HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Engineering Physics and Mathematics

Arto Klami

Regularized Discriminative Clustering

Master's thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Supervisor Docent Samuel Kaski
Instructor M.Sc. Janne Sinkkonen

Helsinki, 27th February 2003

Tekijä: Arto Klami

Työn nimi: Regularisoitu diskriminatiivinen klusterointi

English title: Regularized Discriminative Clustering

Päivämäärä: 27. helmikuuta 2003 **Sivumäärä:** 68

Osasto: Teknillisen fysiikan ja matematiikan osasto

Professuuri: T-61, Informaatiotekniikka

Työn valvoja: dosentti Samuel Kaski

Työn ohjaaja: PsM Janne Sinkkonen

Työssä tutkitaan ja kehitetään eksploratiivisen data-analyysin työkalua, jota kutsutaan diskriminatiiviseksi klusteroinniksi. Menetelmää voidaan käyttää vektorimuotoisten aineistojen analysointiin, kun jokaista näytettä kohden on saatavilla lisätietoa antava luokkamuuttuja. Menetelmä pyrkii löytämään aineistosta klustereita jotka kertovat mahdollisimman paljon tästä lisäaineistosta.

Diskriminatiivinen klusterointi perustuu oppivan metriikan periaatteeseen. Periaate antaa perustellun tavan oppia metriikka data-avaruuteen tarjolla olevan lisätiedon avulla. Tätä voidaan pitää puolittain ohjattuna oppimisena, koska metriikka opitaan ohjatusti, mutta sitä voidaan käyttää ohjaamattoman oppimisen menetelmien yhteydessä.

Diskriminatiivinen klusterointi jakaa data-avaruuden ja sen mukana opetusaineiston erillisiin alueisiin. Se etsii klustereita, jotka ovat paikallisia alkuperäisessä metriikassa, ja joiden sisällä lisäaineiston jakauma on mahdollisimman homogeeninen. Niinpä diskriminatiivisen klusteroinnin tuloksia on helppo tulkita alkuperäisen data-avaruuden kannalta, mutta klusterit kertovat toisaalta paljon myös lisäaineistosta. Menetelmää kutsutaan diskriminatiiviseksi, koska klusterien sisäisestä homogeenisuudesta seuraa klusterien välinen mahdollisimman suuri erilaisuus.

Tässä työssä tutkin diskriminatiiviseen klusterointiin liittyvää optimointiprosessia. Esittelen kolme eri optimointimenetelmää ja vertailen niitä keskenään sekä kokeellisesti että teoreettisesti. Lisäksi käsittelen ja vertailen kolmea regularisointimenetelmää, jotka pohjautuvat informaatioteoriaan ja Bayesilaiseen päättelyyn. Regularisoinnilla tarkoitetaan menetelmän muokkaamista sen optimoinnin helpottamiseksi ja erityisesti yleistyskyvyn parantamiseksi.

Työn kokeelliset tulokset osoittavat, että diskriminatiivinen klusterointi toimii tehtävässään paremmin kuin kolme vertailtavaksi valittua perinteistä klusterointimenetelmää. Regularisoinnin käyttö parantaa menetelmän yleistyskykyä eli toimivuutta aineistoilla joita ei käytetty mallin opettamiseen. Tulosten perusteella suosittelen yhtä optimointimenetelmistä käytettäväksi diskriminatiivisen klusteroinnin yhteydessä. Toisaalta regularisointimenetelmien välille ei löydy selkeää paremmuusjärjestystä.

Avainsanat: Diskriminatiivinen klusterointi, oppiva metriikka, regularisointi, eksploratiivinen data-analyysi

Author: Arto Klami

Title of thesis: Regularized Discriminative Clustering

Finnish title: Regularisoitu diskriminatiivinen klusterointi

Date: 27th February 2003 **Pages:** 68

Department: Department of Engineering Physics and Mathematics

Chair : T-61, Computer and Information Science

Supervisor: Docent Samuel Kaski

Instructor: M.Sc. Janne Sinkkonen

An exploratory data analysis tool called discriminative clustering is studied in this work. The method analyzes vectorial data sets with a categorical auxiliary variable attached to each data sample. The goal is to find clusters of the data samples that would be maximally informative about the auxiliary variable.

Discriminative clustering is based on the principle of learning metrics that is a justified way of creating a good metric for the primary vectorial data space with an aid of auxiliary data. This can be seen as a kind of partially supervised learning, as supervision is used for learning the metric, yet the metric itself is usable with unsupervised methods.

Discriminative clustering partitions the data set to non-overlapping clusters. These clusters are local in the original metric, and homogeneous with respect to the distribution of the auxiliary variable. Hence, discriminative clustering finds clusters that are easy to interpret in terms of the primary data, but are informative about the auxiliary data. The method is called discriminative clustering, as it drives the distributions of different clusters to be maximally different.

I study the optimization process of discriminative clustering. Three different optimization algorithms are presented, and compared both experimentally and theoretically. In addition, three regularization methods, based on information theory and Bayesian inference, are discussed and experimentally analyzed. Regularization modifies the method to make it easier to optimize, and especially to increase the generalization capability.

The experiments show that discriminative clustering outperforms three classical clustering methods in its task. Using regularization improves the method's generalization capability, that is, performance on data sets that were not used for learning the model. Based on the results, I recommend one of the optimization algorithms. On the other hand, no definite ordering of the three regularization methods is found.

Keywords: Discriminative clustering, learning metrics, regularization, exploratory data analysis

Preface

This work was carried out in the Laboratory of Computer and Information Science, Neural Networks Research Centre (NNRC) of the Helsinki University of Technology. The research was funded by the Academy of Finland.

I would like to thank Doc. Samuel Kaski for the supervision, and general guidance on various aspects of the academic world. I would also like to thank my instructor, M.Sc. Janne Sinkkonen, for invaluable directions and comments. The work would have been impossible to do without the two of you.

I have been working in the Learning Metrics for Exploratory Data Analysis group. I am grateful to my colleagues; the discussions in numerous group meetings have been very fruitful and profitable. I am also thankful to the head of the NNRC, Academy Professor Erkki Oja, for providing the excellent facilities and the opportunity to work in this nice environment.

Finally, I would like to thank my fiancée Mikaela for love and support.

Helsinki 27th February 2003

Arto Klami

Contents

List of abbreviations

BMU	Best matching unit
CG	Conjugate gradient
DC	Discriminative clustering
IB	Information bottleneck
MAP	Maximum a posteriori
MDA2	Mixture discriminant analysis 2
MLE	Maximum likelihood estimation
MoG	Mixture of Gaussians
RCA	Relevant component analysis
SA	Simulated annealing
SOM	Self-organizing map
VQ	Vector quantization

List of symbols

$\mathbf{x}$	Real-valued vector
$p(\mathbf{x})$	Probability or probability density of $\mathbf{x}$
S_X	Set of possible realizations of random variable X
$H(X)$	Entropy of random variable X
$I(X;Y)$	Mutual information between random variables X and Y
$d_{KL}(p\|\|q)$	Kullback-Leibler divergence between distributions p and q
V_j	Voronoi region of jth cluster
$d(\mathbf{x},\mathbf{y})$	Distance between $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{m}_j$	Cluster location prototype
$\{\mathbf{m}\}$	Set of cluster location prototypes
$\boldsymbol{\psi}_j$	Cluster distributional prototype
$\{\boldsymbol{\psi}\}$	Set of cluster distributional prototypes
$c(\mathbf{x}_i)$	Auxiliary variable attached to sample $\mathbf{x}_i$
σ	Smoothing parameter or width of a Gaussian function
λ	Regularization parameter
K	Number of components or clusters
N	Number of data samples
L	Dimensionality of feature vectors
L_T	Total dimensionality of an optimization problem
$D^{(x)}$	Primary part of data set
$D^{(c)}$	Auxiliary part of data set
E_{DC}	DC cost function
E_{VQ}	VQ cost function

Chapter 1

Introduction

1.1 Problem setting

Many natural and artificial processes generate vast amounts of data that can be collected with modern computer systems. For example, meteorological stations collect various weather measurements, and complex machines are monitored with several measurement devices. Analyzing such data helps to understand these processes. With the information obtained from the analysis, we can e.g. predict future measurements or develop the machine to work more efficiently.

Unsupervised learning methods aim to ease this analysis by reducing the amount, dimensionality, or complexity of data. The amount of data is usually so high that it is extremely difficult for humans to see any interesting properties in it. With learning methods, we can visualize or summarize the data to simplify further analysis.

This thesis is focused on a particular area of unsupervised learning called clustering, which can be seen as a way of reducing the complexity of data by grouping similar samples together. Consider, for example, a data set of daily weather measurements collected over a period of one year. We can group similar days to clusters, and analyze the clusters instead of the original 365 days. Analyzing all the days separately would have been possible here, but in real applications the number of data samples can be hundreds of thousands or even more.

Traditionally clustering is a fully unsupervised process, that is, data is clustered based only on the data samples that were collected. No criterion for selecting which measurements are important is used, and all variations in data are modeled. In many cases, there exists some additional information about the importance of variation in data. We can, for example, say that the value of some measurement is more important than the values of other measurements. It is reasonable to expect that using such extra information would improve the clustering results.

In this thesis, I discuss a situation where a value of a single categorical variable exists for each data sample in addition to the continuous measurements or features of the sample. In the case of the daily weather measurements, the variable could for example tell the season, i.e. whether it is spring, summer, fall, or winter. It is assumed that this auxiliary variable reveals what is relevant or interesting in the data. A good clustering of data samples is such that takes the relevant properties of data into account, and clusters the data to retain as much information about the auxiliary variable as possible. In our example, it would mean that weather changes indicating season changes would be emphasized. The discriminative clustering (DC) has been proposed for this purpose.

Discriminative clustering optimizes a certain cost function. In this thesis, I study this optimization problem, and try to give suggestions on how to solve some of the difficulties involved in it. To be precise, I study different optimization algorithms, and also present some regularization methods to simplify optimization and improve generalization ability.

1.2 Structure of the thesis

This thesis is organized as follows. First in Chapter 2, I present some general background information related to, or required for understanding, the rest of the thesis. A review of the field of data clustering, and explanations of some common traditional clustering methods are given in Chapter 3. These two chapters introduce the field of machine learning and give a context for discriminative clustering.

Chapters 4 and 5 deal with the main topic of this thesis. In Chapter 4, I present the discriminative clustering model, and discuss different possibilities for learning the parameters of the model. Some theoretical properties and connections are also discussed. Chapter 5 presents the novel contributions of the thesis, namely different types of regularization methods for discriminative clustering.

The performance of the discriminative clustering model is demonstrated on toy and real-world data in Chapter 6. The optimization algorithms and regularization methods are extensively compared on a few data sets. In addition, some properties of the method and the optimization process are studied more extensively on one data set.

Related methods are briefly discussed in Chapter 7. Finally, the thesis is concluded in Chapter 8, where I evaluate the work and give ideas for future research.

1.3 Contributions of the thesis

The discriminative clustering method, along with the optimization methods studied in this thesis, has been published by Samuel Kaski and Janne Sinkkonen in various journal and conference articles, including [28]. Also one of the regularization methods, entropy regularization, has been presented by the same authors [27].

The idea of the Bayesian regularization methods was invented by Samuel Kaski, and the further development has been done jointly by me, Janne Sinkkonen, and Samuel Kaski.

I have contributed by deriving the formulas for the mixture of Gaussians regularization, and by implementing the two Bayesian regularization methods. I have also carried out the extensive testing to compare the performance of the algorithms and regularization methods, and analyzed the results. In addition, I have contributed to the design and implementation of the optimization algorithms.

Chapter 2

Background

This thesis aims to contribute to the field of data analysis. My interest is in the machine learning approach to data analysis; the task is to find interesting properties of data given a set of training samples. The training data is used for fitting a model so that the model learns to represent the data. In other words, the model learns from the data in a sense that after the training it can extract the same properties more effectively from other similar data sets. This approach is especially useful when the amount of data is very large and little prior knowledge of the data is available.

This chapter deals with basic concepts behind machine learning and related fields. Some of them are general knowledge, and some are slightly more special things required for understanding the discriminative clustering model and its connections. Clustering is explained in more detail in the next chapter.

2.1 Brief review of information theory and probability

The methods and algorithms studied in this thesis rely heavily on two bases. The discriminative clustering is a generative probabilistic model, so (Bayesian) probability theory is necessary for understanding it. The other important foundation is information theory, first presented by Shannon [26].

2.1.1 Bayesian probability theory

Probability theory is a quantitative theory of inference under uncertainty. The uncertainty is treated by assigning probabilities to events. One might, for example, assign a probability that it will rain tomorrow. In a more general setting, a probability distribution is a function that gives probabilities for all possible events x of some collection of events S_X. These events are possible realizations of a random variable X.

The frequentist approach to probability is to compute frequencies of events, whereas in Bayesian theory, adopted here, the probabilities are assumed subjective and conditional on prior knowledge. Prior knowledge means information that is known about the system at the moment of probability evaluation. Instead of just giving a probability for raining, we can give a probability that it rains tomorrow, given that it is raining right now. Naturally the probability would be different if today was sunny, so the prior information generally affects the probabilities.

Let us consider here the discrete case, where the set of possible events S_X is finite. The probability of event x conditioned on the prior knowledge of $Y = y$ is denoted by $p(X = x|Y = y)$, or more shortly $p(x|y)$.

Bayesian probability theory consists of two basic rules. The sum rule states that the sum of the probability and its complement is always one, conditioned on whatever information, that is, $p(A = a|B = b) + p(A \neq a|B = b) = 1$. The product rule states that $p(A = a, B = b|C = c) = P(A = a|C = c)p(B = b|A = a, C = c)$.

From the product rule we can derive the Bayes' rule

$$p(a|b, c) = \frac{p(b|a, c)p(a|c)}{p(b|c)} \ . \tag{2.1}$$

Here c tells the prior information, b can be though as a new observation, and a is another event. We can use Bayes' rule to update our beliefs; $p(a|c)$ gives the probability of the event a before the new example b, while $p(a|b, c)$ takes the new observation into account. In the weather example, we can update forecasts according to the new weather observations when they become available.

Another important concept is marginalization. In a full Bayesian treatment, the joint distribution of all variables in the problem, $p(x_1, ..., x_n|c)$, is modeled. Often some of the variables are not really interesting. For example, in the weather case, we might model the joint distribution of temperature and raining probability. If we are considering whether to take an umbrella or not, we only need to know if it is going to rain.

Marginalization means a process, where the uninteresting parameters, often called nuisance parameters, are summed or integrated out. In a two-variable case where y is a nuisance parameter, the marginalization is formulated by

$$p(x|c) = \sum_{y \in S_Y} p(x, y|c) \ . \tag{2.2}$$

This generalizes directly to the multivariate case. Note that the denominator $p(b|c) = \sum_a p(a, b|c)$ in (2.1) is computed by marginalization.

The formulas given above work also in the case of continuous event space, but the summation in the marginalization rule has to be replaced by an integral over the whole event space.

2.1.2 Information theory

Shannon [26] defined information as that which reduces uncertainty, meaning that with information, we can make predictions, as things are not as random as they are without it. Information theory measures the information, and is here briefly explained.

One of the key elements of information theory is entropy. It measures, roughly speaking, the amount of uncertainty in a random variable. For a discrete variable X with probabilities $p(x_i)$ for events $x_i \in S_X$, the entropy is defined as

$$H(X) = -\sum_{x_i \in S_X} p(x_i) \log p(x_i) \, . \tag{2.3}$$

For continuous variables, the sum is replaced with integral and probabilities of outcomes with probability density. The result is called differential entropy, which has similar properties.

High entropy means that the outcome of the random variable is highly uncertain, and the maximum value of entropy is obtained with equal probabilities for all events. In the other extreme case, where $p(x_i) = 1$ for some x_i, the entropy is zero and there is no uncertainty; the realization of X is known to always be that particular x_i.

Entropy is also related to coding length, as it gives a theoretical minimum code length that is on average required to encode the outcomes of the random variable. If the basis of the logarithm is two, then the unit of the code length and entropy is called a bit.

Entropy can also be computed for conditional distributions. The conditional entropy is denoted by $H(X|Y)$, and it tells the entropy of X when Y is already known. We have

$$H(X|Y) = -\sum_{x_i \in S_X, y_j \in S_Y} p(x_i, y_j) \log p(x_i|y_j) \, . \tag{2.4}$$

If X and Y are independent, i.e. $p(x_i, y_j) = p(x_i)p(y_j)$, then the conditional entropy $H(X|Y)$ equals the entropy of X. The minimum of conditional entropy $H(X|Y)$ is obtained if the variables are maximally dependent, that is the realization of Y determines the realization of X without uncertainty.

Another useful concept is mutual information. It gives a measure for how much information we get about one variable if we know another variable. The mutual information can be formulated with entropy and conditional entropy by

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x_i \in S_X, y_j \in S_Y} p(x_i, y_j) \log \frac{p(x_i, y_j)}{p(x_i)p(y_j)} \, . \tag{2.5}$$

This measure if symmetric, that is, $I(X;Y) = I(Y;X)$. Notice that if X and Y are independent, then $I(X;Y) = 0$. High mutual information occurs when the conditional entropy $H(X|Y)$ is low, meaning that the variable Y already gave almost the same information as X.

We will need one more concept from the field of information theory, namely Kullback-Leibler divergence [16]. It is a measure of dissimilarity between two probability distributions p and q, and is defined as

$$d_{KL}(p||q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)} \, . \tag{2.6}$$

This is not a metric between the two distributions as it is not symmetric and does not satisfy the triangle inequality. However, it fulfills the other requirements of a metric: It is non-negative, and zero only if $p(x_i) = q(x_i)$ for all i.

Kullback-Leibler divergence is also called relative entropy. We can think of $d_{KL}(p||q)$ as the average redundancy in the code used to encode the events from distribution p by using a code that is optimal for distribution q.

