AALTO UNIVERSITY
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

Matti Tornio

Natural Gradient for Variational Bayesian Learning

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

Espoo, May 26, 2010

Supervisor: Professor Juha Karhunen
Instructor: Antti Honkela, D.Sc. (Tech.)

AALTO-YLIOPISTO	DIPLOMITYÖN TIIVISTELMÄ
TEKNILLINEN KORKEAKOULU	

Tekijä:	Matti Tornio
Työn nimi:	Luonnollinen gradientti variaatio-Bayes-oppimisessa
Päiväys:	26.5.2010
Työn kieli:	Englanti
Tiedekunta:	Informaatio- ja luonnontieteiden tiedekunta
Tutkinto-ohjelma:	Tietotekniikan tutkinto-ohjelma/koulutusohjelma
Professuurin koodi ja nimi:	T-61 Informaatiotekniikka
Työn valvoja:	Prof. Juha Karhunen
Työn ohjaaja:	TkT Antti Honkela

Tiivistelmä:

Todennäköisyysmalleilla on hyvin tärkeä asema koneoppimisessa, ja näiden mallien tehokas oppiminen on tärkeä ongelma. Valitettavasti näiden mallien matemaattinen käsittely suoraan on usein mahdotonta, ja mallien oppimisessa joudutaankin turvautumaan erilaisiin approksimaatioihin. Eräs tällainen approksimaatio on variaatio-Bayes-menetelmä, jossa todellista posteriorijakaumaa approksimoidaan toisella jakaumalla ja näiden kahden jakauman välistä eroa pyritään minimoimaan.

Variaatio-Bayes-oppimisessa voidaan käyttää monia eri optimointialgoritmeja. Tässä työssä keskitytään gradienttipohjaisiin algoritmeihin. Näillä algoritmeilla on kuitenkin tyypillisesti yksi heikkous. Yleensä nämä menetelmät olettavat, että avaruus jossa funktiota optimoidaan on geometrialtaan euklidinen. Tilastollisissa malleissa tämä ei usein pidä paikkaansa, vaan avaruus on todellisuudessa Riemannin monisto. Luonnolliseen gradienttiin pohjautuvat optimointialgoritmit ottavat tämän geometrisen ominaisuuden huomioon ja ovat usein huomattavasti nopeampia kuin perinteiset optimointialgoritmit. Eräs tehokas ja suhteellisen yksinkertainen menetelmä saadaan yleistämällä konjugaattigradienttialgoritmi Riemannin monistoille. Näin saatua menetelmää kutsutaan Riemannin konjugaattigradientiksi.

Tässä työssä esitellään tehokas Riemannin konjugaattigradienttialgoritmi variaatio-Bayes-menetelmää käyttävien tilastollisten mallien oppimiseen. Esimerkkiongelmana käytetään epälineaarisia tila-avaruusmalleja, joita käytetään sekä keinotekoisten että todellisten data-aineistojen oppimiseen. Näistä kokeista saadut tulokset osoittavat että esitelty algoritmi on huomattavasti tehokkaampi kuin muut vertailussa käytetyt perinteisemmät algoritmit.

Sivumäärä: 65	Avainsanat:	koneoppiminen, luonnollinen gradientti, Riemannin konjugaattigradientti, epälineaariset tila-avaruusmallit, variaatio-Bayes-menetelmä

AALTO UNIVERSITY SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF MASTER'S THESIS

Author:	Matti Tornio
Title:	Natural Gradient for Variational Bayesian Learning
Date:	26.5.2010
Language:	English
Faculty:	Faculty of Information and Natural Sciences
Degree Programme:	Degree Programme of Computer Science and Engineering
Professorship:	T-61 Computer and Information Science
Supervisor:	Prof. Juha Karhunen
Instructor:	Antti Honkela, D.Sc. (Tech.)

