HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences

Mikael Kuusela

ALGORITHMS FOR VARIATIONAL LEARNING OF MIXTURE OF GAUSSIANS

Bachelor's thesis

Espoo 26.1.2009

Thesis instructor:

D.Sc.(Tech.) Tapani Raiko

TEKNILLINEN KORKEAKOULU

KANDIDAATINTYÖN TIIVISTELMÄ

Tekijä: Mikael Kuusela

Työn nimi: Algoritmeja Gaussin mikstuurin variaatio-oppimiseen

Päivämäärä: 26.1.2009 Kieli: Englanti Sivumäärä: 8+57

Tutkinto-ohjelma: Teknillinen fysiikka ja matematiikka

Vastuuopettaja: Prof. Harri Ehtamo

Ohjaaja: TkT Tapani Raiko

Koneoppimisessa halutaan monesti kuvata annettu data käyttäen parametrista mallia, jolloin mallin parametrit voidaan päätellä datasta käyttäen bayesilaista päättelyä. Valitettavasti päättelyä on harvoin mahdollista tehdä turvautumatta jonkinlaisiin approksimatiivisiin menetelmiin. Tässä työssä tarkastellaan erästä tällaista menetelmää nimeltä variaatio-Bayes-oppiminen.

Monien suosittujen mallien tapauksessa VB-oppiminen on mahdollista toteuttaa käyttäen VB EM -algoritmia (variational Bayesian expectation maximization). On kuitenkin olemassa monia mielenkiintoisia malleja, joihin VB EM -algoritmia ei voida soveltaa, jolloin on käytettävä yleisempiä epälineaarisia optimointimenetelmiä kuten gradienttimenetelmää. Viime aikoina on huomattu, että gradienttipohjaisten menetelmien tehokkuutta voidaan parantaa tulkitsemalla parametriavaruus kaareutuneeksi Riemannin monistoksi. Tämä ajatus johtaa luonnolliseen konjugaattigradienttialgoritmiin (natural conjugate gradient, NCG).

NCG-algoritmia on toistaiseksi onnistuneesti käytetty monimutkaisen epälineaarisen tila-avaruusmallin oppimiseen. Tähän malliin ei kuitenkaan voida soveltaa VB EM -algoritmia ja siten algoritmien vertailu ei ole ollut mahdollista. Tässä työssä tämä vertailu suoritetaan käyttäen yksinkertaisempaa Gaussin mikstuurina tunnettua mallia. Tätä varten työssä johdetaan ja implementoidaan NCG-algoritmi Gaussin mikstuuriin sovellettuna. Algoritmeja vertaillaan empiirisesti käyttäen sekä keinotekoista dataa että tosielämän kuvadataa. Lisäksi tutkitaan VB EM:n nopeutusta käyttäen pattern search -menetelmää.

Koetulosten perusteella voidaan todeta, että NCG on kilpailukykyinen VB EM:n kanssa, vaikkakin tuloksissa on suuria eroja eri datajoukkojen välillä. Erityisesti eräissä tapauksissa NCG näyttäisi löytävän parempia optimeita kuin muut algoritmit. Tutkituista algoritmeista pattern search -menetelmällä nopeutettu VB EM osoittautui kuitenkin yksinkertaisuutensa ja tasavahvan suorituskykynsä vuoksi parhaaksi vaihtoehdoksi Gaussin mikstuurin variaatio-oppimiseen.

Avainsanat: koneoppiminen, bayesilainen päättely, variaatio-Bayes-oppiminen, informaatiogeometria, luonnollinen konjugaattigradientti, Gaussin mikstuuri

Author: Mikael Kuusela

Title: Algorithms for Variational Learning of Mixture of Gaussians

Date: 26.1.2009 Language: English Number of pages: 8+57

Degree program: Engineering Physics and Mathematics

Supervisor: Prof. Harri Ehtamo

Instructor: D.Sc.(Tech.) Tapani Raiko

It is a typical problem in machine learning that one wants to represent a given set of data using some parametric model. The parameters of the model can be inferred from the data using Bayesian inference techniques. Unfortunately, exact Bayesian inference is seldom possible and some approximation scheme has to be used. One such method is a technique called variational Bayesian learning.

For many popular models, variational Bayesian learning can be conducted using the variational Bayesian expectation maximization (VB EM) algorithm. There are, however, many interesting models for which the VB EM algorithm is not available and one has to revert to using standard nonlinear optimization methods such as the gradient descent. It has been recently suggested that the performance of these gradient-based algorithms can be improved by interpreting the problem space to be a curved Riemannian manifold. This idea gives rise to the natural conjugate gradient (NCG) algorithm.

Thus far, the NCG algorithm has been successfully applied to learning the complicated nonlinear state-space model where the VB EM algorithm cannot be used and hence comparison of the algorithms has not been possible. This thesis does this comparison using the simpler mixture of Gaussians (MoG) model. The NCG algorithm for learning MoG is derived, implemented and empirically compared to the VB EM algorithm using both artificial data and real-world image data. A speed-up of VB EM using the pattern search method is also studied.

Based on the experiments, it is concluded that NCG is highly competitive against VB EM although there are datasets where one of the algorithms is clearly superior to the other. Especially with some datasets, NCG seems to excel at finding better optima than the other algorithms. However, given its simplicity and good performance across a wide range of datasets VB EM accelerated with pattern searches seems to be the best choice out of the compared algorithms for variational learning of the mixture of Gaussians model.

Keywords: machine learning, Bayesian inference, variational Bayesian learning, information geometry, natural conjugate gradient, mixture of Gaussians

Preface

This work was done in the Adaptive Informatics Research Centre at the Department of Information and Computer Science of Helsinki University of Technology.

I would like to thank professor Harri Ehtamo for supervising this thesis. I would also like to express my gratitude to the instructor of this thesis D.Sc. Tapani Raiko for guidance and encouragement. I would also like to thank docent Antti Honkela for enlightening discussions about technical matters.

Otaniemi, 26.1.2009

Mikael Kuusela

Contents

Symbols and Abbreviations

List of Symbols

$\lvert\mathbf{A}\rvert$	determinant of matrix $\mathbf{A}$
$(\mathbf{A})_{ij}$	the element in the ith row and the jth column of matrix $\mathbf{A}$
$\mathcal{C}$	cost function
$\mathrm{diag}(\mathbf{a})$	a diagonal matrix having the elements of vector $\mathbf{a}$ on its main diagonal
$\mathrm{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_n)$	a block diagonal matrix with matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$ as its diagonal blocks
D	dimensionality of the data
$\mathrm{Dir}(\boldsymbol{\pi}\lvert\boldsymbol{\alpha})$	Dirichlet distribution parametrized by $\boldsymbol{\alpha}$
$D_{KL}(q\lvert\lvert p)$	Kullback-Leibler divergence between distributions q and p
ε	termination criterion
$E_q\{\cdot\}$	expectation over distribution q
$\Gamma(\cdot)$	gamma function
$\mathbf{g}_k = \nabla\mathcal{C}(\boldsymbol{\xi}_k)$	gradient of the cost function on the kth iteration
$\tilde{\mathbf{g}}_k = \tilde{\nabla}\mathcal{C}(\boldsymbol{\xi}_k)$	natural gradient of the cost function on the kth iteration
$\mathbf{G} = (g_{ij})$	Riemannian metric tensor
$\mathbf{I}$	identity matrix
K	number of components in the mixture of Gaussians
λ	step size
$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$	precision matrix of the Gaussian distribution
$\mathcal{M}$	assumptions about the model
$\mathcal{N}(\mathbf{x}\lvert\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
N	number of data points
$\boldsymbol{\pi}$	the mixing coefficients of the mixture of Gaussians
$\psi(\cdot)$	digamma function
$p(x, y)$	joint probability of x and y
$p(x\lvert y)$	conditional probability of x given y
$p(\boldsymbol{\theta}\lvert\mathbf{X})$	exact posterior
$p(\mathbf{X}\lvert\boldsymbol{\theta})$	likelihood
$p(\boldsymbol{\theta})$	prior
$p(\mathbf{X})$	evidence
$\mathbf{p}_k$	search direction on the kth iteration in gradient-based methods
$q(\boldsymbol{\theta}\lvert\boldsymbol{\xi})$	approximate posterior parametrized by $\boldsymbol{\xi}$
R	distance between the Gaussian components of Dataset 1
$\mathbf{R} = (r_{nk})$	responsibility matrix
$\boldsymbol{\theta}$	model parameters
$\tau\mathbf{v}$	parallelly transported version of vector $\mathbf{v}$
$\mathrm{Tr}(\mathbf{A})$	trace of matrix $\mathbf{A}$
$\mathcal{W}(\boldsymbol{\Lambda}\lvert\mathbf{W}, \nu)$	Wishart distribution parametrized by $\mathbf{W}$ and ν
$\boldsymbol{\xi}$	parameters of the approximate posterior q
$\mathbf{X}$	observed data
$\mathbf{Z}$	latent variables

List of Abbreviations

CG	conjugate gradient
EM	expectation maximization
MAP	maximum a posteriori
MCMC	Markov chain Monte Carlo
ML	maximum likelihood
MoG	mixture of Gaussians
NCG	natural conjugate gradient
NSSM	nonlinear state-space model
VB	variational Bayesian (learning)
VB EM	variational Bayesian expectation maximization

1 Introduction

Bayesian inference is a stochastic framework for inferring the posterior distribution of some quantity given our prior beliefs about the quantity. However, in many real life problems, exact Bayesian inference is intractable and some approximation scheme has to be employed. Unfortunately, the traditional point estimate approximations such as maximum likelihood (ML) and maximum a posteriori (MAP) are inaccurate and suffer from overfitting problems. On the other hand, sampling-based Markov chain Monte Carlo (MCMC) methods are in theory able to achieve exact results but are too slow for many real-world problems.

Variational Bayesian (VB) learning [11, 3, 5] is a fairly recently introduced efficient method for conducting approximate Bayesian inference. The idea of VB learning is to approximate the true posterior distribution with another distribution q and introduce a cost function to measure the misfit of the distributions. Hence, the smaller the cost function, the better the approximate distribution q. Tractability is achieved by somehow restricting the functional form of q.

Many important models in machine learning belong to the so called conjugate-exponential family where VB inference can be conducted using a computationally efficient algorithm called the variational Bayesian expectation maximization (VB EM) algorithm. There are, however, a great deal of interesting models such as the nonlinear state-space model (NSSM) of [26] where the VB EM algorithm cannot be used. With such models, one can minimize the cost function directly using gradient-based optimization techniques such as the conjugate gradient (CG) method.

It was shown in [12] that the performance of gradient-based learning of NSSM can be substantially improved if the optimization is interpreted to take place in a curved Riemannian manifold instead of flat Euclidean space. In a Riemannian manifold, the standard gradient is replaced by the natural gradient and, when combined with the conjugate directions of the CG method, the resulting algorithm is called the natural conjugate gradient (NCG) algorithm.

In this thesis, the NCG algorithm is derived, implemented and experimentally studied in the case of the mixture of Gaussians (MoG) model which is a fairly complex model in the conjugate-exponential family. The aim is gain knowledge on the performance of NCG compared to VB EM. A speed-up of VB EM using a method called pattern searches is also studied.

The structure of this thesis is as follows. An introduction to VB inference as well as the VB EM algorithm and its acceleration using pattern searches is given in Section 2. Many central concepts used throughout this thesis such as the conjugate-exponential family are also introduced in that section. Section 3 focuses on the gradient-based methods and gives a brief account of the necessary concepts of information geometry required for the understanding of the NCG algorithm. The various algorithms applied to the MoG model are discussed in Section 4 which also includes a ragbag of important implementation details. The algorithms are experimentally compared to each other using both artificial data and real-world image data in Sec-

tion 5. The results are analyzed in Section 6 after which conclusions are given in Section 7. There are also three appendices. Appendix A gives a summary of the most important results regarding the Dirichlet, Gaussian and Wishart distributions used in this thesis while Appendix B includes derivations of the necessary equations for the implementation of the gradient-based algorithms. Appendix C summarizes this thesis in Finnish.

2 Variational Bayesian Inference

This chapter gives an introduction to Bayesian inference with emphasis on variational Bayesian learning. Two important rules of probability theory, namely the product rule and the sum rule, are presented in Section 2.1. The Bayes' rule, the cornerstone of Bayesian probability theory, is discussed in Sections 2.2 and 2.3. Variational Bayesian learning, an approximate Bayesian inference technique, is introduced in Section 2.4 while Section 2.5 discusses the concept of the conjugate-exponential family. The VB EM algorithm and its speed-up with pattern searches are discussed in Sections 2.6 and 2.7.

2.1 The Basic Rules of Probability

Traditionally, the probability of an event is interpreted to describe the frequency of outcomes favorable to the event in a random experiment. This viewpoint is called the frequentist approach to probability theory. More recently, however, another interpretation called the Bayesian approach has gained ground. In that approach, the probability of an event is regarded as ones subjective degree of belief that the event will take place. This approach was axiomatized by Cox in the 1940s into what is now know as the Cox axioms [7].

Two fundamental rules of probability theory can derived from the Cox axioms, namely the product rule

$$p(x, y) = p(x|y)p(y) \tag{1}$$

and the sum rule

$$p(x) = \sum_Y p(x, y) = \sum_Y p(x|y)p(x) \tag{2}$$

which is also know as the marginalization principle. Here $p(x)$ denotes the probability that random variable X has some value $X = x$, $p(x, y)$ the joint probability $X = x$ *and* $Y = y$ and $p(x|y)$ the probability that $X = x$ *given* $Y = y$. In Equation (2) the sum is taken over all possible values of random variable Y and is replaced by integration in case of continuous random variables.

2.2 The Bayes' Rule

Due to the symmetry of joint probability $p(x, y) = p(y, x)$, one can easily see that the following equation known as the Bayes' rule holds:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{3}$$

Here $p(x|y)$ is called the *posterior* as it gives the probability of $X = x$ when we have observed that $Y = y$. $p(y|x)$ and $p(x)$ are called the *likelihood* and the *prior* respectively while $p(y)$, which is called the *evidence*, can be viewed as a scaling factor

which makes sure that the posterior probabilities will sum to 1 over all possible values of X. The Bayes' rule also holds for probability distributions of continuous random variables.

2.3 The Bayes' Rule in Machine Learning

In the field of machine learning, it is often needed to infer the parameters $\boldsymbol{\theta}$ of some model $\mathcal{M}$ given the data $\mathbf{X}$. The Bayes' rule can be used for this inference task in the form

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{M}) = \frac{p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})}{p(\mathbf{X}|\mathcal{M})} \tag{4}$$

which gives us the posterior probability distribution of model parameters $\boldsymbol{\theta}$. The dependence on the selected model $\mathcal{M}$ is explicitly stated here to emphasize that the general form of the model used to represent the data has to be selected *a priori*. This dependence will, however, be omitted for brevity for the rest if this thesis.

The prior $p(\boldsymbol{\theta})$ in Equation (4) can be interpreted as our knowledge of the model parameters *before* the data $\mathbf{X}$ is observed while the posterior $p(\boldsymbol{\theta}|\mathbf{X})$ gives us the parameter distribution *after* the data is observed. Thus, the observation of the data can be seen as changing our prior beliefs about the parameters. The prior $p(\boldsymbol{\theta})$ is always subjective which is typical of the Bayesian approach but also its most controversial feature as the selection of the prior can greatly influence the outcome of the inference task.

2.4 Variational Bayesian Learning

Using the sum rule (2), the evidence in (4) can be evaluated to be

$$p(\mathbf{X}) = \int_{\boldsymbol{\theta}} p(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{5}$$

The central issue in Bayesian inference is that, apart the simplest models, this integral is intractable and while it can, in principle, be calculated numerically using MCMC methods, they are computationally too costly for efficient learning algorithms. It follows that for example the evaluation of the predictive distribution $p(\mathbf{x}|\mathbf{X})$, that is the distribution of a new observation $\mathbf{x}$ given the observed data $\mathbf{X}$, becomes intractable as its evaluation requires integration over the posterior distribution.

Variational methods attempt to overcome the intractable integral in (5) by approximating the true posterior distribution $p(\boldsymbol{\theta}|\mathbf{X})$ by another distribution $q(\boldsymbol{\theta})$. While there are various ways to achieve this, this thesis focuses on one particular variational Bayesian learning method sometimes called *ensemble learning* which has its roots in statistical physics, more precisely in the variational free energy minimization of Feynman and Bogoliubov [8].

The idea is to use a concept know as Kullback-Leibler divergence to measure the misfit between the true posterior distribution $p(\boldsymbol{\theta}|\mathbf{X})$ and the approximate distribution $q(\boldsymbol{\theta})$. The Kullback-Leibler divergence between $q(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\mathbf{X})$ is defined as

$$D_{KL}(q||p) = \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathbf{X})} d\boldsymbol{\theta} = E_q \left\{ \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathbf{X})} \right\} \tag{6}$$

where $E_q \{\cdot\}$ denotes the expectation over distribution q.

Due to a principle known as the Gibbs' inequality it holds that [14]

$$D_{KL}(q||p) \geq 0 \tag{7}$$

with equality if and only if $q(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|\mathbf{X})$. Thus, the minimization of $D_{KL}(q||p)$ is equivalent to the optimization of the quality of the approximate posterior $q(\boldsymbol{\theta})$ and $D_{KL}(q||p)$ can be used as a cost function to be minimized during learning.

As the true posterior $p(\boldsymbol{\theta}|\mathbf{X})$ in (6) is unknown, we will subtract the log-evidence, which is a constant, from this to obtain the true cost function $\mathcal{C}$ used in the learning process

$$\begin{aligned} \mathcal{C} &= D_{KL}(q||p) - \ln p(\mathbf{X}) \\ &= \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathbf{X})} d\boldsymbol{\theta} - \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln p(\mathbf{X}) d\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}, \mathbf{X})} d\boldsymbol{\theta} \\ &= E_q \left\{ \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta}, \mathbf{X})} \right\}. \end{aligned} \tag{8}$$

To make the integral in (8) tractable one has to somehow restrict the form of the distribution $q(\boldsymbol{\theta})$. One way to accomplish this is to select the functional form of the distribution $q(\boldsymbol{\theta})$ governed by some parameters $\boldsymbol{\xi}$ which we will denote by $q(\boldsymbol{\theta}|\boldsymbol{\xi})$. A popular choice for this fixed form solution is a Gaussian distribution with a limited covariance matrix.

Another popular way of restricting $q(\boldsymbol{\theta})$ again draws from statistical physics, namely from an approach to variational free energy minimization called *mean field theory* [19]. In this approach, the parameters $\boldsymbol{\theta}$ are divided into disjoint groups $\boldsymbol{\theta}_i, i = 1 \ldots M$ which are assumed to be posteriorly independent, that is

$$q(\boldsymbol{\theta}) = \prod_{i=1}^{M} q_i(\boldsymbol{\theta}_i). \tag{9}$$

It is shown for example in [5] that given this factorization the optimal solution for the jth distribution $q_j(\boldsymbol{\theta}_j)$ is

$$q_j(\boldsymbol{\theta}_j) = A \exp(E_{q_i, i \neq j} \{\ln p(\boldsymbol{\theta}, \mathbf{X})\}) \tag{10}$$

where A is a normalizing constant given by

$$A = \frac{1}{\int_{\boldsymbol{\theta}_j} \exp(E_{q_i, i \neq j} \{\ln p(\boldsymbol{\theta}, \mathbf{X})\}) d\boldsymbol{\theta}_j}. \tag{11}$$

Since the expectation in (10) is taken over distributions $q_i, i \neq j$, this results in an iterative algorithm where distributions $q_i, i = 1 \ldots M$ are first set to some initial values and index j in (10) is then cycled through all the values $j = 1 \ldots M$ until convergence is achieved which is determined by evaluating the cost function (8) on each iteration. It should be emphasized that in this approach the only assumption made is the factorization in Equation (9) while the functional forms of the distributions q_i are determined automatically by Equation (10).

VB learning has recently become popular in inference tasks due to its capability of automatically selecting the complexity of the model, thus mostly avoiding overfitting or underfitting the data while still being computationally efficient enough to be able to solve real-world problems. If applied to, for example, the task of fitting a polynomial curve to some measurement data, the VB method is capable of avoiding overfitting even if given a high-order polynomial to fit. This is something that competing methods such as ML, MAP or least squares fail to achieve.

2.5 Conjugate Priors and the Exponential Family

The prior $p(\boldsymbol{\theta})$ is said to be conjugate to the likelihood $p(\mathbf{X}|\boldsymbol{\theta})$ if it is selected so that the posterior $p(\boldsymbol{\theta}|\mathbf{X})$ has the same functional form as the prior. Conjugate priors are used extensively in Bayesian inference as they greatly simplify the required calculations.

