shows that thinking is an essential ability, and being able to think is a necessary condition for a human being. All thinking processes require the ability for distinction, ordering and abstraction, which all involve classification and categorization. Throughout human history, both the objects to be classified and categorized, as well as the human ability to classify and categorize them get increasingly complex, the one influencing the other. This development of the mind required an increasing level of abstraction. The notion of categorization can already be found in the philosophy of Aristotle (384-348 B.C.), whose concept of 'essential and accidental properties' corresponds to the differences of objects between and within categories.

Man has always invented tools to assist him in the task of classification and categorization, the most recent being the computer. Nowadays, computers are found everywhere, from microchips in washing machines or talking rice-cookers, to robots exploring Mars or the moons of Jupiter. Humankind has long been dreaming of 'capturing' human intelligence within a machine, an idea which has inspired numerous novels and movies (e.g. the robot-novels by Asimov and Polanksi's movie 'A.I.'). Such systems are still far beyond reality. However, in the last few decades the construction of learning machines has become an important object of scientific research.

Already in the 1950s learning and adaptive systems were investigated, giving rise to the view of learning as an adjustment of model parameters. This inspired the first works on artificial neural networks. Rosenblatt (1958) studied single layer networks, called them perceptrons and applied them to classification tasks. However, perceptrons could only classify data sets which are linearly separable (Minsky and Papert (1969)). A major breakthrough in the field of neural networks came in the 1980s, with the availability of more powerful computers. This lead to algorithms like Hopfield networks (Hopfield (1987)) and the Back-propagation algorithm for learning multi-layer-perceptrons, which stepped beyond the limitations of single layer networks. Another important milestone of the 1980s was the development of self-organizing maps (Kohonen (1982b,a)), using competitive learning. In the 1990s, a new class of very efficient

algorithms, the support-vector-machines (Vapnik (1995, 1998)), were developed, based on discoveries previously made in statistical learning theory. Further details on the history of machine learning can for example be found in (Haykin (1994); Duda et al. (2001); Ripley (1996)). Presently, the still ongoing increase in computational power, accompanied by the collection of huge data bases, increases the need and at the same time provides the resources for automatic algorithms which discover structure in the data. This development is an example of how scientific and technological progress are closely intertwined.

Computer technology has also had a profound impact on most other scientific fields. A prominent example is biomedical image analysis, in which either the images themselves or some depicted objects have to be classified. Because of the large amounts of image data, automatic algorithms need to be developed to assist in this often very tedious task. Furthermore, the information technological advance basically revolutionized molecular biology, which gave rise to the field of bio-informatics. The immense amounts of data gathered for example by sequencing machines, in an effort to understand the human genome, has made the need for advanced data mining techniques quite obvious. Another breakthrough has occurred with respect to data transfer and communication. Internet and web-based technologies have lead to gigantic pools of information, which are highly linked and dynamically growing. However useful, the informational overload has again caused some other problems: Looking for a certain bit of information has become a task like finding the needle in a haystack. This calls for advanced algorithms for filtering and information retrieval. Closely connected to this is the task of text categorization, in which similarities between documents or web-pages, based on their topics, have to be found. This list could be continued much longer; the field of applications which make use of automatic techniques for classification and categorization, be it supervised or unsupervised, is huge and still growing.

In this thesis, a number of new machine learning algorithms were developed and applied to several of the above-mentioned fields.

1.2 Organization of the thesis

This thesis consists of two parts. In the first part, I derive three novel learning algorithms for learning of nearest prototype (NP) classifiers based on a Gaussian mixture model. The performance of the algorithms is investigated and compared to other NP classifiers derived from different cost functions and to support vector machines. In the second part, clustering and embedding algorithms for pairwise data are derived based on rate distortion theory. Furthermore, the information bottleneck method is extended using the source coding framework such that it preserves the topographic relation between the clusters.

1.2.1 Outline of Part I

In this part, I introduce three new nearest prototype classification (NPC) algorithms and compare them to another three extensions of the LVQ algorithms. Chapter 2 can be seen an introduction to part I. It contains a short review of the Bayesian decision rule, K nearest neighbor classification and the LVQ algorithm. It motivates the use of nearest

prototype classification and discusses the core ideas of the proposed Gaussian mixture based learning algorithms. In chapter 3 an algorithm for learning an NP classifier based on a Gaussian mixture model is derived by minimizing the misclassification rate. This approach shows the relationship between the Bayesian decision rule and nearest prototype classification. The properties of the newly derived loss function are considered from the viewpoint of active data selection. The main contributions of this chapter are published in Seo et al. (2003).

In chapter 4 an NPC algorithm derived based on the maximization of the likelihood ratio and the corresponding hard version are introduced. The properties of the cost functions are investigated w.r.t. the convergence properties. This analysis provides insight into why prototypes tend to diverge when LVQ is used to generate an NP classifier. In order to prevent the divergence of the algorithm, a window rule is suggested which works better than the one of LVQ 2.1 (this is shown in the experiments in chap. 7.) In chapter 5 an algorithm for learning an NP classifier is derived based on a likelihood ratio upper-bounded by 1. This algorithm does not diverge and hence does not need a heuristic window rule for active data selection. The main contributions of chapters 4 and 5 are published in Seo and Obermayer (2003).

In chapter 6 three previously existing learning algorithms (McDermott and Katagiri (1994); Sato and Yamada (1996); Crammer et al. (2002)), which were derived from different cost functions, are shortly reviewed. The properties of their loss functions are investigated. The cost functions of these algorithms are functions of the distance between the nearest correct and the nearest incorrect prototypes. Hence in this thesis I call them 'distance based LVQ' algorithms to distinguish them from the learning algorithms which are derived based on a Gaussian mixture model. The properties of the three soft LVQ algorithms are compared to the three distance based LVQ algorithms. Chapter 7 contains benchmark results on the UCI `letter` and `pendigits` datasets. In order to alleviate the computational complexity of the proposed algorithms and LVQ 2.1 w.r.t the selection of an optimal hyper parameter, I propose a heuristic dynamic hyper parameter scaling method, which very slowly decreases the hyper parameters at each training step. The performance of the algorithms with this dynamic scaling of hyper parameters is investigated and compared to the performance of the algorithms where the hyper parameters are selected via 10-fold cross validation.

1.2.2 Outline of Part II

In the second part of my thesis I motivate and derive several clustering algorithms for pairwise data and Self-Organizing Map methods for pairwise and co-occurrence data, using concepts from source and channel coding. For this purpose, the view of rate distortion theory (which is described in section 8.5) is adopted. I show a generic way to create a Self-Organizing-Map-like embedding of data objects for various kinds of approaches on data sets which are represented as matrices. The new formulation naturally leads to an optimization method which is related to maximum entropy methods (Jaynes (1957a,b)) and deterministic annealing (Rose (1998), Hofmann and Buhmann (1997)). In chapter 9 an algorithm for clustering and embedding of proximity data is derived. Its advantages, disadvantages and relation to the previously proposed maximum entropy methods are discussed. The behavior of the clustering solution and the Lagrange parameter, which weighs the contribution of the entropy term in the

Lagrangian against the constraint, is considered. Furthermore, I suggest an improvement using a 'growing' algorithm which incrementally increases the number of clusters until an optimal value is reached. In chapter 10 an algorithm for the embedding of co-occurrence data is derived using the SOM method, which breaks the permutation symmetry of the clustering problem. The performance of the methods will be assessed using several toy and real world data sets. Chapter 11 contains a summary and a discussion.
The main contributions of this part are published in Seo and Obermayer (2004).

1.3 List of Symbols of Part I

Symbol	Meaning
x, x_i	primary data points
y, y_i	labels of data points x and x_i
$\bar{y}$	set of class labels which differ from y
θ_m	m-th prototype
c_m	label of m-th prototype
$\mathcal{X}$	set of primary data points
$\mathcal{I}$	set of class label, normally set of indices
$\mathcal{S}$	training data set
$\mathcal{T}$	set of labeled prototypes
N	number of training data points
D	dimensionality of x
M	number of prototypes
N_y	number of classes
E	cost function
ls	loss function
$\delta(a)$	delta function: $= 1$, if a is true else 0
$P(l)$	prior probability of component l of the mixture model
$P(l\|x)$	assignment probability of data point x to component l
$P_y(l\|x)$	assignment probability of data point x to component l with correct label
$P_{\bar{y}}(l\|x)$	assignment probability of data point x to component l with incorrect label
$p(x\|l)$	conditional density function
$f(x_i, \theta_l)$	basis function of normalized exponential conditional density function
$p(x\|\mathcal{T})$	probability density of data point x given model $\mathcal{T}$
$p(x, y\|\mathcal{T})$	conditional probability density that a data point x is generated by the correct class label y
$p(x, \bar{y}\|\mathcal{T})$	conditional probability density that a data point x is generated by the incorrect class label $\neq y$
σ^2	uniform width of all components
σ_l^2	width of component l.
α, α^*	learning rate of online learning
$d(a, b)$	distance between two data points in the same space
$\mathcal{L}_c$	likelihood that data points are generated by mixture model for correct class
$\mathcal{L}_u$	likelihood that data points are generated by mixture model for incorrect class
$\mathcal{L}_r$	likelihood ratio
ω	width parameter of window rule, LVQ, SLVQ-LR, LVQ-LR
η	window parameter of distance based LVQ algorithms
$\eta(t)$	function of learning time t of generalized LVQ

1.4 List of Symbols of Part II

In the following index and objects are denoted by lower case, variable and distance functions by upper case, sets of objects and Lagrange functions by calligraphic case and matrices by upper case bold face letter.

Symbol	Meaning
$\mathcal{X}, \mathcal{Y}$	set of objects
$\tilde{\mathcal{X}}, \tilde{\mathcal{Y}}, \mathcal{C}$	set of clusters
x, x', y, z	data points
$i, j, l, n, m, s, t, c, d$	index
$d(i, j)$	pairwise distance between the i-th and j-th data point
$d(x, x')$	pairwise distance between data points x and x'
$\mathbf{D}$	dissimilarity matrix
$P(l\|x), P_1(l\|x)$	assignment probability of data point x to cluster l
$P(x'\|l), P(y\|l)$	read-out probability
$P_2(x'\|d), P_2(y\|d)$	read-out probability after noise process has acted on representations
$P(l)$	probability of cluster l
$P(x)$	probability of data point x
$q(d\|c)$	neighborhood function, confusion probability between clusters c and d
$I(C; X)$	mutual information between two random variable C and X
$I(C\|X)$	conditional entropy of variable C given the variable X
$\mathcal{L}_1, \mathcal{L}_2$	Lagrange functionals
$E, E(X, X'), E(X, Y)$	measure of reconstruction error
E_0	constant
$D_{KL}(P\|\|Q)$	Kullback-Leibler Divergence

Part I

Nearest Prototype Classification based on Gaussian Mixtures

Chapter 2

Prototype Based Classification

2.1 Classification

In a classification problem, we are given a data set $\mathcal{S}$ which consists of N pairs of data points

$$\begin{aligned} \mathcal{S} &= \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}, \\ x_i &\in \mathcal{X} \subset \mathbb{R}^D, \\ y_i &\in \mathcal{Y} \equiv \mathcal{I} = \{1, 2, \ldots, N_y\}, \qquad \forall\, i = 1, \ldots, N, \end{aligned} \tag{2.1}$$

where $\mathcal{X}$ is the input space, which is a D-dimensional real space, and $\mathcal{Y}$ is the output space, which is a space of discrete class labels. x_i is a D-dimensional vector, and y_i is the class label of x_i. $\mathcal{I}$ denotes a set of indices of class labels and N_y is the number of classes. The task of classification is to find the relationship f (a classifier), between the input space $\mathcal{X}$ and the output space $\mathcal{Y}$

$$f : \mathcal{X} \to \mathcal{Y}$$

based on the given training data set $\mathcal{S}$. For the classification problem, the output space is a index space of discrete class labels with finite cardinality. If the output space was a subset of $\mathbb{R}$, then we would have a regression problem, which is a problem of function approximation. The commonly used confidence measure for a classifier is the generalization error of the classification, a risk functional,

$$\begin{aligned} E(f) &= \int_{(x,y)\in\mathcal{X}\times\mathcal{Y}} l(f(x), y) p(x, y) dx dy, \\ &= \sum_{j=1}^{N_y} \int_{\mathcal{X}} l(f(x), j) P(j|x) p(x) dx, \end{aligned}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \{0, 1\}$ is the zero-one loss function and $p(x, y)$ is the probability density over labeled instances (x, y). In practice, it is impossible to obtain this measure, because the probability density $p(x, y)$ is unknown, and we do not have knowledge about it. Under the assumption that the training data points $(x, y) \in \mathcal{S}$ were sampled independently and identically distributed (i.i.d.) from the underlying distribution, the

generalization error can be estimated by the empirical error

$$E(f, \mathcal{S}) \;=\; \frac{1}{N}\sum_{i=1}^{N} l(f(x_i), y_i),$$

where the index i runs over all data points of the training set $\mathcal{S}$.

2.2 Classification Using the Bayesian Decision Rule

Bayesian decision theory is a probabilistic approach which assigns a data point to the class $f^*(x)$ with the maximal posterior probability

$$f^*(x) \;=\; \arg\max_l P(l|x), \tag{2.2}$$

where the posterior probability $P(l|x)$ is given by the Bayesian theorem

$$P(l|x) \;=\; \frac{p(x|l)P(l)}{\sum_l p(x|l)P(l)} \;=\; \frac{p(x|l)P(l)}{\hat{p}(x)}. \tag{2.3}$$

Here $P(l)$ is the prior probability of the l-th class and $p(x|l)$ is the class conditional probability density, which is called the likelihood of the l-th class with respect to x. The term $\hat{p}(x)$ is not the true probability density of x, but an evidence factor and serves as a scaling factor, which guarantees that the posterior probabilities sum to one. Because the evidence is independent of the class label, a posterior maximization (MAP) can be reduced to the maximization of the term $p(x|l)P(l)$. The individual conditional risk for a zero-one classification error is as following

$$\begin{aligned} \mathrm{ls}(f, x) \;&=\; \sum_{j=1}^{N_y}(1 - \delta(f(x) == j))P(j|x), \\ &=\; 1 - P(f(x)|x), \end{aligned} \tag{2.4}$$

where δ is a Kronecker function, i.e. $\delta(\alpha) = 1$, if α is true and 0 otherwise. Because of the eq. (2.4), the Bayesian decision rule, eq. (2.2), minimizes the conditional risk for every x and so minimizes the expected generalization error (Duda et al. (2001))

$$\begin{aligned} E(f) \;&=\; \sum_{j=1}^{N_y}\int_{\mathcal{X}} \mathrm{l}(f(x), j)P(j|x)p(x)dx, \\ &=\; \int_{\mathcal{X}} \mathrm{ls}(f, x)p(x)dx. \end{aligned} \tag{2.5}$$

The basic ideas underlying Bayesian decision theory are very simple. To minimize the generalization risk, one should assign a data point to the class which minimizes the conditional risk, eq. (2.4). Particularly, in order to minimize the probability of error of classification, a data point should be classified to the class which maximizes the posterior probability, eq. (2.3).

2.3 K-Nearest Neighbor Classification

Nearest neighbor (NN) classification is a metric[1] or distance based classification. For the classification of a test data point x, based on a given distance measure, the nearest neighbor (x') is found from the training set, and x is assigned to the same class as x'. The K-nearest neighbor (KNN) classification rule finds the k data points from the training set which are closest to a test data point x and assigns x to the class the majority of the K neighbors belong to. The KNN classifier can be constructed with the help of the Bayesian theorem, eq. (2.3). Using the class label of the KNN of a data point x, we can estimate the class conditional probability density, the prior probability of classes and the evidence as following (Bishop (1995))

$$p(x|l) \;=\; \frac{K_l}{N_l V}, \qquad \hat{p}(x) \;=\; \frac{K}{NV}, \qquad P(l) \;=\; \frac{N_l}{N},$$

where V is the volume of the hyper-sphere enclosing the KNN of x, and N_l and K_l are the number of data points with class $y = l$ in the training data set and the hyper-sphere respectively. Using these estimations we can calculate the posterior probability applying the Bayesian theorem, eq. (2.3).

$$P(l|x) \;=\; \frac{p(x|l)P(l)}{\hat{p}(x)} \;=\; \frac{K_l}{K}, \tag{2.6}$$

To minimize the misclassification rate, x should be classified to the class which is most frequently represented within the sphere: $h^*(x) = \arg\max_l \frac{K_l}{K}$. If there are enough examples and if the hyper-sphere is sufficiently small, the performance of KNN approaches its best possible value (Duda et al. (2001)). The NN-rule is a special case of the KNN rule with $K = 1$. For the NN-rule the posterior probability is a hard assignment probability, i.e. the posterior probability $P(l|x)$ is 1, if the class l matches the class of the NN of x, and 0 otherwise. The NN rule partitions the input space into the Voronoi tessellation cells of the data space, which consist of all the points closer to a given data point x than to any other training data point. It has been shown that the NN classifier has an asymptotic ($N \to \infty$) error rate that is at most twice the Bayesian error rate $E(f^*)$, which we get by inserting eq. (2.2) in eq. (2.5), independent of the used distance metric (Duda et al. (2001)). But the NN classification relies on the assumption that the class conditional probabilities are locally constant. Due to the curse of dimensionality this assumption becomes invalid in a high dimensional data spaces with a finite number of examples (Bellman (1961); Friedman (1994)).

Nevertheless, a NN-classifier is simple to model, requires no training and is ideal for fast adaptation and the natural handling of the multi-class case. However, it has the disadvantage that all of the training data points must be retained and suffers from the fact that the number of variables in the model grows directly with the number of training data points. This might lead to a computer storage problem and results in models which require a very large amount of processing, which makes the classification of a new data point very slow. The main problem of the NN classifier, the large number of prototypes, causes not only a computational burden and a slow classification, but

[1] A metric is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ that fulfills the following properties for given arbitrary three data points $x_1, x_2, x_3 \in \mathcal{X}$, (i) nonegativity: $d(x_1, x_2) \geq 0$, (ii) reflexity: $d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$ (iii) symmetry: $d(x_1, x_2) = d(x_2, x_1)$, (iv) triangle inequality: $d(x_1, x_2) + d(x_2, x_3) \geq d(x_1, x_3)$.

also the other drawback that the erroneously labeled or noisy prototypes may lead to arbitrary large deviations from the asymptotically optimal results. In order to solve this problem, many methods for prototype selection from the data set were proposed (Chang (1974); Mollineda et al. (2000); Kuncheva (1997); Kuncheva and Jain (1999); Vuori et al. (2000)).

2.4 Nearest Prototype Classifier

The nearest prototype (NP) classifier is a Bayesian classifier under the assumption that the distance measure emphasizes the local neighborhood relation. A nearest prototype classifier consists of a set of labeled prototype vectors

$$\begin{aligned} \mathcal{T} &= \{(\theta_1, c_2), (\theta_2, c_2), \ldots, (\theta_M, c_M)\}, \\ \theta_l \in \mathcal{X} \equiv \mathbb{R}^D, &\qquad c_l \in \mathcal{Y} \equiv \mathcal{I} = \{1, 2, \ldots, N_y\}, \qquad \forall\, l = 1, \ldots, M. \end{aligned} \tag{2.7}$$

The parameters θ_l are vectors in data space and the c_l are their corresponding class labels. Typically, the number of prototypes is larger than the number of classes, such that every classification boundary is determined by more than one prototype. The class y of a new data point x is determined

1. by selecting the prototype θ_q which is closest to x,

$$q = \arg\min_l d(x, \theta_l), \tag{2.8}$$

   where $d(\cdot, \cdot)$ is the distance between data point and prototype, and

2. by assigning the label c_q of this prototype to the data point x.

A popular choice for d is the Euclidean distance

$$d(x, \theta_l) = \|x - \theta_l\|_2 = \left(\sum_{i=1}^{D} (x^i - \theta_l^i)^2 \right)^{1/2}, \tag{2.9}$$

where x^i and θ_l^i are the i-th feature of the vectors x and θ_l. Other distance measures may be chosen depending on the problem at hand. From eq. (2.8) it is clear that the set of prototype vectors induces a tessellation of data space (see fig. 2.1), and that the classifier assigns the same class label to all data points which fall into the same tessellation. The tessellation is optimal, if all data points within one cell indeed belong to the same class. Like the K-nearest neighbor method, Nearest Prototype Classification (NPC) is a local classification method in the sense that classification boundaries are approximated locally. Instead of making use of all the data points of a training set, however, NPC relies on a set of appropriately chosen prototype vectors. This makes the method computationally more efficient, because the number of items which must be stored and to which a new data point must be compared for classification is considerably less. Therefore, NPC has been a popular method for real time applications, like speech recognition (Komori and Katagiri (1992); McDermott and Katagiri (1994)) and for the analysis of gene data, like the analysis of breast tumor cells (Weigelt et al. (2003)) or cancer class prediction from gene expression profiling (Tibshirani et al.

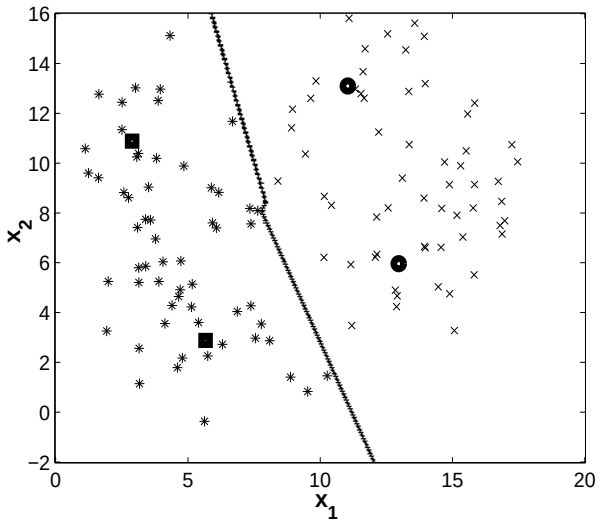


Figure 2.1: Nearest Prototype Classification
The 2D data space with two classes ($\circ, \square$) is split into two subspaces using 4 prototypes, 2 prototypes for each class. A data point is assigned to the same class as the nearest prototype.

(2002)), where it has been used with good success. There are different strategies and algorithms for finding the prototypes, like Learning vector quantization (LVQ) classifiers (Kohonen (1990); Kohonen et al. (1995); Centre (2003)), edited nearest neighbor rules (Dasarathy (1990)), fuzzy nearest neighbor rules (Bezdek (1985); Keller et al. (1985); Yau and Manry (1991)), a number of neural network implementations of the nearest neighbor design (Decaestecker (1993); Lippmann (1989)), NP signal classifiers (Gardner (1981)) and learning algorithms for NPC (Urahama and Nagao (1993)).

2.5 Learning Vector Quantization

Learning Vector Quantization (LVQ) (Kohonen (2001), Kohonen et al. (1995)) is a class of learning algorithms for nearest prototype classification (NPC), which make use of the distribution of the training data as well as their class labels when selecting a useful set of prototypes. LVQ was introduced by T. Kohonen (Kohonen (1986)) almost 20 years ago. Since then, it has been widely used (see Centre (2003) for an extensive bibliography), and new versions with improved classification performance have been proposed (Kohonen (1990); Kohonen et al. (1995)). Let us consider LVQ 2.1 as an example which has been shown to provide good NPC classifiers (Kohonen (1990); Centre (2003)). For every data point (x, y) from the training set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N}$, LVQ 2.1 first selects the two nearest prototypes θ_l, θ_m according to the Euclidean distance. If the labels c_l and c_m are different and if one of them is equal to the label y of the data point, then the two nearest prototypes are adjusted according to

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) + \alpha(t)(x - \theta_l), & c_l &= y, \\ \theta_m(t+1) &= \theta_m(t) - \alpha(t)(x - \theta_m), & c_m &\neq y. \end{aligned} \tag{2.10}$$

If the labels c_l and c_m are equal or both labels differ from the label y of the data point, no parameter update is being performed. The prototypes, however, are changed only if the data point x is close to the classification boundary, i.e. if it falls into a *window*

$$\min\left(\frac{d(x,\theta_m)}{d(x,\theta_l)}, \frac{d(x,\theta_l)}{d(x,\theta_m)}\right) > s, \quad \text{where} \quad s = \frac{1-\omega}{1+\omega}, \tag{2.11}$$

of relative width $0 < \omega \leq 1$ (Kohonen (2001)). This 'window rule' had to be introduced, because otherwise prototype vectors may diverge. Figure 2.2 shows, for different width parameters ω, the active region of LVQ 2.1, in which data points fall into the window and are used for the adjustment of the prototypes. The active region (denoted as black area) lies in the vicinity of the class boundary for small ω and it is enlarged by increasing ω. For fixed ω, the active region depends on the distance between data point and class boundary and on the distance between data point and prototypes. The influence of the latter on the active region is decreased by decreasing ω. For $\omega = 1$, every data point falls into the window, i.e. the prototypes are changed by every data point according to the rule, eq. (2.10). For large ω and for reasonable prototypes, the average distance between the data points and the nearest incorrect prototypes is larger than the average distance between the data points and the correct prototypes. Therefore, during the learning process the average amount of repulsion from the data set is larger than the average amount of attraction to the data set. This phenomenon leads to the divergence of the prototypes. For small ω, the average amount of change of the two nearest prototypes is almost the same, because (i) the prototypes are adjusted only by the data points near the class boundary and (ii) (for inseparable class problems) generally the class distribution near the boundary is random. Therefore, using appropriate ω, the divergence problem can be solved.

2.6 Two Motivations for NPC

NP classifiers have been motivated in the literature in two ways. One motivation comes from Bayesian decision theory (see section 2.2), which is a fundamental approach to classification problems. Construction of the classifier involves two steps, (*i*) the construction of models for the probability densities of the different classes and (*ii*) the construction of the classification boundaries using the criterion of maximum a posteriori (MAP) probability. If - for example - probability densities are well approximated by Gaussian mixtures (Kambhatla and Leen (1995)) and if all components are assumed to be of equal strength and variance, the MAP classifier reduces to an NPC based on Euclidean distances between the data points and the centers of the components. Approaches of this kind can work well (Kambhatla and Leen (1995)) but have to cope with the disadvantage that a more difficult problem (estimation of probability densities) has to be solved than necessary (estimation of the classification boundary) and more training data is needed.

The second motivation comes from the idea of directly estimating the discriminant functions for multi class classification problems. In NPC, the discriminant functions are parameterized using a set of prototype vectors for each class, and classification is based on the distance between a data point and the class to which its closest prototype belongs to. Often an Euclidean distance measure is used. One of the most common

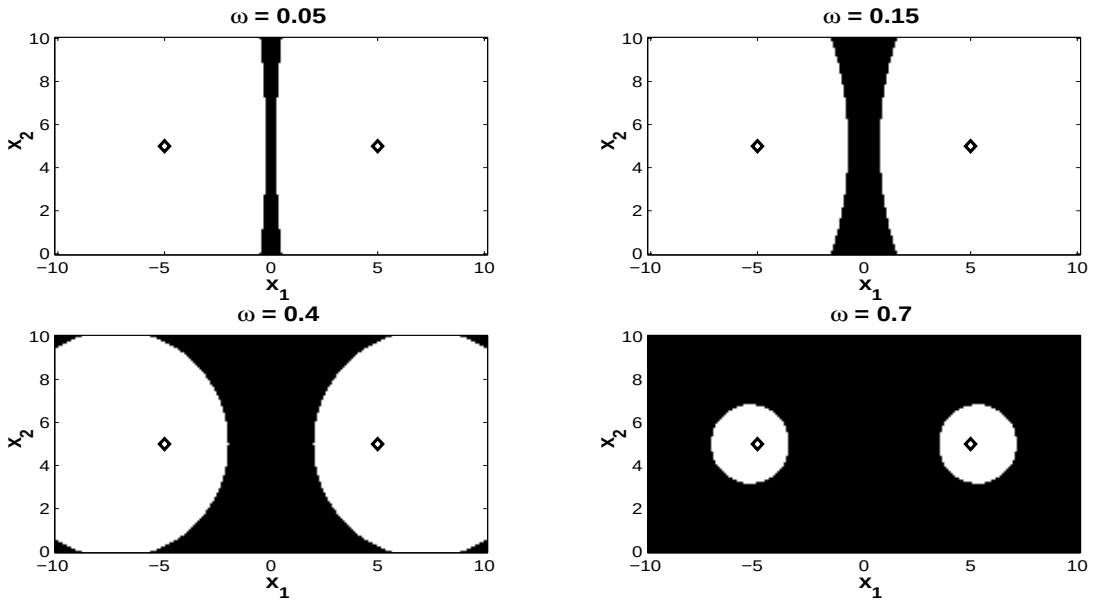


Figure 2.2: Active Region of LVQ 2.1 for different width parameters ω
This figure depicts a 2D problem with two prototypes of different classes. It shows how the active region, which is the region of data space where the data points fall into the window, changes with the window width parameter ω. The active region is narrow and close to the class boundary for small ω, whereas it spreads and and is enlarged by increasing ω. For fixed ω, the active region depends on the distance between the data point and the class boundary as well as on the distance between the data point and the prototypes. The influence of the latter distance on the active region decreases with decreasing ω.

methods for the construction of prototype-based discriminant functions is Learning Vector Quantization (LVQ) (Kohonen (1990, 2001)) for which several variants have been developed in the past. The constructed classifiers work well in many classification tasks (see Centre (2003)), but the selection ('learning') rules have the disadvantage of being heuristic and may not be optimal. In order to improve classification performance, cost functions whose minimization is related to LVQ learning procedures have been proposed by Juang and Katagiri (1992); McDermott and Katagiri (1994); Sato and Yamada (1996); Crammer et al. (2002). The suggested cost functions were derived in a two step procedure. First a measure of performance was constructed which depends on (squared) Euclidean distance between a data point (x, y) and the two nearest prototypes with correct ($c_i = y$) and with incorrect label ($c_i \neq y$), respectively. Then an individual loss for every data point is defined based on the predefined performance measure. A continuous loss function was used in order to apply gradient-based optimization and several hyper-parameters were introduced for this purpose. Model selection is performed by (stochastic) gradient descent on the total loss over the training set. These approaches provide good classification results for various applications, and the derivation of LVQ-like learning procedures using a cost-function approach is valuable when it comes to the analysis of convergence properties. But a disadvantage remains, because the choice of the cost functions is still a heuristic method. They were constructed with the goal in mind to derive LVQ as an optimization procedure, but it may be hard to judge whether their particular form is the best choice for the data at hand.

2.7 New Approaches for NPC

In Seo et al. (2003); Seo and Obermayer (2003) we tried to overcome some of the abovementioned difficulties by combining an explicit ansatz for the probability densities of the classes (cf. the Bayesian approach) with a criterion for model selection which directly minimizes the rate of misclassification (cf. the discriminant function approaches). On the one hand, this ansatz helps to make the assumptions underlying the choice of discriminant functions and model selection more explicit - because the underlying generative model is made explicit. On the other hand, we expected the method to make more efficient use of the information which is contained in the class labels of the training set, because the discriminant function is optimized directly. Using a Gaussian mixture ansatz we derived three different LVQ learning procedures. In the first part of my thesis, I present the variants of the LVQ method which are derived using a principled approach. In order to construct the learning algorithms, we first assume that the probability density $p(x)$ of the data points x can be described by a mixture model and that every component l of the mixture is homogeneous in the sense that it generates data points which belong only to one class C_l. The probability density of the data is then given by:

$$p(x|\mathcal{T}) \;=\; \sum_{c=1}^{N_y} \sum_{\{l:c_l=c\}} p(x|l)P(l), \tag{2.12}$$

where N_y is the number of classes and c_l is the class label of the data points generated by component l. $P(l)$ is the probability that data points are generated by a particular

component l and $p(x|l)$ is the conditional probability that this component l generates a particular data point x. Then we consider a data point x and its true class label y, and define the restricted probability densities

$$\begin{aligned} p(x, y|\mathcal{T}) &= \sum_{\{l:c_l=y\}} p(x|l)P(l), \\ p(x, \bar{y}|\mathcal{T}) &= \sum_{\{l:c_l\neq y\}} p(x|l)P(l). \end{aligned} \tag{2.13}$$

$p(x, y|\mathcal{T})$ is the probability density that a data point x is generated with the correct class label y, and $p(x, \bar{y}|\mathcal{T})$ is the probability density that a data point x is generated with another class label $\neq y$. From this we construct three cost functions,

1. $$\frac{1}{N}\sum_{j=1}^{N}\frac{p(x_j, \bar{y}_j|\mathcal{T})}{p(x|\mathcal{T})}$$
   the average probability that a data point is assigned to the 'incorrect' class

2. $$\prod_{j=1}^{N}\frac{p(x_j, y_j|\mathcal{T})}{p(x_j, \bar{y}_j|\mathcal{T})}$$
   the ratio of the likelihoods that the data is generated by the components of the mixture model with the correct label and that it is generated by the components with the incorrect label.

3. $$\prod_{j=1}^{N}\frac{p(x_j, y_j|\mathcal{T})}{p(x_j|\mathcal{T})}$$
   the ratio of the likelihoods that the data is generated by the components of the mixture model with the correct label and that it is generated by the mixture model.

(1) should be minimized, while (2) and (3) should be maximized. We consider the general case that the conditional density functions $p(x|l)$ are from the normalized exponential form

$$p(x|l) = K(l)\exp f(x, \theta_l), \tag{2.14}$$

and the special case that the conditional density functions $p(x|l)$ are Gaussian. From the second case we derive the LVQ-like learning algorithms for NPC. Because classification using NP classifiers depends only on the relative distances between data points and prototypes, we assume that every component has the same width and strength i.e.

$$\sigma_l^2 = \sigma^2, \qquad P(l) = \frac{1}{M}, \quad \forall\, l = 1, \ldots, M. \tag{2.15}$$

For a mixture ansatz with D-dimensional Gaussian components of the same width and strength,

$$p(x|l) = \frac{1}{(2\pi\sigma^2)^{D/2}}\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right), \tag{2.16}$$

LVQ-like learning algorithms are derived using a stochastic gradient descent/ascent method, i.e. the prototypes with the same label as x are attracted to the data point x and the prototypes with different labels as x are repelled. In contrast to the LVQ algorithms, for a given data point these three algorithms adjust not only the two nearest prototypes but all prototypes. Hence in this thesis I call the three algorithms based on a Gaussian mixture model 'soft LVQ algorithms'. In the following chapters, the properties of the cost functions are analyzed and the performance of the derived algorithms is investigated.

Chapter 3

Soft Nearest Prototype Classification (SNPC)

In this chapter I introduce a new method for the construction of nearest prototype classifiers which is based on a Gaussian mixture ansatz and which can be interpreted as an annealed version of Learning Vector Quantization. This method is a combination of an explicit ansatz for the probability densities of the classes with a criterion for model selection which directly minimizes the rate of misclassification. Under the assumption that the probability density $p(x)$ of the data points x can be described by a mixture model, two conditional probability densities are defined that a data point x is generated by the correct class and by the incorrect classes. I define a cost function as the ratio of the two probability densities. In order to obtain an NP classifier, it is assumed that the width and strength of the components of the Gaussian mixture are almost the same for all components, because the NP classifier depends only on the distance between data points and prototypes. The algorithm performs a gradient descent on a cost-function minimizing the classification error on the training set. This algorithm will be called soft nearest prototype classification (SNPC). The properties of the algorithm are investigated and the learning process is illustrated using several toy data sets.

3.1 Cost Function and Learning Rule: General Case

Let us consider a data set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^N, x_i \in \mathcal{X} \equiv \mathbb{R}^D, y_i \in \mathcal{I}$, where N is the number of (training) data points and $\mathcal{I}$ is a set of class labels. Our goal is to find the NPC which optimizes classification performance. Therefore, we select a set $\mathcal{T} = \{(\theta_l, c_l)\}_{l=1}^M$ of labeled prototype vectors whose parameters Θ are determined by minimizing the rate $E(\mathcal{S}, \mathcal{T})$ of misclassification with respect to θ_l for the training set, i.e.

$$
\begin{aligned}
E(\mathcal{S}, \mathcal{T}) &= \frac{1}{N} \sum_{k=1}^{N} \sum_{l=1}^{M} P(l|x_k) \left(1 - \delta(y_k == c_l)\right) && (3.1)\\
&\stackrel{!}{=} \min, \\
P(l|x_k) &= \delta(l == q_k), && (3.2)\\
q_k &= \arg\min_r \|x_k - \theta_r\|. && (3.3)
\end{aligned}
$$

$\delta(\gamma)$ is 1, if γ is true, and 0 else. $(1 - \delta(y_k == c_l))$ is the zero-one classification loss and is 0, if the label of the nearest prototype matches the label of data point x_k, and 1 else. $P(l|x_k)$ is the assignment probability, i.e. the probability that a data point x_k is assigned to a prototype l. If $P(l|x_k)$ is 1, for l being the nearest prototype to x_k, and 0 else, the NPC operates in a '*winner-takes-all*' mode.

In order to minimize $E(\mathcal{S}, \mathcal{T})$ with respect to the parameters θ, we introduce fuzzy assignment probabilities $P(l|x)$. Replacing hard by soft assignments allows for a gradient-based optimization procedure, as we will see soon. Also, classification errors which result from misclassification of data points near the class boundaries are weighted less, which avoids oscillations in the values θ during learning and leads to a faster convergence rate. If the assignment probabilities are from the normalized exponential form

$$P(l|x) \;=\; \frac{\exp\left(-d(x,\theta_l)\right)}{\sum_{k=1}^{M}\exp\left(-d(x,\theta_k)\right)}, \tag{3.4}$$

where $d(x,\theta)$ is the distance measure between data point x and prototype θ, we can rewrite the cost function, eq. (3.1), as the sum

$$E(\mathcal{S},\mathcal{T}) \;=\; \frac{1}{N}\sum_{k=1}^{N}\mathrm{ls}((x_k,y_k),\mathcal{T}) \tag{3.5}$$

of the individual costs, which correspond to the expected conditional loss

$$\mathrm{ls}_k \;=\; \sum_{l=1}^{M} P(l|x_k)(1-\delta(y_k == c_l)) \tag{3.6}$$

$$\;=\; \sum_{\{l:c_l\neq y_k\}} P(l|x_k), \tag{3.7}$$

where ls_k is an abbreviation of $\mathrm{ls}((x_k,y_k),\mathcal{T})$. Equation (3.7) shows that the individual loss of a data point x_k is the sum of the assignment probabilities of the data point x_k to all the prototypes of the incorrect classes. Because the individual costs are continuous and bounded by $0 \leq \mathrm{ls}((x,y),\mathcal{T}) \leq 1$ with respect to Θ, the cost function, eq. (3.5), can be minimized by stochastic gradient descent (Robbins and Monro (1951); Bottou (1998)),

$$\theta_l(t+1) \;=\; \theta_l(t) - \alpha(t)\frac{\partial \mathrm{ls}_t}{\partial \theta_l}, \tag{3.8}$$

where t is the iteration number and $\alpha(t)$,

$$\sum_{t=0}^{\infty}\alpha(t) = \infty \quad \text{and} \quad \sum_{t=0}^{\infty}\alpha(t)^2 < \infty, \tag{3.9}$$

is the learning rate. Using

$$\frac{\partial \mathrm{ls}_t}{\partial \theta_l} = -P(l|x)(\delta(c_l \neq y_t) - \mathrm{ls}_t)\frac{\partial d(x,\theta_t)}{\partial \theta_l}, \tag{3.10}$$

we obtain the following learning rule:

$$\begin{aligned}\theta_l(t+1) &= \theta_l(t) - \alpha(t)\Delta(\theta_l)(t) \\ \Delta(\theta_l)(t) &= \begin{cases} P(l|x_t)\mathrm{ls}_t \frac{\partial d(x_t,\theta_l)}{\partial \theta_l}, & \text{if } c_l = y_t, \\ -P(l|x_t)(1-\mathrm{ls}_t)\frac{\partial d(x,\theta_l)}{\partial \theta_l}, & \text{if } c_l \neq y_t. \end{cases}\end{aligned} \tag{3.11}$$

Because of the 'soft' assignment probabilities, eq. (3.4), the prototypes are modified for a given data point according to the data-dependent learning rate $P(l|x)(\delta(c_l \neq y_t) - \mathrm{ls}_t)$. Once the prototypes are determined, new data points x can be classified using

$$c = \arg\max_{c'} \sum_{\{l:c_l=c'\}} P(l|x). \tag{3.12}$$

If the distance measure emphasizes the local neighborhood, eqs. (3.12) and (3.4) reduce to an NP classifier, eq. 2.8.

3.2 SNPC Based on a Gaussian Mixture Ansatz

Let us consider the mixture model, eq. (2.12), and the definitions of the two conditional probability densities, eq. (2.13). Then we can define the cost function of the classification problem via the loss function of the likelihood ratio

$$E(\mathcal{S},\mathcal{T}) = \frac{1}{N}\sum_{k=1}^{N} \mathrm{ls}((x_k,y_k),\mathcal{T}), \tag{3.13}$$

$$\mathrm{ls}((x_t,y_t),\mathcal{T}) = \frac{p(x_t,\bar{y}_t|\mathcal{T})}{p(x_t,y_t|\mathcal{T}) + p(x_t,\bar{y}_t|\mathcal{T})} = \frac{p(x_t,\bar{y}_t|\mathcal{T})}{p(x_t|\mathcal{T})}. \tag{3.14}$$

The likelihood ratio is a continuous monotonically increasing loss function which takes on the values in the interval [0 1]. Minimization of the cost function, eq.(3.13) leads to the minimization of the average probability that the data points are assigned to the 'incorrect' class (i.e. the average probability that data points are assigned to the 'correct' class is implicitly maximized). For a mixture ansatz with D-dimensional Gaussian components of the same width and strength, eq. (2.15),

$$p(x|l) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right), \tag{3.15}$$

we obtain

$$\begin{aligned}\mathrm{ls}((x,y),\mathcal{T}) &= \frac{p(x,\bar{y}|\mathcal{T})}{p(x|\mathcal{T})} \\ &= \frac{\sum_{\{l:c_l\neq y\}} \exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{k=1}^{M} \exp\left(-\frac{(x-\theta_k)^2}{2\sigma^2}\right)} \\ &= \sum_{\{l:c_l\neq y\}} P(l|x),\end{aligned} \tag{3.16}$$

where

$$P(l|x) = \frac{p(x|l)P(l)}{p(x)} = \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{k=1}^{M}\exp\left(-\frac{(x-\theta_k)^2}{2\sigma^2}\right)} \tag{3.17}$$

is the posterior probability that the data point x was generated by the component l. The individual cost given by eq. (3.16) is a special case of the individual cost given by eq. (3.7) with the distance measure $d(x,\theta_l) = \frac{(x-\theta_l)^2}{2\sigma^2}$.

Stochastic gradient descent then leads to the learning rule

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) - \alpha^*(t)\Delta(\theta_l)(t), \\ \Delta(\theta_l)(t) &= P(l|x_t)(\delta(c_l \neq y_t) - \mathrm{ls_t})(\mathrm{x_t} - \theta_l) \\ &= \begin{cases} -P(l|x_t)\mathrm{ls}_t(x_t - \theta_l), & \text{if } c_l = y_t, \\ P(l|x_t)(1-\mathrm{ls}_t)(x_t - \theta_l), & \text{if } c_l \neq y_t, \end{cases} \end{aligned} \tag{3.18}$$

with $\alpha^*(t) = \frac{\alpha(t)}{\sigma^2}$. In contrast to the algorithms of the LVQ-family, which are based on hard assignments (only the two winner prototypes from the correct and incorrect classes are affected), all of the prototypes with correct label are attracted towards the data point x, proportionally to their distance and weighted by the factor $P(l|x)\cdot \mathrm{ls}$, whereas all of the incorrect prototypes are repelled from the data point x proportionally to their distance and weighted by the factor $P(l|x)(1-\mathrm{ls})$.