Abstract:

Probabilistic models play a very important role in machine learning, and the efficient learning of such models is a very important problem. Unfortunately, the exact statistical treatment of probabilistic models is often impossible and therefore various approximations have to be used. One such approximation is given by variational Bayesian (VB) learning which uses another distribution to approximate the true posterior distribution and tries to minimise the misfit between the two distributions.

Many different optimisation algorithms can be used for variational Bayesian learning. This thesis concentrates on gradient based optimisation algorithms. Most of these algorithms suffer from one significant shortcoming, however. Typically these methods assume that the geometry of the problem space is flat, whereas in reality the space is a curved Riemannian manifold. Natural-gradient-based optimisation algorithms take this property into account, and can often result in significant speedups compared to traditional optimisation methods. One particularly powerful and relatively simple algorithm can be derived by extending conjugate gradient to Riemannian manifolds. The resulting algorithm is known as Riemannian conjugate gradient.

This thesis presents an efficient Riemannian conjugate gradient algorithm for learning probabilistic models where variational approximation is used. Nonlinear state-space models are used as a case study, and results from experiments with both synthetic and real-world data sets are presented. The results demonstrate that the proposed algorithm provides significant performance gains over the other compared methods.

Number of pages: 65 Keywords: machine learning, natural gradient, Riemannian conjugate gradient, nonlinear state-space models, variational Bayes

Preface

This work has been done in the Laboratory of Computer and Information Science at Helsinki University of Technology.

I thank professor Juha Karhunen for supervision. I also thank Dr. Antti Honkela for guidance on the subject and technical help with Matlab and LaTeX. Finally, I also wish to thank Tapani Raiko for creative discussion and help with technical matters.

Otaniemi, October 15, 2007

Matti Tornio

Contents

List of Abbreviations

CG	Conjugate gradient
EM	Expectation maximisation
FA	Factor analysis
ICA	Independent component analysis
KL	Kullback-Leibler (divergence)
MAP	Maximum a posteriori (solution)
MCMC	Markov chain Monte Carlo
ML	Maximum likelihood (solution)
MLP	Multi-layer perceptron (network)
NDFA	Nonlinear dynamical factor analysis
NG	Natural gradient
NSSM	Nonlinear state-space model
PCA	Principal component analysis
pdf	Probability density function
RBF	Radial basis function
VB	Variational Bayes

List of Symbols

$\overline{\theta}$	Mean of the parameter θ in the approximating posterior distribution q
$\widetilde{\theta}$	Variance of the parameter θ in the approximating posterior distribution q
$\hat{\theta}$	Linear dependence parameter of the parameter θ in the approximating posterior distribution q
$\mathbf{A}, \mathbf{B}.\mathbf{C}, \mathbf{D}$	The mixing matrices of the generative mappings
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$	Bias terms of the nonlinear generative mappings
$D_{KL}(q\|\|p)$	The Kullback–Leibler divergence between the distributions q and p
$\mathrm{diag}(\mathbf{x})$	A diagonal matrix with the elements of vector $\mathbf{x}$ on the main diagonal
$E\left\{\cdot\right\}$	Expectation over a distribution
$\exp(\mathbf{x})$	Exponential function applied component-wise to the vector $\mathbf{x}$
$\mathbf{f}, \mathbf{g}$	Nonlinear generative mappings
$\mathbf{G} = (g_{ij})$	Riemannian metric tensor
$\gamma : [0, 1] \mapsto S$	A curve on the manifold S
$\nabla\mathcal{F}(\boldsymbol{\xi})$	Gradient of the scalar function $\mathcal{F}$
$\tilde{\nabla}\mathcal{F}(\boldsymbol{\xi})$	Natural gradient of the scalar function $\mathcal{F}$
$N(\mathbf{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian (or normal) distribution for variable $\mathbf{x}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$\boldsymbol{\theta}$	The vector of all model parameters
ϕ_i	Coordinate curves (or functions) of a Riemannian manifold
φ	The activation function of a MLP network
$\boldsymbol{\xi}$	The parameters defining the approximating distribution q
$p(A\|B)$	Probability density of A with condition B
$p(\boldsymbol{\theta}\|\mathbf{X})$	Posterior density