Many of the most commonly used probability distributions belong to a class of distributions called the *exponential family* [4, 5] which are of the form

$$p(\mathbf{x}|\boldsymbol{\theta}) = h(\mathbf{x}) g(\boldsymbol{\theta}) \exp(\boldsymbol{\theta}^T \mathbf{u}(\mathbf{x})) \tag{12}$$

where $\mathbf{x}$ is a random variable, $\boldsymbol{\theta}$ is called the natural parameters of the distribution, $h(\mathbf{x})$ and $\mathbf{u}(\mathbf{x})$ are some functions of $\mathbf{x}$ and $g(\boldsymbol{\theta})$ is a normalizing constant.

The exponential family is of great importance, not only because of its ubiquity, but also because all distributions in the exponential family have conjugate priors [10]. It can be shown, for example, that the Gaussian distribution belongs to the exponential family and that its conjugate prior is another Gaussian distribution when we are inferring the mean and the Wishart distribution when we are inferring the precision matrix [5] (for details on the multivariate Gaussian distribution see Appendix A). When conjugate priors are used with exponential family distributions, the model is said to be in the *conjugate-exponential family*.

2.6 The VB EM Algorithm

The concept of a *latent variable* plays an important role in many models used in machine learning. By definition, latent variables are quantities related to each observed data point which can not be observed directly. Thus, the number of latent variables is proportional to the number of observations. Latent variables are also sometimes called *hidden* or *unobserved* variables.

If the used model $\mathcal{M}$ incorporates latent variables they will have to be inferred from the data along with the model parameters. Thus, we can substitute $\boldsymbol{\theta} \hookrightarrow (\boldsymbol{\theta}, \mathbf{Z})$ in the previous discussion where $\mathbf{Z}$ is used to denote the latent variables of the model $\mathcal{M}$.

By assuming that the approximate posterior $q(\boldsymbol{\theta}, \mathbf{Z})$ will factorize between the parameters $\boldsymbol{\theta}$ and the latent variables $\mathbf{Z}$, that is

$$q(\boldsymbol{\theta}, \mathbf{Z}) = q(\boldsymbol{\theta})q(\mathbf{Z}), \tag{13}$$

one can use Equation (10) to cyclically first update the distribution over the latent variables followed by the distribution over the model parameters until convergence is achieved. This algorithm is called the variational Bayesian expectation maximization (VB EM) algorithm as it bears a close similarity to the traditional expectation maximization (EM) algorithm used in ML and MAP learning. Following the naming conventions of the EM algorithm, the update of the latent variable distribution $q(\mathbf{Z})$ is called the *E-step* and the update of the parameter distribution $q(\boldsymbol{\theta})$ is called the *M-step*.

The VB EM algorithm is the standard way of conducting variational Bayesian inference for models $\mathcal{M}$ in the conjugate-exponential family. Its computational efficiency is comparable to the EM algorithm while it does not suffer from the same overfitting and singularity problems as ML and MAP methods tend to do.

2.7 Acceleration of VB EM with Pattern Searches

It was proposed in [13] that cyclic parameter update algorithms could be accelerated using a technique called pattern searches which will now be described for the case of the VB EM algorithm. Let $q(\mathbf{Z}|\boldsymbol{\xi}_\mathbf{Z})$ be the approximate latent variable distribution parametrized by some parameters $\boldsymbol{\xi}_\mathbf{Z}$ and $q(\boldsymbol{\theta}|\boldsymbol{\xi}_{\boldsymbol{\theta}})$ the approximate parameter distribution parametrized by $\boldsymbol{\xi}_{\boldsymbol{\theta}}$. If we further denote $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_\mathbf{Z} \\ \boldsymbol{\xi}_{\boldsymbol{\theta}} \end{bmatrix}$, we can now consider the cost function $\mathcal{C}$ in (8) to be a function of $\boldsymbol{\xi}$, that is $\mathcal{C} = \mathcal{C}(\boldsymbol{\xi})$. The algorithm is as follows:

1. Set $\boldsymbol{\xi}_\mathbf{Z}$ and $\boldsymbol{\xi}_{\boldsymbol{\theta}}$ to some initial values
2. Perform the E-step, let the new parameters be $\boldsymbol{\xi}_\mathbf{Z}{}'$
3. Perform the M-step, let the new parameters be $\boldsymbol{\xi}_{\boldsymbol{\theta}}{}'$

4. Calculate search direction $\Delta\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\xi}_{\mathbf{Z}}{}' - \boldsymbol{\xi}_{\mathbf{Z}} \\ \boldsymbol{\xi}_{\boldsymbol{\theta}}{}' - \boldsymbol{\xi}_{\boldsymbol{\theta}} \end{bmatrix}$
5. Perform a line search in the search direction $\Delta\boldsymbol{\xi}$ to obtain the step size λ: $\lambda \leftarrow \underset{\lambda}{\operatorname{argmin}}\, \mathcal{C}(\boldsymbol{\xi} + \lambda\Delta\boldsymbol{\xi})$
6. Update parameters: $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} + \lambda\Delta\boldsymbol{\xi}$
7. Repeat from 2 until convergence

It was further found out in [13] that in practice the algorithm works much better if the parameter updates are let to stabilize between line searches by performing multiple cyclic parameter updates in a row. This is also beneficial in terms of time complexity of the algorithm as performing a line search can be computationally expensive. The pattern searches approach is illustrated in Figure 1.

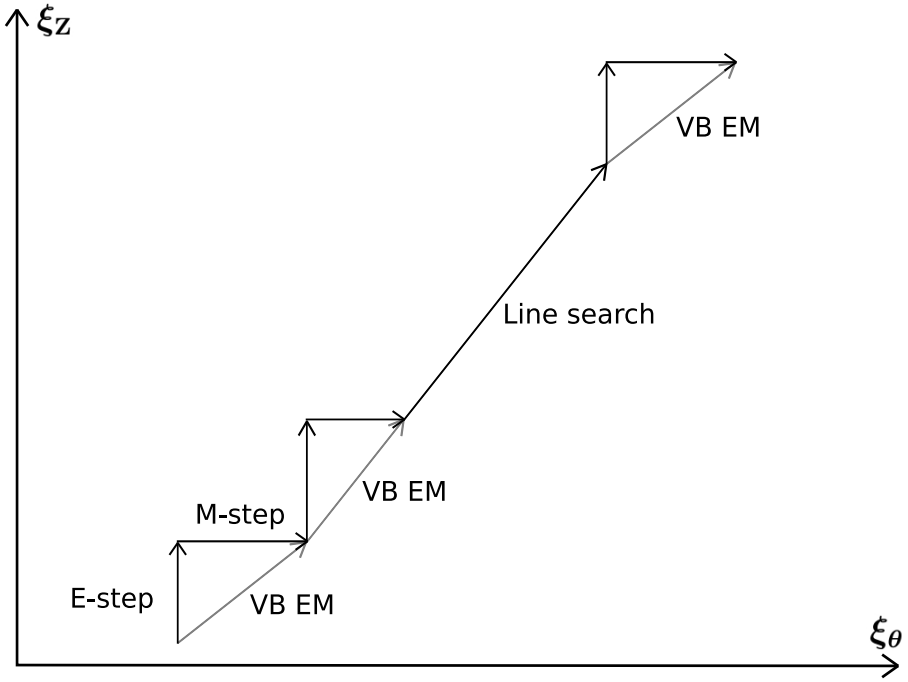


Figure 1: Illustration of pattern searches applied to VB EM updates. The horizontal axis represents the value of the parameters $\boldsymbol{\xi}_{\boldsymbol{\theta}}$ of the approximate distribution over model parameters while the vertical axis represents the value of the parameters $\boldsymbol{\xi}_{\mathbf{Z}}$ of the approximate distribution over latent variables. During the VB EM E-step $\boldsymbol{\xi}_{\mathbf{Z}}$ is updated and during the M-step $\boldsymbol{\xi}_{\boldsymbol{\theta}}$ is updated. The figure illustrates a situation in which we first perform two VB EM iterations and then combine the results of the E-step and M-step of the last VB EM iteration to obtain a search direction. We then perform a line search in the search direction followed by another set of VB EM iterations.

3 Gradient-based Learning Algorithms

This chapter focuses on minimizing the cost function with gradient-based optimization techniques. The widely used gradient descent and conjugate gradient methods are discussed in Sections 3.1 and 3.2. Section 3.3 begins with a brief introduction to information geometry followed by discussion of algorithms based on the natural gradient including the NCG algorithm.

3.1 Gradient Descent

While the VB EM algorithm provides a straightforward way of learning models in the conjugate-exponential family, there are more complicated models for which the VB EM algorithm is not available such as the nonlinear state-space model (NSSM) of [26]. In this case, one has to apply the fixed form learning approach described in Section 2.4 and thus select some approximate distribution $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ governed by parameters $\boldsymbol{\xi}$. Consequently, the cost function $\mathcal{C}$ becomes a function of the parameters $\boldsymbol{\xi}$, $\mathcal{C} = \mathcal{C}(\boldsymbol{\xi})$. As a results, one can apply standard nonlinear optimization techniques to find the minimum of $\mathcal{C}$.

The most elementary nonlinear optimization technique is the gradient descent. In that method, one first evaluates the negative gradient of the cost function

$$\mathbf{p}_k = -\nabla\mathcal{C}(\boldsymbol{\xi}_k), \tag{14}$$

then performs a line search in the direction of $\mathbf{p}_k$ to obtain a step size λ and finally updates the parameters using this step size

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + \lambda\mathbf{p}_k. \tag{15}$$

Instead of line search, the step size λ can also be set to some sufficiently small constant or adjusted adaptively during the learning process.

3.2 Conjugate Gradient Methods

The gradient descent method tends to use approximately the same search directions multiple times during the iteration process which slows down the convergence as it would be beneficial to make an optimal step in a particular direction the first time without having to correct later. This problem can be solved by a method called the conjugate gradient (CG).

In the CG method, the search direction is set to the negative of the gradient on the first iteration just like in the gradient descent but on subsequent iterations Equation (14) is replaced with

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k\mathbf{p}_{k-1} \tag{16}$$

where $\mathbf{g}_k = \nabla\mathcal{C}(\boldsymbol{\xi}_k)$, $\mathbf{p}_{k-1}$ is the previous search direction and β_k can be calculated either using the Fletcher-Reeves formula [9]

$$\beta_k = \frac{\|\mathbf{g}_k\|^2}{\|\mathbf{g}_{k-1}\|^2} = \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}} \tag{17}$$

or the Polak-Ribiére formula [20]

$$\beta_k = \frac{\langle(\mathbf{g}_k - \mathbf{g}_{k-1}), \mathbf{g}_k\rangle}{\|\mathbf{g}_{k-1}\|^2} = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1})^T\mathbf{g}_k}{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}}. \tag{18}$$

The CG algorithm can be justified by looking at a quadratic cost function

$$\mathcal{C}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c. \tag{19}$$

It can be shown that in this case the search directions $\{\mathbf{p}_k\}$ are conjugate with respect to $\mathbf{A}$ which means that

$$\mathbf{p}_i^T\mathbf{A}\mathbf{p}_j = 0, i \neq j. \tag{20}$$

From this, it follows that for a quadratic cost function (19) the CG method converges in at most n steps where n is the number of variables in $\mathbf{x}$. [9]

The CG method can nevertheless also be applied to a general cost function $\mathcal{C}$ where the convergence in n steps can no longer be guaranteed. The closer $\mathcal{C}$ is to being quadratic, the better the CG method will perform. The Fletcher-Reeves and the Polak-Ribiére formulas are equivalent when $\mathcal{C}$ is quadratic but for a general $\mathcal{C}$ they result in two distinctive algorithms [21]. There are also various other ways of selecting the value of β_k and making an optimal selection remains a subject of ongoing research. It is generally thought, however, that the Polak-Ribiére formula (18) is the best alternative for selecting β_k [18].

There are a few details one has to take care of when implementing a nonlinear conjugate gradient algorithm. First of all, when using the Polak-Ribiére formula, the convergence of the algorithm can only be guaranteed if β_k is restricted to be positive, that is

$$\beta_k \leftarrow \max(\beta_k, 0). \tag{21}$$

The performance of the algorithm can be further improved if the search direction $\mathbf{p}_k$ is reset to the negative of the gradient every n iterations. This is based on the fact that the CG algorithm is capable of generating only n conjugate search directions in a row. Restarting is especially important if the Fletcher-Reeves formula is used. It should be noted that the condition (21) can also be considered as a restart when $\beta_k < 0$. [23]

3.3 Natural Gradient Methods

3.3.1 Riemannian Manifolds

Manifolds are abstract mathematical spaces which locally resemble the Euclidean space but have a more complex global structure. In a manifold S, a vector $\mathbf{v}$ has to be

defined in a tangent space T_p which is a vector space consisting of the tangent vectors of all the smooth curves passing through point $p \in S$. One particularly important type of manifolds is the Riemannian manifold which is a manifold equipped with an inner product which varies smoothly between different points p. This inner product is given by

$$\langle \mathbf{v}, \mathbf{u} \rangle_p = \mathbf{v}^T \mathbf{G} \mathbf{u} \tag{22}$$

where $\mathbf{G} = (g_{ij})$ is called the Riemannian metric tensor of manifold S at point p and $\mathbf{v}, \mathbf{u} \in T_p$ [16]. For Euclidean space $\mathbf{G} = \mathbf{I}$ [1] and Equation (22) simplifies to the Euclidean inner product

$$\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{v}^T \mathbf{u}. \tag{23}$$

As a consequence of (22), the squared norm of vector $\mathbf{v}$ in Riemannian manifold is

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle_p = \mathbf{v}^T \mathbf{G} \mathbf{v} \tag{24}$$

which is analogous with the squared norm in Euclidean space

$$\|\mathbf{v}\|^2 = \langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v}^T \mathbf{v}. \tag{25}$$

If vectors $\mathbf{v}$ and $\mathbf{u}$ in (22) belong to different tangent spaces, that is $\mathbf{v} \in T_p$ and $\mathbf{u} \in T_{p'}$, a procedure called parallel transport has to be performed in order to transfer vector $\mathbf{u}$ from tangent space $T_{p'}$ to tangent space T_p. In this thesis, a parallelly transported version of vector $\mathbf{u}$ is denoted by $\tau\mathbf{u}$. As the process of parallel transport is quite complicated involving the introduction of the concept of a covariant derivative, it will not be reviewed here. For more information on parallel transport, one can turn to, for example, [1].

In differential geometry, a geodesic is a curve which locally minimizes the distance between points in a curved space. It is analogous to the concept of a straight line in Euclidean geometry. For example, if the manifold in question is spherical then its great circles can be regarded as geodesics.

3.3.2 Information Geometry

The application of differential geometry to probability and information theory is called information geometry. In information geometry, the parameter space $\boldsymbol{\xi}$ of probability distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ is regarded as a Riemannian manifold whose Riemannian metric tensor $\mathbf{G}$ is given by the Fisher information matrix [1, 16, 6]

$$g_{ij}(\boldsymbol{\xi}) = E_q \left\{ \frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i} \frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_j} \right\} = E_q \left\{ -\frac{\partial^2 \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i \partial \xi_j} \right\}. \tag{26}$$

There are also other ways of selecting the Riemannian metric tensor $\mathbf{G}$ but using the Fisher information is the most popular choice because of its invariance and covariance properties [25].

3.3.3 Natural Gradient Descent

If the geometry of the parameter space of $\mathcal{C}(\boldsymbol{\xi})$ is considered Riemannian, the direction of the steepest ascent is given, instead of the gradient, by the natural gradient [2]

$$\tilde{\nabla}\mathcal{C}(\boldsymbol{\xi}) = \mathbf{G}^{-1}(\boldsymbol{\xi})\nabla\mathcal{C}(\boldsymbol{\xi}). \tag{27}$$

Thus, the gradient descent method of Section 3.1 becomes

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - \lambda\tilde{\nabla}\mathcal{C}(\boldsymbol{\xi}_k). \tag{28}$$

This is called the natural gradient descent algorithm.

The step size λ is again obtained with line search which in the presence of a Riemannian manifold should be made along a geodesic. Since the derivation of geodesics is typically difficult and their use computationally expensive, it is customary to approximate using Euclidean straight lines.

The matrix inversion required for the evaluation of the natural gradient in (27) would be prohibitively expensive if the full matrix had to be inverted. Luckily, because of the typical factorizing approximation of Equation (9), the matrix $\mathbf{G}$ becomes block diagonal [12]. Hence, it can be inverted effectively since

$$\mathrm{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_n)^{-1} = \mathrm{diag}(\mathbf{A}_1^{-1}, \ldots, \mathbf{A}_n^{-1}) \tag{29}$$

where $\mathrm{diag}(\cdot)$ denotes a block diagonal matrix which has the given matrices as its diagonal blocks.

3.3.4 Riemannian Conjugate Gradient Algorithm

The natural gradient descent algorithm can be improved by using conjugate directions as described in Section 3.2 by replacing the gradient in (16) with the natural gradient

$$\mathbf{g}_k \hookrightarrow \tilde{\mathbf{g}}_k = \tilde{\nabla}\mathcal{C}(\boldsymbol{\xi}_k). \tag{30}$$

Furthermore, the squared norms and the inner products in (17) and (18) have to be taken in the Riemannian sense in accordance with Equations (22) and (24) and the previous natural gradient vector $\tilde{\mathbf{g}}_{k-1}$ has to be transferred to the tangent space of the current gradient with parallel transport. When the line search is, in addition, performed along a geodesic, the resulting algorithm is called the Riemannian conjugate gradient algorithm [24].

3.3.5 Natural Conjugate Gradient Algorithm

As in the case of natural gradient descent, it is beneficial to approximate the line search along geodesics in the Riemannian conjugate gradient with line search along Euclidean straight lines. It was further suggested in [12], that the parallel transport of $\tilde{\mathbf{g}}_{k-1}$ could be approximated with an identity mapping and that the squared norms

and the inner products in (17) and (18) could be taken in an Euclidean sense. This resulting algorithm is called the natural conjugate gradient (NCG) algorithm and it was shown to significantly improve the performance of the CG algorithm when applied to the NSSM of [26].

Another version of the NCG algorithm can be derived by performing the vector operations in (17) and (18) in accordance with Equations (22) and (24) while still approximating the parallel transport of $\tilde{\mathbf{g}}_{k-1}$ with an identity mapping. The costly vector-matrix multiplications can be avoided by noting that

$$\|\tilde{\mathbf{g}}_k\|^2 = \tilde{\mathbf{g}}_k^T \mathbf{G}_k \tilde{\mathbf{g}}_k = \tilde{\mathbf{g}}_k^T \mathbf{G}_k \mathbf{G}_k^{-1} \mathbf{g}_k = \tilde{\mathbf{g}}_k^T \mathbf{g}_k. \tag{31}$$

Because the norm of a vector is invariant under parallel transport [1], the resulting equations are

$$\beta_k = \frac{\|\tilde{\mathbf{g}}_k\|^2}{\|\tau\tilde{\mathbf{g}}_{k-1}\|^2} = \frac{\|\tilde{\mathbf{g}}_k\|^2}{\|\tilde{\mathbf{g}}_{k-1}\|^2} = \frac{\tilde{\mathbf{g}}_k^T \mathbf{G}_k \tilde{\mathbf{g}}_k}{\tilde{\mathbf{g}}_{k-1}^T \mathbf{G}_{k-1} \tilde{\mathbf{g}}_{k-1}} = \frac{\tilde{\mathbf{g}}_k^T \mathbf{g}_k}{\tilde{\mathbf{g}}_{k-1}^T \mathbf{g}_{k-1}} \tag{32}$$

in the case of the Fletcher-Reeves formula (17) and

$$\beta_k = \frac{\langle (\tilde{\mathbf{g}}_k - \tau\tilde{\mathbf{g}}_{k-1}), \tilde{\mathbf{g}}_k \rangle}{\|\tau\tilde{\mathbf{g}}_{k-1}\|^2} \approx \frac{(\tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1})^T \mathbf{G}_k \tilde{\mathbf{g}}_k}{\tilde{\mathbf{g}}_{k-1}^T \mathbf{G}_{k-1} \tilde{\mathbf{g}}_{k-1}} = \frac{(\tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1})^T \mathbf{g}_k}{\tilde{\mathbf{g}}_{k-1}^T \mathbf{g}_{k-1}} \tag{33}$$

in the case of the Polak-Ribiére formula (18). It should be noted that with this approach the Riemannian Fletcher-Reeves formula (32) can be calculated without any approximations and the only approximation made in the calculation of the Riemannian Polak-Ribiére formula is the approximation $\tau\tilde{\mathbf{g}}_{k-1} \approx \tilde{\mathbf{g}}_{k-1}$ in the numerator.

4 Variational Mixture of Gaussians

In this chapter, the machine learning algorithms discussed above are applied to learning the mixture of Gaussians model which is introduced in Section 4.1. Section 4.2 constructs a probabilistic model for learning the mixture of Gaussians and gives the update equations for the VB EM algorithm. The cost function, which is used to derive the NCG algorithm of Section 4.4, is given in Section 4.3. The quadratic polynomial interpolation based line search used in this thesis is introduced in Section 4.5 while Section 4.6 discusses some important implementation details.