σ^2 is a hyper parameter of the learning rule (3.18) and its choice is critical for good performance. One can select an optimal value from a set of candidates using a model-selection method, like the hold-out test set method or the k-fold cross-validation method. Reasonable candidates for σ^2 lie within $0.1 \times \sigma^2_{\mathrm{mean}} \leq \sigma^2 \leq \sigma^2_{\mathrm{mean}}$, where $\sigma^2_{\mathrm{mean}} = \frac{1}{N_y}\sum_{k=1}^{N_y}\sigma_k^2$ is the average value over all labels k of σ_k^2, which is the mean of the diagonal elements of the covariance matrix of the subset of data points with label k, and N_y is the number of class labels. Alternatively, one can use deterministic annealing of σ^2, which is a useful optimization procedure for clustering problems (cf. Graepel et al. (1997); Miller et al. (1996)). Initially σ^2 is set to a large value and is decreased during optimization ('annealing') until an optimal value σ_f is reached. Figure 3.1 summarizes the learning algorithm, which we will call soft nearest prototype classification (SNPC) in the following.

3.3 The 'Window Rule'

Equations (3.18) show, that SNPC - like the other algorithms of the LVQ family - performs an update of the prototype vectors which depends on whether the label y of the data point x and the label c_l of the prototype θ_l are identical or different. SNPC, however, differs from LVQ by a data-dependent learning rate

$$\begin{aligned} &P(l|x)(\delta(c_l \neq y) - \mathrm{ls}) \\ &\quad = \begin{cases} \mathrm{ls}(1-\mathrm{ls})P_{\bar{y}}(l|x), & \text{if} \quad c_l \neq y \\ -\mathrm{ls}(1-\mathrm{ls})P_y(l|x), & \text{if} \quad c_l = y, \end{cases} \end{aligned} \tag{3.19}$$

1. **set** initial value σ_i^2 and final value σ_f^2 of the width parameter, $0 \ll \beta_0 < 1$ $\gamma > 1$, no. of prototypes for each class, and annealing schedule for $\alpha(t)$,
2. **initialize** prototypes $\{(\theta_k, c_k)\}_{k=1}^M$,
3. **while** ($\sigma^2 > \sigma_f^2$)
   (a) set initial learning rate α_0,
   (b) **repeat**
      i. draw a data point from the training set
      ii. calculate assignment probabilities (eq. (3.17))
      iii. update prototypes (eq. (3.18))
      iv. update learning rate
   until a stopping criterion is fulfilled,
   (c) $\sigma^2 \longleftarrow \beta\sigma^2$.
   (d) $\beta \longleftarrow \beta^\gamma$

Figure 3.1: Summary of the SNPC method.

where $P_y(l|x)$ and $P_{\bar{y}}(l|x)$,

$$P_y(l|x) = \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{n:c_n=y\}} \exp\left(-\frac{(x-\theta_n)^2}{2\sigma^2}\right)},$$

$$P_{\bar{y}}(l|x) = \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{n:c_n\neq y\}} \exp\left(-\frac{(x-\theta_n)^2}{2\sigma^2}\right)},$$

denote the posterior probabilities that data point x belongs to component l with label y or $\neq y$, respectively (see appendix A.1 for a brief derivation).

The common factor $0 \leq \mathrm{ls}(1-\mathrm{ls}) \leq 0.25$ is data-dependent and gives rise to the fact that only data points which fall into a particular area of input space contribute to the update of prototypes. This area is characterized by ls being sufficiently larger than zero and sufficiently smaller than one. It covers all data points close to the current classification boundary of the SNPC. This '*active area*' corresponds to the *window* of the LVQ 2.1 methods and performs a similar function (making the methods more robust) as we will see below. Figure 3.2 shows the size of the active region for a two-dimensional toy example with four prototypes (white symbols) from four classes, for different values of the width parameter σ^2. Gray values indicate the strength of the factor $\mathrm{ls}(1-\mathrm{ls})$ for datapoints which have the same label as their closest prototype. The figure shows that the width of the active area grows with increasing σ^2 and that the value of $\mathrm{ls}(1-\mathrm{ls})$ decreases with increasing distance from the current class boundaries. To accelerate the learning process, one can define a threshold value $0 < \eta \ll 0.25$ for

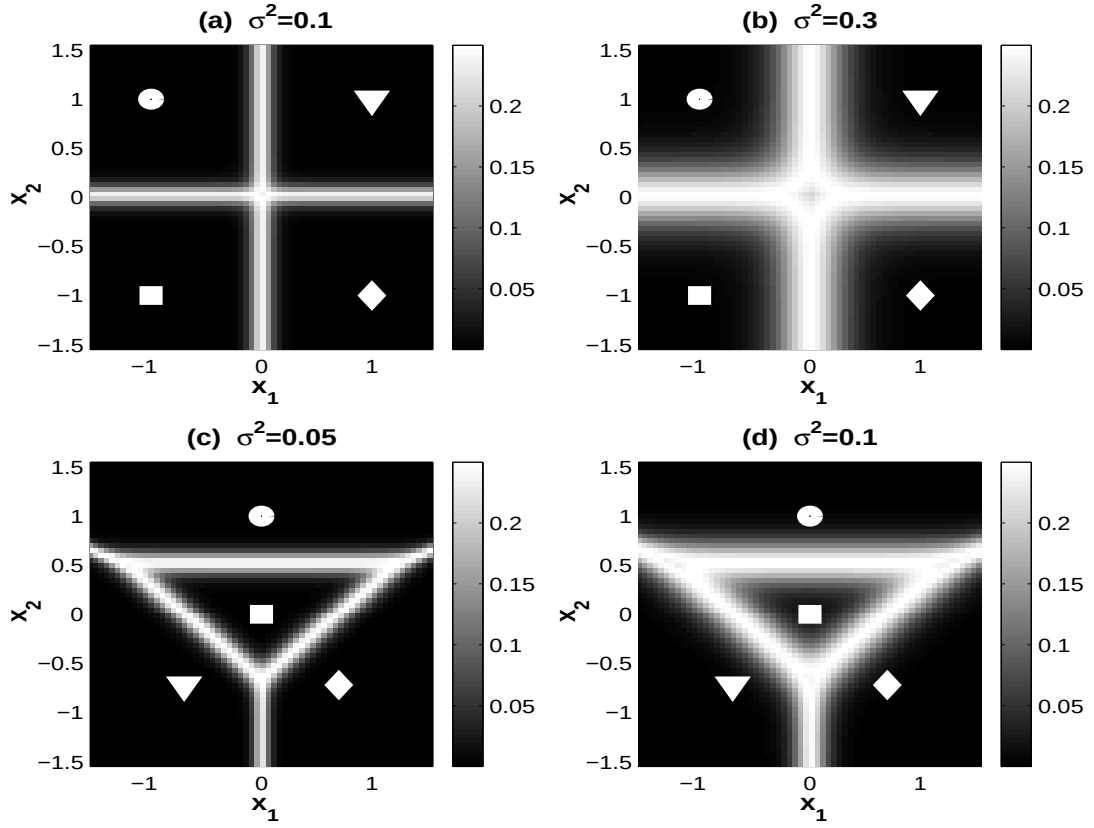


Figure 3.2: Active areas for different values of the width parameter σ^2 and for two different prototype configurations.
White symbols indicate the positions of the prototypes in 2D data space; their class is indicated by the shape of the symbol. The strength of the factor $\mathrm{ls}(1-\mathrm{ls})$ is indicated by brightness ($0 \rightarrow 0.25 \quad \equiv$ black $\rightarrow$ white), for each location in data space and for data points which have the same label as their nearest prototype.

the update and change a prototype only if $\mathrm{ls}(1-\mathrm{ls}) > \eta$. The emergence of a window zone for the update of prototypes can also be exploited by active learning strategies (Freund et al. (1997); MacKay (1992); Seo et al. (2000)): Only data points which fall into the window region must be labeled.

3.4 Numerical Experiments for Toy Examples

I first consider two simple one-dimensional classification problems (Fig. 3.3). In problem no. 1 (Fig. 3.3 a), datapoints are drawn independently and identically distributed (iid) from two Gaussian distributions, which are located at $\mu_1 = -1.5$ (class 1) and $\mu_2 = 1.5$ (class 2) and whose widths are $\sigma_d = 1$. The classifier consists of two prototypes $\theta_{1,2}$, one for each class. In problem no. 2 (Fig. 3.3 b), datapoints are drawn iid from three Gaussian distributions, which are located at $\mu_1 = -3$, $\mu_3 = 3$ (class 1), and $\mu_2 = 0$ (class 2) and whose widths are $\sigma_d = 1$. The classifier consists of three prototypes, two ($\theta_{1,3}$) for class 1 and one (θ_2) for class 2. Figures 3.3 c,d, show the log of the test error E_s^T[1] measured on the test data set with N_t data points,

$$E_s^T = \frac{1}{N_t}\sum_{k=1}^{N_t} \mathrm{ls}((x_k, y_k), \mathcal{T}), \tag{3.20}$$

as a function of the positions of the prototypes for a value $\sigma^2 = 1$ of the width parameter. N_t denotes the number of test data points. For problem no. 2 there is a pronounced minimum at finite values for θ_i and SNPC converges to the these optimal values (Fig. 3.3 f). For problem no. 1 the minimum of E_s^T is assumed at $\theta_1 = \theta_2 \to \infty$, hence the values of the prototypes diverge (Fig. 3.3 e). For the case $\theta_1 = \theta_2 \to \infty$, the class boundary $x_1 = 0$ is fixed and therefore the rate of misclassification,

$$\begin{aligned} E_h^T &= \frac{1}{N_t}\sum_{k=1}^{N_t}(1 - \delta(y_k == c_{q_k})), \\ q_k &= \arg\min_r \|x_k - \theta_r\|^2 \end{aligned} \tag{3.21}$$

remains constant. The 'soft' loss function, however, decreases, if the prototypes diverge. The divergence of prototypes in certain situations is a known feature of the algorithms in the LVQ family and arises because a ratio of probability densities, eq. (3.14), is minimized during learning. Note, however, that this divergence problem does not affect the usefulness of the resulting classifier, because (*i*) the decrease in test error, eq. (3.20), is marginal for large $|\theta_i|$, (*ii*) it rarely occurs in practical problems, and (*iii*) the classification boundary for hard classification, eqs. (3.2, 3.3), does not change (crosses in figs. 3.3,e,f). Learning may thus be terminated as soon as $|\theta_i|$ crosses a given threshold.

Next I consider a two-dimensional classification problem (Fig. 3.4). 300 data points are drawn iid from each of four Gaussian components for training (Figs. 3.4 a,b); the classifier consists of two prototypes for every class. The final prototypes are determined using SNPC with annealing in the width parameter. The schedule of annealing was

[1] The subscript s of the test error denotes risk for the 'soft' assignment probabilities, while the subscript h is for 'hard' assignment .

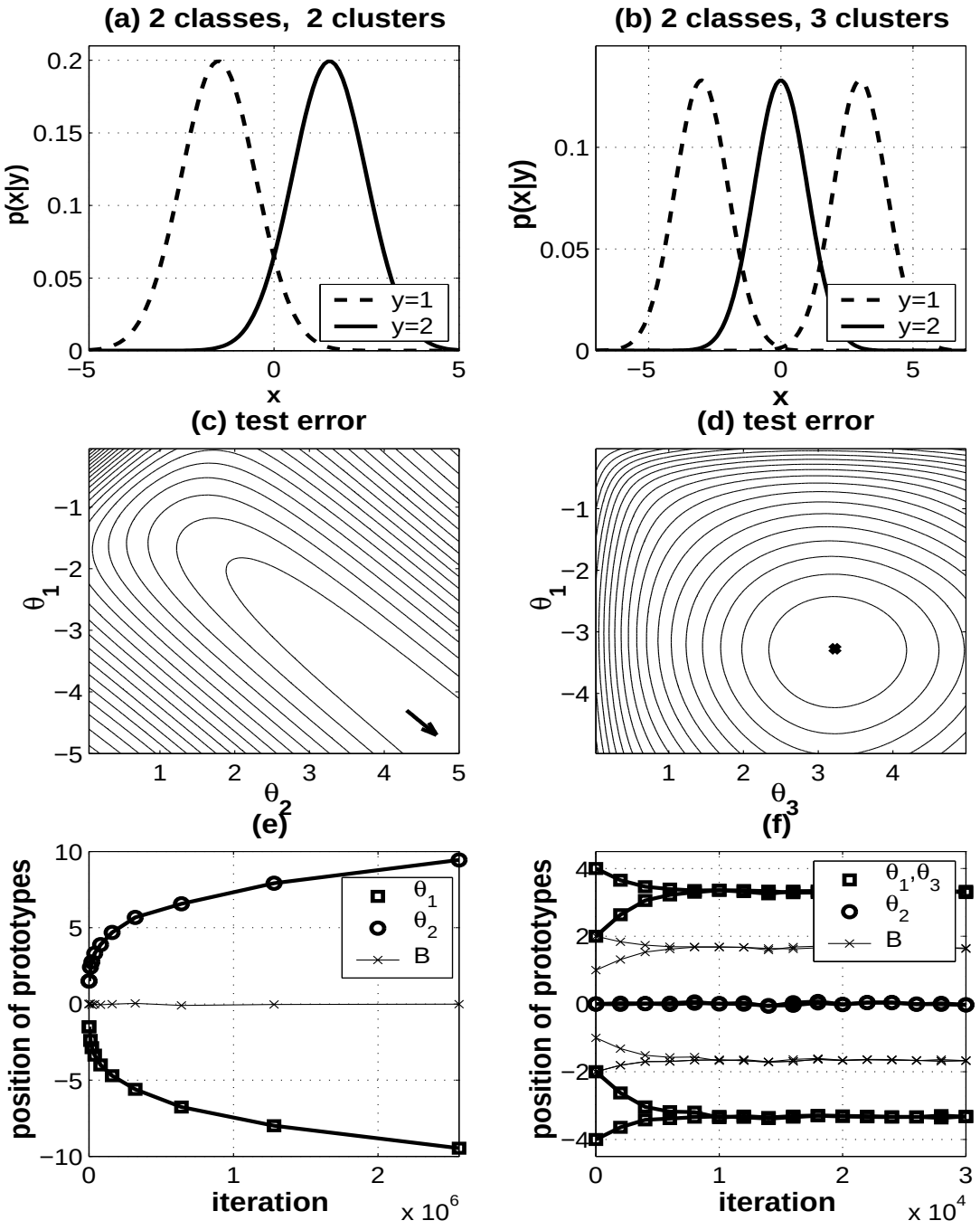


Figure 3.3: SNPC applied to two 1D toy problems.

(a) Problem no. 1: Data points are drawn iid from two Gaussian distributions, one for each class (dashed and solid line for class 1 and 2). Parameters are: location $\mu_1 = -1.5$ (class 1), $\mu_2 = 1.5$ (class2), width $\sigma_d = 1$. (b) Problem no. 2: Data points are drawn iid from three Gaussian distributions, two for class 1 (dashed line) and one for class 2 (solid line). Parameters are: location $\mu_1 = -3$, $\mu_3 = 3$ (class 1), $\mu_2 = 0$ (class 2), width $\sigma_d = 1$. The classifier consists of three prototypes, two for class 1 (θ_1, θ_3) and one for class 2 (θ_2). (c) Test error, eq. (3.20), as a function of the location θ_1 and θ_2

of the prototypes of problem no. 1. Contour lines indicate steps of 20. The minimum is located towards the lower right corner (see arrow). (d) Test error, eq. (3.20), as a function of the location θ_1 and θ_3 of the prototypes 1 and 3 of problem no. 2. θ_2 was set to 0. Contour lines indicate step of 20. The minimum is indicated by the dot. (e) Trajectories of the prototypes θ_1 and θ_2 during learning with SNPC for constant value of the width parameter. The thin line and the crosses indicate the classification boundary for hard classification, eq. (2.8). Parameters are: Initialization: $(\theta_1^{\text{ini}}, c_1) = (-2, 1), (\theta_2^{\text{ini}}, c_2) = (2, 1), \sigma^2 = 1$. 10000 and 1000 data points per class were drawn iid from the distributions shown in (a) for the training and test sets, respectively. The learning rate was set to $\alpha = 0.01$. (f) Trajectories of the prototypes $\theta_1, \theta_2, \theta_3$ during learning with SNPC for constant value of the width parameter and for two different prototype initializations. The thin line and the crosses indicate the classification boundary for hard classification, eq. (2.8). Parameters are: Initialization: $(\theta_1^{\text{ini}}, c_1) = (-4, 1), (\theta_2^{\text{ini}}, c_2) = (0, 2), (\theta_3^{\text{ini}}, c_3) = (4, 1), \sigma^2 = 1$ and $(\theta_1^{\text{ini}}, c_1) = (-2, 1), (\theta_2^{\text{ini}}, c_2) = (0, 2), (\theta_3^{\text{ini}}, c_3) = (2, 1), \sigma^2 = 1$. 10000 and 1000 data points per class were drawn iid from the distributions shown in (b) for the training and test sets, respectively. The learning rate was set to $\alpha = 0.01$.

$\sigma^2(t) = 0.02 \times 0.9^{(t-1)}$ and $\sigma_f^2 = 0.0005$. The learning rate α was changed according to $\alpha(t) = 0.1 \times \frac{6000}{6000+t}$. Figure 3.4 c shows the log of the 'soft' test error, eq. (3.20), and the log of the 'hard' test error, eq. (3.21), as a function of the log of the inverse of the width parameter σ^2. The test set contained $N_t = 4000$ data points (1000 data points drawn iid from each class). Figure 3.4 shows (*i*) that during the annealing process the generalization performance of the classifier is improved (cf. Fig. 3.4 c), (*ii*) that annealing optimizes the structure of the model (the number of prototypes is reduced from 8 to 5, cf. Fig. 3.4 b), and (*iii*) that there is an optimal value for the width parameter σ^2 (cf. arrow in Fig. 3.4 (c)). The latter can be understood as follows: For large values of σ^2 prototypes are located outside the data distribution (in order to minimize $E(\mathcal{S}, \mathcal{T})$). Therefore, the classification boundary is quite different from the optimal boundary of a Bayesian classifier, and the values for E_h^T remain large. If, however, σ^2 becomes too small, only very few data points are located within the window zone and over-fitting due to noise increases the generalization error again.

3.5 Summary

In this chapter I investigated a principled approach to Learning Vector Quantization. Starting from a cost function which can be interpreted as the rate of misclassification for the case of fuzzy assignments of data points to prototypes, I derive a learning algorithm for those prototypes using stochastic gradient descent. The principled approach provides an explanation of several features of the (heuristic) LVQ methods, including the divergence of prototypes and the role of the window region. Because of the emergence of a window region in SNPC, active learning strategies can be used to minimize the number of labeled data points necessary for learning. SNPC is more robust than standard LVQ with respect to different initial conditions of the prototypes.

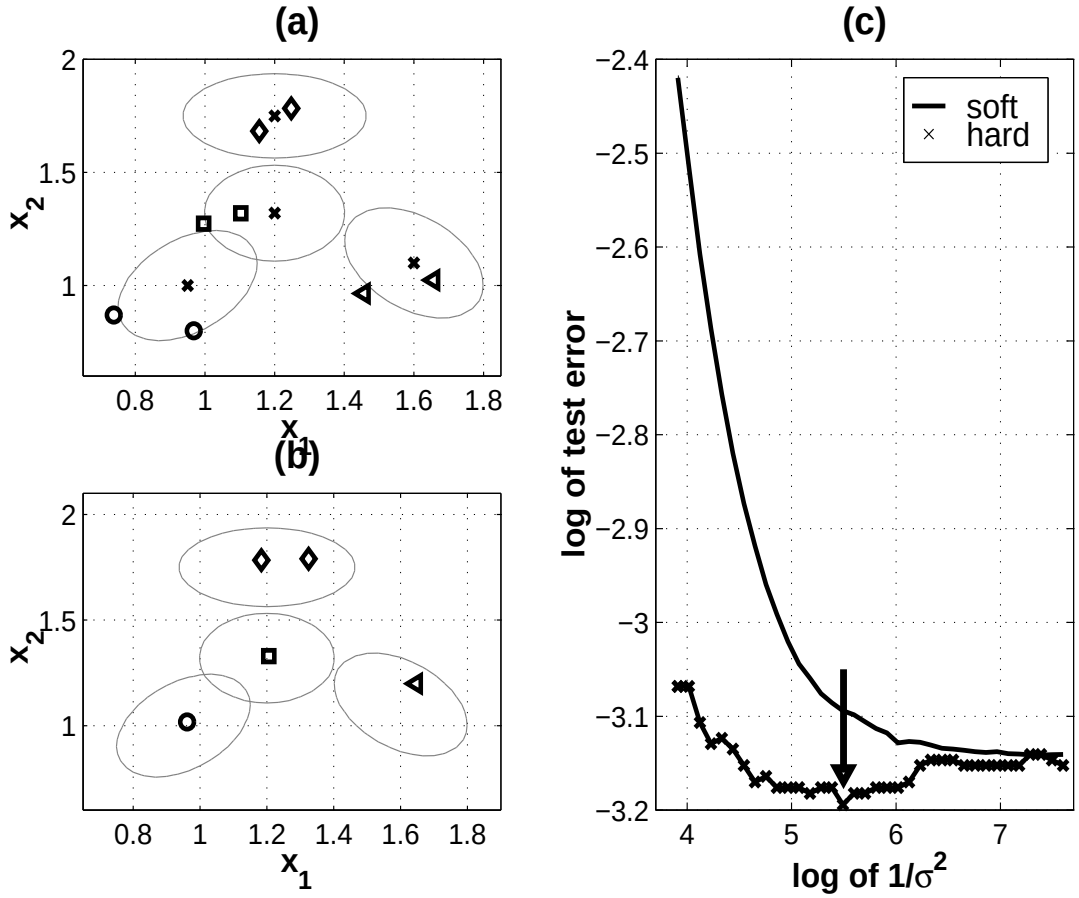


Figure 3.4: SNPC with annealing applied to a 2D toy problem.
For training, 300 data points are drawn iid from each of four Gaussian components (parameters: $\mu_1 = (0.95, 1), \mu_2 = (1.2, 1.32)$, $\mu_3 = (1.6, 1.1), \mu_4 = (1.2, 1.75), \Sigma_{11} = (0.01, 0.005), \Sigma_{12} = (0.005, 0.015)$, $\Sigma_{21} = (0.01, 0)$, $\Sigma_{22} = (0, 0.011)$, $\Sigma_{31} = (0.01, -0.005)$, $\Sigma_{32} = (-0.005, 0.015)$, $\Sigma_{41} = (0.02, 0)$, $\Sigma_{42} = (0, 0.01)$). The classifier consists of 8 prototypes, 2 for each class. (a) Initial values of the prototypes of the classifier (open symbols, class is indicated by symbol type). Crosses indicate the centers of the four Gaussian components of the data distribution, the ellipses indicate the widths in every direction as given by the covariance matrix Σ. (b) Final values of prototypes determined by SNPC with annealing and for an optimal σ_f^2 (cf. arrow Fig. 3.4 c). Note, that the two prototypes of the classes $\circ, \square$ and $\triangleleft$ lie on top of each other. (c) Log of the test errors E_s^T and E_h^T as a function of $\log 1/\sigma^2$ for SNPC with annealing. Parameters: For training 300 data points per class are drawn iid from the Gaussian distribution, schedule for learning rate: $\alpha(t+1) = 0.1 \times \frac{1200}{1200+t}$; schedule for σ^2-annealing: $\sigma^2(t) = 0.02 \times 0.9^{(t-1)}$, $\sigma_f^2 = 0.0005$.

Chapter 4

Soft LVQ Based on Likelihood Ratio (SLVQ-LR)

Learning Vector Quantization is a class of algorithms for the determination of prototype vectors for NPC, which make use of the distribution of the training data points as well as their class labels when selecting a useful set of prototypes. In this chapter I introduce a variant of the LVQ method using a principled approach, which is published in Seo and Obermayer (2003). I first make the ansatz, that the probability density of the data is well described by one Gaussian mixture model for each class. Then, I consider the probability density that a data point x is generated by the Gaussian mixture model of the correct class and compare it to the probability density that this data point is generated by the Gaussian mixture models of the incorrect classes. The logarithm of the ratio of the 'correct' vs. the 'incorrect' probability densities serves as a cost-function and is maximized. The choice of cost-function is motivated by the fact that the learning rule of LVQ leads to both attraction and repulsion of prototypes, depending on the class labels (cf. eqs. (2.10)). The 'attractive' term is a result of maximizing the log probability that a data point is generated by the 'correct' class, while the 'repulsive' term is a consequence of minimizing the log probability, that the data point is generated by the classes with the 'incorrect' label. It is shown analytically that the proposed cost function leads to the divergence of the prototypes. In order to avoid this divergence, I propose to use a window rule, which is the same as for LVQ 2.1, except that the constraint that exactly one label of the two nearest prototypes must match the label of the data point is relaxed to the constraint that both the labels must be different from each other. The hard version (i.e. $\sigma^2 \to 0$) of soft LVQ is described in section 4.4, and exactly corresponds to LVQ 2.1, except for the slightly modified window rule. Thus LVQ 2.1 is minimizing the proposed cost function in the limit $\sigma^2 \to 0$.

4.1 Cost-Function and Learning Rule: General Case

We are given a training data set $\mathcal{S}$, eq. (2.1), and a NP classifier consisting of a set of labeled prototypes $\mathcal{T}$, eq. (2.7). Our goal is to construct an optimal learning algorithm for NPC. We use the mixture ansatz, eq. (2.12), and the definition of the two conditional probability densities, eq. (2.13). $p(x, y|\mathcal{T})$ is the probability density that a data point x is generated by the mixture model for the 'correct' class, i.e. the class denoted by

the label y. $p(x, \bar{y}|\mathcal{T})$ is the probability density that a data point x is generated by the mixture models for the 'incorrect' classes, i.e. the classes denoted by labels $\neq y$. Now we consider the two likelihoods:

$$L_c(\mathcal{S}|\mathcal{T}) \;=\; \prod_{k=1}^{N} p(x_k, y_k|\mathcal{T}), \qquad L_u(\mathcal{S}|\mathcal{T}) \;=\; \prod_{k=1}^{N} p(x_k, \bar{y}_k|\mathcal{T}). \tag{4.1}$$

The mixture model, eq. (2.12), is a good model of the data if the likelihood functions $L_c(\mathcal{S}|\mathcal{T})$ and $L_u(\mathcal{S}|\mathcal{T})$ are maximal and minimal, respectively. A natural approach is to maximize the likelihood ratio

$$L_r(\mathcal{S}|\mathcal{T}) \;=\; \prod_{k=1}^{N} \frac{p(x_k, y_k|\mathcal{T})}{p(x_k, \bar{y}_k|\mathcal{T})} \;\stackrel{!}{=}\; \max, \tag{4.2}$$

w.r.t. the parameters of the model densities. This leads to a classifier which maximizes the rate of correct classification and at the same time minimizes the rate of misclassification. For computational simplicity we maximize $\log L_r$,

$$\log L_r(\mathcal{S}|\mathcal{T}) \;=\; \sum_{k=1}^{N} \log \frac{p(x_k, y_k|\mathcal{T})}{p(x_k, \bar{y}_k|\mathcal{T})} \;\stackrel{!}{=}\; \max\,. \tag{4.3}$$

Using stochastic gradient ascent (Robbins and Monro (1951)) we obtain the learning rule

$$\theta_l(t+1) \;=\; \theta_l(t) \;+\; \alpha(t) \frac{\partial}{\partial \theta_l} \log \frac{p(x, y|\mathcal{T})}{p(x, \bar{y}|\mathcal{T})}, \tag{4.4}$$

where t is the iteration number, and $\alpha(t)$ is the learning rate. In order to ensure convergence, $\alpha(t)$ must fulfill the conditions $\sum_{t=0}^{\infty} \alpha(t) = \infty$ and $\sum_{t=0}^{\infty} \alpha^2(t) < \infty$ (Robbins and Monro (1951)).

If the conditional density functions $p(x|l)$ are of the normalized exponential form

$$p(x|l) \;=\; K(l) \exp f(x, \theta_l), \tag{4.5}$$

we obtain (see appendix A.2)

$$\begin{aligned} \frac{\partial}{\partial \theta_l} \log \frac{p(x, y|\mathcal{T})}{p(x, \bar{y}|\mathcal{T})} \;&=\; \delta(c_l \;==\; y) P_y(l|x) \frac{\partial f(x, \theta_l)}{\partial \theta_l} \\ &\quad - \delta(c_l \neq y) P_{\bar{y}}(l|x) \frac{\partial f(x, \theta_l)}{\partial \theta_l}, \end{aligned} \tag{4.6}$$

where $\delta(x) = 1$ if x is true and zero else. $K(l)$ is the normalization constant and $P_y(l|x)$ and $P_{\bar{y}}(l|x)$ are given by

$$\begin{aligned} P_y(l|x) \;&=\; \frac{P(l)\ \exp f(x, \theta_l)}{\sum_{\{l:c_l=y\}} P(l)\ \exp f(x, \theta_l)}, \\ P_{\bar{y}}(l|x) \;&=\; \frac{p(l)\ \exp f(x, \theta_l)}{\sum_{\{l:c_l\neq y\}} p(l)\ \exp f(x, \theta_l)}. \end{aligned} \tag{4.7}$$

$P_y(l|x)$ and $P_{\bar{y}}(l|x)$ are assignment probabilities. $P_y(l|x)$ describes the (posterior) probability that the data point x is assigned to the component l of the mixture, given that the data point was generated by the correct class. $P_{\bar{y}}(l|x)$ describes the (posterior) probability that the data point x is assigned to the component l of the mixture, given that the data point was generated by the incorrect classes. Using the gradient given in eq. (4.6) we obtain the following learning rule

$$\theta_l(t+1) \;=\; \theta_l(t) \;+\; \alpha(t) \begin{cases} P_y(l|x) \left[\frac{\partial f(x,\theta_l)}{\partial \theta_l}\right], & \text{if } c_l = y, \\ -P_{\bar{y}}(l|x) \left[\frac{\partial f(x,\theta_l)}{\partial \theta_l}\right], & \text{if } c_l \neq y. \end{cases} \tag{4.8}$$

The log of the likelihood ratio (4.3) can also be optimized using standard gradient ascent,

$$\theta_l^{(new)} = \theta_l^{(old)} + \eta \sum_{k=1}^{N} \frac{\partial}{\partial \theta_l} \log \frac{p(x_k, y_k|\mathcal{T})}{p(x_k, \bar{y}_k|\mathcal{T})}, \tag{4.9}$$

where η is the learning rate. This optimization algorithm can be considered as the batch version of our on-line method, eq (4.8), which follows the idea of stochastic approximation. Batch learning shows faster convergence. On-line learning however is more robust against a bad choice of the initialization of parameters, w.r.t. an escape from shallow local optima of the cost function, and may be used for on-line adaptation in non-stationary environments. In the following I will, therefore, consider the on-line learning rule eq. (4.8).

4.2 Cost-Function and Learning Rule: Gaussian Mixture Ansatz

Let us now consider a Gaussian mixture with

$$K(l) \;=\; (2\pi\sigma_l^2)^{-D/2}, \qquad f(x,\theta_l) = -\frac{(x-\theta_l)^2}{2\sigma_l^2}.$$

Because classification using NPCs depends only upon the relative distances between data points and prototypes, we assume that every component has the same width and strength, i.e. $\sigma_l^2 = \sigma^2$ and $p(l) = \frac{1}{M}, \quad \forall\, l = 1, \ldots, M$. Eq. (4.8) then becomes

$$\theta_l(t+1) \;=\; \theta_l(t) \;+\; \alpha^*(t) \begin{cases} P_y(l|x)(x-\theta_l), & \text{if } c_l = y, \\ -P_{\bar{y}}(l|x)(x-\theta_l), & \text{if } c_l \neq y, \end{cases} \tag{4.10}$$

with $\alpha^*(t) = \frac{\alpha(t)}{\sigma^2}$ and

$$\begin{aligned} P_y(l|x) \;&=\; \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{m:c_m=y\}} \exp\left(-\frac{(x-\theta_m)^2}{2\sigma^2}\right)}, \\ P_{\bar{y}}(l|x) \;&=\; \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{m:c_m\neq y\}} \exp\left(-\frac{(x-\theta_m)^2}{2\sigma^2}\right)}. \end{aligned} \tag{4.11}$$

The learning rule, eq. (4.10), shows that the prototypes with the 'correct' labels are attracted towards a data point x proportionally to their distance to x and weighted by a factor $P_y(l|x)$. 'Incorrect' prototypes are repelled from the data point x, and the strength of the repulsion is again proportional to their distance to x and is weighted by a factor $P_{\bar{y}}(l|x)$.

Fig. 4.1 illustrates the loss function and the factor for the update of the prototypes, eqs. (4.10) and (4.11). Figures (a,b) show the loss function,

$$\mathrm{ls}(x,\mathcal{T}) \;=\; -\log\frac{p(x,y|\mathcal{T})}{p(x,\bar{y}|\mathcal{T})} \;=\; -\log\left[\frac{\sum_{\{l:c_l=y\}}\exp\left(\frac{-(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{m:c_m\neq y\}}\exp\left(-\frac{(x-\theta_m)^2}{2\sigma^2}\right)}\right] \tag{4.12}$$

for $\sigma^2 = 0.5$ and $\sigma^2 = 2$ as function of the location of the data points. For given σ^2, the loss function depends on the distance between data point and class boundary. Figures (c,d) and (e,f) show the update factors, eqs. (4.10) and (4.11), for the nearest correct and incorrect prototypes, respectively. The left side of the figure illustrates the learning rule for the hard version of the algorithm for small σ^2 (see sec. 4.4). In this case, each data point adjusts its nearest correct and incorrect prototypes with factor 1. The other prototypes are not changed. But for the data points which lie midway between the two correct prototypes or the two nearest incorrect prototypes, the factor is 0.5. The right figure shows the case with large width parameter and illustrates the typical dynamics of the SLVQ-LR algorithm. Given a data point, the update factor for the correct (incorrect) prototypes, fig. 4.1d(f), results from a competition among the prototypes with correct (incorrect) labels, as can be seen from eqs. (4.11). For fixed σ^2, this factor approximates a winner-takes-all manner, if the distance among the prototypes with correct (incorrect) labels increases.

4.3 The Window Rule

The expectation of the cost-function eq. (4.3) w.r.t. the true distribution $p(x,y)$ of the data is given by

$$\begin{aligned}
E[\log L_r] &= \sum_{i=1}^{N_y}\int dx\; p(x,y_i)\log\frac{p(x,y_i|\mathcal{T})}{p(x,\bar{y}_i|\mathcal{T})} \\
&= \sum_{i=1}^{N_y}\int dx\; p(x,y_i)\log\left[\frac{p(x,y_i|\mathcal{T})}{p(x,\bar{y}_i|\mathcal{T})}\cdot\frac{p(x,y_i)}{p(x,y_i)}\right] \\
&= \sum_{i=1}^{N_y}\int dx\; p(x,y_i)\log\frac{p(x,y_i)}{p(x,\bar{y}_i|\mathcal{T})} - \sum_{i=1}^{N_y}\int dx\; p(x,y_i)\log\frac{p(x,y_i)}{p(x,y_i|\mathcal{T})} \\
&= D_{KL}(p(x,y)||p(x,\bar{y}|\mathcal{T})) - D_{KL}(p(x,y)||p(x,y|\mathcal{T})).
\end{aligned} \tag{4.13}$$

Maximization of the cost-function in the limit of a large number of data points is therefore equivalent to (i) the maximization of the Kullback-Leibler divergence between the true distribution $p(x,y)$ and the data distribution $p(x,\bar{y}|\mathcal{T})$ which is generated by the 'incorrect' classes and (ii) the minimization of the Kullback-Leibler divergence between the true distribution $p(x,y)$ and the data distribution $p(x,y|\mathcal{T})$ which is generated by

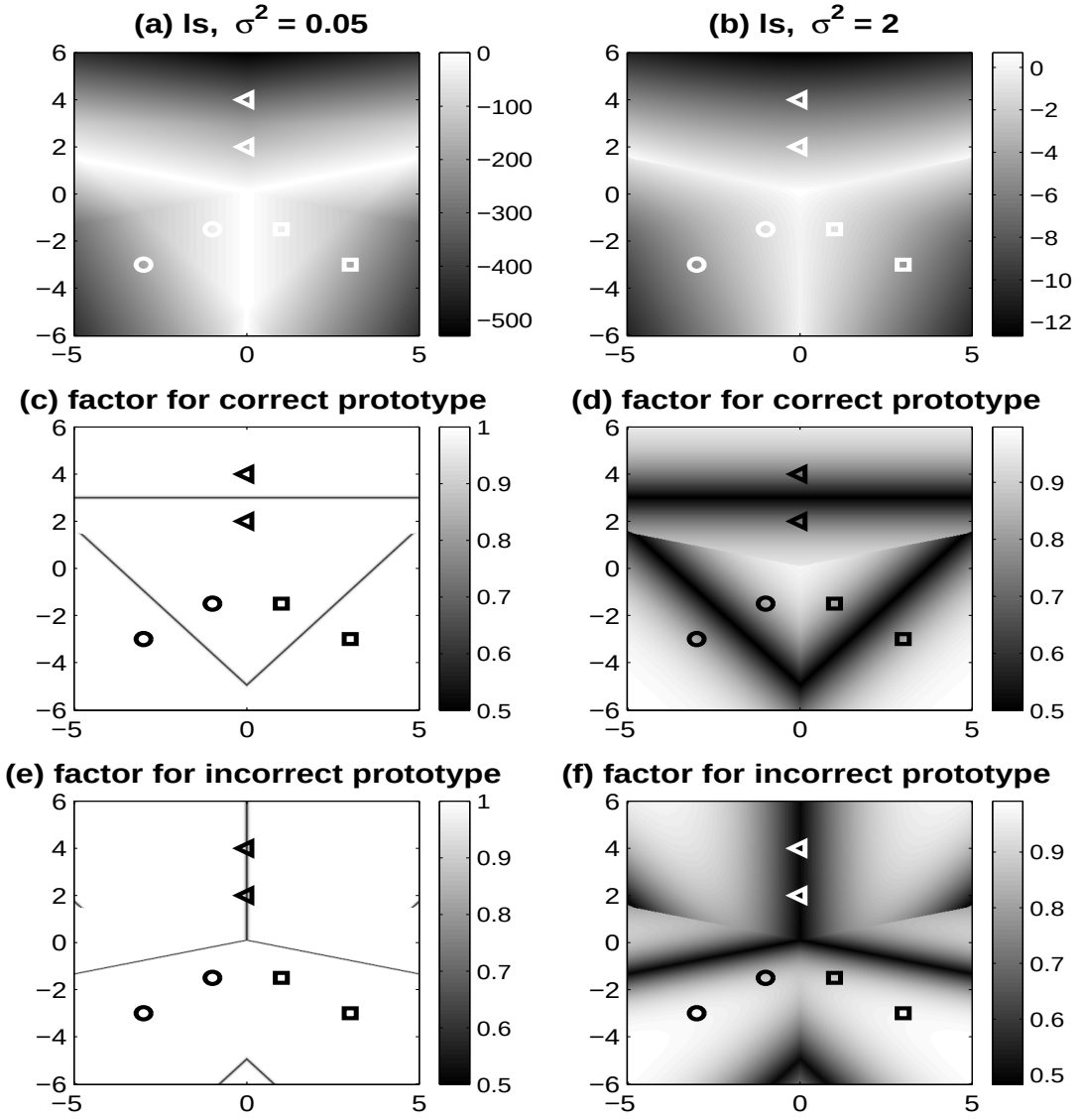


Figure 4.1: Loss and update factor of the SLVQ-LR algorithm for different σ^2. The figure shows the individual loss of SLVQ-LR, eq. (4.12), for different width parameters, $\sigma^2 = 0.05$ (left column) and $\sigma^2 = 2$ (right column), and the update factors $P_y(l|x)$ and $P_{\bar{y}}(l|x)$, eq.(4.11), of the nearest prototypes with correct label (fig. (c,d)) and with incorrect label (fig. (e,f)), respectively. Here, a three class problem in 2D space with an NP classifier having two prototype per class is constructed. The x and y axis denote the location of the data points, and the symbols $\circ$, $\triangleleft$ and $\square$ denote the position of prototypes belonging to different classes; $\theta_\circ \in \{(-3,-3),(-1,-1.5)\}$, $\theta_\triangleleft \in \{(0,2),(0,4)\}$, $\theta_\square \in \{(1,-1.5),(3,-3)\}$. It is assumed that every data point is classified correctly by the current prototypes, i.e. all data points have the same label as their nearest prototype. For details see the text.

1. **set**
   - the value of the width parameter σ^2
   - the value for the window parameter ω
   - number of prototypes for each class
   - initial value α_0 and annealing schedule for learning rate
2. **initialize** prototypes $\{(\theta_k, c_k)\}_{k=1}^{M}$,
3. **repeat**
   (a) draw a data point from the training data set
   (b) **if** the data point falls into a window **then**
      i. calculate assignment probabilities (eq. (4.11))
      ii. update prototypes (eq. (4.10))
   (c) update learning rate

   until a stopping criterion is fulfilled

Figure 4.2: Summary of the SLVQ-LR method.

the 'correct' classes. Minimizing the second term forces the model to optimally represent the true data distribution while maximizing the first term ensures an optimal class boundary. Maximizing the first term, however, may lead to the divergence of the prototype vectors θ_l because (i) the Kullback-Leibler divergence has no upper bound and (ii) the Kullback-Leibler divergence between two Gaussian distributions increases with increasing distance between their centers.

In order to prevent the divergence of the prototypes we introduce a 'window rule' similar to the 'window rule' eq. (2.11) used in LVQ 2.1,

$$\min\left(\frac{\|x-\theta_m\|_2}{\|x-\theta_l\|_2}, \frac{\|x-\theta_l\|_2}{\|x-\theta_m\|_2}\right) > s, \quad \text{where} \quad s = \frac{1-\omega}{1+\omega}. \tag{4.14}$$

θ_l and θ_m are the two prototypes which are closest to the data point x and which have different labels, $c_l \neq c_m$. $0 < w \leq 1$ is the width parameter which determines the window size. According to this window rule, for small ω the prototypes are only modified if a data point lies near the actual class boundary. The learning rule, eq. (4.10), together with the window rule, eq. (4.14), implement the Soft Learning Vector Quantization based on Likelihood Ratios (SLVQ-LR) algorithm. Here, in contrast to the window rule of LVQ 2.1, the constraint that exactly one of the two nearest prototypes must match the label of the data point is dropped. Instead, only the inequality of the class label of the two nearest prototypes is required. Fig. 4.2 summarizes the SLVQ-LR learning algorithm for fixed σ^2 and ω. σ^2 is a hyper-parameter of the learning rule (4.11), which, as in section 3.2, can either be chosen by hyper parameter validation or

be annealed.

4.4 Hard Classification: The Winner-Takes-All Case

If the width σ of the Gaussian components goes to zero, both assignment probabilities, eq. (4.11), become hard assignments, i.e.

$$\begin{aligned} P_y(l|x) &= \delta(l == q_y), \qquad q_y = \arg\min_{\{k:c_k=y\}} ||x-\theta_k||, \\ P_{\bar{y}}(l|x) &= \delta(l == q_{\bar{y}}), \qquad q_{\bar{y}} = \arg\min_{\{k:c_k\neq y\}} ||x-\theta_k||. \end{aligned}$$

If the two nearest prototypes for a given data point (x, y) have different class labels and x falls into the window, eq. (4.14), then the nearest prototype in the set of prototypes with 'correct' label y and the nearest prototype in the set of prototypes with 'incorrect' label $\neq y$ will be modified according to

$$\theta_l(t+1) = \theta_l(t) + \alpha(t) \begin{cases} (x-\theta_l), & \text{if } c_l = q_y, \\ -(x-\theta_l), & \text{if } c_l = q_{\bar{y}}, \\ 0, & \text{else.} \end{cases} \tag{4.15}$$

This algorithm will be referred to as LVQ-LR afterwards. LVQ-LR is similar to LVQ 2.1 with the only exception that LVQ 2.1 requires one of the two nearest prototypes to have the same class label as the data point x. The dynamics of LVQ 2.1 minimize the limit case ($\sigma^2 \to 0$) of the likelihood ratio, eq. (4.2), with the assumed Gaussian mixture model. Its window rule is a heuristically chosen method to prevent the divergence of the algorithm. We will see in chapter 7 that this window rule is not the best one. LVQ-LR, which is the LVQ 2.1 algorithm with the slightly modified window rule, performs much better than LVQ 2.1.