2.2 Model-based learning

Modern machine learning methods are often based on generative probabilistic models of data, which means that the model defines a probability for all potential data sets, and can in this sense generate samples from the same (or similar) distribution as the data. For example, a model for $\mathbf{x}$ parameterized by θ is written as $p(\mathbf{x}|\theta)$. The model should be selected so that it is able to describe the data well if suitable parameter values are selected. The model can then be used to analyze the data by various ways.

Such a model can be fitted to given data by selecting the parameters θ so that the likelihood of the data is maximized: the parameters are selected to maximize the probability for generating the observed data $\mathbf{x}$. For a finite data set consisting of N samples we thus maximize the likelihood $\prod_i^N p(\mathbf{x}_i|\theta)$ with respect to θ. This process is called maximum likelihood estimation (MLE). No prior information is here taken into account.

The parameters can also be estimated on the basis of the so-called maximum a posteriori (MAP) criterion. It is a Bayesian approach, where suitable prior distribution for θ is chosen, and probability of the model, or actually its posterior $p(\theta|\mathbf{x})$, is maximized instead of the likelihood. The posterior is obtained from the likelihood and the prior by the Bayes' rule (2.1). In MAP estimation we thus maximize

$$p(\theta|\mathbf{x}) = \frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})} \propto p(\mathbf{x}|\theta)p(\theta) \tag{2.7}$$

with respect to θ; the probability $p(\mathbf{x})$ of the observed data can be left out, because it does not depend on θ and thus does not affect the optimization.

In many cases, the logarithm of the likelihood or the posterior is used as an optimization criterion instead of the original criterion. This is done to simplify the computation; the product of individual likelihoods changes to a sum when the logarithm is taken, and sums are usually easier to handle than products.

2.2.1 Supervised and unsupervised learning

Learning methods can be roughly classified into two main categories: supervised and unsupervised. The approaches differ by the methods that are used, and the goal of learning.

Supervised methods learn a mapping from input to a specific output. This is usually achieved by having examples of input-output pairs, and training the model with these pairs. Other kinds of supervision could be used, but often the process can be formulated in the same way. For example, a feedback of a human supervisor can usually be coded as a preferred output variable.

In unsupervised learning, there is no specific output. Only the input samples can be used, and the task is to find some inherent structure in the data, or to reveal some properties of it.

From a probabilistic point of view, we may say that supervised learning methods try to model $p(\mathbf{y}|\mathbf{x})$, that is, the conditional distribution of outputs $\mathbf{y}$ given the inputs $\mathbf{x}$. Often this is achieved by actually modeling the joint distribution $p(\mathbf{y}, \mathbf{x})$ of inputs and outputs, and using the Bayes' rule to get the conditional distribution. In unsupervised learning, we are interested, and have means to handle, only the distribution of inputs $p(\mathbf{x})$.

Some combinations of these approaches have been presented, and a term semi-supervised learning is sometimes used. It might mean, for example, learning from partially labeled data, using supervised methods for a task common for unsupervised learning, or combining supervised methods with unsupervised methods. One example of semi-supervised learning approach is the learning metrics principle presented in the next section.

2.3 Learning metrics

The main topic of this thesis is the discriminative clustering method. It is based on a more general theory, the learning metrics principle [12].

Most learning methods for continuous-valued data rely on distances, that is, we need to compute how far two samples are from each other, or is a sample close to a certain prototype. This means that properties of those methods are largely determined by the selection of the distance measure or metric. This is especially the case in unsupervised learning, as no supervision can be used to select or tune the metric.

Traditional choices for the metric in the case of real valued feature vectors are the Euclidean metric or the metric induced by inner products of feature vectors. In many cases these choices are rather arbitrary. Often some kinds of feature selection or scaling methods are used to improve the situation, i.e. the metric is made to fit the problem at hand by scaling the data space appropriately. This usually requires lots of work and is

often done by an expert of the application field.

The learning metrics principle is designed to automatically create a metric that reflects relevant aspects of data. This means that we create a metric such that the distance between two points is large if the difference between them is relevant to the specific task. Uninteresting differences are mapped to small distances.

The learning metric is created by using additional information that defines what changes in data are relevant. In works published so far, the additional information has been expressed as a categorical auxiliary variable associated with each data sample.

Data points $\mathbf{x}$ are samples from a distribution $p(\mathbf{x})$. With attached auxiliary variables c, they are sampled from $p(c, \mathbf{x})$. We want to measure distances between points in primary space by changes in the distribution of the auxiliary variable. This is motivated by assuming that variation in c means important variation in $\mathbf{x}$. The variations in the conditional distribution $p(c|\mathbf{x})$ then reveal the importance of the differences in the primary data space.

The learning metrics principle defines the distance between two close-by data points $\mathbf{x}$ and $\mathbf{x} + d\mathbf{x}$ to be the difference between the corresponding conditional distributions of c. The difference is measured by the standard Kullback-Leibler divergence d_{KL} (2.6). It has been shown [16] that the divergence is locally equivalent to the quadratic form

$$d(\mathbf{x}, \mathbf{x} + d\mathbf{x}) \equiv d_{KL}(p(c|\mathbf{x})||p(c|\mathbf{x} + d\mathbf{x})) = d\mathbf{x}^T \mathbf{J}(\mathbf{x}) d\mathbf{x} \,, \tag{2.8}$$

where $\mathbf{J}(\mathbf{x})$ is the Fisher information matrix formulated as

$$\mathbf{J}(\mathbf{x}) = E_{p(c|\mathbf{x})} \left[(\nabla_{\mathbf{x}} \log p(c|\mathbf{x})) (\nabla_{\mathbf{x}} \log p(c|\mathbf{x}))^T \right] \,. \tag{2.9}$$

Here $E_{p(c|\mathbf{x})}$ denotes the expectation with respect to the distribution $p(c|\mathbf{x})$. The matrix has been classically used for constructing metrics for probabilistic model families (see e.g. [19]). Here the data vector $\mathbf{x}$ is considered as the parameters of the matrix, hence we construct a new metric for the primary data space. The metric is such that the conditional distribution $p(c|\mathbf{x})$ changes evenly in all directions.

The metric is defined locally for small changes in data. Global distances, if necessary, can be obtained by computing path integrals of (2.8). With these global distances, the learning metrics principle defines a Riemannian metric (see [19]) into the data space. In this thesis, we need not compute global distances.

Notice that we could, in theory, use the Kullback-Leibler divergence to directly compute the distance between two samples that are not close to each other by $d_{KL}(p(c|\mathbf{x}_1)||p(c|\mathbf{x}_2))$. This would lead to the original topology of the space being lost, and the results could not be interpreted in terms of the original data space.

The learning metrics principle only defines the distance measure by (2.8). The principle can be thus realized in many ways. One way is to explicitly estimate the conditional density $p(c|\mathbf{x})$, and then approximate the distances with (2.8) where $p(c|\mathbf{x})$ is replaced

by the estimate. This approach to the learning metrics principle has been studied for example in [13], and is briefly summarized in Chapter 7.

The learning metrics principle resembles supervised learning as suitable auxiliary data is required for learning the metric. The difference is that in supervised learning the purpose is to learn to predict the auxiliary data. Here the goal is to use it to help in analyzing, exploring or mining the primary data. After the metric is learned, it can be used with any unsupervised method and the auxiliary values are no longer needed. Hence, it could be called semi-supervised learning.

2.4 Optimization

As explained in Section 2.2, a generative model is usually fitted to data by maximizing the likelihood of the data or the posterior of the parameters. In practice this means that we have a cost function, and we need to find the values of continuous parameters that maximize it. Model fitting is thus a traditional optimization problem. Usually the optimization algorithms are presented for minimization, and maximization is done by minimizing the negative of the original cost function.

Here I introduce two methods that are used to solve optimization problems in this thesis.

2.4.1 Conjugate gradient algorithm

One of the simplest optimization algorithms is the method of steepest descent. The algorithm proceeds iteratively, and at each step the parameter vector is moved along the gradient of the cost function. One problem with the steepest descent algorithm is that it often converges slowly because of a kind of zig-zag behavior: if a line-search method is used to find the minimum along the gradient direction, then the next direction is forced to always be in right angle with respect to the previous direction.

The zig-zag property can be prevented by selecting the descending directions more wisely. Vectors $\mathbf{d}_1 ... \mathbf{d}_k$ are called H-conjugate if they are linearly independent and if $\mathbf{d}_i^T H \mathbf{d}_j = 0$ for all $i \neq j$.

A quadratic cost function

$$f(\mathbf{x}) = \mathbf{x}^T H \mathbf{x} + c^T \mathbf{x} \quad , \tag{2.10}$$

where $\mathbf{x}$ and c are L-dimensional real-valued vectors and $H \in \mathbb{R}^{L \times L}$, can be minimized in L iterations if a set of L H-conjugate minimization directions are used (see for example [1] for a proof). At each iteration, a minimum along one of the vectors $\mathbf{d}_i$ is searched.

One practical algorithm for performing optimization using conjugate directions is conjugate gradient algorithm (see, for example [1]). It is a gradient-based algorithm, where at each iteration the optimization direction is selected by modifying the current gradient with the previous optimization direction. This results in modest storage requirements,

as only three vectors are required at each step, instead of having to store an estimate of the possibly unknown H, a $L \times L$ matrix.

The first direction $\mathbf{d}_1$ is selected as the negative gradient at the starting location $\mathbf{x}_1$. After that, the following steps are iterated

1. Find the minimum along the direction $\mathbf{d}_i$, that is, find λ_i so that $f(\mathbf{x}_i + \lambda_i \mathbf{d}_i)$ is minimized. Let $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i \mathbf{d}_i$.
2. Let $\mathbf{d}_{i+1} = -\nabla f(\mathbf{x}_{i+1}) + \alpha_i \mathbf{d}_i$, where
$$\alpha_i = \frac{\nabla f(\mathbf{x}_{i+1})^T \left(\nabla f(\mathbf{x}_{i+1}) - \nabla f(\mathbf{x}_i)\right)}{\|\nabla f(\mathbf{x}_i)\|^2} \tag{2.11}$$
3. Stop if $i = L$, otherwise replace i by $i + 1$ and return to step 1.

The rule for α presented here is the so-called Polak-Ribiere variant. Other choices are also possible, and in the case of quadratic cost function and exact line searches, the proposed rules are identical. It is generally considered that the Polak-Ribiere variant outperforms other classical variants when the cost function is non-quadratic.

The algorithm is designed for quadratic problems. However, it can be used also with other kinds of cost functions. Locally all differentiable functions can be approximated by a quadratic function, where H is the Hessian matrix, and conjugate gradient algorithm finds a local minimum of such functions if enough iterations are used.

While the conjugate gradient algorithm is guaranteed to converge in L iterations for quadratic problems, this is not the case with non-quadratic cost functions. More iterations have to be used; in practice the algorithm can be run until converged or for a suitably large predefined number of iterations. It is suggested to restart the procedure every now and then, as the H-conjugacy of the minimizing directions is lost because of numerical inaccuracies, and also because the cost function is not quadratic. Various restarting criteria have been proposed; one of the most often used is to restart to the negative gradient direction after every L iterations.

2.4.2 Simulated annealing

Annealing is a process where hot material is slowly cooled down. The slow cooling enables the material to end up to a structure that is more stable than what is obtained with faster cooling. High stability corresponds to low bounded energy in the structure, so this can be seen as an optimization problem: the system seeks a structure where the energy is the lowest.

Simulated annealing is a stochastic optimization method imitating physical annealing. The method mimics physical annealing by introducing a pseudo-temperature, a variable

that controls how "hot" the optimization problem is. This variable is slowly decreased while the parameters are being optimized.

The optimal solution is sought by making small random changes into the system. In the case of a discrete system (for which the simulated annealing was originally proposed in the works by Metropolis, Kirkpatrick and Černy, see for example [21] for references), we take one variable and change its state. The total energy (or cost function) is evaluated with both the old and the new structure. If the new structure has lower energy, the change is accepted.

If the new structure has higher energy, the change might still be accepted due to "thermal motion". The probability of accepting an energy-increasing change at step t is given by the acceptance function

$$p_A(t) \propto \exp\left(-\frac{(E(t+1) - E(t))}{T_A(t)}\right) , \tag{2.12}$$

where $T_A(t)$ is the temperature and $E(t)$ the energy of the configuration at step t. Higher temperature leads to higher probability of accepting energy-increasing changes.

During the annealing, the temperature is slowly decreased. This means that in the beginning, the configuration makes almost a random walk in the space of all configurations. With lower temperatures, energy-increasing steps are more likely to be rejected, and so the annealing process leads to states with lower energy. If the temperature schedule (how the temperature is decreased) is suitable, the algorithm is guaranteed to converge into a global optimum, though only for a very large number of iterations (see e.g. [21] for discussion).

2.5 Regularization

Regularization refers to a process where a problem is modified so that it becomes possible or easier to solve. For example, solving some problems require inverting a matrix. If the matrix is singular (it has zero determinant) that cannot be done. Also if the matrix is very close to singular, numerical inaccuracies may cause large errors. A simple regularization that helps in both cases, is to modify the original matrix by adding an identity matrix multiplied with a small constant.

Another use for regularization is to improve the generalization capability of the method. It is important that the results obtained on one data set should generalize also to other data sets. This means that a model fitted to some training data should represent also new samples well.

Unfortunately, many learning algorithms and models lead to solutions that are very good for the training data, but do not generalize well. This is called overfitting. It can be overcome in some situations with suitable regularization that prevents the model parameters from drifting to too extreme values. For example, a very flexible model can

be regularized by adding constraints that makes the model more rigid, thus increasing the generalization capability.

The regularization method presented above for the matrix inversion is an example of a more general case, where the problem is modified so that the solution becomes more robust. Often it is done by introducing a simpler problem that is easier to solve, and by changing the current problem slightly towards the easier problem. In the case of model based learning, we can regularize a model, for example, with a simpler model for the same problem. The amount of regularization is often controlled by a variable that determines how large an effect the simpler problem has.

Some classical forms of regularization are interpretable in the Bayesian framework as incorporation of prior knowledge to the solution. A suitable prior density over the parameters is specified, which is equivalent to changing from maximum likelihood estimation to MAP estimation. Increasing the importance of the prior information in MAP estimation can also be considered regularization.

Chapter 3

Traditional clustering

Clustering means a process where groups of similar data samples are sought from data. Consider, for example, a data set consisting of weights and heights of different animal species. The data contains several measurements of each specimen, but we do not know what kinds of animals the samples represent. If we plot the samples into a diagram with weight and height as axes (Figure 3.1), we see how the samples are distributed in the feature space. One should immediately notice that the samples form three separate groups or clusters. By making this kind of clustering, we can more easily analyze the data. We can, for example, analyze each cluster further and try to figure out what kinds of animals the cluster could represent.

While such visual clustering is simple for humans in low-dimensional data spaces, it is a lot more difficult when the dimensionality of the feature space is tens or hundreds, or if the cluster are not as clearly separated as in the example. In such cases, specially designed clustering methods or algorithms can be used.

Clustering methods can be divided roughly to three categories: partitional clustering methods, probabilistic model-based clustering methods, and hierarchical clustering methods. The first two categories are relevant to this thesis, as discriminative clustering shares properties of both categories.

Partitional and model-based clustering methods are explained here, and some examples of actual methods are given. The hierarchical clustering methods are only briefly mentioned.

3.1 Partitional clustering

Partitional clustering divides the data space into non-overlapping regions. These regions present the clusters, and they cover the whole data space. Each data sample belongs to exactly one of these regions, and is thus assigned to the corresponding cluster.

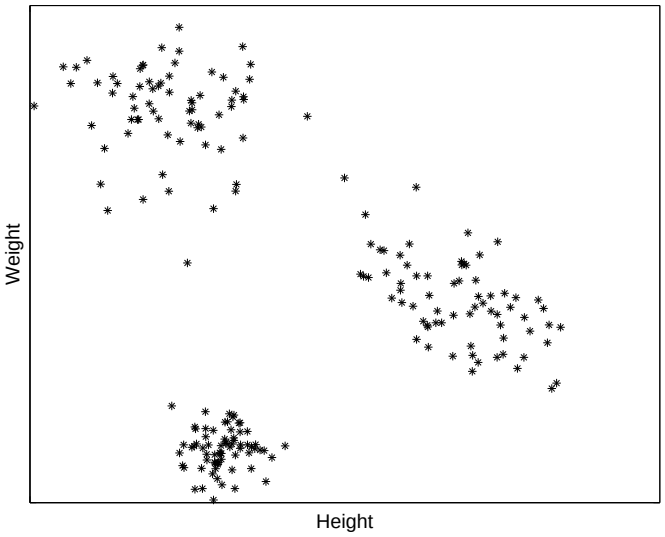

Figure 3.1: Imaginary two-dimensional data set, where it is easy to locate three clusters by simply looking at this plot.

Partitional clustering is conceptionally simple, and the results are easy to interpret. It is especially useful for summarizing large data sets. Because of these properties, partitional clustering methods are widely used.

3.1.1 Vector quantization and K-means clustering

Vector quantization means a process, where data is presented by a small set of prototype vectors. Each data sample is assigned to one of these prototypes. The set of prototypes is called a codebook, because we can think this as coding the data with the prototypes in the codebook.

The prototype vectors are here denoted by $\mathbf{m}_j \in \mathbb{R}^L$, where L is the dimensionality of the data space. The data space is partitioned into Voronoi regions formed by the cluster prototypes. The Voronoi region V_j of prototype $\mathbf{m}_j$ is the part of the data space where that particular prototype is the closest prototype. More formally

$$\mathbf{x} \in V_j \text{ if } d(\mathbf{x}, \mathbf{m}_j) < d(\mathbf{x}, \mathbf{m}_k) \text{ for all } 1 \leq k \leq K \,, \tag{3.1}$$

where K is the number of prototypes in the codebook.