4.1 The Mixture of Gaussians Model

The model $\mathcal{M}$, which is studied in detail in this thesis, is the mixture of Gaussians model. It is a probability distribution which is a linear combination of K Gaussian distributions [5]

$$p(\mathbf{x}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{34}$$

where $\mathbf{x}$ is a D-dimensional random variable and $\boldsymbol{\pi} = [\pi_1 \cdots \pi_K]^T$ are called the mixing coefficients while $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the mean and the covariance matrix of the kth Gaussian component. The inverse of the covariance matrix $\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$ is called the precision matrix. Figure 2 shows a plot of a two dimensional mixture of Gaussians. More information on the multidimensional Gaussian distribution can be found in Appendix A.

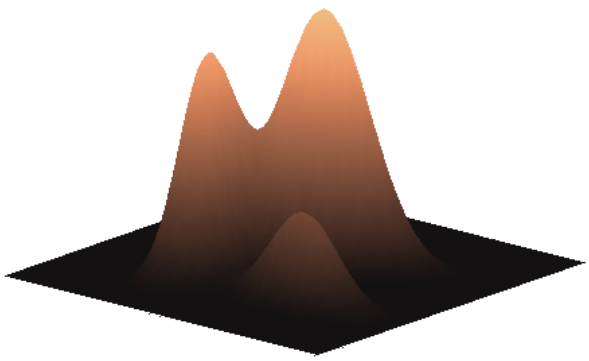


Figure 2: A surface plot of a two dimensional mixture of Gaussians with three Gaussian components. All components have different mixing coefficients, means and covariances.

4.2 VB EM for the Mixture of Gaussians Model

This section is completely based on the variational treatment of the mixture of Gaussians model in [5]. Because of this, some details of the derivation of the VB EM algorithm for the MoG model will be omitted here and we will concentrate only on the most important results.

In the case of the mixture of Gaussians model, the latent variables discussed in Section 2.6 are the information on which one of the K Gaussian components has generated a particular observation $\mathbf{x}_n$. This information will be represented with a K-dimensional binary vector $\mathbf{z}_n$ whose elements z_{nk} are either 0 or 1 where 1 denotes the component responsible for generating the observed data point $\mathbf{x}_n$ in question. It should be noted that only one component can be responsible for generating a single observation and thus the elements of the vector $\mathbf{z}_n$ sum to unity. Let N denote the total number of observed data points. Now, all the N latent variables of the model can be regarded as forming a latent variable matrix $\mathbf{Z} = (z_{nk})$ of the order $N \times K$.

Given the mixing coefficients $\boldsymbol{\pi}$, the probability distribution over the latent variables is given by

$$p(\mathbf{Z}|\boldsymbol{\pi}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \pi_k^{z_{nk}}. \tag{35}$$

As we want to use conjugate priors in our treatment, we next introduce a Dirichlet prior for the mixing coefficients

$$p(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_0) \tag{36}$$

where $\boldsymbol{\alpha}_0 = [\alpha_0 \cdots \alpha_0]^T$ is a K-dimensional hyperparameter vector whose elements are all given by α_0 due to symmetry.

Similarly, the distribution over the data $\mathbf{X}$ given the latent variables $\mathbf{Z}$, the means $\boldsymbol{\mu}$ and the precision matrices $\boldsymbol{\Lambda}$ can be written as

$$p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N}\prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})^{z_{nk}}. \tag{37}$$

Note that we are assuming here that the data vectors $\mathbf{x}_n$ are independent and identically distributed. In this case, the conjugate prior for the component parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ is given by the Gaussian-Wishart distribution

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_0, (\beta_0\boldsymbol{\Lambda}_k)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_0, \nu_0). \tag{38}$$

The joint distribution over all the random variables of the model is then given by

$$p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})p(\mathbf{Z}|\boldsymbol{\pi})p(\boldsymbol{\pi})p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda}). \tag{39}$$

This resulting model can be illustrated with the graphical model shown in Figure 3. More information on the Dirichlet and Wishart distributions can be found in Appendix A.

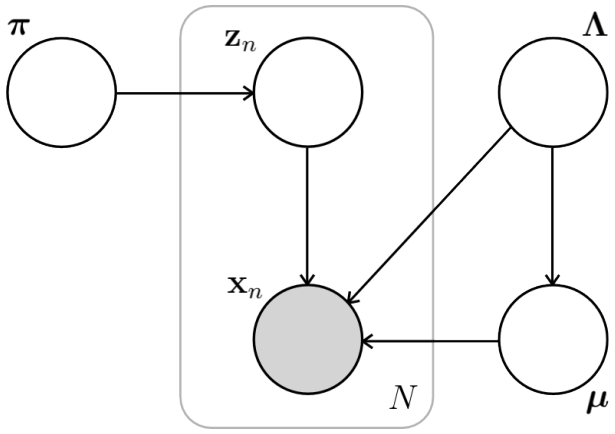


Figure 3: A graphical model representing the Bayesian treatment of the mixture of Gaussians model [5]. Observed values are marked with a grey circle. In this case, only the data $\mathbf{X}$ is observed and all latent variables and model parameters have to be inferred from the data. The rectangular plate denotes a set of N independent and identically distributed observations $\mathbf{x}_n, n = 1 \dots N$ along with corresponding latent variables $\mathbf{z}_n, n = 1 \dots N$.

We now make the factorizing approximation described by Equation (13)

$$q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\mathbf{Z})q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \tag{40}$$

and use Equation (10) along with Equation (39) to first update $q(\mathbf{Z})$ (E-step) and subsequently update $q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$ (M-step). The resulting approximate posterior distributions are

$$q(\mathbf{Z}) = \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{z_{nk}} \tag{41}$$

and

$$q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\boldsymbol{\pi})q(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\boldsymbol{\pi}) \prod_{k=1}^{K} q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \tag{42}$$

where

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \tag{43}$$

$$q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k, \nu_k). \tag{44}$$

In expressing the update rules for the distribution parameters in Equations (41),

(43) and (44), we will find the following definitions useful:

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{45}$$

$$\overline{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{46}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \overline{\mathbf{x}}_k)(\mathbf{x}_n - \overline{\mathbf{x}}_k)^T \tag{47}$$

$$\ln \tilde{\Lambda}_k = \sum_{i=1}^{D} \psi \left(\frac{\nu_k + 1 - i}{2} \right) + D \ln 2 + \ln |\mathbf{W}_k| \tag{48}$$

$$\ln \tilde{\pi}_k = \psi(\alpha_k) - \psi(\hat{\alpha}) \tag{49}$$

where D is the dimensionality of the data, $\hat{\alpha}$ is defined by Equation (A3) and $\psi(\cdot)$ is the digamma function which is defined as the derivative of the logarithmic gamma function, that is

$$\psi(x) = \frac{d}{dx} \ln \Gamma(x). \tag{50}$$

Using these definitions, the parameters r_{nk} of the approximate posterior over latent variables $q(\mathbf{Z})$ which are updated in the E-step are given by

$$r_{nk} = \frac{\rho_{nk}}{\sum_{l=1}^{K} \rho_{nl}} \tag{51}$$

where

$$\rho_{nk} = \tilde{\pi}_k \tilde{\Lambda}_k^{1/2} \exp \left(-\frac{D}{2\beta_k} - \frac{\nu_k}{2} (\mathbf{x}_n - \mathbf{m}_k)^T \mathbf{W}_k (\mathbf{x}_n - \mathbf{m}_k) \right). \tag{52}$$

The parameters r_{nk} are called *responsibilities* because they represent the responsibility the kth component takes in explaining the nth observation. The responsibilities can be arranged into a matrix $\mathbf{R} = (r_{nk})$ and will have to satisfy the following conditions:

$$0 \leq r_{nk} \leq 1 \tag{53}$$

$$\sum_{k=1}^{K} r_{nk} = 1. \tag{54}$$

The parameter update equations for the M-step are then given by

$$\alpha_k = \alpha_0 + N_k \tag{55}$$

$$\beta_k = \beta_0 + N_k \tag{56}$$

$$\nu_k = \nu_0 + N_k \tag{57}$$

$$\mathbf{m}_k = \frac{1}{\beta_0 + N_k} (\beta_0 \mathbf{m}_0 + N_k \overline{\mathbf{x}}_k) \tag{58}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k} (\overline{\mathbf{x}}_k - \mathbf{m}_0)(\overline{\mathbf{x}}_k - \mathbf{m}_0)^T. \tag{59}$$

4.3 The Cost Function

We can use Equation (8) along with Equations (35)-(44) to evaluate the cost function for the learning process

$$\begin{aligned}\mathcal{C} =& \sum_{\mathbf{Z}} \int_{\boldsymbol{\pi}} \int_{\boldsymbol{\mu}} \int_{\boldsymbol{\Lambda}} q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \ln \frac{q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})} d\boldsymbol{\pi} d\boldsymbol{\mu} d\boldsymbol{\Lambda} \\ =& E_q \left\{\ln q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} - E_q \left\{\ln p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} \\ =& E_q \left\{\ln q(\mathbf{Z})\right\} + E_q \left\{\ln q(\boldsymbol{\pi})\right\} + E_q \left\{\ln q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} - E_q \left\{\ln p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} \\ & - E_q \left\{\ln p(\mathbf{Z}|\boldsymbol{\pi})\right\} - E_q \left\{\ln p(\boldsymbol{\pi})\right\} - E_q \left\{\ln p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right\}. \end{aligned} \tag{60}$$

These expectations can be evaluated to give [5]

$$E_q \left\{\ln q(\mathbf{Z})\right\} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \ln r_{nk} \tag{61}$$

$$E_q \left\{\ln q(\boldsymbol{\pi})\right\} = \sum_{k=1}^{K} (\alpha_k - 1) \ln \tilde{\pi}_k + \ln C(\boldsymbol{\alpha}) \tag{62}$$

$$E_q \left\{\ln q(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} = \sum_{k=1}^{K} \left\{ \frac{1}{2} \ln \tilde{\Lambda}_k + \frac{D}{2} \ln \frac{\beta_k}{2\pi} - \frac{D}{2} - H_q\{\boldsymbol{\Lambda}_k\} \right\} \tag{63}$$

$$\begin{aligned} E_q \left\{\ln p(\mathbf{X}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} =& \frac{1}{2} \sum_{k=1}^{K} N_k \Big\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) \\ & - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \Big\} \end{aligned} \tag{64}$$

$$E_q \left\{\ln p(\mathbf{Z}|\boldsymbol{\pi})\right\} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \ln \tilde{\pi}_k \tag{65}$$

$$E_q \left\{\ln p(\boldsymbol{\pi})\right\} = \ln C(\boldsymbol{\alpha}_0) + (\alpha_0 - 1) \sum_{k=1}^{K} \ln \tilde{\pi}_k \tag{66}$$

$$\begin{aligned} E_q \left\{\ln p(\boldsymbol{\mu}, \boldsymbol{\Lambda})\right\} =& \frac{1}{2} \sum_{k=1}^{K} \Big\{ D \ln \frac{\beta_0}{2\pi} + \ln \tilde{\Lambda}_k - \frac{D\beta_0}{\beta_k} \\ & - \beta_0 \nu_k (\mathbf{m}_k - \mathbf{m}_0)^T \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0) \Big\} + K \ln B(\mathbf{W}_0, \nu_0) \\ & + \frac{\nu_0 - D - 1}{2} \sum_{k=1}^{K} \ln \tilde{\Lambda}_k - \frac{1}{2} \sum_{k=1}^{K} \nu_k \operatorname{Tr}(\mathbf{W}_0^{-1} \mathbf{W}_k) \end{aligned} \tag{67}$$

where $\operatorname{Tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A}$ and $H_q\{\boldsymbol{\Lambda}_k\}$ is the entropy of the distribution $q(\boldsymbol{\Lambda}_k)$ given by Equation (A12) of Appendix A. The coefficients $C(\boldsymbol{\alpha})$ and $B(\mathbf{W}_0, \nu_0)$ are given by Equations (A2) and (A9) respectively.

The cost function $\mathcal{C}$ given by Equation (60) can be used to determine when the VB EM algorithm has converged. The cost function will decrease during each iteration

and when the difference between the previous cost function value $\mathcal{C}_{k-1}$ and the current value $\mathcal{C}_k$ becomes sufficiently small we can assume that the learning process has converged.

4.4 Natural Conjugate Gradient for the Mixture of Gaussians Model

To be able to compare the VB EM and NCG algorithms, we assume that the approximate posterior distribution $q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$ takes the same functional form as in the case of the VB EM algorithm. Thus, the fixed form posterior distributions are given by Equations (41), (43) and (44) and the cost function which is to be minimized by the NCG algorithm is given by Equation (60). In this thesis, we will only be optimizing the responsibilities $r_{nk}, n = 1 \dots N, k = 1 \dots K$ and the means $\mathbf{m}_k, k = 1 \dots K$ using gradient-based methods. All other model parameters, namely the parameters $\alpha_k, k = 1 \dots K$ of the Dirichlet distribution, the parameters $\beta_k, k = 1 \dots K$ controlling the covariance of the component means as well as the parameters $\mathbf{W}_k, k = 1 \dots K$ and $\nu_k, k = 1 \dots K$ of the Wishart distribution, are updated using the VB EM update Equations (55), (56), (57) and (59) during the line search procedure.

There are a few things that have to be taken into account when deriving gradient-based algorithms for the mixture of Gaussians model. Firstly, the responsibilities have to satisfy the constraints given by Equations (53) and (54). This can be enforced by using the *softmax* parametrization

$$r_{nk} = \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}}. \tag{68}$$

It can be easily seen that by using this parametrization the responsibilities are always positive and $\sum_{k=1}^{K} r_{nk} = 1$. As a results it holds that $0 \leq r_{nk} \leq 1$ and the restricting conditions (53) and (54) are satisfied for all values of γ_{nk}.

Secondly, if we set the responsibilities $r_{nk}, n = 1 \dots N, k = 1 \dots K-1$ to some values, the values of $r_{nK}, n = 1 \dots N$ are given by condition (54), that is $r_{nK} = 1 - \sum_{k=1}^{K-1} r_{nk}$. As a results, the number of degrees of freedom in the responsibilities of the model is not the number of responsibilities NK but instead $N(K-1)$. When we are using the parametrization (68), this means that we can regard the parameters γ_{nK} as constants and only optimize the cost function with respect to parameters $\gamma_{nk}, n = 1 \dots N, k = 1 \dots K-1$. This is especially important when using the natural gradient.

As a results, as far as the gradient is concerned, the cost function is a function of the means $\mathbf{m}_k, k = 1 \dots K$ and the parameters $\gamma_{nk}, n = 1 \dots N, k = 1 \dots K-1$

$$\mathcal{C} = \mathcal{C}(\mathbf{m}_1, \dots, \mathbf{m}_K, \gamma_{11}, \dots, \gamma_{1K-1}, \dots, \gamma_{N1}, \dots, \gamma_{NK-1}). \tag{69}$$

We will denote the gradient with respect to both the means $\mathbf{m}_k$ and the parameters

γ_{nk} by $\nabla_{\mathbf{m},\boldsymbol{\gamma}}$. Hence

$$\nabla_{\mathbf{m},\boldsymbol{\gamma}}\mathcal{C} = \begin{bmatrix}\nabla_{\mathbf{m}}\mathcal{C} \\ \nabla_{\boldsymbol{\gamma}}\mathcal{C}\end{bmatrix} = \begin{bmatrix}\nabla_{\mathbf{m}_1}\mathcal{C} \\ \vdots \\ \nabla_{\mathbf{m}_K}\mathcal{C} \\ \partial_{\gamma_{11}}\mathcal{C} \\ \vdots \\ \partial_{\gamma_{1K-1}}\mathcal{C} \\ \vdots \\ \partial_{\gamma_{N1}}\mathcal{C} \\ \vdots \\ \partial_{\gamma_{NK-1}}\mathcal{C}\end{bmatrix} \tag{70}$$

where $\partial_{\gamma_{nk}}\mathcal{C} = \frac{\partial\mathcal{C}}{\partial\gamma_{nk}}$.

The gradient of the cost function (60) with respect to $\mathbf{m}_k$ is given by

$$\nabla_{\mathbf{m}_k}\mathcal{C} = \nu_k\mathbf{W}_k(N_k(\mathbf{m}_k - \overline{\mathbf{x}}_k) + \beta_0(\mathbf{m}_k - \mathbf{m}_0)), k = 1\ldots K \tag{71}$$

and the derivative with respect to γ_{nk} is given by

$$\frac{\partial\mathcal{C}}{\partial\gamma_{nk}} = E_{nk} - r_{nk}F_n, n = 1\ldots N, k = 1\ldots K-1 \tag{72}$$

where

$$E_{nk} = r_{nk}\left\{\ln r_{nk} - \ln\tilde{\pi}_k - \frac{1}{2}\left(\ln\tilde{\Lambda}_k - \frac{D}{\beta_k} - D\ln 2\pi - \nu_k(\mathbf{x}_n - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_n - \mathbf{m}_k)\right)\right\} \tag{73}$$

and

$$F_n = \sum_{k=1}^{K} E_{nk}. \tag{74}$$

These equations are derived in Appendix B.

We can update the responsibilities r_{nk} without having to evaluate and store the parameters γ_{nk} by noting that

$$r'_{nk} = \frac{e^{\gamma_{nk}+\Delta\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}+\Delta\gamma_{nl}}} = \frac{\sum_{l=1}^{K} e^{\gamma_{nl}}}{\sum_{l=1}^{K} e^{\gamma_{nl}+\Delta\gamma_{nl}}}\frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}}e^{\Delta\gamma_{nk}} = c_n r_{nk}e^{\Delta\gamma_{nk}} \tag{75}$$

where r'_{nk} is the new responsibility, $\Delta\gamma_{nk}$ is the change in the parameter γ_{nk} determined by line search in the direction of the negative gradient and c_n is a normalizing constant which makes sure that $\sum_{k=1}^{K} r'_{nk} = 1$. Thus, c_n can also be expressed in the form $c_n = (\sum_{k=1}^{K} r_{nk}e^{\Delta\gamma_{nk}})^{-1}$ and we can update the responsibilities using the formula

$$r'_{nk} = \frac{r_{nk}e^{\Delta\gamma_{nk}}}{\sum_{l=1}^{K} r_{nl}e^{\Delta\gamma_{nl}}}. \tag{76}$$

In order to use the natural gradient, we need to know the Riemannian metric tensor $\mathbf{G}$ of the parameter space $(\mathbf{m}, \boldsymbol{\gamma})$. The matrix $\mathbf{G}$ is given by Equation (26) and is derived in Appendix B. The resulting matrix is the block diagonal matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{A}_1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & \mathbf{A}_k & & & & 0 & & \\ & & & \ddots & & & & & \\ & & & & \mathbf{A}_K & & & & \\ & & & & & \mathbf{B}_1 & & & \\ & & & & & & \ddots & & \\ & & 0 & & & & & \mathbf{B}_n & \\ & & & & & & & & \ddots \\ & & & & & & & & & \mathbf{B}_N \end{bmatrix} \tag{77}$$

where $\mathbf{A}_k = \beta_k \nu_k \mathbf{W}_k$, $\mathbf{B}_n = -\mathbf{r}_n^T \mathbf{r}_n + \text{diag}(\mathbf{r}_n)$ and $\mathbf{r}_n$ is the nth row of the responsibility matrix $\mathbf{R}$ except for element r_{nK}, that is $\mathbf{r}_n = [r_{n1} \cdots r_{nK-1}]$. diag($\mathbf{a}$) is used here to denote a square matrix which has the elements of vector $\mathbf{a}$ on its diagonal.

The inverse of matrix $\mathbf{G}$ required for the evaluation of the natural gradient is easily calculated using Equation (29). We can now also motivate our earlier discussion about the number of degrees of freedom in responsibilities. Had we considered the number of degrees of freedom to be NK, the row vector $\mathbf{r}_n$ would have consisted of the whole nth row the the matrix $\mathbf{R}$. This would have made the matrices $\mathbf{B}_n$ singular and we would not have been able to evaluate the natural gradient.

4.5 Line Search

Performing line search in gradient-based algorithms and in the pattern search acceleration of VB EM is not as trivial as it might first seem. In general, there is a trade-off of performance and accuracy. The accuracy of the line search procedure can be improved by increasing the number of cost function evaluations in the search direction which increases the computation time of the procedure. While many line search algorithms have been developed, there is no perfect solution and the selection has to be made based on the problem at hand.