4.5 Summary

In this chapter, a learning rule for NPC is proposed which is based on the maximization of the likelihood ratio of a Gaussian mixture model. The learning algorithm is derived from the cost function using stochastic gradient ascent. For a given data point (x, y), all of the prototypes with correct label $= y$ are attracted to x, and all of the prototypes with incorrect labels $\neq y$ are repelled from x. This learning rule is called soft learning vector quantization based on the likelihood ratio (SLVQ-LR). In the limit case $\sigma^2 \to 0$ the learning factors take on the values 0 or 1, and the assignment works according to the so-called winner-takes-all principle. For a given data point (x, y) the nearest prototype with correct label will be attracted to x, and the nearest prototype with incorrect label will be repelled from x. The dynamics are equivalent to the learning rule of LVQ 2.1. This shows that the learning rule of LVQ 2.1 maximizes the likelihood ratio of a Gaussian mixture model. The prototypes will diverge, if they are optimized using SLVQ-LR or LVQ-LR. The convergence properties of both learning algorithms are analyzed based on the expected value of the cost function. In order to solve the divergence problem, I introduced a window rule, which is the same as in LVQ 2.1, except for the condition that exactly one of the two nearest prototypes must have

the same label as the data point. By the derivation and analysis of the new learning algorithm, I can give an analytical explanation for LVQ 2.1:

- LVQ 2.1 maximizes the likelihood ratio of a Gaussian mixture model
- the divergence of the prototypes of LVQ 2.1 results from the corresponding cost function
- as we will see in chapter 7, the window rule of LVQ 2.1 is not the best choice.

In contrast to LVQ 2.1 they were derived by optimizing a cost-function. Hence the principled approach has several benefits:

- It provides insight into why prototypes tend to diverge when LVQ is used to generate an NPC.
- It provides a way to extend the LVQ family to different distance measures (cf. eq. (4.8)), and it relates these distance measures to assumptions about the components of the mixture model.
- It provides a way to extend the LVQ family to soft assignments (eq. (4.10)).
- It provides better classification results, as we will see in chapter 7.

Note, however, that the window rule is still heuristic and introduces a further hyper parameter ω which must be optimized.

Chapter 5

Robust Soft LVQ (RSLVQ)

In the previous chapter a novel NPC algorithm, the so-called SLVQ-LR, was introduced, and also a hard version of it was considered. The learning algorithm was derived by maximizing the likelihood ratio function which was constructed based on a generative Gaussian mixture model. The objective function of SLVQ-LR is not upper bounded, and the maximization of the likelihood ratio leads to the divergence of the model parameters, the prototypes. To assure the convergence of the algorithm, a window rule was introduced. In this chapter, I will introduce a novel variant of LVQ, using an upper bounded objective function based on a Gaussian mixture model. This objective function is the ratio of the likelihoods that the data is generated by the components of the Gaussian mixture model with the correct label and that it is generated by the Gaussian mixture model. By maximizing the upper bounded objective function a data-depended learning factor is derived, which selects informative data points and leads to the convergence of the algorithm. Due to the learning factor, the prototypes are adjusted only by the data points near the class boundary. The properties of the cost function and the learning rule will be considered.

5.1 Cost-Function and Learning Rule: General Case

We are given a training data set $\mathcal{S}$, eq. (2.1), and a NP classifier, a set of labeled prototypes $\mathcal{T}$, eq. (2.7). Using the mixture ansatz, eq. (2.12) and the definition of the two conditional probability densities, eq. (2.13), we can construct a new objective function to be maximized:

$$L_r \;=\; \prod_{k=1}^{N} \frac{p(x_k, y_k|\mathcal{T})}{p(x_k, y_k|\mathcal{T}) + p(x_k, \bar{y}_k|\mathcal{T})} \;=\; \prod_{k=1}^{N} \frac{p(x_k, y_k|\mathcal{T})}{p(x_k|\mathcal{T})} \;\stackrel{!}{=}\; \max\,. \tag{5.1}$$

The ratio $\frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})}$ is bounded from below by zero and - in contrast to the likelihood ratio used in eq. (4.2) - bounded from above by one. Because the ratio is continuous and bounded by $0 \leq \frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})} \leq 1$ with respect to all θ_l, the objective function, eq. (5.1), can be maximized by stochastic gradient ascent (Robbins and Monro (1951); Bottou

(1998)). For convenience, we can maximize the logarithm of the ratio L_r,

$$\log L_r \;=\; \sum_{k=1}^{N} \log \frac{p(x_k, y_k|\mathcal{T})}{p(x_k|\mathcal{T})} \;\stackrel{!}{=}\; \max, \tag{5.2}$$

instead of eq. (5.1). Stochastic gradient ascent then leads to the learning rule

$$\theta_l(t+1) \;=\; \theta_l(t) \;+\; \alpha(t) \frac{\partial}{\partial \theta_l} \left[\log \frac{p(x, y|\mathcal{T})}{p(x|\mathcal{T})} \right], \tag{5.3}$$

where $\alpha(t)$ is learning rate with $\sum_{t=1}^{\infty} \alpha(t) = \infty$ and $\sum_{t=1}^{\infty} \alpha^2(t) < \infty$. If the conditional density functions $p(x|l)$ are of the normalized exponential form

$$p(x|l) = K(l) \exp f(x, \theta_l), \tag{5.4}$$

the learning rule becomes (see appendix A.3)

$$\begin{aligned} \frac{\partial}{\partial \theta_l} \left[\log \frac{p(x, y|\mathcal{T})}{p(x|\mathcal{T})} \right] \;&=\; \delta(c_l = y)(P_y(l|x) - P(l|x)) \frac{\partial f(x, \theta_l)}{\partial \theta_l} \\ &\quad - \delta(c_l \neq y) P(l|x) \frac{\partial f(x, \theta_l)}{\partial \theta_l}, \end{aligned}$$

where $P_y(l|x)$ and $P(l|x)$ are assignment probabilities,

$$\begin{aligned} P_y(l|x) \;&=\; \frac{p(l) \exp f(x, \theta_l)}{\sum_{\{m:c_m = y\}} p(m) \exp f(x, \theta_m)}, \\ P(l|x) \;&=\; \frac{p(l) \exp f(x, \theta_l)}{\sum_{m=1}^{M} p(m) \exp f(x, \theta_m)}. \end{aligned} \tag{5.5}$$

$P_y(l|x)$ describes the (posterior) probability that the data point x is assigned to the component l of the mixture, given that the data point was generated by the correct class. $P(l|x)$ describes the (posterior) probability that the data point x is assigned to the component l of the complete mixture using all classes. Using the gradient given in the eq. (5.3) we obtain the learning rule

$$\theta_l(t+1) \;=\; \theta_l(t) \;+\; \alpha(t) \begin{cases} (P_y(l|x) - P(l|x)) \left[\frac{\partial f(x, \theta_l)}{\partial \theta_l} \right], & c_l = y, \\ -P(l|x) \left[\frac{\partial f(x, \theta_l)}{\partial \theta_l} \right], & c_l \neq y. \end{cases} \tag{5.6}$$

Note, that the factor $P_y(l|x) - P(l|x)$ is always positive.

5.2 Cost-Function and Learning Rule: Gaussian Mixture Ansatz

We again choose a Gaussian mixture model whose components have similar widths and strengths, i.e. $\sigma_l = \sigma, \forall l$, and $p(l) = \frac{1}{M}, l = 1, \ldots, M$. We then obtain

$$f(x, \theta_l) = \frac{-(x - \theta_l)^2}{2\sigma^2}, \qquad \frac{\partial}{\partial \theta_l} f(x, \theta_l) = \frac{1}{\sigma^2}(x - \theta_l). \tag{5.7}$$

1. **set**
   - σ^2 value of the width parameter
   - number of prototypes for each class
   - initial value α_0 and annealing schedule for learning rate
2. **initialize** prototypes $\{(\theta_k, c_k)\}_{k=1}^{M}$,
3. **repeat**
   (a) draw a data point from the training data set
   (b) calculate assignment probabilities (eq. (5.9))
   (c) update prototypes (eq. (5.8))
   (d) update learning rate

   until a stopping criterion is fulfilled,

Figure 5.1: Summary of the RSLVQ method.

By substituting the derivative of $f(x, \theta_l)$ into eq. (5.6) we obtain

$$\theta_l(t+1) \;=\; \theta_l(t) \;+\; \alpha^*(t) \begin{cases} (P_y(l|x) - P(l|x))(x - \theta_l), & c_l = y, \\ -P(l|x)(x - \theta_l), & c_l \neq y, \end{cases} \tag{5.8}$$

where $\alpha^*(t) = \frac{\alpha(t)}{\sigma^2}$ and

$$\begin{aligned} P_y(l|x) &= \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{m:c_m=y\}} \exp\left(-\frac{(x-\theta_m)^2}{2\sigma^2}\right)}, \\ P(l|x) &= \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{m=1}^{M} \exp\left(-\frac{(x-\theta_m)^2}{2\sigma^2}\right)}. \end{aligned} \tag{5.9}$$

The prototypes whose labels match the label of data point x will be attracted to it proportionally to the (always positive) factor $P_y(l|x) - P(l|x)$, while the prototypes whose labels differ from the label of data point x will be repelled from it proportionally to the factor $P(l|x)$. Eqs. (5.8),(5.9) implement the Robust Soft Learning Vector Quantization (RSLVQ) algorithm. Fig. 5.1 summarizes the RSLVQ learning algorithm for a fixed value of the width parameter σ^2 of the Gaussian components, which is a hyper-parameter of the learning rule (5.8). The RSLVQ algorithm can be optimized using deterministic annealing of σ^2.

Fig. 5.2 shows the loss function and the factors for the update of the prototypes, eqs. (5.8), (5.9), for two different width parameters $\sigma^2 = 0.5$ (left), $\sigma^2 = 4$ (right) on a

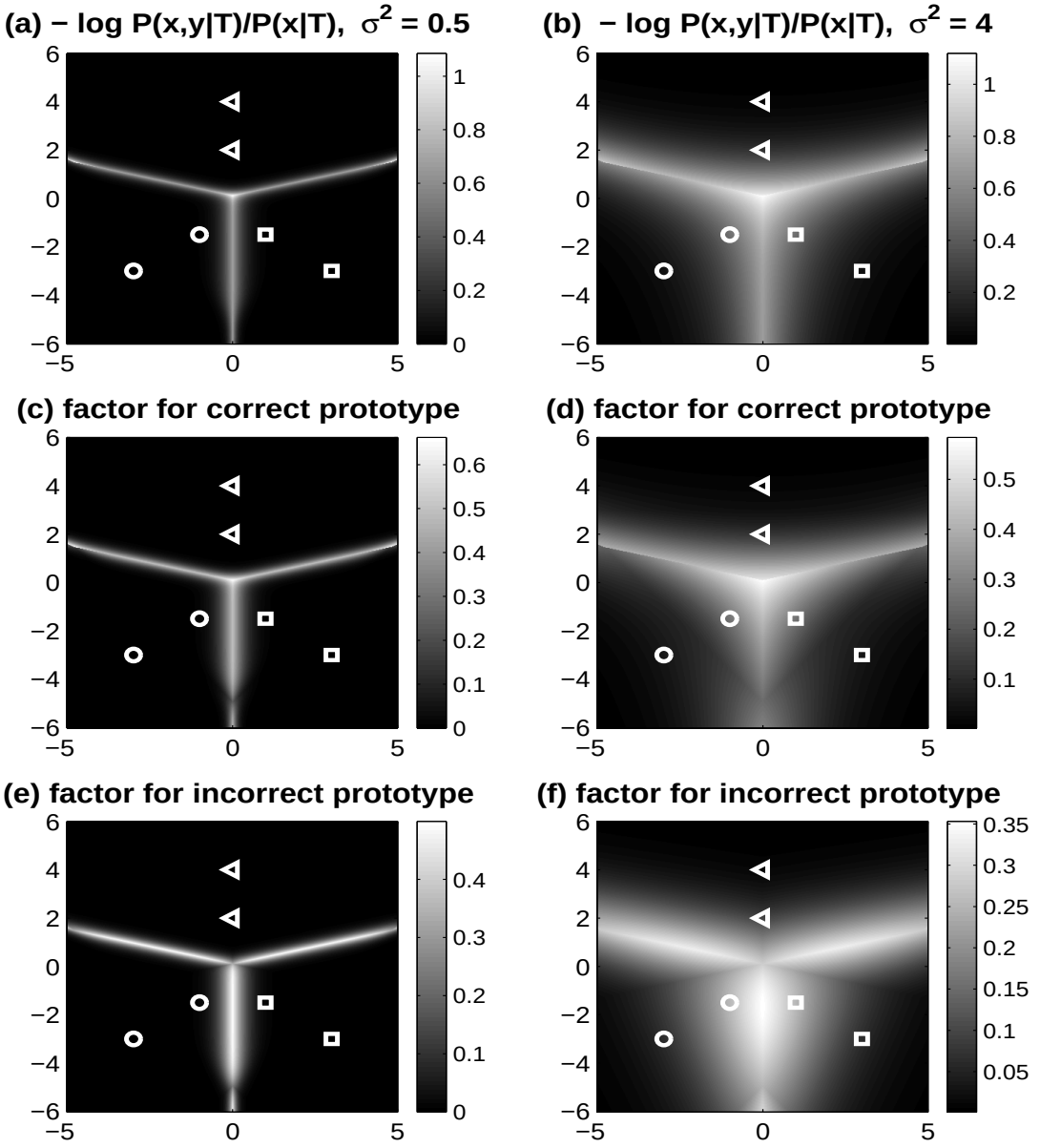


Figure 5.2: Loss and update factor of the RSLVQ algorithm for different σ^2 (I) A NPC with two prototypes per class. The width of the components, σ^2, increases from left ($\sigma^2 = 0.5$) to right ($\sigma^2 = 4$). The symbols $\circ, \triangleleft$ and $\square$ indicate the positions of the prototypes with different classes. The x and y axis denotes the location of the data points (x_1, x_2). It is assumed that the data points are correctly classified by the given NPC, i.e. the data points have the class label of the nearest prototype. Figures (a), (b) show the negative log of the ratio of probability densities $-\log \frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})}$, which is the loss function to be minimized. Figures (c)–(f) show the factor of update of the nearest prototype with correct and incorrect label, respectively (eq. (5.8)) as a function of the position of the data points. The active area for the update of the prototypes is around the class boundary and is expanded by increasing σ^2. The RSLVQ algorithm adjusts the prototypes only by using the data points near the class boundary.

2D toy example. Figures (a,b) show the loss function, the negative log of the ratio of the probability densities $-\log \frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})}$, as a function of the position of x:

$$\mathrm{ls}((x,y),\mathcal{T}) \;=\; -\log\left[\frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})}\right] \;=\; -\log\left[\frac{\sum_{\{l:c_l=y\}}\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{m=1}^{M}\exp\left(-\frac{(x-\theta_m)^2}{2\sigma^2}\right)}\right]. \quad (5.10)$$

Figures (c,d) show the factor for the update of the nearest correct prototype as a function of the position of the data points and figures (e,f) show the factor for the update of the nearest prototype with incorrect class label. The figure demonstrates that by increasing the width parameter σ^2, the update factors get smoother and the scale of the active region becomes larger. Furthermore, the RSLVQ algorithm changes the prototypes using the data points near the class boundary, and in contrast to SLVQ, outliers are not used for the adjustment of the prototypes (compare to fig. 4.1). This property shows that the RSLVQ algorithm implicitly selects informative data which is near the class boundary. It leads to the convergence of the prototypes without using a heuristic window rule.

5.3 Hard Classification

If the width σ of the Gaussian components goes to zero, both assignment probabilities, eq. (5.9), become hard assignments, i.e.

$$\begin{aligned} P_y(l|x) &= \delta(l == q_y), \qquad q_y = \arg\min_{\{k:c_k=y\}} ||x-\theta_k||, \\ P(l|x) &= \delta(l == q), \qquad q = \arg\min_{k} ||x-\theta_k||. \end{aligned}$$

If the label of the nearest prototype is equal to the label y of the data point x, then no update is performed, except for the case that the data point x lies on the class boundary, because

(i) x not on the class boundary:
$P(q_y|x) = 1, \qquad P(q_{\bar{y}}|x) = 0, \qquad P_y(q_y|x) - P(q_y|x) = 0.$

(ii) x on the class boundary:
$P(q_y|x) = 0.5, \qquad P(q_{\bar{y}}|x) = 0.5, \qquad P_y(q_y|x) - P(q_y|x) = 0.5,$

where $q_{\bar{y}}$ is the index of the nearest incorrect prototype of x. For case (ii) the nearest incorrect prototype and the nearest correct prototype are updated with probability 0.5. If the nearest prototype has a label which differs from the label of x, then

$$P(q_y|x) = 0, \qquad P(q_{\bar{y}}|x) = 1, \quad P_y(q_y|x) - P(q_y|x) = 1.$$

In this case the nearest incorrect prototype and the nearest correct prototype are updated with probability one. In the 'hard' version of RSLVQ, prototypes are modified only by the data points on the class boundary and by the data points which are not correctly classified.

Fig. 5.3 shows the strength of attraction and repulsion of correctly (left) and incorrectly (right) classified data for a classification problem with two classes (□ and

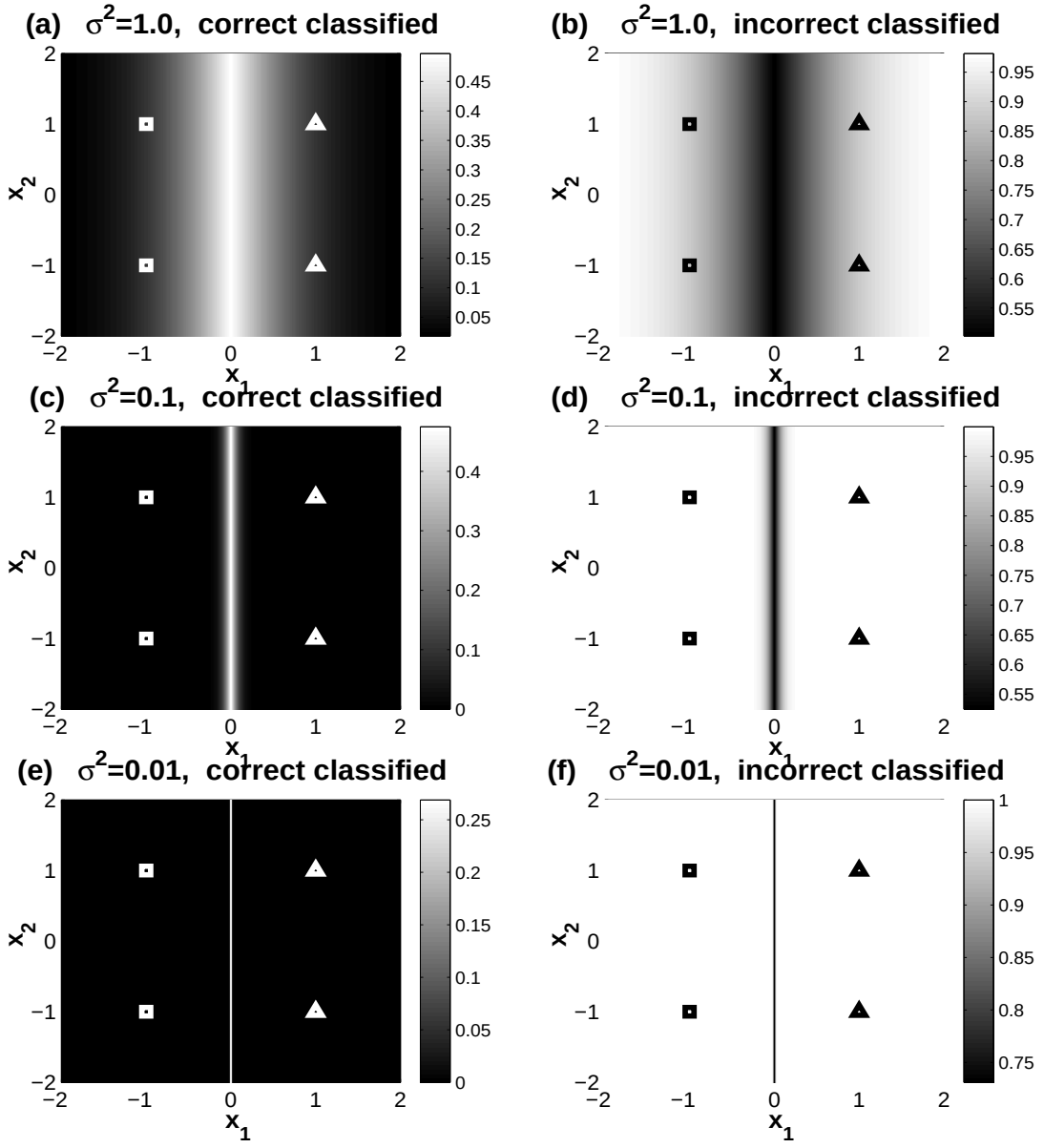


Figure 5.3: Update factor of the RSLVQ algorithm for different σ^2 (II)
The figure shows the strength of attraction and repulsion for a two-dimensional classification problem with two classes (□ and △). A NPC with two prototypes per class. The width σ decreases from top to bottom. The symbols □ and △ indicate the positions of the prototypes of the corresponding classes; the classification boundary is given by $x_1 = 0$. **left column:** The strength of the factors $\sum_{i=1}^{2} P_{\square}(\square_i|x) - P(\square_i|x)$ $(x_1 < 0)$ and $\sum_{i=1}^{2} P(\square_i|x)$ $(x_1 > 0)$ of correctly classified data points as a function of their location (x_1, x_2). **right column:** The strength of the factors $\sum_{i=1}^{2} P_{\square}(\square_i|x) - P(\square_i|x)$ (for $x_1 < 0$) and $\sum_{i=1}^{2} P(\square_i|x)$(for $x_1 > 0$) of incorrectly classified data points as a function of their location (x_1, x_2).

$\triangle$) and an NPC with two prototypes per class. The left column shows the values of $\sum_{i=1}^{2} P_{\square}(\square_i|x) - P(\square_i|x)$ (for $x_1 \leq 0$) and $\sum_{i=1}^{2} P(\square_i|x)$ (for $x_1 \geq 0$) of correctly classified data points as a function of their location (x_1, x_2). The smaller the parameter σ becomes, the smaller is the window around the classification boundary in which data points cause an update of prototypes. In the limit $\sigma \to 0$, no update is being performed by the correctly classified data points with the exception of the data points lying on the class boundary. The right column in fig. 5.3 shows the values of $\sum_{i=1}^{2} P_{\square}(\square_i|x) - P(\square_i|x)$ (for $x_1 \leq 0$) and $\sum_{i=1}^{2} P(\square_i|x)$ (for $x_1 \geq 0$) of incorrectly classified data points as a function of their location (x_1, x_2). This figure shows two things: (i) The smaller the parameter σ becomes, the stronger is the effect wrongly classified data points along the classification boundary have on learning. Hence, the candidates for σ^2 used in RSLVQ should not be very small. (ii) For the soft version, incorrectly classified outliers have a strong influence on the change of the prototypes (see fig. 5.3(b)). Under the assumption that the (noiseless) data set in principle is separable for a given number of prototypes, an additive noise term will lead to wrongly classified data points in the vicinity of the class boundary, even for the correct set of prototypes. However, under the assumption that there is no class label noise[1], data points farther away from the class boundary will usually not be effected. But for a reasonable set of prototypes, incorrectly classified data points are seldom found far away from their own class border. On the other hand, the strong influence of far-away data points leads to the effect that the prototypes quickly adjust to the correct data distribution, whenever they are located in a false region of the data distribution, e.g. due to bad initialization.

5.4 Summary

The RSLVQ algorithm was derived by maximizing the likelihood ratio using a stochastic gradient ascent method. The objective function (likelihood ratio) was constructed based on a Gaussian mixture model. Because the proposed likelihood ratio is upper bounded by 1, its maximization using a stochastic gradient method leads to the convergence of the prototypes. Note that RSLVQ does not require a 'window rule' because of this implicit selection of informative data in the learning factor. The influence of wrongly classified data points along the classification boundary on learning gets stronger by decreasing σ^2. For the limit case of RSLVQ ($\sigma^2 \to 0$), the prototypes are changed only by the data points on the class boundary and the data points which are incorrectly classified.

[1]The algorithms proposed in my thesis assume that the data distribution can be described by a Gaussian mixture model. Hence, they are not suited to deal with label noise. However, LVQ 2.1 can deal with label noise due to its window rule, because (i) only data points near the class boundary are considered and (ii) exactly one of the two nearest prototypes must match the label of the data point. Any data points far away from their own class border are filtered out. But the strong filtering implied by (ii) can have a negative effect on the performance of the learning algorithm, as we will see in chapter 7.

Chapter 6

Relation to Other Work

In this chapter, three distance based learning algorithms for nearest prototype classifiers are shortly reviewed, which are constructed based on three different objective functions. For all of these algorithms, first a quality measure for NPCs is constructed, which depends on the (squared) Euclidean distance between a data point (x, y) and the nearest prototype with correct label $(c_i = y)$ and the nearest prototype with incorrect label $(c_i \neq y)$. Then a (partially) smooth loss function is defined based on the predefined quality measure for NPCs, and the learning algorithm is derived by minimizing the empirical risk using a stochastic gradient descent method. Because the proposed loss functions for a data point depend only on the distance between the data point and its nearest correct and incorrect prototype, only these prototypes will be adjusted by a given data point. Furthermore, these algorithms optimize prototypes in the style of LVQ 2.1, i.e. the nearest correct prototype is attracted to the data point, while the nearest incorrect prototype is repelled from the data point, if the data point falls into an active area. The proposed cost functions have a window-parameter which determines the active region for learning. The three proposed loss functions are lower bounded and imply the selection of informative data points for the learning of the prototypes. Hence minimization of the cost function leads to an algorithm which uses only the data points in the critical area for the prototype update. The window rules are implicitly integrated in the bounded loss function. In my thesis I will call these three algorithms 'distance based LVQ' algorithms. The properties of the three distance based LVQ algorithms are discussed and they are compared to the properties of our novel learning algorithms derived based on a generative Gaussian mixture model.

6.1 Generalized Probabilistic Descent (GPD) methods

Katagiri et al. (Juang and Katagiri (1992); Katagiri et al. (1991); Komori and Katagiri (1992); McDermott and Katagiri (1994)) proposed a new formulation for the minimum error classification problem, together with a technique for designing a classifier that approaches the objective of minimum classification error in a direct manner. They constructed their classifiers using a two-step procedure, which they called the minimum classification error (MCE) and generalized probabilistic decent (GPD) methods. First, a parameterized set of discriminant functions $g(x, k|\mathcal{T})$ is constructed for each class k, where $\mathcal{T}$ denotes the set of parameters. Then, an individual loss $\mathrm{ls}(x|\mathcal{T})$ is

constructed for each data point x, based on its distance $\mu(x)$ from the discriminant function, such that (i) the loss is continuous and differentiable w.r.t. the parameters in $\mathcal{T}$ and (ii) the loss is monotonously related to the rate of misclassification obtained with hard classification using the final classifiers. Here I focus on the method described in McDermott and Katagiri (1994). Let $\mathcal{T} = \{\theta_l, c_l\}_{l=1}^{M}$, eq. (2.7), be a set of M labeled prototypes. Then

$$g(x, k|\mathcal{T}) = \left[\sum_{\{l:c_l=k\}} \|x - \theta_l\|^{-\xi} \right]^{-\frac{1}{\xi}}, \tag{6.1}$$

$$\mu(x) = g(x, y|\mathcal{T}) - \left[\frac{1}{N_y - 1} \sum_{j \neq y} g(x, j|\mathcal{T})^{-\zeta} \right]^{-\frac{1}{\zeta}} \tag{6.2}$$

$$\mathrm{ls}(x, \mathcal{T}) = \frac{1}{1 + \exp(-\eta\mu(x))} \tag{6.3}$$

where N_y is the number of classes and ξ, ζ, η are positive constants (the hyper-parameters of the method). The average loss over the training set $\mathcal{S}$, eq.(2.1):

$$E(\mathcal{S}, \mathcal{T}) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{ls}(x_i, \mathcal{T})$$

is then minimized using gradient descent methods. For large values of ξ and ζ the distance measure can be simplified to

$$\begin{aligned} \mu(x) &\cong g(x, y|\mathcal{T}) - g(x, y'|\mathcal{T}), \qquad (\zeta \to \infty) \\ &\cong \|x - \theta_{q_y}\| - \|x - \theta_{q_{\bar{y}}}\|, \qquad (\xi \to \infty) \end{aligned}$$

where y' is the class with the largest discriminant value among those classes other than class y, because $(N_y - 1)^{1/\infty} \cong 1$ and q_y and $q_{\bar{y}}$ are the indices of the nearest prototypes with correct and incorrect label respectively. The resulting classifier is equivalent to an NP classifier based on a Euclidean distance measure, and the following learning rule is obtained:

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) + \alpha^*(t)\Delta\theta_l(t) \\ \Delta\theta_l(t) &= \begin{cases} \mathrm{ls}(1 - \mathrm{ls})\frac{1}{\|x-\theta_l\|}(x - \theta_l), & \text{if} \quad l = q_y, \\ -\mathrm{ls}(1 - \mathrm{ls})\frac{1}{\|x-\theta_l\|}(x - \theta_l), & \text{if} \quad l = q_{\bar{y}}, \\ 0, & \text{else}, \end{cases} \end{aligned} \tag{6.4}$$

where $\alpha^*(t) = \eta \times \alpha(t)$ and ls is an abbreviation of $\mathrm{ls}(x, \mathcal{T})$. For convenience, we call this algorithm from now on GPD. The term $\frac{(x-\theta_l)}{\|x-\theta_l\|}$ is a direction vector of length one and leads to a change of prototypes which is independent of the distance between the data point and the prototypes and also independent of the scale of the data space. The normalization of the amount of change of the prototypes by one is too small for a data set with large variance $\sigma_x^2 \gg 1$ and results in very slow learning. It is too large for a data set with small variance $\sigma_x^2 \ll 1$ and the algorithm may diverge. Therefore, a scaling factor[1] is required as further hyper parameter in order to learn the

[1] In the following I will call this scaling factor γ.

prototypes effectively. Such a factor can be integrated in the learning rate α and should be optimized. In this case, the learning rate implies the scaling factor. Therefore, it may not fulfill the convergence condition of a learning rate that $\sum_t \alpha(t)^2 < \infty$. Equation (6.4) describes a LVQ learning rule; the window rule of LVQ 2.1 is implemented via the pre-factor $\mathrm{ls}(1-\mathrm{ls})$ which depends on the hyper parameter η and the distance from the data point to the class boundary. For this algorithm, there are two hyper parameters, a scaling parameter γ and a window parameter η, to be optimized, whereas LVQ 2.1 has only the window parameter ω to be optimized. In contrast to our three algorithms based on a generative Gaussian mixture model, (i) the factor for the prototypes with correct label and incorrect label are the same and (ii) only the two nearest prototypes are adjusted by a given data point and (iii) the amount of change of the prototypes does not depend on the distance between a data point and the prototypes, but only on the factor $\mathrm{ls}(1-\mathrm{ls})$ and the learning rate α. Figure 6.1 shows the loss function and the update factor for the prototypes, eq. (6.4), for two different window parameters $\eta = 1, 10$. This figure illustrates that the active area is near the class boundary and gets more and more confined to a narrow band along the class boundary with increasing η.

6.2 Generalized Learning Vector Quantization

Sato and Yamada (1996) suggested a learning algorithm for NPC based on minimization of a cost function. They first define the relative distance difference $\mu(x)$ as a quality measure for NPC and then the cost function as the sum of the individual loss $\mathrm{ls}(x, \mathcal{T})$ over all data points in the training data set $\mathcal{S}$, eq. (2.1):

$$E(\mathcal{S}, \mathcal{T}) \;=\; \sum_{i=1}^{N} \mathrm{ls}(x_i, \mathcal{T}), \qquad \mu(x) \;=\; \frac{d_1 - d_2}{d_1 + d_2}, \tag{6.5}$$

where $\mathrm{ls}(x, \mathcal{T})$ is a non-linear monotonically increasing function of $\mu(x)$ and d_1 and d_2 are the squared Euclidean distances between the data point x and the nearest correct and incorrect prototype, respectively. The relative distance difference $\mu(x)$ takes on values between -1 and 1. A negative $\mu(x)$ indicates that x is classified incorrectly and a positive $\mu(x)$ that x is classified correctly . Sato and Yamada (1996) suggested to use a Sigmoid function as a loss function and introduced a further so-called learning time parameter $\eta(t)$:

$$\mathrm{ls}(x, \mathcal{T}) = \frac{1}{1 + \exp(-\mu(x)\eta(t))}. \tag{6.6}$$

The hyper parameter $\eta(t)$ works as width parameter, like the η in the GPD learning algorithm, eq. (6.2). For small $\eta(t)$, the loss function is smooth and takes on a value between 0 and 1, while for large $\eta(t)$, the loss function gives zero-one loss. Hence it is reasonable that $\eta(t)$ is a monotonically increasing function of the learning time t. Note, that in the extended version of this algorithm, like in generalized relevance LVQ (Hammer and Villmann (2002)), the loss function is independent of the learning time t, $\mathrm{ls}(x, \mathcal{T}) = 1/(1 + \exp(-\mu(x)))$. I consider here the original suggestion in Sato

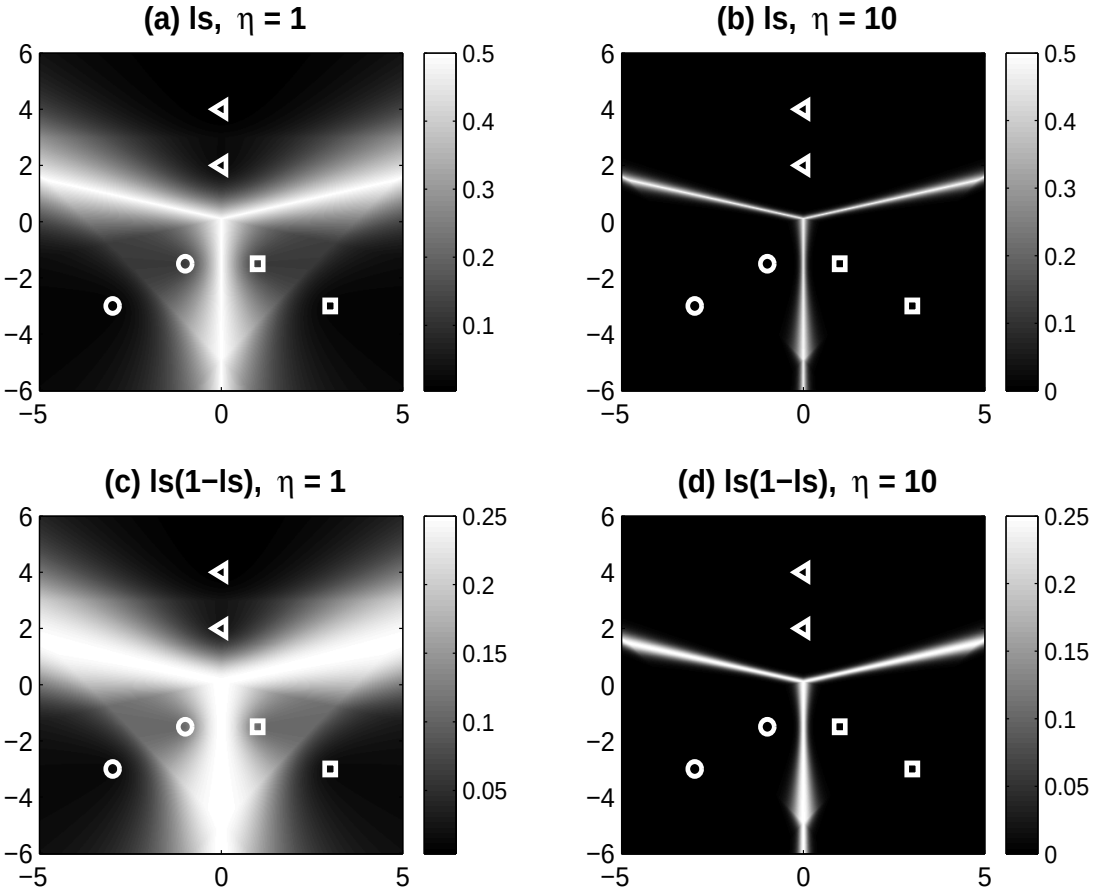


Figure 6.1: Loss and update factor of the GPD algorithm for different η.
A NPC with two prototypes per class. The symbols $\circ, \triangleleft$ and $\square$ indicate the positions of the prototypes of different classes; $\theta_\circ \in \{(-3,-3),(-1,-1.5)\}$, $\theta_\triangleleft \in \{(0,2),(0,4)\}$, $\theta_\square \in \{(1,-1.5),(3,-3)\}$. The x and y axes denote the location of the data points (x_1, x_2). It is assumed that the data points are correctly classified by the given NPC, i.e. the data points have the class label of the nearest prototype. The loss function (a,b), eq. 6.3 and the factor (c,d) for the update of the prototypes, eq. (6.4), are shown as a function of the position of the data point x for different window parameters, $\eta = 1,\ 10$. The figure demonstrates that the active area is near the class boundary and confines to a narrow region alongside the class boundary for increasing η.

and Yamada (1996)[2]. According to my personal experience, GLVQ with a learning time parameter works better than without. It furthermore has the interesting property that the learning time parameter controls the active area for learning and is changed during learning. This is different from the deterministic annealing method, because the learning time parameter increases in every learning step, while the annealing parameter in deterministic annealing is updated first after the convergence of the model parameter to a local extremum. By increasing d_1 and decreasing d_2, the value of the loss function grows, i.e. a data point lying farther from its nearest correct prototype and closer to its nearest incorrect prototype is the more informative for learning. The learning rule is obtained using the gradient descent method and is called Generalized LVQ (GLVQ) (for the derivation see appendix A.4.);

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) + \alpha\Delta\theta_l(t) \\ \Delta\theta_l(t) &= \begin{cases} \mathrm{ls}(1-\mathrm{ls})\frac{d_2}{(d_1+d_2)^2}(x-\theta_l), & \text{if } l = q_y, \\ -\mathrm{ls}(1-\mathrm{ls})\frac{d_1}{(d_1+d_2)^2}(x-\theta_l), & \text{if } l = q_{\bar{y}}, \\ 0, & \text{else}, \end{cases} \end{aligned} \tag{6.7}$$

where ls is an abbreviation of $\mathrm{ls}(x, \mathcal{T})$ and q_y and $q_{\bar{y}}$ are the indices of the nearest prototype with correct and incorrect label, respectively. The factor $\mathrm{ls}(1-\mathrm{ls})$ of GLVQ depends on the learning time t, while the factor of GMD depends on a fixed η (see eqs. (6.2) and (6.4)). Fig. 6.2 shows contour plots of the loss function, eq. (6.6), and the factor $\mathrm{ls}(1-\mathrm{ls})d_j/(d_1+d_2)^2,\ j \in \{1,2\}$ of the GLVQ algorithm, eq. (6.7), for different learning time parameters $\eta(t) = 0.1,\ 10$. Contour lines indicate steps of 10. The figure illustrates that (i) the GLVQ algorithm learns from the data points which lie near the class boundary, (ii) during learning, the active area along the class boundary gets narrower with increasing learning time parameter $\eta(t)$, and (iii) during learning, the loss function approximates zero-one loss. (In this figure the loss takes on values between 0 and 0.5, because the data points are assumed to be correctly classified.) Because of the factor $d_j/(d_1+d_2)^2,\ j \in \{1,2\}$, in the learning rule, eq. (6.7), the prototypes are adjusted only by the data points with small d_1 and d_2, i.e. the data points whose nearest correct and incorrect prototypes are close to each other. This fact is shown in figure 6.2(c)–(f); the data points between the two inner prototypes $\circ$ and $\square$ possess a larger factor than the other data points near the class boundary. On the one hand, this factor has the positive effect of punishing outliers, while on the other hand it leads to a learning rate which is too large for data sets with small variance $\sigma_x^2 \ll 1$ and too small for data sets with large variance $\sigma_x^2 \gg 1$. Hence, as in the GPD learning algorithm, a scaling factor (γ) is required for an adequate adjustment of the learning rate to a given data space.

[2]Sato and Yamada (1996) suggests a loss function depending on the learning time, $\mathrm{ls}(x, \mathcal{T}) = 1/(1+\exp(-\mu(x)t))$. In order to prevent the rapid decrease of the cost function, I used a monotonically increasing function of learning time, $\eta(t)$, instead of using the learning time t directly. The algorithm using this construction results in much better classification performance than with the original cost function and the cost function which is independent of the learning time. The performance of the three different settings is not compared in this thesis.

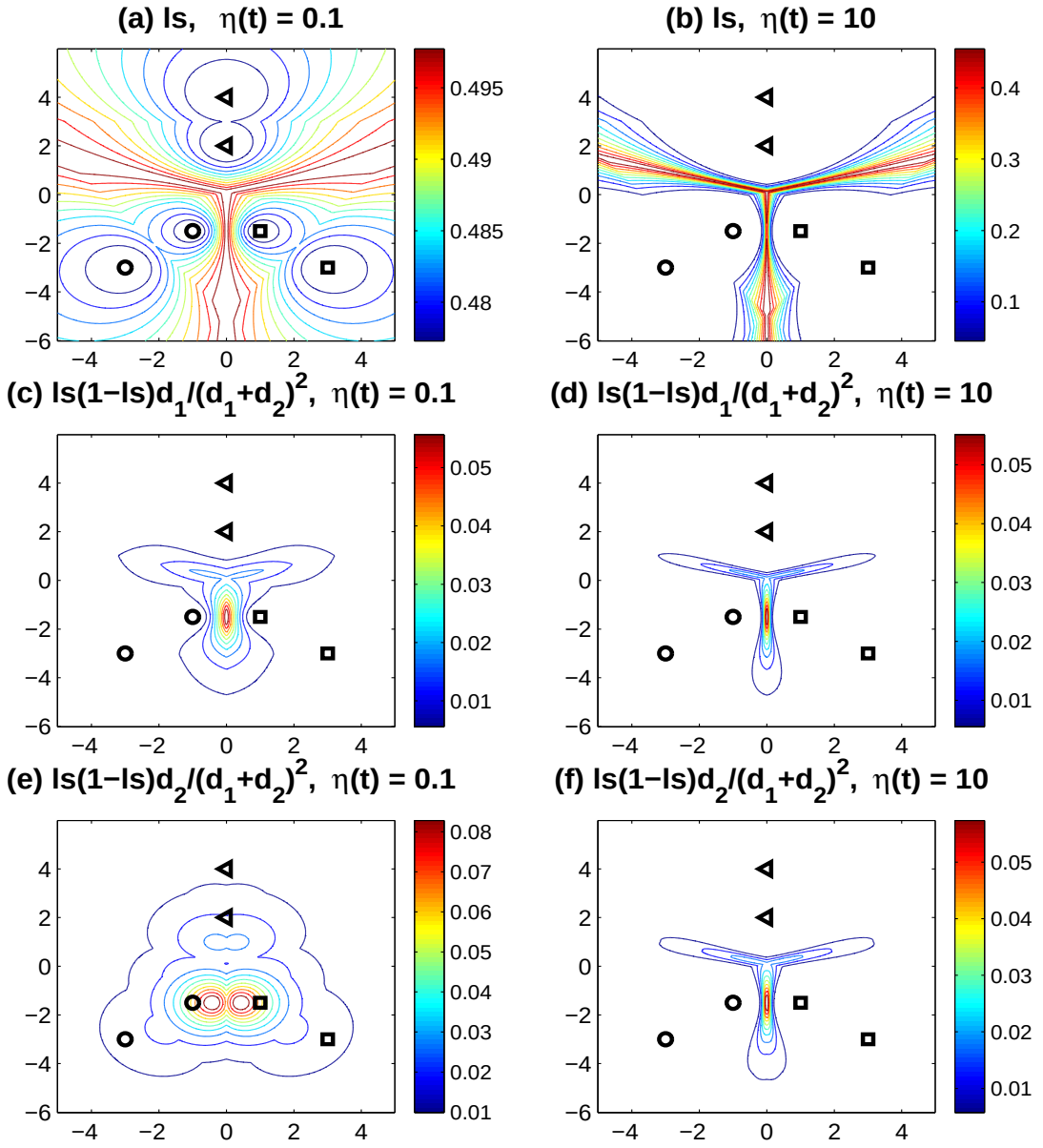


Figure 6.2: Loss and update factor of the GLVQ algorithm for different $\eta(t) = 0.1,\ 10$. Figures (a),(b) show contour plots of the loss function, eq. (6.6), and figs. (c)–(f) show contour plots of the update factor, eq. (6.7), for the nearest correct prototype (c,d) and for the nearest incorrect prototype (e,f), as a function of the position of the data points for two different learning time parameters $\eta(t) = 0.1,\ 10$. Contour lines indicate steps of 10. The figures show that the selection of active data for the learning of prototypes is integrated in the loss function and the learning factor. The loss function depends not only on the distance between data point and class boundary, but also on the distance between data point and the two nearest prototypes. For (small) $\eta(t) = 0.1$ (at the beginning of the learning process) all of the data points have similar loss, while for (large) $\eta(t) = 10$ (with time of learning) only the data points near the class boundary have large loss. For large $\eta(t)$ the prototypes are significantly adjusted only by the data points near the class boundary. For a description of the problem see the caption of fig. 6.1.