Representing data points by prototype vectors causes errors or distortions since the data points differ from the prototypes. A natural criterion for selecting the codebook is thus to minimize the distortions. Given the probability distribution $p(\mathbf{x})$, the expected distortion or quantization error is

$$E_{VQ} = \sum_{j=1}^{K} \int_{V_j} d(\mathbf{x}, \mathbf{m}_j) p(\mathbf{x}) d\mathbf{x} \,, \tag{3.2}$$

where $d(\mathbf{x}, \mathbf{m}_j)$ measures the distortion between the sample $\mathbf{x}$ and the prototype vector $\mathbf{m}_j$. One popular choice for the distortion function is the squared Euclidean distance. For that choice and a finite data set, the total distortion is

$$E_{VQ} = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in V_j} \|\mathbf{x}_i - \mathbf{m}_j\|^2 . \tag{3.3}$$

For squared Euclidean distortion, an iterative algorithm called K-means [7] can be used for finding the codebook. Each prototype is initialized randomly, and data samples are clustered into the closest prototypes. Next the prototypes are updated to be the centroids of the data samples currently belonging to that cluster. That is

$$\mathbf{m}_j \leftarrow \frac{\sum_{\mathbf{x}_i \in V_j} \mathbf{x}_i}{N_j} , \tag{3.4}$$

where N_j is the number of samples in cluster j.

After this, each sample is again assigned to the cluster with the closest prototype. This assignment and updating the cluster prototypes is repeated until the prototypes do not change (significantly) anymore.

3.2 Probabilistic model-based clustering

Model-based clustering is a way of clustering based on a probabilistic model of data. The model describes the probability of data, and is formed such that we are able to compute the probabilities of samples belonging to clusters, i.e. it we can find out how probable it is that sample $\mathbf{x}_i$ belongs to the jth cluster, denoted by v_j. This is expressed as probability $p(v_j|\mathbf{x}_i)$. The model-based clustering methods thus give us a kind of soft clustering.

Probabilistic models can be used for partitional clustering by selecting the most probable cluster for each data sample. In that case some of the information captured by the probabilistic model is lost, but the results can be interpreted as easily as partitional clustering results.

Here I present some traditional model-based clustering methods relevant to my work. These models can be used also for other tasks than clustering, but in this thesis I focus only on their usage as clustering models. Later, in Chapter 6, I compare discriminative clustering with these methods.

3.2.1 Mixture of Gaussians

A simple way of generating complex probability densities is to write the probability as a sum (or mixture) of simple densities. In the case of continuous data, the Gaussian

function is a suitable simple density function, and thus densities are often expressed as mixtures of Gaussians (see [18] for a textbook account).

The mixture of Gaussians is a generative model, where each component u_j generates samples according to the Gaussian distribution associated with it. The density $p(\mathbf{x})$ is expressed as $p(\mathbf{x}) = \sum_{j=1}^{K} p(\mathbf{x}|u_j)p(u_j)$. Here $p(\mathbf{x}|u_j)$ is the Gaussian density of component u_j, and $p(u_j)$ is the probability of the component. Here K denotes the number of components.

With traditional parameterization, the model is written as

$$p(\mathbf{x}) = \sum_{j=1}^{K} \rho_j \frac{1}{(2\pi)^{L/2} |\Sigma_j|^{1/2}} \exp\left((\mathbf{x} - \mathbf{m}_j)^T \Sigma_j^{-1} (\mathbf{x} - \mathbf{m}_j)^T\right) , \tag{3.5}$$

where ρ_j is the weight of the jth mixture component, that is, it is a parameter representing the probability $p(u_j)$, and L is the dimensionality of the data space. The Σ_j is the covariance matrix of the Gaussian associated with component u_j. Naturally various simplifications are possible, for example the covariance matrices can be forced to be diagonal, or the covariance matrices of different components can be made equal.

The parameters of mixtures of Gaussians are selected by maximizing the logarithmic likelihood of training samples. This can be done efficiently via an expectation maximization algorithm (EM algorithm) [5].

A way to justify the EM algorithm is based on an idea of so-called missing data. Imagine that there exists an indicator variable z_{ji} for each data point $\mathbf{x}_i$ and each component u_j. The indicator variable tells which component generated that particular data point. Thus $z_{ji} = 1$ for one component u_j and zero for the others.

If the values z_{ji} were known, the maximum likelihood solution for the mixture of Gaussians could easily be computed by fitting each Gaussian independently. With the EM algorithm, these missing variables can be used even though they are unknown. The algorithm proceeds iteratively by estimating the probabilities of z_{ji} (the expectation step) and maximizing the likelihood given the estimated probabilities (the maximization step).

For a mixture of Gaussians, the expected value for z_{ji} is just the posterior probability $p(u_j|\mathbf{x}_i)$. The formulas for the maximization step depend on the restrictions posed to the covariance matrices, but all maximizations can be done analytically in a linear time. See, for example, [18] for exact formulas.

The mixture of Gaussians is used for clustering by computing the posterior probabilities of generating components u_j given a data point $\mathbf{x}$. By Bayes' rule we get $p(u_j|\mathbf{x}) \propto p(\mathbf{x}|u_j)p(u_j)$, and each component u_j can be identified as a cluster v_j. For partitional clustering, the v_j with the largest probability is selected.

3.2.2 Supervised clustering: MDA2

Mixture discriminant analysis (MDA2) [8] is a mixture model for the joint probability of one categorical variable and a set of continuous variables collected to the vector $\mathbf{x}$, $p(c, \mathbf{x})$. Clustering with MDA2 can be thought of as clustering $\mathbf{x}$ with supervision available via the variable c.

MDA2 assumes that given the component u_j, $\mathbf{x}$ and c are conditionally independent, that is, $p(c_i, \mathbf{x}|u_j) = p(c_i|u_j)p(\mathbf{x}|u_j)$. Prior probabilities of components and the probability of $\mathbf{x}$ given the component u_j are modeled like in plain mixtures of Gaussians. The probability $p(c_i|u_j)$ is modeled by a multinomial distribution, here parameterized by ξ_{ji}, and thus we get the model

$$p(c_i, \mathbf{x}) = \sum_{j=1}^{K} \rho_j \xi_{ji} \frac{1}{(2\pi\sigma^2)^{L/2}} \exp\left(-\|\mathbf{x} - \mathbf{m}_j\|^2/\sigma^2\right) . \tag{3.6}$$

The model is here written with less general covariances: The covariance matrices of different components are forced to be identical and of the form $\sigma^2 I$ where I is the identity matrix.

The likelihood of $p(c, \mathbf{x})$ is optimized with an EM algorithm similar to the optimization algorithm of mixture of Gaussians. The details can be found in [8].

MDA2 is used for clustering similarly as the mixture of Gaussians. The posterior probability of component u_j given a sample pair $(\mathbf{x}, c)$ is $p(u_j|\mathbf{x}, c) \propto p(\mathbf{x}|u_j)p(c|u_j)p(u_j)$. By marginalization we get $p(u_j|\mathbf{x}) = \sum_k p(u_j|\mathbf{x}, c_k)p(c_k)$, and hence MDA2 can be used for clustering based only on the sample $\mathbf{x}$. Again the cluster identity v_j can be identified with the component u_j, and partitional clustering is achieved by selecting the cluster with the highest probability.

3.2.3 Vector quantization as a probabilistic model

Although vector quantization is a partitional clustering method, it can be formulated probabilistically as the so-called classification mixture model [3]. The probability is defined piece-wise in the Voronoi regions as (unnormalized) Gaussian functions:

$$p(\mathbf{x}_i|\{\mathbf{m}\}) = \frac{1}{Z(\{\mathbf{m}\})} \exp(-\lambda\|\mathbf{x}_i - \mathbf{m}_{j(i)}\|^2) , \tag{3.7}$$

where $\mathbf{m}_{j(i)}$ is the closest prototype for sample $\mathbf{x}_i$, and $Z(\{\mathbf{m}\})$, parameterized by the whole codebook $\{\mathbf{m}\}$, normalizes the integral of probabilities over the whole space to one. Here the parameter λ controls the width of the Gaussians.

The normalization term Z is unknown and difficult to compute as it depends on the size of the Voronoi regions, and therefore also on the cluster prototypes $\mathbf{m}_j$. With a suitable Bayesian treatment, we can get rid of it by assigning a prior for the set of

cluster prototypes: If the somewhat artificial prior $p(\{\mathbf{m}\}) = Z(\{\mathbf{m}\})$ is used, the MAP cost includes only the exponential terms. That prior favors solutions where Z is large, that is, solutions where Voronoi regions are large and cluster prototypes are far from each other. The particular probability to be maximized is

$$\sum_{i=1}^{N} \log p(x_i|\{\mathbf{m}\})p(\{\mathbf{m}\}) = \sum_{i=1}^{N} -\lambda \|x_i - \mathbf{m}_{j(i)}\|^2 \,. \tag{3.8}$$

Maximizing the posterior probability of parameters is thus equivalent to minimizing the quadratic Euclidean distortion resulting from replacing the samples with their closest prototypes. This is exactly the cost function of VQ with squared Euclidean distortion (3.3). Note that the parameter λ does not affect the optimization. By making λ smaller, we effectively increase the separation between cluster prototypes, and thus also increase $Z(\{\mathbf{m}\})$. In the limit $\lambda \to 0$, the $Z(\{\mathbf{m}\})$ approaches a value that is constant with respect to the parameters $\mathbf{m}_j$, making the importance of prior distribution negligible.

3.3 Model order selection

The number of components or clusters K is a free parameter in all of the clustering methods. It is often difficult to determine the correct or optimal number of clusters for a specific task. Sometimes it can be chosen by the properties of the problem, but often the choice is arbitrary.

Various methods have been suggested for selecting the optimal number of clusters. Many are based on the principle of Occam's razor: the simplest adequate explanation is the best. More complex models can effectively minimize the error on training data, but the results do not generalize well to test data because of overfitting.

As described in Section 2.2, the fitting of a model usually maximizes the likelihood or the posterior probability of the model. Effectively this means that the parameters are selected by comparing likelihoods (or posteriors) of models with different parameters. The same approach could in theory be used for selecting the complexity of the model, that is, we could select the number of clusters that maximizes the likelihood.

Model selection by comparing likelihoods directly is, however, usually a bad idea. This is the case especially with nested model families, that is, sets of models where simpler model is a special case of more complex model. With nested models, the optimal likelihood can never decrease when the complexity increases, meaning that the most complex model can always obtain at least the same likelihood as every other model.

Mixture of Gaussians is a nested model family. A mixture of $K+1$ Gaussians can always be used for presenting exactly the same density as a mixture of K Gaussians by setting one of the component probabilities ρ_j to zero.

Many model selection criteria are based on comparing adjusted likelihoods. They apply

the Occam's razor by penalizing the models by their complexity, and thus overcome the problem just described. Different penalization methods have been suggested based on information theory, Bayesian analysis, and coding theory.

None of these model selection criteria are used in this work, but many of them are applicable to the traditional clustering methods presented here and discriminative clustering.

Model selection by validation

A reason for selecting lower order models is that they usually generalize better than complex models. Instead of explicitly penalizing the models by their complexity, we can empirically estimate the generalization capabilities of models by measuring the likelihood of so-called validation set consisting of data not used during the model fitting. If a number of different validation sets are used, we can compute various statistics for the likelihood of unseen test sets. The best model or models can then be selected by comparing the mean values and confidence intervals of the likelihoods of different models.

An efficient way of using limited number of data for estimating generalization ability is the so-called N-fold cross-validation. The data is split into N (roughly) equally sized and mutually exclusive subsets. The model is fitted N times, each time using $N-1$ subsets for fitting and 1 subset for validation. This results in likelihoods of N independent validation sets. We can then select the model that gave the best average likelihood. Cross-validation can also be used to compare the relative performance of different models and optimization algorithms. Then in each fold the independent set is not used for choosing model parameters but only for evaluating the generalization ability.

3.4 Other clustering methods

There are numerous other clustering methods, and the models explained here are just a few examples. In addition to the partitional and probabilistic clustering methods, there exist several clustering methods that either group or divide data iteratively.

The classical agglomerative clustering algorithm starts by assigning each data point into its own cluster. After initialization, the algorithm proceeds iteratively by merging the two most similar clusters into one larger cluster. The similarity is often defined by measuring the average, minimum, or maximum distance between samples of the two clusters. This process is continued until all samples belong to the same cluster.

Divisive clustering algorithms work the other way. The data points are initialized to form a single cluster. The clusters are then iteratively divided according to some similarity criterion until each sample is in its own cluster.

Both agglomerative and divisive clustering methods are hierarchical methods. They provide a number of clusterings ranging from one cluster for all data points to one point

in each cluster. A suitable clustering result can then be selected from this hierarchy, for example by selecting a fixed number of clusters or by searching a stable (in some sense) clustering. The clustering hierarchy is often presented as a so-called dendrogram, a tree-shaped figure (see e.g. [6]).

These types of clustering methods are not studied in this thesis, as they have little in common with the current versions of discriminative clustering.

Chapter 4

Discriminative clustering

4.1 Motivation

In Chapter 3, clustering was discussed as a form of unsupervised learning. As explained in the Section 2.3, the results of unsupervised learning are to a large degree determined by the selected features and metric.

If labels for the vectorial sample are available, then we can apply methods that model the joint distribution, such as MDA2 presented in Chapter 3. There are, however, situations when the traditional Bayesian approach of modeling the joint distribution of variables leads to poor clustering results. If the data, for example, is high-dimensional, but most of the dimensions are irrelevant noise, the joint modeling of variables is inefficient as all the noise dimensions have to be modeled. Naturally this is feasible if nothing indicates which dimensions are relevant.

Discriminative clustering introduced in [28] is proposed to help in situations like this. The idea is to make the clusters internally as homogeneous as possible in terms of the conditional distribution $p(c|\mathbf{x})$, where an auxiliary variable c conveys the relevancy information as suggested by the learning metrics principle explained in Section 2.3. This implies that the differences between the distributions $p(c|\mathbf{x})$ of different clusters are maximized, which gives a reason to call the method discriminative.

The clusters of DC are kept local in the primary space to make the results interpretable in terms of the original features, and thus DC is a practical tool for exploratory data analysis of unknown data sets in the same way as unsupervised clustering methods.

One example use of DC would be to cluster customers based on some background information such as residence, age and sex. The customers' buying behavior can be used as auxiliary data to get clusters that are local in the primary space of background information but at the same time informative about the buying behaviors.

DC also reminds classification in that it models the conditional distribution. However,

the goal of DC is different. In classification, the aim is to predict the classes of future samples well. This is optimally achieved when then changes in $p(c|\mathbf{x})$ are modeled only at the Bayes decision border, the theoretically optimal class separator. DC, on the other hand, represents the changes of $p(c|\mathbf{x})$ in the whole primary space. In the customer example, we could use a classifier to predict whether some customer belongs to a certain buying behavior class. With DC we can also analyze which aspects of the background information explain this, and what kinds of other customers have similar buying behaviors.

Another difference to classification is that the number of clusters can be, and often is, different from the the number of classes (or different values of auxiliary variable). With DC, the data can be clustered into a number of clusters even with binary auxiliary variable. This suggests another use for DC: it can be used for altering the coarseness of an existing classification.

DC is also related to conditional density estimation, as the traditional model-based clustering methods are related to density estimation of $p(\mathbf{x})$. The density estimates are, however, used here only as means to achieve the preferred goal, and they are uninteresting as itself. For plain conditional density estimation, better methods than DC can be presented, and in DC the predictions can even be marginalized out.

4.2 Model for known probability distributions of infinite data sets

Discriminative clustering is defined as vector quantization with distortion measure originating from the learning metrics principle. While the traditional Euclidean vector quantization tries to minimize the distance from samples to cluster centroids, discriminative clustering finds clusters that are internally as homogeneous as possible in terms of the conditional distribution $p(c|\mathbf{x})$ of the auxiliary variable. In addition, the clusters are restricted to be Euclidean Voronoi regions (see Section 3.1.1) in the primary space.

More formally, a set of distributional prototypes $\boldsymbol{\psi}_j$ are introduced, one for each cluster j. These prototypes are parameters of multinomial distributions, so that $\boldsymbol{\psi}_{ji}$ denotes the probability of auxiliary variable c_i given the cluster j. In addition, we have location prototypes $\mathbf{m}_j$ that reside in the primary data space. The distortion made by replacing the sample $\mathbf{x}$ by the cluster prototype $\mathbf{m}_j$ is measured by the Kullback-Leibler divergence between the distributional prototype $\boldsymbol{\psi}_j$ and the conditional distribution $p(c|\mathbf{x})$.

The task of discriminative clustering is hence to minimize the average distortion

$$E_{DC} = \sum_{j=1}^{K} \int_{V_j} d_{KL}(p(c|\mathbf{x})||\boldsymbol{\psi}_j) p(\mathbf{x}) d\mathbf{x} \tag{4.1}$$

over the partitioning V_j and prototypes $\boldsymbol{\psi}_j$, where V_j means the Voronoi region of cluster j. The Voronoi regions are defined by the prototypes $\mathbf{m}_j$ and the Euclidean distance (3.1)

like in Euclidean VQ. This means that while discriminative clustering makes the clusters homogeneous in the learning metric, it still keeps them local as understood traditionally by defining the cluster membership by the Voronoi regions of Euclidean distance.

4.2.1 Stochastic online algorithm

The cost (4.1) is minimized with respect to the both sets of prototypes, $\{\mathbf{m}\}$ and $\{\boldsymbol{\psi}\}$. Gradient-based optimization methods cannot be used for this task as such, as the cost function is piecewise constant. Because of that, only samples located exactly on the borders of the Voronoi regions would affect the gradient.

To overcome this problem, a smoothed variant was introduced [28]. Instead of assigning each sample $\mathbf{x}$ into one cluster, they are assigned with smooth membership functions $y_j(\mathbf{x})$ into all clusters j. The membership function has to be such that $0 \leq y_j(\mathbf{x}) \leq 1$ for all j and $\sum_{j=1}^{K} y_j(\mathbf{x}) = 1$.