In this thesis, line search is based on quadratic polynomial interpolation. Therefore, a second degree polynomial $p(x) = ax^2 + bx + c$ is fitted to the cost function values in the search direction, and the point x_{min} where the polynomial has its minimum is used as the optimal step size. This point is given by [25]

$$x_{min} = \frac{1}{2} \frac{\alpha_{23} f(x_1) + \alpha_{31} f(x_2) + \alpha_{12} f(x_3)}{\beta_{23} f(x_1) + \beta_{31} f(x_2) + \beta_{12} f(x_3)} \tag{78}$$

where $f(x_1)$, $f(x_2)$ and $f(x_3)$ are three known cost function values with step sizes $x_1 < x_2 < x_3$ in the search direction, $\alpha_{ij} = x_i^2 - x_j^2$ and $\beta_{ij} = x_i - x_j$.

The advantage of this method is that in the best case scenario the cost function has to be evaluated only three times in the search direction in order to find the optimal step size. The problem is that in order to ensure that x_{min} is an interpolated minimum of the polynomial $p(x)$ the condition

$$f(x_1) > f(x_2) \wedge f(x_2) < f(x_3) \tag{79}$$

has to be satisfied. If the condition is not satisfied the points x_1, x_2 and x_3 have to be adjusted accordingly. This can be done by performing interpolation or extrapolation based on Equation (78). However, multiple safeguards have to be added in order to make sure that the interpolated or extrapolated value is reasonable. For example, if $f(x_1) > f(x_2) \wedge f(x_2) > f(x_3)$ and we get $x_{min} > x_3$ we can adjust the points so that $x_2 \leftarrow x_3$ and $x_3 \leftarrow 2x_{min}$.

4.6 Implementation Details

At least the following details have to be taken into account when implementing the algorithms discussed above:

- The gradient-based algorithms have to be initialized with both initial parameters α_k, β_k, ν_k, $\mathbf{m}_k$ and $\mathbf{W}_k$ as well as initial responsibilities r_{nk}. This is achieved by setting the parameters to some initial values followed by a VB EM E-step giving us the initial responsibilities. Additionally, a VB EM M-step updating all the parameters except for the means $\mathbf{m}_k$ is performed before using the gradient-based learning algorithm since otherwise the gradient $\nabla_{\boldsymbol{\gamma}}\mathcal{C}$ would be zero on the first iteration.

- Numerical problems caused be the limited precision of floating point numbers are likely to cause problems with the algorithms. It was found out the there are three operations which are prone to numerical problems in the algorithms discussed here. These are the evaluations of the logarithms $\ln r_{nk}$ in Equations (61) and (73) and the inversion of the matrices $\mathbf{B}_n$ in Equation (77). All these operations are likely to cause problems with small responsibilities. While there are many possible solutions to this kind of problems, the approach taken in this thesis was to limit the responsibilities to always satisfy the inequation $r_{nk} \geq 10^{-10}$.

- From a performance point of view, it is reasonable to completely remove a Gaussian component from the data structures if the responsibility it takes in explaining the data becomes sufficiently small, that is $N_k < \delta$. It was found out that the value $\delta = 0.1$ seems to work well for most datasets. It should also be noted that the conjugate gradient algorithms have to be restarted with the negative of the gradient when a component is removed.

- The algorithms were considered to have converged when $\mathcal{C}_{k-1} - \mathcal{C}_k < \varepsilon$ for two consecutive iterations. Checking for the termination criterion for two

iterations before terminating the algorithm is required because the conjugate gradient algorithm can produce a search direction where the optimal step size is $\lambda = 0$. The value $\varepsilon = 10^{-8}N$ is used in all the experiments unless otherwise mentioned.

- As discussed above, there are different variants of the CG algorithms. In initial testing, it was found out that the Polak-Ribiére-based CG outperformed the Fletcher-Reeves-based CG in most cases. With NCG, it was also found out that Equation (33) gave generally better results than Equation (18). Therefore, all the experiments in Section 5 are conducted using the Polak-Ribiére formula and the version of NCG where Equation (33) is used.

- Similarly, the performance of the pattern search acceleration of VB EM is greatly affected by the frequency of pattern searches. It was found out that performing the pattern search on every 8th iteration was a good compromise in terms of algorithm performance.

- It is a natural choice to initially select the points where to evaluate the cost function in line search to be $x_1 = 0$ and $x_2 = \frac{x_3}{2}$. However, the initial value of x_3 has to be somehow predetermined. It was found out that the following choices for the first line search gave good results: $x_3 = 10$ for VB EM with pattern search, $x_3 = 0.002$ for algorithms based on standard gradient and $x_3 = 2$ for algorithms based on natural gradient. After the first line search, the gradient-based algorithms use two times the optimal step size of the previous iteration as the value of x_3.

- Because the number of variables in the gradient is large, the conjugate gradient algorithms are restarted every $\sqrt{n}$ iterations instead of the typically recommended n iterations.

5 Experiments

The algorithms for learning the mixture of Gaussians model discussed in Chapter 4 are experimentally studied and compared to each other in this chapter using both artificial and real-world data. The experiment setup is outlined in Section 5.1. The results of the experiments with artificial data are given in Section 5.2 followed by Section 5.3 where the algorithms are applied to the task of image segmentation.

5.1 Experiment Setup

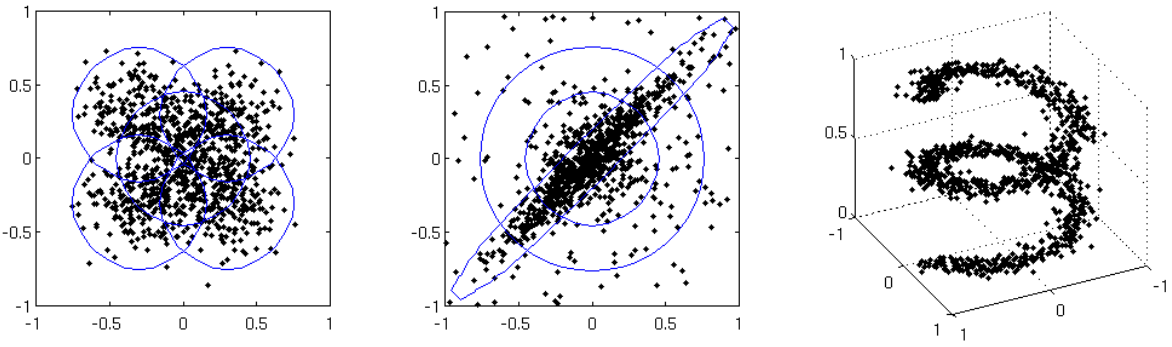
In the following experiments, all the datasets are scaled so that the data is within the hypercube which has its center point in the origin of the space and a side length of 2. For example, for a two dimensional dataset, this means that the data is within the square $[-1, 1] \times [-1, 1]$. The priors are set to the following values for all the experiments: $\alpha_0 = 1$, $\beta_0 = 1$, $\nu_0 = D$, $\mathbf{W}_0 = \frac{4}{D}\mathbf{I}$ and $\mathbf{m}_0 = \mathbf{0}$. These priors can be interpreted to describe our prior beliefs of the model when we anticipate having Gaussian components near the origin but are fairly uncertain about the number of the components.

The initial number of components is set to $K = 8$ unless otherwise mentioned with each component having a randomly generated initial mean $\mathbf{m}_k$ drawn from a Gaussian distribution with mean $\mathbf{m} = \mathbf{0}$ and covariance $\mathbf{\Sigma} = 0.16\mathbf{I}$. Other distribution parameters are initially set to the following values: $\alpha_k = 1$, $\beta_k = 10$, $\nu_k = D$ and $\mathbf{W}_k = \frac{4}{D}\mathbf{I}$ for all k.

Because different initial means can produce significantly different results in terms of required CPU time and achieved cost function value, all the experiments are repeated 30 times with different initial means.

5.2 Artificial Data

Figure 4 shows the artificial datasets used to compare the different algorithms. Dataset 1 shown in Figure 4(a) consists of 5 Gaussian components drawn from a real mixture of Gaussians using the Netlab library [17]. All the components are spherical and the distance between the means of the center component and the other components R can be changed. All the components have mixing coefficients $\pi_k = 0.2$ and the amount of data points is $N = 1000$ unless otherwise mentioned. Dataset 2 shown in Figure 4(b) consists of three Gaussian components with different covariances and mixing coefficients. This dataset is also generated using the Netlab library and has $N = 1000$ data points. Dataset 3 shown in Figure 4(c) is a three dimensional helix which, unlike the other datasets, is not drawn from a mixture of Gaussians. It has $N = 1000$ data points.



(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 4: Artificial datasets used in the experiments. Dataset 1 shown in Figure (a) consists of data drawn from a mixture of Gaussians with five identical components. The distance R between the center component and the other components can be changed. The data shown has $R = 0.3$. Dataset 2 shown in Figure (b) consists of data drawn from a mixture of Gaussians with three components while Dataset 3 shown in Figure (c) is a three dimensional helix.

5.2.1 Dataset 1

When different gradient-based algorithms are compared using Dataset 1 with $R = 0.3$, the results shown in Figure 5 are obtained. The curves shown are the median cost function values as a function of CPU time for the different algorithms. It can be seen that standard gradient and CG algorithms have problems locating even a decent optimum. Using the natural gradient, the quality of the optimum can be improved while NCG further improves the performance. It should be emphasized that the time scale of Figure 5 is logarithmic. Therefore, standard gradient is over 100 times slower than NCG. In contrast to other experiments discussed here, this experiment was conducted using the values $N = 500$, $\varepsilon = 10^{-7}N$ and the initial number of components $K = 5$ in order to make the standard gradient converge in a reasonable time. Out of these algorithms, only NCG is used in the experiments that follow.

When the performance of NCG, VB EM and VB EM accelerated with pattern searches is compared using Dataset 1 with $R = 0.3$, the results shown in Figure 6 are obtained. Figure 6(a) shows the median learning curves for the three different algorithms. It can be seen that the performance of NCG and VB EM with pattern searches is quite similar while the performance of VB EM is inferior to these two algorithms. Figure 6(b) shows the results of the individual initializations. The plotted values are the obtained final cost when the algorithm has converged and the CPU time required to achieve this. As can be seen, most of the initializations converge to a cost of just below 561 but there are a few exceptions to this, most notably the best optimum is only found by a single NCG run and a single pattern search run.

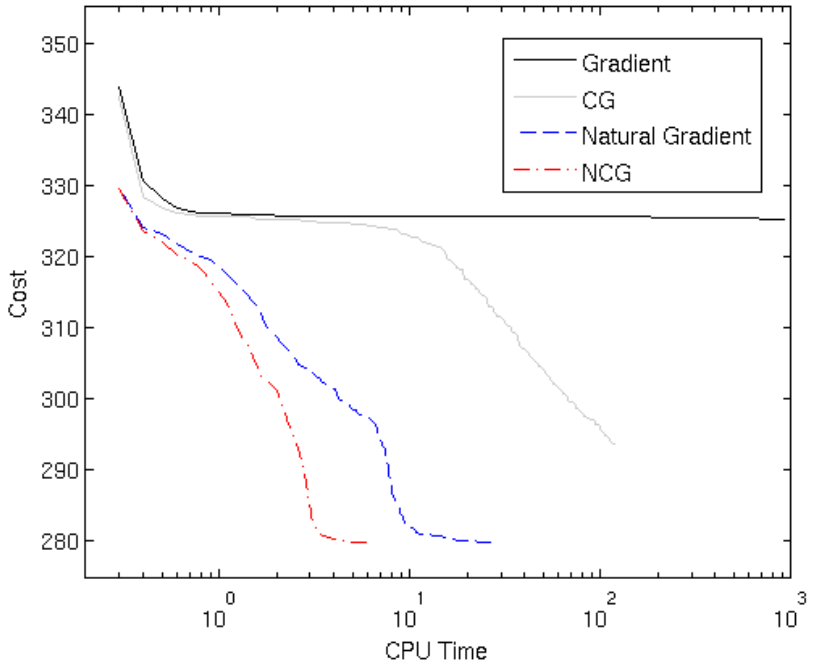
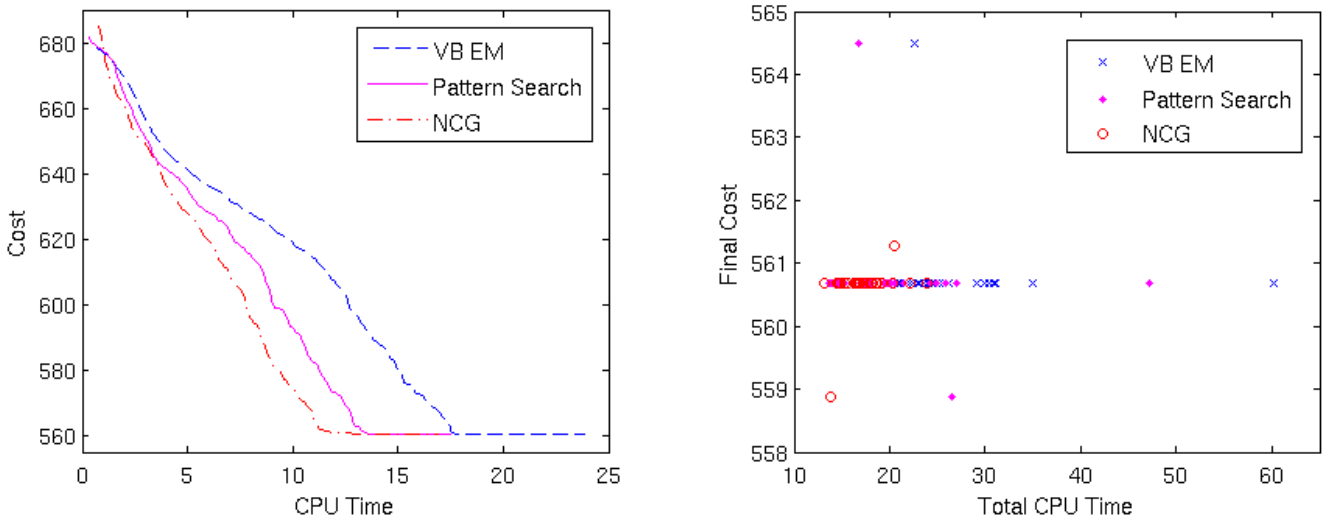


Figure 5: Comparison of gradient-based algorithms using Dataset 1 with $R = 0.3$. The learning curves are medians of 30 different initializations. Note that the time scale is logarithmic.

Different algorithms can also be compared with different values of R. For each value, the experiment was repeated 30 times with different initializations and the CPU times shown in Figure 7 are the medians of these experiments. It can be seen that with small values of R NCG outperforms VB EM while with large values of R VB EM performs better. Curiously, VB EM with pattern searches seems to achieve good results with all values of R. All algorithms achieved approximately the same cost function values in this experiment so the CPU times shown can be compared directly.

One of the most important features of an algorithm is how it performs as the number of data points becomes larger. This was studied using Dataset 1 with $R = 0.3$ by changing the number of data points N. For each value of N, the experiment was repeated 30 times with different initializations. Figure 8 shows the median CPU time per data point required for the convergence of the algorithms. Once again, there are no big differences in the cost function values achieved by the algorithms so the CPU times can be compared directly. It can be seen that NCG scales better than the other algorithms as the number of data points grows. Especially the performance of VB EM is greatly reduced with a large number of data points.



(a) (b)

Figure 6: Comparison of algorithms using Dataset 1 with $R = 0.3$. Figure (a) shows the median cost of 30 runs as a function of CPU time while Figure (b) shows the final cost when the algorithms have converged and the corresponding CPU time for all the 30 initializations.

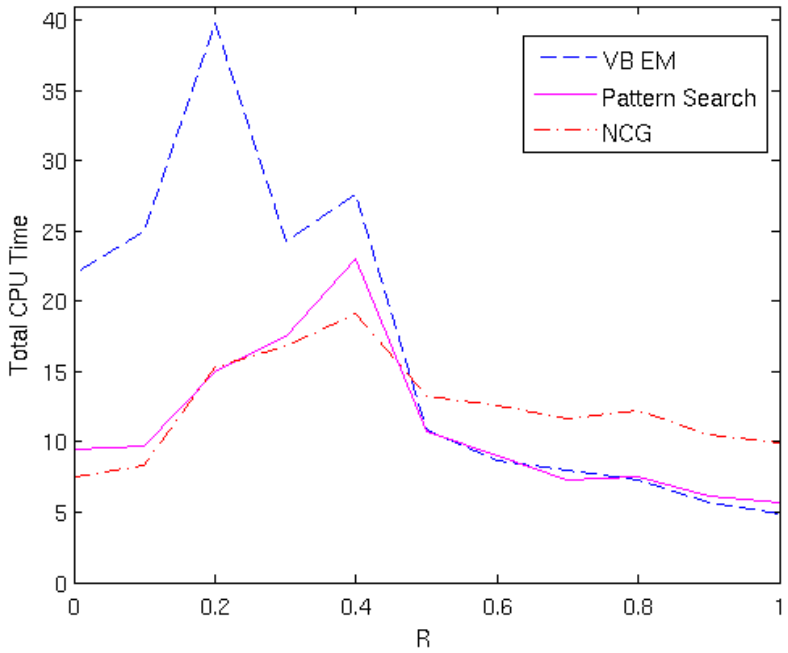


Figure 7: The median total CPU time required for convergence as a function of R in Dataset 1.

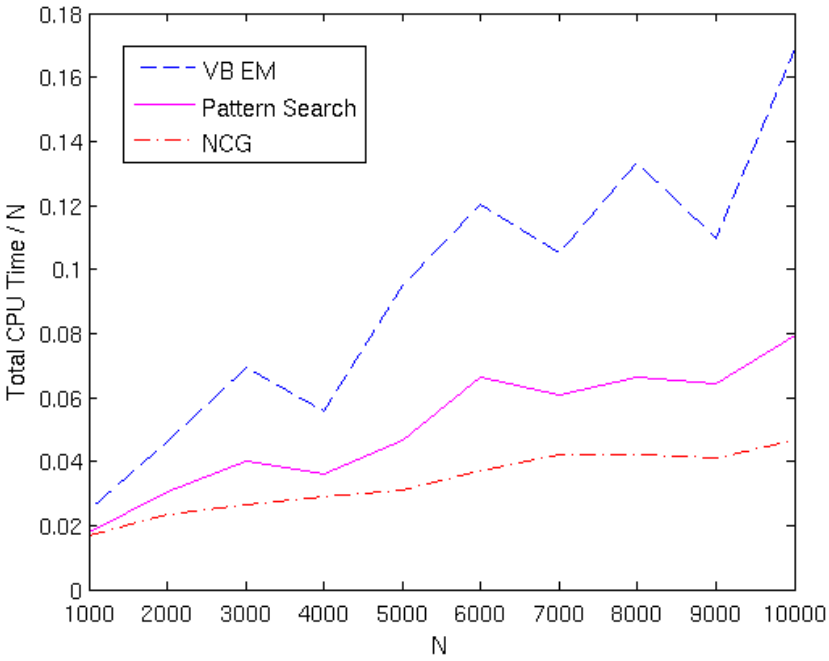


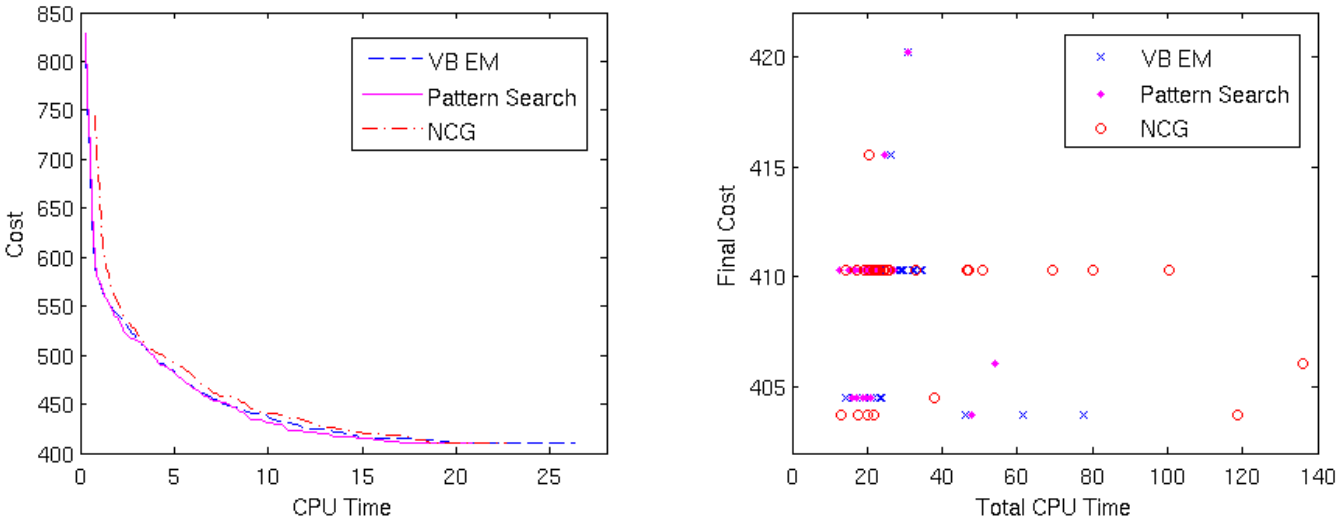
Figure 8: The median total CPU time divided by the number of data points N as a function of N using Dataset 1 with $R = 0.3$.