6.3 LVQ Maximizing Hypothesis Margin Through Loss Function

Crammer et al. (2002) analyzed the 1-nearest neighbor classifier and showed that LVQ is a natural emergence of the algorithm based large margin principle. For a set of M labeled prototypes $\mathcal{T} = \{(\theta_i, c_i)\}_{i=1}^{M}$, eq. 2.7, they defined the hypothesis margin[3] of $\mathcal{T}$ with respect to a data point x as

$$\mu(x) = \frac{1}{2}(d_2 - d_1), \tag{6.8}$$

where d_1 and d_2 are the Euclidean distances between x and the nearest correct and incorrect prototype, respectively. If an individual loss function $\mathrm{ls}(x, \mathcal{T})$ is defined, the cost function is the sum of the individual loss functions over all data points in the training set $\mathcal{S}$, eq. (2.1);

$$E(\mathcal{S}, \mathcal{T}) = \sum_{i=1}^{N} \mathrm{ls}(x_i, \mathcal{T}).$$

As a suitable loss function for LVQ 2.1 Crammer et al. (2002) suggested the broken linear loss function

$$\mathrm{ls}(x, \mathcal{T}) = \min(2, (1 - \eta\mu(x))_+),$$

where $(\xi)_+$ is a function which takes on the value ξ, if ξ is positive and otherwise 0. Using this loss function they derived an LVQ 2.1-like algorithm.

$$\begin{aligned} \theta_l(t+1) &= \theta_l(t) + \alpha^*(t)\Delta\theta_l(t) \\ \Delta\theta_l(t) &= \begin{cases} \frac{1}{\|x_i - \theta_l\|}(x - \theta_l), & \text{if } |\mu(x_i)| \le 1/\eta, \;\; l = q_y, \\ -\frac{1}{\|x_i - \theta_l\|}(x - \theta_l), & \text{if } |\mu(x_i)| \le 1/\eta, \;\; l = q_{\bar{y}}, \\ 0, & \text{else}, \end{cases} \end{aligned} \tag{6.9}$$

where $\alpha^*(t) = \eta\alpha(t)$ and q_y and $q_{\bar{y}}$ are the indices of the nearest prototype with correct and incorrect class label, respectively. In my thesis, this algorithm will be called Maximal Margin LVQ (MMLVQ). The loss is constant if $|\mu(x)| > 1/\eta$, i.e. the derivative of the loss w.r.t the prototypes is zero. This gives a window rule for selection of active data for the update of the prototypes, thus the prototypes are changed only by the data points which fulfill the condition $|\mu(x)| \le 1/\eta$. For large η the constraint becomes stronger, and the prototypes are changed only by data points with a small difference of the two distances d_1 and d_2, i.e. which lie near the class boundary. With decreasing η the constraint gets looser and more data points can contribute to the learning of the prototypes. Fig. 6.3 shows the loss function as a function of the position of the data points, depending on two different values of η. It illustrates the effect η has on the selection of data points for the adjustment of the prototypes. The term

[3]Hypothesis margin is defined as the distance that the classifier can travel without changing the way it labels any of the data points, while the sample margin is defined as the distance between an instance and the decision boundary induced by the classification rule (Crammer et al. (2002); Vapnik (1995)).

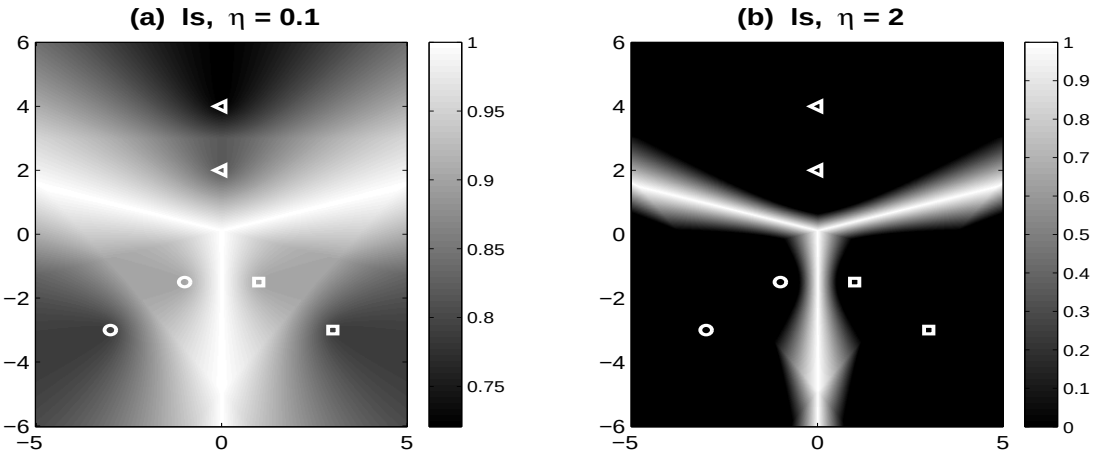


Figure 6.3: Broken linear loss function with different η.
The loss function $\min(2, (1 - \eta\mu(x))_+)$ is shown as a function of the position of the data points for different $\eta = 0.1$, 2. For details on the construction of the problem, see the caption of fig. 6.1. The data selection is implicit in the loss function. For small η almost all data points are used for the learning of the prototypes, while for large η only the data points near the class boundary are used for learning.

$(x - \theta)/\|x - \theta\|$ is a direction vector with length one. Hence, if the given data point fulfills the constraint, the algorithm finds two prototypes to be changed. The amount of change is independent of the distance between the data point and the prototypes and also independent of the scale of the data space. Since the absolute value of change is only influenced by the learning rate α, it may be be too small for data sets with large variance ($\sigma_x^2 \gg 1$) and too large for data sets with small variance ($\sigma_x^2 \ll 1$). Therefore, the introduction of a scaling factor is required in order to accomplish an appropriate learning step for a specific data set. Thus, MMLVQ has two hyper parameters, the scaling factor γ and the window parameter η.

6.4 Summary and Discussion

In this section, the three novel algorithms, derived based on a Gaussian mixture model, and the other three algorithms, derived based on the minimization of a cost function, are summarized and their properties are discussed.

6.4.1 Summary of NPC Based on a GM Model

I constructed three novel algorithms for NPC based on a Gaussian mixture model, under the constraint that the components have similar width and strength. I assumed that the probability density $p(x)$ of the data points x can be described by a mixture model, and that every component l of the mixture is homogeneous in the sense that it generates data points which belong only to one class C_l, eq. (2.12). Then I considered the restricted probability densities of a data point x and its true label y; the conditional probability

densities that a data point x is generated by the mixture model with the correct class label y, $p(x, y|\mathcal{T})$, and with an incorrect class label $\neq y$, $p(x, \bar{y}|\mathcal{T})$, eq. (2.13). Using these, I proposed different objective functions for the learning of NPCs. The SNPC algorithm (in chap. 3) is derived by minimizing the rate of 'soft' misclassification error, while the two soft learning algorithms, SLVQ-LR(in chap. 4) and RSLVQ (in chap. 5), are derived by maximizing two different likelihood ratios:

$$\frac{1}{N}\sum_i \frac{p(x_i, \bar{y}_i|\mathcal{T})}{p(x_i|\mathcal{T})} \stackrel{!}{=} \min, \qquad \texttt{SNPC}$$

$$\prod_i \frac{p(x_i, y_i|\mathcal{T})}{p(x_i, \bar{y}_i|\mathcal{T})} \stackrel{!}{=} \max \equiv \sum_i \log\left(\frac{p(x_i, y_i|\mathcal{T})}{p(x_i, \bar{y}_i|\mathcal{T})}\right) \stackrel{!}{=} \max, \quad \texttt{SLVQ-LR}$$

$$\prod_i \frac{p(x_i, y_i|\mathcal{T})}{p(x_i|\mathcal{T})} \stackrel{!}{=} \max \equiv \sum_i \log\left(\frac{p(x_i, y_i|\mathcal{T})}{p(x_i|\mathcal{T})}\right) \stackrel{!}{=} \max, \qquad \texttt{RSLVQ}$$

The LVQ 2.1-style algorithms are obtained from a Gaussian mixture ansatz, under the assumption that every component has the same width and strength, eq. (2.15). This assumption is made, because classification using NPCs depends only on the relative distances between data points and prototypes. In the three algorithm every prototype with correct (incorrect) label will be attracted to (repelled from) the data point x weighted by different factors:

`correct prototypes`	`incorrect prototypes`	
$\mathrm{ls}(1-\mathrm{ls})P_y(l|x)$	$\mathrm{ls}(1-\mathrm{ls})P_{\bar{y}}(l|x)$	`SNPC`
$P_y(l|x)$	$P_{\bar{y}}(l|x)$	`SLVQ-LR`
$P_y(l|x) - P(l|x)$	$P(l|x)$	`RSLVQ`

where ls is an abbreviation for $\mathrm{ls}(x, \mathcal{T})$ and

$$\mathrm{ls}(x, \mathcal{T}) = \frac{p(x_i, \bar{y}_i|\mathcal{T})}{p(x_i|\mathcal{T})} = \sum_{\{l:c_l\neq y\}} P(l|x), \qquad P(l|x) = \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{k=1}^{M}\exp\left(-\frac{(x-\theta_k)^2}{2\sigma^2}\right)},$$

$$P_y(l|x) = \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{k:c_k=y\}}\exp\left(-\frac{(x-\theta_k)^2}{2\sigma^2}\right)}, \qquad P_{\bar{y}}(l|x) = \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_{\{k:c_k\neq y\}}\exp\left(-\frac{(x-\theta_k)^2}{2\sigma^2}\right)}.$$

In contrast to LVQ 2.1, not only the two nearest prototypes but also all the other prototypes are adjusted by a given training data point. The individual loss function of SNPC, $\sum_{\{c_l\neq y\}} P(l|x)$ is lower bounded by 0 and and the individual gain function of RSLVQ, $\log\left(p(x, y|\mathcal{T})/p(x|\mathcal{T})\right)$, is upper bounded by 1. Hence their learning factors imply the selection of an informative data point for the learning of the prototypes. The individual gain function of SLVQ-LR, $\log\left(p(x, y|\mathcal{T})/p(x, \bar{y}|\mathcal{T})\right)$ is not upper bounded, and it causes a divergence of the prototypes (for details see section 4.3 and figure 4.1). In order to ensure the convergence of the algorithm, a window rule is introduced, which is similar to the window rule of LVQ 2.1. However, the factors of the SNPC and RSLVQ algorithms determine the active region in the data space for the update of prototypes,

and they converge without using a heuristic window rule (for details see figures 3.2, and 5.2). The parameter σ^2 determines the width of the active region of the data points for the update of the prototypes. It is a hyper parameter of the learning rules of the three algorithms, and must be optimized. LVQ-LR, the limit case $\sigma^2 \to 0$ of SLVQ-LR, is the same as LVQ 2.1, except for the heuristically chosen window rule. All three algorithms can be improved by using deterministic annealing of the parameter σ^2.

6.4.2 Summary of LVQ Based on Cost Function Minimization

The three algorithms reviewed in sec. 6.1–6.3 are derived by minimizing different cost functions. The cost functions are defined as the sum of (for GPD: average of) the individual loss functions over all data points in the training set. In order to construct an individual loss function for a data point x, a quality measure $\mu(x)$ for NPC is defined using two distances d_1 and d_2:

$$\begin{aligned} \mu(x) &= d_1 - d_2, \quad \text{for large } \xi,\ \zeta, && \texttt{GPD} \\ \mu(x) &= \frac{d_1^2 - d_2^2}{d_1^2 + d_2^2}, && \texttt{GLVQ} \\ \mu(x) &= \frac{1}{2}(d_2 - d_1), && \texttt{MMLVQ}, \end{aligned} \tag{6.10}$$

where d_1 and d_2 (d_1^2 and d_2^2) are the (squared) Euclidean distances between a data point x and its nearest prototypes with correct and incorrect class label, respectively. The individual loss functions are constructed as a function of the quality measure $\mu(x)$ for NPC:

$$\begin{aligned} \mathrm{ls}(x, \mathcal{T}) &= \frac{1}{1 + \exp(-\eta\mu(x))}, && \texttt{GPD}, \\ \mathrm{ls}(x_t, \mathcal{T}) &= \frac{1}{1 + \exp(-\eta(t)\mu(x_t))}, && \texttt{GLVQ}, \\ \mathrm{ls}(x, \mathcal{T}) &= \min\left(2, (1 - \eta\mu(x))_+\right), && \texttt{MMLVQ}, \end{aligned}$$

where $\eta, \eta(t)$ are window parameters. η is fixed during the whole learning process, while $\eta(t)$ is a monotonically increasing function of the learning time t. These parameters determine the width of the window for the selection of active data used in the update of the prototypes (see the figs. 6.1–6.3). The proposed individual loss functions are lower bounded by 0. This leads to the convergence of the algorithm, if the prototypes are optimized using a stochastic gradient descent method with an appropriate learning rate (Robbins and Monro (1951); Bottou (1998)). Because the loss function is defined using the two nearest prototypes, at every learning step only the two nearest prototypes are adjusted according to the LVQ 2.1 style: For a given data point x, the nearest correct prototype will be attracted to it and the nearest incorrect prototype will be repelled

from it, using the following learning factor:

nearest correct prototype	nearest incorrect prototype	
$\mathrm{ls}(1-\mathrm{ls})\frac{1}{d_1}$,	$\mathrm{ls}(1-\mathrm{ls})\frac{1}{d_2}$,	`GPD`,
$\mathrm{ls}(1-\mathrm{ls})\frac{d_2^2}{(d_1^2+d_2^2)^2}$,	$\mathrm{ls}(1-\mathrm{ls})\frac{d_1^2}{(d_1^2+d_2^2)^2}$,	`GLVQ`,
$\frac{1}{d_1}$,	$\frac{1}{d_2}$,	`MMLVQ`,

where ls is an abbreviation of $\mathrm{ls}(x, \mathcal{T})$. The factors $1/d_1,\ 1/d_2$ of the algorithms GPD and MMLVQ result from the derivative of the Euclidean distance w.r.t. the two nearest prototypes. This term together with the term $(x - \theta_l)$, $l = 1, 2$, generates a direction vector of length one. Because of this factor, the amount of change of the prototypes is independent of the distance between a data point and the two nearest prototypes and of the scale of the data space. The post-factor of GLVQ, $d_j^2/(d_1^2 + d_2^2)^2$, $j = 1, 2$, is too large for data sets with small variance and too small for data sets with large variance. Due to the normalization factor, $1/d_i,\ i = 1, 2$, and the factor, $d_i^2/(d_1^2 + d_2^2)^2,\ i = 1, 2$, the three algorithms require a scaling factor to achieve a suitable learning rate for a given data set. For the GPD and GLVQ algorithms, the window rule is integrated in the learning rule via the factors , while for MMLVQ it is enforced by the broken linear loss function. Note, that the derivative of the sigmoid loss function gives rise to the factor $\mathrm{ls}(1 - \mathrm{ls})$, which implies the selection of data points near the class boundary. With increasing window parameters η and $\eta(t)$, the active regions of the three algorithms are narrowing alongside the class boundary. A similar behavior can be observed in LVQ 2.1 with decreasing window parameter ω (see fig. 2.2).

6.4.3 Discussion

In this section, the properties of the algorithms described so far are summarized and compared to each other. The results of numerical benchmarks will be compared in the next chapter.

- The three soft LVQ algorithms and LVQ-LR were derived based on a generative Gaussian mixture model, where it is assumed that the strength and width of the components of the mixture are identical. SNPC was derived by minimizing the misclassification rate, while SLVQ-LR and RSLVQ were derived by maximizing the likelihood ratio. Because the three soft LVQ algorithms are related to a generative model approach, however, the distance measure can be related to the properties of the probability distribution of the data. The classifier and its selection procedure can be adapted, if prior knowledge about the density functions is available.

- GPD, GLVQ and MMLVQ are derived by minimizing the sum or average of different individual losses, which depends on the own, predefined quality measure for NPC. These quality measures are defined for NPC using the two (squared) Euclidean distances from a data point to its nearest correct and incorrect prototype, respectively.

- The individual loss or gain functions of the three soft LVQ algorithms are constructed using all of the prototypes, while the individual loss functions of the three distance based LVQ algorithm are constructed using only the two nearest prototypes. Hence for every learning step, the soft algorithms adjust all of the prototypes, while the distance based LVQ algorithms adjust only the two nearest prototypes.

- Due to using Euclidean distance and using the relative distance difference, the three distance based LVQ algorithms require scaling factors for a suitable adjustment of the prototypes to the data set. However, the three soft versions of LVQ change the prototypes depending on the distance between data point and prototype, and their learning factors take on values between 0 and 1. Hence, the amount of change of the prototypes is already adjusted to a particular data space, without having to introduce an auxiliary scaling factor.

- The cost function of SNPC is lower bounded by zero, and the likelihood ratio of RSLVQ is upper bounded by one. Hence, minimizing the cost function of SNPC and maximizing the likelihood ratio of RSLVQ leads to the convergence of the algorithms. The individual loss and the cost functions of the distance based LVQ algorithms are lower bounded by zero, and minimizing the cost function leads to the convergence of the algorithms. The selection of informative data points around the class boundary is integrated in the Sigmoid and broken linear loss functions. The likelihood ratio of SLVQ-LR does not possess an upper bound, and maximizing the likelihood ratio of SLVQ-LR causes the divergence of the prototypes. Therefore, a heuristic window rule is introduced in order to prevent this divergence.

- During learning, the window rule is realized in the learning factors for the algorithms SNPC, RSLVQ, GPD and GLVQ, while the MMLVQ algorithm filters the data points fulfilling the condition $|\mu(x)| \leq 1/\eta$.

- The active area is narrowing alongside the class boundary with decreasing width of the mixture components σ^2 (SNPC,RSLVQ), decreasing window width ω (SLVQ-LR, LVQ-LR, LVQ 2.1), increasing window parameter η (GPD,MMLVQ) and increasing learning time parameter $\eta(t)$(GLVQ). These parameters are hyper parameters to be optimized. Furthermore, the three distance based LVQ algorithms contain a scaling factor to be optimized. The three soft LVQ algorithms can be improved via deterministic annealing of the width parameter σ^2.

Chapter 7

Numerical Experiments

In this chapter the performance of the novel algorithms for learning NPC, which were derived based on a Gaussian mixture model (chap. 3–5), is compared to the three distance based algorithms (chap. 6) on the two real-world data sets `letter` and `pendigits`. Furthermore, I propose a new heuristic method for the treatment of hyper parameters, which can be used as an alternative to model selection methods. This approach is motivated by the fact that the problem of finding the optimal hyper parameters suffers from high computational complexity and the occurrence of 'bad' local optima caused by the high non-convexity of the cost function. The proposed method consists of a dynamic scaling of the hyper parameters during the learning process, which, for more than one hyper parameter, is conducted simultaneously. Here, the proposed method is applied to the algorithms based on a Gaussian mixture model and LVQ 2.1. The generalization performance using dynamic hyper parameter scaling is compared to the adjustment of hyper parameters via the cross-validation method on the two real world data sets.

7.1 Design of the Experiments

7.1.1 Data Sets

The data sets `letter` and `pendigits` from the UCI Machine Learning Repository[1] are used for the investigation of the performance of the algorithms.

Letter

is generated from a large number of black-and-white rectangular pixel displays of the 26 capital letters in the English alphabet. The letters were taken from 20 different fonts, and each symbol was randomly distorted. Each pixel image was afterwards converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values between 0 and 15. The dataset contains a total of 20,000 items. For the experiment the data set was split into two subsets, each of which contained the same number of data points for each label. The first subset, the training set, was used to find the optimal values of the

[1] ftp://ftp.ics.uci.edu/pub/machine-learning-databases /letter-recognition/

hyper-parameters; the window parameter ω (LVQ 2.1, LVQ-LR, SLVQ-LR), the width parameters σ^2 (SNPC,SLVQ-LR, RSLVQ) and η (GPD,MMLVQ) and the scaling parameter γ (GPD,GLVQ,MMLVQ). 10-fold cross-validation was performed on the training set for several values of the hyper parameters to determine the values with minimal average validation error. The generalization error for the optimal hyper parameters was then estimated by 10-fold cross-validation on the second (test-) set.

Pendigits

is a database of hand-written digits created by collecting 250 samples from each of 44 writers. Each writer was asked to write 250 digits in random order, while the first 10 digits are ignored. Furthermore, the writers were asked to clear the display if they made any mistakes; those samples were also ignored. The remaining samples written by 30 writers are used for training (7494 samples), and the remaining digits written by the other 14 writers are used for writer independent testing (3498 samples). The data was collected using a pressure sensitive tablet, providing a time series of x and y coordinates. This is spatially normalized and resampled using linear interpolation, yielding 8 pairs of (x, y) coordinates, which are integer values in the range of [0..100]. Therefore, the feature vector has 16 dimensions. The class label takes on code values between [0..9], corresponding to the digits[2]. The training data set is used to find optimal hyper-parameters of the algorithms (see above) and the test data set is used to measure the generalization error. The optimal hyper parameters are found via a 10-fold cross validation method. Using the found optimal hyper parameters, the prototypes are optimized using all of the training data points, and the generalization error of the optimized prototypes is measured via the error on the hold-out test set.

7.1.2 Initialization of Prototypes

The prototypes of each class were initialized by adding random vectors to the centroid $\mu_l \in \mathbb{R}^D$ of all the data points which belong to this class, i.e. $\theta_l = \mu_l + 0.02 \times \zeta \times \sigma_l \times \xi$. $\sigma_l \in \mathbb{R}^D_+$ is the vector of standard deviations along the coordinate axes of data points with label l, ζ is the number of prototypes per class, and $\xi \in [-1\ 1]$ is a number drawn randomly from an uniform distribution.

7.1.3 Learning Rate

The learning rate for stochastic on-line learning is annealed by the formula

$$\alpha(t+1) \ = \ \alpha(0) \times \left[\frac{\alpha_T}{\alpha_T + t} \right],$$

where t is the learning step in the on-line learning process, $\alpha(0)$ is the initial learning rate, and α_T is a positive integer number. The three soft LVQ algorithms without annealing and the algorithms GPD and MMLVQ use similar values $\alpha(0)$ and α_T. The initial learning rate, $\alpha(0)$, and the parameter for the annealing of the learning rate, α_T, are chosen *ad-hoc* after some experiments, and they are not optimized. The used values can be found in appendix A.5. The GLVQ learning algorithm implies an annealing

[2]For more details, refer to `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/pendigits`.

process in the loss function, because of the learning time parameter. Instead of the learning rate, the learning time parameter $\eta(t)$, eq. (6.6), is changed for every learning step t according to

$$\eta(t) = \frac{t}{N}, \tag{7.1}$$

where N is the number of training data points. The learning rates $\alpha(t)$ are set to one during the whole learning process[3], but GLVQ uses indeed an unknown constant learning rate α, which is optimized by optimizing the scaling factor γ.

7.1.4 σ^2 Annealing Schedule

The performance of the three soft LVQ algorithms can be improved via deterministic annealing. The annealed versions of the three algorithms are applied on the letter data set, where the following annealing schedule is used:

$$\begin{aligned} \sigma^2(t+1) &= \sigma^2(t) \times \beta(t) \\ \beta(t+1) &= \beta(t)^{(1.1)}, \end{aligned}$$

where $\beta(0) = 0.99$ and $\sigma^2(0) = \xi_1 \times \sigma^2_{opt}$, where ξ_1 was chosen from $[\,2, 1.2\,]$ depending on the number of prototypes per class. Annealing was terminated if $\sigma^2(t) < \xi_2 \times \sigma^2_{opt}$, where ξ_2 was chosen from $[\,0.5,\ 0.8\,]$ depending on the number of prototypes per class.

7.1.5 Termination of the Algorithms

The algorithms are terminated if the data sets were used at least 30 (`letter`) or 40 (`pendigits`) times iteratively for the training (learning epochs) and the difference of the misclassification (training error) rate of the current and previous epoch is smaller than 10^{-5}. The misclassification error is computed after each epoch, and the indices of the training data points are randomly permuted before the next learning epoch starts. This is done to ensure the randomness of the stochastic learning and to present each data point with similar frequency.

7.2 Experimental Results

7.2.1 letter

Fig. 7.1 shows the average of the misclassification rate and the standard deviation as a function of the number of prototypes per class, which determines the complexity of the model. The generalization ability is measured via 10-fold cross validation error on the test set (50% of the data), where the error bars indicate the standard deviation. The hyper-parameters of the algorithms are optimized using 10-fold cross-validation on the training set (50% of the data). The used values of the learning rate and the optimal values of hyper parameters can be found in appendix A.5 and A.6. Fig. 7.1 (a) shows the results of LVQ 2.1 and three soft LVQ algorithms, SNPC, SLVQ-LR and RSLVQ, while (b) shows the annealing version of the three soft LVQ algorithms. The

[3]Because of using a constant learning rate and decreasing η during the learning process, the amount of the change of prototypes will not decrease, but the active region will be diminished.

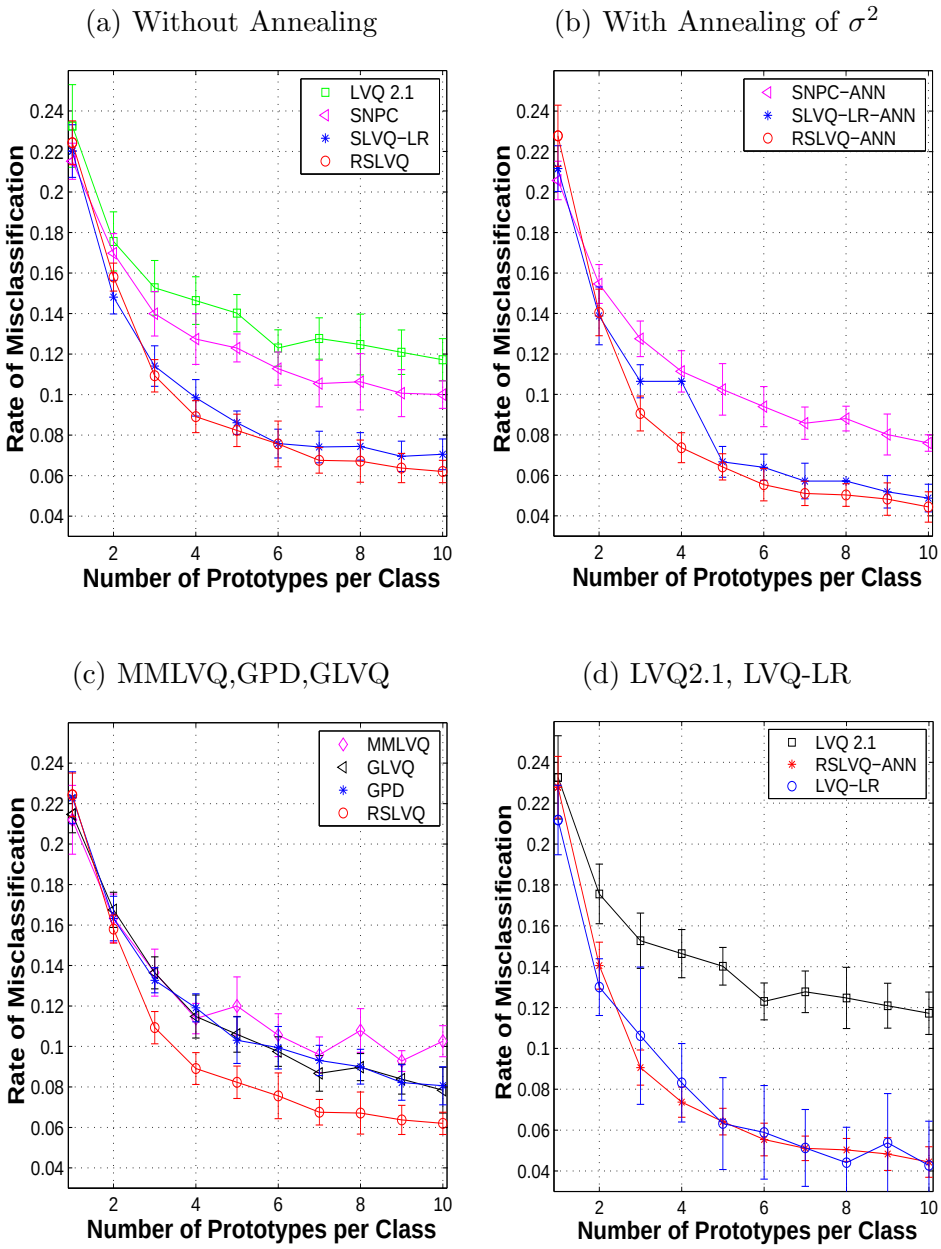


Figure 7.1: The performance of different LVQ algorithms on the data set `letter`. The generalization ability of the three soft LVQ algorithms, the three distance based LVQ algorithms and the LVQ 2.1 algorithm is measured on the data set `letter` via 10-fold cross validation error. Error bars indicate standard deviation. Fig. (a) show the results of LVQ 2.1 and three soft LVQ algorithms, SNPC, SLVQ-LR and RSLVQ and (b) the annealing version of the three soft LVQ algorithms. Fig. (c) shows the results of the three distance based LVQ algorithms and RSLVQ for easy comparison. Fig. (d) shows the results of LVQ 2.1 and LVQ-LR, which are exactly the same algorithm, except the ad-hoc chosen window rule. The results of the annealing version of RSLVQ are shown for easy comparison. The used values of the learning rate and the optimal value of hyper parameters can be found in appendix A.5 and A.6.

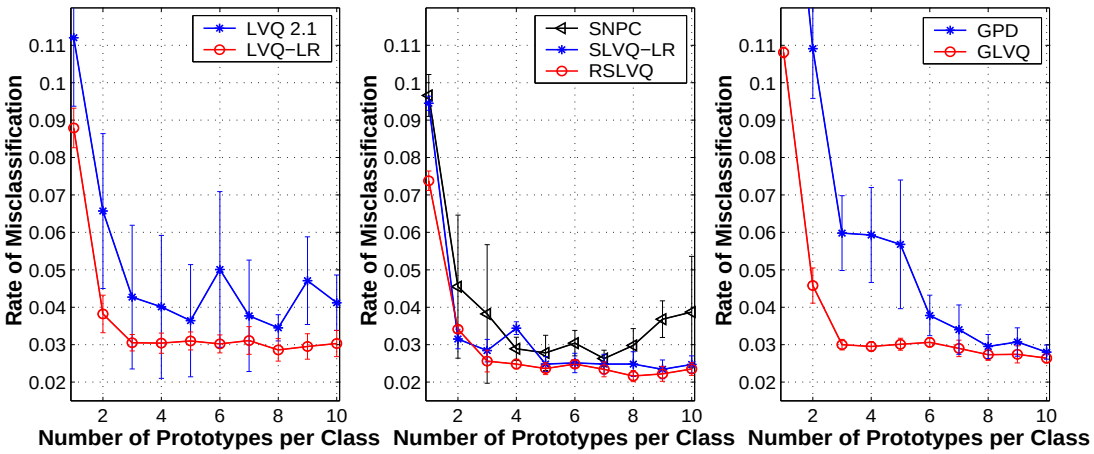


Figure 7.2: The performance of different LVQ algorithms on the data set `pendigits`. Hyper parameters are optimized via 10-fold cross validation on the training data set and the generalization ability is measured on the test data set. The figure shows the average misclassification rate and standard deviation of the test error during 10 runs of training at the selected hyper parameters. The left figure shows the results of LVQ 2.1 and LVQ-LR, which are the same algorithms, except for the window rule. The middle figure shows the performance of the three soft LVQ algorithms, while the right one shows the results of two distance based LVQ algorithms. The results of MMLVQ are too bad to be drawn in the same figure. The x- and y- axes of the three figures are set to the same range, for easy comparison.

two figures demonstrate that the performance of the three soft LVQ algorithms can be improved via deterministic annealing of σ^2. Fig. 7.1 (c) shows the results of the three distance based LVQ algorithms and RSLVQ for easy comparison of their performance. Fig. 7.1 (d) shows LVQ 2.1 and LVQ-LR. The annealing version of RSLVQ is shown for easy comparison. The figure shows that all of the three soft LVQ algorithms and the three distance based LVQ algorithms perform better than LVQ 2.1. Especially the LVQ-LR algorithm and the annealing version of RSLVQ perform outperform the other algorithms. Nevertheless, LVQ-LR is exactly the same algorithm as LVQ 2.1, except for the different heuristically chosen window rule. Figure (d) demonstrates that the window rule of LVQ 2.1, particularly the condition, that exactly one of two nearest prototype must match the label of the data point, is not the best suited for efficient learning of an NP classifier. However, the standard deviation of LVQ-LR is larger than that of RSLVQ with deterministic annealing of σ^2. RSLVQ, SLVQ and LVQ-LR perform better than the three distance based LVQ algorithms.

7.2.2 pendigits

Fig. 7.2 shows the average rate of misclassification and the standard deviation obtained on the test set as a function of the number of prototypes per class. The optimal hyper-parameters of the algorithms are picked from some candidates using 10-fold cross-validation on the given training set (7494 samples). The value of the used learning rate

and the selected hyper parameters can be found in appendix A.5 and A.6. Using the selected hyper parameters, the prototypes are optimized on the whole training set, and the performance of the optimized prototypes is measured via the hold-out test set (3498 samples) error method. Training and test are performed 10 times, to get more representative results. The difference of the 10 runs is caused by the different initial values for the prototypes and the stochasticity of the learning process. Small variance over the 10 runs indicates that an algorithm is robust against the randomness in the training process. The figure shows that the three soft LVQ algorithms, LVQ-LR and GLVQ perform better than LVQ 2.1, while LVQ 2.1 performs better than MMLVQ. The latter one performs too bad to be drawn within the same figure, whose x- and y-ranges are matched to the other figures for the purpose of easy comparison. The left figure demonstrates the advantage of using the alternative window rule, because the two algorithms LVQ 2.1 and LVQ-LR are exactly the same, except for the different heuristically chosen window rules. The results of SNPC in the middle figure illustrates the over-fitting phenomenon which occurs for increasing number of prototypes, corresponding to increasing model complexity. SLVQ-LR and RSLVQ perform better than GLVQ, while LVQ-LR performs almost as well as GLVQ, but with a large standard deviation. SLVQ-LR is more complex to optimize than RSLVQ, because two hyper parameters must be optimized rather than one, hence RSLVQ should be preferred.

7.3 Dynamic Hyper Parameter Scaling

7.3.1 Motivation

In machine learning the hyper parameters are usually kept fixed during training and control the circumstances of the model, e.g. the window width for data selection, the smoothness of the assignment probabilities, the complexity of the resulting function, etc. Usually an optimal hyper parameter is selected from a set of candidates as the one giving the best performance according to a model selection criterion, such as the hold-out test set method, the k-fold cross validation or the leave-one-out method [4]. In the hold-out test set method a test set independent of the training set is used to estimate the generalization error of the optimized model. The k-fold cross validation method makes use of all available data, by splitting the data set into k disjoint subsets (folds) of the same size $\equiv N/K$. The algorithm is trained k times, each time using a different fold as hold-out test set and the remaining $k-1$ subsets as training set. If k is equal to the number of data points, then this method is called leave-one-out cross-validation. Because of the repeated training, cross validation is a computational intensive method. The LVQ algorithms which are introduced in this thesis have one or two hyper parameters, and the optimal hyper parameters are selected using the 10-fold cross validation method. The computational cost is exponentially increasing w.r.t. the number of hyper parameters, e.g. if a model has two hyper parameters, for each of which 10 candidates should be evaluated, then all 100 resulting combinations of pairs of candidates have to be evaluated 10 times, resulting in a total of 1000 training and test processes. Bengio (2000b,a) introduced a method for the simultaneous optimization of many hyper parameters for the case in which the objective function is a quadratic

[4] For more details on model selection methods, see (Duda et al. (2001); Scheffer (1999)).

polynomial of the model parameters, and furthermore continuous and differentiable w.r.t. the hyper parameters. They optimize the hyper parameters using gradient descent on the objective function.

The hyper parameters of the introduced LVQ algorithms control the width of the active region of the data space during the training of the prototypes. The width of the active region is monotonic w.r.t. the hyper parameters, e.g. by decreasing ω and σ^2 the window width narrows along the class boundary. Using these properties, I introduce a simple heuristic learning method for our novel LVQ learning algorithms and LVQ 2.1, in which the hyper parameters are set to a large value initially and decrease very slowly at each learning step. I call this method 'dynamic hyper parameter scaling'. The difference between this method and the deterministic annealing method described in section 7.1.4, is that this method changes the hyper parameters at each learning step, while deterministic annealing changes the parameters after the convergence of the prototypes (at a fixed value of the hyper parameters). With our heuristic method, the parameters are learned only once. If the cost function has many local minima with respect to the hyper parameters, dynamic hyper parameter scaling might be superior to an algorithm which chooses a fixed sigma. The main advantage, however, lies in the lower computational complexity.

7.3.2 Design for the Experiment

Measure of Generalization Error

The generalization on the data set `letter` is measured via 10-fold cross validation using the whole data set. Because there is no need for hyper parameter optimization, one does not have to split the data set into two subsets. In the case of the data set `pendigits` the prototypes are optimized using the training set, and the generalization error is measured via the hold-out test set method. In order to obtain more representative results, the training and test processes are repeated 10 times, and the average of the 10 runs is calculated.

Schedule of Hyper Parameter Scaling

For the dynamic hyper parameter scaling method, the hyper parameter σ^2 (for the three soft LVQ algorithms) and the hyper parameter ω (for LVQ 2.1, SLVQ-LR and LVQ-LR) are scaled down at each learning step. For the initial value of σ^2, I used a multiple of the smallest within-class variance, which is measured by the average of the diagonal elements of the covariance matrix of the data set with the same class label. σ^2 is dynamically scaled according to the following schedule

$$\sigma^2(t+1) = \sigma^2_{\text{ini}} \times \frac{\epsilon_\sigma \times N}{\epsilon_\sigma \times N + t}, \tag{7.2}$$

where σ^2_{ini} is the initial value, ϵ_σ is a positive number, N is the number of training data points and t is the number of learning steps. For dynamic ω scaling, I start with a large value, $0.04 < \omega < 0.25$, and then scale it down very slowly at each learning step. ω is dynamically scaled according to following rule

$$\omega(t+1) = \omega_{\text{ini}} \times \frac{\epsilon_\omega \times N}{\epsilon_\omega \times N + t}, \tag{7.3}$$

where ω_{ini} is the initial value of ω and ϵ_ω is a positive number.

Annealing of Learning Rate

The learning rate of the three algorithms with dynamic σ^2 and ω scaling is changed according to the following rule

$$\alpha(t+1) = \alpha(0) \times \frac{\epsilon_\alpha \times N}{\epsilon_\alpha \times N + t}, \tag{7.4}$$

where $\alpha(0)$ is an initial learning rate and ϵ_α is a positive number.

7.3.3 Experiment Results

In this section, the classification results of the algorithms with the proposed method and with selection of hyper parameters via 10-fold cross validation are compared on the data sets `letter` and `pendigits`.

Data Set Letter: Dynamic σ^2 Scaling

Figure 7.3 shows the generalization error of the soft LVQ algorithms on the data set `letter` for two different treatments of the hyper parameters. The error bar denotes the standard deviation of 10 runs. The blue line denotes the average test errors of an NP classifier whose optimization is performed with a fixed optimal σ^2 selected via the 10-fold cross validation method. The red line denotes the average of test errors of an NP classifier which is trained with decreasing σ^2, while the window parameter ω of SLVQ-LR is set to the value selected via the 10-fold cross validation method. The resulting NP classifier using the dynamic scaling method performs almost as well as the one using 10-fold cross validation for the SNPC and RSLVQ algorithms. The error bar of NPC using the dynamic scaling method is smaller than the one of NPC using cross validation. The SLVQ-LR algorithm with the dynamic scaling method performs consistently better than that with the selection method. Due to the absence of a selection process, the training of an NPC using the dynamic scaling method is much faster than the training with the selection method.

Data Set Letter: Dynamic ω Scaling

Figure 7.4 shows the performance of LVQ 2.1, SLVQ-LR and LVQ-LR with two different methods for the treatment of hyper parameter ω. The blue line denotes the average 10-fold cross validation errors of NPC trained with the selected optimal ω, the red line denotes the average 10-fold cross validation errors of an NP classifier trained with ω-dynamic scaling, while σ^2 for the SLVQ-LR algorithm is set to the selected optimal value. LVQ 2.1 and SLVQ-LR with ω dynamic scaling outperform the algorithms with the selected optimal ω. LVQ-LR with ω dynamic scaling performs worse than with the selected optimal ω, but it results in a much more robust classification performance. Note, that the performance of SLVQ-LR and LVQ-LR with the dynamic scaling method is almost the same, and that they perform slightly better than RSLVQ with the dynamic scaling method.

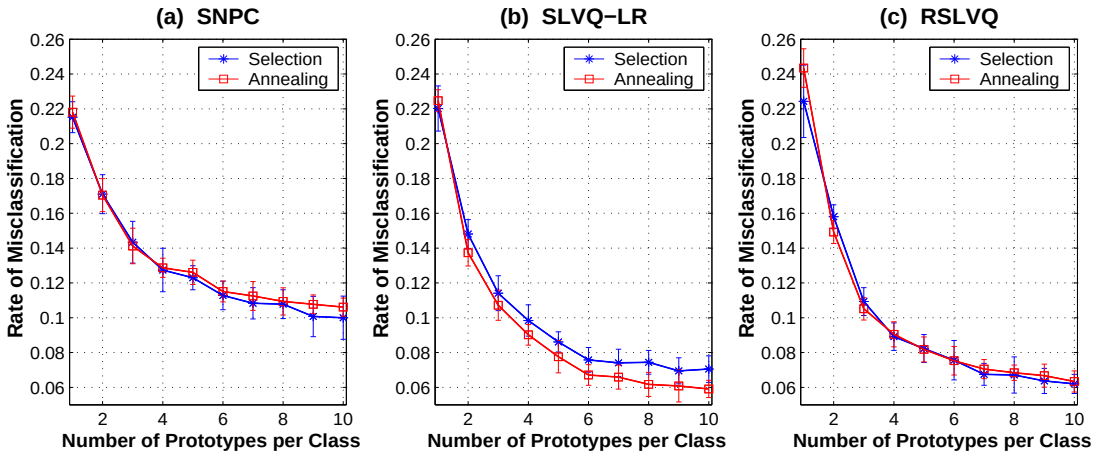


Figure 7.3: Performance of soft LVQ algorithms with dynamic scaling and selection of σ^2

The figure shows the average 10-fold cross validation errors of the soft LVQ algorithms with the σ^2 dynamic scaling method and the selection via 10-fold CV method on the data set `letter`. The blue line shows the performance of an NP classifier optimized with an optimal hyper-parameter selected using the 10-fold cross-validation method. The red line shows the performance of an NP classifier trained with decreasing σ^2, where σ^2 is annealed according to the schedule, eq. (7.2). For the SLVQ-LR algorithm the hyper-parameter ω used is the optimal one selected via the 10-fold cross-validation method. The used parameters for the dynamic scaling schedules are given in appendix A.8.