The cost function with the smoothing becomes

$$E_{DC} = \sum_{j=1}^{K} \int y_j(\mathbf{x}) d_{KL}(p(c|\mathbf{x})||\boldsymbol{\psi}_j) p(\mathbf{x}) d\mathbf{x} \; . \tag{4.2}$$

The membership functions y_j are here chosen to be normalized spherical Gaussians with means equal to the cluster prototypes $\mathbf{m}_j$, that is

$$y_j(\mathbf{x}) = \frac{\exp(-\|\mathbf{x} - \mathbf{m}_j\|^2/\sigma^2)}{\sum_k \exp(-\|\mathbf{x} - \mathbf{m}_k\|^2/\sigma^2)} \; , \tag{4.3}$$

where σ governs the amount of smoothing. Small values of σ mean that only little smoothing is used, and at the limit of zero, the smoothed version equals the original formulation. The parameter σ is not optimized with the optimization algorithm presented below, but is selected by validation. More general forms besides spherical Gaussians for smoothing could naturally be used, in the expense of more complicated computation.

The smoothed cost function can be optimized with standard gradient-based algorithms, such as the stochastic approximation algorithm originally introduced in [28]. As a preliminary step, distributional prototypes are reparameterized by the soft-max, that is, $\log \boldsymbol{\psi}_{ji} = \gamma_{ji} - \log \sum_k \exp(\gamma_{jk})$. The parameters γ_j can be freely modified, yet the prototypes $\boldsymbol{\psi}_{ji}$ stay summed to unity.

The algorithm proceeds iteratively by taking one data point $\mathbf{x}(t)$ and its corresponding auxiliary variable $c(\mathbf{x}(t))$ (denoted by i) at a time. For each sample, two clusters, j and l, are drawn independently with probabilities given by the membership functions $y_j(\mathbf{x}(t))$. The prototypes are adapted by

$$\mathbf{m}_j(t+1) = \mathbf{m}_j(t) - \alpha(t) \left[\mathbf{x}(t) - \mathbf{m}_j(t)\right] \log \frac{\boldsymbol{\psi}_{li}(t)}{\boldsymbol{\psi}_{ji}(t)} \text{ and} \tag{4.4}$$

$$\gamma_{jm}(t+1) = \gamma_{jm}(t) - \alpha(t) \left[\boldsymbol{\psi}_{jm}(t) - \delta_{mi}\right] \; , \tag{4.5}$$

where δ_{mi} is the Kronecker delta, and $\alpha(t)$ is a learning parameter.

The clusters j and l are drawn independently, so they are interchangeable in the update rule. We can take advantage of this by making a second update with j and l swapped. Note that if $j = l$, the location prototypes are not updated at all.

The learning parameter $\alpha(t)$ is decreased gradually towards zero during the learning. If the decreasing schedule fulfills the conditions of the stochastic approximation theory (see [17]), the algorithm is guaranteed to converge (see [11] for proof).

4.3 Finite data version and optimization of the marginalized likelihood

For finite data with unknown distribution $p(\mathbf{x})$, minimizing the cost function (4.1) is equivalent to maximizing

$$L_{DC} = \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in V_j} \log \boldsymbol{\psi}_{j,c(\mathbf{x}_i)} \ , \tag{4.6}$$

where $c(\mathbf{x}_i)$ is the auxiliary variable of sample $\mathbf{x}_i$. This is the logarithmic likelihood of a piece-wise constant conditional density estimator. The conditional density $p(c_i|\mathbf{x})$ is predicted to be $\boldsymbol{\psi}_{ji}$ within the Voronoi region of cluster j.

The connection can be shown as follows. The cost (4.1) can be written as

$$\begin{aligned} E_{DC} &= \sum_{j=1}^{K} \int_{V_j} \sum_i p(c_i|\mathbf{x}) \log \frac{p(c_i|\mathbf{x})}{\boldsymbol{\psi}_{j,c_i}} p(\mathbf{x}) d\mathbf{x} \\ &= \sum_j \int_{V_j} \sum_i p(c_i, \mathbf{x}) \log p(c_i|\mathbf{x}) d\mathbf{x} - \sum_j \int_{V_j} \sum_i p(c_i, \mathbf{x}) \log \boldsymbol{\psi}_{j,c_i} d\mathbf{x} \ . \end{aligned} \tag{4.7}$$

Here the first term (conditional entropy $H(C|X)$) is constant with respect to the parameters, and thus does not affect the optimization. For finite data, the integral over V_j with probability measure $p(c_i, \mathbf{x})$ (the second term) corresponds to the likelihood of the sample pairs $\{\mathbf{x}_i, c(\mathbf{x}_i)\}_i$. Therefore maximizing the likelihood (4.6) is asymptotically equivalent to minimizing (4.1).

The interesting outcome of discriminative clustering is the clustering result, that is, the assignments of data samples into the clusters. The locations of the cluster prototypes $\mathbf{m}_j$ are additionally useful, as they can be used to summarize the data. On the other hand, the distributional prototypes $\boldsymbol{\psi}$ are often unimportant. They are needed during the optimization, but they do not provide any insights on the final result. In fact, the distributional probabilities might even cause problems: two models with equal prototypes $\mathbf{m}_j$ but different prototypes $\boldsymbol{\psi}_j$ produce identical clusters with unequal cost function values.

As explained in Chapter 2, the marginalization principle can be used for handling these nuisance parameters. For that purpose we need a prior distribution for the distributional prototypes. It is convenient to use the Dirichlet prior, which is a so-called conjugate prior for the multinomial distribution. Conjugate prior is a prior which produces a posterior in the same parameter family, thus making inference computationally easier.

For a full Bayesian treatment, a prior is needed also for the location prototypes $\mathbf{m}_j$; here a uniform prior was selected. With the incorporated prior information, the model is optimized by maximizing the partly marginalized posterior, leading to the term MAP estimation being used in publications [27, 29].

Denote the observed primary data set by $D^{(x)}$, and the auxiliary data set by $D^{(c)}$. The marginalized posterior can then be written as

$$p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) = \int_{\{\boldsymbol{\psi}\}} p(\{\mathbf{m}\}, \{\boldsymbol{\psi}\}|D^{(c)}, D^{(x)}) d\{\boldsymbol{\psi}\} \; . \tag{4.8}$$

By assigning a separable prior $p(\{\mathbf{m}\}, \{\boldsymbol{\psi}\}) = \prod_j p(\boldsymbol{\psi}_j)$ and applying the Bayes' rule we get

$$\begin{aligned} p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) &\propto \int_{\{\boldsymbol{\psi}\}} p(D^{(c)}|\{\mathbf{m}\}, \{\boldsymbol{\psi}\}, D^{(x)}) p(\{\boldsymbol{\psi}\}) d\{\boldsymbol{\psi}\} \\ &= \prod_j \int_{\boldsymbol{\psi}_j} p(D_j^{(c)}|\boldsymbol{\psi}_j) p(\boldsymbol{\psi}_j) \, d\boldsymbol{\psi}_j = \prod_j \int_{\boldsymbol{\psi}_j} \prod_i \boldsymbol{\psi}_{ji}^{n_{ji}} p(\boldsymbol{\psi}_j) \, d\boldsymbol{\psi}_j \; . \end{aligned} \tag{4.9}$$

Here $D_j^{(c)}$ means the subset of auxiliary data in the cluster j, and n_{ji} is the number of samples with auxiliary variable c_i in cluster j.

The Dirichlet prior is used for the distributional probabilities: $p(\boldsymbol{\psi}_j) = \prod_i \boldsymbol{\psi}_{ji}^{n_i^0 - 1}$. Here $n^0 = \{n_i^0\}_i$ are the prior parameters common to all j.

The resulting integral is known as the Dirichlet integral, and the analytical solution is familiar from the normalization of the Dirichlet distribution. The result is

$$p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) \propto \prod_j \int_{\boldsymbol{\psi}_j} \prod_i \boldsymbol{\psi}_{ji}^{n_i^0 + n_{ji} - 1} \, d\boldsymbol{\psi}_j = \prod_j \frac{\prod_i \Gamma(n_i^0 + n_{ji})}{\Gamma\left(N^0 + N_j\right)} \; , \tag{4.10}$$

where Γ is the gamma function. Here N_j is the total number of samples in cluster j, and $N^0 = \sum_i n_i^0$.

Finally, we take the logarithm of the posterior to simplify the optimization process, so the cost function to be maximized becomes

$$\log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) = \sum_{ij} \log \Gamma(n_i^0 + n_{ji}) - \sum_j \log \Gamma(N^0 + N_j) \; . \tag{4.11}$$

Notice that the optimal solution for this cost function is the maximum a posteriori estimate for the piece-wise constant conditional density approximation. The cost can be used to compare clustering methods in the task of DC, regardless of the optimization criterion or algorithm.

4.3.1 Optimization by conjugate gradients

The finite data version shares the same difficulty as the infinite data version: the gradient of the cost function is zero almost everywhere. Fortunately, the same solution is applicable, that is, the hard membership function is replaced with a soft one. Note that the smoothing is only used as a trick to allow optimization; the cost (4.11) is still used for evaluating the performance.

The posterior depends only on the numbers of samples in the clusters. The smoothing can be applied by computing a kind of smoothed numbers, instead of those resulting from the hard assignment. The smoothed number of samples of auxiliary variable c_i in cluster j is $n_{ji} = \sum_{c(\mathbf{x})=i} y_j(\mathbf{x})$, where $y_j(\mathbf{x})$ is the smoothing function. With the smoothing, the MAP cost function becomes

$$\log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) = \sum_{ij} \log\Gamma\left(n_i^0 + \sum_{c(\mathbf{x})=i} y_j(\mathbf{x})\right) - \sum_j \log\Gamma\left(N^0 + \sum_{\mathbf{x}} y_j(\mathbf{x})\right) . \tag{4.12}$$

The smoothing function (4.3) is used also here. The parameter σ again governs the amount of smoothing, and is selected by validation. With this choice, the gradient of the cost function with respect to the jth location parameter is

$$\sigma^2 \frac{\partial}{\partial \mathbf{m}_j} \log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) = \sum_{\mathbf{x},l} (\mathbf{x} - \mathbf{m}_j) y_l(\mathbf{x}) y_j(\mathbf{x}) (L_{j,c(\mathbf{x})} - L_{l,c(\mathbf{x})}) , \tag{4.13}$$

where

$$L_{ji} \equiv \Psi(n_{ji} + n_i^0) - \Psi(N_j + N^0) . \tag{4.14}$$

Here Ψ is the digamma function, the derivative of the logarithm of the gamma function Γ. The derivation of the gradient is presented in detail in Appendix A.

With the gradient information, the conjugate gradient algorithm explained in Section 2.4.1 can be used for optimizing the cost function. As the problem is non-quadratic, more iterations than the dimensionality $L_T = KL$ of the problem are applied. The descension direction $\mathbf{d}_i$ is restarted to the negative gradient after every L_T steps.

4.3.2 Optimization by simulated annealing

Introducing the soft membership functions allows the use of gradient-based optimization algorithms. However, presenting such smoothing clearly alters the problem. The functional form of the soft membership functions and the amount of smoothing affect the problem, and it cannot be theoretically guaranteed that the optimal solution to the soft version is actually optimal (or even close) to the problem without smoothing.

An alternative presented here is to optimize the cost function with simulated annealing, presented in Section 2.4.2. As the cluster prototypes $\mathbf{m}_j$ are parameterized by continuous

functions, a continuous variant of the simulated annealing algorithm, originally proposed for state-based optimization, is needed. Using simulated annealing for solving continuous optimization problems is studied for example in [21].

The main difficulty when applying simulated annealing to continuous problems is how to generate a new candidate solution. In the discrete case, it is often simple to switch one of the variables to another state, but in the continuous case we have to generate one of infinitely many possible candidates.

A simple solution is to use a so-called jumping kernel, a distribution with mean equal to the current solution candidate, from which the new solution candidate is sampled. In this work, an independent jumping kernel for each location prototype $\mathbf{m}_j$ is used. In practice, this means that each $\mathbf{m}_j$ is altered slightly at every step. As a conventional and computationally simple choice, the jumping kernels were selected to be a spherical Gaussians centered on the current prototype. The variances of the cluster-wise jumping kernels were set equal.

In an Adaptive Simulated Annealing (ASA) algorithm [10] used here, a decreasing temperature is used not only in the acceptance function, but also in the jumping kernel. The width of the sampling distribution is decreased with decreasing temperature. Changes to the solution candidate therefore get smaller and smaller during the optimization process.

Two temperature schedules are needed; one for the jumping kernel width and another for the acceptance temperature. The jumping kernel width was selected to decrease by the schedule

$$T_W(t) = \frac{T_W(0)}{\log t + C} \,, \tag{4.15}$$

where t is the iteration step, and the initial temperature $T_W(0)$ and C are constants. This selection fulfills the theoretical convergence criteria [21] for the jumping kernels

$$p(\mathbf{m}_j(t+1)|\mathbf{m}_j(t)) \propto \exp\left(-\frac{\|\mathbf{m}_j(t+1) - \mathbf{m}_j(t)\|^2}{\sigma^2 \sqrt{T_W(t)}}\right) \tag{4.16}$$

used in this thesis. As the initial width of the jumping kernels can be controlled by the parameter σ, the initial temperature $T_W(0)$ was fixed to be one. The constant C was set to 10.

The only theoretical requirement for the schedule of acceptance temperature T_A used in (2.12) is that it should decrease monotonically towards zero. Here the schedule

$$T_A(t) = T_A(0)\left(1.0 - \frac{t}{t_{MAX}}\right) \tag{4.17}$$

was used, where t_{MAX} is the total number of iterations, and $T_A(0)$ is the initial acceptance temperature. The initial temperature was selected for each optimization by running the algorithm for 5000 steps with $T_A(t) = 1$, and computing the average difference in energy (E_{AVE}) for the energy-increasing steps of the last 4000 iterations. The initial temperature is then selected by $T_A(0) = E_{AVE}/2$. This type of temperature selection

should lead to a reasonable number of energy-increasing steps to be accepted, as the acceptance is roughly proportional to $\exp(-E_{AVE}/T_A)$. The divisor 2 was selected by experimenting with various values, and selecting the one that seemed to provide good results. Better justified choices could have been made, or the parameter could have been selected by validation.

4.4 Properties

4.4.1 Connection to learning metrics

In Section 4.2, the connection to learning metrics was given only by the fact that the distortion is measured by the Kullback-Leibler divergence of the conditional distributions $p(c|\mathbf{x})$. A more formal connection to the principle of learning metrics can be derived with suitable assumptions.

Learning metrics means using (2.8) as the local distance measure. It has been shown [27] that DC is vector quantization with that distance measure. The connection is, however, only theoretical, as it holds only in the asymptotic limit of large number of clusters. In practice, the number of clusters is small.

Assume that the densities $p(c|\mathbf{x})$ are differentiable, and that they become arbitrarily close to linear with respect to $\mathbf{x}$ within almost all Voronoi regions as the number of clusters is increased. Here "almost all" refers to the density $p(\mathbf{x})$, and means that the linearity assumption does not have to hold for Voronoi regions that have asymptotically vanishing probability, for example the non-compact Voronoi regions at the borders of the data space.

Denote the expectation over Voronoi region V_j with respect to the probability $p(\mathbf{x})$ by E_{V_j}. At the optimum of the cost function (4.1), we have $\boldsymbol{\psi}_j = E_{V_j}[p(c|\mathbf{x})]$. This is easy to see from (4.6); as the probabilities are predicted to be constant inside the Voronoi region, the optimal solution for finite data set is $\boldsymbol{\psi}_{ji} = \frac{n_{ji}}{N_j}$. For data distributions this corresponds to the expectation of the conditional probabilities.

Since $p(c|\mathbf{x})$ was assumed linear within Voronoi region, there exists a linear operator T_j for each V_j, for which $p(c|\mathbf{x}) = T_j\mathbf{x}$. We can write

$$\boldsymbol{\psi}_j = E_{V_j}[p(c|\mathbf{x})] = E_{V_j}[T_j\mathbf{x}] = T_j E_{V_j}[\mathbf{x}] \equiv T_j\tilde{\mathbf{m}}_j = p(c|\tilde{\mathbf{m}}_j) \;, \tag{4.18}$$

where $\tilde{\mathbf{m}}_j = E_{V_j}[\mathbf{x}]$.

At the optimum, the cost function (4.1) then becomes

$$E_{DC} = \sum_j \int_{V_j} d_{KL}\left(p(c|\mathbf{x}), p(c|\tilde{\mathbf{m}}_j)\right) p(\mathbf{x}) d\mathbf{x} \;. \tag{4.19}$$

In the original formulation, the optimum was obtained when $\boldsymbol{\psi} = E_{V_j}[p(c|\mathbf{x})]$. For that, the average over the Voronoi region would have to be computed. With the linearity

assumption, the distortion can be measured directly with respect to the distribution $p(c|\tilde{\mathbf{m}}_j)$.

As the Voronoi regions are local in the primary space, we can further modify the cost function with the local approximation of the Kullback-Leibler divergence:

$$E_{DC} = \sum_j \int_{V_j} (\mathbf{x} - \tilde{\mathbf{m}}_j)^T \, \mathbf{J}(\tilde{\mathbf{m}}_j) \, (\mathbf{x} - \tilde{\mathbf{m}}_j) \, p(\mathbf{x}) d\mathbf{x} \; , \tag{4.20}$$

i.e., the distortion is the local distance measure (2.8) of learning metrics. Note that the Voronoi regions V_j are still defined in the original metric.

4.4.2 Connection to mutual information

Minimizing the cost function (4.1) equals maximizing the mutual information (2.5) between the auxiliary variable C and cluster identity V considered as random variables [27]. Let us concentrate on the smoothed infinite data variant to simplify the proof.

Denote the cluster identity by a random variable V so that v_j means jth cluster. The membership functions can be interpreted by $y_j(\mathbf{x}) \equiv p(v_j|\mathbf{x})$. At the optimum of the cost function (4.1), the prototypes $\boldsymbol{\psi}_j$ have the value $E_{V_j}[p(c|\mathbf{x})] \equiv p(c|v_j)$ as explained in the previous section.