$p(\mathbf{X}\|\boldsymbol{\theta})$	Likelihood density
$p(\boldsymbol{\theta})$	Prior density
$p(\mathbf{X})$	Marginal likelihood
$q(\boldsymbol{\theta}\|\boldsymbol{\xi})$	The parametric approximation of the posterior pdf
$\tau\mathbf{v}$	Parallely transported version of vector $\mathbf{v}$ on a Riemannian manifold
T_p	The tangent space of the point p on a Riemannian manifold
$\mathbf{p_k}$	The search direction of a conjugate gradient method at iteration k
$\mathbf{s}(t)$	A state vector at time t
$\mathbf{S}$	The set of all the states
$\mathbf{x}(t)$	An observed data vector at time t
$\mathbf{X}$	The set of all the observed data
$\mathrm{Var}\left\{\cdot\right\}$	Variance over a distribution

Chapter 1

Introduction

1.1 Problem Setting

The typical goal in machine learning is to build a model for a given set of data. Usually these models are specified by a set of parameters, values of which are optimised until the model describes the data well enough. Many different optimisation algorithms are used to learn these models, including the EM-algorithm and various direct optimisation algorithms such as gradient descent.

Most traditional optimisation algorithms assume that this parameter space is flat. However, in many cases, especially in statistical problems, the actual geometry of the problem space is not flat but a curved Riemannian manifold. Taking this property into account can lead to more efficient optimisation algorithms, the most popular example of which is the natural gradient algorithm [3].

Variational Bayes [9, 35, 37, 11], also previously known as Bayesian ensemble learning, is an efficient algorithm for approximate Bayesian inference and it is often used for statistical learning of probabilistic models. One such class of probabilistic models is nonlinear state-space model (NSSM).

1.2 Aim of the Thesis

The aim of this thesis has been to develop a more efficient learning algorithm for variational Bayesian learning of NSSMs based on natural gradient learning. The particular NSSM used in this work is the nonlinear dynamical factor analysis (NDFA) model developed by Dr. Harri Valpola and Prof. Juha Karhunen [71].

The algorithm was implemented by extending the publicly available NDFA package [70]. The performance of the algorithm was verified by using it to model two different synthetic data sets and a real-world speech data set.

Even though state-space models are used as an example in this work, the presented algorithm can be applied to almost any probabilistic model where the parameter space is a Riemannian manifold.

1.3 Structure and Contributions of the Thesis

This thesis is organised as follows. Chapter 2 gives an introduction to Bayesian learning in general and variational Bayes in particular. Information geometry and natural gradient learning are discussed in Chapter 3. Conjugate gradient method and its extension to Riemannian manifolds are studied in Chapter 4 as more efficient alternatives to gradient descent learning. Chapter 5 introduces nonlinear state-space models as a case study for the presented algorithm and introduces the dynamical model used in the examples and experiments. This chapter also includes an overview of implementation details.

Experimental results with two synthetic data sets and real world speech data are presented and analysed in Chapter 6. The benefits and restrictions of the proposed algorithm and potential future work are discussed in Chapter 7. Finally, overview of the work and some conclusions are presented in Chapter 8.

The original idea to use methods based on natural gradient with nonlinear dynamical factor analysis (NDFA) package [70] arose from the observation of the poor performance of the conjugate gradient method with NDFA. Discussion between Dr. Antti Honkela, Tapani Raiko, and the author lead to an implementation of a natural gradient method based on a remark in [69]. The idea to use Riemannian conjugate gradient to further improve the performance is due to the author. The implementation of both the original natural gradient method and the Riemannian conjugate gradient method for NDFA were also done by the author. The code is based on the original nonlinear dynamical factor analysis implementation by Dr. Harri Valpola and Dr. Antti Honkela and its later extensions by Dr. Antti Honkela. All the experiments presented in Chapter 6 were done by the author.