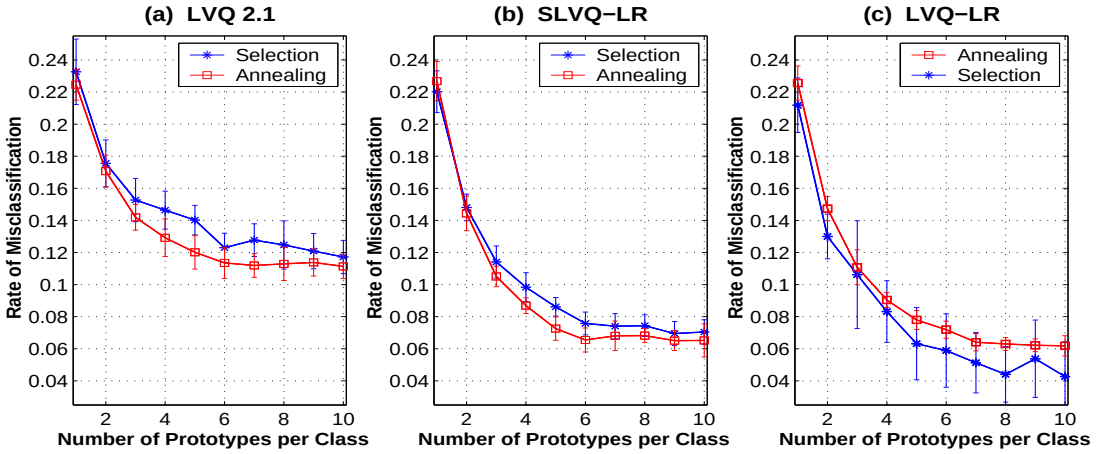


Figure 7.4: Performance of different LVQ algorithms with dynamic scaling and selection of ω

The figure shows the average 10-fold cross validation errors of LVQ 2.1, SLVQ-LR and LVQ-LR on the data set `letter`. The blue line shows the performance of an NP classifier optimized with a selected optimal ω. The red line shows the performance of the resulting NP classifier trained with ω annealed according to the schedule eq. (7.3). For the SLVQ-LR algorithm, the hyper parameter σ^2 used is the optimal one selected via the 10-fold cross-validation method.

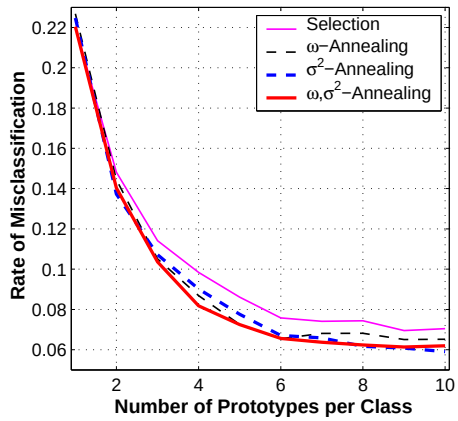


Figure 7.5: Performance of the SLVQ-LR algorithm with a combination of selection and dynamic scaling of ω and σ^2.
The figure shows the average 10-fold cross validation errors of SLVQ-LR with four different treatments of the hyper parameters ω and σ^2. SLVQ-LR with ω/σ^2 dynamic scaling performs better than with fixed selected hyper parameters via the 10-fold cross-validation method. Furthermore, the performance of SLVQ-LR with simultaneous dynamic scaling of σ^2 and ω is superior to the other variants.

Now it is interesting to consider how SLVQ-LR will perform with simultaneous dynamic scaling of σ^2 and ω. Figure 7.5 shows the average 10-fold cross validation error of SLVQ-LR with four different treatments of the hyper parameters. It shows that SLVQ-LR with simultaneous dynamic scaling of two hyper parameters, σ^2 and ω, performs superiorly to the others. Using simultaneous dynamic scaling of two hyper parameters, the computational cost for $N_{\sigma^2} \times N_\omega \times k$ training and test processes is avoided, where N_{σ^2} and N_ω are the number of candidates for σ^2 and ω, respectively, and k is the number of repetitions for the k-fold cross validation.

Data Set Pendigits

The same experiment is conducted on the data set `pendigits`. The results of the experiment are shown in figure 7.6. The figure shows that the performances of SNPC and SLVQ-LR with the dynamic scaling method are better than of the algorithms with the selected optimal hyper parameters. Especially for a large number of prototypes per class ($\geq$ 5), SNPC with dynamic hyper parameter scaling performs almost as well as SLVQ-LR. Furthermore, its variance is much smaller than SNPC with the selection method. SNPC using the dynamic scaling method is more robust against the stochasticity of the training process than SNPC using the selection method. RSLVQ with the selection method is superior to that with the dynamic scaling method. LVQ 2.1 with the dynamic scaling method performs well for a large number of prototypes per class ($\geq$ 7), in which case it performs almost as well as LVQ-LR and its performance is much more robust than the one with the selection method. LVQ-LR with dynamic scaling and with the selection method perform almost equally.

7.4 Comparison to Support Vector Machines

I compared the performance of the novel four Gaussian mixture based algorithms to the performance of support vector machines (SVM) (Schoelkopf and Smola (2001)) using a radial basis function (RBF) kernel[5] on the UCI data sets `letter` and `pendigits`.

[5]I used the implementation of SVMTorch (Collobert and Bengio (2001)) which can be found at http://www.idiap.ch/ bengio/projects/SVMTorch.html.

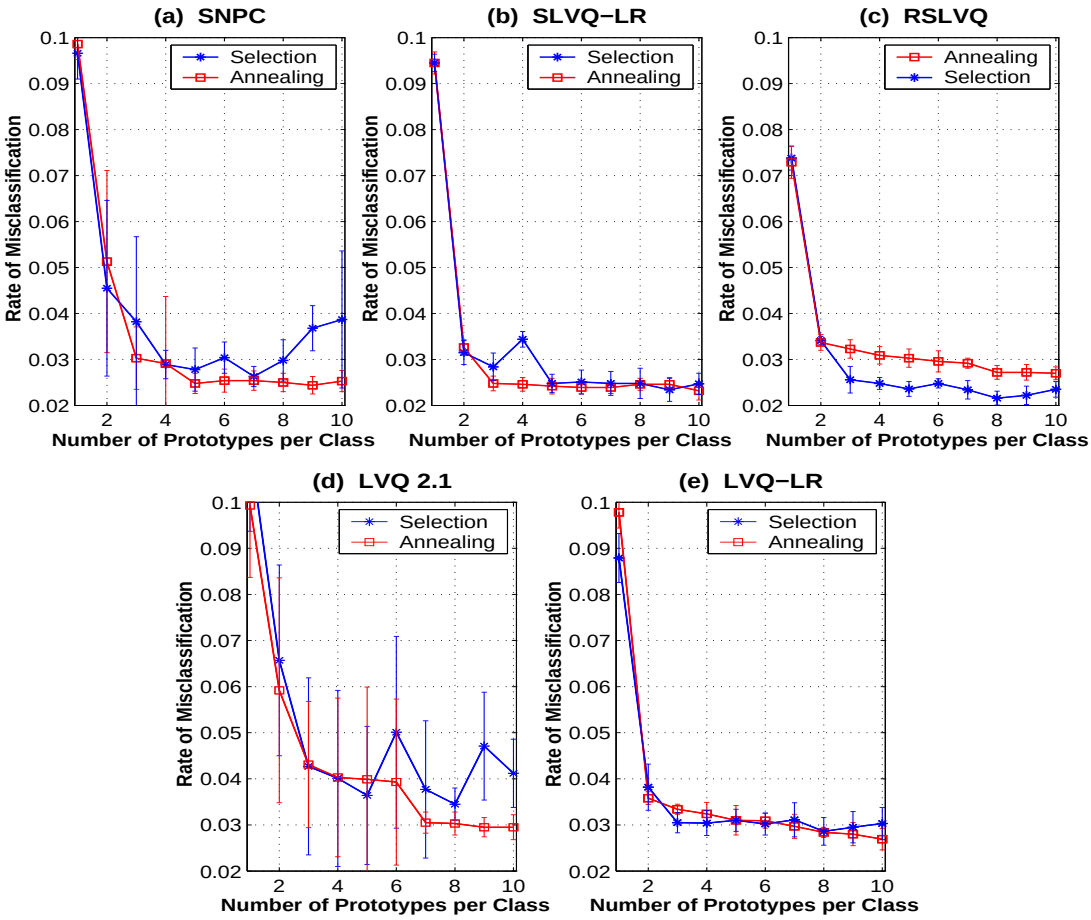


Figure 7.6: Performance of different LVQ algorithms with selection and dynamic scaling of the hyper parameters on the data set `pendigits`.
The figure shows the average hold-out set errors standard deviation of the 10 runs for five different LVQ algorithms on the data set `pendigits`. The blue line denotes the performance of the LVQ algorithms with the optimal hyper parameter selected via 10-fold cross validation. The red line denotes the performance of the LVQ algorithms with dynamic scaling of the hyper parameters: SNPC(σ^2), SLVQ-LR(σ^2, ω), RSLVQ(σ^2), LVQ 2.1 (ω), LVQ-LR (ω). σ^2 is scaled down using the rule in eq. (7.2), and ω is scaled down using the rule in eq. (7.3). α is annealed using the rule in eq. (7.4). The used parameter for the dynamic scaling schedule can be found in appendix A.8.

The hyper parameters σ and C were selected via grid search from a set of candidates using 10-fold cross validation on the training data set. For the data set `letter`, the generalization error is measured using 10-fold cross validation at the selected hyper parameters on the test set. The found optimal hyper parameters are $C = 900,\ \sigma = 19$, and the resulting generalization error is 0.0689 with an average number of 35 support vectors. From a number of 7 prototypes per class on, RSLVQ outperforms the SVM. LVQ-LR and the (deterministic) annealing version of SLVQ-LR and RSLVQ perform superiorly to the SVM from a number of 5 prototypes per class on. SLVQ-LR, RSLVQ and LVQ-LR with the heuristic method of hyper parameter dynamic scaling outperform n the SVM from a number of 5, 8 and 7 prototypes per class on, respectively. For the data set `pendigits` the generalization error is estimated by the hold-out set error on the test set. The selected hyper parameters are $C = 1000,\ \sigma = 200$ and the resulting generalization error is 0.0292 with an average number of 80 support vectors. RSLVQ and SLVQ-LR perform better than the SVM already from a number of 3 prototypes per class on, corresponding to 30 prototypes altogether. By using the dynamic hyper parameter scaling method SNPC, SLVQ-LR, RSLVQ and LVQ-LR outperform the SVM from a number of 4, 3, 7 and 8 prototypes per class on, respectively.

7.5 Summary

In this chapter, the performances of three soft LVQ algorithms (SNPC, SLVQ-LR, RSLVQ) and five hard algorithms (LVQ 2.1, LVQ-LR, GPD, GLVQ, MMLVQ) for NPC are investigated on the UCI `letter` and `pendigits` data sets. The Benchmark shows that the classification results of the four algorithms (SNPC, SLVQ-LR, LVQ-LR, RSLVQ) which are derived based on a Gaussian mixture model have improved performance compared to LVQ 2.1, and the three algorithms based on maximizing the likelihood (SLVQ-LR, LVQ-LR, RSLVQ) perform consistently better than the three hard LVQ algorithms (GPD, GLVQ, MMLVQ) derived from different cost functions. LVQ-LR and LVQ 2.1 differ from each other only in the heuristic window rule, however LVQ-LR far outperforms LVQ 2.1. Note, that our novel algorithms for NPC outperform an SVM, while the number of hyper parameters to be optimized in our new approaches (RSLVQ, LVQ-LR and all proposed algorithms using the annealing hyper-parameter) is less. Furthermore, Nearest Prototype Classifiers allow faster classification then even an SVM, because only the distances between data points and prototypes must be measured and compared. Hence NPC should be preferred over the SVM-method especially in applications like speech processing, where real-time classification is a serious constraint.

In order to alleviate the computational complexity of the proposed algorithms and LVQ 2.1 w.r.t the selection of an optimal hyper parameter, I proposed a heuristic dynamic hyper parameter scaling method, which very slowly decreases the hyper parameters at each training step. The performance of the algorithms with this dynamic scaling of hyper parameters is investigated and compared to the performance of the algorithms where the hyper parameters are selected via 10-fold cross validation. The differences in performance turned out to be not significant, while the results using dynamic hyper parameter scaling are more robust against the randomness induced in the training. Especially SLVQ-LR using the dynamic scaling method performed better than the one using the selection method. This result was consistent for all number of

prototypes per class on both of the two data sets `letter` and `pendigits`.

Part II

Clustering and Embedding of Matrix Data

Chapter 8

Clustering and Embedding

8.1 Clustering of Data

Clustering of data is one of the basic problems in exploratory data analysis, pattern recognition, machine learning and data mining. Data clustering belongs to the category of unsupervised learning problems, where the only accessible information is the original data, and no teacher is available. A main task of unsupervised learning is to uncover hidden structures in the data. This can be either done by estimating the density of the data distribution (Duda et al. (2001),Bishop (1995), Buhmann (1998)), by partitioning the data into subsets according to a given (dis)similarity measure, or by reducing the dimensionality, i.e. embedding high dimensional data into a low dimensional space. The main approaches to density estimation use a parameterized model of the probability density of the data distribution. The basic principle can be explained considering Bayes' theorem :

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta} \tag{8.1}$$

For a given data set $\mathcal{D} = \{x_1, \ldots, x_N\}$, $p(\mathcal{D}|\theta)$ can be considered as a function of the model parameters θ and is called the likelihood of the data. In the maximum Likelihood (ML) approach one directly maximizes the likelihood (or, more conveniently, its logarithm) over the parameters. Often a prior $p(\theta)$ can be assumed for the model parameters, usually to favor less complex models. According to the principle called *Occam's Razor* (Duda et al. (2001); Blumer et al. (1987); Domingos (1999)) this bias on model complexity will usually lead to improved generalization abilities and help to avoid over-fitting the data. The Maximum A Posteriori (MAP) approach makes use of this a priori information by maximizing the posterior $p(\theta|\mathcal{D})$ (or log posterior) instead of the likelihood. Usually the maximization of the likelihood by optimizing the model parameters

$$\theta_{\max} = \arg\max_{\theta} p(x|\theta) \tag{8.2}$$

is done using the Expectation-Maximization (EM) method (Dempster et al. (1977)). The EM algorithm is a general algorithm for maximum likelihood estimation in parameterized models with incomplete data. It consists of two steps, the E (expectation) step and the M (maximization) step. During the E-step, the expected value of the missing

data values is computed given the current value of the model parameters. During the M-step the values of the model parameters are changed such that the likelihood is maximized, while for the missing values in the data one imputes the expected values computed in the E-step.

From the viewpoint of distribution estimation, the task of data clustering can be interpreted as a parameter estimation of a mixture model of the data density (McLachlan and Basford (1988))

$$p(x) \;=\; \sum_{j=1}^{M} p(x|j)P(j),$$

where M is the number of components and the coefficients $P(j)$ are called the mixing parameters. It is assumed that the data has been generated by an unknown number of stochastic processes, which are qualitatively similar to each other. Each process is generated by a unimodal parameterized probability density. Usually, the individual stochastic processes are assumed to be Gaussian,

$$p(x|j) \;=\; \frac{1}{(2\pi)^{D/2}\det(\mathbf{\Sigma}_j)^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu_j)^T\mathbf{\Sigma}_j^{-1}(x-\mu_j)\right\}, \tag{8.3}$$

where D is the dimensionality of x, and the clustering algorithm finds the means μ_j and the $D \times D$ covariance matrix $\mathbf{\Sigma}_j$ of the j-th component of the Gaussian mixture model. Clustering processes group the data set into subsets based on a given dissimilarity measure. This is done in such a way, that the elements assigned to the same group are as homogenous as possible, while the inhomogeneity between different groups is as large as possible. The result of clustering therefore depends on the distance measure, which is usually chosen heuristically, based on prior knowledge or some assumptions about the underlying true data distribution. In the literature many different clustering algorithms have been suggested, with different similarity measures and using different optimization methods. For an overview, see (Duda et al. (2001), Ripley (1996), Jain and Dubes (1988)).

8.2 Distance-Based Clustering

Distance-based clustering techniques allow a quantitative analysis of data points based on their similarity (Gray (1984), Rose et al. (1992)), which is measured by a distance measure, a simple and widely used example being the squared Euclidean distance. An important subgroup of such algorithms is central clustering, where a set of representation vectors, usually called prototype or reference vectors, are derived. A well-known example for this is K-means clustering (MacQueen (1967), Jain and Dubes (1988)), in which the goal is to minimize the squared distance between data points belonging to a cluster and the corresponding cluster center, or 'prototype'. It results in a set of k prototype vectors which cluster the data points with minimal quantization error. In information and communication theory this approach has been known as vector quantization (Gray (1984)), where the prototypes are called a codebook. K-means clustering is considered as the EM optimization of a Gaussian mixture. K-means clustering uses

the squared Euclidean distance $\|x-\mu_j\|^2$ instead of the squared Mahalanobis distance[1] which appears in the exponent in eq. (8.3).

The EM update formula for the center μ_j of j-th component is given by

$$\mu_j \;=\; \frac{\sum_k^N P(j|x_k)x_k}{\sum_k P(j|x_k)},$$

where $P(j|x)$ is the assignment probability of data point x to the component j. The K-means clustering algorithm is obtained under the assumption that (i) each feature of every component has the same variance, i.e. $\mathbf{\Sigma}_j = \sigma^2\mathbf{I},\ \forall\, j = 1,\ldots,M$, where $\mathbf{I}$ is the identity matrix and (ii) every component has the same strength, i.e. $P(j) = \frac{1}{M},\ \forall\, j = 1,\ldots,M$ and (iii) the assignment probability of data points to the clusters is based on the 'winner-takes-all' principle, i.e. $P(j|x)$ equals one, if μ_j is the nearest mixture center from x, and zero else. The fuzzy K-means algorithm (Dunn (1974)) is an extension, where the assigned probabilities take on values between 0 and 1, depending on the distance between the data point and the prototypes and are normalized as $\sum_{j=1}^M P(j|x) = 1,\ j = 1,\ldots,M$. The K-means clustering algorithm is simple, and has been found to perform well on a variety of problems, for instance in Klose, Seo and Obermayer (2005), we have applied it to the analysis of directional geologic data, using arc-length as distance measure.

A drawback of the K-means algorithm is that it might converge to a local minimum of the cost function, which is often high dimensional and non-convex. The obtained local minimum depends upon the initial set of prototypes and the order of the training data points. In order to overcome the problem of local minima, the use of simulated annealing (Kirkpatrick et al. (1983), Cerny (1985)) and of deterministic annealing (Rose et al. (1990); Rose (1998)) has been suggested. Simulated annealing is a stochastic search procedure, which tries to find the optimal solution within a solution space. The transition between states corresponding to different solutions is determined by a Markov process. A new solution with decreased cost is always accepted, while a new solution with increased cost is accepted with an exponentially weighted probability. Deterministic annealing, another optimization technique for central clustering, has been introduced by Rose et al. (1990). It is based on the fuzzy assignment of data points to clusters and also helps to prevent getting stuck in local minima of the cost function. It anneals the parameter of the assignment probability which governs the level of randomness. At the beginning the parameter is set to a large value so that the randomness of the assignment probability is such high that every data point is equally associated with all clusters. The parameter will be annealed after the convergence of the model parameter at a fixed level of randomness. This process is repeated until a given fixed final value of the randomness parameter is reached. The repeated process gradually localizes the influence of the data points. In the case of a very small randomness parameter, the assignment probabilities become hard, and the data points are assigned to the nearest prototype with probability one.

Another drawback of K-means clustering is that it cannot partition data points which are not linearly separable. In order to cluster such linearly non-separable data

[1]The Mahalanobis distance is the same as the Euclidean distance if the covariance matrix is the identity matrix.

kernel K-means clustering and spectral clustering methods can be used. Kernel K-means Clustering (Schölkopf et al. (1998), Zhang and Rudnicky (2002)) is an extension of K-Means which uses a general distance measure in form of a kernel function. In kernel methods (Schoelkopf and Smola (2001)) a kernel function is used to map the data points from a lower dimensional input space to a much higher dimensional feature space, where the data is linearly separable. This so-called 'kernel trick' is based on the fact that according to Mercer's theorem any positive definite kernel can be interpreted as a scalar product in a Hilbert space. The non-linear mapping from input to feature space is implicit in the kernel function and needs not to be performed explicitly.

Spectral clustering (Weiss (1999); Ng et al. (2004); Yu and Shi (2003); Bengio et al. (2004b)) uses the eigenvectors of a dissimilarity matrix derived from the distance between data points to partition the data points into disjoint clusters. In this method, the data points are usually treated as the nodes of a graph, and the pairwise dissimilarities of the data points as the weighted edges in the graph. This way, the problem of clustering the data points is equivalent to the problem of partitioning the graph. Spectral clustering is performed in two steps. First, the data points are embedded into a space in which the structure of the clusters is more pronounced. Then, the data points in the mapping space are clustered using some standard clustering algorithm, like K-means. This approach has been proven useful in many areas, such as circuit layout (Chan et al. (1994)), VLSI design (Alpert et al. (1999)) and image segmentation (Shi and Malik (2000)). Recently, it has been shown that spectral clustering is a special case of weighted kernel K-means (I.S.Dhillon et al. (2004)) and also a special case of weighted Kernel principle component analysis (Wang et al. (2005)).

8.3 Clustering of Matrix Data

8.3.1 Matrix Data

Electronic data processing has made available large amounts of data from different sources all over the world. Part of the data is available in form of a collection of numbers which describe the properties of objects and which are combined to so-called feature vectors. A common assumption is that the most appropriate mathematical description of objects and their relationships are metric (for example Euclidean) spaces. However, there are many kinds of data sets for which this assumption may not hold. Examples are text documents, web-pages, images, and DNA micro-array data. A more flexible way to describe these data sets are matrices, where rows and columns correspond to the objects, and where the entries in the matrix describe the relationships between them (fig. 8.1 a). Documents, for example, can be described by co-occurrence matrices, where the column objects are documents, the row objects are selected keywords and the entries in the matrix describe word frequencies. DNA micro-array data, for example, can be described by matrices, where the column objects are tissue samples, the row objects are genes, and the entries in the matrix correspond to the strength of gene expression. When row and column objects belong to the same set, the data is called pairwise data (fig.8.1 b). When column objects are interpreted as features, the usual feature vector description is obtained.

(a) General Matrix Data

	a	b	c	d	e
α	1.2	0.3	1.1	0.2	0.7
β	1.0	0.1	0.5	1.2	0.3
χ	0.4	1.5	0.7	1.0	2.0
δ	0.7	0.2	0.1	1.9	1.2
ε	2.1	0.9	0.3	0.9	0.1
ϕ	0.1	1.1	0.2	0.8	1.2
γ	0.7	0.5	0.9	2.0	1.4
η	0.3	0.8	1.6	0.1	0.9

(b) Pairwise Data

	a	b	c	d	e	f	g	h
a	0	1.0	0.4	0.7	2.1	0.1	0.7	0.3
b	1.0	0	1.5	0.2	0.9	1.1	0.5	0.7
c	0.4	1.5	0	0.1	0.3	0.2	0.9	1.4
d	0.7	0.2	0.1	0	0.9	0.8	0.9	0.1
e	2.1	0.9	0.3	0.9	0	1.2	1.4	0.9
f	0.1	1.1	0.2	0.8	1.2	0	0.2	0.7
g	0.7	0.5	0.9	0.9	1.4	0.2	0	0.6
h	0.3	0.7	1.4	0.1	0.9	0.7	0.6	0

Figure 8.1: Matrix data.
(a) General form of matrix data. Two sets of objects, column objects a–e and row objects $\alpha - \eta$, are described by their mutual relationships. (b) Pairwise data. Here row and column objects are from the same set.

8.3.2 Clustering of Matrix Data

One important problem which often arises in the analysis of matrix data is the problem of grouping objects in terms of similarity as it can be inferred from the matrix entries. Clustering methods which explicitly refer to matrix data are pairwise clustering methods based on the maximum entropy ansatz (Buhmann and Hofmann (1994)), model-based grouping methods (Hofmann et al. (1999)), and information bottleneck methods (Tishby et al. (1999); Chechik and Tishby (2002); Gondek and Hofmann (2003); Parker et al. (2003)) which are derived from rate-distortion theory (Cover and Thomas (1991),Lin (1991)). Information bottleneck methods have the advantage that they are applicable to matrix data in general. More details on clustering algorithms for pairwise and co-occurrence data can be found in sections 9.1 and 10.1.

8.4 Embedding

The dimension reduction achieved by embedding of high dimensional data in a low dimensional space is applied in many fields of machine learning. Examples are feature extraction as preprocessing for other algorithms, visualization, interpolation and compression. Classic techniques used for embedding are principal component analysis(PCA)(Jolliffe (1986)) and multi-dimensional scaling (MDS)(Borg and Groenen (1997); Cox and Cox (1994); Schiffman et al. (1981); Takane et al. (1977)), which are designed to discover the true structure of data lying on a linear or almost linear subspace of the data space. For example, Cox and Cox (1994) computes at first the affinity

matrix $\mathbf{M}$

$$\mathbf{M}_{ij} = -\frac{1}{2}\left(d(i,j) - \frac{1}{N}\sum_k d(k,j) - \frac{1}{N}\sum_l d(i,l) + \frac{1}{N^2}\sum_{k,m} d(k,j)d(i,m)\right),$$

The embedding of data point x_i is given by $(\sqrt{\lambda_1}v_{i1}, \ldots, \sqrt{\lambda_N}v_{iN})$, where λ_k and $v_{\cdot k}$ are the k-th eigenvalue and eigenvector of $\mathbf{M}$, respectively. In PCA the data is projected onto a linear orthogonal subspace which captures as much of the variance as possible. This is done by computing the sample covariance matrix of the data set and determining its eigenstructure, $\mathbf{S}v_i = \lambda_i v_i$, where $\mathbf{S}$ is the covariance matrix and v_i is the eigenvector with the i-th largest eigenvalue λ_i. The eigenvectors are the principle components, and the eigenvalues are the corresponding variances. PCA is extended to kernel PCA by Schölkopf et al. (1998) using the kernel trick (see also 8.2), which offers nonlinear extensions to many methods based on the scalar (dot) product of vectors (Cristianini and Shawe-Taylor (2000); Schoelkopf and Smola (2001); Vapnik (1998)). Kernel PCA corresponds to linear PCA in the implicitly defined feature space. MDS (Schiffman et al. (1981); Cox and Cox (1994); Borg and Groenen (1997)), in either its linear or its non-linear (Sammon (1969)) form, seeks an embedding in a low (often two- or three-) dimensional space, which preserves the distances between objects as well as possible. When applied to data in high-dimensional spaces preservation of both large and small distances, however, may lead to artifacts in the embedding space which make the interpretation of the data difficult. In order to avoid this problem, local embedding methods have been suggested, which put an emphasis on the preservation of small rather than large distances. These 'local' embedding methods include well known algorithms like Isomap (Tenenbaum et al. (2000)), local linear embedding (Roweis and Saul (2000)), stochastic neighbor embedding (Hinton and Roweis (2003)), Laplacian Eigenmap (Belkin and Niyogi (2002)) , Generative Topographic Mapping (Bishop et al. (1998)), or the Self Organizing Map approaches (Centre (2003)).

Isomap was proposed by Tenenbaum et al. (2000) as a nonlinear extension of MDS, which replaces Euclidean distance by an approximation of the geodesic distance on a manifold. The method is based on constructing a graph $\mathbf{G}$ on the data which is only locally connected and serves as a discrete network representation of the manifold. The geodesic distance $d_G \in \mathbb{R}_+^{N\times N}$ between two points is measured by adding up a sequence of 'short hops' between neighboring points on the graph: Initially, $d_G(i,j)$ is set to $d(i,j)$, the Euclidean distance between the two data points x_i and x_j, if x_i belongs to the k-nearest neighborhood of x_j (K-Isomap) or x_i and x_j are closer than ϵ (ϵ-Isomap), else it is set to ∞. Then all entries $d_G(i,j)$ are replaced by $\min\{d_G(i,j), d_G(i,k) + d_G(k,j) |\ k = 1, \ldots, N\}$, which is the shortest path distance between the two points x_i and x_j in the graph $\mathbf{G}$. Applying the standard MDS on this graph-derived geodesic distance d_G results in an Isomap which preserves the intrinsic geometry of the data at all scales in a high dimensional data space. The local linear embedding (LLE) method (Roweis and Saul (2000)) captures the local geometric properties of a complex embedding manifold by a set of linear coefficients. LLE constructs a neighborhood-preserving embedding by mapping the high dimensional data to a low dimensional vector, which represents global internal coordinates on the manifold and is obtained by optimization of the embedding cost function. LLE is a three step procedure. At first,

the k nearest neighbors of each data point are found, and the weight matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$ is constructed according to the following constraints: (i) $\mathbf{W}_{ij} = 0$, if x_j does not belong to the k nearest neighbors of x_i, and (ii) the rows of the matrix must sum to one: $\sum_j \mathbf{W}_{ij} = 1$. Then the weight matrix is optimized such that all data points x are best linearly reconstructed such that $\sum_{i=1}^{N}(x_i - \sum_j \mathbf{W}_{ij}x_j)^2$ is minimized. At last, the low dimensional embedding vectors y are computed by minimizing $\sum_i(y_i - \sum_j \mathbf{W}_{ij}y_j)^2$ using the optimized $\mathbf{W}$. In Laplacian Eigenmaps (Belkin and Niyogi (2002)) a nonlinear dimensionality reduction is achieved by solving a generalized eigenvector problem using a Laplacian approximation on the graph. The manifold is again approximated by the adjacency graph computed from the data points.

LLE, Isomap, Laplacian Eigenmaps are unsupervised non-parametric spectral methods for embedding data into a lower dimension. They share the advantages of PCA and MDS of having few free parameters and being based on a non-iterative global optimization of a natural cost function. They provide local manifold learning using eigenvalue decomposition based on an adaptive kernel, and can be interpreted as a special form of kernel PCA, where the kernel represents the manifold based on local neighborhood relationship (Ham et al. (2003); Bengio et al. (2004a); Bengio and Monperrus (2005)). This relation is constructed by a fixed local neighborhood graph $\mathcal{G}(V, E)$, where the nodes V represent different data points and the edges E represent the neighborhood relationship. However, these three spectral embedding methods do not provide a projection function for a new data point, and this is the main difference between the three spectral embedding methods and kernel PCA (Ham et al. (2003)). In order to solve this problem Bengio et al. (2004a) proposed to use the so-called Nyström formula (Baker (1977)) for projecting a new data point onto the manifold. The advantage and the justification of using this formula and its relation to kernel PCA can be found in Williams and Seeger (2000, 2001).

Generative Topographic Mapping (GTM) (Bishop et al. (1998)) is a non-linear latent variable model for modeling continuous parametric probability distributions. Generative approaches assume that the observed data was generated by a density function, which they try to model under a set of constraints. These constraints usually restrict the set of possible models to those with a low intrinsic dimensionality. GTM defines a non-linear continuous and differentiable parametric mapping $x(y, \mathbf{W})$ from a intrinsically low dimensional latent space ($y \in \mathbb{R}^L$) to a high dimensional data space ($x \in \mathbb{R}^D$), where nearby points in the latent space map to nearby points in the data space. For example $x(y, \mathbf{W})$ could be a multi-layer perceptron (Bishop (1995); Haykin (1994)), in which case $\mathbf{W}$ is a matrix containing the network weights and the bias. Because of the non-linearity of the transformation, two clusters of data which are far from each other in the data space may appear much closer when visualized in the latent space. Using this mapping function and some constraints a probability distribution can be constructed in the data space, which is a constrained mixture of Gaussian whose centers depend on the mapping function:

$$p(t|y, \mathbf{W}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{D}(t^i - x^i(y, \mathbf{W}))^2\right),$$

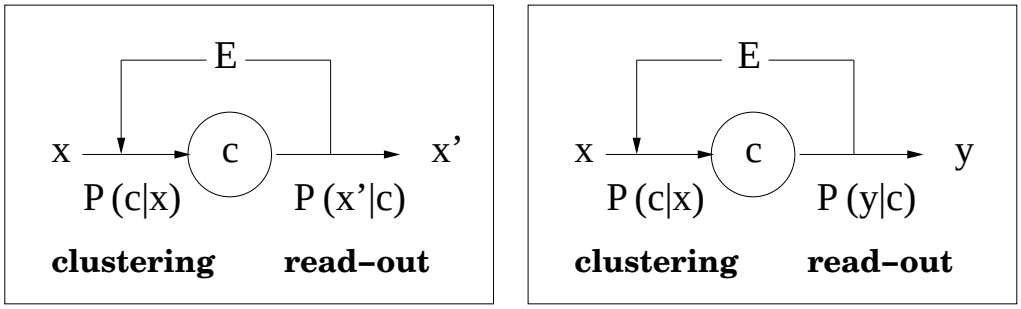
$$p(t|\mathbf{W},\sigma^2) = \int p(t|y,\mathbf{W},\sigma^2)p(y)dy, \tag{8.4}$$

$$\approx \frac{1}{M}\sum_{k=1}^{M} p(t|y_k,\mathbf{W},\sigma^2), \tag{8.5}$$

where $t \in \mathbb{R}^D$ is a point in the data space of dimensionality D, M is the number of components of the Gaussian mixture, and σ^2 denotes the noise variance. Eq. 8.5 is obtained by assuming $p(y)$ to be a set of M equally weighted delta functions on a regular grid: $p(y) = 1/M \sum_{k=1}^{M} \delta(y == y_k)$. Points y on a regular grid in the low-dimensional latent space are mapped using a parameterized non-linear mapping $x(y, \mathbf{W})$ to corresponding centers of Gaussians. The order of the latent points is reflected in the ordering of the centers of Gaussians in the data set, if the mapping $x(y, \mathbf{W})$ is continuous. The (log) likelihood for the function, eq. 8.5, can be maximized by optimizing the model parameters which are learned using the Expectation-Maximization (EM) method (Dempster et al. (1977)). The main application of GTM is the visualization of high dimensional data from the modeled distribution, based on the posterior distribution over the latent space induced by points in the data space.

Self-Organizing Maps were introduced by Teuvo Kohonen in the beginning of the 1980s as a well-founded heuristic method for the clustering and the embedding of data for the purpose of visualization (Kohonen (1982b, 2001)). Theoretical investigations followed (Kohonen (1982a); Kangas (1994)), but a complete characterization of stationary states and convergence properties turned out to be difficult to achieve (Erwin et al. (1992); Cottrell et al. (1998)). A new view was provided by Luttrell (1994). He interpreted the Self-Organizing Map and the underlying clustering method as a stochastic process and described it by a folded Markov chain, thus making contact to concepts from coding theory. These ideas were subsequently taken up by Graepel et al. (1997), who derived a family of Self-Organizing Map methods using the concept of source-channel coding. The main advantage of this new interpretation was that it allowed the derivation of a quality measure, a cost-function, for the clustering and the embedding solutions which were to be optimized. The existence of a cost function had several advantages: (*i*) It allowed a better interpretation of the methods' results, (*ii*) it allowed the application of robust optimization and regularization techniques (Rose (1998), Hofmann and Buhmann (1997)), and (*iii*) it allowed the application of a wider range of model selection techniques (Linhart and Zucchini (1986); Scheffer (1999)).

Relating clustering and embedding to data compression and to noisy channels naturally leads to the application of rate distortion theory (Berger (1971)), a step which has recently been taken by the information bottleneck approaches (Tishby et al. (1999)). However, most approaches were geared towards the analysis of vectorial data. Extensions to matrix data have so far been proposed only for the Self Organizing Map (for pairwise matrices (Graepel and Obermayer (1999),Roth et al. (2003))) and model-based grouping methods (for co-occurrence data (Hofmann (2000))).



(a) pairwise clustering (b) clustering of matrix data

Figure 8.2: Clustering by minimizing rates.
(a) Pairwise data: objects x are assigned to clusters c with probabilities $P(c|x)$. In order to asses the quality of the representation $\mathcal{C}$, objects x' are reconstructed using a read-out process with reconstruction probabilities $P(x'|c)$ and are compared to the assigned objects using a proper dissimilarity measure E. (b) Matrix data: objects x are assigned to clusters c with probabilities $P(c|x)$. The quality of the representation $\mathcal{C}$, however, is accessed according to its 'prediction' quality, i.e. to the information, this representation contains about a set of related object y.

8.5 Grouping and Embedding of Data Object based on Rate Distortion Theory

8.5.1 Clustering

Let us consider a set $\mathcal{X}$ of data objects x. Given these objects and their properties, the goal is to group them into clusters by taking into account how similar these objects are. This can be done by assigning objects to a given set $\mathcal{C}$ of groups c, which I will call 'clusters' in the following. In general, the assignment can be probabilistic, and the assignment probabilities are given by the posterior probabilities $P(c|x)$. A special case occurs if $P(c|x) \in \{0, 1\}$. In this case, assignments are deterministic and the clustering method is called 'hard'. Otherwise, the clustering method is called 'soft' or sometimes, 'fuzzy'.

In order to determine good values for the assignment probabilities a performance criterion is needed, which allows the evaluation of a clustering solution. Here two conflicting goals have to be balanced in a proper way. The first goal is to reduce the information contained in the set of objects by mapping them onto a smaller number of clusters. Clearly, a good clustering solution compresses the data and discards as much information about 'irrelevant' object details as possible. The second goal is to keep the relevant differences between the objects, which then characterize the different groups. Clearly, a good clustering solution keeps this information about the data objects. But how can this relevant information be quantified? A generic way to do so is to postulate a 'read-out' process, which takes the groups c as inputs and reconstructs the original objects x. In general, the reconstruction is probabilistic and must be described by read-out probabilities $P(x'|c)$. By comparing the set of reconstructed objects x' to the set x

of originally observed objects, a distortion measure E can be defined, which quantifies to what extent relevant information has been kept. Figure 8.2a illustrates this scheme and makes the connection to source coding problems very explicit. Data objects x are coded using $P(c|x)$. The representations c are then decoded by the read-out device using $P(x|c)$, and are then compared with the original objects to assess performance.

Now we are able to quantify the clustering goal and the balance between information loss and information preservation. A good clustering solution is one for which the rate, measured by the mutual information $I(C;X)$, between the sets $\mathcal{X}$ and $\mathcal{C}$ is minimized under the constraint that the distortion does not exceed a given value E_0,

$$I(C;X) \;=\; \min \qquad \text{such that} \qquad E(X,X') \;\leq\; E_0. \tag{8.6}$$

In classical information theory $E(X,X')$ is the expectation of a suitable defined distortion with respect to the joint distribution of X and X' (Berger (1971)). Depending on the problem setting, the cost-function (8.6) must be optimized with respect to the assignment probabilities $P(c|x)$ or with respect to both the probabilities for assignment, $P(c|x)$, and read-out, $P(x|c)$. In a clustering setting, the above constraint is often replaced by an equality constraint and the modified optimization problem,

$$I(C;X) \;=\; \min \qquad \text{such that} \qquad E(X,X') \;=\; E_0, \tag{8.7}$$

is considered. Obviously, there are historical reasons for this, but there are also two arguments in favor of doing so: (i) The equality constraint often leads to a simpler optimization problem. (ii) E_0 may be interpreted as a 'resolution parameter' for clustering which - implicitly - determines the number of groups.

A somewhat different clustering problem arises for matrix data, where objects from two different sets $\mathcal{X}$ and $\mathcal{Y}$ are observed and characterized as pairs (x,y). If data objects x should be grouped such that information about the corresponding data objects y is preserved, it suffices for the read-out process to reconstruct (predict) the 'target' objects y. The corresponding source coding problem is depicted in fig. 2b. In this case, eqs. (8.6,8.7) must be changed to:

$$I(C;X) \;=\; \min \qquad \text{such that} \qquad E(X,Y) \;\leq\; E_0 \tag{8.8}$$

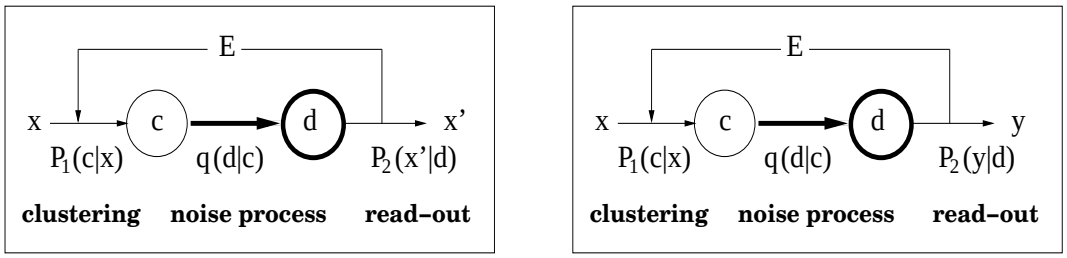
and

$$I(C;X) \;=\; \min \qquad \text{such that} \qquad E(X,Y) \;=\; E_0. \tag{8.9}$$

The optimization problem, eq. (8.8), corresponds to the information bottleneck method (Tishby et al. (1999)), and I will show in chapter 9 that the optimization problem, eq. (8.7), is related to the mean field approximation methods of Buhmann and Hofmann (1994).

8.5.2 Embedding and Visualization with Self-Organizing Maps

Self-Organizing Maps are a widely used, versatile tool for constructing a neighborhood preserving embedding of the data objects for the purpose of visualization. The neighborhood preserving map of the data objects is generated by breaking the permutation symmetry of the corresponding clustering problem. First a so-called 'map space' or



(a) Self-Organizing Maps for pairwise data (b) Self-Organizing Maps for matrix data

Figure 8.3: Topographic clustering of matrix data.
A noise process $q(d|c)$ is introduced which acts on the representation c (cf. fig. 8.2). (a) Pairwise data. (b) Matrix data.

'index space' for clusters is defined, then a measure of similarity between indices is provided, and - finally - permutation symmetry is broken by assigning groups of similar objects to clusters with similar indices.

Luttrell (1994) and Graepel et al. (1997) have shown, that a simple way to break the permutation symmetry in a communication framework is to consider noise. Figure 8.3 illustrates this approach. Before the read-out device operates on the representations, cluster assignments are changed with probability $q(d|c)$. Solving any of the optimization problems eqs. (8.6-8.9) will then lead to solutions which favor the assignment of similar data objects to either the same cluster c or to clusters c and d which have a high probability $q(d|c)$ and $q(c|d)$ of confusion. If - like for the standard Self-Organizing Map - an index space and a measure of similarity between indices is provided, a proper choice of $q(d|c)$ enforces a neighborhood preserving map. Indeed it has been shown in Luttrell (1994) and Graepel et al. (1997), that for proper choices of the distortion measure the confusion probabilities $q(d|c)$ can be directly identified with the neighborhood function of the standard Self-Organizing Map. In chapter 9, I will use this trick to re-derive the Self-Organizing Map for pairwise data developed by Graepel and Obermayer (1999) and to extend the Self-Organizing Map approach to matrix data in general, using co-occurrence data as an example. However, this approach to the construction of SOM-like methods is applicable to all data analysis problems which can be cast into the form illustrated by fig. 8.3. This naturally includes data representations based on feature vectors.

Chapter 9

Clustering and Embedding of Pairwise Data

9.1 Introduction

There are many classes of algorithms for exploratory data analysis which are based on a vectorial representation of the data. In vectorial data each measurement corresponds to a certain feature which is evaluated at an external scale. In fields such as biochemistry, economics and psychology the data is often given only as set of scores of pairwise comparisons, usually similarity or dissimilarity measures. For the analysis of vectorial data instead of the full distribution often only a few lower-order moments, e.g. mean and covariance, are used. The dissimilarity based approach has the advantage that it does not reduce the distributional information before comparing the individuals, thereby preserving more information. Additionally, it has the advantage that it is not necessary to specify a suitable metric, like weighted squared distance (e.g. Fuzzy K-means), on vectors. Often such a specification of a metric is quite heuristic and not appropriate for a given problem. The pairwise dissimilarity matrix can also be generated from vectorial data, if a metric or distance function is available. An overview of the properties of pairwise proximity data can be found in Everitt and Rabe-Hesketh (1997).