5.2.2 Dataset 2

Experimental results acquired when using Dataset 2 are shown in Figure 9. From the median learning curves of Figure 9(a), it can be seen that the performance of the algorithms is almost identical. Some differences can however be noted by looking at Figure 9(b) which shows the CPU time and cost function value when the algorithms have converged for all the 30 initializations. It can be seen that the best optimum is achieved by 5 NCG runs compared to 3 for VB EM and only a single run for pattern search.

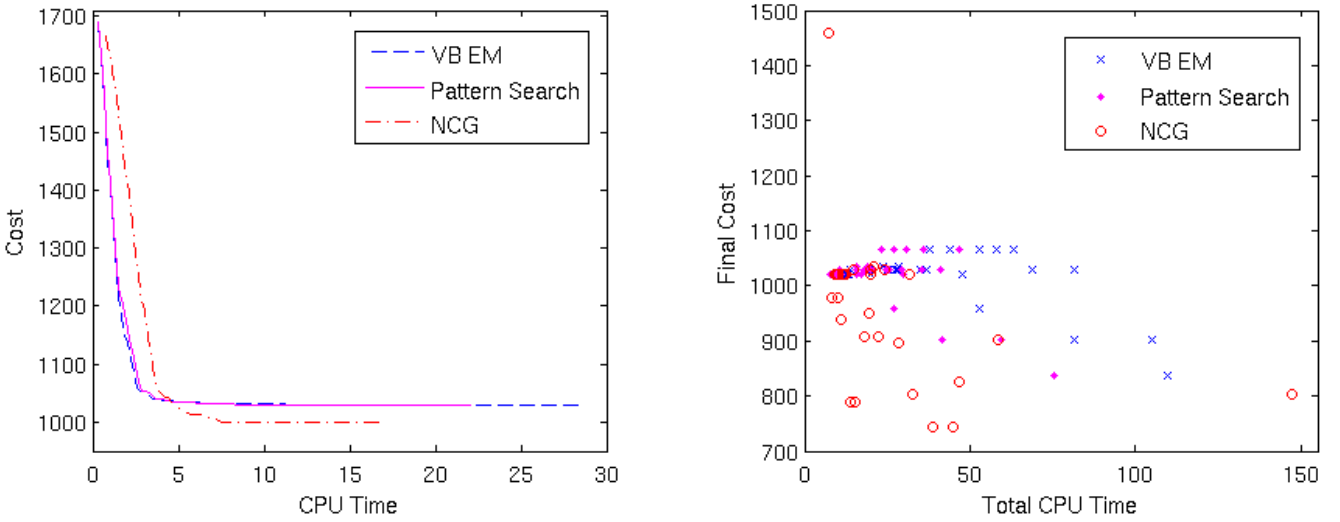
5.2.3 Dataset 3

Using Dataset 3, the results shown in Figure 10 are acquired. It can be seen both by looking at the learning curves of Figure 10(a) and the convergence results of Figure 10(b) that NCG clearly outperforms the other algorithms. It should be especially noted that the best optima are only achieved by NCG.



(a) (b)

Figure 9: Comparison of algorithms using Dataset 2. Figure (a) shows the median learning curves while Figure (b) shows the results of the individual initializations.



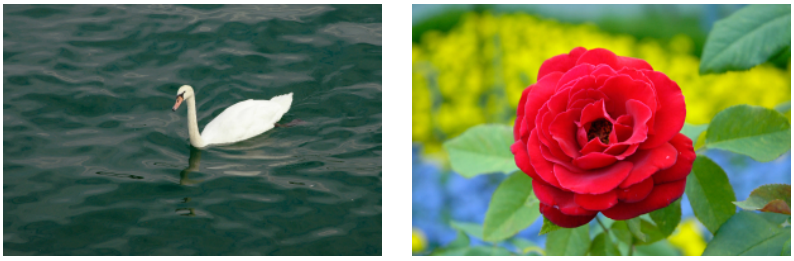
(a) (b)

Figure 10: Comparison of algorithms using Dataset 3. Figure (a) shows the median learning curves while Figure (b) shows the results of the individual initializations.

5.3 Image Data

In order to be able to compare the different algorithms using real-world data the algorithms were applied to the task of image segmentation. The goal is to divide a digital image into meaningful regions so that some characteristic of the image significantly changes on the boundary of two regions. This can be achieved, for example, by fitting a mixture of Gaussians to the image data which is interpreted so that each pixel of the image is a five dimensional data point with three color coordinates and two spatial coordinates. Segmented regions of the image can then be constructed using the responsibilities as a measure of which region each pixel should be part of. It should be noted that in the variational approach the number of regions does not have to be determined beforehand but is instead selected automatically during learning. It should also be emphasized that there exists many other algorithms more suitable for image segmentation than the ones used here. Image segmentation is used here simply in order to easily obtain real-world data where the results of the experiments are easily visualized.

The image datasets used are shown in Figure 11. Dataset 5 is a 100×66 pixel image of a swan where the desired outcome of the segmentation is clearly the separation of the swan and the background. Dataset 6 is a 200×133 pixel image of a rose where the desired outcome of the segmentation is not as clear especially when the background is considered.



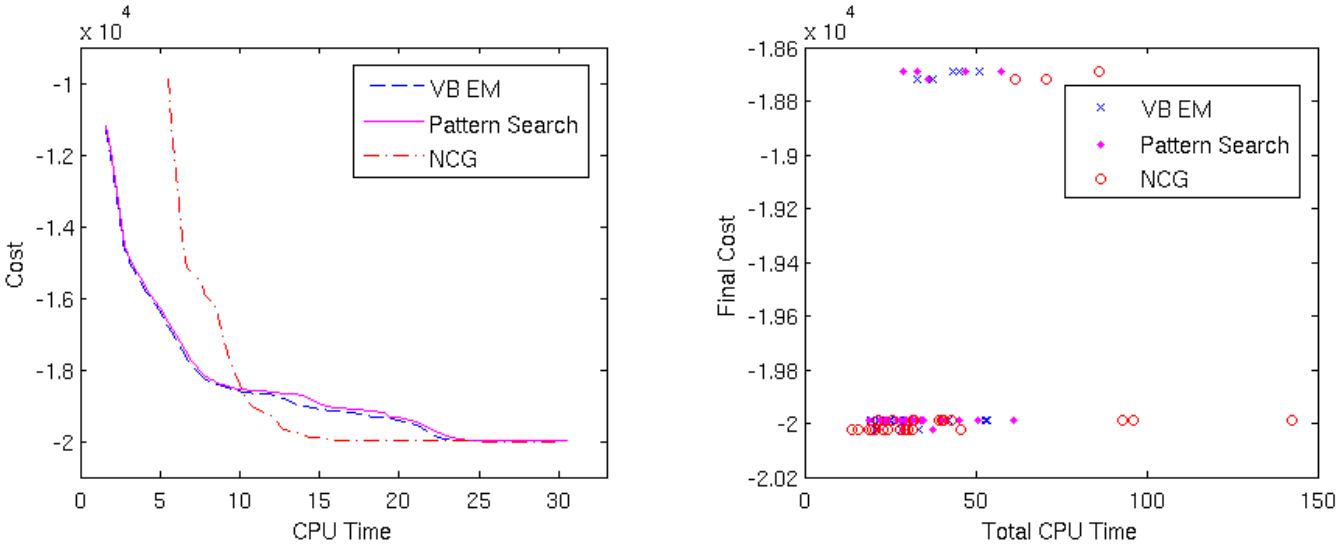
(a) Dataset 4 (b) Dataset 5

Figure 11: Image datasets used in the image segmentation task.

5.3.1 Dataset 4

Comparison of NCG, VB EM and VB EM with pattern searches using Dataset 4 produces results shown on Figure 12. From the median learning curves of Figure 12(a), it can be seen that the initial convergence of NCG is faster than with the other algorithms but nevertheless no big differences in the convergence speed to the final optimum are observed. Examination of Figure 12(b), however, reveals an interesting phenomenon. The best optimum is reached with 16 NCG runs out of a total of 30 while both in the case of VB EM and pattern search only 3 runs reached the best optimum. Although the difference in the best cost function value of -2.002 and the second best value of -1.999 is small, there is a big difference in the outcome

of the segmentation task. This can be seen by plotting the responsibilities of each Gaussian component in the final solution. This is done in Figure 13 where white represents responsibility 1 and black responsibility 0. It can be seen that the best optimum represents the desired solution of segmentation into two regions while the second best optimum represents a situation where the image is segmented into three regions.



(a) (b)

Figure 12: Comparison of algorithms using Dataset 4. Figure (a) shows the median learning curves while Figure (b) shows the results of the individual initializations.

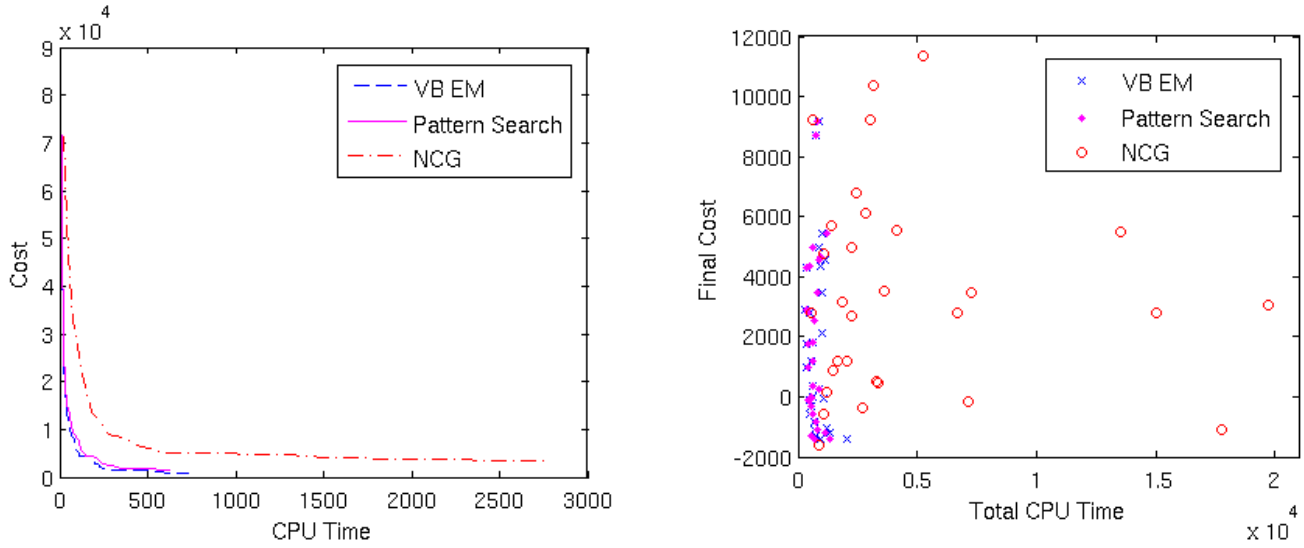
(a) Best, region 1 (b) Best, region 2

(c) 2nd best, region 1 (d) 2nd best, region 2 (e) 2nd best, region 3

Figure 13: Visual comparison of the best and the second best optimum found in the case of Dataset 4. It can be seen that the second best optimum represents a situation where the image is divided into three regions instead of the desired two regions.

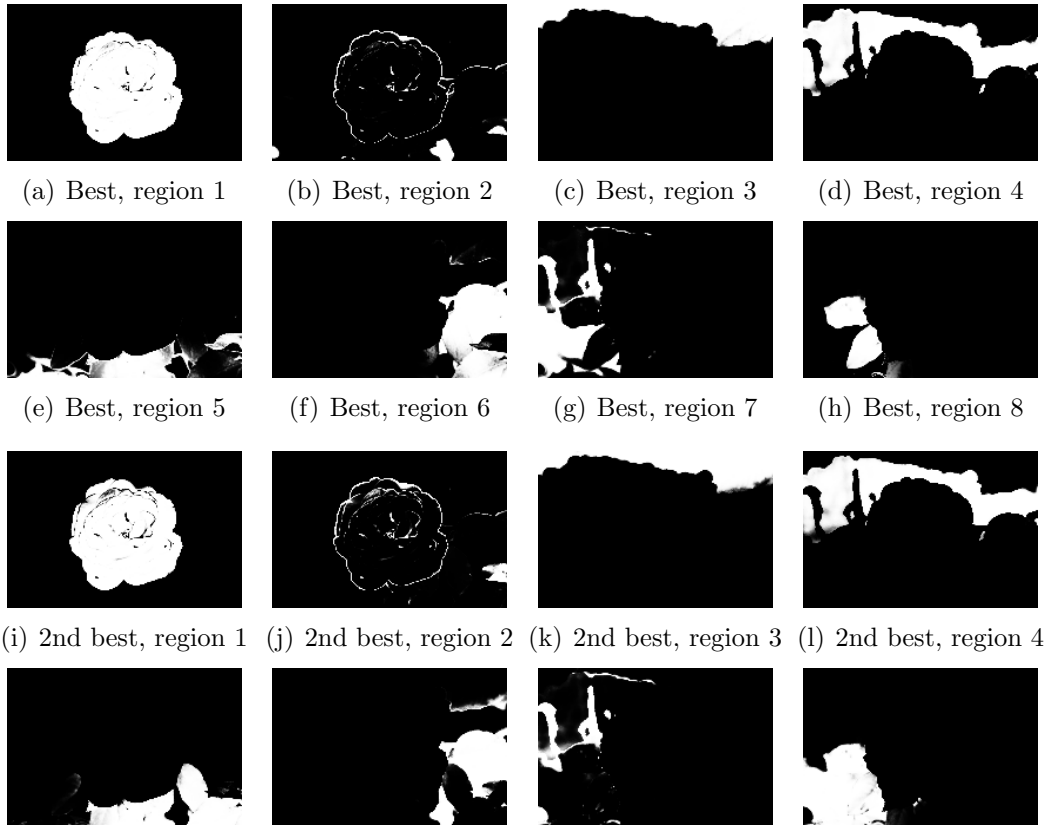
5.3.2 Dataset 5

Results with Dataset 5 are shown in Figure 14. By looking at the learning curves of Figure 14(a), it is clear that the median performance of VB EM and VB EM with pattern searches is superior to NCG. On the other hand, Figure 14(b) reveals that the best optimum of -1600 is only reached by a single NCG run while the best cost function value reached by VB EM is -1400. The difference of these optima is visualized in Figure 15 where it can be seen that the main difference is that NCG has been able to distinguish the leaves to the left of the rose from the background. Comparison of these optima is, however, much more difficult than in the case of the previous dataset. It should also be noted that, unlike Dataset 4, there is a large amount of different optima that the algorithms can converge into. This is a result of the image of Dataset 5 being, as far as segmentation is concerned, much more complex than the image of Dataset 4.



(a) (b)

Figure 14: Comparison of algorithms using Dataset 5. Figure (a) shows the median learning curves while Figure (b) shows the results of the individual initializations.



(a) Best, region 1 (b) Best, region 2 (c) Best, region 3 (d) Best, region 4

(e) Best, region 5 (f) Best, region 6 (g) Best, region 7 (h) Best, region 8

(i) 2nd best, region 1 (j) 2nd best, region 2 (k) 2nd best, region 3 (l) 2nd best, region 4

(m) 2nd best, region 5 (n) 2nd best, region 6 (o) 2nd best, region 7 (p) 2nd best, region 8

Figure 15: Visual comparison of the best and the second best optimum found in the case of Dataset 5. It can be seen that the main difference of the optima is that in the best optimum the leaves to the left of the rose in Figure (h) have been segmented from the background.

6 Discussion

By looking at the experimental results of Section 5, it can be said that NCG clearly outperforms other gradient-based algorithms. This is in keeping with the previous results using NCG [12]. It can also be said that NCG is highly competitive against algorithms based on VB EM. While in many cases the median performance of NCG is fairly close to algorithms based on VB EM, it seems to generally find the best optimum more often than the other algorithms. With the current implementation, it is up to the dataset which algorithm performs the best.

Although in some cases the pattern search acceleration of VB EM provides only a subtle performance improvement, the results shown in Figure 7 suggest that the pattern searches approach might perform well with a variety of datasets where one of the other algorithms produces suboptimal results. Because of its easy derivation, implementation and potential for great performance improvement, it is a worthwhile way to improve the performance of existing VB EM implementations.

It might also be possible to further improve the performance of NCG. The CPU time required by the algorithm is mainly spent on two things: on the evaluation of the gradient and on performing the line search. Therefore, it would be worthwhile to study how various other line search methods compare against the quadratic search used here. It might also be possible to completely eliminate the need for a line search by using a scaled conjugate gradient algorithm [15] with natural gradient. Application of the natural gradient to other state-of-the-art nonlinear optimization methods might also be a possible subject of future study. Another interesting unexplored question is how the NCG algorithm would perform if line search was conducted along a geodesic of the Riemannian manifold.

Further research is also required on determining exactly what kind of data is most suitable for the NCG algorithm. It is known that EM-like algorithms are prone to slow convergence in situations where the inference of latent variables is difficult, such as in the case of the cluster data of Dataset 1 when the clusters are overlapping [22]. Based on the results shown in Figure 7, it can be said that NCG seems to be superior to VB EM in such cases. Interestingly, the pattern search method seems to also remarkably improve the performance of VB EM with such data. Additionally, the results shown in Figure 8 suggest that when the number of data points is large it might be beneficial to use NCG although further experiments with other datasets are needed to verify this.

7 Conclusions

The aim of this thesis was to gain knowledge on the performance of the natural conjugate gradient algorithm on learning models in the conjugate-exponential family. When compared to the standard algorithm for performing variational inference in this family of models, the VB EM algorithm, we acquired experimental data which suggests that the NCG algorithm is highly competitive with VB EM when used to learn the mixture of Gaussians model. Especially the quality of the optima found using NCG seems to outperform VB EM in some cases. It should also be emphasized that while being a fairly complex algorithm to derive and implement, NCG generalizes to a much broader family of models than VB EM.

On the question of which one of the compared algorithms is the best choice for variational learning of the mixture of Gaussians model, it is fairly easy to conclude that given its simplicity and good performance across a wide range of datasets VB EM accelerated with pattern searches is the best choice. However, with certain types of data, NCG does provide better performance than the other algorithms, especially when it comes to the quality of the optima.

References

[1] S. Amari. *Differential-Geometrical Methods in Statistics*, volume 28 of *Lecture Notes in Statistics*. Springer-Verlag, 1985.

[2] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 1998.

[3] D. Barber and C. Bishop. Ensemble learning in Bayesian neural networks. In C. Bishop, editor, *Neural Networks and Machine Learning*, pages 215–237. Springer, Berlin, 1998.

[4] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. Wiley, 1994.

[5] C. M. Bishop. *Pattern recognition and machine learning*. Springer Science+Business Media, LLC., 2006.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory, 2nd edition*. Wiley, 2006.

[7] R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

[8] R. P. Feynman. *Statistical Mechanics*. W. A. Benjamin, Inc., 1972.

[9] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 1964.

[10] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC Press, 1995.

[11] G. E. Hinton and D. van Camp. Keeping neural network simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, Santa Cruz, CA, USA, 1993.

[12] A. Honkela, M. Tornio, T. Raiko, and J. Karhunen. Natural conjugate gradient in variational inference. In *The Proceedings of the 14th International Conference on Neural Information Processing (ICONIP 2007)*, pages 305–314, Kitakyushu, Japan, November 2007.

[13] A. Honkela, H. Valpola, and J. Karhunen. Accelerating cyclic update algorithms for parameter estimation by pattern searches. *Neural Processing Letters*, 17(2):191–203, April 2003.

[14] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[15] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[16] M. K. Murray and J. W. Rice. *Differential Geometry and Statistics*. Chapman & Hall, 1993.

[17] I. Nabney and C. Bishop. Netlab toolbox. http://www.ncrg.aston.ac.uk/netlab/index.php, June 2004. Version 3.3.1.

[18] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1991.

[19] G. Parisi. *Statistical Field Theory*. Addison-Wesley, 1988.

[20] E. Polak and G. Ribiére. Note sur la convergence de méthodes de directions conjugées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 1969.

[21] M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 1977.

[22] R. Salakhutdinov, S. Roweis, and Z. Ghahramani. Expectation-conjugate gradient: An alternative to EM. *IEEE Signal Processing Letters*, 11(7), 2004.

[23] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, 1994.

[24] S. T. Smith. *Geometric Optimization Methods for Adaptive Filtering*. PhD thesis, Harvard University, Cambridge, Massachusetts, 1993.