Chapter 2

Bayesian Inference

This chapter gives a brief introduction to Bayesian probability theory and introduces the variational approximation of the posterior probability density. More detailed description of the variational Bayesian learning (sometimes referred to as ensemble learning) can be found e.g. in [9, 69, 35, 33, 37, 11].

A brief introduction to Bayesian statistics is given in Section 2.1. The important concept of Kullback-Leibler divergence is introduced in Section 2.2. Different methods of approximating the typically intractable posterior probability distribution are discussed in Section 2.3. The variational Bayesian approximation is discussed in more detail in Section 2.4. Finally, the popular EM-algorithm is introduced in Section 2.5.

2.1 Introduction to Bayesian Inference

In the Bayesian approach to probability theory, probability is a subjective measure of degree of belief of an uncertain event. In a contrast to frequentist approach, any kinds of events can be assigned probabilities, even if the event itself is completely deterministic.

It has been shown [14] that from some very general assumptions and compatibility with common sense these degrees of beliefs must satisfy

$$p(B|A) + p(\neg B|A) = 1 \tag{2.1}$$

and

$$p(C, B|A) = p(C|B, A)p(B|A), \tag{2.2}$$

where A, B, and C are propositions and $\neg B$ is the negation of B. The two rules are known as the sum rule and the product rule, respectively. From these rules it is relatively straightforward to derive the basic laws of Bayesian probability, namely the Bayes' rule and the marginalisation principle.

2.1.1 Bayes' Rule

The Bayes' rule

$$p(C|B,A) = \frac{p(B|C,A)p(C|A)}{p(B|A)}. \tag{2.3}$$

directly follows from the product rule (2.2). Bayes' rule determines how a learning system should update its prior beliefs A after receiving new information (observation) B. Under the usual naming conventions, C is known as the proposition of interest, $p(B|C,A)$ is known as the likelihood and $p(C|A)$ is the prior probability. The scaled product $p(C|B,A)$ of the prior probability and the likelihood is known as the posterior probability [27].

2.1.2 Marginalisation Principle

In addition to Bayes' rule, we can also derive the marginalisation principle from Equations (2.1) and (2.2). Given a set of mutually exclusive propositions $\{C_k\}$ which satisfy

$$\sum_{i=1}^{n} p(C_i|A) = 1, \tag{2.4}$$

the marginalisation principle can be written as

$$p(B|A) = \sum_{i=1}^{n} p(B,C_i|A) = \sum_{i=1}^{n} p(B|C_i,A)p(C_i|A) \tag{2.5}$$

for the discrete case and

$$p(B|A) = \int_{\theta} p(B,\theta|A)d\theta = \int_{\theta} p(B|\theta,A)p(\theta|A)d\theta \tag{2.6}$$

for the continuous case. Whereas Bayes' rule is used to update the beliefs of the system, the marginalisation principle can be used to make predictions and generalisations.

2.1.3 Model Comparison

While building a model for a set of observations, too simple models tend to represent the observations poorly. This problem is known as underfitting. On the other hand, while very complex models can represent the observations accurately, they often generalise poorly. This is known as overfitting. This can be used to justify the principle known as Occam's Razor: the simplest explanation that adequately describes the observations is usually the best.

Occam's Razor has a straightforward intepretation in statistics. Given a set of observations $\mathbf{X}$ and assuming a constant prior, different models $\mathcal{H}_1$, $\mathcal{H}_2$, ... can be directly compared by their marginal likelihood

$$p(\mathbf{X}|\mathcal{H}_i) = \int_{\boldsymbol{\theta}} p(\mathbf{X}, \boldsymbol{\theta}|\mathcal{H}_i) d\boldsymbol{\theta} = \int_{\boldsymbol{\theta}} p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{H}_i) p(\boldsymbol{\theta}|\mathcal{H}_i) d\boldsymbol{\theta}. \tag{2.7}$$

2.1.4 Conjugate Priors

An important way to simplify Bayesian inference is provided by conjugate priors. Given a class of likelihood function $p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{H})$, the priors $p(\boldsymbol{\theta}|\mathcal{H})$ are called conjugate if the posteriors $p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{H})$ belong to the same distribution class $\mathcal{P}$ as the priors.