This chapter focuses on the analysis of pairwise data with respect to clustering and embedding (see. chapter 8). Pairwise clustering aims to detect hidden structure in the data by partitioning the data into subsets based on similarity (or dissimilarity). Similar data points should be assigned to the same cluster, achieving high homogeneity within a cluster and high inhomogeneity between different clusters. Clustering algorithms for pairwise proximity data have previously been derived by Buhmann and Hofmann (1994). Their approach is based on the maximum entropy principle and uses a cost function for pairwise clustering which aims at minimizing the sum of dissimilarities between data points belonging to the same cluster. They showed that the cost function of central clustering is a special case of pairwise clustering, if squared Euclidean distance is taken as distance measure. In this case, the cost function for pairwise data can be obtained by substituting the prototype in central clustering with the weighted mean of the data points within a cluster. The Maximum entropy method has been known as the least unbiased method being maximally noncommittal with respect to

missing data and its use stresses the robustness of this inference technique (Jaynes (1957a,b)). Application of the maximum entropy principle on the central clustering problem was originally suggested by Rose et al. (1992). The Gibbs probability distribution of pairwise clustering obtained from the maximum entropy principle cannot be factorized, unlike in central clustering. This problem is caused by the dependency of assignments on each other; each assignment variable interacts with all other assignment variables. The Gibbs probability distribution for pairwise clustering is estimated using a mean-field approximation method which neglects the interaction between assignment variables in the pairwise clustering cost function. This way, a factorized distribution from the family of Gibbs distributions is determined, which is most similar to the un-factorized distribution. In order to obtain a solution of pairwise clustering which is robust against noise and to prevent bad local minima caused by the high non-convexity of the cost function, Hofmann and Buhmann (1997) suggested deterministic annealing w.r.t. the temperature parameter. The method was further extended by a hierarchical clustering approach for pairwise data (Hofmann and Buhmann (1995a); Puzicha et al. (2000)). For the purpose of simultaneous visualization and clustering a multidimensional scaling method for pairwise clustering was suggested by Hofmann and Buhmann (1995b). A further embedding method, Topological mapping of proximity (TMP), was proposed by Graepel and Obermayer (1999) based on a cost function which is derived in a Bayesian framework for Folded Markov Chains describing auto-encoders according to Luttrell (1994).

A drawback of working with pairwise data is the scaling behavior with respect to the number of data points. The number of dissimilarity values grows quadratically with the number of data objects. Thus, N objects give rise to $N(N-1)/2$ relations. However, it was shown by Kruskal (1964) and Spence (1982) that dissimilarity matrices are highly redundant if there is some structure in the data to be discovered. The scaling problem for dissimilarity based approaches can be overcome by training only with selected data points. Hofmann and Buhmann (1998) devised an approach for pairwise data called 'active data clustering', which consists of two strategies: the treatment of missing data and the selection of the dissimilarity values which are most relevant for the clustering problem. They suggested an active data clustering approach especially suited for sparse pairwise data with missing values, based on statistical decision theory (see e.g. Raiffa and Schlaifer (2000)). This so-called 'expected value of sampling information(EVSI)' assesses how much gain can be expected from additional data. The missing dissimilarity value between two data objects is replaced by the average value of the dissimilarities of data objects which are assigned to the same cluster. Zöller and Buhmann (2000) suggested a strategy of active learning for hierarchical pairwise data clustering (Puzicha et al. (2000)). Hasenjaeger et al. (1999) applied the active learning strategy by Hofmann and Buhmann (1998) to topographic pairwise clustering based on SOMs (Graepel and Obermayer (1999)).

In this chapter, first the problem of clustering of pairwise proximity data is considered. Then the source coding framework of fig. 8.3(a) is applied to pairwise clustering for the purpose of visualization. In section 9.2 the clustering method for pairwise data is derived based on rate distortion theory, using eq. (8.7), and its relation to the pre-

viously proposed maximum entropy methods is discussed. Section 9.3 concentrates on the behavior of the clustering solution for different values of E_0 and the corresponding Lagrange parameter β. In section 9.4 an optimization method, deterministic annealing, is considered. In order to prevent the convergence to local optima, a novel deterministic annealing scheme is proposed in section 9.5. There, the cardinality of the set of clusters is not fixed, but increases with every structural change of the clustering solution. This method considerably improves the deterministic annealing scheme. Section 9.6 summarizes pairwise clustering with the incremental splitting method. In section 9.7, the embedding of pairwise data is derived by applying the source coding framework, which is obtained by breaking the permutation symmetry of the clustering problem. The performance of the algorithms derived and proposed in this chapter is tested on a protein data set and DNA microarray data set in section 9.8.1.

9.2 Derivation of the Pairwise Clustering Method based on Minimization of Rate

Let us consider a set $\mathcal{X}$ of N objects x_i which are characterized by a matrix $\mathbf{D}$ of pairwise dissimilarities $d(i,j)$, $i,j = 1,...,N$ (see fig.8.1(b)). In order to solve eq. (8.7), a read-out process has to be specified, and a distortion measure has to be constructed. As long as there are no constraints imposed by any later processing of the data, an ideal read-out process should be considered, which makes use of the information content of the representation c in an optimal way. This optimal read-out process is given by the Bayes rule:

$$P(x_i|l) \;=\; \frac{P(l|x_i)P(x_i)}{P(l)}, \qquad P(l) \;=\; \frac{1}{N}\sum_i P(l|x_i), \tag{9.1}$$

where $P(l|x_j)$ denotes the assignment probability of data point x_j to the lth cluster, and every data object x_j used for training has been weighted by the same factor $P(x_j) = \frac{1}{N}$. The individual distortions between objects are given by the matrix $\mathbf{D}$. If there are no reasons to give certain pairs of objects or certain values of dissimilarities more weight, a natural choice for the distortion measure is the average dissimilarity E given by:

$$E \;=< d(x,x') >= \; \frac{1}{N}\sum_{i,j}\sum_l P(l|x_i)P(x_j|l)d(i,j), \tag{9.2}$$

where $P(x_j|l)$ is given by eq. (9.1).

The optimization problem, eq. (8.7), is then solved using the method of Lagrange multipliers (Bertsekas (1982); Duda et al. (2001)). The Lagrange functional is given by

$$\mathcal{L}_1 \;=\; I(C;X) \;+\; \beta(< d(x,x') > -E_0) \;+\; \sum_i \lambda_i \left(\sum_m P(m|x_i) - 1\right), \tag{9.3}$$

where the rightmost term enforces the normalization of the assignment probabilities and β is the Lagrange parameter enforcing the constraint $E = E_0$. Eqs. (9.2) and (9.3) state that the optimal representation $\{P(c|x)^{\mathrm{opt}}\}$ should contain the minimal amount

of information about $\mathcal{X}$, under the constraint that the average distance between an object and its optimal reconstruction should have a fixed (small) value. By setting the derivative of the Lagrangian with respect to $P(c|x)$ to zero, we obtain

$$P(l|x_s) = \frac{\exp(-\beta\mathcal{E}(l,s))P(l)}{\sum_n \exp(-\beta\mathcal{E}(n,s))P(n)}, \tag{9.4}$$

$$\mathcal{E}(l,s) = \sum_i P(x_i|l)\left[d(i,s) - \frac{1}{2}\sum_j P(x_j|l)d(i,j)\right], \tag{9.5}$$

which can be solved via fixed point iteration for every value of β. $P(l|x_s)$ is sometimes also called the posterior probability for assignment. The normalization condition $\sum_c P(c|x_s) = 1$, $\forall\, s$, is fulfilled if $\exp(\lambda_s N) = \sum_n \exp\left(-\beta\mathcal{E}(n,s)\right) P(n)$, and positivity of the $P(c|x)$ is assured by the exponential function in eq. (9.4).

Eqs. (9.4,9.5) are related to the pairwise clustering methods previously proposed by Hofmann and Buhmann (1997) and Graepel and Obermayer (1999). These algorithms are obtained, if the mutual information $I(X;C)$ in the cost function, eq. (8.7), is replaced by the negative conditional entropy

$$-H(C;X) = \sum_l \sum_i P(l,x_i)\log P(l|x_i).$$

This reflects the fact that the methods from Hofmann and Buhmann (1997) and Graepel and Obermayer (1999) are based on the maximum entropy principle while the new method is derived from the idea of minimizing a rate. Both algorithms differ by the factor $P(l)$ in the assignment probability, i.e. the previously proposed methods calculate $P(l|x_s)$ using

$$P(l|x_s) = \frac{\exp(-\beta\mathcal{E}(l,s))}{\sum_n \exp(-\beta\mathcal{E}(n,s))}, \tag{9.6}$$

where $\mathcal{E}(l,s)$ is given by eq. (9.5). Note, that eqs. (9.4,9.5) were derived with the idea in mind to maximally compress the data under the constraint that the average distortion is fixed. Eqs. (9.5,9.6) however, were derived by minimizing the average distortion for a fixed complexity of the representation. If vectorial data is considered, where the term $\mathcal{E}(l,s)$ corresponds to the averaged squared distance between feature vectors which describe data objects, abovementioned algorithms reduce to standard soft vector quantization. Whereas minimization of the rate, eqs. (9.4,9.5), leads to the mass constrained clustering approach proposed by Rose et al.(Rose et al. (1993)), maximization of the entropy, eqs. (9.5,9.6) reduces to standard soft clustering (Rose et al. (1990, 1992); Karayiannis (1999)).

9.3 Structural Transitions in Pairwise Clustering

It is known from previous studies on central (Rose et al. (1990, 1992); Rose (1998)) and pairwise (Hofmann and Buhmann (1997)) clustering, that the structure of the clustering solution depends on the value of the Lagrange parameter β which weighs the contribution of the entropy term in the Lagrangian against the constraint. For

small values of β the number of clusters is small. Increasing β, i.e. increasing the importance of the constraint in eq. (8.7), leads to an increased number of groups, and finally approaches the number given by the cardinality of the set $\mathcal{C}$. It has been shown for central clustering that these structural transitions occur at so-called critical values of β which are connected with a bifurcation (Rose (1998)). A similar behavior is to be expected in connection with the pairwise clustering methods suggested here.

The stability of a clustering solution for a given value of β can be assessed by a linear stability analysis. I abbreviate the term in the exponent of eqs. (9.4,9.6) by $\mathcal{E}_{ls}^{\beta} = -\beta\mathcal{E}(l,s)$, and linearize eq. (9.5) around $\mathcal{E}_{mt}^{\beta(t)}$ using a Taylor-expansion to the first order:

$$\begin{aligned}\mathcal{E}_{ls}^{\beta} - \mathcal{E}_{ls}^{\beta(t)} &= \sum_{m=1}^{M}\sum_{t=1}^{N}\left[\frac{\partial\mathcal{E}_{ls}^{\beta}}{\partial\mathcal{E}_{mt}^{\beta}}\right]_{\mathcal{E}_{mt}^{\beta(t)}}\left(\mathcal{E}_{mt}^{\beta} - \mathcal{E}_{mt}^{\beta(t)}\right), \\ &= \beta\sum_{m=1}^{M}\sum_{t=1}^{N}(\Gamma)_{mt}^{ls}\left(\mathcal{E}_{mt}^{\beta} - \mathcal{E}_{mt}^{\beta(t)}\right), \end{aligned} \tag{9.7}$$

where the coefficients are given by (for details of the derivation see appendices B.1 and B.2)

$$\begin{aligned}\beta\,(\Gamma_I)_{mt}^{ls} &= \frac{\partial\mathcal{E}_{ls}^{\beta}}{\partial\mathcal{E}_{mt}^{\beta}} \\ &= -\beta\,P(x_t|l)(\delta(l,m) - P(m|x_t))\left[d(t,s) - \sum_j P(x_j|l)d(t,j)\right] \\ &\quad + \beta\sum_i\sum_k \frac{P(k|x_i)}{P(k)}\frac{\partial P(k)}{\partial\mathcal{E}_{m,t}^{\beta}}P(x_i|l) \\ &\qquad\qquad \times\left[d(i,s) - \sum_j P(x_j|l)d(i,j)\right] \end{aligned} \tag{9.8}$$

for the mutual information based cost function, eq. (8.7), and

$$\begin{aligned}\beta\,(\Gamma_H)_{mt}^{ls} &= \frac{\partial\mathcal{E}_{ls}^{\beta}}{\partial\mathcal{E}_{mt}^{\beta}} \\ &= \beta P(x_t|l)(\delta(l,m) - P(m|x_t))\left[\sum_i P(x_i|l)d(i,s) - \right. \\ &\quad \left.\sum_{i,j} P(x_i|l)P(x_j|l)d(i,j) - d(t,s) + \sum_j P(x_j|l)d(t,j)\right] \end{aligned} \tag{9.9}$$

for the entropy based cost function. In the following I suppress the indices and use the notation Γ for both cases when appropriate.

Equation (9.7) can be decoupled by transforming the vector

$$\Delta\mathcal{E}_{ls} = \mathcal{E}_{ls}^{\beta} - \mathcal{E}_{ls}^{\beta(t)}$$

into the eigenbasis of Γ. After the transformation we obtain

$$\Delta\tilde{\mathcal{E}}_{ls} = \beta\Gamma\Delta\tilde{\mathcal{E}}_{ls},$$

where $\tilde{}$ denotes the quantities in the new coordinate system. Let $\lambda_{\max}$ be the largest eigenvalue of Γ. If $\beta\lambda_{\Gamma}^{\max}$ is smaller than one, the solution is stable. Marginal stability - which indicates a structural change - occurs if:

$$\beta = \frac{1}{\lambda_{\Gamma}^{\max}}. \tag{9.10}$$

Figure 9.1 illustrates this result. 80 two-dimensional data vectors were generated using a mixture of Gaussian density with four components (fig. 9.1a). A dissimilarity matrix based on the two-dimensional Euclidean distances was calculated and used as input for the pairwise clustering method. The cardinality of C is four. For every value of β (see legend of fig. 9.1 for details) assignment probabilities were calculated using the fixed point iteration, eqs. (9.4,9.5), and the maximum eigenvalue $\lambda_{\Gamma}^{\max}$ of the matrix Γ_I was determined. Figure 9.1 b shows the averaged difference between the stationary assignment probabilities for neighboring values of β. Structural changes are reflected by large differences, which indeed coincide with the values of β for which the condition eq. (9.10) of marginal stability is fulfilled (see fig. 9.1 b shows the averaged difference between the stationary e). For the data set of fig. 9.1, which is characterized by four nicely separated components, the sequence of structural changes with increasing value of β is given by: one effective cluster (clusters 1 through 4) $\rightarrow$ two effective clusters (clusters 1, 2 vs. clusters 3, 4) $\rightarrow$ three effective clusters (clusters 1, 2 vs. cluster 3 and cluster 4) $\rightarrow$ four effective clusters. This sequence of events is similar to what is observed for central clustering (Rose et al. (1993)). The clusters with the larger variance of the assigned data points become unstable at smaller values of β. Figures 9.1c,d,f-h illustrate this result. The structural changes for pairwise clustering can be detected by calculating the differences between averaged values of the stationary assignment probabilities: the value of β at which these differences become large and positive indicates the critical value.

The critical values β_c determined from figs. 9.1 b,c nicely match the values $\beta_c = 1/\lambda_{\Gamma}^{\max} = 0.3005,\ 1.0447,\ 3.2347$, which can be directly computed using a stationary state for a value of β for which the averaged difference between the assignment probabilities for neighboring values of β is close to zero. Here I computed Γ and β_c for values of β for which $< |P(m|x)(t) - P(m|x)(t)| >= \frac{1}{N\times M}\sum_{m,x}|P(m|x)(t) - P(m|x)(t)| < 0.01$ (β were $0.1653, 0.5706$ and 1.4799). However, these kinds of stationary states may not exist for arbitrary distributions of data objects. For these cases, fig. 9.1e suggests another approach. Because the graph of $1/\lambda_{\Gamma}^{\max}$ vs. β is concave and always larger than β between the critical values β_c, the values of β_c can be determined via the fixed point iteration

$$\beta(t+1) \ = \ \frac{1}{\lambda_{\Gamma}^{\max}(\beta(t))}. \tag{9.11}$$

Figure 9.2 shows the result. It takes only seven iterations instead of 38 iterations (fig. 9.1e) to find all three critical points. Note, however, that it may be computationally expensive to calculate the eigenvalues of Γ if the matrix is large.

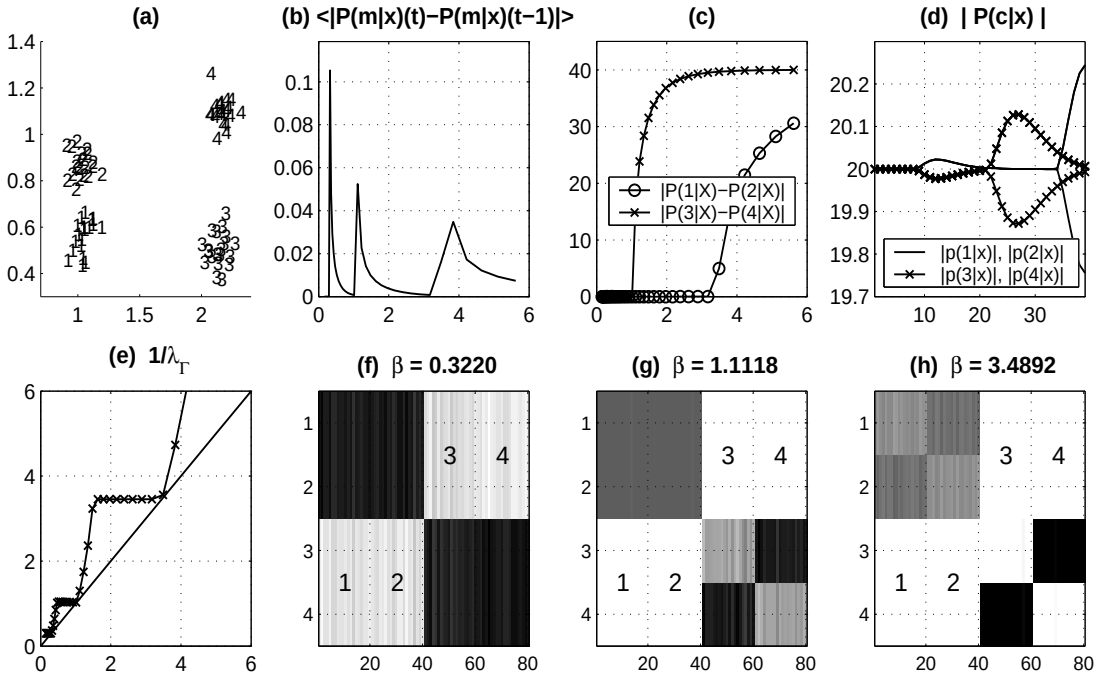


Figure 9.1: Structural changes and phase transitions for the pairwise clustering method. (a) Two-dimensional toy data set generated by a mixture of Gaussian with four components of equal strength, equal variance $\sigma_C^2 = 0.005$, and centroids $\theta_1 = (\,1,\, 0.55\,)$, $\theta_2 = (\,1,\, 0.95\,)$, $\theta_3 = (\,2.1,\, 0.5\,)$ and $\theta_4 = (\,2.1,\, 1.1\,)$. Location of the numbers indicate the locations of the generated data points and the numbers themselves denote the number of the component of origin. 20 data points were generated for every component. An 80×80 dissimilarity matrix was calculated using the normalized squared distance $d(i,j) = (x_i - x_j)^2/(2\sigma_D^2)$ with variance of data distribution σ_D^2 and used as input for the pairwise clustering method. The cardinality of $\mathcal{C}$ was four. (b) Average difference $< |P(m|x)(t) - P(m|x)(t-1)| >= \frac{1}{N\times M}\sum_{x,m} |P(m|x)(t) - P(m|x)(t-1)|$ between the total sums of the converged assignment probabilities $P(c|x)(t)$ for $\beta(t)$ and $P(c|x)(t-1)$ for $\beta(t-1)$ as function of β. Values for β were generated according to $\beta_0 = 0.1502$, $\beta(t) = 1.1 \times \beta(t-1)$, $\beta \leq 6$. (c) Difference between pairs of assignment probabilities averaged over all data points, $\sum_i |P(1|x_i) - P(2|x_i)|$ and $\sum_i |P(3|x_i) - P(4|x_i)|$, as a function of β. (d) Sums $\sum_x P(l|x)$ of the assignment probabilities as a function of β. (e) Values of $\frac{1}{\lambda_{\Gamma_H}}$ as a function of β. Points for which $\frac{1}{\lambda_{\Gamma_H}}$ (crosses) equals β (solid line) indicate values of β for which structural changes occur. (f)–(h) Assignment probabilities $P(l|x)$ for all pairs l and x for values of β which are slightly larger than the critical values shown in (e). Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. Component i of the mixture generated data points with numbers between $20 \times (i-1) + 1$ and $20 \times i$. Numbers written into the figures (f)-(h) refer to the numbers of the components which generated the corresponding data points.

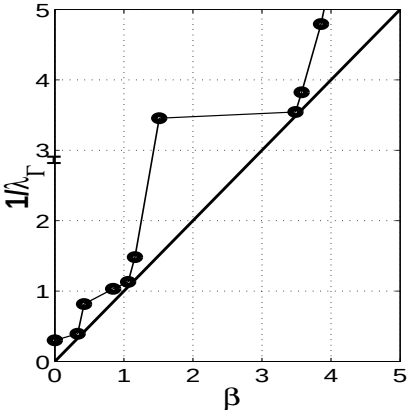


Figure 9.2: Result of the fixed point iteration, eq. (15), for the matrix Γ_H and for the toy data set shown in fig. 9.1.
Solid circles denote the values of $(\beta, 1/\lambda_\Gamma^{\max})$ for the different iteration steps. The fixed point iteration always converges to the critical value of β to the right of its initialization value. For every fixed point iteration, the initial value of β was taken to be slightly larger (here 0.03) than the previously calculated critical value.

9.4 Optimization Using Deterministic Annealing

It has been shown for the previously proposed central (Rose et al. (1992)) and pairwise (Hofmann and Buhmann (1997)) clustering methods, that annealing in the inverse temperature during the optimization of the cost function improves the clustering result. Since the Lagrange multiplier β, which enforces the constraint $E = E_0$, formally corresponds to the inverse temperature in Rose et al. (1992) and Hofmann and Buhmann (1997), one would expect that annealing in the parameter β would also improve the solution to the optimization problem given by eq. (9.3). For $\beta \to \infty$, $P(l|x_s)$ converges to $\delta(l, q_s)$, where $q_s = \arg\min_r \mathcal{E}(r, s)$, and a so-called hard clustering method is obtained. In this limit, the Lagrangian, eq. (9.3), is dominated by the constraint and the complexity of the representation is given by the cardinality of the set $\mathcal{C}$ of clusters.

Figure 9.3 shows the results of an optimization using deterministic annealing for a toy example. 70 two-dimensional data vectors were generated using a mixture of Gaussian density with five components consisting of three 'main' (nos. $3-5$) and two 'outlier' (nos. $1, 2$) components (fig. 9.3 a). A dissimilarity matrix was calculated using squared Euclidean distances and used as input for the pairwise clustering method. The cardinality of C was ten. Annealing was started at an inverse temperature slightly below the first critical value for both entropy based and mutual information based cost functions. Figs. 9.3 b and c show the final assignment probabilities for high values of β. In both cases, the clustering solution consists of four effective clusters only, because the data points which correspond to the two outlier components (nos. 1 and 2) are assigned to one group. The actual values of the assignment probabilities are different because the elements from the set $\mathcal{C}$ are grouped differently. This is a result of the different cost functions. Because

$$H(C|X) \;=\; H(C) - I(C;X) \tag{9.12}$$

holds, maximizing the entropy $H(C|X)$ is equivalent to simultaneously minimizing the mutual information $I(C;X)$ and maximizing the cluster entropy $H(C)$,

$$\begin{aligned} H(C) &= -\sum_c \frac{1}{N} \sum_x P(c|x) \log \frac{1}{N} \sum_x P(c|x) \\ &= -\sum_c P(c) \log P(c) \end{aligned}$$

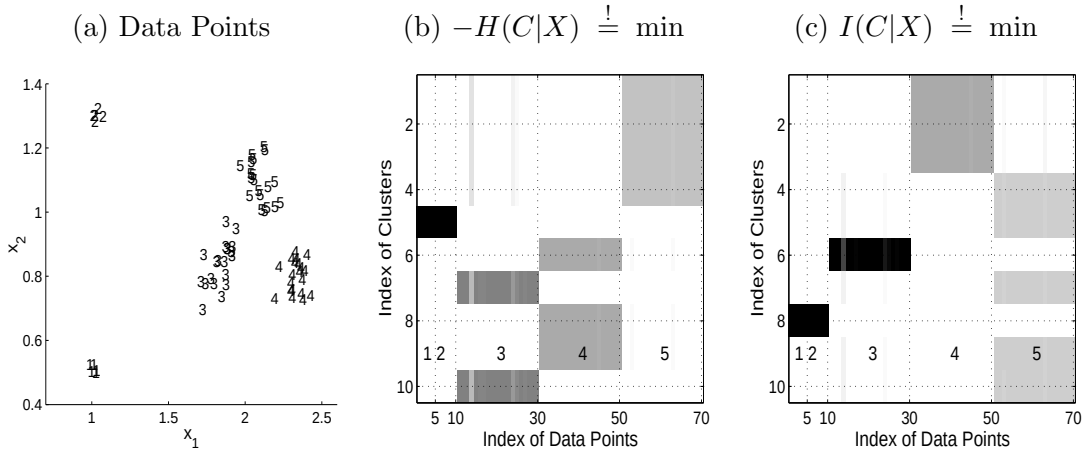


Figure 9.3: Results of an optimization using deterministic annealing.
(a) Two-dimensional toy data set generated by a mixture of Gaussian with five components. The location of the centroids and the variances of the components were: $\theta_1 = (1, 0.5)$, $\theta_2 = (1, 1.3)$, $\theta_3 = (1.8, 0.85)$, $\theta_4 = (2.3, 0.8)$ and $\theta_5 = (2.1, 1.1)$, $\sigma_1^2 = \sigma_2^2 = 0.0005$, $\sigma_3^2 = \sigma_4^2 = \sigma_5^2 = 0.005$. Location of numbers indicate the locations of generated data points and its values denote the number of the component of origin. 5 data points were generated for the components 1 and 2, 20 data points for each of the remaining components. A dissimilarity matrix of size 70×70 was calculated using the squared Euclidean distance $d(i,j) = 0.5 \times (x_i - x_j)^2$ and used as input for the pairwise clustering method. The cardinality of $\mathcal{C}$ was 10. (b and c) Assignment probabilities $P(l|x)$ for all pairs l and x using a cost function based on the negative entropy (b) and the mutual information (c). Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. Components 1 and 2 of the mixture generated data points with numbers $1-5$ and $6-10$, components 3 to 5 generated data points with numbers $11-30$, $31-50$, and $51-70$. Numbers written into the figure refer to the numbers of the components which generated the corresponding data points. The value of β was 112.33 for (b) and 112.01 for (c). The annealing schedule was $\beta(0) = 5.0549$ for (b) and $\beta(0) = 5.0405$ for (c), $\beta(t+1) = 1.1 \times \beta(t)$. The first critical value of β was 5.6165 for the entropy based and 5.6005 for the mutual information based cost functions. The values for $I(C;X)$, $H(C|X)$, and $< d(x,x') >$ were $1.8742, 1.3975, 0.0296$ for (b) and $1.8863, 1.1924, 0.0294$ for (c).

is large, if the probabilities of clusters are similar to each other.

Figure 9.3 shows that deterministic annealing without further modification leads to a suboptimal clustering result, because the two outlier components are always grouped together. The problem of the convergence to a local optimum of the Lagrangian remains also for sets $\mathcal{C}$ with larger cardinality (for example 20 or 30). The reason for this behavior is that elements of the set $\mathcal{C}$ split along the direction of high variance of the data, hence often along the direction of outlier groups, but without any control of the distribution of the elements of $\mathcal{C}$ across the effective clusters. For the data set of fig. 9.3 a, one element of $\mathcal{C}$ splits at the lowest critical value of β and assumes responsibility for both components 1 and 2. Since this effective cluster is represented by one element only, no further splits can occur and the algorithm converges to a local optimum of the cost function.

9.5 Optimization Using an Incremental Splitting Method

In order to prevent the convergence to local optima I propose a deterministic annealing scheme, where the cardinality of the set $\mathcal{C}$ is not fixed, but increases with every structural change of the clustering solution. Annealing starts with a set $\mathcal{C}$ of two elements and with a value of β below the lowest critical value. At the critical value two effective clusters form, and the elements of $\mathcal{C}$ are duplicated. This procedure is repeated: whenever one of the effective clusters becomes unstable, the corresponding elements of $\mathcal{C}$ are duplicated and the cardinality of $\mathcal{C}$ increases by two.

In analogy to incremental splitting methods for central clustering (Rose et al. (1993)) I consider the normalized distance $D(l, l')$,

$$D(l, l') \;=\; \frac{\sum_x |P(l|x) - P(l'|x)|}{\sum_x |P(l|x) + P(l'|x)|}, \tag{9.13}$$

between two elements l and l' which belong to the same effective cluster. As long as l and l' belong to the same cluster, $D(l, l')$ is close to zero. If, however, a structural change occurs, the measure assumes a finite value, and the elements l and l' should be duplicated. Here I propose to duplicate l and l', if $D(l, l')$ exceeds a predefined threshold ξ.

Figure 9.4 shows an application of the incremental splitting method to the data set shown in fig. 9.1 a using the maximum entropy based cost function. The figures show the assignment probabilities after the first (left), second (center), and third (right) split. The method is able to robustly find the different components. Similar results were obtained with the cost function based on minimizing the mutual information. Figure 9.5 shows the results for the data set described in fig. 9.3 a. After four splitting events, both methods to reliably detect all five components, and the sequence of splits provides insight into the hierarchical structure of the data set. Here the threshold ξ was set to a larger value in order to induce a split only after the initially rapid change of the stationary assignment probabilities with β.

I then compared the values for the mutual information $I(C; X)$, the entropy $H(C|X)$, and the average distortion $< d(x, x') >$ for the clustering solutions obtained with and without the incremental splitting method for the results shown in fig. 9.3. The results are summarized in table 9.1. The table shows, that the clustering solutions obtained

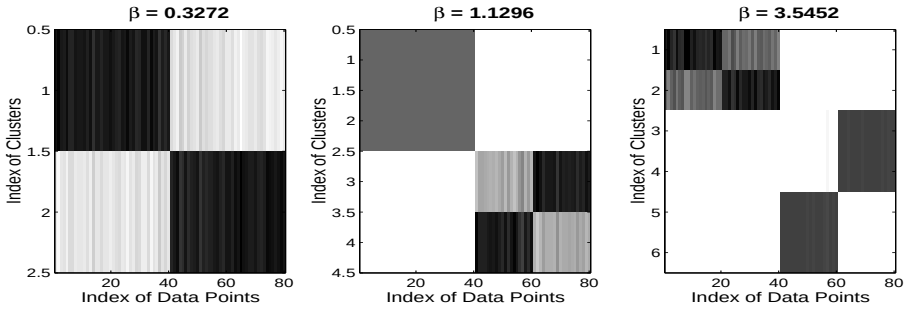
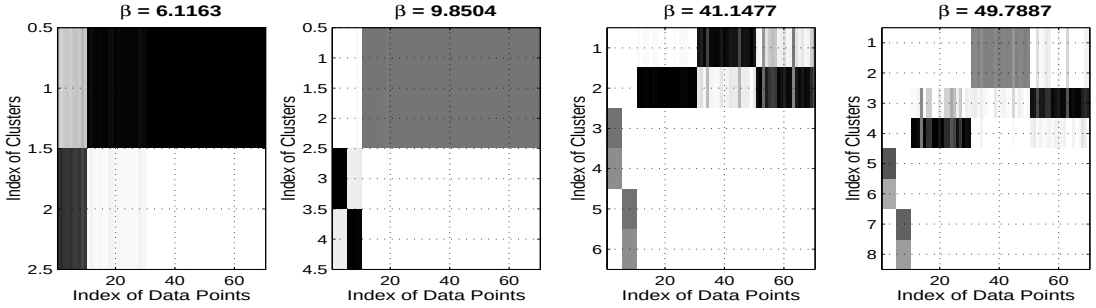
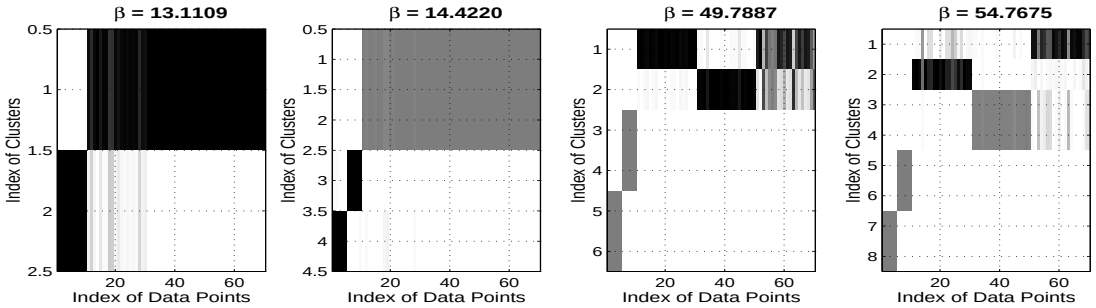


Figure 9.4: Deterministic annealing with the incremental splitting method
The Experiment is applied to the data set shown in fig. 9.1(a) for the maximum entropy based cost function. The figures show the assignment probabilities after the first (left), second (center), and third (right) split. Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. ξ was set to 0.1. The annealing schedule was $\beta_0 = 0.2704$, $\beta(t+1) = 1.1 \times \beta(t)$.



(a) Clustering solution for the mutual information based cost function.



(b) Clustering solution for the entropy based cost function.

Figure 9.5: Deterministic annealing with the incremental splitting method applied to the data set shown in the fig. 9.3 a. The figures show the assignment probabilities after the first, second, third, and fourth (from left to right) split for the cost functions based on the mutual information (a) and the entropy (b). Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. ξ was set to 0.8. The annealing schedule was $\beta_0 = 5.6165$ for (a), $\beta_0 = 5.6005$ for (b), and $\beta(t+1) = 1.1 \times \beta(t)$ for both.

clustering method	β	$< d(x,x') >$	$I(C;X)$	$H(C\|X)$	$\mathcal{R}$
eqs. (9.4,9.5), (PCRD)	49.5359	0.0364	1.5583	1.5537	5.8136
eqs. (9.4,9.5), (PCRD-S)	49.5359	0.0142	1.7012	0.8264	2.9473
eqs. (9.6,9.5), (PCM)	54.7675	0.0350	1.3752	1.6151	0.3018
eqs. (9.6,9.5), (PCM-S)	54.7675	0.0133	0.8106	1.7451	-1.0167

Table 9.1: Characterization of the clustering solution by the values of $I(C;X)$, $H(C|X)$, $< d(x,x') >$ and the cost function $\mathcal{R}$ for the results shown in figs. 9.3 and 9.5. $\mathcal{R}$ is $I(C;X)+\beta < d(x,x') >$ for the **P**airwise **C**lustering based on **R**ate **D**istortion theorem without (PCRD) and with the **S**plitting method (PCRD-S), and $-H(C|X) + \beta < d(x,x') >$ for the **P**airwise **C**lustering using **M**eanfield approximation without (PCM) and with the **S**plitting method (PCM-S). For the last column, β is interpreted as a regularization parameter rather than a Lagrange multiplier.

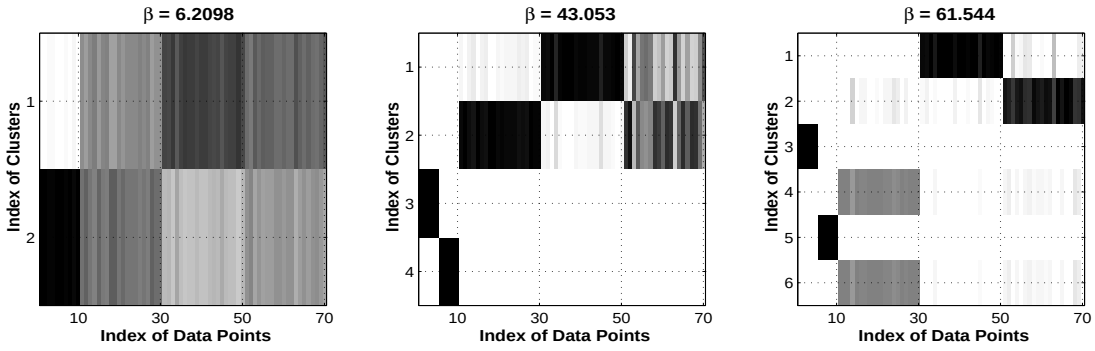


Figure 9.6: Incremental splitting method combined with the fixed point iteration, eq. (15), and applied to the data set shown in fig. 9.3 a.
The figures show the assignment probabilities after the first (left), second (center), and third (right) split. Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. The data is shown for the entropy based cost function. The values for $H(C|X)$ and $< d(x,x') >$ were 0.5515 and 0.011 ($\beta = 61.5440$).

by the incremental splitting method are characterized by lower values of the cost. Figure 9.6 shows results for the data set shown in fig. 9.3 a obtained by combining the incremental splitting method with the fixed point iteration (cf. fig. 9.2) for determining the critical values of β. Only six annealing steps are needed to obtain the desired result (fig. 9.6, right). This is much less than the number of steps needed for regular deterministic annealing.

9.6 Preferred Implementation of Pairwise Clustering

Initialization

1. Initialization of the assignment probabilities: $P(l|x_i) = 0.5 + \eta$, $l = 1,2$, $i \in \{1, \ldots, N\}$, $\eta \sim \mathcal{N}(0, 0.01)$.
2. Set maximum number of elements, M_F, and initialize the number of elements, $M_I = 2$. Set the hyper-parameters: $0 < \xi < 1$, $0 < \alpha < 1$.
3. Compute the critical value λ_Γ for the first split according to eqs. (9.4,9.5) or (9.5,9.6) and set $\beta = \alpha/\lambda_\Gamma$.

Repeat

1. **Repeat** Calculation of $P(l|x_i)$, $P(x_i|l)$, and $P(l)$ according to eqs. (9.1), (9.4,9.5), or (9.5,9.6)
   Until $\{P(l|x_i)\}$ converge.
2. **If** there is a cluster l with $D(l, l+1) > \xi$ **then**
   (a) Split cluster $q = \arg\max_l \{l | D(l, l+1) > \xi\}$ into two clusters.
   Set $h(1|x) = P(q|x) + \epsilon$ and $h(2|x) = P(q+1|x) + \epsilon$ for all x, where ϵ is drawn randomly and uniformly from $[\,0,\, 0.02\,]$.
   (b) Shift the cluster indices:
   $p \leftarrow p + 2$, $\forall\, p > q + 1$.
   (c) Update the assignment probabilities:
   $P(q+2|x) = P(q+1|x)$, $P(q+1|x) = h(1|x)$, $P(q+3|x) = h(2|x)$.
   (d) $M \leftarrow M + 2$
3. $\beta \leftarrow 1.1 \times \beta$ (standard annealing) or $\beta \leftarrow 1.1 \times 1/\lambda_\Gamma$ (fixed point iteration).

Until $M = M_F$

Repeat Calculation of $P(l|x_i)$, $P(x_i|l)$, and $P(l)$ according to eqs. (9.1), (9.4,9.5) or (9.5,9.6)

Until $\{P(l|x_i)\}$ converge.

For the fixed point iteration, Γ is computed on the converged assignment probabilities for which $D(l, l+1) \geq \xi$. The factor 1.1 is used in order to automatically 'jump' across the critical values of β when close.

9.7 Embedding and Visualization of Pairwise Data

Let us again consider a set $\mathcal{X}$ of N objects x_i, which are characterized by a matrix $\mathbf{D}$ of their pairwise dissimilarities $d(i,j),\ i,j = 1,\ldots,N$ (see fig. 8.1 b). In order to solve eq. (8.7) under the scheme depicted in fig. 8.3 a, the read-out process has to be specialized and a distortion measure has to be constructed. Adapting the arguments from section 9.2 I choose the optimal read-out process given by Bayes rule

$$P_2(x_j|d) = \frac{P_2(d|x_j)P(x_j)}{q(d)} = \frac{\sum_c P_1(c|x_j)q(d|c)}{Nq(d)}, \tag{9.14}$$

where

$$\begin{aligned} q(d) &= \sum_c q(d|c)P(c), \\ P(c) &= \sum_i P_1(c|x_i)P(x_i) = \frac{1}{N}\sum_i P_1(c|x_i), \end{aligned} \tag{9.15}$$

and where I assigned a weight $P(x_i) = 1/N$ to every data point of the set $\mathcal{X}$. The average distortion is then given by

$$E = < d(x,x') > = \frac{1}{N}\sum_{i,j}\sum_{l,m} P_2(x_j|l)q(l|m)P_1(m|x_i)d(i,j), \tag{9.16}$$

Eqs. (8.7) and (9.16) state that the optimal representation $\left\{P_1(c|x)^{\text{opt}}\right\}$ should contain the minimal amount of information about $\mathcal{X}$ under the constraint, that the average distortion between an object and its optimal reconstruction should have a fixed (small) value.

The optimization problem is solved using the method of Lagrange multipliers and Lagrange functional is given by

$$\mathcal{L}_1 = I(C;X) + \beta\left(< d(x,x') > - E_0\right) + \sum_i \lambda_i \left(\sum_n P_1(n|x_i) - 1\right). \tag{9.17}$$

By setting the derivative of the Lagrangian w.r.t. $P_1(c|x)$ to zero (see appendix B.3) we obtain

$$P_1(l|x_t) = \frac{\exp(-\beta\mathcal{E}(l,t))P(l)}{\sum_n \exp(-\beta\mathcal{E}(n,t))P(n)}, \tag{9.18}$$

$$\mathcal{E}(l,t) = \sum_m q(m|l)\sum_i P_2(x_i|m)\left[d(i,t) - \frac{1}{2}\sum_j P_2(x_j|m)d(i,j)\right], \tag{9.19}$$

which can be solved via fixed point iteration. The normalization condition $\sum_n P_1(n|x_k) = 1,\ \forall\, k$, is fulfilled if $\exp(\lambda_t N) = \sum_n \exp\left(-\beta\mathcal{E}(n,t)\right)P(n)$, and positivity of the $P_1(l|x)$ is assured by the exponential function in eq. (9.18).

The first convolution in the equation in eq. (9.19) is usually omitted in standard Self-Organizing Map methods. This makes the method computationally more efficient, but has the drawback that eqs. (9.18) may not converge to an optimum of eq. (8.7) in all circumstances. If $q(l|m) = \delta(l,m)$ everywhere, then the pairwise clustering method

of section 9.2 is obtained. For $\beta \to \infty$ a 'hard' clustering method is obtained. In this limit, the Lagrangian, eq. (9.17), is dominated by the constraint, and the complexity of the representation is given by the size of the set $\mathcal{C}$ of clusters.

Eqs. (9.18) and (9.19) are related to the STMP approach of Graepel and Obermayer (1999). This method is obtained, if the mutual information $I(X;C)$ in the cost function eq. (8.7) is replaced by the negative conditional entropy $-H(C;X)$, similar to the case of pairwise clustering (section 9.2). Again both methods differ by the factor $P(l)$ in the eq. (9.19). Since the Lagrange multiplier β, which enforces the constraint $E = E_0$, formally corresponds to the inverse temperature in Graepel and Obermayer (1999), I expect that annealing of the parameter β would also improve the solution of the optimization problem.

Figure 9.7 shows the results of an optimization of the clustering and embedding problem for pairwise data on a toy example. 200 3D data vectors were generated using a 'noisy spiral' mixture of Gaussian densities (fig. 9.7 a). Ten mixture components with equal variance were centered equidistantly on a noisy spiral backbone given by $x_1 = \sin(\theta)$, $x_2 = \cos(\theta)$, and $x_3 = \theta/\pi$. A different number of data points were then randomly drawn from every component (for detail see legend of fig. 9.7). A dissimilarity matrix was then calculated using squared Euclidean distances and was used as input data for a one dimensional Self-Organizing Map with ten units. The neighborhood function $q(d|c)$ was a standard normalized Gaussian function

$$q(d|c) = \frac{\exp\left(-(d-c)^2/\sigma_q^2\right)}{\sum_r \exp\left(-(d-r)^2/\sigma_q^2\right)}. \tag{9.20}$$

Optimization using the mutual information based cost function was performed using the fixed point iteration (eq. 9.18,9.19) and standard deterministic annealing. Figure 9.7c shows the final assignment probabilities for the Self-Organizing Map in comparison to the solution obtained for standard pairwise clustering (fig. 9.7 b, $q(d|c) = \delta(d,c)$ in eq. 9.19). The Self-Organizing Map indeed finds the one dimensional structure of the noisy spiral, which was hidden in the dissimilarity matrix used as input. Moreover, the ten components of the mixture model were found and almost all data points were correctly assigned. This is different to compared to the clustering solution generated by the PCRD method, where the 'smaller' components 7 and 8 were merged into one cluster (5), leaving cluster 9 almost empty. Minimizing the entropy based cost function led to qualitatively similar results. While the incremental splitting method of section 9.5 must be applied in order to improve the representation for PCRD and PCM, deterministic annealing is sufficient in the case of the Self-Organizing Map. Due to the coupling of clusters through the neighborhood function $q(d|c)$ clusters may be reassigned to different components of the data during optimization, and the problems similar to the ones encountered in section 9.4 do not arise.