Instead of the cost function (4.1), we can use the likelihood interpretation (4.6). With the smoothing, minimizing the original cost function is thus equivalent to minimizing

$$E_{DC} = -\sum_{j,i} \int y_j(\mathbf{x}) p(c_i|\mathbf{x}) p(\mathbf{x}) \log \boldsymbol{\psi}_{ji} d\mathbf{x} \; . \tag{4.21}$$

At the optimum,

$$\begin{aligned} E_{DC} &= -\sum_{j,i} \int p(v_j|\mathbf{x}) p(c_i|\mathbf{x}) p(\mathbf{x}) \log p(c_i|v_j) d\mathbf{x} \\ &= -\sum_{j,i} p(v_j) p(c_i) \log p(c_i|v_j) \\ &= -\sum_{j,i} p(c_i, v_j) \log \frac{p(c_i, v_j)}{p(v_j) p(c_j)} + \sum_{j,i} p(c_i, v_j) \log p(c_j) \\ &= -I(C;V) + \text{const.} \end{aligned} \tag{4.22}$$

Minimizing the DC cost is thus equivalent to maximizing the mutual information between C and V.

The same connection holds also for the finite data version. For a fixed number of clusters, maximizing the marginalized MAP cost function (4.11) becomes asymptotically equivalent to maximizing the mutual information as the number of data samples increase. The proof [27] can be found in Appendix C.

4.4.3 Connection to contingency tables

Numerous classical methods based on contingency tables exist for measuring statistical dependency between categorical random variables. The measurements are collected into a table of co-occurrences of the possible values of the two variables. The row and column variables represent the covariates whose dependency is tested.

We can naturally interpret any clustering result as a contingency table, where the clusters correspond to the rows and the possible values of discrete auxiliary variable correspond to the columns. Now the row categories are not fixed; instead a clustering method is used to find a suitable categorization. It can be shown that MAP DC produces maximally dependent contingency tables in a Bayesian sense, under the constraints of the model. See [27, 29] for proof and detailed analysis of this connection.

Chapter 5

Regularized discriminative clustering

As explained in Chapter 2, the generalization ability of models can often be increased with suitable regularization. In addition, regularization can help solving difficult optimization problems. Based on these facts, discriminative clustering should benefit from regularization, as it is difficult to optimize and it is meant for clustering unseen data sets.

Here I present two types of regularizations suitable for discriminative clustering. Both regularization types are proposed for the finite data version of discriminative clustering. In theory, similar kinds of regularizations could be used also with the infinite data variant. However, the finite data variant seems more promising (see results in [27]), and is better justified for real-life applications, so the effort is focused on it.

5.1 Entropy regularization

Early tests with the DC algorithms showed that occasionally one or more of the cluster prototypes drifted far from training data. This sometimes resulted in clusters with no data samples at all. This is equivalent to using the model with a smaller number of clusters. The results were, however, better when this did not happen, so the reason for selecting a smaller number of clusters did not originate from the data, but rather from problems in the optimization process.

One way of preventing empty clusters is to favor solutions where clusters have a roughly equal number of samples. If some clusters have no samples at all, the distribution of samples is clearly uneven. The results may also be easier to interpret interpret if the distribution of the samples is more even.

The proportion N_j/N can be interpreted as the probability p_j of a random sample

belonging to the cluster v_j. The entropy (2.3) of this probability distribution measures how evenly the samples are distributed, and obtains its maximum when each cluster has an equal probability, or in the case of finite data, equal number of samples.

Penalizing or regularizing the DC by adding a small amount of the entropy of the cluster probabilities, or an equivalent measure of evenness was suggested in [27]. This regularization method is simple to implement by increasing the multiplier in front of the second term in the cost function (4.11):

$$\log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) = \sum_{ij} \log \Gamma(n_i^0 + n_{ji}) - (1+\lambda) \sum_j \log \Gamma(N^0 + N_j) \; . \tag{5.1}$$

The value $\lambda \geq 0$ governs the amount of regularization. The value $\lambda = 0$ means that no regularization is used, and the cost function reverts back to the unregularized cost (4.11). Larger values regularize the solution. The connection [27] of the cost function (5.1) and entropy is derived in Appendix C as a by-product of the proof for the mutual information connection presented in the previous chapter. Note that the connection is again only asymptotic and holds at the limit of large number of samples.

5.2 Regularization by modeling the joint density

The discriminative clustering model is effectively modeling the conditional distribution $p(c|\mathbf{x})$. An alternative approach to discriminative modeling would be to model $p(c, \mathbf{x})$ and to infer the conditional distribution from it.

Ng and Jordan have recently shown [20] that modeling the joint distribution instead of the conditional distribution could be useful in some situations, even when the task is to model the conditional distribution. Their results suggest that joint probability models can converge more rapidly, meaning that they can be optimized in shorter time. Joint probability models seem to be especially useful with small training data sets, as a kind of a regularization method.

These results give support for the idea of complementing the DC model with a generative model of $p(\mathbf{x})$ to obtain

$$p(c, \mathbf{x}|\{\mathbf{m}\}, \boldsymbol{\psi}) = p(c|\mathbf{x}, \{\mathbf{m}\}, \boldsymbol{\psi}) p(\mathbf{x}|\{\mathbf{m}\}) \; . \tag{5.2}$$

Note that both models are parameterized by the same cluster prototypes $\mathbf{m}_j$ to connect the otherwise distinct models. The model for $p(\mathbf{x})$ is here selected so that the compromise between $p(c|\mathbf{x})$ and $p(\mathbf{x})$ can be tuned explicitly, which is not always the case with standard joint models (for example MDA2, presented in Chapter 3).

Remember that the MAP DC was originally formulated with a uniform prior for $p(\{\mathbf{m}\})$. The model $p(\mathbf{x}|\{\mathbf{m}\})$ can be interpreted as a "prior density" for the cluster prototypes if we apply the Bayes' rule to get

$$p(D^{(x)}|\{\mathbf{m}\}) = \frac{p(\{\mathbf{m}\}|D^{(x)}) p(D^{(x)})}{p(\{\mathbf{m}\})} \propto p(\{\mathbf{m}\}|D^{(x)}) \; . \tag{5.3}$$

Here we assume a uniform hyperprior to the cluster prototypes and drop the constant $p(D^{(x)})$. Prototypes located on dense areas of primary data would then be favored, which should regularize the results by making likelihood computation more robust and by preventing empty clusters.

This is not a real prior as it depends on the data. Thus we have to think of this as a two-stage process: we first infer the posterior distribution of $\{\mathbf{m}\}$ given the primary data $D^{(x)}$. Then the posterior distribution is used as a "prior" for the DC stage. Because of these interpretations, the regularization by joint modeling is later on called Bayesian regularization.

The regularized cost function is formulated as the marginalized posterior

$$p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) \propto \int_{\{\boldsymbol{\psi}\}} p(D^{(c)}|\{\mathbf{m}\}, \{\boldsymbol{\psi}\}, D^{(x)}) p(D^{(x)}|\{\mathbf{m}\}) p(\{\boldsymbol{\psi}\}) d\{\boldsymbol{\psi}\} \ , \qquad (5.4)$$

which is exactly the marginalized posterior of unregularized DC complemented with the model for $p(\mathbf{x})$. After computing the Dirichlet integral as in Section 4.3, the terms can be separated. We end up with the log-posterior

$$\log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) \propto \sum_{ij} \log \Gamma(n_i^0 + n_{ji}) - \sum_j \log \Gamma(N^0 + N_j) + \log p(D^{(x)}|\{\mathbf{m}\}) \ , \quad (5.5)$$

which is used as a cost function in the optimization process. In the following two sections, two models of $p(\mathbf{x}|\{\mathbf{m}\})$ useful for regularizing DC are presented.

5.2.1 Euclidean vector quantization as prior density

Discriminative clustering is essentially a vector quantization method, with the distortion measured by the Kullback-Leibler divergence of $p(c|\mathbf{x})$. Intuitively, adding a small portion of the traditional Euclidean distortion of vector quantization to the distortion would regularize the method, as the density $p(\mathbf{x})$ would be taken into account.

In Chapter 3, a probabilistic model (3.7) equivalent to the traditional vector quantization was given. As it is a model for $p(\mathbf{x})$, it can be used for regularizing the DC solutions as described in the previous section. This is a justified way of realizing the intuitive idea of the previous paragraph, and can be used as an extra motivation for this type of regularizations.

As we plug the classification mixture model (3.7), parameterized by same cluster prototypes $\mathbf{m}_j$, into the cost function (5.5), we get the regularized cost

$$\begin{aligned} \log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) \ &\propto \ \sum_{ij} \log \Gamma(n_i^0 + n_{ji}) - \sum_j \log \Gamma(N^0 + N_j) \\ &- \ \lambda \sum_j \sum_{\mathbf{x} \in V_j} \|\mathbf{x} - \mathbf{m}_j\|^2 \ . \end{aligned} \qquad (5.6)$$

It follows the intuitive idea of regularizing by Euclidean distortion to the cost function. The parameter $\lambda \geq 0$ controls the amount of regularization with zero meaning no regularization. The value of λ is below chosen by validation.

The cost function can be optimized much like the unregularized one. By applying the smoothing for the DC part to get the gradient information, and computing the gradient of the VQ part, the conjugate gradient algorithm can be used. In simulated annealing, regularization only affects the computation of the cost function.

5.2.2 Mixture of Gaussians as prior density

Mixture of Gaussians is perhaps a better justified density model than the classification mixture (which leads to the vector quantization), and using a mixture of Gaussians to regularize discriminative clustering could therefore be a good idea. The intuitive idea of regularizing with the Euclidean distance is now lost, but regularizing by joint modeling is reasonable in itself.

The mixture of Gaussians model (3.5) was given in Chapter 3. Plugging it into (5.5) gives

$$\begin{aligned} \log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) \quad &\propto \quad \sum_{ij} \log \Gamma(n_i^0 + n_{ji}) - \sum_j \log \Gamma(N^0 + N_j) \\ &+ \quad \sum_{\mathbf{x} \in D^{(x)}} \log \sum_j \rho_j \exp\left(-\lambda \|\mathbf{x} - \mathbf{m}_j\|^2\right) \ . \end{aligned} \tag{5.7}$$

Gaussians with covariance matrix $\frac{1}{2\lambda} I$ have been chosen as the mixture components. The parameter $\lambda > 0$ is thus related to the width of the Gaussians. Smaller λ means less regularization, and at the limit of zero the distribution $p(\mathbf{x}|\{\mathbf{m}\})$ becomes uniform, thus reverting to the unregularized case.

This model has more parameters to be optimized than the unregularized or VQ-regularized versions, because in addition to the cluster prototypes $\mathbf{m}_j$ also the mixing weights ρ_j have to be optimized. Traditionally all the parameters of mixture of Gaussians are optimized with the EM algorithm. The approach is not usable now, because either conjugate gradients or simulated annealing is used for optimizing the other part of the cost function.

The gradient with respect to the cluster prototypes and mixing weights is fairly simple to compute, and the conjugate gradient algorithm could therefore be used. The full derivations are shown in Appendix B.

The cluster prototypes $\mathbf{m}_j$ are also easy to optimize with simulated annealing, as it is sufficient that the value of the cost function can be computed. For optimizing the mixing weights there are a few possible alternatives. The alternative most faithful to the original formulation of SA would be to introduce a jumping kernel also for the mixing weights, and then optimize the whole parameter space with SA.

Another approach, used here, is to optimize the cluster prototypes with simulated an-

nealing, but to adapt the mixing weights using the analytical solution

$$\rho_j(t+1) = \frac{1}{N}\sum_i p(u_j|\mathbf{x}_i) \tag{5.8}$$

after each SA step. It means that the mixing weights are set to the posterior probabilities of cluster identities after each accepted SA step. This algorithm keeps the parameter space of the problem smaller, but might lead to slightly suboptimal convergence.

Chapter 6

Experiments

This chapter reports empirical tests showing that discriminative clustering works effectively. Different optimization and regularization methods are also benchmarked.

Before the actual tests, I illustrate the capabilities of discriminative clustering with an artificial toy data to give a better idea about how the algorithm works and what it is supposed to do.

6.1 Demonstration on toy data

The learning metrics principle formulates how to do unsupervised learning in a metric that is learning based on auxiliary data. The metric is such that only relevant properties of data contribute to distances. Discriminative clustering follows this principle, so it should ignore irrelevant features of data when clustering.

This property is here illustrated on a simple toy data. The data has two dimensions, but only one of them is relevant. The relevance information is given to discriminative clustering as a binary auxiliary variable c. The primary data (10000 samples for learning) is sampled from a Gaussian distribution, and the distribution of the binary values varies only in the vertical direction. The probability $p(c_1|\mathbf{x})$ approaches one in the limit of large vertical coordinate, and zero in the limit of small. The probability for c_2 has, naturally, the opposite behavior, and both probabilities change monotonically.

As the auxiliary data changes only in one dimension, the other dimension is completely irrelevant according to the learning metrics definition of relevance. Therefore the optimal clustering is one-dimensional, and it clusters the space into roughly evenly sized horizontal bars. On the left in Figure 6.1, we see that the DC has been able to find a correct solution.

The same data is also used for demonstrating how regularization affects the clustering result. Here the Bayesian regularization with the VQ model is used as an example. The

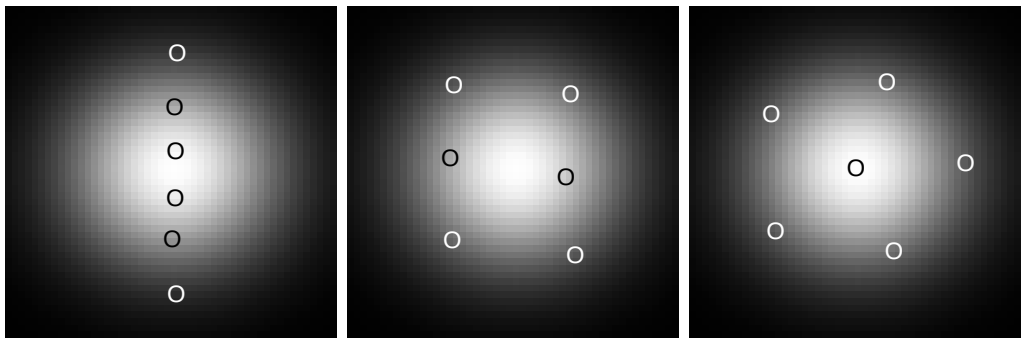

Figure 6.1: Discriminative clustering regularized with vector quantization makes a compromise (middle, $\lambda = 0.02$) between the plain discriminative clustering (left) and the Euclidean vector quantization (right). The unregularized discriminative clustering solution is an optimal solution to the problem, and the Euclidean VQ solution, and the regularized middle solution with $\lambda = 0.02$ shares properties of both extremes. Circles denote the cluster centroids $\mathbf{m}_j$, and gray shades express the density $p(\mathbf{x})$. White means high density and black means low.

parameter λ tunes the compromise between pure DC and pure Euclidean VQ.

In the middle on Figure 6.1 we see how the regularization changes the clustering from the optimal DC solution towards the Euclidean VQ solution, but still the actual partitioning remains somewhat similar to the unregularized case. The Euclidean VQ result on the right clusters $p(\mathbf{x})$ well, but loses even more information of $p(c|\mathbf{x})$.

This toy experiment demonstrates the usability of discriminative clustering, and illustrates that it finds the correct solution at least in one simple case. Notice that the regularization does not actually help here, but rather weakens the solution. It is assumed that with small real-world data sets in particular, the unregularized version does not always find the optimal solution, and that regularization should then improve the results.

6.2 Experiments with real-world data

The optimization algorithms and regularization methods are tested extensively on three data sets. These tests have two main purposes. Firstly, the goal is to study the optimization algorithms and regularization methods of DC. Based on these experiments, I hope to be able to give suggestions on how DC should be optimized.

The second goal is to study the performance of DC in general. For this purpose, the DC variants are compared to some standard clustering methods: Euclidean VQ, mixture of Gaussians, and MDA2, all presented in Chapter 3. The results are evaluated by the DC criteria, so DC should perform clearly better than traditional variants that are designed for somewhat different tasks.

Table 6.1: Properties of the data sets

Data set	Dimensions	Classes	Samples
TIMIT phoneme data	12	41	99983
Letter Recognition Data	16	26	20000
Forest CoverType data	10	7	100000

6.2.1 Data sets

The data sets used for testing were chosen to be suitable for the task of discriminative clustering. The feature vectors are continuous and real-valued, and a categorical variable to be used as auxiliary data is available for each data set. Thus these data sets could also be used for testing classification and other kinds of supervised methods.

The first data set is phoneme data from the DARPA TIMIT CD-ROM [32]. The primary features consist of 12 cepstral components of phonemes spoken by various speakers. The samples are classified into 41 groups, each presenting one phoneme, and these class labels are used as the auxiliary information. In the original data set, the number of different phonemes is slightly larger; some of them are here dropped because of too few samples available. The total number of samples becomes 99983.

The second set is about letter recognition, obtained from UCI Machine learning repository [2]. Primary feature vectors consist of 16 measures, all scaled into the same range. Each sample represents one of 26 letters, and the label is used as the auxiliary variable. The data set consists of 20000 samples, roughly equally distributed over classes.

In the Forest CoverType data, from the UCI Knowledge Discovery in Databases Archive [9], small sections of forests are described by 10 continuously valued features and classified into 7 cover type classes. As with the other two data sets, the class labels are used as the auxiliary variable. The additional 44 binary features present in the original data were simply dropped, and a set of 100000 samples were randomly picked from the original 581012 samples.

The data sets are summarized in Table 6.1.

6.2.2 Testing procedure

The goal is to find out how the performance of discriminative clustering compares to traditional clustering methods. Additionally, we want to reveal possible differences in discriminative clustering performance between different optimization algorithms and regularization methods. Performances are compared by the paired t-test (see [25] for a textbook account), the null-hypothesis being that there exists no difference.

It would be possible to use a single data set for multiple tests very efficiently by N-fold

cross-validation as described in Section 3.3. We get N independent test that each use all of the available data. While the test sets are independent, the training sets unfortunately are not. In fact, different training sets consist mostly of the same samples (to be precise, $\frac{N-2}{N-1}$ of samples are identical in any two training sets). Often this is ignored, as the data has to be used efficiently to provide enough independent tests for checking statistical importance.