If the class $\mathcal{P}$ has a common functional form, conjugate priors will greatly simplify inference. Conjugate priors exist for many important distribution families. For example, all distributions in the exponential family have conjugate priors [19].

2.2 Entropy and Kullback-Leibler Divergence

The information content of a discrete random variable x is given by the entropy of the distribution $p(x)$

$$H(x) = -\sum_i p(x_i) \log p(x_i), \tag{2.8}$$

where the summation is done over all the possible values of x_i. The discrete entropy $H(x)$ is always non-negative and it gives the lower bound to the number of bits needed on average to encode the information contained in x [37, 24, 13].

It is also possible to generalise the concept of entropy to the continuous variables. If the variable x is continous, summation is replaced by integration and the differential entropy is given by

$$h(x) = \int_{\mathbb{R}} p(x) \log p(x) dx. \tag{2.9}$$

In contrast to discrete entropy, differential entropy has no lower bound and it is typically affected by reparametrisation. In the space of probability distributions, the discrete entropy is maximised by the uniform distribution. For the particular case of fixed covariance, differential entropy is maximised by the Gaussian distribution [62, 13, 37].

2.2.1 Kullback-Leibler Divergence

The information difference between two different distributions $p(x)$ and $q(x)$ is measured by the relative entropy or Kullback-Leibler divergence

$$D_{KL}(q||p) = E_q \left\{ \log \frac{q(x)}{p(x)} \right\} = \int_{\mathbb{R}} q(x) \log \frac{q(x)}{p(x)} dx. \tag{2.10}$$

Kullback-Leibler divergence is non-negative and it is invariant under invertible reparameterisations. Even though Kullback-Leibler can be seen as a measure of distance between two distributions, it is not an actual metric since it is neither symmetric nor satisfies the triangle inequality [27, 13].

2.3 Posterior Approximations

From the theoretical point of view, Bayesian statistics provide the tools for performing optimal inference. All the required information is contained in the posterior distribution, which can in theory be computed using the relatively simple tools of Bayesian statistics. Unfortunately, in practice the exact computation of the posterior probability distribution is not feasible except for some simple special cases. Typical solutions to overcome this problem include approximating the exact posterior with point estimates, sampling, or parametric approximations.

2.3.1 Point Estimates

Examples of point estimates include maximum a posteriori (MAP) estimation and the related maximum likelihood (ML) estimation, which aim to maximise the posterior density and the likelihood, respectively. Point estimates are easy to compute, but unfortunately they are often prone to overfitting. Especially in higher dimensions MAP estimates suffer from the fact that high probability density does not guarantee the presence of high probability mass. Narrow spikes with high probability density may actually have very little probability mass as seen in Figure 2.1 [69].

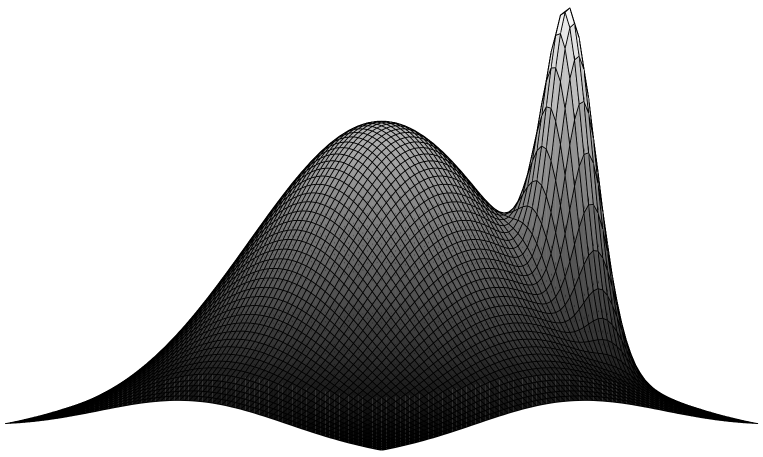


Figure 2.1: Example of probability density in a two dimensional case. The spike on the right has the highest probability density even though most of the probability mass is elsewhere.