[25] M. Tornio. Natural gradient for variational Bayesian learning. Master's thesis, Helsinki University of Technology, Espoo, Finland, 2007. To appear.

[26] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 2002.

Appendix A: Probability Distributions

Dirichlet Distribution

The Dirichlet distribution is a multidimensional probability distribution whose probability density function is given by [5]

$$\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}) = C(\boldsymbol{\alpha}) \prod_{k=1}^{K} \pi_k^{\alpha_k - 1} \tag{A1}$$

where $\boldsymbol{\pi} = [\pi_1 \cdots \pi_K]^T$ and $\boldsymbol{\alpha} = [\alpha_1 \cdots \alpha_K]^T$. Here the normalizing constant is given by

$$C(\boldsymbol{\alpha}) = \frac{\Gamma(\hat{\alpha})}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \tag{A2}$$

where $\Gamma(\cdot)$ is the gamma function and

$$\hat{\alpha} = \sum_{k=1}^{K} \alpha_k. \tag{A3}$$

The continuous random variable $\boldsymbol{\pi}$ of the Dirichlet distribution must satisfy the constraints

$$0 \leq \pi_k \leq 1, k = 1 \ldots K \tag{A4}$$

and

$$\sum_{k=1}^{K} \pi_k = 1, k = 1 \ldots K. \tag{A5}$$

In order for the Dirichlet distribution to be normalized, the parameters α_k must also satisfy the condition $\alpha_k > 0$. For Dirichlet distribution, it holds that $E\{\pi_k\} = \alpha_k/\hat{\alpha}$.

Gaussian Distribution

The probability density function of a one dimensional Gaussian distribution is given by the famous bell curve

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{A6}$$

where the distribution is parametrized by the mean μ and the variance $\sigma^2 > 0$.

The Gaussian distribution can be generalized to the case of multidimensional random variables as follows [5]:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right) \tag{A7}$$

where $\mathbf{x}$ is a D dimensional continuous random variable and the distribution is parametrized by the mean $\boldsymbol{\mu}$ and the symmetric, positive definite covariance matrix $\boldsymbol{\Sigma}$. The inverse of the covariance matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is called the precision matrix which is also symmetric and positive definite. For the multidimensional Gaussian distribution, it holds that $E\{\mathbf{x}\} = \boldsymbol{\mu}$ as one would expect by analogy to the one dimensional case.

Wishart Distribution

The Wishart distribution arises in Bayesian inference as the conjugate prior of the precision matrix of the multidimensional Gaussian distribution. Its probability density function is given by [5]

$$\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W},\nu) = B(\mathbf{W},\nu)|\boldsymbol{\Lambda}|^{(\nu-D-1)/2}\exp\left(-\frac{1}{2}\operatorname{Tr}(\mathbf{W}^{-1}\boldsymbol{\Lambda})\right) \tag{A8}$$

where

$$B(\mathbf{W},\nu) = |\mathbf{W}|^{-\nu/2}\left(2^{\nu D/2}\pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma\left(\frac{\nu+1-i}{2}\right)\right)^{-1}. \tag{A9}$$

The Wishart distribution is parametrized by a symmetric, positive definite matrix $\mathbf{W}$ of order $D \times D$ and the degrees of freedom parameter $\nu > D-1$. The following results apply to the expectation and entropy of the Wishart distribution:

$$E\{\boldsymbol{\Lambda}\} = \nu\mathbf{W} \tag{A10}$$

$$E\{\ln|\boldsymbol{\Lambda}|\} = \sum_{i=1}^{D}\psi\left(\frac{\nu+1-i}{2}\right) + D\ln 2 + \ln|\mathbf{W}| \tag{A11}$$

$$H\{\boldsymbol{\Lambda}\} = -\ln B(\mathbf{W},\nu) - \frac{\nu-D-1}{2}E\{\ln|\boldsymbol{\Lambda}|\} + \frac{\nu D}{2} \tag{A12}$$

where $\psi(\cdot)$ denotes the digamma function defined by Equation (50).

Appendix B: Derivation of Equations for Section 4.4

In this appendix, we will derive Equations (71) and (72) which are needed for the evaluation of the gradient $\nabla_{\mathbf{m},\boldsymbol{\gamma}}\mathcal{C}$ of the cost function (60) with respect to the means $\mathbf{m}_k$ and the parameters γ_{nk}. We will also derive the Riemannian metric tensor given in Equation (77). We will be using the cyclic property of traces

$$\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{CAB}) = \mathrm{Tr}(\mathbf{BCA}) \tag{B1}$$

as well as the chain rule for several variables

$$f(\mathbf{x}(t)) \Rightarrow \frac{\partial f}{\partial t} = (\nabla_{\mathbf{x}} f)^T \frac{\partial \mathbf{x}}{\partial t}. \tag{B2}$$

We will also need the following lemma.

Lemma 1. *Let* $\mathbf{x}$ *and* $\mathbf{b}$ *be* D *dimensional column vectors and* $\mathbf{C}$ *a square matrix of the order* $D \times D$*. Then it holds that*

$$\nabla_{\mathbf{x}}(\mathbf{x}+\mathbf{b})^T\mathbf{C}(\mathbf{x}+\mathbf{b}) = (\mathbf{C}+\mathbf{C}^T)(\mathbf{x}+\mathbf{b}). \tag{B3}$$

Proof. For the quadratic form $(\mathbf{x}+\mathbf{b})^T\mathbf{C}(\mathbf{x}+\mathbf{b})$ it holds that

$$\begin{aligned}(\mathbf{x}+\mathbf{b})^T\mathbf{C}(\mathbf{x}+\mathbf{b}) = &\sum_{m=1}^{D}\sum_{n=1}^{D} C_{mn}(\mathbf{x}+\mathbf{b})_m(\mathbf{x}+\mathbf{b})_n = \\ &\sum_{m=1}^{D}\sum_{n=1}^{D} C_{mn}(x_m+b_m)(x_n+b_n).\end{aligned}$$

Thus, the derivative with respect to variable x_i is

$$\begin{aligned}&\frac{\partial}{\partial x_i}(\mathbf{x}+\mathbf{b})^T\mathbf{C}(\mathbf{x}+\mathbf{b}) \\ =&\frac{\partial}{\partial x_i}\sum_{m=1}^{D}\sum_{n=1}^{D} C_{mn}(x_m+b_m)(x_n+b_n) \\ =&\frac{\partial}{\partial x_i}\sum_{n=1,n\neq i}^{D} C_{in}(x_i+b_i)(x_n+b_n) + \frac{\partial}{\partial x_i}\sum_{m=1,m\neq i}^{D} C_{mi}(x_m+b_m)(x_i+b_i) \\ &+\frac{\partial}{\partial x_i}C_{ii}(x_i+b_i)(x_i+b_i) \\ =&\sum_{n=1,n\neq i}^{D} C_{in}(x_n+b_n) + \sum_{m=1,m\neq i}^{D} C_{mi}(x_m+b_m) + 2C_{ii}(x_i+b_i) \\ =&\sum_{n=1}^{D} C_{in}(x_n+b_n) + \sum_{m=1}^{D} C_{mi}(x_m+b_m)\end{aligned}$$

$$
\begin{aligned}
&= \sum_{n=1}^{D} (C_{in} + C_{ni})(x_n + b_n) \\
&= \sum_{n=1}^{D} (C_{in} + C_{in}^T)(x_n + b_n) \\
&= \sum_{n=1}^{D} (\mathbf{C} + \mathbf{C}^T)_{in}(\mathbf{x} + \mathbf{b})_n.
\end{aligned}
$$

It follows that

$$
\nabla_{\mathbf{x}} (\mathbf{x} + \mathbf{b})^T \mathbf{C} (\mathbf{x} + \mathbf{b}) = \begin{bmatrix} \partial_{x_1} \\ \vdots \\ \partial_{x_i} \\ \vdots \\ \partial_{x_D} \end{bmatrix} (\mathbf{x} + \mathbf{b})^T \mathbf{C} (\mathbf{x} + \mathbf{b}) = (\mathbf{C} + \mathbf{C}^T)(\mathbf{x} + \mathbf{b}).
$$

□

It immediately follows from Equation (B3) that if $\mathbf{C}$ is symmetric, it holds that

$$
\nabla_{\mathbf{x}} (\mathbf{x} + \mathbf{b})^T \mathbf{C} (\mathbf{x} + \mathbf{b}) = 2\mathbf{C}(\mathbf{x} + \mathbf{b}). \tag{B4}
$$

Derivation of $\nabla_{\mathbf{m}_k} \mathcal{C}$

Let us first derive Equation (71). Consider the terms of the cost function (60) which depend on parameter $\mathbf{m}_k$

$$
\mathcal{C}_{\mathbf{m}_k} = \frac{1}{2} N_k \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) + \frac{1}{2} \beta_0 \nu_k (\mathbf{m}_k - \mathbf{m}_0)^T \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0).
$$

As a parameter of the Wishart distribution, the matrix $\mathbf{W}_k$ is symmetric and thus

$$
\begin{aligned}
&\nabla_{\mathbf{m}_k} \mathcal{C} \\
=&\nabla_{\mathbf{m}_k} \mathcal{C}_{\mathbf{m}_k} \\
=&\frac{1}{2} N_k \nu_k \nabla_{\mathbf{m}_k} (\mathbf{m}_k - \overline{\mathbf{x}}_k)^T \mathbf{W}_k (\mathbf{m}_k - \overline{\mathbf{x}}_k) + \frac{1}{2} \beta_0 \nu_k \nabla_{\mathbf{m}_k} (\mathbf{m}_k - \mathbf{m}_0)^T \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0) \\
=&N_k \nu_k \mathbf{W}_k (\mathbf{m}_k - \overline{\mathbf{x}}_k) + \beta_0 \nu_k \mathbf{W}_k (\mathbf{m}_k - \mathbf{m}_0) \\
=&\nu_k \mathbf{W}_k (N_k (\mathbf{m}_k - \overline{\mathbf{x}}_k) + \beta_0 (\mathbf{m}_k - \mathbf{m}_0))
\end{aligned}
$$

which gives us Equation (71).

Derivation of $\partial_{\gamma_{ij}} \mathcal{C}$

Derivation of Equation (72) is a more difficult task. Let us first consider the terms of the cost function (60) that have dependence on the softmax parameters $\gamma_{nk}, n =$

$1 \dots N, k = 1 \dots K$

$$\begin{aligned}\mathcal{C}_\gamma =& \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \ln r_{nk} - \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} \ln \tilde{\pi}_k - \frac{1}{2}\sum_{k=1}^{K} N_k \Big\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) \\ & - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \Big\} \\ =& \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} (\ln r_{nk} - \ln \tilde{\pi}_k) - \frac{1}{2}\sum_{k=1}^{K} N_k \Big\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) \\ & - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \Big\}.\end{aligned}$$

It should be emphasized that due to Equations (45), (46) and (47) also N_k, $\overline{\mathbf{x}}_k$ and $\mathbf{S}_k$ depend on γ_{nk}. Now, let us make the following definitions:

$$\begin{aligned}\mathcal{C}_{\gamma_1} =& \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk} (\ln r_{nk} - \ln \tilde{\pi}_k) \\ \mathcal{C}_{\gamma_2} =& \sum_{k=1}^{K} N_k \left\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \right\}.\end{aligned}$$

With this notation, Equation (72) is given by

$$\frac{\partial \mathcal{C}}{\partial \gamma_{ij}} = \frac{\partial \mathcal{C}_\gamma}{\partial \gamma_{ij}} = \frac{\partial \mathcal{C}_{\gamma_1}}{\partial \gamma_{ij}} - \frac{1}{2}\frac{\partial \mathcal{C}_{\gamma_2}}{\partial \gamma_{ij}}. \tag{B5}$$

Let us first derive an expression for $\partial_{\gamma_{ij}} \mathcal{C}_{\gamma_1}$. By using the definition of the softmax parametrization (68), we get

$$\begin{aligned}&\frac{\partial \mathcal{C}_{\gamma_1}}{\partial \gamma_{ij}} \\ =& \frac{\partial}{\partial \gamma_{ij}} \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} \left(\gamma_{nk} - \ln \sum_{l=1}^{K} e^{\gamma_{nl}} - \ln \tilde{\pi}_k \right) \\ =& \frac{\partial}{\partial \gamma_{ij}} \sum_{k=1}^{K} \frac{e^{\gamma_{ik}}}{\sum_{l=1}^{K} e^{\gamma_{il}}} \left(\gamma_{ik} - \ln \sum_{l=1}^{K} e^{\gamma_{il}} - \ln \tilde{\pi}_k \right) \\ =& \frac{\partial}{\partial \gamma_{ij}} \frac{1}{\sum_{l=1}^{K} e^{\gamma_{il}}} \sum_{k=1}^{K} e^{\gamma_{ik}} \left(\gamma_{ik} - \ln \sum_{l=1}^{K} e^{\gamma_{il}} - \ln \tilde{\pi}_k \right) \\ =& \left(\frac{\partial}{\partial \gamma_{ij}} \frac{1}{\sum_{l=1}^{K} e^{\gamma_{il}}} \right) \sum_{k=1}^{K} e^{\gamma_{ik}} \left(\gamma_{ik} - \ln \sum_{l=1}^{K} e^{\gamma_{il}} - \ln \tilde{\pi}_k \right) \\ & + \frac{1}{\sum_{l=1}^{K} e^{\gamma_{il}}} \frac{\partial}{\partial \gamma_{ij}} \sum_{k=1}^{K} e^{\gamma_{ik}} \left(\gamma_{ik} - \ln \sum_{l=1}^{K} e^{\gamma_{il}} - \ln \tilde{\pi}_k \right).\end{aligned}$$

Here we can evaluate

$$\frac{\partial}{\partial\gamma_{ij}}\frac{1}{\sum_{l=1}^{K}e^{\gamma_{il}}}=-\frac{e^{\gamma_{ij}}}{(\sum_{l=1}^{K}e^{\gamma_{il}})^2}$$

and

$$\begin{aligned}
&\frac{\partial}{\partial\gamma_{ij}}\sum_{k=1}^{K}e^{\gamma_{ik}}\left(\gamma_{ik}-\ln\sum_{l=1}^{K}e^{\gamma_{il}}-\ln\tilde{\pi}_k\right)\\
=&\frac{\partial}{\partial\gamma_{ij}}\left\{\sum_{k=1}^{K}e^{\gamma_{ik}}\gamma_{ik}-\sum_{k=1}^{K}\left(e^{\gamma_{ik}}\ln\sum_{l=1}^{K}e^{\gamma_{il}}\right)-\sum_{k=1}^{K}e^{\gamma_{ik}}\ln\tilde{\pi}_k\right\}\\
=&\frac{\partial}{\partial\gamma_{ij}}e^{\gamma_{ij}}\gamma_{ij}-\frac{\partial}{\partial\gamma_{ij}}\left(\sum_{k=1}^{K}e^{\gamma_{ik}}\ln\sum_{l=1}^{K}e^{\gamma_{il}}\right)-e^{\gamma_{ij}}\ln\tilde{\pi}_j\\
=&e^{\gamma_{ij}}+e^{\gamma_{ij}}\gamma_{ij}-\left(\frac{\partial}{\partial\gamma_{ij}}\sum_{k=1}^{K}e^{\gamma_{ik}}\right)\ln\sum_{l=1}^{K}e^{\gamma_{il}}-\sum_{k=1}^{K}e^{\gamma_{ik}}\frac{\partial}{\partial\gamma_{ij}}\ln\sum_{l=1}^{K}e^{\gamma_{il}}-e^{\gamma_{ij}}\ln\tilde{\pi}_j\\
=&e^{\gamma_{ij}}+e^{\gamma_{ij}}\gamma_{ij}-e^{\gamma_{ij}}\ln\sum_{l=1}^{K}e^{\gamma_{il}}-\sum_{k=1}^{K}e^{\gamma_{ik}}\frac{1}{\sum_{l=1}^{K}e^{\gamma_{il}}}e^{\gamma_{ij}}-e^{\gamma_{ij}}\ln\tilde{\pi}_j\\
=&e^{\gamma_{ij}}\left(\gamma_{ij}-\ln\sum_{l=1}^{K}e^{\gamma_{il}}-\ln\tilde{\pi}_j\right).
\end{aligned}$$

We can now write

$$\begin{aligned}
&\frac{\partial\mathcal{C}_{\gamma_1}}{\partial\gamma_{ij}}\\
=&-\frac{e^{\gamma_{ij}}}{(\sum_{l=1}^{K}e^{\gamma_{il}})^2}\sum_{k=1}^{K}e^{\gamma_{ik}}\left(\gamma_{ik}-\ln\sum_{l=1}^{K}e^{\gamma_{il}}-\ln\tilde{\pi}_k\right)+\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\left(\gamma_{ij}-\ln\sum_{l=1}^{K}e^{\gamma_{il}}-\ln\tilde{\pi}_j\right)\\
=&-\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\sum_{k=1}^{K}\frac{e^{\gamma_{ik}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\left(\ln\frac{e^{\gamma_{ik}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}-\ln\tilde{\pi}_k\right)+\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\left(\ln\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}-\ln\tilde{\pi}_j\right)\\
=&-r_{ij}\sum_{k=1}^{K}r_{ik}(\ln r_{ik}-\ln\tilde{\pi}_k)+r_{ij}(\ln r_{ij}-\ln\tilde{\pi}_j).
\end{aligned}\tag{B6}$$

Let us now consider the derivative

$$\begin{aligned}
&\frac{\partial\mathcal{C}_{\gamma_2}}{\partial\gamma_{ij}}\\
=&\frac{\partial}{\partial\gamma_{ij}}N_j\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\,\mathrm{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\
&+\frac{\partial}{\partial\gamma_{ij}}\sum_{k=1,k\neq j}^{K}N_k\left\{\ln\tilde{\Lambda}_k-\frac{D}{\beta_k}-\nu_k\,\mathrm{Tr}(\mathbf{S}_k\mathbf{W}_k)-\nu_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)-D\ln 2\pi\right\}.
\end{aligned}\tag{B7}$$

We will first derive an expression for the first one of these two terms

$$\begin{aligned}&\frac{\partial}{\partial\gamma_{ij}}N_j\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\operatorname{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\&=\left(\frac{\partial}{\partial\gamma_{ij}}N_j\right)\{\dots\}+N_j\frac{\partial}{\partial\gamma_{ij}}\{\dots\}.\end{aligned}\tag{*}$$

Using Equation (45) we can write

$$\begin{aligned}&\frac{\partial}{\partial\gamma_{ij}}N_j\\=&\frac{\partial}{\partial\gamma_{ij}}\sum_{n=1}^{N}r_{nj}\\=&\frac{\partial}{\partial\gamma_{ij}}\sum_{n=1}^{N}\frac{e^{\gamma_{nj}}}{\sum_{l=1}^{K}e^{\gamma_{nl}}}\\=&\frac{\partial}{\partial\gamma_{ij}}\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\\=&\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}-\left(\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\right)^2\\=&r_{ij}-r_{ij}^2.\end{aligned}\tag{*}$$

We can further write

$$\begin{aligned}&\frac{\partial}{\partial\gamma_{ij}}\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\operatorname{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\=&-\nu_j\frac{\partial}{\partial\gamma_{ij}}\operatorname{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j\frac{\partial}{\partial\gamma_{ij}}(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j).\end{aligned}\tag{*}$$

Here we can write using the symmetry of the matrix $\mathbf{W}_j$ as well as Equations (B2) and (B4)

$$\begin{aligned}&\frac{\partial}{\partial\gamma_{ij}}(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)\\=&\left(\nabla_{\overline{\mathbf{x}}_j}(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)\right)^T\frac{\partial\overline{\mathbf{x}}_j}{\partial\gamma_{ij}}\\=&(2\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j))^T\frac{\partial\overline{\mathbf{x}}_j}{\partial\gamma_{ij}}\end{aligned}$$

$$=2(\overline{\mathbf{x}}_j - \mathbf{m}_j)^T \mathbf{W}_j \frac{\partial \overline{\mathbf{x}}_j}{\partial \gamma_{ij}}.$$

Using Equation (46), the derivative of $\overline{\mathbf{x}}_j$ is given by

$$\begin{aligned}
&\frac{\partial \overline{\mathbf{x}}_j}{\partial \gamma_{ij}} \\
=&\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_j} \sum_{n=1}^{N} r_{nj} \mathbf{x}_n \\
=&\left(\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_j} \right) \sum_{n=1}^{N} r_{nj} \mathbf{x}_n + \frac{1}{N_j} \frac{\partial}{\partial \gamma_{ij}} \frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K} e^{\gamma_{il}}} \mathbf{x}_i \\
=&-\frac{r_{ij} - r_{ij}^2}{N_j^2} \sum_{n=1}^{N} r_{nj} \mathbf{x}_n + \frac{r_{ij} - r_{ij}^2}{N_j} \mathbf{x}_i \\
=&\frac{r_{ij} - r_{ij}^2}{N_j} (\mathbf{x}_i - \overline{\mathbf{x}}_j).
\end{aligned}$$