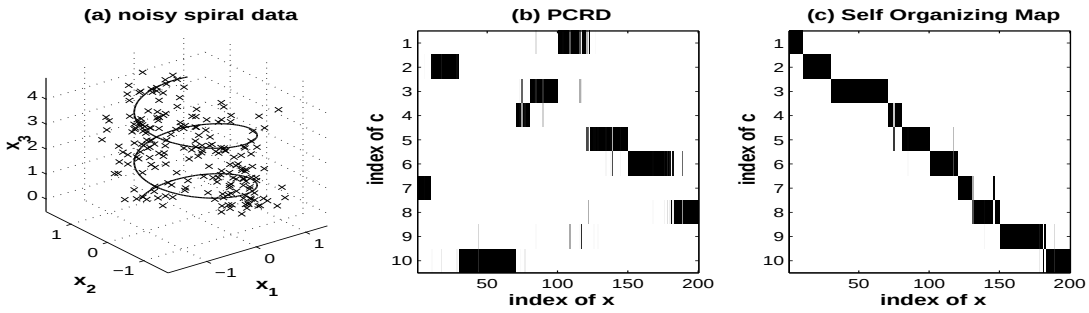
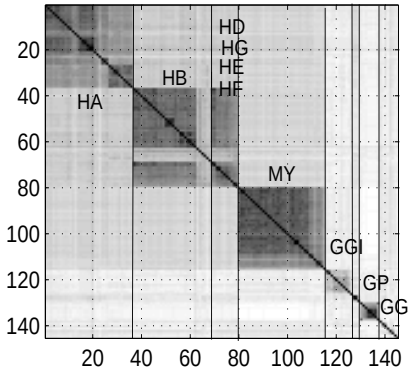


Figure 9.7: Self-Organizing Maps (mutual information based cost function) and PCRD method applied to the noisy spiral data set.
(a) Three-dimensional toy data set generated by a 'noisy spiral' mixture of Gaussian with ten components of equal variance 0.2. The mixture components were centered equidistant along a noisy spiral backbone (solid line) given by $x_1 = \sin(\theta)$, $x_2 = \cos(\theta)$ and $x_3 = \theta/\pi$. The centroids were located at $\theta_i = (i-1) \times \frac{4}{\pi}$ for the components $i = 1, \ldots, i = 10$. The number of data points (crosses) drawn randomly from the individual components were {10, 20, 40, 10, 20, 20, 10, 20, 30, 20}. A dissimilarity matrix of size 200×200 was calculated using the squared Euclidean distance $d(i,j) = 0.5 \times (x_i - x_j)^2$. The cardinality of $\mathcal{C}$ was ten. (b) Results for the PCRD pairwise clustering method. (c) Results for the pairwise Self-Organizing Map. The assignment probabilities are shown for all pairs c (nos. along vertical axis) and x (nos. along horizontal axis). Black $\rightarrow$ wite indicates large $\rightarrow$ small probabilities. The final value of β was 4.6425 for (b) and 4.7836 for (c). The assignment probabilities were initialized using uniform random number from the interval $[\,1,\,2\,]$ and then normalized. The annealing schedule was $\beta(0) = 0.2382$ for (b) and $\beta(0) = 0.794$ for (c), and $\beta(t+1) = 1.1 \times \beta(t)$ for both. For every new value $\beta(t+1)$ a small random number was added to the assignment probabilities before fixed point iteration was applied: $P_1(c|x)^{ini}(t+1) = P_1(c|x)^{conv.}(t) + \eta$, η drawn randomly from a uniform distribution between 0 and 0.01 for (b) and between 0 and 0.5 for (c). $P_1(c|x)^{ini}(t+1)$ were normalized before fixed point iteration. Convergence criterion for the fixed point iteration was $\frac{1}{N \times M} \sum_{c,x} \| P_1(c|x)^{old} - P_1(c|x)^{new} \| < 10^{-15}$, where N is the number of data points and M is number of the clusters. σ_q^2 was 0.3.

(a) Dissimilarity Matrix



(b) Membership of the Proteins

1-36	37–68	69–70	71–72
HA	HB	HD	HG
73–77	78	79	80–115
HE	HF	HE	MY
116–126	127-128	129	130,137
GGI	GGL	GGH	GPY
131	132	133–136	138-145
GPV	GPF	GPS	GG

Figure 9.8: The protein data set.
(a) Dissimilarity matrix for 145 proteins from the globin family. Bright and dark values indicate large and small values. (b) The table indicates the membership of the proteins in one of the protein families as it has been determined by experts. HA: hemoglobin α, HB: hemoglobin β, HD: hemoglobin δ, HE: hemoglobin ϵ, HG: hemoglobin γ, HF: hemoglobin F, MY: myoglobin, GGI–GG: other globin families.

9.8 Application to Real World Data

9.8.1 Protein Data Set

I applied deterministic annealing and the incremental splitting method to a dissimilarity matrix for 145 protein sequences. The sequences belong to globin proteins from different protein families like hemoglobin or myoglobin. The dissimilarity values between pairs of sequences are determined by a sequence alignment program which took biochemical and structural information into account (Mevissen and Vingron (1996)). Figure 9.8 shows the dissimilarity matrix for this data set. Every number corresponds to one protein, and the short abbreviations in capital letters denote the different globin families. Pairwise clustering was then applied to the dissimilarity matrix.

Figure 9.9 shows the assignment probabilities for the following variants of the pairwise clustering method: (*i*) minimization of the mutual information vs. maximization of the entropy and (*ii*) standard deterministic annealing vs. the incremental splitting method. The variants PCRD and PCRD-S (see legends of table 9.1 and fig. 9.9 for abbreviations) both lead to one cluster for the hemoglobin group (HA-HE), one and two clusters for the myoglobin (MY), and five and four clusters for the remaining groups (GGI-GG). A comparison of PCRD-S and PCM-S show a similar cluster structure for HA, myoglobin, and the group 'other globins'. PCM, however, groups the 'other globins' into only three clusters and the group HB into two. Again, the incremental splitting methods generates cluster structures with lower costs compared to the standard annealing methods (see table 9.2) and the average distortion is reduced. The two variants PCM and PCM-S lead to solutions with a smaller average distortion than the variants PCRD and PCRD-S which minimize the mutual information between clusters and data.

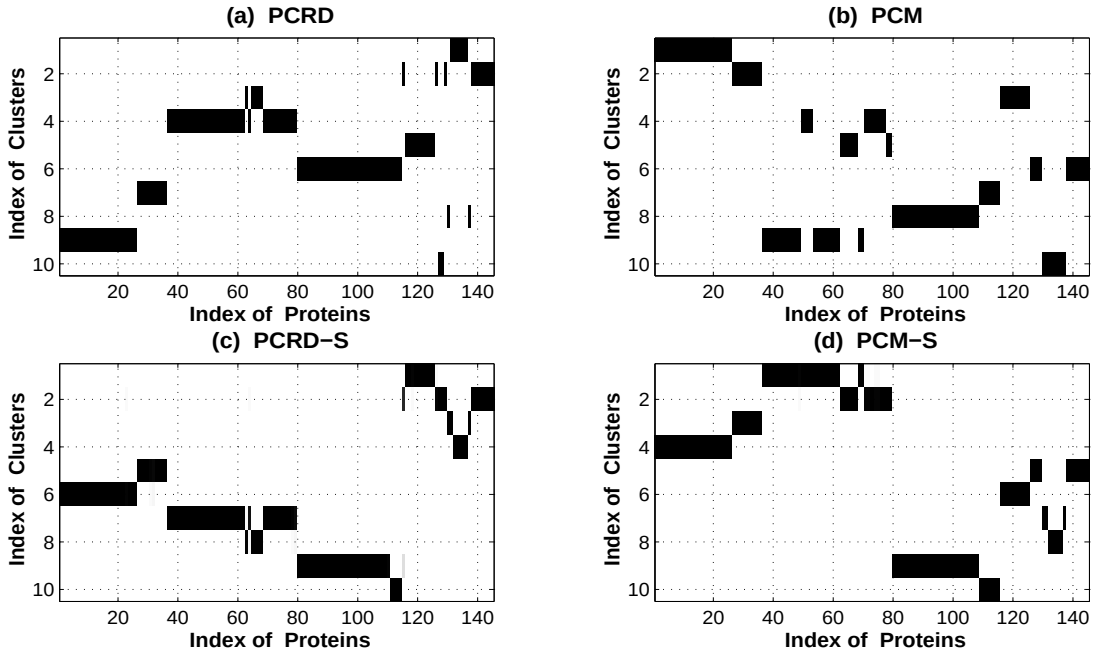


Figure 9.9: Pairwise clustering results for the protein data set.
The figures show the assignment probabilities after annealing for four variants of the pairwise clustering method: (a) mutual information based cost function and standard deterministic annealing (PCRD), (b) entropy based cost function and standard deterministic annealing (PCM), (c) mutual information based cost function and the incremental splitting method (PCRD-S), and (d) entropy based cost function and the incremental splitting method (PCM-S). Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. The annealing schedule was: $\beta(0) = 0.9/\lambda_\Gamma$, where $1/\lambda_\Gamma$ is the critical value of β for the first split ($1/\lambda_\Gamma$ was 5.6005 for (a)(c) and 5.6165 for (b)(d)) and $\beta(t+1) = 1.3 \times \beta(t)$. ξ was set to 0.8.

clustering method	β	$< d(x,x') >$	$I(C;X)$	$H(C\|X)$	$\mathcal{R}$
eqs. (9.4,9.5), (PCRD)	3.8697	3.8885	2.6342	0.0454	17.6816
eqs. (9.4,9.5), (PCRD-S)	3.8697	3.8045	2.8237	0.0406	17.5460
eqs. (9.6,9.5), (PCM)	6.5398	3.7752	0.0415	3.1078	24.6477
eqs. (9.6,9.5), (PCM-S)	6.5398	3.7609	0.0156	3.0142	24.5800

Table 9.2: Characterization of the clustering solution by the values of the cost function and the values of $I(C;X)$, $H(C|X)$, $< d(x,x') >$, and the cost function $\mathcal{R}$ for the results shown in fig. 9.9. $\mathcal{R}$ is $I(C;X) + \beta < d(x,x') >$ for PCRD and PCRD-S, and $-H(C|X) + \beta < d(x,x') >$ for PCM and PCM-S (see legends of table 9.1 and fig. 9.9 for abbreviation). For the last column β is interpreted as a regularization parameter rather than a Lagrange multiplier.

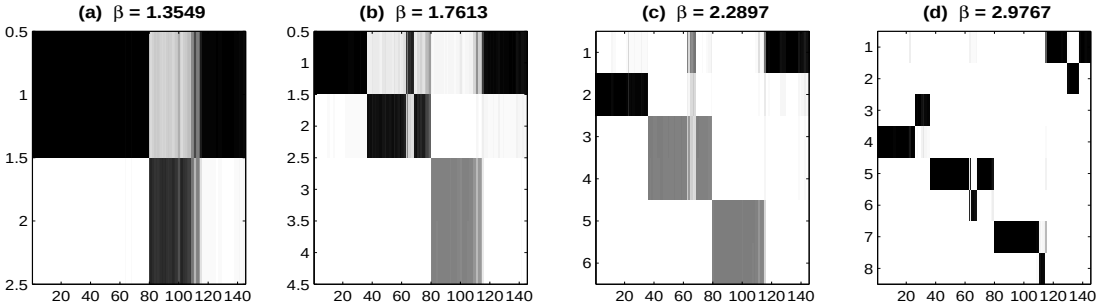


Figure 9.10: Incremental Splitting Method applied on the Protein Data in fig.9.9 Assignment probabilities close to but larger than the critical values of β for the PCRD-S variant (mutual information based cost function minimized by the incremental splitting method). Assignment probabilities are coded by gray values, where white and black indicate values of zero and one. Indices of data points and clusters are plotted along the horizontal and vertical axes. For parameters see caption of fig. 9.9.

Figure 9.10 shows the assignment probabilities close to the critical values of β for the PCRD-S method. In contrast to pairwise clustering without annealing, PCRD-S at first finds the global structure of the data set and then improves the clustering solution by taking finer details into account. After the first split (fig. 9.10 a) the myoglobin family is separated from the other globin proteins. The other globin proteins separate into the groups HB and HD-HF after the second split (fig. 9.10 b). After the third splitting (fig. 9.10 c) two hemoglobin clusters, one myoglobin cluster and one cluster for the other globins is present. After the fourth splitting (fig. 9.10 d) every group is represented by two clusters. After the fifth split the 'other globin' group is refined (fig. 9.9c).

9.8.2 DNA Microarray Data

In this section I apply SOM method to the analysis of `DNA microarray` data. The `DNA micro-array` dataset was taken from Pomeroy et al. (2002). It contains 60 samples from different brain tumors of the medullablastoma kind, each of which is characterized by the expression levels of 7,129 genes. The samples were obtained from 60 patients, which were treated in a similar way and the samples were labeled according to whether a patient responded well to chemo- or radiation therapy. 71 genes were selected using the -slightly modified- ranking method of Hochreiter and Obermayer (2003) and a dissimilarity matrix between patients was constructed using the normalized Euclidean distance measure

$$d(i,j) \;=\; \frac{\|\tilde{x}_i - \tilde{x}_j\|^2}{2\sigma_X}, \quad \tilde{x}_i = (\frac{x_i^1}{\sigma_1}, \ldots, \frac{x_i^N}{\sigma_D}), \quad \sigma_X = \frac{1}{D}\sum_k \sigma_k^2,$$

where σ_i is standard deviation of i-th gene expression level across samples. Clustering and visualization was performed using a SOM with 9 units arranged in a 3×3 lattice.

(18, 1)	(2 , 0)	(1 , 8)
(0 , 8)	(0 , 4)	(0 , 0)
(0 , 11)	(0 , 0)	(0 , 7)

Figure 9.11: Topographic clustering of DNA microarray data. The figure shows the final map of samples for a SOM network with 3×3 units. The patients assigned to every unit are characterized by the number of positive and negative treatment outcomes (+, -).

The neighborhood function is given by

$$q(l|m) = \frac{\exp(-\|l-m\|^2/0.45)}{\sum_k \exp(-\|k-m\|^2/0.45)}, \qquad l, m \in \{(i,j)\}_{i,j=1}^{3}.$$

An exponential annealing schedule is choosen, given by $\beta_0 = 57.74, \quad \beta_F = 30 \times \beta_0$ $\beta(t+1) = 1.1 \times \beta(t)$. The convergence criterion for the fixed point iteration is given by $\frac{1}{N \times M} \sum_{k=1}^{N} \sum_{m=1}^{M} |p_1(m|x_k)^t - p_1(m|x_k)^{t-1}| < 10^{-7}$.

Fig. 9.11 shows the resulting two-dimensional 'map' of patient space. Every box corresponds to one unit of the SOM lattice, and the groups of patients, which are assigned to a particular unit, are characterized by the number of positive and negative treatment outcomes. With one exception, all patients which respond favorably are clustered and are assigned to the upper left neuron, indicating a strong similarity in the corresponding expression patterns and a common set of causes. The distribution of the other patients across all clusters, however, indicate that the population may still be heterogeneous. If one would label the upper left cluster as 'favorable treatment outcome', only two patients would have been wrongly assigned.

Chapter 10

Clustering and Embedding for Co-occurrence Data

10.1 Analysis of Co-occurrence Data

Co-occurrence data is matrix data, whose column objects x and row objects y are from two different sets $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_M\}$, where the objects from those two sets are observed in pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Each entry of the matrix is the probability $P(x, y)$ or the frequency n_{ij} that a particular object x_i from the column set and a particular object y_j from the row set co-occur in the observation. Examples for co-occurrence data are the verbs and direct objects in sentences, words and documents, galaxies and spectral components, web-pages and user browsing or tissues and gene expression patterns. Clustering and embedding of co-occurrence data is a practical important tool for a wide variety of applications including web search engines, navigating and browsing document collections. The major advantage of co-occurrence data analysis is that it does not require to make any ad-hoc assumptions about the metric of data space and it relies only on the co-occurrence statistics. The algorithms for analyzing co-occurrence data can be roughly categorized into two approaches. The first one is a low rank decomposition approach which includes latent semantic indexing (LSI) (Deerwester et al. (1990); Dumais (1995); Kontostathis et al. (2005)) and probabilistic LSI (Hofmann et al. (1999); Hofmann (1999)). The other approach is distributional clustering (Pereira et al. (1993)). While both approaches were initially developed with the analysis and clustering of 'words and documents' data in mind (Pereira et al. (1993); Deerwester et al. (1990); Hofmann (1999)), they were also generalized to universal co-occurrence data (Tishby et al. (1999); Slonim and Tishby (1999); Hofmann et al. (1999)).

Latent semantic indexing (LSI) was developed in the early 1990s (Deerwester et al. (1990)) as an automatic method for information retrieval which applies singular value decomposition (SVD) for dimension reduction. In LSI it is assumed that there is some latent structure in the pattern of word usage across documents. The LSI structure is obtained by SVD in that the set of documents are projected orthogonally onto a k-dimensional subspace. The documents are represented as continuous values in this so-called LSI space. This orthogonal projection on the reduced k dimensional LSI space guarantees to have the lowest possible least-square distance to the original documents

(Schütze and Silverstein (1997)). Because of the use of SVD in LSI, the L_2 norm is taken, which implicitly assumes additive Gaussian noise. In LSI space the similarity between documents, and between documents and queries, can be more reliably estimated than in the original space, in which the documents are represented as sets of independent words. Because the dimension of LSI space is smaller than the original number of words, the words will be not independent in LSI space, and two documents with frequent co-occurrence in LSI space are similar to each other, even if they have no word in common. LSI performs well in many standard information retrieval settings, including dimension and noise reduction for document data (Bellegarda (1998); Foltz and Dumais (1992); Landauer and Dumais (1997)).

Probabilistic latent Semantic Indexing (PLSI) was proposed by Hofmann (2001, 1999) as a probabilistic variant of LSI based on a so-called aspect model (Saul and Pereira (1997)), which is a statistical latent class model for co-occurrence data. In the aspect model the co-occurrence data is assumed to be associated to an unobserved class variable, C, and the words and documents are assumed conditionally independent, given an associated latent variable, i.e. that the words are generated conditioned on the latent class independently of the documents. Using these assumptions the joint probabilities of documents and words are constructed by a mixture model, while the latent class variable is discarded, $P(x_i, y_j) = \sum_m P(c_m)P(x_i|c_m)P(y_j|c_m)$, where $P(x_i|c_m)$ and $P(y_j|c_m)$ are the parameters of the aspect (class) conditional distributions and $P(c_m)$ the prior probabilities. The algorithm is derived based on the maximization of the log-likelihood (ML), $\log\left[\prod_{i,j} P(x_i, y_j)^{n_{ij}}\right]$, and it is maximized using the Expectation-Maximization (EM) method. In contrast to LSI, PLSI does not make any assumption about the noise model. PLSI maximizes the likelihood of the multinomial sampling using a mixture approximation of the co-occurrence matrix data. As a generalization of maximum likelihood for mixtures, the tempered EM is proposed which introduces a temperature parameter and applies the deterministic annealing method.

Pereira et al. (1993) proposed a distributional clustering algorithm for document data which is represented by a conditional distributional probability $P(y_j|x_i) = n_{ij}/\sum_{\{k:y_k\in\mathcal{Y}\}} n_{ik}$. For clustering the Kullback-Leibler distance is used as the dissimilarity measure of the document distribution data. The generalization of this algorithm suggested by Tishby et al. (1999) is an important and widely used universal clustering algorithm for co-occurrence data and called 'information bottleneck (IB)'. The data is given in form of co-occurrence probability, the joint probability of two objects $\mathcal{X}$ and $\mathcal{Y}$. In Information Bottleneck it is assumed the object $\mathcal{Y}$ contains relevant information about the other object $\mathcal{X}$ to be clustered. The main idea of IB is to cluster the object $\mathcal{X}$, such that the information about the object $\mathcal{Y}$ is preserved as much as possible. The object $\mathcal{Y}$ having the relevant information about the object $\mathcal{X}$ is sometimes called auxiliary or relevant variable (Tishby et al. (1999); Sinkkonen and Kaski (2002); Chechik and Tishby (2002)). The objective function of IB is constructed based on rate distortion theory (Cover and Thomas (1991)) where the negative mutual information $I(\tilde{X};Y)$ is given as constraint instead of the distortion, where $\tilde{X}$ denotes the clusters. The objective function of IB is $I(\tilde{X};X) - \beta\, I(\tilde{X};Y)$, where β is a Lagrangian multiplier, which reflects the tradeoff between compression and preservation of mutual information. By applying the deterministic annealing procedure to the Lagrangian multiplier, Informa-

tion Bottleneck derives a top-down hierarchical clustering and generates soft clustering assignments. The hierarchical bottom up hard version of IB, which was suggested by Slonim and Tishby (1999), can be considered as the limit case of the soft assignment, where the Lagrangian multiplier goes to infinity. The algorithm starts in this limit with trivial clustering in which every object belongs to its own singleton cluster. Two clusters are merged in a way that the local minimum of the loss of mutual information, i.e. the difference of mutual information between cluster and auxiliary object before and after merging, $I(\tilde{X}_{\text{before}};Y) - I(\tilde{X}_{\text{after}};Y)$, is obtained. This is equivalent to minimizing the Jensen-Shannon divergence (see Lin (1991)), such that at each step this algorithm merges the two most similar clusters. For agglomerative IB further distance measures like the L_1 norm and 'ward's method are suggested by Slonim and Tishby (2000). In contrast to PLSI the IB does not make any assumption about the data distribution, e.g. the existence of generative hidden variables and the conditional independence of words and documents given the latent variable. Information Bottleneck optimizes the relevant information in the primary given co-occurrence data to extract the new representation.

Using LSI and PLSI not only documents, but also words are projected to a low dimensional latent semantic space, i.e. the two-fold dimension reduction occurs during the optimization. From the aspect of clustering, the two methods can be considered as a double-sided clustering of co-occurrence data using a single latent variable model. In this sense, IB and agglomerative IB are one-side clustering algorithms of co-occurrence data. A double clustering (DC) algorithm of document data based on the agglomerative IB was introduced by Slonim and Tishby (2000). The proposed method consists of a two-stage clustering process. At first, words are clustered such that the cluster $\tilde{\mathcal{Y}}$ contains the most information about the document. After word clustering, the co-occurrence probabilities between documents and the word-clusters, $P(x,\tilde{y})$, are computed and used as new co-occurrence data in the second stage. In the second stage, documents are clustered using this new representation of documents. The double clustering algorithm was extended by El-Yaniv and Souroujon (2001) to an iterative double clustering (IDC) algorithm. By repetitive clustering of documents and words, it achieves clustering which is robust against feature noise. DC and IDC are heuristic procedures that can yield good clustering solutions. Using Bayesian belief networks Friedman et al. (2001) introduced the general multivariate principled framework for IB. Via multivariate extension of IB they considered the cases where the clusters of $\mathcal{X}$ and $\mathcal{Y}$ are constructed simultaneously (*simultaneous double clustering*) based on two different objective functions. In Slonim et al. (2002) a hard agglomerative algorithm is presented for extension of the multivariate IB problem. Dhillon et al. (2003) derived a co-clustering algorithm based on a cost function being the divergence between two mutual informations, $I(X;Y) - I(\tilde{X};\tilde{Y})$. In this algorithm, the clustering of row and column objects is performed simultaneously for given number of clusters for both objects, such that the resulting loss in mutual information is minimized.

For the purpose of visualization of the topical contents of document data and the representation of topical similarities between aspects of documents Hofmann (2000) extended the PLSI to a topic map model, called 'ProbMap'. In order to capture additional information about the relationships between topics, ProbMap introduces an additional latent variable and an additional layer of randomness, where the two latent

variables are switched with some probability. Seo and Obermayer (2004) extended the information bottleneck for the purpose of visualization, which I will present in this chapter. This chapter is organized as following: The information bottleneck method of Tishby et al. (1999) is extended in section 10.2 to incorporate topological constraints. In section 10.3 the extended algorithm is applied on a document data set and its results are analyzed.

10.2 Derivation of Topographic Information Bottleneck

I now consider clustering and its visualization for the case of co-occurrence data, whose column and row objects are from two different sets $\mathcal{X}$ and $\mathcal{Y}$, and where the objects from those two sets are observed in pairs (x, y). Let the entries K_{ij} in the matrix $\mathbf{K}$ denote the co-occurrence probabilities

$$P_1(y_i|x_j) = \frac{n(y_i|x_j)}{\sum_{\{y_i \in \mathcal{Y}\}} n(y_i|x_j)}, \tag{10.1}$$

where $n(y_i|x_j)$ is the number of occurrences of the object y_i, given that the other object in the observed pair is x_j. The goal of information bottleneck (cf. Tishby et al. (1999)) is to assign objects x to clusters c in a way that preserves information about the objects y as much as possible. The objective function of IB is constructed based on rate distortion theory, where the mutual information $I(Y;C)$ is used as distortion measure E, $I(C;X) - \beta I(Y;C)$. Here, β is a Lagrange multiplier, which reflects the tradeoff between compression and preservation of mutual information.

In order to generalize information bottleneck to topology preserving IB, the problem eq. (8.9) should be solved under the scheme depicted in fig. 8.3 b. The read-out process has to be specified and a distortion measure has to be constructed. We therefore propose the distortion measure E to be

$$E \;=\; -I(D;Y) = -\sum_{i,l} P_2(y_i, l) \log \frac{P_2(y_i|l)}{P(y_i)}, \tag{10.2}$$

where

$$\begin{aligned} P_2(y_i|l) &= \frac{1}{q(l)} \sum_k P_1(y_i, x_k) \sum_m q(l|m) P_1(m|x_k), \\ q(d) &= \sum_c q(d|c) P(c), \\ P(c) &= \sum_j P_1(c|x_j) P(x_j) \;=\; \frac{1}{N} \sum_j P_1(c|x_j) \end{aligned}$$

Similar to our approach for pairwise data, we again consider an optimal, Bayesian read-out process. Eqs. (8.9) and (10.2) state that the optimal representation $\{P_1(c|x)^{opt}\}$ should contain the minimal amount of the information about $\mathcal{X}$ under the constraint that a certain amount of information is kept about the related objects from the set $\mathcal{Y}$. The Lagrange functional is given by

$$\mathcal{L}_2 \;=\; I(C;X) \;-\; \beta I(Y;D) \;+\; \sum_k \lambda_k \sum_m P_1(m|x_k), \tag{10.3}$$

and setting the derivative of the Lagrangian w.r.t. $P_1(l|x_t)$ to zero (see appendix B.4) we obtain

$$P_1(l|x_t) = \frac{\exp(-\beta\mathcal{E}(l,t))P(l)}{\sum_n \exp(-\beta\mathcal{E}(n,t))P(n)} \tag{10.4}$$

$$\mathcal{E}(l,t) = \sum_m q(m|l) D_{KL}\left[P_1(y|x_t)||P_2(y|m)\right], \tag{10.5}$$

where $D_{KL}[\cdot||\cdot]$ denotes the Kullback-Leibler Divergence. The normalization condition $\sum_n P_1(n|x_k) = 1, \ \forall k$, is fulfilled if $\exp(\lambda_t N - \beta D_{KL}(P_1(y|x_t)||P(y))) = \sum_n \exp\left(-\beta\mathcal{E}(n,t)\right) P(n)$ and positivity of the $P_1(l|x)$ is assured via the exponential function in eq. (10.4).

The convolution in eq. (10.5) can again be omitted as in standard Self-Organizing Map methods. This makes the method computationally more efficient but - again - it has the drawback that eqs. (10.4), (10.5) may not converge to the optimum of eq. (8.9) in all circumstances. If $q(l|m) = \delta(l,m)$, then a clustering method without embedding is obtained (cf. (Tishby et al. (1999))). Annealing of the inverse temperature parameter β may be used to improve optimization. For $\beta \to \infty$ clustering becomes 'hard', i.e. the assignment probabilities become zero or one. If the mutual information $I(X;C)$ in the cost function eq. (8.9) is replaced by the negative conditional entropy $-H(C;X)$, we obtain fixed point equations similar to eqs. (10.4,10.5), but which lack the prior probabilities $P(c)$ as a pre-factor in eq. (10.4).

10.3 Application to the Document Data Set

In this section I apply the Self-Organizing Map approach to the clustering, embedding and visualization of co-occurrence data. The data was taken from the TREC3 corpus of articles from the San Jose Mercury News. The column objects are documents, the row objects are index words, and the entries of the data matrix denote word frequencies, i.e. how often a particular index word occurred in a particular document. A total of 1000 documents and 22,064 index words are used from the corpus. Clustering and embedding is performed using a Self-Organizing Map with 49 units arranged in a 7×7 lattice, which is trained using the mutual information based cost function and the distortion measure given by eq. (10.2). The neighborhood function is a normalized two dimensional Gaussian neighborhood function

$$q(l|m) = \frac{\exp(-\frac{(l-m)^2}{\sigma_q^2})}{\sum_k \exp(-\frac{(k-m)^2}{\sigma_q^2})}, \tag{10.6}$$

where $l, m \in \{(i,j)\}_{i,j=1}^{7}$ are vector indices which describe the location of units in the two-dimensional lattice. Optimization was performed using standard deterministic annealing and the fixed point iteration of eqs. (10.4,10.5). Figure 10.1 shows the resulting two-dimensional 'map' of document space. Every box corresponds to one unit of the Self-Organizing Map and the collection of documents which are assigned to a particular unit is characterized by the 10 index words which occur most often. Units which contain no documents are left blank. These units occur for finite range neighborhood functions

	1	2	3	4	5	6	7
1	"market" "stock" "company" "companies" "star" "million" "year" "percent" "apple" "ibm"	"rates" "death" "back" "percent" "mcdonald" "cars" "door" "chrysler" "kroc" "mitsubishi"	"union" "western" "company" "time" "million" "years" "species" "plants" "franchise" "grimaud"	"forest" "route" "climbing" "locomotives" "grand" "whitney" "hornick" "mountaineers" "boulder" "junction"	"river" "west" "ve" "frequent" "miles" "boat" "airlines" "gambling" "harley" "davidson"		"caption"
2		"time" "body" "man" "years" "year" "garden" "dirt" "percent" "bricker" "adaptationists"	"rate" "sales" "price" "increase" "total" "average" "year" "percent" "august" "jobs"	"valley" "project" "county" "san" "santa" "library" "highway" "traffic" "paving" "coyote"	"including" "set" "card" "check" "cards" "racing" "pumpkin" "moffett" "harness" "aw"	"direct" "telephone" "line" "time" "customers" "year" "lines" "check" "mail" "larson"	
3	"committee" "budget" "tax" "bush" "house" "senate" "year" "wilson" "percent" "gates"	"american" "farmers" "chemicals" "credit" "bank" "loan" "county" "charge" "deaver" "moeller"	"state" "free" "trade" "japan" "water" "dumping" "mexico" "san" "years" "year"	"soviet" "union" "president" "food" "bush" "tuesday" "soviets" "nixon" "credits" "schorr"		"war" "military" "world" "david" "years" "battles" "feel" "americans" "battlefield" "tour"	"report" "people" "education" "bush" "goals" "students" "school" "years" "year" "irs"
4	"federal" "state" "california" "court" "bill" "police" "rights" "wilson" "veto" "gay"	"federal" "case" "court" "criminal" "trial" "city" "judge" "district" "attorney" "exxon"	"problems" "bond" "level" "nuclear" "low" "power" "measure" "waste" "needed" "fremont"		"people" "military" "states" "united" "president" "government" "bush" "army" "haiti" "aristide"		"numbers" "results" "box" "match" "win" "published" "drawing" "saturdays" "lotto" "decco"
5	"research" "people" "women" "study" "abortion" "tissue" "drug" "girls" "percent" "researchers"	"federal" "state" "local" "people" "health" "air" "lead" "county" "year" "mice"	"people" "development" "plan" "residents" "council" "housing" "city" "years" "neighbor" "mayor"	"war" "president" "nuclear" "weapons" "history" "george" "bush" "don" "years" "remembered"		"people" "national" "union" "government" "world" "countries" "children" "country" "year" "germany"	"pacific" "service" "workers" "strike" "railroad" "san" "francisco" "southern" "caltrain" "trains"
6		"people" "community" "program" "city" "school" "san" "year" "center" "district" "schools"	"show" "high" "time" "school" "university" "stanford" "pistol" "booten" "student" "koppel"		"border" "milken" "monday" "police" "county" "san" "years" "told" "year" "santana"	"state" "high" "ransom" "school" "san" "league" "logan" "year" "athletes" "lang"	"league" "team" "play" "season" "players" "sharks" "hockey" "nhl" "kingston" "petty"
7	"fund" "jose" "high" "school" "san" "call" "girl" "boy" "fremont" "aug"	"people" "jose" "award" "school" "san" "years" "mr" "arts" "goodman" "cini"	"hospital" "monterey" "center" "artists" "binder" "lobianco" "courtwright" "murals" "stanton" "ariss"	"jose" "police" "man" "san" "years" "year" "box" "woman" "shot" "abc"	"time" "palmer" "series" "sat" "video" "twin" "viewers" "laura" "peaks" "rtl"		"time" "home" "game" "san" "team" "play" "season" "stanford" "run" "los"

Figure 10.1: Topographic Map of the TREC3 corpus of documents.

The figure shows the final 'map' of document space for a Self-Organizing Map network with 7×7 units. The documents assigned to every unit (small rectangular boxes) are characterized by the 10 index words which occur most often across all documents assigned to this particular unit. Assignment probabilities were initialized using uniform random numbers from the interval $[1, 2]$ and then normalized. Then an annealing schedule with $\beta(0) = 1$, $\beta(t+1) = \beta_0 \times 1.03^{(t-1)}$ with $\beta_0 = 1$ and $\beta_F = 40$ (almost 'hard' clustering) was used. For every new value $\beta(t+1)$ a small random number η was added to the assignment probabilities (for details see the legend of fig. 9.7). The convergence criterion was $\frac{1}{N \times M} \sum_{k=1}^{N} \sum_{m=1}^{M} |P_1(m|x_k)^{old} - P_1(m|x_k)^{new}| < 10^{-5}$. σ_q^2 was 0.4.

	1	2	3	4	5	6	7
1	economy computers	economy car industry	economy nature traffic	nature traffic	travel gambling		
2		gardening	balance economy	nature traffic	gambling racing	communication telephone	
3	financial politics	banking farming	international trade	cold war politics		military war	politics education
4	politics law	law court	ecology nuclear energy		military politics		lottery
5	research women's health	public health	public housing	military history		international politics unions	politics strike
6		public education	university education		border politics	high school sports	sports hockey
7	high school education	high school arts	arts	crime news	television		sports baseball

Figure 10.2: Interpretation of the map shown in fig. 10.1. For details see text.

because of the interpolation properties of the Self-Organizing Map. Figure 10.1 shows that documents are grouped by content and that clusters which contain documents with related topics are located closely in the map. The latter is once more illustrated in table 10.2, where I have characterized every unit by one or two titles which summarize the topic area. All documents related to economy, for example, are located in the upper left corner, international politics is represented in the center / right units, and community issues are mapped to the lower left. Documents which are related to two of the topic areas are mapped to units which lie at the topic borders. Moving from unit (1,3) to (1,5), for example, the topics of the assigned documents change from economy and traffic over nature and spare time activities to travel.

Chapter 11

Summary

In part II of my thesis I have derived a clustering algorithm for pairwise data based on rate distortion theory. This algorithm is very similar to the clustering algorithm based on the maximum entropy principle (Buhmann and Hofmann (1994)). Both algorithms differ in the assignment probability by a factor which corresponds to the prior probability of a cluster (see eqs. 9.5,9.6). Furthermore, the algorithm derived by the minimization of the rate does not need any kind of approximation, whereas the algorithm derived by the minimization of the entropy approximates its Gibbs-probability using a mean-field approximation method. If vectorial data is considered, the minimization of the rate leads to the mass constrained clustering approach, and maximization of the entropy results in standard soft clustering. The behavior of the derived clustering algorithm with respect to the Lagrange multiplier is studied using a linear stability analysis, and the structural phase transition of the algorithm is illustrated using a toy example. In order to overcome the local minima, an optimal incremental deterministic algorithm is suggested. Here, the cardinality of the set of clusters is not fixed, but increases with every structural change of the clustering solution. The clustering algorithm starts with two clusters, and at every critical value two clusters which fulfill the proposed criterion are duplicated. This algorithm outperforms clustering with a fixed number of clusters. Furthermore, the proposed algorithm brings an increase in speed, because the incremental method has the desired number of clusters at the final structural phase transition.

For the purpose of visualization, the proposed algorithm derived by minimization of the rate is extended to an embedding algorithm, which is derived from the source coding framework, by breaking of permutation symmetry (SOM). The performance of the derived algorithms is tested on a protein data set. For the same goal, I extended the information bottleneck algorithm, which is a widely used clustering algorithm for co-occurrence data, using the SOM method. The resulting topographic information bottleneck is applied to a real world document data set. Since data sets and maps were still fairly small, these examples serve as a proof of principle. Real world applications, like the navigation through a large database are characterized by a much larger number of objects and, consequently, require a much larger number of units for visualization. For our example, one and two-dimensional lattices of units were chosen for visualization and embedding. When deterministic annealing is used for optimization, however,

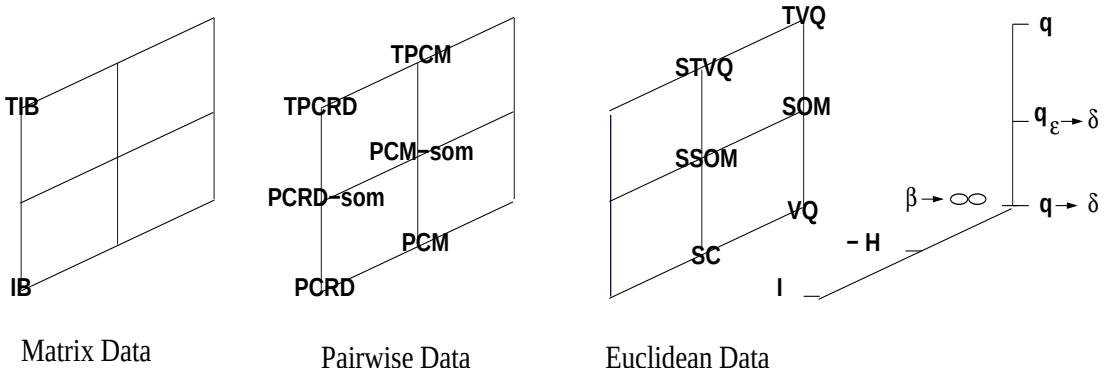


Figure 11.1: Summary of several algorithms for the clustering, the embedding, and the visualization of data.
Abbreviations mean: TIB: topographic information bottleneck (section 4.2), IB: information bottleneck (Tishby et al. (1999) but with equality constraints), PCRD: pairwise clustering based on the rate distortion theorem (section 3.1), TPCRD: topographic PCRD (section 4.1), PCRD-som: SOM approximation for TPCRD, PCM: pairwise clustering using the mean field approximation (Buhmann and Hofmann (1994)), TPCM: topographic PCM (Graepel and Obermayer (1999)), PCM-som: SOM approximation for TPCM (Graepel and Obermayer (1999)), STVQ: soft TVQ (Graepel et al. (1997)), SSOM: soft SOM (Graepel et al. (1997)), SC: soft clustering (Rose (1998)), TVQ: topographic vector quantization (Luttrell (1994)), SOM: the Self Organizing Map (Kohonen (2001)), VQ: vector quantization (Gray (1984)). For details see chapters 9 and 10.

nodes of Self-Organizing Map can be arranged in arbitrary graph structures (including trees), because the probabilities $q(d|c)$, i.e. the elements of the neighborhood function, remain fixed during training and can be chosen accordingly.

Figure 11.1 gives an overview of the different clustering and SOM methods, and shows how they are related to the rate-distortion framework discussed in the second part of the thesis. The listed algorithms are related to the Self-Organizing Map and to the rate-distortion approaches discussed in chapters 9 and 10. In order to visualize the relationship among all methods, algorithms are indicated as nodes of three 3×3 grids, one for matrix (left), pairwise (center), and Euclidean (right) data, respectively. Labels are abbreviations of methods, which have been discussed in the literature (see caption), unlabeled nodes indicate methods which have not yet been thoroughly explored. Lines between nodes refer to the coordinate axes drawn at the right end of the figure. Methods along a horizontal axis differ by the cost function which they optimize: the mutual information based cost function (I), eq. (8.7), the entropy based cost function ($-H$), which is obtained by changing I to $-H$ in eq. (8.7), or a cost function which is solely based on the corresponding distortion measures ($\beta \to \infty$, often referred to as the 'hard clustering' case). Methods along the vertical axis differ by the way the noise process (cf. fig. 8.3) is included: full inclusion of the noise process for topographic clustering (q), application of the SOM approximation ($q_\epsilon \to \delta$), i.e. $q(d|c) = \delta_{dc}$ in $\mathcal{E}$, and the

plain clustering case ($q \to \delta$), i.e. $q(d|c) = \delta_{dc}$ in $\mathcal{E}$ and in $P_2(x_j|d)$. Nodes on equivalent locations within the three grids differ by the type of data, for which the corresponding methods were designed; matrix data, pairwise data, or Euclidean data. Note, that the 'splitting' optimization method (section 9.5) can in principle be applied to all methods listed in the figure which involve an annealing protocol. Work related to the grouping and the embedding of matrix data (leftmost 3×3 grid) is also discussed in Sinkkonen and Kaski (2002) and Hofmann (2000).