2.3.2 Sampling Methods

Sampling methods are based on drawing samples from the true posterior distribution, which is usually accomplished by constructing a Markov chain for the model parameters $\boldsymbol{\theta}$ and using the posterior distribution as the stationary distribution of the Markov chain. These samples can then be used to approximate computations such as integration over the true posterior.

The resulting method is known as Markov Chain Monte Carlo (MCMC) and the most important such algorithms are the Metropolis-Hastings algorithm and Gibbs sampler. Sampling methods can be applied to a very wide range of different problems and with enough samples the results are very accurate and robust against overfitting. Unfortunately, sampling methods scale poorly to high dimemsional problems as the number of samples needed grows extremely large and in some problems it is also hard to determine when the algorithm has converged [46, 37].

2.3.3 Parametric Approximations

Parametric approximations strike a balance between the point estimates and sampling methods; they can be computed quite efficiently and yet they are typically robust against overfitting. This work concentrates on the variational approximation, which is presented in the next section.

2.4 Variational Bayes

There exists numerous different parametric approximations, the one considered in this work is the variational approximation, which leads to variational Bayesian learning. Variational Bayes [37, 11, 35, 9] is a way to approximate the posterior density. For a model with parameters $\boldsymbol{\theta}$ and observed data $\mathbf{X}$, variational Bayes tries to maximise a lower bound on the marginal log-likelihood

$$\mathcal{B}(q(\boldsymbol{\theta}|\boldsymbol{\xi})) = \left\langle \log \frac{p(\mathbf{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta}|\boldsymbol{\xi})} \right\rangle = \log p(\mathbf{X}) - D_{\mathrm{KL}}(q(\boldsymbol{\theta}|\boldsymbol{\xi})||p(\boldsymbol{\theta}|\mathbf{X})), \tag{2.11}$$

where $\boldsymbol{\xi}$ are the parameters of the approximating distribution. This optimisation problem is equivalent to minimising the misfit between the exact posterior pdf $p(\boldsymbol{\theta}|\mathbf{X})$ and its parametric approximation $q(\boldsymbol{\theta}|\boldsymbol{\xi})$ characterised by the Kullback-Leibler divergence $D_{\mathrm{KL}}(q||p)$ between p and q [20, 72].

The variational approximation has several desirable properties. First of all, the approximation is very robust against overfitting and the density estimates are relatively fast to evaluate compared to e.g. sampling methods. In addition, variational approximation provides a cost function for comparing different models. From the point of view of this work, it is also important to note that variational approximation has a straightforward geometric interpretation on curved manifolds as discussed in Section 3.1.3.

Unfortunately, variational Bayes also has some shortcomings. First of all, even though the estimates are fast to evaluate compared to sampling methods, the approximation is in many cases much slower to evaluate than a point estimate. Additionally, variational Bayes has a tendency to underestimate the variance of the true posterior distribution, which can lead to problems in some cases. An important alternative to variational Bayes is given by expectation propagation (EP) algorithm [41], which can solve some of the problems of the variational Bayes method. Unfortunately, exceptation propagation algorithms are more difficult to implement than variational Bayesian alternatives, and the lack of a simple cost function in exceptation propagation also means that it is hard to guarantee the convergence of the algorithm.

2.4.1 Factorisation

In many problems where the posterior dependencies are relatively weak, it is beneficial to assume that the different model parameters are independent. Under this assumption the posterior approximations can be written as

$$q(\boldsymbol{\theta}) = \prod_i q(\theta_i). \tag{2.12}$$

This factorisation will greatly simplify the computation of the bound $\mathcal{B}$ as the equation can be written as a sum of simple terms and the integrals over the posterior approximation become independent.