Combining these results we get

$$\frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_j - \mathbf{m}_j)^T \mathbf{W}_j (\overline{\mathbf{x}}_j - \mathbf{m}_j) = -2 \frac{r_{ij} - r_{ij}^2}{N_j} (\overline{\mathbf{x}}_j - \mathbf{m}_j)^T \mathbf{W}_j (\overline{\mathbf{x}}_j - \mathbf{x}_i). \qquad (*)$$

Returning to the derivative $\partial_{\gamma_{ij}} \operatorname{Tr}(\mathbf{S}_j \mathbf{W}_j)$, we can use the linearity of the trace along with Equations (47) and (B1) to write

$$\begin{aligned}
&\frac{\partial}{\partial \gamma_{ij}} \operatorname{Tr}(\mathbf{S}_j \mathbf{W}_j) \\
=&\frac{\partial}{\partial \gamma_{ij}} \operatorname{Tr} \left(\frac{1}{N_j} \sum_{n=1}^{N} r_{nj} (\mathbf{x}_n - \overline{\mathbf{x}}_j)(\mathbf{x}_n - \overline{\mathbf{x}}_j)^T \mathbf{W}_j \right) \\
=&\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_j} \sum_{n=1}^{N} r_{nj} \operatorname{Tr}((\mathbf{x}_n - \overline{\mathbf{x}}_j)(\mathbf{x}_n - \overline{\mathbf{x}}_j)^T \mathbf{W}_j) \\
=&\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_j} \sum_{n=1}^{N} r_{nj} \operatorname{Tr}((\mathbf{x}_n - \overline{\mathbf{x}}_j)^T \mathbf{W}_j (\mathbf{x}_n - \overline{\mathbf{x}}_j)) \\
=&\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_j} \sum_{n=1}^{N} r_{nj} (\mathbf{x}_n - \overline{\mathbf{x}}_j)^T \mathbf{W}_j (\mathbf{x}_n - \overline{\mathbf{x}}_j) \\
=&-\frac{r_{ij} - r_{ij}^2}{N_j^2} \sum_{n=1}^{N} r_{nj} (\overline{\mathbf{x}}_j - \mathbf{x}_n)^T \mathbf{W}_j (\overline{\mathbf{x}}_j - \mathbf{x}_n) \\
&+ \frac{1}{N_j} \frac{\partial}{\partial \gamma_{ij}} \sum_{n=1}^{N} r_{nj} (\overline{\mathbf{x}}_j - \mathbf{x}_n)^T \mathbf{W}_j (\overline{\mathbf{x}}_j - \mathbf{x}_n). \qquad (*)
\end{aligned}$$

The derivative here is given by

$$
\begin{aligned}
&\frac{\partial}{\partial\gamma_{ij}}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)\\
=&\frac{\partial}{\partial\gamma_{ij}} r_{ij}(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)+\frac{\partial}{\partial\gamma_{ij}}\sum_{n=1,n\neq i}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)\\
=&\left(\frac{\partial}{\partial\gamma_{ij}} r_{ij}\right)(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)+r_{ij}\frac{\partial}{\partial\gamma_{ij}}(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
&+\sum_{n=1,n\neq i}^{N} r_{nj}\frac{\partial}{\partial\gamma_{ij}}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)\\
=&\left(\frac{\partial}{\partial\gamma_{ij}}\frac{e^{\gamma_{ij}}}{\sum_{l=1}^{K}e^{\gamma_{il}}}\right)(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)+\sum_{n=1}^{N} r_{nj}\frac{\partial}{\partial\gamma_{ij}}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)\\
=&(r_{ij}-r_{ij}^2)(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)-2\frac{r_{ij}-r_{ij}^2}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i). \quad (*)
\end{aligned}
$$

Combining the results marked with (*), we get

$$
\begin{aligned}
&\frac{\partial}{\partial\gamma_{ij}}N_j\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\,\mathrm{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\
=&(r_{ij}-r_{ij}^2)\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\,\mathrm{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\
&-\nu_j N_j\Bigg\{-\frac{r_{ij}-r_{ij}^2}{N_j^2}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)+\frac{r_{ij}-r_{ij}^2}{N_j}(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
&-2\frac{r_{ij}-r_{ij}^2}{N_j^2}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)-2\frac{r_{ij}-r_{ij}^2}{N_j}(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\Bigg\}\\
=&(r_{ij}-r_{ij}^2)\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\,\mathrm{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\
&+(-\nu_j r_{ij}+\nu_j r_{ij}^2)\Bigg\{(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)-2(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
&-\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)-2\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\Bigg\}.
\end{aligned}
\tag{B8}
$$

We will next consider the second term of Equation (B7). Keeping in mind that $k\neq j$, we can write

$$
\frac{\partial}{\partial\gamma_{ij}}\sum_{k=1,k\neq j}^{K} N_k\left\{\ln\tilde{\Lambda}_k-\frac{D}{\beta_k}-\nu_k\,\mathrm{Tr}(\mathbf{S}_k\mathbf{W}_k)-\nu_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)-D\ln 2\pi\right\}
$$

$$= \sum_{k=1,k\neq j}^{K} \frac{\partial}{\partial \gamma_{ij}} N_k \{\dots\}$$
$$= \sum_{k=1,k\neq j}^{K} \left(\frac{\partial}{\partial \gamma_{ij}} N_k \right) \{\dots\} + \sum_{k=1,k\neq j}^{K} N_k \frac{\partial}{\partial \gamma_{ij}} \{\dots\}. \quad (*)$$

Equation (45) now gives us

$$\begin{aligned} &\frac{\partial}{\partial \gamma_{ij}} N_k \\ =&\frac{\partial}{\partial \gamma_{ij}} \sum_{n=1}^{N} \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} \\ =&\frac{\partial}{\partial \gamma_{ij}} \frac{e^{\gamma_{ik}}}{\sum_{l=1}^{K} e^{\gamma_{il}}} \\ =&-\frac{e^{\gamma_{ik}} e^{\gamma_{ij}}}{(\sum_{l=1}^{K} e^{\gamma_{il}})^2} \\ =&-r_{ij} r_{ik}. \end{aligned} \quad (*)$$

We can also write

$$\begin{aligned} &\frac{\partial}{\partial \gamma_{ij}} \left\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \right\} \\ =&-\nu_k \frac{\partial}{\partial \gamma_{ij}} \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k \frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k). \end{aligned} \quad (*)$$

Here the latter derivative is given by

$$\begin{aligned} &\frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) \\ =&2(\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k \frac{\partial \overline{\mathbf{x}}_k}{\partial \gamma_{ij}} \end{aligned}$$

where the derivative of $\overline{\mathbf{x}}_k$ is given by

$$\begin{aligned} &\frac{\partial \overline{\mathbf{x}}_k}{\partial \gamma_{ij}} \\ =&\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \\ =&\left(\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_k} \right) \sum_{n=1}^{N} r_{nk} \mathbf{x}_n + \frac{1}{N_k} \frac{\partial}{\partial \gamma_{ij}} \frac{e^{\gamma_{ik}}}{\sum_{l=1}^{K} e^{\gamma_{il}}} \mathbf{x}_i \\ =&\frac{r_{ij} r_{ik}}{N_k^2} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n - \frac{r_{ij} r_{ik}}{N_k} \mathbf{x}_i \end{aligned}$$

$$= \frac{r_{ij} r_{ik}}{N_k} (\overline{\mathbf{x}}_k - \mathbf{x}_i).$$

Thus, we can write

$$\frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) = 2 \frac{r_{ij} r_{ik}}{N_k} (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i). \qquad (*)$$

In accordance with our previous discussion, we can also write

$$\begin{aligned}
&\frac{\partial}{\partial \gamma_{ij}} \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) \\
=&\frac{\partial}{\partial \gamma_{ij}} \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n) \\
=&\frac{r_{ij} r_{ik}}{N_k^2} \sum_{n=1}^{N} r_{nk} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n) \\
&+ \frac{1}{N_k} \frac{\partial}{\partial \gamma_{ij}} \sum_{n=1}^{N} r_{nk} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n). \qquad (*)
\end{aligned}$$

The derivative here is given by

$$\begin{aligned}
&\frac{\partial}{\partial \gamma_{ij}} \sum_{n=1}^{N} r_{nk} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n) \\
=&\frac{\partial}{\partial \gamma_{ij}} r_{ik} (\overline{\mathbf{x}}_k - \mathbf{x}_i)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i) + \frac{\partial}{\partial \gamma_{ij}} \sum_{n=1, n \neq i}^{N} r_{nk} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n) \\
=&\left(\frac{\partial}{\partial \gamma_{ij}} \frac{e^{\gamma_{ik}}}{\sum_{l=1}^{K} e^{\gamma_{il}}} \right) (\overline{\mathbf{x}}_k - \mathbf{x}_i)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i) + r_{ik} \frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_k - \mathbf{x}_i)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i) \\
&+ \sum_{n=1, n \neq i}^{N} r_{nk} \frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n) \\
=&- r_{ij} r_{ik} (\overline{\mathbf{x}}_k - \mathbf{x}_i)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i) + \sum_{n=1}^{N} r_{nk} \frac{\partial}{\partial \gamma_{ij}} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_n) \\
=&- r_{ij} r_{ik} (\overline{\mathbf{x}}_k - \mathbf{x}_i)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i) + 2 \frac{r_{ij} r_{ik}}{N_k} \sum_{n=1}^{N} r_{nk} (\overline{\mathbf{x}}_k - \mathbf{x}_n)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{x}_i). \qquad (*)
\end{aligned}$$

Combining the results marked with (*) gives us

$$\begin{aligned}
&\frac{\partial}{\partial \gamma_{ij}} \sum_{k=1, k \neq j}^{K} N_k \left\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \right\} \\
=& \sum_{k=1, k \neq j}^{K} -r_{ij} r_{ik} \left\{ \ln \tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k \operatorname{Tr}(\mathbf{S}_k \mathbf{W}_k) - \nu_k (\overline{\mathbf{x}}_k - \mathbf{m}_k)^T \mathbf{W}_k (\overline{\mathbf{x}}_k - \mathbf{m}_k) - D \ln 2\pi \right\}
\end{aligned}$$

$$
\begin{aligned}
&+\sum_{k=1,k\neq j}^{K}-\nu_k N_k\Bigg\{\frac{r_{ij}r_{ik}}{N_k^2}\sum_{n=1}^{N}r_{nk}(\overline{\mathbf{x}}_k-\mathbf{x}_n)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_n)-\frac{r_{ij}r_{ik}}{N_k}(\overline{\mathbf{x}}_k-\mathbf{x}_i)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)\\
&+2\frac{r_{ij}r_{ik}}{N_k^2}\sum_{n=1}^{N}r_{nk}(\overline{\mathbf{x}}_k-\mathbf{x}_n)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)+2\frac{r_{ij}r_{ik}}{N_k}(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)\Bigg\}\\
=&\sum_{k=1,k\neq j}^{K}-r_{ij}r_{ik}\left\{\ln\tilde{\Lambda}_k-\frac{D}{\beta_k}-\nu_k\operatorname{Tr}(\mathbf{S}_k\mathbf{W}_k)-\nu_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)-D\ln 2\pi\right\}\\
&+\sum_{k=1,k\neq j}^{K}\nu_k r_{ij}r_{ik}\Bigg\{(\overline{\mathbf{x}}_k-\mathbf{x}_i)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)-2(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)\\
&-\frac{1}{N_k}\sum_{n=1}^{N}r_{nk}(\overline{\mathbf{x}}_k-\mathbf{x}_n)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_n)-2\frac{1}{N_k}\sum_{n=1}^{N}r_{nk}(\overline{\mathbf{x}}_k-\mathbf{x}_n)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)\Bigg\}.
\end{aligned}
\tag{B9}
$$

By combining the results (B7), (B8) and (B9), we get

$$
\begin{aligned}
&\frac{\partial\mathcal{C}_{\gamma_2}}{\partial\gamma_{ij}}\\
=&r_{ij}\left\{\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\operatorname{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi\right\}\\
&-\nu_j r_{ij}\Bigg\{(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)-2(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
&-\frac{1}{N_j}\sum_{n=1}^{N}r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)-2\frac{1}{N_j}\sum_{n=1}^{N}r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\Bigg\}\\
&-\sum_{k=1}^{K}r_{ij}r_{ik}\left\{\ln\tilde{\Lambda}_k-\frac{D}{\beta_k}-\nu_k\operatorname{Tr}(\mathbf{S}_k\mathbf{W}_k)-\nu_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)-D\ln 2\pi\right\}\\
&+\sum_{k=1}^{K}\nu_k r_{ij}r_{ik}\Bigg\{(\overline{\mathbf{x}}_k-\mathbf{x}_i)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)-2(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)\\
&-\frac{1}{N_k}\sum_{n=1}^{N}r_{nk}(\overline{\mathbf{x}}_k-\mathbf{x}_n)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_n)-2\frac{1}{N_k}\sum_{n=1}^{N}r_{nk}(\overline{\mathbf{x}}_k-\mathbf{x}_n)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{x}_i)\Bigg\}\\
=&r_{ij}A_j-\nu_j r_{ij}B_{ij}-\sum_{k=1}^{K}r_{ij}r_{ik}A_k+\sum_{k=1}^{K}\nu_k r_{ij}r_{ik}B_{ik}
\end{aligned}
\tag{B10}
$$

where we have defined

$$
A_j=\ln\tilde{\Lambda}_j-\frac{D}{\beta_j}-\nu_j\operatorname{Tr}(\mathbf{S}_j\mathbf{W}_j)-\nu_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{m}_j)-D\ln 2\pi
$$

and

$$
B_{ij}=(\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)-2(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)
$$

$$-\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n) - 2\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i).$$

Note that

$$\mathrm{Tr}(\mathbf{S}_j\mathbf{W}_j) = \frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_n)$$

and due to Equations (45) and (46)

$$\begin{aligned}
&\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
=&\left(\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)^T\right)\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
=&\left(\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}(\overline{\mathbf{x}}_j-\mathbf{x}_n)\right)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
=&\left(\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}\overline{\mathbf{x}}_j-\frac{1}{N_j}\sum_{n=1}^{N} r_{nj}\mathbf{x}_n\right)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
=&\left(\frac{1}{N_j}\overline{\mathbf{x}}_j N_j-\overline{\mathbf{x}}_j\right)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i)\\
=&0.
\end{aligned}$$

Thus, B_{ij} simplifies to

$$B_{ij} = (\overline{\mathbf{x}}_j-\mathbf{x}_i)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i) - 2(\overline{\mathbf{x}}_j-\mathbf{m}_j)^T\mathbf{W}_j(\overline{\mathbf{x}}_j-\mathbf{x}_i) - \mathrm{Tr}(\mathbf{S}_j\mathbf{W}_j).$$

We can now combine the results (B5), (B6) and (B10) to get

$$\begin{aligned}
&\frac{\partial\mathcal{C}}{\partial\gamma_{ij}}\\
=&r_{ij}\left(\sum_{k=1}^{K} -r_{ik}(\ln r_{ik}-\ln\tilde{\pi}_k) + (\ln r_{ij}-\ln\tilde{\pi}_j) - \frac{1}{2}(A_j-\nu_j B_{ij}) + \sum_{k=1}^{K}\frac{1}{2}r_{ik}(A_k-\nu_k B_{ik})\right)\\
=&r_{ij}\left(\ln r_{ij}-\ln\tilde{\pi}_j - \frac{1}{2}(A_j-\nu_j B_{ij}) - \sum_{k=1}^{K} r_{ik}\left\{\ln r_{ik}-\ln\tilde{\pi}_k - \frac{1}{2}(A_k-\nu_k B_{ik})\right\}\right).
\end{aligned}$$

Here we can write

$$\begin{aligned}
&\ln r_{ik}-\ln\tilde{\pi}_k - \frac{1}{2}(A_k-\nu_k B_{ik})\\
=&\ln r_{ik}-\ln\tilde{\pi}_k - \frac{1}{2}\Bigg(\ln\tilde{\Lambda}_k - \frac{D}{\beta_k} - \nu_k\,\mathrm{Tr}(\mathbf{S}_k\mathbf{W}_k) - \nu_k(\overline{\mathbf{x}}_k-\mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k-\mathbf{m}_k) - D\ln 2\pi
\end{aligned}$$

$$
\begin{aligned}
&- \nu_k(\overline{\mathbf{x}}_k - \mathbf{x}_i)^T\mathbf{W}_k(\overline{\mathbf{x}}_k - \mathbf{x}_i) + 2\nu_k(\overline{\mathbf{x}}_k - \mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k - \mathbf{x}_i) + \nu_k \operatorname{Tr}(\mathbf{S}_k\mathbf{W}_k)\Big) \\
=&\ln r_{ik} - \ln\tilde{\pi}_k - \frac{1}{2}\Big(\ln\tilde{\Lambda}_k - \frac{D}{\beta_k} - D\ln 2\pi \\
&- \nu_k((\overline{\mathbf{x}}_k - \mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k - \mathbf{m}_k) - 2(\overline{\mathbf{x}}_k - \mathbf{m}_k)^T\mathbf{W}_k(\overline{\mathbf{x}}_k - \mathbf{x}_i) + (\overline{\mathbf{x}}_k - \mathbf{x}_i)^T\mathbf{W}_k(\overline{\mathbf{x}}_k - \mathbf{x}_i))\Big).
\end{aligned}
$$

Because it holds for symmetric matrices $\mathbf{M}$ that

$$
\begin{aligned}
&(\mathbf{a}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{b}) - 2(\mathbf{a}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) + (\mathbf{a}-\mathbf{c})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) \\
=&(\mathbf{a}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{b}) - (\mathbf{a}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) - (\mathbf{a}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) + (\mathbf{a}-\mathbf{c})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) \\
=&(\mathbf{a}-\mathbf{b})^T\mathbf{M}((\mathbf{a}-\mathbf{b}) - (\mathbf{a}-\mathbf{c})) + (-(\mathbf{a}-\mathbf{b})^T + (\mathbf{a}-\mathbf{c})^T)\mathbf{M}(\mathbf{a}-\mathbf{c}) \\
=&(\mathbf{a}-\mathbf{b})^T\mathbf{M}(\mathbf{c}-\mathbf{b}) + (\mathbf{b}-\mathbf{c})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) \\
=&((\mathbf{c}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{b}))^T - (\mathbf{c}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) \\
=&(\mathbf{c}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{b}) - (\mathbf{c}-\mathbf{b})^T\mathbf{M}(\mathbf{a}-\mathbf{c}) \\
=&(\mathbf{c}-\mathbf{b})^T\mathbf{M}((\mathbf{a}-\mathbf{b}) - (\mathbf{a}-\mathbf{c})) \\
=&(\mathbf{c}-\mathbf{b})^T\mathbf{M}(\mathbf{c}-\mathbf{b}),
\end{aligned}
$$

we can further simplify this to give

$$
\begin{aligned}
&\ln r_{ik} - \ln\tilde{\pi}_k - \frac{1}{2}(A_k - \nu_k B_{ik}) \\
=&\ln r_{ik} - \ln\tilde{\pi}_k - \frac{1}{2}\left(\ln\tilde{\Lambda}_k - \frac{D}{\beta_k} - D\ln 2\pi - \nu_k(\mathbf{x}_i - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_i - \mathbf{m}_k)\right).
\end{aligned}
$$

When we now define

$$
E_{ik} = r_{ik}\left\{\ln r_{ik} - \ln\tilde{\pi}_k - \frac{1}{2}\left(\ln\tilde{\Lambda}_k - \frac{D}{\beta_k} - D\ln 2\pi - \nu_k(\mathbf{x}_i - \mathbf{m}_k)^T\mathbf{W}_k(\mathbf{x}_i - \mathbf{m}_k)\right)\right\}
$$

and

$$
F_i = \sum_{k=1}^{K} E_{ik},
$$

we can write

$$
\frac{\partial\mathcal{C}}{\partial\gamma_{ij}} = E_{ij} - r_{ij}\sum_{k=1}^{K} E_{ik} = E_{ij} - r_{ij}F_i
$$

which finally gives us Equation (72).