Out situation, however, is fortunate in that the data sets used for empirical comparisons are large enough not to enforce the data to be used in the most efficient way. Therefore a test setup with more independent data sets was selected.

Each data set was divided into N equally sized sets, and two-fold cross-validation was performed for each set. This results in $2N$ tests, where each training set is independent of every other training set, and tests sets are also mutually exclusive. Each data sample is used once as a training sample and once as a test sample. For the two larger data sets, TIMIT and Forest CoverType, $N = 10$ resulting in 20 tests. For Letter recognition $N = 5$.

Another reason for selecting this setup instead of the traditional N-fold cross-validation was the need to test how the regularization methods perform on small data sets. Regularization is often most useful when the size of the training set is small, and the importance of regularization decreases with the number of available training samples. If regularization helps DC, it is most likely to happen with small training sets.

6.2.3 DC variants and benchmark methods

In the previous sections, one optimization algorithm for the infinite data version of discriminative clustering, and two optimization algorithms for the finite data version were presented. We also have three different regularization methods, suitable for using with both the finite data optimization algorithms.

Table 6.2 contains the full list of methods with shorthand notations that are used when presenting the results. In total the number of different versions of DC is nine, and in addition there are three benchmark methods. Notice that the classical Euclidean vector quantization (VQ in Table 6.2) is here optimized with conjugate gradient algorithm rather than the K-means algorithm. With this choice, the VQ solution is obtained at the limiting case of high regularization of DC-CG-VQ. The solutions differ only little from standard K-means optimization algorithm.

All methods are initialized randomly. A set of K random samples are selected from the training data, and the K cluster prototypes $\mathbf{m}_j$ are initialized to the locations of the selected samples.

Shorthand notation	Optimization algorithm	Regularization
sDC	stochastic online	none
DC-CG	conjugate gradient	none
DC-CG-VQ	conjugate gradient	vector quantization
DC-CG-MoG	conjugate gradient	mixture of Gaussians
DC-CG-Ent	conjugate gradient	entropy
DC-SA	simulated annealing	none
DC-SA-VQ	simulated annealing	vector quantization
DC-SA-MoG	simulated annealing	mixture of Gaussians
DC-SA-Ent	simulated annealing	entropy
VQ	conjugate gradient	not applicable
MoG	expectation maximization	not applicable
MDA2	expectation maximization	not applicable

Table 6.2: List of clustering methods that are tested for relative performance. The first column shows a short name for each method; these names are used in the results section. Nine methods in the top of the table are DC variants, and the bottom three methods are classical clustering methods used as reference methods.

6.2.4 Selection of learning parameters

All algorithms presented in this thesis have a few parameters that need to be selected before optimization. The one common to all of them is the number of clusters. As explained in Chapter 2, this could be chosen by various model selection criteria. The number of clusters is also sometimes determined by the problem setup. Here I chose to use five clusters with all data sets, mostly for computational reasons. While this certainly is not the optimal number of clusters for all data sets, it is a value that could be used in real data analysis. The number of clusters is the same for all methods, so that they can be compared fairly.

Another parameter required for all algorithms is the length of the iterative learning. It can be presented as a fixed number of iterations, or we could measure the convergence of the algorithms and stop when a suitable criterion is fulfilled. In this thesis, I fixed the number of iterations to a large enough value.

For the conjugate gradient algorithm, I used $30L_T$ iterations, where L_T is the dimensionality of the parameter space. The dimensionality is higher for the DC-CG-MoG than for the other DC-CG variants, because the K mixture weights have to be parameterized in addition to the cluster prototypes $\mathbf{m}_j$. This would lead to larger number of iterations for DC-CG-MoG, which could give it an unfair advantage. To compensate, I used the dimensionality of MoG-regularized DC also with the other methods optimized with CG.

With the simulated annealing algorithm and the stochastic online algorithm, I used $100000K$ iterations, which is also roughly proportional to the dimensionality of the problem L_T, as $L_T = KL$ (for the SA version, for infinite data variant we have also

the parameters for $\{\boldsymbol{\psi}\}$), where L is the dimensionality of the feature vectors. As we remember from Table 6.1, the dimensionality of feature vectors is quite similar for all data sets. Increasing the number of iterations would probably have improved the results, but the computation time would have been unproportionally high.

For the two classical comparison methods optimized with EM-algorithms, 100 steps of the algorithm was used. The results seemed well converged after that many iterations.

As explained in the previous two chapters, every algorithm presented in this thesis (with the exception of Euclidean VQ) has at least one parameter that is not optimized by the learning algorithm, but is chosen by validation. Each algorithm has a parameter σ which governs the width of Gaussians, the role of which varies between algorithms (smoothing in DC-CG-versions, jumping kernel in DC-SA-versions, and generative component in MoG and MDA2). In addition to this, all regularized variants have the parameter λ, governing the amount of regularization. Again, the way the parameter affects the amount of regularization differs with the regularization method.

The parameters were selected by traditional cross-validation with M folds inside each training data set. The parameter values that provided the best results averaged over the cross-validation runs were selected. The model was then re-trained with the whole training data to produce the final clustering. The value of M was five in Letter recognition, and three for the other two data sets.

I had to keep the number of candidate values, from which the best value was selected, quite small because of computational reasons. Preliminary runs were used to select a set of reasonable values for each parameter to be used in validation. The number of candidate values for σ and λ was roughly five with each DC variant. For the comparison methods I validated σ from a notably larger set of candidates to ensure fair comparison.

The regularization parameter λ for the variants optimized with SA was not validated. Instead, the value that provided the best result with the corresponding CG variant (same regularization type, same σ, etc.) was used with the SA variant. This is because of computational reasons; validating over the λ parameters would have been too tedious with simulated annealing. Some tests were, however, performed, and the performance did not seem to suffer significantly from this suboptimal selection.

Finally we have the prior parameters $\{n_i^0\}$ of the Dirichlet prior of the MAP version of DC. These parameters control the importance of the prior distribution of the distributional probabilities $\boldsymbol{\psi}_i$. I chose to use $n_i^0 = 1$ for all i, as proposed in [4]. This prior is non-informative in the sense of entropy, and as a conjugate prior has the same effect as adding one sample of each possible auxiliary variable value c to each cluster.

	Letter	TIMIT	CoverType
sDC	5127.7	13302	4706.0
DC-CG	5134.3	13323	4737.6
DC-CG-VQ	5128.3	13209	4773.5
DC-CG-MoG	**5075.6**	13202	4670.2
DC-CG-Ent	5157.5	13185	4738.2
DC-SA	5307.4	13602	4703.0
DC-SA-VQ	5115.4	13194	4569.9
DC-SA-MoG	5172.0	13219	4484.5
DC-SA-Ent	5105.9	**13172**	**4373.2**
VQ	6320.8	13537	5709.9
MDA2	5476.6	13388	5532.5
MoG	6333.6	13478	5697.7

Table 6.3: Average costs of discriminative clustering variants and comparison methods on all three data sets. The best result for each data is in boldface, and significantly worse results are underlined. Note that the unregularized CG-variant is almost significantly worse than the best result on Letter ($p = 0.013$) and TIMIT ($p = 0.014$).

6.2.5 Results

Comparison of algorithm variants

The results of the tests are presented in Table 6.3 using the negative of the logarithmic posterior of cluster centroids (4.11) as the cost function. The paired t-test is used to find out whether the differences between different variants are significant. All variants that are significantly worse than the best obtained result are underlined, and p-values below 0.01 are considered to be significant. Note that the number of tests is here so large that the probability of false significances is relatively high.

On all three data sets, a regularized version provides the best result. On Letter recognition data, the best method is DC optimized with conjugate gradient algorithm and regularized by mixture of Gaussians, that is, DC-CG-MoG. For the other two data sets, entropy-regularized DC optimized by simulated annealing, DC-SA-Ent, shows to be the best.

The differences between the optimization algorithms and regularization methods are, however, quite small. On two of the data sets, none of the regularized versions is significantly better than any other. Therefore, no definite ordering of methods can be seen, and according to the null hypothesis they are formally equally good.

On the other hand, the tests show that the regularized versions are usually better than unregularized, plain DC. On the Forest CoverType data, the difference is clear. On the other two data sets, the unregularized DC optimized with CG is not significantly worse than the best regularized result, but the differences are almost significant ($p = 0.013$ for

Letter recognition, and $p = 0.014$ for TIMIT phoneme data). The infinite data version is also worse than the regularized finite data variants, though the difference is insignificant ($p = 0.11$) on the Letter recognition data.

Rather surprisingly, the stochastic DC is here slightly better than DC-CG and DC-SA. However, the differences are insignificant with the exceptions of DC-SA being significantly worse than the other two unregularized variants on Letter recognition and TIMIT phoneme data.

While the relative ordering of DC variants is unclear, also clear results are seen in the Table 6.3: the best DC variant outperforms the classical reference methods on all three data sets. In fact, on two data sets all DC variants are significantly better than reference methods. On TIMIT phoneme data, on the other hand, only the best regularized DC variants are significantly better than MDA2, and DC-SA is even worse than MDA2 or MoG.

Computational cost of optimization algorithms

The differences in performance between the three optimization algorithms are rather small. Another important property of an algorithm is computational time. The most time-consuming part of the optimization algorithms is the computation of Euclidean distances between samples $\mathbf{x}$ and model vectors $\mathbf{m}_j$. In the case of the two MAP algorithms, NK such distances are needed for each iteration step because of N training samples, and each distance costs L units. Thus the main difference between the computational cost in the algorithms comes from the number of iterations. With CG, I used $30KL$ iterations, while the SA uses $100000K$ iterations. Even as the cost of CG has to be multiplied with a small constant, as we need to compute the distances a few times for line searches, the total number of distance computations is still clearly smaller for CG when the dimensionality L of the feature vectors is on the scale of the present experiments.

For the sDC, K distances are computed at each step, and $100000K$ steps are used. This results to cost $100000K^2L$ compared to $30NL^2K^2$ computations for the CG. With the values of L and N used in this thesis, these costs are roughly equal. However, the sDC needs to update also the $\boldsymbol{\psi}_j$ parameters, which adds a cost of $200000KC$, where C is the number of auxiliary values.

With the implementations and parameter values used here, the simulated annealing algorithm takes roughly ten times longer to optimize than the conjugate gradient algorithm. The stochastic online algorithm is slower than the conjugate gradient algorithm as well. Some tests were made so that the number of iterations for SA was lowered to bring the computation time roughly equal to the CG; the results were clearly worse. On the contrary, the CG can be speeded up by reducing the number of iterations, as the performance decreases less.

As the performance of different optimization algorithms is quite similar, the choice of algorithm should be based on computational time. Therefore the conjugate gradient

	Letter	TIMIT	Covertype
DC-CG	4961.9	12981	4578.2
DC-CG-VQ	4933.4	12905	**4565.5**
DC-CG-MoG	**4857.9**	**12866**	4615.7
DC-CG-Ent	4864.1	12942	4575.8
VQ	6194.9	13487	5651.3
MDA2	5206.4	13012	5393.6
MoG	5174.9	13515	5636.3

Table 6.4: Average costs of discriminative clustering variants and comparison methods on all three data sets. The best result for each data is written in boldface, and significantly worse results are underlined. The difference between DC-CG-MoG and DC-CG is almost significant ($p = 0.015$) on the Letter recognition data.

algorithm is chosen for all remaining tests. Note however, that the simulated annealing algorithm would probably give better results on some data sets.

On the Forest CoverType data, where the scales of different dimensions differ by two orders of magnitude, simulated annealing performs very well compared to conjugate gradient algorithm. On the other two data sets, the different dimensions of feature vectors are roughly on the same scale. There is a reason to expect that smoothing with spherical Gaussians does not work well with uneven dimensionalities, which could explain the superiority of simulated annealing on Forest CoverType data. Even though the jumping kernels are also spherical, the algorithm is able to decline bad jumps, especially with good regularization.

Tests with more clusters

As the original tests proved to be inadequate for determining the relative quality of regularization methods, I decided to do some additional tests. It is possible that the small number of clusters (five) diminishes the differences, so the tests were repeated with the number of clusters doubled to ten. This time only the versions optimized with conjugate gradient algorithm were included, together with the benchmark methods. All the other parameters were kept same, and the validation procedure was identical.

The average costs of discriminative clustering variants and reference methods in the case of 10 clusters are presented in Table 6.4. Differences between regularization methods and unregularized version are similar to the case of five clusters. Here DC-CG-MoG is best on two data sets, and DC-CG-VQ gives the best average cost on Forest CoverType data.

Again the difference between the regularized variants and the unregularized DC is quite small. The difference between the best regularized DC and DC-CG is significant on TIMIT data, and almost significant ($p = 0.015$) on the Letter recognition data. The performance of unregularized DC is surprisingly good on the Forest CoverType data.

This could also be because of the differently scaled variables.

The reference methods are again clearly inferior to the DC variants. All DC variants, with the exception of DC-CG and DC-CG-Ent on TIMIT data, are significantly better than any of the classical clustering algorithms.

The results of these two sets of experiments can be summarized as follows: DC outperforms classical clustering methods, regularization generally increases the performance of DC, and the regularization methods produce mutually similar results.

6.3 Exploring TIMIT data and properties of DC regularization

The plain cost function values, or significance of differences between methods may not tell everything about the quality of the solutions. In this section one of the data sets, the TIMIT phoneme data set, is studied more closely. At the same time, I try to further explore the properties of DC and the regularization methods.

6.3.1 Effects of regularization

In this section I briefly study the effects of parameter λ with the two Bayesian regularizations and CG optimization algorithm. The two parameters to be validated with these variants are the smoothing parameter σ and the regularization parameter λ. Selecting σ with validation is rather simple, and the cost as a function of σ, as illustrated in [27], seems usually to be rather smooth and unimodal.

Effects of λ have not been studied earlier, so some experiments are included here. First I study how variable the results of validation are, by running 10-fold cross-validation. The effects of data set size are also studied. Supposedly less regularization, meaning smaller λ, is needed with large data sets. To study this, the runs were repeated with four different sizes of training data. I used data sets of sizes 4000, 8000, 16000, and 32000. In each case, 9/10 of samples were used for training, and 1/10 for validation. The number of clusters in this experiment was five.

The average value and standard deviation (computed in the logarithmic scale) of λ are illustrated in Figure 6.2. We see that the variance of λ is rather large, and hence selecting the amount of regularization by validation is difficult; we need to use a wide range of candidate values to find the best value. The amount of data does not seem to affect the variability of λ. On the other hand, the average optimal λ decreases while the amount of data is increased, as expected. The behavior is quite similar on both Bayesian regularization types. This result, however, is not statistically significant, but merely illustrates the phenomenon.

As another example, I illustrate how the compromise between DC and the model used for

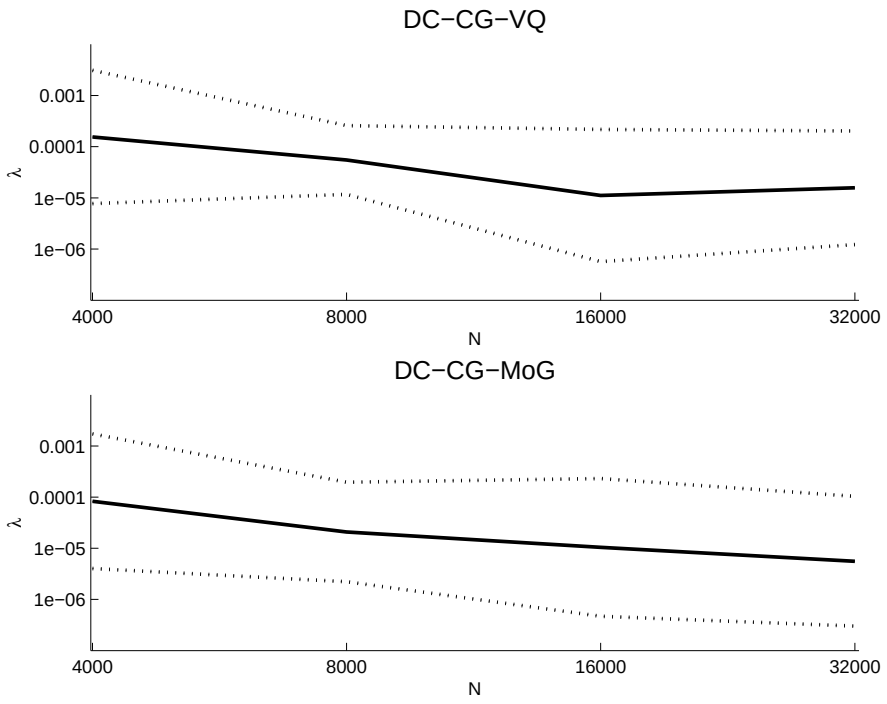

Figure 6.2: The optimal value of regularization parameter decreases with increasing size of the data set. On the other hand the variability in the estimate of the optimal parameter does not seem to decrease. Solid line: the average λ, dotted line: the standard deviations. Note the logarithmic axes.