Appendix A

A.1 Derivation of eq. (3.19)

$$
\begin{aligned}
& P(l|x)\ (\delta(c_l \neq y) - \mathrm{ls}) \\
&\quad = \delta(c_l \neq y)\frac{Z_{\bar{y}}(x)}{Z_{\bar{y}}(x)}P(l|x)(1-\mathrm{ls}) \\
&\qquad -\delta(c_l = y)\frac{Z_y(x)}{Z_y(x)}P(l|x)\mathrm{ls} \\
&\quad = \delta(c_l \neq y)(1-\mathrm{ls})\frac{Z_{\bar{y}}(x)}{Z(x)}\frac{\exp(-\frac{(x-\theta_l)^2}{2\sigma^2})}{Z_{\bar{y}}(x)} \\
&\qquad -\delta(c_l = y)\mathrm{ls}\frac{Z_y(x)}{Z(x)}\frac{\exp(-\frac{(x-\theta_l)^2}{2\sigma^2})}{Z_y(x)} \\
&\quad = \begin{cases} \mathrm{ls}(1-\mathrm{ls})P_{\bar{y}}(l|x) & \text{if} \quad c_l \neq y \\ -\mathrm{ls}(1-\mathrm{ls})P_y(l|x) & \text{if} \quad c_l = y \end{cases} .
\end{aligned}
$$

$Z(x), Z_y(x), Z_{\bar{y}}(x)$ are the normalization factors of the assignment variables, i.e.

$$
\begin{aligned}
\frac{Z_y(x)}{Z(x)} &= \frac{\sum_{\{l:c_l=y\}}\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_r \exp\left(-\frac{(x-\theta_r)^2}{2\sigma^2}\right)} \\
&= 1-\mathrm{ls}, \\
\frac{Z_{\bar{y}}(x)}{Z(x)} &= \frac{\sum_{\{l:c_l\neq y\}}\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{\sum_r \exp\left(-\frac{(x-\theta_r)^2}{2\sigma^2}\right)} \\
&= \mathrm{ls},
\end{aligned}
$$

and

$$
\begin{aligned}
P_y(l|x) &= \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{Z_y(x)} \\
P_{\bar{y}}(l|x) &= \frac{\exp\left(-\frac{(x-\theta_l)^2}{2\sigma^2}\right)}{Z_{\bar{y}}(x)}.
\end{aligned}
$$

A.2 The gradient of $\log \frac{p(x,y|\mathcal{T})}{p(x,\bar{y}|\mathcal{T})}$ w.r.t. θ_l

$$
\begin{aligned}
& \frac{\partial}{\partial \theta_l} \log \frac{p(x,y|\mathcal{T})}{p(x,\bar{y}|\mathcal{T})} \\
= \; & \frac{p(x,\bar{y}|\mathcal{T})}{p(x,y|\mathcal{T})} \left[\frac{1}{p(x,\bar{y}|\mathcal{T})} \frac{\partial p(x,y|\mathcal{T})}{\partial \theta_l} - \frac{p(x,y|\mathcal{T})}{p(x,\bar{y}|\mathcal{T})^2} \frac{\partial p(x,\bar{y}|\mathcal{T})}{\partial \theta_l} \right] \\
\overset{(a)}{=} \; & \delta(c_l = y) \frac{p(l) \exp f(x,\theta_l)}{p(x,y|\mathcal{T})} \frac{\partial f(x,\theta_l)}{\partial \theta_l} \\
& -\delta(c_l \neq y) \frac{p(l) \exp f(x,\theta_l)}{p(x,\bar{y}|\mathcal{T})} \frac{\partial f(x,\theta_l)}{\partial \theta_l} \\
= \; & \delta(c_l = y) P_y(l|x) \frac{\partial f(x,\theta_l)}{\partial \theta_l} - \delta(c_l \neq y) P_{\bar{y}}(l|x) \frac{\partial f(x,\theta_l)}{\partial \theta_l}
\end{aligned}
$$

where (a)

$$
\begin{aligned}
\frac{\partial p(x,y|\mathcal{T})}{\partial \theta_l} &= \delta(c_l = y) p(l) \exp f(x,\theta_l) \; \frac{\partial f(x,\theta_l)}{\partial \theta_l}, \\
\frac{\partial p(x,\bar{y}|\mathcal{T})}{\partial \theta_l} &= \delta(c_l \neq y) p(l) \exp f(x,\theta_l) \; \frac{\partial f(x,\theta_l)}{\partial \theta_l}.
\end{aligned}
$$

A.3 The gradient of $\log \frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})}$ w.r.t. θ_l

$$
\begin{aligned}
& \frac{\partial}{\partial \theta_l} \log \frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})} \\
= \; & \frac{p(x|\mathcal{T})}{p(x,y|\mathcal{T})} \left[\frac{1}{p(x|\mathcal{T})} \frac{\partial p(x,y|\mathcal{T})}{\partial \theta_l} \right. \\
& \qquad \left. - \frac{p(x,y|\mathcal{T})}{p(x|\mathcal{T})^2} \left(\frac{\partial p(x,y|\mathcal{T})}{\partial \theta_l} + \frac{\partial p(x,\bar{y}|\mathcal{T})}{\partial \theta_l} \right) \right] \\
\overset{(a)}{=} \; & \delta(c_l = y) \frac{p(l) \exp f(x,\theta_l)}{p(x,y|\mathcal{T})} \frac{\partial f(x,\theta_l)}{\partial \theta_l} \\
& - \delta(c_l = y) \frac{p(l) \exp f(x,\theta_l)}{p(x|\mathcal{T})} \frac{\partial f(x,\theta_l)}{\partial \theta_l} \\
& - \delta(c_l \neq y) \frac{p(l) \exp f(x,\theta_l)}{p(x|\mathcal{T})} \frac{\partial f(x,\theta_l)}{\partial \theta_l} \\
= \; & \delta(c_l = y)(P_y(l|x) - P(l|x)) \frac{\partial f(x,\theta_l)}{\partial \theta_l} \\
& - \delta(c_l \neq y) P(l|x) \frac{\partial f(x,\theta_l)}{\partial \theta_l}
\end{aligned}
$$

A.4 The Gradient of $\mathrm{ls}(x_t, \mathcal{T})$ w.r.t. θ_1

$$
\begin{aligned}
\frac{\partial \mathrm{ls}(x_t,\mathcal{T})}{\partial \theta_1} &= \frac{\partial \mathrm{ls}(x_t,\mathcal{T})}{\partial \mu}\frac{\partial \mu(x)}{\partial \theta_1} \\
\frac{\partial \mathrm{ls}(x_t,\mathcal{T})}{\partial \mu} &= -\eta(t)\frac{\exp(-\eta(t)\mu(x_t))}{(1-\exp\left(-\eta(t)\mu(x_t)\right))^2} = \eta(t)\mathrm{ls}(x_t,\mathcal{T})(1-\mathrm{ls}(x_t,\mathcal{T})) \\
\frac{\partial \mu(x)}{\partial \theta_1} &= -\frac{d_1-d_2}{(d_1+d_2)^2}\frac{\partial d_1}{\partial \theta_1} + \frac{1}{d_1+d_2}\frac{\partial d_1}{\partial \theta_1} = \frac{2d_2}{(d_1+d_2)^2}\frac{\partial d_1}{\partial \theta_1} \\
&= \frac{2d_2}{(d_1+d_2)^2}2(x_t-\theta_1) \\
\frac{\partial \mu(x)}{\partial \theta_2} &= -\frac{d_1-d_2}{(d_1+d_2)^2}\frac{\partial d_2}{\partial \theta_2} - \frac{1}{d_1+d_2}\frac{\partial d_2}{\partial \theta_2} = -\frac{2d_1}{(d_1+d_2)^2}2(x_t-\theta_2) \\
\frac{\partial \mathrm{ls}(x_t,\mathcal{T})}{\partial \theta_1} &= \mathrm{ls}(x_t,\mathcal{T})(1-\mathrm{ls}(x_t,\mathcal{T}))\frac{4d_2\eta(t)}{(d_1+d_2)^2}(x_t-\theta_1) \\
\frac{\partial \mathrm{ls}(x_t,\mathcal{T})}{\partial \theta_2} &= -\mathrm{ls}(x_t,\mathcal{T})(1-\mathrm{ls}(x_t,\mathcal{T}))\frac{4d_1\eta(t)}{(d_1+d_2)^2}(x_t-\theta_2)
\end{aligned}
$$

By setting $\alpha^* = \alpha/(4\eta(t))$ one obtains the learning rule, eq. (6.7).

A.5 Learning Rate

For the learning rate

$$\alpha(t) = \alpha(0) \times \frac{\alpha_t}{\alpha_T \times t} \tag{A.1}$$

the following values are used. ζ is the number of prototypes per class and N is the number of training data points.

algorithms	**letter** $\alpha(0)$	α_T	**pendigits** $\alpha(0)$	α_T
LVQ 2.1	0.1	$12000 \times \zeta$	0.1	$2 \times N \times \zeta$
SNP	$0.1 \times \zeta + 0.15$	52000	$0.5 \times \zeta + 0.15$	$N \times 10$
SLVQ-LR	0.1	$12000 \times \zeta$	$0.1 \times (0.5 \times \zeta + 3)$	N
LVQ-LR	0.1	$12000 \times \zeta$	0.1	$N \times \zeta$
RSLVQ	0.1	$12000 \times \zeta$	$0.1 \times (0.5 \times \zeta + 3)$	N
GPD	0.1	$12000 \times \zeta$	$0.1 \times (0.5 \times \zeta + 3)$	N
MMLVQ	0.1	$N \times (\zeta + 1)$	$0.1 \times (0.5 \times \zeta + 3)$	N

For the annealing version of SNPC, SLVQ-LR and RSLVQ $\alpha(0)$ is set to 0.1 for the first epoch and to 0.02 from the second epoch on. α_T is set the same value as without annealing.

A.6 Optimal Hyper Parameters for the Data Set `Letter`

Values of the hyper parameters as selected by cross-validation on the training set. ζ is the number of prototypes per class.

ζ	lvq 2.1: ω	snpc: σ^2	lvq-lr: ω	slvq-lr: ω	slvq-lr: σ^2	rslvq: σ^2
1	0.02	0.5	0.03	0.015	0.3	1.5
2	0.035	1.25	0.03	0.025	1.5	2.0
3	0.03	1.6	0.02	0.035	2.25	1.5
4	0.035	1.4	0.045	0.045	2.25	2.75
5	0.04	1.75	0.035	0.03	1.75	2.5
6	0.055	2.1	0.045	0.045	1.75	2.0
7	0.065	2.4	0.05	0.045	2.25	2.25
8	0.06	3.0	0.05	0.055	2.5	1.75
9	0.065	3.0	0.055	0.045	2.0	2.5
10	0.065	2.25	0.055	0.055	1.75	2.25

ζ	gpd η	gpd: γ	glvq: γ	mmlvq: η	mmlvq: γ
1	5	15	5	3.5	2
2	2	10	13	2.5	2
3	2.25	5	13	3	3.5
4	2.75	5	15	2	3.5
5	2	5	15	2.5	3
6	1.25	4	30	2	3.5
7	1.5	4	30	1.5	4
8	1.5	10	15	1.5	4.5
9	1.5	10	50	1.5	4.5
10	1.5	15	30	1.5	4.5

A.7 Optimal Hyper Parameters for the Data Set `pendigits`

Values of the hyper-parameters as selected by cross-validation on the training set. ζ is the number of prototypes per class.

ζ	lvq 2.1: ω	snpc: σ^2	lvq-lr: ω	slvq-lr: ω	slvq-lr: σ^2	rslvq: σ^2
1	0.03	400	0.015	0.015	100	500
2	0.055	250	0.04	0.035	700	600
3	0.055	300	0.045	0.045	500	600
4	0.065	600	0.045	0.08	800	600
5	0.075	1000	0.06	0.06	700	900
6	0.11	875	0.07	0.06	600	500
7	0.09	575	0.085	0.07	800	600
8	0.11	500	0.075	0.07	500	600
9	0.13	525	0.085	0.075	600	500
10	0.135	475	0.09	0.075	600	600

ζ	gpd η	gpd: γ	glvq: γ	mmlvq: η	mmlvq: γ
1	0.02	150	3000	0.01	150
2	0.025	200	5000	0.0075	150
3	0.04	250	4000	0.001	100
4	0.03	250	3500	0.005	75
5	0.03	200	4500	0.0025	125
6	0.03	200	5500	0.005	125
7	0.04	200	4000	0.0025	125
8	0.04	250	4500	0.01	125
9	0.04	250	3500	0.0075	100
10	0.05	250	5000	0.0075	75

A.8 Scaling Schedule for Dynamic Hyper Parameter Scaling

ζ is the number of prototypes per class and N is the number of training data points.

Learning Rate α

$$\alpha(t+1) = \alpha_{\text{ini}} \times \frac{\epsilon_\alpha \times N}{\epsilon_\alpha \times N + t}$$

algorithms	**letter:** $\alpha(0)$	ϵ_α	**pendigits:** $\alpha(0)$	ϵ_α
LVQ 2.1	0.05	4	0.05	4
SNPC	if $(\zeta < 4)$ $0.05 \times \zeta$ else 0.2	6	if $(\zeta < 4)$ $0.5 \times \sigma^2_{\text{mean}}$ else $0.7 \times \sigma^2_{\text{mean}}$	15
LVQ-LR	0.05	4	0.05	4
SLVQ	0.05	4	0.05	10
RSLVQ	$0.1 \times (\zeta/2 + 3)$	0.5	0.05	10

σ^2 Scaling Schedule

$$\sigma^2(t+1) = \sigma^2_{\text{ini}} \times \frac{\epsilon_\sigma \times N}{\epsilon_\sigma \times N + t}$$

algorithms	**letter:** σ^2_{ini}	ϵ_σ	**pendigits:** σ^2_{ini}	ϵ_σ
SNPC	if $(\zeta = 1)$ $0.5 \times \sigma^2_{\text{mean}}$ else σ^2_{mean}	4	$0.4 + 0.1 \times \zeta$	10
SLVQ-LR	$0.7 \times \sigma^2_{\text{mean}}$	4	$1.5 \times \sigma^2_{\text{mean}}$	10
RSLVQ	if $(\zeta = 1)$ $0.5 \times \sigma^2_{\text{mean}}$ else σ^2_{mean}	6	if $(\zeta = 1)$ $0.5 \times \sigma^2_{\text{mean}}$ else $1.5 \times \sigma^2_{\text{mean}}$	10

$\sigma^2_{\text{mean}} = \frac{1}{N_y} \sum_{k=1}^{N_y}$, σ^2_k is the average value of σ^2_k, which is the mean of the diagonal elements of the covariance matrix of the subset of data points with label k, and N_y is the number of the class labels.

ω Scaling Schedule

$$\omega(t+1) = \omega_{\text{ini}} \times \frac{\epsilon_\omega \times N}{\epsilon_\omega \times N + t}$$

algorithms	**letter:** ω_{ini}	ϵ_ω	**pendigits:** ω_{ini}	ϵ_ω
LVQ 2.1	if $(\zeta = 1)$ 0.04 else $0.01 \times (\zeta/2 + 5)$	8	$0.04 + (0.015 \times \zeta)$	15
LVQ-LR	if $(\zeta < 4)$ 0.04 else 0.08	8	$0.04 + (0.015 \times \zeta)$	15
SLVQ-LR	if $(\zeta < 4)$ 0.04 else 0.08	8	if $(\zeta = 1)$ 0.04 else $0.01 \times (\zeta/2 + 5)$	15

Appendix B

B.1 Γ_I: Calculation of the Derivative of $\mathcal{E}_{ls}^{\beta}$ w.r.t. $\mathcal{E}_{mt}^{\beta}$

For the mutual information based cost function we obtain

$$\begin{aligned}
\frac{\partial \mathcal{E}_{ls}^{\beta}}{\partial \mathcal{E}_{mt}^{\beta}} &= \frac{\beta}{P(l)} \frac{\partial P(l)}{\partial \mathcal{E}_{mt}^{\beta}} \left[\frac{1}{NP(l)} \sum_i P(l|x_i) d(i,s) - \frac{1}{N^2 P(l)^2} \sum_{i,j} P(l|x_i) P(l|x_j) d(i,j) \right] \\
&\quad - \frac{\beta}{NP(l)} \sum_i \frac{\partial P(l|x_i)}{\partial \mathcal{E}_{mt}^{\beta}} \left[d(i,s) - \frac{1}{NP(l)} \sum_j P(l|x_j) d(i,j) \right], \\
&= \frac{\beta}{P(l)} \mathcal{S}_{mt}^{l} \left[\sum_i P(x_i|l) d(i,s) - \sum_{i,j} P(x_i|l) P(x_j|l) d(i,j) \right] \\
&\quad - \frac{\beta}{P(l)N} \sum_i \frac{\partial P(l|x_i)}{\partial \mathcal{E}_{mt}^{\beta}} \left[d(i,s) - \sum_j P(x_j|l) d(i,j) \right] \\
&= \frac{\beta}{P(l)} \mathcal{S}_{mt}^{l} \left[\sum_i P(x_i|l) d(i,s) - \sum_{i,j} P(x_i|l) P(x_j|l) d(i,j) \right] \\
&\quad - \beta P(x_t|l)(\delta(l,m) - P(m|x_t)) \left[d(t,s) - \sum_j P(x_j|l) d(t,j) \right] \\
&\quad - \frac{\beta}{P(l)} \mathcal{S}_{mt}^{l} \sum_i P(x_i|l) \left[d(i,s) - \sum_j P(x_j|l) d(i,j) \right] \\
&\quad + \beta \sum_i \sum_k \frac{P(k|x_i)}{P(k)} \mathcal{S}_{mt}^{k} P(x_i|l) \left[d(i,s) - \sum_j P(x_j|l) d(i,j) \right] \\
&= \text{eq. (9.8)},
\end{aligned}$$

where we have used

$$\frac{\partial P(l|s)}{\partial \mathcal{E}_{mt}^{\beta}} = P(l|s) \left[\delta(s,t)\delta(l,m) + \frac{1}{P(l)} \mathcal{S}_{mt}^{l} - \delta(t,s) P(m|x_t) - \sum_k \frac{P(k|s)}{P(k)} \mathcal{S}_{mt}^{k} \right],$$

$$\begin{aligned}\mathcal{S}^k_{mt} &= \frac{\partial P(k)}{\partial \mathcal{E}^\beta_{mt}} = \frac{1}{N}\sum_i \frac{\partial P(k|x_i)}{\partial \mathcal{E}^\beta_{mt}},\\ Q(k|q) &= \frac{1}{N}\sum_i P(k|x_i)\frac{P(q|x_i)}{P(q)},\end{aligned}$$

and

$$\sum_i \frac{\partial P(k|x_i)}{\partial \mathcal{E}^\beta_{mt}} = P(k|x_t)\left(\delta(k,m) - P(m|x_t)\right) + N\frac{\partial P(k)}{\partial \mathcal{E}^\beta_{mt}} - N\sum_q Q(k|q)\mathcal{S}^q_{mt}.$$

Eq. (9.8) can be solved numerically. Let

$$\mathcal{W}^k_{m,t} = \frac{1}{N}P(k|x_t)\left(\delta(k,m) - P(m|x_t)\right).$$

Using a matrix formulation and the Einstein notation we obtain

$$\sum_q Q(k|q)\mathcal{S}^q_{m,t} = \mathcal{Q}_{k,q}\mathcal{S}^q_{m,t} = \mathcal{W}^k_{m,t} \quad \Rightarrow \quad \mathcal{S}^q_{m,t} = \mathcal{Q}^{-1}_{q,k}\mathcal{W}^k_{m,t}.$$

Taking the inverse of the matrix $\mathcal{Q}^{-1}_{q,k}$, however, might cause a singularity problem. In order to prevent this, we sometimes approximate Γ_I by setting $Q(k|q) = \delta(k,q)$ which leads to the equivalence between $\Gamma_I = \Gamma_H$.

B.2 Γ_H: Calculation of the Derivative of $\mathcal{E}^\beta_{ls}$ w.r.t. $\mathcal{E}^\beta_{mt}$

For the entropy based cost function we obtain

$$\begin{aligned}\frac{\partial \mathcal{E}^\beta_{ls}}{\partial \mathcal{E}^\beta_{mt}} &= \frac{\beta}{NP(l)^2}\frac{\partial P(l)}{\partial \mathcal{E}^\beta_{mt}}\sum_i P(l|x_i)d(i,s) - \frac{\beta}{NP(l)}\sum_i \frac{\partial P(l|x_i)}{\partial \mathcal{E}^\beta_{mt}}d(i,s)\\ &\quad - \frac{\beta}{N^2P(l)^3}\frac{\partial P(l)}{\partial \mathcal{E}^\beta_{mt}}\sum_{i,j} P(l|x_i)P(l|x_j)d(i,j)\\ &\quad + \frac{\beta}{N^2P(l)^2}\sum_{i,j}\frac{\partial P(l|x_i)}{\partial \mathcal{E}^\beta_{mt}}P(l|x_j)d(i,j),\\ &= \frac{\beta}{P(l)}\frac{\partial P(l)}{\partial \mathcal{E}^\beta_{mt}}\left[\frac{1}{NP(l)}\sum_i P(x_i|l)d(i,s) - \frac{1}{N^2P(l)^2}\sum_{i,j}P(x_i|l)P(x_j|l)d(i,j)\right]\\ &\quad - \frac{\beta}{NP(l)}\sum_i \frac{\partial P(l|x_i)}{\partial \mathcal{E}^\beta_{mt}}\left[d(i,s) - \frac{1}{NP(l)}\sum_j P(l|x_j)d(i,j)\right],\\ &= \frac{\beta}{P(l)N}P(l|x_t)(\delta(l,m) - P(m|x_t))\\ &\quad \times \left[\sum_i P(x_i|l)d(i,s) - \sum_{i,j}P(x_i|l)P(x_j|l)d(i,j)\right]\\ &\quad - \frac{\beta}{P(l)N}\sum_i \delta(i,t)P(l|x_i)\left[\delta(l,m) - P(m|x_t)\right]\left[d(i,s) - \sum_j P(x_j|l)d(i,j)\right]\\ &= \text{eq. (9.9)},\end{aligned}$$

where we have used

$$\begin{aligned}
\frac{\partial P(l|s)}{\partial \mathcal{E}^{\beta}_{mt}} &= \frac{1}{Z_l}\exp(\mathcal{E}^{\beta}_{ls})\delta(t,s)\delta(l,m) - \frac{\exp(\mathcal{E}^{\beta}_{ls})}{Z_l^2}\delta(t,s)\exp(\mathcal{E}^{\beta}_{mt}) \\
&= \delta(l,t)P(l|s)\left[\delta(l,m) - P(m|x_t)\right], \\
\sum_i \frac{\partial P(k|x_i)}{\partial \mathcal{E}^{\beta}_{mt}} &= P(l|x_t)\left[\delta(l,m) - P(m|x_t)\right]
\end{aligned}$$

and

$$\frac{\partial P(l)}{\partial \mathcal{E}^{\beta}_{mt}} = \frac{1}{N}P(l|x_t)\left[\delta(l,m) - P(m|x_t)\right].$$

B.3 Calculation of the Derivative of the Lagrangian $\mathcal{L}_1$ w.r.t. $P_1(l|x_t)$

$$\begin{aligned}
\frac{\partial \mathcal{L}_1}{\partial P_1(l|x_t)} &= \frac{\partial I(C;X)}{\partial P_1(l|x_t)} + \beta\frac{\partial < d(x,x') >}{\partial P_1(l|x_i)} - \lambda_t \\
\frac{\partial I(C;X)}{\partial P_1(l|x_t)} &= \frac{1}{N}\log\frac{P_1(l|x_t)}{P(l)} \\
\frac{\partial < d(x,x') >}{\partial P_1(l|x_i)} &= \frac{1}{N^2}\sum_j d(t,j)\sum_{n,d}\frac{1}{q(d)}q(d|n)q(d|l)P_1(n|x_j) \\
&\quad + \frac{1}{N^2}\sum_i d(i,t)\sum_{m,d}\frac{1}{q(d)}q(d|l)q(d|m)P_1(m|x_i) \\
&\quad - \frac{1}{N^3}\sum_{i,j} d(i,j)\sum_d \frac{1}{q(d)^2}q(d|l)\sum_{m,n} q(d|n)q(d|m)P_1(m|x_i)P_1(n|x_j) \\
&= \frac{2}{N}\sum_{i,m,d}\left[\frac{q(d|m)P_1(m|x_i)}{q(d)N}\right]q(d|l)d(i,t) \\
&\quad - \frac{1}{N}\sum_{i,j,m,n,d} d(i,j)q(d|l)\left[\frac{q(d|n)P_1(n|x_j)}{q(d)N}\right]\left[\frac{q(d|m)P_1(m|x_i)}{q(d)N}\right] \\
&\overset{9.14}{=} \frac{2}{N}\sum_{i,d} q(d|l)P_2(x_i|d)d(i,t) - \frac{1}{N}\sum_{i,j,d} q(d|l)P_2(x_i|d)P_2(x_j|d)d(i,j) \\
&= \frac{2}{N}\sum_i P_2(x_i|l)d(i,t) - \frac{1}{N}\sum_{i,j} P_2(x_i,x_j|l)d(i,j) \\
\frac{\partial \mathcal{L}}{\partial P_1(l|x_t)} &= \frac{1}{N}\left(\log\frac{P_1(l|x_t)}{p(l)} + 2\beta\sum_i P_2(x_i|l)d(i,t)\right. \\
&\qquad \left. - \beta\sum_{i,j} P_2(x_i,x_j|l)d(i,j) + \lambda_t N\right)
\end{aligned}$$

B.4 Calculation of the Derivative of the Lagrangian $\mathcal{L}_2$ w.r.t. $P_1(l|x_t)$

$$
\begin{aligned}
\frac{\partial \mathcal{L}_2}{\partial P_1(l|x_t)} &= \frac{\partial I(C;X)}{\partial P_1(l|x_t)} - \beta \frac{\partial I(D;Y)}{\partial P_1(l|x_t)} + \lambda_t \\
\frac{\partial I(D;Y)}{\partial P_1(l|x_t)} &= \frac{1}{N} \sum_d q(d|l) \sum_i P_1(y_i|x_t) \log \frac{P_2(y_i|d)}{P(y_i)} \\
&= \frac{1}{N} \sum_d q(d|l) \sum_i P_1(y_i|x_t) \log \left(\frac{P_2(y_i|d)}{P(y_i)} \frac{P_1(y_i|x_t)}{P_1(y_i|x_t)} \right) \\
&= \frac{1}{N} \sum_d q(d|l) \sum_i P_1(y_i|x_t) \left(\log \frac{P_1(y_i|x_t)}{P(y_i)} - \log \frac{P_1(y_i|x_t)}{P_2(y_i|d)} \right) \\
&= \frac{1}{N} D_{KL}(P_1(y|x_t)||P(y)) - \frac{1}{N} \sum_d q(d|l) D_{KL}(P_1(y|x_t)||P_2(y|d)) \\
\frac{\partial \mathcal{L}_2}{\partial P_1(l|x_t)} &= \frac{1}{N} \Bigg(\log \frac{P_1(l|x_t)}{P(l)} - \beta D_{KL}(P_1(y|x_t)||P(y)) \\
&\qquad + \beta \sum_d q(d|l) D_{KL}(P_1(y|x_t)||P_2(y|d)) + N\lambda_t \Bigg)
\end{aligned}
$$

Bibliography

Alpert, C., Kahng, A., and Yao, S. (1999). Spectral partition: The more eigenvectors, the better. *Discrete Applied Math*, 90:3–26.

Baker, C. (1977). *The numerical treatment of integral equation*. Clarendon Press, Oxford.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 14.

Bellegarda, J. (1998). Exploiting both local and global constraints for multi-span language modeling. In *Proceedings of ICASSP'98*, pages 677–680.

Bellman, R. E. (1961). *Adaptive Control Processes*. Princeton University Press.

Bengio, Y. (2000a). Continuous optimization of hyper-parameters. In *International Joint Conference on Neural Networks*, volume 1, pages 1305–1310.

Bengio, Y. (2000b). Gradient-based optimization of hyperparameters. *Neural Computation*, 12(8):1889–1900.

Bengio, Y., Delalleau, O., Roux, N. L., Paiement, J., Vincent, P., and M.Ouimet (2004a). Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 16(10):2197–2219.

Bengio, Y. and Monperrus, M. (2005). Non-local manifold tangent learning. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 129–136. MIT Press, Cambridge, MA.

Bengio, Y., Paiement, J. F., and Vincent, P. (2004b). Out-of-sample extensions for lle, isomap, mds, eigenmaps,and spectral clustering. In *Advances in Neural Information Processing Systems 16*.

Berger, T. (1971). *Rate Distortion Theory*. Prentice Hall.

Bertsekas, D. P. (1982). *Constrained optimization and Lagrange multiplier methods*. Academic Press, New York.

Bezdek, J. (1985). Generalized k-nearest neighbor rules. *Fuzzy Sets and Systems*, 18:237–256.

Bishop, C., Svens'en, M., and Williams, C. (1998). GTM: The generative topographic mapping. *Neural Computation*, 10(1):215–234.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. (1987). Occam 's razor. *Information Processing Letters*, 24:377–380.

Borg, I. and Groenen, P. (1997). *Modern multidimensional scaling: theory and applications*. Springer-Verlag, New York.

Bottou, L. (1998). Online learning and stochastic approximations. In Saad, D., editor, *Online Learning and Neural Networks*, pages 9–42. Cambridge University Press, Cambridge, UK.

Buhmann, J. (1998). *Stochastic Algorithms for Exploratory Data Analysis: Data Clustering and Data Visualisation*, pages 405–420. MIT Press.

Buhmann, J. and Hofmann, T. (1994). A maximum entropy approach to pairwise data clustering. In *International Conference on Pattern Recognition*, pages 207–212. IEEE Computer Society Press.

Centre, N. N. R. (2003). *Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ)*. Helsinki Univ. of Tech. http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html.

Cerny, V. (1985). A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51.

Chan, P., Schlag, M., and Zien, J. (1994). Spectral k-way ratio cut partitioning. *IEEE Transactions on CAD-Integrated Circuits and Systems*, 13:1088–1096.

Chang, C. (1974). Finding prototypes for nearest neighbor classifier. *IEEE Transactions on Computers*, 23(11):1179–1184.

Chechik, G. and Tishby, N. (2002). Extracting relevant structures with side information. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 857–864. MIT press.

Collobert, R. and Bengio, S. (2001). Svmtorch: Support vector machines for large-scale regression p roblems. *Journal of Machine Learning Research*, 1:143–160.

Cottrell, M., Fort, J. C., and Pagés, G. (1998). Theoretical aspects of the som algorithm. *Neurocomputing*, 21(1):119–138.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Thoery*. Wiley & Sons, Inc., New York.

Cox, T. and Cox, M. (1994). *Multidimensional Scaling*. Chapman & Hall, London.

Crammer, K., Gilad-Bachrach, R., Navot, A., and Tishby, N. (2002). Margin analysis of the lvq algorithm. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT press.

Cristianini, N. and Shawe-Taylor, J. (2000). *AN INTRODUCTION TO SUPPORT VECTOR MACHINES (and other kernel-based learning methods)*. Cambridge University Press.

Dasarathy, B. (1990). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society, Los Alamitos, California.

Decaestecker, C. (1993). Nnp: a neural net classifier using prototypes. In *IEEE Internation Conference on Neural Networks*, pages 822–824, San Francisco.

Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood for incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, pages 1–38.

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003). Information-theoretic co-clustering. In Getoor, L., Senator, T. E., Domingos, P., and Faloutsos, C., editors, *International Conference on Knowledge Discovery and Data Mining*, pages 24 – 27, Washington, DC, USA. ACM.

Domingos, P. (1999). The role of occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Sohn Wiley & Sons, New York, second edition.

Dumais, S. (1995). Using lsi for information filtering: Trec-3 experiments. Technical report, National Institute of Standards and Technology.

Dunn, J. (1974). Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104.

El-Yaniv, R. and Souroujon, O. (2001). Iterative double clustering for semi and unsupervised. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 15*, pages 1025–1032. MIT Press.

Erwin, E., Obermayer, K., and Schulten, K. (1992). Self-organizing maps: Ordering, convergence properties and energy functions. *Biol. Cyb.*, 67(1):47–55.

Everitt, B. and Rabe-Hesketh, S. (1997). *The Analysis of Proximity Data*. Arnold, London.

Foltz, P. and Dumais, S. (1992). An analysis of information filtering methods. *Comunications of the ACM*, 35(12):51–60.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997.). Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168.

Friedman, J. (1994). Flexible metric nearest neighbor classification. Technical report, Dept. of Statistics, Standford University.

Friedman, N., Mosenzon, O., Slonim, N., and Tishby, N. (2001). Multivariate information bottleneck. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 152 – 161.

Gardner, W. (1981). Design of nearest prototype signal classifiers (corresp.). *IEEE Transactions on Information Theory*, 27(3):368 – 372.

Gondek, D. and Hofmann, T. (2003). Conditional information bottleneck clustering. In *3rd IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets*.

Graepel, T., Burger, M., and Obermayer, K. (1997). Phase transduction in stochastic self-organizing maps. *Physical Review E.*, 56:3876–3890.

Graepel, T. and Obermayer, K. (1999). A self-organizing map for proximity data. *Neural Computation*, 11:139–155.

Gray, R. M. (1984). Vector quantization. *IEEE Acoustics, Speech and Signal Processing*, 1(2):4–29.

Ham, J., Lee, D., Mika, S., and Schölkopf, B. (2003). A kernel view of the dimensionality reduction of manifold. Technical report, Max Planck Institute for Biological Cybernetics.

Hammer, B. and Villmann, T. (2002). Generalized relevance learning vector qunatization. *Neural Network*, 15:1059–1068.

Hasenjaeger, M., Ritter, H., and Obermayer, K. (1999). Active learning in self-organizing maps. In Oja, E. and Kaski, S., editors, *Kohonen Maps*, pages 57–70. Elsevier.

Haykin, S. (1994). *Neural Networks: A comprehensive Foundation*. Prentice Hall.

Hinton, G. and Roweis, S. (2003). Stochastic neighbor embedding. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 857–864.

Hochreiter, S. and Obermayer, K. (2003). Feature selection and classification on matrix data: From large margins to small covering numbers. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the 22nd International Conference on Research and Development*.

Hofmann, T. (2000). Probmap - a probabilistic approach for mapping large document collections. *Journal for Intelligent Data Analysis*, 4:149–164.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning Journal*, 42(1):177–196.

Hofmann, T. and Buhmann, J. (1995a). Hierarchical pairwise data clustering by mean-field annealing. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 197–202.

Hofmann, T. and Buhmann, J. (1995b). Multidimensional scaling and data clustering. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 104–111. MIT Press.

Hofmann, T. and Buhmann, J. (1997). Pairwise data clustering by deterministic annealing. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1), pages 1–14.

Hofmann, T. and Buhmann, J. (1998). Active data clustering. In *Advances in Neural Information Processing Systems 10*, volume 10, pages 528–534. MIT Press.

Hofmann, T., Puzicha, J., and Jordan, M. (1999). Learning from dyadic data. In Kearns, M., Solla, S., and Cohn, D., editors, *Advances in Neural Information Processing Systems 11*, pages 466–472. The MIT Press.

Hopfield, J. (1987). Learning algorithms and probability distributions in feed-forward and feed-back networks. In *Proceedings of the National Academy of Sciences of the U.S.A.*, volume 84, pages 8429–8433.

I.S.Dhillon, Y.Guan, and B.Kulis (2004). Kernel k-means, spectral clustering and normalized cuts. In *The Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD).*

Jain, A. and Dubes, R. (1988). *Algorithms for Clustering Data*. Prentice Hall.

Jaynes, E. T. (1957a). Information theory and statistical mechanics. *Physical Review*, 106:629–630.

Jaynes, E. T. (1957b). Information theory and statistical mechanics II. *Physical Review*, 108:171–190.

Jolliffe, I. (1986). *Principal Component Analysis*. Springer-Verlag, Berlin.

Juang, B.-H. and Katagiri, S. (1992). Discriminative learning for minimum error classification (pattern recognition). *IEEE Transactions on Signal Processing*, 40:3043 –3054.

Kambhatla, N. and Leen, T. K. (1995). Classifying with gaussian mixtures and clusters. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 681–688.

Kangas, J. (1994). *On the Analysis of Pattern Sequences by Self-Organizing Maps.* PhD thesis, Helsinki University of Technology, Espoo, Finland.

Karayiannis, N. B. (1999). An axiomatic approach to soft learning vector quantization and clustering. *IEEE Transactions on Neural Networks*, 10:1153 –1165.

Katagiri, S., Lee, C.-H., and Juang, B.-H. (1991). New discriminative training algorithms based on the generalized probabilistic descent method. In *Neural Networks for Signal Processing*, pages 299 –308. Proceedings of IEEE Workshop.

Keller, K., Gray, R., and Givens, J. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems Man Cybernet*, 15(4):580–585.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kohonen, T. (1982a). Analysis of a simple self-organizing process. *Biological. Cybernetics.*, 44(2):135–140.

Kohonen, T. (1982b). Self-organizing formation of topologically correct feature maps. *Biological. Cybernetics*, 43(1):59–69.

Kohonen, T. (1986). Learning vector quantization. Technical report, Helsinki Univ. of Tech., Otaniemi.

Kohonen, T. (1990). Improved versions of learning vector quantization. In *International Joint Conference on Neural Networks*, volume 1, pages 545 –550.

Kohonen, T. (2001). *Self Organization Maps.* Springer-Verlag, New York.

Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J., and Torkkola, K. (1995). *Lvq Pak: The Learning Vector Quantization Program Package.*

Komori, T. and Katagiri, S. (1992). Application of a generalized probabilistic descent method to dynamic time warping-based speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 497 –500.

Kontostathis, A., Pottenger, W., and Davison, B. (2005). Identification of critical values in latent semantic indexing. In Lin, T., Ohsuga, S., Liau, C., and Tsumoto, S., editors, *Foundations of Data Mining and Knowledge Discovery.* Springer-Verlag.

Kruskal, J. (1964). Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27.

Kuncheva, L. (1997). Fintness functions in editing k-nn reference set by genetic algorithms. *Pattern Recognition*, 30(6):1041–1049.

Kuncheva, L. and Jain, L. (1999). Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters*, 20:1149–1156.

Landauer, T. and Dumais, S. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240.

Lin, J. (1991). Divergence measures based on the shannon entrop. *IEEE Transactions on Information Theory*, 37(1):145–151.

Linhart, H. and Zucchini, W. (1986). *Model Selection*. Wiley Series in Probability and Mathematical Statistics. Wiley.

Lippmann, R. (1989). Pattern classification using neural networks. *IEEE Comunication Magazine*, pages 47–64.

Luttrell, S. P. (1994). A bayesian analysis of self-organizing maps. *Neural Computation*, 6(5):767–794.

MacKay, D. (1992). Information-based objective functions for active data selection. *Neural Computation*, 4:590–604.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *the Proccedings of the Fifth Berkeley Symposium on Mathematical Statistic and Probability*, pages 281–297, Berkely, CA. University of Califonia Press.

McDermott, E. and Katagiri, S. (1994). Prototype-based minimum classification error/generalized probabilistic descent training for various speech units. *Computer Speech and Language*, 8(4):351–368.

McLachlan, G. and Basford, K. (1988). *Mixture Models*. Marcel Dekker, New York, Base.

Mevissen, H. and Vingron, M. (1996). Quantifying the local reliability of a sequence alignment. *Protein Engineering*, 9(2):127–132.

Miller, D., Rao, A. V., Rose, K., and Gersho, A. (1996). A global optimization technique for statistical classifier design. *IEEE Transactions on Signal Processing*, 44(12):3108–22.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MA:MIT press, Cambridge.

Mollineda, R., Ferri, F., and Vidal, E. (2000). A cluster-based merging strategy for nearest prototype classifiers. In *Proceedings of 4th World Multiconference on Systemics, Cybernetics and Informatics*, volume VII, pages 640–645, Orlando, Florida, USA.

Ng, A., Jordan, M., and Weiss, Y. (2004). On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. B. and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems*, volume 14.

Parker, A. E., Gedeon, T., and Dimitrov, A. (2003). Annealing and the rate distortion problem. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 993–1000. MIT Press.

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Meeting of the Association for Computational Linguistics*, pages 183–190.

Pomeroy, S., Tamayo, P., Gaasenbeek, M., Sturla, L., Angelo, M., McLaughlin, M. E., Kim, J., Goumnerova, L., Black, P., Lau, C., Allen, J., Zagzag, D., Olson, J., Curran, T., Wetmore, C., Biegel, J., Poggio, T., Mukherjee, S., Rifkin, R., Califano, A., Stolovitzky, G., Louis, D., Mesirov, J., Lander, E., and Golub, T. (2002). Prediction of central nervous system embryonal tumors outcome based on gene expression. *Nature*, 415(6870):436–442.

Puzicha, J., Hofmann, T., and Buhmann, J. (2000). A theory of proximity based clustering: Structure detection by optimization. *Pattern Recognition*, 33(4):617–634.

Raiffa, H. and Schlaifer, R. (2000). *Applied Statistical Decision Theory*. WILEY & SONS.

Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407.

Rose, K. (1998). Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. In *Proceedings of IEEE*, volume 86, pages 2210–2239.

Rose, K., Gurewitz, E., and Fox, G. (1990). Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948.

Rose, K., Gurewitz, E., and Fox, G. (1992). Vector quantization by deterministic annealing. *IEEE Transactions on Information Theory*, 38(4):1249–1257.

Rose, K., Gurewitz, E., and Fox, G. (1993). Constrained clustering as an optimization method. *IEEE Transactions on Pattern Analysis and Machine Inteliigence*, 15(8):785–794.

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

Roth, V., Laub, J., Kawanabe, M., and Buhmann, J. (2003). Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1540–1550.

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Sammon, W. (1969). A nonlinear mapping algorithm for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409.

Sato, A. and Yamada, K. (1996). Generalized learning vector quantization. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems 8*, pages 423–429. MIT Press.

Saul, L. and Pereira, F. (1997). Aggregate and mixed-order Markov models for statistical language processing. In Cardie, C. and Weischedel, R., editors, *Proceedings of the Second Conference on Empirical Methods in Na tural Language Processing*, pages 81–89, Somerset, New Jersey. Association for Computational Linguistics.

Scheffer, T. (1999). *Error Estimation and Model Selection*. PhD thesis, Universität Magdeburg.

Schiffman, S., Reynolds, M., and Young, F. (1981). *Introduction to Multidimensional Scaling: Theory, Methods and Application*. Academic Press, New York.

Schoelkopf, B. and Smola, A. (2001). *Learning With Kernels*. MIT Press.

Schölkopf, B., Smola, A., and Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

Schütze, H. and Silverstein, H. (1997). Projections for efficient document clustering. In *Proceedings of Special Interest Group on Information Retrieval (SIGIR)*, pages 74–81, Philadelphia.

Seo, S., Bode, M., and Obermayer, K. (2003). Soft nearest prototype classification. *IEEE Transactions on Neural Network*, 14:390–398.

Seo, S. and Obermayer, K. (2003.). Soft learning vector quantization. *Neural Computation*, 15:1589–1604.

Seo, S. and Obermayer, K. (2004). Self-organizing maps and clustering methods for matrix data. *Neural Networks*, 17:1211–1229.

Seo, S., Wallat, M., Graepel, T., and Obermayer, K. (2000). Gaussian process regression: Active data selection and test point rejection. In *International Joint Conference on Neural Networks*, volume III, pages 241–246.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Sinkkonen, J. and Kaski, S. (2002). Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239.

Slonim, N., Friedman, N., and Tishby, N. (2002). Agglomerative multivariate information bottleneck. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 929–936. MIT Press.

Slonim, N. and Tishby, N. (1999). Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems 12*. MIT press.

Slonim, N. and Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. In *Research and Development in Information Retrieval*, pages 208–215.

Spence, I. (1982). Incomplete designs for multidimensional scaling. In Golledge, R. and Rayner, J., editors, *Proximity and preference: Problems in the multidimensional analysis of large data sets*, chapter 3, pages 29–46. University of Minnesota Press, Minneapolis.

Takane, Y., Young, F., and de Leeuw, J. (1977). Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features. *Psychometrika*, 42(1):7–67.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323.

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 99, pages 6567–6572.

Tishby, N., Pereira, F. C., and Bialek, W. (1999). The information bottleneck method. In *the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

Urahama, K. and Nagao, T. (1993). Learning algorithm for nearest-prototype classifiers. In *International Joint Conference on Neural Networks*, volume 1, pages 585 – 588.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* Springer, Berlin.

Vapnik, V. N. (1998). *Statistical Learning Theory.* Springer, New YorK.

Vuori, V., Laaksonen, J., Oja, E., and Kangas, J. (2000). Controlling on-line adaptation of a prototype-based classifier for handwritten characters. In *15th IEEE International Conference on Pattern Recognition*, volume 2, pages 331–334.

Wang, F., Wang, J., and Zhang, C. (2005). Spectral feature analysis. In *To appear in the International Joint Conference on Neural Networks (IJCNN'05)*, Montreal, Canada.

Weigelt, B., Glas, A., Wessels, L., Witteveen, A., Peterse, J., and Veer, L. (2003). Gene expression profiles of primary breast tumors maintained in distant metastases. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 100, pages 15901–15905.

Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982.

Williams, C. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In Langley, P., editor, *International Conference on Machine Learning 17*, pages 1159–1166. Morgan Kaufmann.

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T., Dietterich, T., and Tresp, V., editors, *Neural Information Processing Systems 13*, pages 682–688. MIT Press.

Yau, H. and Manry, M. (1991). Iterative improvement of a nearest neighbor classifier. *Neural Networks*, 4(4):517–524.

Yu, X. and Shi, J. (2003). Multiclass spectral clustering. In *International Conference on Computer Vision*.

Zhang, R. and Rudnicky, A. (2002). A large scale clustering scheme for kernel k-means. In *Proceeding of the International Conference on Pattern Recognition*, pages 289–292.

Zöller, T. and Buhmann, J. M. (2000). Active learning for hierarchical pairwise data clustering. In *Proceeding of the International Conference on Pattern Recognition*, pages 186–189.