Derivation of the Riemannian metric tensor G

We will now proceed to derive the Riemannian metric tensor $\mathbf{G}$ of Equation (77). The elements g_{ij} of matrix $\mathbf{G}$ are given by Equation (26) where the approximate distribution $q(\boldsymbol{\theta}, \mathbf{Z}|\boldsymbol{\xi})$ is given by Equations (40), (41) and (42), namely

$$q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N} \prod_{k=1}^{K} r_{nk}^{z_{nk}} q(\boldsymbol{\pi}) \prod_{k=1}^{K} q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k). \tag{B11}$$

It should be noted that because we are taking the gradient with respect to only the means $\mathbf{m}_k$ and the softmax parameters γ_{nk}, the approximate distribution of Equation (B11) is regarded as a function of only the parameters $\gamma_{nk}, n = 1 \dots N, K = 1 \dots K-1$ and the elements of the mean vectors $m_{dk}, d = 1 \dots D, k = 1 \dots K$. Because of the factorization of Equation (B11) is holds that

$$\ln q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \ln r_{nk} + \ln q(\boldsymbol{\pi}) + \sum_{k=1}^{K} \ln q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k). \tag{B12}$$

Now we can immediately see that

$$\begin{aligned}
&- \frac{\partial^2 \ln q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{\partial m_{dk_1} \partial \gamma_{nk_2}} = 0 \\
&- \frac{\partial^2 \ln q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{\partial m_{d_1 k_1} \partial m_{d_2 k_2}} = 0, k_1 \neq k_2 \\
&- \frac{\partial^2 \ln q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{\partial \gamma_{n_1 k_1} \partial \gamma_{n_2 k_2}} = 0, n_1 \neq n_2
\end{aligned}$$

and thus the matrix $\mathbf{G}$ becomes block diagonal with K square blocks of size $D \times D$ corresponding to the means $\mathbf{m}_k$ and N square blocks of size $(K-1) \times (K-1)$ corresponding to the responsibilities of each observation.

Let us proceed to derive the diagonal blocks of matrix $\mathbf{G}$. It holds that

$$\begin{aligned}
&\ln q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \\
=&\ln \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) + \ln \mathcal{W}(\boldsymbol{\Lambda}_k | \mathbf{W}_k, \nu_k) \\
=&- \frac{D}{2} \ln 2\pi - \frac{1}{2} \ln |(\beta_k \boldsymbol{\Lambda}_k)^{-1}| - \frac{\beta_k}{2} (\boldsymbol{\mu}_k - \mathbf{m}_k)^T \boldsymbol{\Lambda}_k (\boldsymbol{\mu}_k - \mathbf{m}_k) + \ln \mathcal{W}(\boldsymbol{\Lambda}_k | \mathbf{W}_k, \nu_k)
\end{aligned}$$

and thus

$$\begin{aligned}
&- \frac{\partial^2 \ln q(\mathbf{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})}{\partial m_{d_1 k} \partial m_{d_2 k}} \\
=&- \frac{\partial^2}{\partial m_{d_1 k} \partial m_{d_2 k}} \ln q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) \\
=&\frac{\beta_k}{2} \frac{\partial^2}{\partial m_{d_1 k} \partial m_{d_2 k}} (\boldsymbol{\mu}_k - \mathbf{m}_k)^T \boldsymbol{\Lambda}_k (\boldsymbol{\mu}_k - \mathbf{m}_k)
\end{aligned}$$

$$
\begin{aligned}
=&\frac{\beta_k}{2}\frac{\partial}{\partial m_{d_1k}}\frac{\partial}{\partial m_{d_2k}}(\boldsymbol{\mu}_k-\mathbf{m}_k)^T\boldsymbol{\Lambda}_k(\boldsymbol{\mu}_k-\mathbf{m}_k)\\
=&\beta_k\frac{\partial}{\partial m_{d_1k}}\sum_{i=1}^{D}(\boldsymbol{\Lambda}_k)_{d_2i}(\mathbf{m}_k-\boldsymbol{\mu}_k)_i\\
=&\beta_k\frac{\partial}{\partial m_{d_1k}}(\boldsymbol{\Lambda}_k)_{d_2d_1}(m_{d_1k}-\mu_{d_1k})\\
=&\beta_k(\boldsymbol{\Lambda}_k)_{d_2d_1}\\
=&(\beta_k\boldsymbol{\Lambda}_k)_{d_2d_1}
\end{aligned}
$$

where we have used similar reasoning for the expression of the derivative of the quadratic form $(\boldsymbol{\mu}_k-\mathbf{m}_k)^T\boldsymbol{\Lambda}_k(\boldsymbol{\mu}_k-\mathbf{m}_k)$ as in the derivation of Equation (B3). The corresponding element of matrix $\mathbf{G}$ is then given by

$$
\begin{aligned}
&E_{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}\left\{-\frac{\partial^2\ln q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}{\partial m_{d_1k}\partial m_{d_2k}}\right\}\\
=&E_{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}\left\{(\beta_k\boldsymbol{\Lambda}_k)_{d_2d_1}\right\}\\
=&E_{q(\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k)}\left\{(\beta_k\boldsymbol{\Lambda}_k)_{d_2d_1}\right\}\\
=&\int_{\boldsymbol{\mu}_k,\boldsymbol{\Lambda}_k}(\beta_k\boldsymbol{\Lambda}_k)_{d_2d_1}\mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k,(\beta_k\boldsymbol{\Lambda}_k)^{-1})\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k,\nu_k)d\boldsymbol{\mu}_k d\boldsymbol{\Lambda}_k\\
=&\int_{\boldsymbol{\Lambda}_k}(\beta_k\boldsymbol{\Lambda}_k)_{d_2d_1}\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k,\nu_k)d\boldsymbol{\Lambda}_k\\
=&\beta_k\nu_k(\mathbf{W}_k)_{d_1d_2}
\end{aligned}
$$

where the expectation over the Wishart distribution is given by Equation (A10). Consequently, the matrices $\mathbf{A}_k$ in Equation (77) are given by $\mathbf{A}_k=\beta_k\nu_k\mathbf{W}_k$.

Similarly, when $k_1\neq k_2$

$$
\begin{aligned}
&-\frac{\partial^2\ln q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}{\partial\gamma_{nk_1}\partial\gamma_{nk_2}}\\
=&-\frac{\partial}{\partial\gamma_{nk_1}}\frac{\partial}{\partial\gamma_{nk_2}}\sum_{k=1}^{K}z_{nk}\ln r_{nk}\\
=&-\frac{\partial}{\partial\gamma_{nk_1}}\frac{\partial}{\partial\gamma_{nk_2}}\left(\sum_{k=1}^{K}z_{nk}\gamma_{nk}-\sum_{k=1}^{K}z_{nk}\ln\sum_{l=1}^{K}e^{\gamma_{nl}}\right)\\
=&\frac{\partial}{\partial\gamma_{nk_1}}\frac{\partial}{\partial\gamma_{nk_2}}\sum_{k=1}^{K}z_{nk}\ln\sum_{l=1}^{K}e^{\gamma_{nl}}\\
=&\frac{\partial}{\partial\gamma_{nk_1}}\frac{\partial}{\partial\gamma_{nk_2}}\ln\sum_{l=1}^{K}e^{\gamma_{nl}}\\
=&\frac{\partial}{\partial\gamma_{nk_1}}\frac{e^{\gamma_{nk_2}}}{\sum_{l=1}^{K}e^{\gamma_{nl}}}\\
=&-\frac{e^{\gamma_{nk_1}}e^{\gamma_{nk_2}}}{(\sum_{l=1}^{K}e^{\gamma_{nl}})^2}
\end{aligned}
$$

$$= - r_{nk_1} r_{nk_2}.$$

Because the responsibilities are not random variables but parameters

$$E_{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})} \left\{ -\frac{\partial^2 \ln q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}{\partial \gamma_{nk_1} \partial \gamma_{nk_2}} \right\} = E_{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})} \left\{ -r_{nk_1} r_{nk_2} \right\} = -r_{nk_1} r_{nk_2}, k_1 \neq k_2.$$

When $k_1 = k_2 = k$, we get

$$\begin{aligned}
& - \frac{\partial^2 \ln q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}{\partial \gamma_{nk} \partial \gamma_{nk}} \\
=& \frac{\partial}{\partial \gamma_{nk}} \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} \\
=& e^{\gamma_{nk}} \frac{\partial}{\partial \gamma_{nk}} \frac{1}{\sum_{l=1}^{K} e^{\gamma_{nl}}} + \frac{1}{\sum_{l=1}^{K} e^{\gamma_{nl}}} \frac{\partial}{\partial \gamma_{nk}} e^{\gamma_{nk}} \\
=& - \left(\frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} \right)^2 + \frac{e^{\gamma_{nk}}}{\sum_{l=1}^{K} e^{\gamma_{nl}}} \\
=& - r_{nk}^2 + r_{nk}
\end{aligned}$$

and

$$E_{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})} \left\{ -\frac{\partial^2 \ln q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})}{\partial \gamma_{nk} \partial \gamma_{nk}} \right\} = E_{q(\mathbf{Z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})} \left\{ -r_{nk}^2 + r_{nk} \right\} = -r_{nk}^2 + r_{nk}.$$

Thus, the matrices $\mathbf{B}_n$ of Equation (77) are given by $\mathbf{B}_n = -\mathbf{r}_n^T \mathbf{r}_n + \text{diag}(\mathbf{r}_n)$ where $\mathbf{r}_n = [r_{n1} \cdots r_{nK-1}]$.

Appendix C: Suomenkielinen yhteenveto

Bayesilainen päättely on stokastinen menetelmä, joka mahdollistaa todennäköisyyksien antamisen jonkin suureen erilaisille arvoille perustuen käyttäjän ennakko-odotuksiin eli priorijakaumaan. Näin saatua todennäköisyysjakaumaa kutsutaan posteriorijakaumaksi. Koneoppimisessa bayesilaista päättelyä käytetään estimoimaan dataa kuvaavan mallin parametreja. Tällöin bayesilaisen päättelyn ydinajatus voidaan kiteyttää Bayesin kaavaan

$$p(\boldsymbol{\theta}, \mathbf{Z}|\mathbf{X}, \mathcal{M}) = \frac{p(\mathbf{X}|\boldsymbol{\theta}, \mathbf{Z}, \mathcal{M})p(\boldsymbol{\theta}, \mathbf{Z}|\mathcal{M})}{p(\mathbf{X}|\mathcal{M})}, \tag{C1}$$

jossa $\boldsymbol{\theta}$ on malliparametrit, $\mathbf{Z}$ latentit muuttujat eli jokaiseen havaintoon liittyvät muuttujat, joita ei voida havaita suoraan, $\mathbf{X}$ havaittu data ja $\mathcal{M}$ mallioletukset. Yhtälön vasen puoli on posteriorijakauma, joka kertoo todennäköisyyden eri malliparametreille $\boldsymbol{\theta}$ ja latenteille muuttujille $\mathbf{Z}$, kun ollaan havaittu data $\mathbf{X}$ ja datan kuvaamiseen käytetään mallia $\mathcal{M}$. Kun posteriorijakaumasta otetaan odotusarvo, voidaan bayesilaista päättelyä siten käyttää mallin $\mathcal{M}$ sovittamiseen dataan $\mathbf{X}$.

Bayesilaisen päättelyn keskeinen ongelma on kuitenkin, että monesti laskut johtavat integraaleihin, joiden laskeminen on hankalaa. Esimerkiksi yhtälön (C1) oikean puolen nimittäjässä olevaa jakaumaa $p(\mathbf{X}|\mathcal{M})$ ei pystytä useimmissa tosielämän tilanteissa laskemaan analyyttisesti. Jakauma voidaan määrittää numeerisesti käyttäen Monte Carlo -integrointia, mutta tämä on monissa tilanteissa liian hidasta. Useimmat perinteiset approksimaatiot taas ylisovittavat mallin dataan. Tässä työssä tarkastellaan vaihtoehtoista, viime aikoina suosituksi noussutta approksimatiivista bayesilaisen päättelyn menetelmää, joka tunnetaan nimellä variaatio-Bayes-oppiminen (variational Bayesian learning) [11, 3, 5]. VB-oppimisessa todellinen posteriorijakauma korvataan toisella jakaumalla q, jonka poikkeamaa todellisesta posteriorijakaumasta kuvataan Kullback–Leibler-divergenssiin perustuvalla kustannusfunktiolla. Mitä pienempi kustannusfunktion arvo on, sitä paremmin approksimatiivinen jakauma q kuvaa todellista posterioria. Kun jakauman q funktionaalista muotoa rajoitetaan sopivasti, voidaan kustannusfunktion arvo laskea analyyttisesti.

Kun jakaumaa q rajoitetaan olettamalla malliparametrit ja latentit muuttujat riippumattomiksi satunnaismuuttujiksi, päädytään niin sanottuun VB EM -algoritmiin (variational Bayesian expectation maximization). VB EM on paljon tutkittu ja laajalti käytössä oleva syklinen algoritmi, mutta valitettavasti se soveltuu vain rajoitetun malliperheen oppimiseen. Erityisesti niin sanotun konjugaattieksponentiaaliperheen ulkopuolisten mallien oppimisessa on pääsääntöisesti käytettävä muita menetelmiä.

Vaihtoehtoinen tapa suorittaa VB-oppimista on valita jakaumalle q jokin muoto ja minimoida kustannusfunktio käyttäen sopivaa epälineaarista optimointimenetelmää. Näistä menetelmistä yksinkertaisin on gradienttimenetelmä, jossa jakauman q parametriavaruudessa liikutaan kustannusfunktion negatiivisen gradientin suuntaan. Menetelmää voidaan parantaa huomioimalla edellinen päivityssuunta, jolloin saadaan konjugaattigradienttimenetelmä.

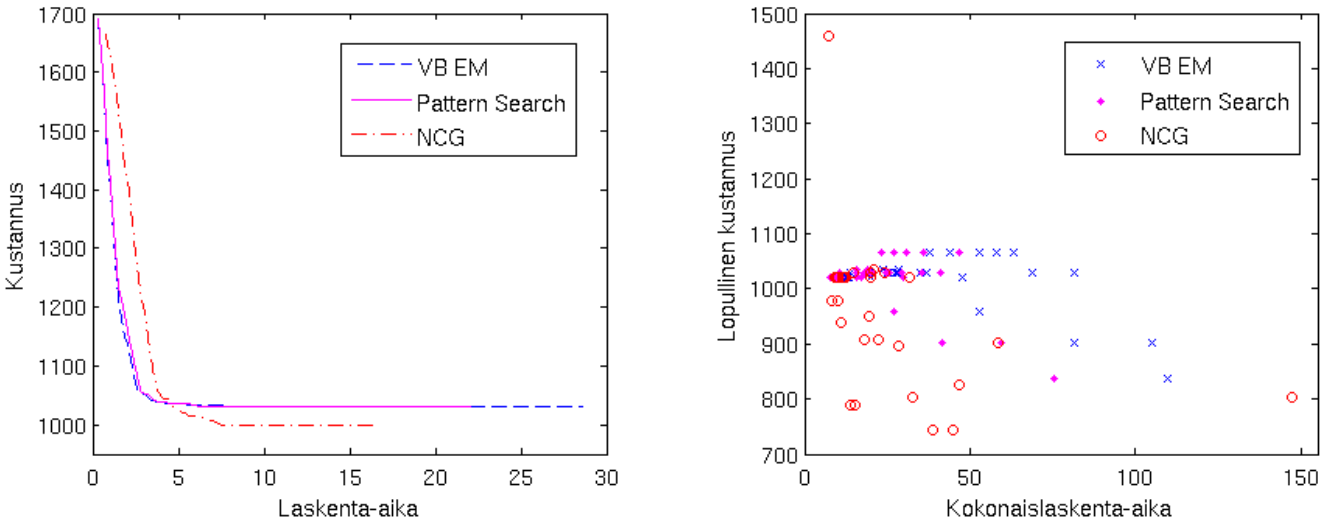
Perinteiset gradienttimenetelmät pitävät kuitenkin kaikkia jakauman q parametreja tasavertaisina, vaikka todellisuudessa niillä voi olla hyvinkin erilainen vaikutus jakauman muotoon. Esimerkiksi Gaussin jakauman keskiarvon ja varianssin vaikutus jakauman muotoon on huomattavan erilainen. Gradienttimenetelmää voidaankin tehostaa tulkitsemalla parametriavaruus kaareutuneeksi Riemannin monistoksi laakean euklidisen avaruuden sijaan. Tällöin perinteinen gradientti korvataan luonnollisella gradientilla. Kun luonnollista gradienttia sovelletaan konjugaattigradienttimenetelmään, saadaan luonnollinen konjugaattigradienttialgoritmi (natural conjugate gradient, NCG) [12].

NCG:tä on tähän mennessä tutkittu käyttäen monimutkaista epälineaarista tilaavaruusmallia (nonlinear state space model, NSSM), jonka oppimisessa se oli huomattavasti muita gradienttipohjaisia menetelmiä tehokkaampi. NSSM:n monimutkaisuuden vuoksi NCG:tä ei kuitenkaan voitu verrata VB EM:ään. Tässä kandidaatintyössä tämä vertailu suoritetaan käyttäen Gaussin mikstuuriksi kutsuttua mallia, joka on usean moniulotteisen Gaussin jakauman lineaarikombinaationa saatava konjugaattieksponentiaaliperheen malli, jonka oppimiseen VB EM:n tiedetään soveltuvan hyvin. Tätä varten työssä johdetaan ja implementoidaan NCG-algoritmi Gaussin mikstuurin oppimiseen.

Gaussin mikstuurin tapauksessa VB EM -algoritmi toimii siten, että ensin päivitetään latenttien muuttujien jakaumaa eli tietoa siitä, mistä Gaussin jakaumasta kukin havainto on peräisin. Tätä kutsutaan E-askeleeksi. Tätä seuraavassa M-askeleessa päivitetään malliparametrien jakaumaa eli Gaussin jakaumien keskiarvoja, kovarianssimatriiseja ja painoarvoja. Näin jatketaan, kunnes algoritmi konvergoi eli kustannusfunktion arvo ei enää muutu. VB EM:ää voidaan nopeuttaa yhdistämällä peräkkäisten E- ja M-askelten päivityssuunnat ja suorittamalla viivahaku näin saatavaan suuntaan. Tämän jälkeen suoritetaan muutama VB EM -päivitys, joita seuraa jälleen viivahaku. Tätä kutsutaan pattern search -menetelmäksi [13]. Tässä työssä tutkitaan myös, miten pattern search -menetelmän käyttö vaikuttaa VB EM:n toimintaan Gaussin mikstuurin oppimisessa.

Työn kokeellisessa osiossa algoritmien toimintaa verrataan käyttäen sekä keinotekoista dataa että todellisen maailman kuvadataa. Kokeiden perusteella käytetyllä datalla on suuri vaikutus algoritmien toimintaan. Esimerkiksi käytettäessä datajoukkoa, jonka datapisteet muodostavat kolmiulotteisen spiraalin, saadaan kuvan C1 mukaiset tulokset. Kuvassa C1(a) on esitetty kustannusfunktion mediaani laskenta-ajan funktiona kullakin algoritmilla, kun taas kuvassa C1(b) on esitetty yksittäisten laskenta-ajojen saavuttamat lopulliset kustannusfunktion arvot ja näiden saavuttamiseen vaadittu laskenta-aika. Kokeet toistettiin 30 kertaa, sillä, kuten kuvasta C1(b) nähdään, satunnaisesti valituilla alkuarvoilla on suuri vaikutus algoritmien saavuttamaan lopputulokseen. Tällä datalla NCG saavutti keskimäärin lyhyemmässä laskenta-ajassa paremman lopputuloksen kuin muut algoritmit. Pattern search -menetelmä vuorostaan paransi VB EM:n nopeutta.

Myös muilla tutkituilla datajoukoilla saavutettiin vastaavanlaisia tuloksia: Pattern search -menetelmä paransi lähes poikkeuksetta VB EM:n nopeutta. NCG:n nopeus



(a) (b)

Kuva C1: Algoritmien vertailua käytettäessä kolmiulotteista spiraalidataa. Kuvassa (a) on esitetty oppimiskäyrät eli kustannusfunktion mediaani laskenta-ajan funktiona eri algoritmeilla. Kuvassa (b) on vuorostaan esitetty eri laskenta-ajojen saavuttamat lopputulokset.

oli useimmissa kokeissa erittäin kilpailukykyinen VB EM -pohjaisten algoritmien kanssa. Parhaissa tapauksissa NCG löysi nopeammin parempia optimeita kuin muut algoritmit. Toisaalta eräillä kuvadatajoukoilla NCG:n suorituskyky oli selvästi VB EM:ää huonompi. Gradienttipohjaisista algoritmeista NCG osoittautui ylivoimaisesti parhaaksi, kuten NSSM:stä saatujen tulosten perusteella voitiin olettaa. Työn yhteenvetona voidaankin todeta, että käytetty data määrää sen, mikä tutkituista algoritmeista soveltuu tilanteeseen parhaiten. Pattern search -menetelmällä nopeutettu VB EM osoittautui varmatoimiseksi ja hyväksi vaihtoehdoksi erittäin monella tutkitulla datajoukolla, joten näiden kokeiden perusteella se olisi ensisijainen algoritmivalinta Gaussin mikstuurin oppimiseen. Tästä huolimatta sopivilla datajoukoilla NCG voi olla selvästi muita algoritmeja parempi erityisesti parhaan optimin löytämisessä.