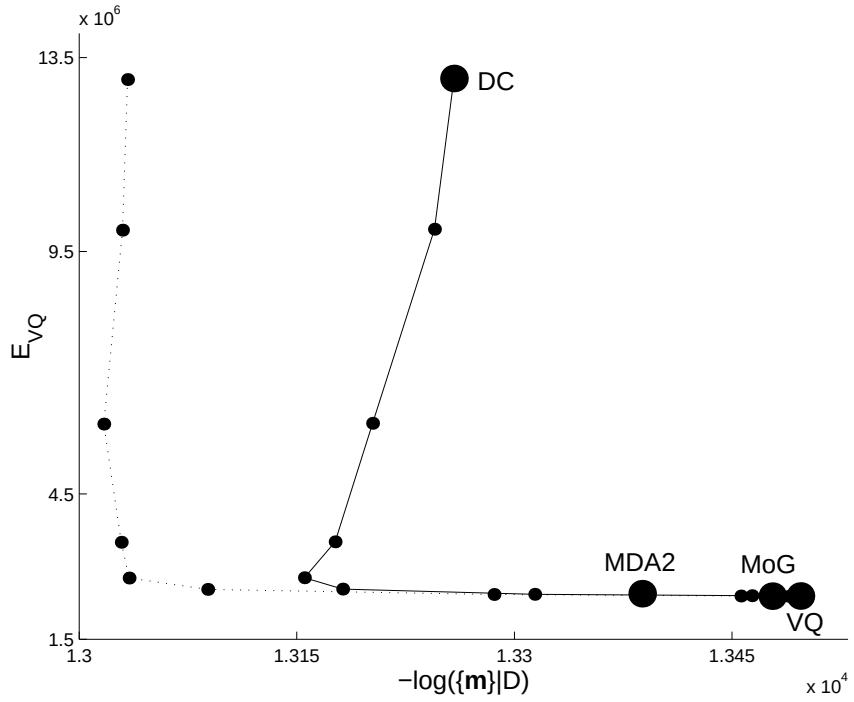

Figure 6.3: Illustration of the effects of λ on the cost function terms of DC-CG-VQ. The solid line presents the compromise between unregularized DC and Euclidean VQ on test set, and dashed line shows the same for training set. The large dots show the unregularized DC and the reference methods on the test set, while small dots depict the regularized DC solutions with different values of λ. Notice that the DC cost is better for slightly regularized DC than it is for the plain DC-CG, especially on the test set.

regularization changes with the value of λ. For this experiment, I chose to use the vector quantization model for regularization, as the compromise is most easily interpretable with it. This is the same compromise that was illustrated on toy data in Figure 6.1.

I validated the best σ for a range of λ-values, and computed both the discriminative clustering cost and the Euclidean VQ cost for each λ (using the σ that gave the best DC cost). The models with each λ are plotted (Figure 6.3) into two-dimensional space formed of the DC cost and the Euclidean VQ cost. The costs are computed as averages over 10 validation runs. For reference, the average results of the traditional clustering methods on the same setup are also included. Again, five clusters were used.

As we see, changing the λ moves the result from the pure DC result towards the pure Euclidean VQ. The form of the curve for the test set is here the most interesting part. The best value of the DC cost is not obtained without regularization, but with a small value of λ. The result is consistent with the previous experiments, where regularization in general improved the performance of DC. We also see that the VQ cost of the best DC results is clearly smaller than the VQ cost of unregularized DC, or to be precise, the best VQ-regularized DC solution models the primary data almost as well as the classical

models.

Note also the difference in the forms of the two curves. For training data, the unregularized DC is roughly as good as the best regularized DC. The increase in test set performance of the regularized DC is explained by the increase in its generalization capability. As we remember from Chapter 2, increasing the generalization ability is one of the motivations for regularization.

6.3.2 Effects of initialization

The results obtained so far have all been quite noisy. The average cost difference over cross-validation runs might be clear, yet insignificant because some unusually bad "outlier" runs. These outliers increase the variance of the results and thus also increase the p-value of the paired t-test. With highly variable results, more tests are needed to discover possible significance.

Two of the optimization algorithms presented for DC are stochastic algorithms, so high variability in the results is understandable and easily explained. The CG algorithm, however, is a deterministic algorithm, and always converges to a local minimum of the cost function.

It is well known that for non-quadratic problems, the CG is fairly sensitive to initialization. This suggests that the bad results with CG on some runs might be because of bad initialization leading to poor local minima. In all previous tests, the initialization has been done by selecting random samples from the training data as location prototypes $\mathbf{m}_j$ of clusters.

Here I study the initialization in some detail in order to verify whether the variability in results could be caused by it, and use the two best variants, DC-CG-MoG and DC-CG-Ent, as examples. I optimized DC a number of times with different random initializations to study the variability of results. All other parameters were kept constant, and were selected to be close to optimal on the basis of earlier tests. The number of clusters was 10.

In addition to the random initialization, I tried initializing by Euclidean vector quantization, which in turn was initialized randomly. The Euclidean VQ solution was computed for the same random initialization that was used with DC in that particular run, and it was used as an initial configuration for the cluster prototypes $\mathbf{m}_j$ of DC. This kind of initialization forces the cluster prototypes to be initially located at areas of high density $p(\mathbf{x})$, and it could thus prevent very poor solutions by choosing a local minima with robust estimates for the distributions.

The results of 40 runs with both initialization methods are depicted in Figure 6.4. We see that the variability is clearly greater with random initialization. Also the average cost is slightly larger. This strongly suggests that the variability in the previous tests could be lowered with suitable initialization methods.

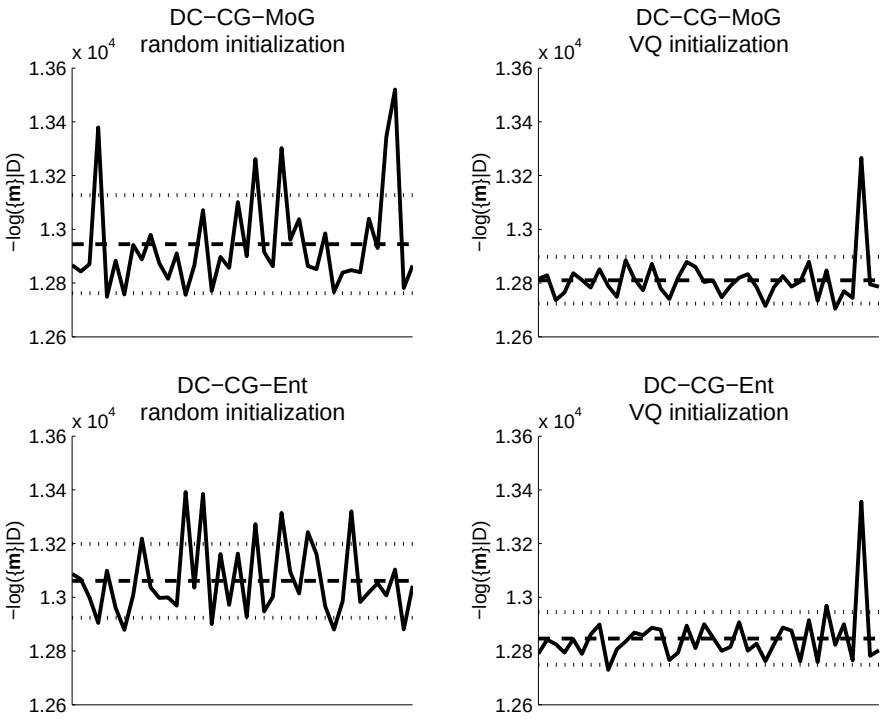

Figure 6.4: Random initialization (left) causes large variation to the results of DC. When DC is initialized with Euclidean VQ (right), both the average cost and the variability decrease. The effects are similar on the two tested methods, DC-CG-MoG and DC-CG-Ent. The x-axis is here used to present the 40 different runs. Solid line: runs, dashed line: average cost, dotted line: standard deviations.

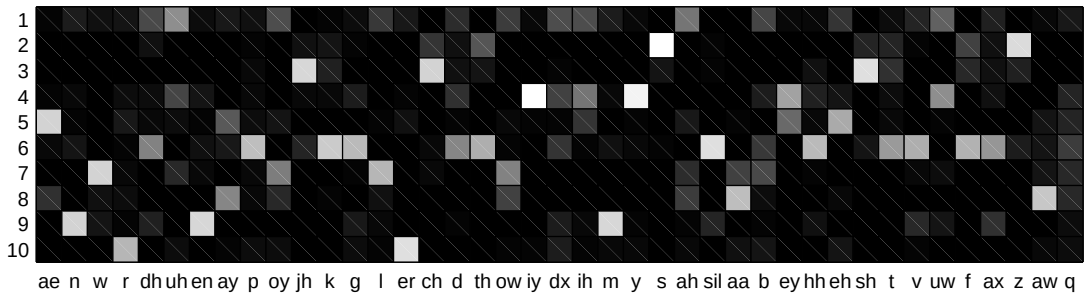

Figure 6.5: The distribution of test phonemes in the ten clusters. Each column is normalized, so the elements are empirical estimates of the conditional probabilities of certain phoneme being in certain cluster. White means high probability, and black is low. Note that for clarity of labels, the phoneme /ax-h/ is here denoted by /ax/, and /eng/ by /ng/.

The DC-CG-Ent seems to benefit from VQ initialization slightly more than DC-CG-MoG; the results obtained with VQ initialization are similar on both regularization methods, while the average cost of DC-CG-MoG is smaller with the random initialization. This is probably because the Bayesian regularization methods guide the cluster prototypes towards the model of $p(\mathbf{x})$, and thus can overcome bad initializations in some cases.

In the previous section we saw that validating the regularization parameter can be quite difficult. Here it was shown that the random initialization causes large variations to the cost function values. Both of these phenomena affect the comparisons in Tables 6.3 and 6.4 by increasing the variation of the results and thus reducing the significances.

6.3.3 Resulting clusters

Here I take a brief look at the contents of the clusters produced by discriminative clustering. I also try to interpret them to see whether the results really are reasonable.

For this purpose, DC-CG-MoG was trained with 10 clusters. 10000 samples were used in training set, and another 10000 samples were then clustered with the model. I chose to use parameter values $\sigma = 6.0$ and $\lambda = 0.0001$, which should provide a good result in terms of the cost function.

Distribution of test samples into clusters is presented in Figure 6.5. Each column has been normalized, so they present the distributions of samples into clusters given its phoneme label.

If we map phonemes into unique clusters by majority voting (Table 6.5), we can analyze

Cluster	Phonemes
1	uh dx ah
2	s z
3	jh ch sh
4	iy ih y ey uw
5	ae eh
6	dh p k g d th sil hh t v f ax-h q
7	w oy l ow b
8	ay aa aw
9	n m eng
10	r er

Table 6.5: Cluster memberships of phonemes, chosen by majority voting, that is, the phoneme belongs to a cluster with most samples of that particular phoneme. See text for detailed analysis of the result.

whether similar phonemes fall into the same clusters. The same short names are used for phonemes in both the figure and the table. Comprehensive explanation of these labels are given in the documentation of the data set [32], here I only address their meanings to the point it is necessary for this simple analysis.

In phonology, different kinds of higher level groupings can be given for phonemes. The most fundamental of such groupings is vowels and consonants. The consonants can be further divided into stops, affricates, fricatives, and nasals. In addition, some phonemes are classified to semivowels or glides, which are sort of intermediates between consonants and vowels.

Let us now study the cluster structure obtained with DC from this viewpoint. In cluster 3 we have /ch/ and /jh/, the only two affricates present in the data set. They are grouped together with /sh/, which clearly sounds similar. All the nasal phonemes (/n/, /m/, and /eng/) are clustered into cluster 9 with no other phonemes.

The cluster 10 contains only two phonemes, /er/ and /r/, which are so-called tremulants. In cluster 2, we have only two mutually similar phonemes, /s/ and /z/.

The cluster 6 has quite a number of phonemes. It contains nearly all the stops (/k/, /p/, /t/, /g/, /d/, and /q/) with most of the fricatives (/f/, /th/, /dh/, /v/), and yet two other phonemes (/ax-h/, /hh/). In addition, the silence belongs to this cluster. It is understandable, as the stops consist mostly of silence.

The remaining clusters consist mostly of vowels. The cluster 8 has three quite similar vowels (/ay/, /aa/, and /aw/) and no other phonemes, while cluster 5 has another two mutually similar vowels, /ae/ and /eh/ (both are front vowels). In cluster 4 we have one semivowel /y/, and all the other phonemes are vowels. The cluster 7 contains two vowels (/oy/ and /ow/) with two semivowels (/l/ and /w/), and even one stop, /b/. Finally, in cluster 1, we have two vowels (/uh/ and /ah/) together with one stop, /dx/.

Overall, the clusters seem quite natural. DC has been able to separate the vowels and semivowels from the consonants relatively well. In addition, the vowel groups are quite homogeneous. An expert of phonetics would be required to analyze the results in more detail.

If we imagine that the clustering had been done without any prior knowledge of the problem, the results are partially good, but partially only tentative. The nasals, tremulants and affricates become grouped and could probably be detected as natural clusters. The cluster 6, on the other hand, has pretty large number of different phonemes, and would definitely require further analysis, for example DC with more clusters.

Chapter 7

Related works

7.1 Other applications of learning metrics

As discussed in Chapter 4, DC does vector quantization in learning metrics, though only asymptotically. Therefore it is closely related to other works of learning metrics. Below I briefly summarize two other approaches of the principle.

7.1.1 Self-organizing map in learning metrics

As described in Chapter 2, the learning metrics principle can be applied by explicitly estimating the conditional distribution of auxiliary variable, $p(c|\mathbf{x})$, and by computing the distances with (2.8). This approach can be used with any unsupervised method that computes distances; here I review works [13, 23] on self-organizing maps [15].

The self-organizing map (SOM) consists of a set of units u_j. Each unit is presented by a model vector $\mathbf{m}_j$ in the primary data space. In addition, the units have a topology in a sense that the units are arranged on a lattice, usually of a form of a two-dimensional hexagonal grid. The model vectors $\mathbf{m}_j$ are adapted to represent $p(\mathbf{x})$. At the same time, units close to each other on the lattice are forced to have relatively similar model vectors $\mathbf{m}_j$. As a result, an ordered representation of $p(\mathbf{x})$ is obtained.

A simple online algorithm can be used for adaptation. At each step, one training sample $\mathbf{x}(t)$ is chosen randomly, the distance $d(\mathbf{x}(t), \mathbf{m}_j)$ to all model vectors $\mathbf{m}_j$ is computed, and the unit $w(t)$ with the closest model vector is selected as the best matching unit (BMU). After that, the model vectors of the BMU and its neighboring units are adapted a little towards the data sample. See for example [15] for details.

Often the Euclidean distance is used. The learning metrics distance would be given by (2.8), where $p(c|\mathbf{x})$ is replaced by an estimate. In principle, this approach requires that the conditional density $p(c|\mathbf{x})$ is to be estimated, and that the global distances, which

are path integrals of (2.8), have to be approximated.

Various density estimation methods based on Gaussian functions have been tested for the purpose of estimating $p(c|\mathbf{x})$ [23]. The results suggest that the conditional density should be estimated directly, and not by first estimating $p(c, \mathbf{x})$ and using Bayes' rule to obtain $p(c|\mathbf{x})$.

Different distance approximation methods have been studied in [23]. The simplest approach uses (2.8) directly for global distances. The results are sometimes good but in general (2.8) is inadequate for global distances.

A better approximation assumes that the shortest path is a straight Euclidean line between the sample $\mathbf{x}(t)$ and the model vector $\mathbf{m}_j$. The distance can then be approximated by dividing the line into small segments, and computing the local approximation at the beginning of each segment. A SOM in learning metrics with this distance approximation seems to preserve the distributions of the auxiliary variable better than the traditional SOM [14]. It also outperforms a supervised variant of SOM [15].

The approach taken with SOM could also be used for clustering. Either we could use the SOM in learning metrics as a clustering algorithm, as SOM is sometimes used, or we could apply learning metrics to another unsupervised clustering method by estimating the conditional density, and using the resulting metric with vector quantization, for example.

An obvious problem with this type of approach is the need of a density estimator. Optimization of the density estimator is in no way related to the final goal, be it clustering or something else, and currently there exists no justified criteria for selecting the best estimator. In the works so far, the estimator has always been selected by validation.

7.1.2 Relevant component analysis

The learning metrics principle has also been applied to component analysis [22]. In relevant component analysis, a linear projection $\mathbf{W}^T\mathbf{x}$ is sought to maximally preserve the relevant properties of data. The relevance is here again given by an auxiliary variable c.

The data $\mathbf{x}$ is projected to smaller-dimensional vectors $\mathbf{y} = \mathbf{W}^T\mathbf{x}$ by an orthogonal transformation matrix $\mathbf{W}$. The columns $\mathbf{w}_i$ of $\mathbf{W}$ are the basis vectors of the reduced-dimensional subspace of the original primary data space. The data can therefore be presented by components $\mathbf{w}_i^T\mathbf{x}$. Note that the projection is defined solely on the basis of the primary data.

The goal is to find a transformation that makes the subspace as informative as possible of the auxiliary data. Informativeness is here measured by predictive power of a generative probabilistic model of c given the projected value $\mathbf{W}^T\mathbf{x}$. The projection is optimized by

maximizing the likelihood

$$L = \sum_{(\mathbf{x},c)} \log \hat{p}(c|\mathbf{W}^T\mathbf{x}) \ , \tag{7.1}$$

where $\hat{p}$ is some density estimator. If the estimator is parametric, the parameters are optimized as well. In [22], a Parzen kernel estimator is used, and the projection matrix is estimated by maximizing the likelihood with a stochastic approximation algorithm.

Here the optimization of the estimator is connected to the primary goal of the method, and thus is not as arbitrary as in the previous section.

Like DC, Relevant component analysis has a connection to mutual information. Asymptotically, as the amount of data increases, RCA is equivalent to maximizing the mutual information between C and Y considered as random variables. The connection to the learning metrics is also asymptotic; RCA minimizes a certain reconstruction error in learning metrics (see [22] for details).

We can think of RCA as a modification of DC. The function that assigns samples to clusters is changed from minimum Euclidean distance to a linear projection operator. In addition, the estimation of within-cluster probabilities by constant parameters is replaced by a continuous parameterized or non-parametric estimate in the projection subspace.

7.2 Information Bottleneck

Distributional clustering refers to marginal clustering of co-occurrence data, that is, data that consists of occurrences of nominal variables. The concept and models for distributional clustering were introduced in [24]. The ideas have been developed further in many directions by several authors. Here we introduce the information bottleneck (IB) principle [33].

In the notation used in this thesis, the idea in IB is to build a representation V for a discrete random variable X that would be be maximally informative about another random variable C while using as little resources as possible. Finding such representation can be conceptualized as clustering, as the amount of resources used to present V is smaller than what is needed for presenting X.

More formally this means that the mutual information $I(C;V)$ is maximized while the mutual information $I(X;V)$ is minimized. The cost is formulated as $I(X;V) - \beta I(C;V)$, where β controls the importance of the two contradicting goals. By variational optimization, the solution

$$p(v_l|x) = \frac{p(v_l)\exp\left[-\beta d_{KL}(p(c|x)||p(c|v_l))\right]}{\sum_j p(v_j)\exp\left[-\beta d_{KL}(p(c|x)||p(c|v_j))\right]} \tag{7.2}$$

is obtained. Note that the solution is self-referential through $p(c|v)$, and thus does not lead to an algorithm directly. Various algorithms have been proposed for finding the solutions, for example, agglomerative methods [31] and a sequential method [30].

In terms of DC, we can think IB as clustering X into clusters V that are informative about C, which is the task of DC as well. The main differences are that IB is defined for discrete X instead of continuous as DC, and that the IB is non-generative. While both methods are originally introduced for distributions of data, the DC can be interpreted generatively (see Section 4.3), and can thus be used to handle finite data sets in a justified way.

Chapter 8

Conclusions

8.1 Evaluation

In this thesis, I have studied discriminative clustering and the optimization process required for learning the parameters of the model. Three optimization algorithms and three regularization methods aimed to ease optimization and to improve generalization capability were discussed. In addition, discriminative clustering has been compared with traditional methods applicable for the same task.

The experimental results support the previously known fact that discriminative clustering performs better than traditional unsupervised model-based clustering methods. DC also outperforms a joint distribution model called MDA2. This holds for both the regularized and unregularized versions of DC.

I have demonstrated that the conjugate gradient algorithm is a relatively quick and effective way of optimizing the discriminative clustering model. Sometimes better results can be obtained with simulated annealing, but the computational cost is roughly an order of magnitude higher.

The smoothing required for the conjugate gradient (or any other gradient-based optimization algorithm) could have caused some artifacts to the obtained clustering solution. The results here show that the changes to the problem are so small that they do not affect the optimization dramatically. This is the case at least with data sets where using spherical smoothing is reasonable; if the distribution of data is badly elongated into irrelevant directions, the results are worse. The conjugate gradient algorithm is not, however, perfect optimization algorithm for the problem, as it has problems with local minima and is sensitive to the initialization.

All three regularization methods presented in this thesis improve the performance of discriminative clustering on previously unseen test sets. The difference to the unregularized discriminative clustering is, however, sometimes quite small. Also the relative

performance of regularization methods is unclear; they seem to perform very similarly on most experiments.

The regularization methods each have some pros and cons. The vector quantization regularization has a nice interpretation as a compromise between Euclidean VQ and DC. The mixture of Gaussians lacks that property, but usually performs somewhat better. Another drawback of MoG regularization is the slightly higher number of parameters.

The entropy regularization has a nice and simple interpretation as well, which unfortunately holds only in the asymptotic case. The regularization is trivial to implement, and does not increase the computation time at all. Moreover, validating the regularization parameter is easier than with the Bayesian regularization methods.

Based on these experimental results, the following suggestions could be given. Discriminative clustering should be optimized with conjugate gradient algorithm, unless excessive amounts of computational resources are available or the data sets are small. In such exceptional cases, experimenting with simulated annealing could be advisable.

Applying one of the presented three regularization methods would probably be beneficial. The MoG and entropy regularization methods generally give the best results. The relative ordering of the two methods depends on the application. The best approach would obviously be to test both (or even all three) ways to regularize.

8.2 Future work

Many of the aspects studied in this thesis would benefit from additional research. Here I list the most important ones and give suggestions on the directions of future studies.

The optimization process still has some unclear points. The conjugate gradient algorithm performs well, but the obtained solutions depend highly on the initialization. The variability is rather high, and even a simple non-random initialization method helps to stabilize the results while giving equally good or even better average results.

This suggests that more work should be done on the initialization of the optimization algorithms. More clever initialization methods should at least help to find out the relative ordering of regularization methods with greater accuracy, because the random noise in evaluation is diminished. It is also possible that even the maximal performance could be increased.

The same problem could perhaps also be addressed with other kinds of approaches, for example by using other gradient-based optimization algorithms. Some stochastic elements in the optimization could help preventing poor local minima.

Selecting the amount of regularization is another partly open question. In this thesis the selection has been done by validation. Validation, however, gets complicated with the Bayesian regularization methods. More clever methods for the parameter selection could

be studied. In addition, it could be fruitful to experiment using Bayesian regularization together with the entropy regularization.

One possible direction for future studies is to examine suitable preprocessing methods for the data to be analyzed with DC. As cluster memberships are based on Euclidean distances, the scaling of the primary space affects the result. Experiments on the Forest CoverType data show that DC smoothed with spherical Gaussians might not work well when data dimensions have large variations. Whitening the data, or using other more clever preprocessing methods, could help.

The version of simulated annealing used in this thesis is decent, but clearly not optimal. For example, we could use more sophisticated jumping kernels to guarantee quicker theoretical convergence [21]. Re-annealing procedures could be studied to prevent poor results. The computation time is likely to be greater than that of CG even with these improvements, but the SA algorithm could be useful in some special cases.

In the future, the evaluation of DC should be extended. In this thesis, I have compared DC with simple traditional clustering methods. One problem in this procedure is that the same auxiliary variable is used during learning and evaluation, making the setup closely resemble classification. DC is, however, an exploratory data analysis tool, and to get a more realistic idea of its performance, we could use some auxiliary variable to indicate relevance during learning, and evaluate the performance of DC by another, independent relevance-indicating variable.

I was not able to do such indirect evaluation of performance in this thesis because of lack of suitable data sets. If there were a data set with two separate classifications, one could use one of them for training and the other for testing. If the two classifications are relevant with respect to each other, the results of DC should still be good.

A fixed number of clusters has been used in all experiments. As mentioned in Chapter 3, several methods could be used to automatically select the best order of a model. Many of them should be applicable to DC. Therefore studying the model selection in the context of DC would be worthwhile, as these methods could then be used to automatically select the number of clusters.

Appendix A

Gradient of maximum a posteriori DC

The cost function of MAP DC used in optimization is the logarithm of the smoothed posterior $p(\{\mathbf{m}\}|D^{(c)}, D^{(x)})$ (4.12). Here we denote for brevity $t_{ji} = n_{ji} + n_i^0$ and $T_j = \sum_i t_{ji}$. The gradient of the cost with respect to m_j is

$$\begin{aligned}\frac{\partial}{\partial \mathbf{m}_j} \log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) &= \sum_{il} \Psi(t_{li}) \sum_{c(\mathbf{x})=i} \frac{\partial}{\partial \mathbf{m}_j} y_l(\mathbf{x}) \\ &\quad - \sum_{\mathbf{x},l} \Psi(T_l) \frac{\partial}{\partial \mathbf{m}_j} y_l(\mathbf{x}) \\ &= \sum_{\mathbf{x},l} [\Psi(t_{l,c(\mathbf{x})}) - \Psi(T_l)] \frac{\partial}{\partial \mathbf{m}_j} y_l(\mathbf{x}) \; . \end{aligned} \tag{A.1}$$

For the normalized Gaussian membership functions we have

$$\frac{\partial}{\partial \mathbf{m}_j} y_l(\mathbf{x}) = \frac{1}{\sigma^2} (\mathbf{x} - \mathbf{m}_j)(\delta_{lj} - y_l(\mathbf{x})) y_j(\mathbf{x}) \; . \tag{A.2}$$

Substituting this to the gradient gives

$$\sigma^2 \frac{\partial}{\partial \mathbf{m}_j} \log p(\{\mathbf{m}\}|D^{(c)}, D^{(x)}) = \sum_{\mathbf{x},l} (\mathbf{x} - \mathbf{m}_j)(\delta_{lj} - y_l(\mathbf{x})) y_j(\mathbf{x}) [\Psi(t_{l,c(\mathbf{x})}) - \Psi(T_l)] \; . \tag{A.3}$$

The form (4.13) is obtained by applying the identity

$$\sum_l (\delta_{lj} - y_l) y_j L_l = \sum_l y_l y_j (L_j - L_l) \tag{A.4}$$

to (A.3).

Appendix B

Gradient of the mixture of Gaussians

Consider the mixture of Gaussians with covariance matrices of form $\Sigma_j = \sigma^2 I$. Denote the mixture components by $G(\mathbf{x}|m_j) = \frac{1}{(2\pi\sigma^2)^{L/2}} \exp(-\|\mathbf{x}-\mathbf{m}_j\|^2/\sigma^2)$. Then the mixture is

$$p(\mathbf{x}) = \sum_j^K \rho_j G(\mathbf{x}|\mathbf{m}_j) \; . \tag{B.1}$$

The sum of the component probabilities ρ_j has to be one. Hence they are reparameterized by "soft-max"

$$\rho_j = \frac{\exp(\beta_j)}{\sum_k^K \exp \beta_k} \tag{B.2}$$

to enforce $\sum_j \rho_j = 1$.

The gradient of the observed log-likelihood

$$E_{MoG} = \sum_i^N \log \left(\sum_j^K \rho_j G(\mathbf{x}_i|\mathbf{m}_j) \right) \tag{B.3}$$

with respect to $\mathbf{m}_l$ is

$$\begin{aligned} \frac{\partial E_{MoG}}{\partial \mathbf{m}_l} &= \sum_i^N \frac{\rho_l}{p(\mathbf{x}_i)} \frac{\partial}{\partial m_l} G(\mathbf{x}_i|\mathbf{m}_l) \\ &= \frac{1}{\sigma^2} \sum_i^N \frac{\rho_l G(\mathbf{x}|\mathbf{m}_l)}{p(\mathbf{x}_i)} \left(\mathbf{x}_i - \mathbf{m}_l\right) \\ &= \frac{1}{\sigma^2} \sum_i^N p(u_l|\mathbf{x}_i) \left(\mathbf{x} - \mathbf{m}_l\right) \; . \end{aligned} \tag{B.4}$$

Here $p(u_l|\mathbf{x}_i)$ is the posterior probability of component u_l having generated the data point $\mathbf{x}_i$.

With respect to the parameters β_l we have

$$\begin{aligned}\frac{\partial E_{MoG}}{\partial \beta_l} &= \sum_i^N \frac{1}{p(\mathbf{x}_i)} \frac{\partial}{\partial \beta_l} \left[\sum_j^K \rho_j G(\mathbf{x}_i|\mathbf{m}_j) \right] \\ &= \sum_i^N \sum_j^K \frac{G(\mathbf{x}_i|\mathbf{m}_j)}{p(\mathbf{x}_i)} \frac{\partial \rho_j}{\partial \beta_l} \\ &= \sum_i^N \left[\frac{\rho_l G(\mathbf{x}_i|\mathbf{m}_l)}{p(\mathbf{x}_i)} - \rho_l \sum_j^K \frac{\rho_j G(\mathbf{x}_i|\mathbf{m}_j)}{p(\mathbf{x}_i)} \right] = \sum_i^N \left[p(u_l|\mathbf{x}_i) - \rho_l \right] \ , \quad \text{(B.5)}\end{aligned}$$

where

$$\frac{\partial \rho_j}{\partial \beta_l} = \delta_{jl}\rho_l - \rho_j \rho_l \quad \text{(B.6)}$$

has been used.

If the latter gradient is set to zero, we get

$$\sum_i^N \left[p(u_l|\mathbf{x}_i) - \rho_l \right] = 0 \ \leftrightarrow \ \rho_l = \frac{1}{N} \sum_i^N p(u_l|\mathbf{x}_i) \ , \quad \text{(B.7)}$$

which constitutes the analytical solution (5.8) for the mixture weights used when optimizing MoG model with simulated annealing.

Appendix C

Connection of MAP DC to mutual information

The asymptotic connection between the MAP DC and mutual information is shown here. At the same time, it is shown that the entropy regularization in fact adds a small portion of the entropy of the distribution of samples to the original cost function.

Consider the entropy regularized cost

$$E_{Ent} = \sum_{ij} \log \Gamma(n_i^0 + n_{ji}) - (1+\lambda) \sum_j \log \Gamma(N^0 + N_j) \ , \tag{C.1}$$

where λ controls the amount of regularization. Apply the Stirling approximation $\log \Gamma(s+1) = s \log s - s + \mathcal{O}(\log s)$ to all Gamma functions in (C.1) to get

$$\begin{aligned} E_{Ent} &= \sum_{ij} (n_{ji} + n_i^0) \log(n_{ji} + n_i^0) \\ &\quad -(1+\lambda) \sum_j (N_j + N_j^0) \log(N_j + N^0) + \mathcal{O}(\log N) \ . \end{aligned} \tag{C.2}$$

Note that $N \geq N_j \geq n_{ji}$.

The prior parameters n_i^0 are small compared to the counts n_{ji}. We can therefore use the zeroth order Taylor expansion $\log(s+k) = \log s + \mathcal{O}(k/s)$. Applied, for example, to $(n_{ji} + n_i^0) \log(n_{ji} + n_i^0)$ gives $n_{ji} \log n_{ji} + n_i^0 \log n_{ji} + (n_{ji} + n_i^0)\mathcal{O}(n_i^0/n_{ji})$. Here the last term is negligible as n_i^0/n_{ji} is small, and the second term is $\mathcal{O}(\log N)$. By dropping $\mathcal{O}(n_i^0/n_{ji})$ for simplicity, we get

$$E_{Ent} = \sum_{ij} n_{ji} \log n_{ji} - (1+\lambda) \sum_j N_j \log N_j + \mathcal{O}(\log N) \ . \tag{C.3}$$

Division by N then gives

$$\frac{E_{Ent}}{N} = \sum_{ij} \frac{n_{ji}}{N} \log \frac{n_{ji}/N}{N_j/N} - \lambda \sum_j \frac{N_j}{N} \log \frac{N_j}{N} - \lambda \log N + \mathcal{O}\left(\frac{\log N}{N}\right) \ , \tag{C.4}$$

where n_{ji}/N approaches p_{ji}, the probability of auxiliary variable c_i in cluster j, and N_j/N approaches p_j as the number of data samples increases. Hence,

$$\frac{E_{Ent}}{N} \to \sum_{ij} p_{ji} \log \frac{p_{ji}}{p_j p_i} - \sum_i p_i \log 1/p_i + \lambda \sum_j p_j \log 1/p_j - \lambda \log N \ . \tag{C.5}$$

Here the first term is the mutual information between auxiliary variable C and cluster identity V. The second and the last terms do not depend on $\{\mathbf{m}\}$. The third term is λ times the entropy of p_j, thus giving a right to call the method entropy regularization.

Bibliography

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, New York, 1993.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.

[3] Gilles Celeux and Gérard Govaert. A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14:315–332, 1992.

[4] Greg F. Cooper and Edrward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[6] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[7] J. A. Hartigan and M. A. Wong. Algorithm AS136: A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[8] Trevor Hastie, Robert Tibshirani, and Andreas Buja. Flexible discriminant and mixture models. In J. Kay and D. Titterington, editors, *Neural Networks and Statistics*. Oxford University Press, 1995.

[9] S. Hettich and S. D. Bay. UCI KDD archive. `http://kdd.ics.uci.edu/`, 1999.

[10] Lester Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54, 1996.

[11] Samuel Kaski. Convergence of a stochastic semisupervised clustering algorithm. Technical Report A62, Helsinki University of Technology, Publications in Computer and Information Science, Espoo, Finland, 2000.

[12] Samuel Kaski and Janne Sinkkonen. Principle of learning metrics for data analysis. *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology, Special Issue on Data Mining and Biomedical Applications of Neural Networks*, forthcoming.

[13] Samuel Kaski, Janne Sinkkonen, and Jaakko Peltonen. Bankruptcy analysis with self-organizing maps in learning metrics. *IEEE Transactions on Neural Networks*, 12:936–947, 2001.

[14] Arto Klami, Jaakko Peltonen, and Samuel Kaski. Accurate self-organizing maps in learning metrics. In Pekka Ala-Siuru and Samuel Kaski, editors, *STeP 2002 — Intelligence, The Art of Natural and Artificial*, pages 41–49, Helsinki, 2002. Finnish Artificial Intelligence Society.

[15] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1995. (Third, extended edition 2001).

[16] Solomon Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[17] Harold J. Kushner and G. George Yin. *Stochastic Approximation Algorithms and Applications*. Springer, New York, 1997.

[18] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley, New York, 2000.

[19] Michael K. Murray and John W. Rice. *Differential Geometry and Statistics*. Chapman & Hall, London, 1993.

[20] Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

[21] Panos M. Pardalos and H. Edwin Romeijn, editors. *Handbook of Global Optimization Volume 2*, chapter 6. Kluwer Academic Publishers, 2002.

[22] Jaakko Peltonen and Samuel Kaski. Informative components of data. Submitted for publication.

[23] Jaakko Peltonen, Arto Klami, and Samuel Kaski. Learning more accurate metrics for self-organizing maps. In José R. Dorronsoro, editor, *Artificial Neural Networks - ICANN 2002*, pages 999–1004. Springer-Verlag, 2002.

[24] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 183–190. ACL, Columbus, OH, 1993.

[25] C. R. Rao. *Linear Statistical Inference and its Applications*. Wiley, New York, 1973. Second edition. Originally published 1965.

[26] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[27] Janne Sinkkonen and Samuel Kaski. Discriminative clustering. Submitted for publication.

[28] Janne Sinkkonen and Samuel Kaski. Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, 14:217–239, 2002.

[29] Janne Sinkkonen, Samuel Kaski, and Janne Nikkilä. Discriminative clustering: Optimal contingency tables by learning metrics. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Machine Learning: ECML 2002*, pages 418–430. Springer, 2002.

[30] Noam Slonim, Nir Friedman, and Naftali Tishby. Unsupervised document classification using sequential information maximization. In M. Beaulieu, R. Baeza-Yates, S.H. Myaeng, and K. Järvelin, editors, *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 2002. ACM Press.

[31] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 617–623. MIT Press, Cambridge, MA, 2000.

[32] CD-ROM prototype version of the DARPA TIMIT acoustic-phonetic speech database, 1998.

[33] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *37th Annual Allerton Conference on Communication, Control, and Computing*, Urbana, Illinois, 1999.