Experiments by Miskin and MacKay [42] with variational Bayes indicate that in the case of blind source separation the difference in model quality between full covariance and factorial approximation is small while the difference in computational complexity is significant. However, experiments by Ilin and and Valpola [32] suggest that using fully factorised posterior approximation can lead to very poor results in some cases, and care must be taken while choosing the level of factorisation.

Therefore in problems where the posterior dependencies are significant, the full factorial approximation cannot be used. In many such problems it is still sufficient to model only some of dependencies, and the full covariance may not be needed. Example of such partial factorial approximation is modeling only the dependencies between subsequent samples of the same variable in a dynamical model, which is used in nonlinear dynamical factor analysis (NDFA) model presented in Section 5.2.

2.5 EM Algorithm

Traditionally, the expectation maximisation (EM) algorithm [15] and more recently its variational Bayesian extension [47] have been used to solve a wide variety of machine learning problems. This work concentrates on direct optimisation algorithms such as the conjugate gradient method, but for the sake of completeness, the EM algorithm is shortly introduced as well.

The EM algorithm alternates between the E-step, where the posterior distribution of the states $\mathbf{S}$ is computed using the current estimate of parameters $\boldsymbol{\theta}_{t-1}$:

$$q_t(\mathbf{S}) = p(\mathbf{S}|\mathbf{X}, \boldsymbol{\theta}_{t-1}, \mathcal{H}), \tag{2.13}$$

and the M-step, where the expected log-likelihood is maximised with respect to the parameters $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_t = \mathrm{argmax}_{\boldsymbol{\theta}}\, E_q(\log p(\mathbf{S}, \mathbf{X}|\boldsymbol{\theta}, \mathcal{H})). \tag{2.14}$$

The EM algorithm can be applied to a wide variety of different problems and it is guaranteed to converge to a local optimum apart from some special cases [15, 47]. Unfortunately, in certain problems the EM algorithm can converge very slowly. There exists a number of ways to speed up the convergence of EM algorithm. One simple way is to use pattern search methods [30, 29]. Another solution is given by adaptive overrelaxation [58]. These methods are easy to implement, but typically they increase performance only by a small constant factor while retaining the linear convergence of EM algorithm.

Another more complex approach is proposed in [59]. Based on the fact that the perfomance of the EM algorithm is related to the amount of missing information, an algorithm is derived which approximates this ratio of missing information, and based on this information, updates the parameters using either the EM algorithm or a conjugate gradient based optimization method, in this case expectation-conjugate-gradient (ECQ) [59].

Chapter 3

Information Geometry

Applying differential geometry to families of probability distributions is known as information geometry. This chapter provides only a brief introduction to many important concepts of information geometry, and is mostly restricted to concepts relevant to this work. More detailed and comprehensive introductions can be found e.g. in [44, 1, 5].

The basic concepts of information geometry are presented in Section 3.1. In Section 3.2 the natural gradient is presented, and its exact form is also derived for some example distribution families.

3.1 Introduction to Information Geometry

For the purposes of this work, we restrict ourselves to manifolds for which global coordinate systems exist. Under this assumption, we can informally define a manifold as follows. The set S is a (C^∞ differentiable) n-dimensional manifold, if there exists a set of coordinate systems $\mathcal{A}$ for S which satisfies [5]

(i) Each element ϕ of $\mathcal{A}$ is a one-to-one mapping from S to some open subset of $\mathbb{R}^n$.

(ii) For all $\psi \in \mathcal{A}$, given any one-to-one mapping ϕ from S to $\mathbb{R}^n$, the following holds:

$$\phi \in A \Longleftrightarrow \phi \cdot \psi^{-1} \text{ is a } C^\infty \text{ diffeomorphism,} \tag{3.1}$$

where C^∞ diffeomorphism means an invertible function from one manifold to another manifold, such that both the function and its inverse are smooth (infinitely many times differentiable).

Let S be a manifold with a smoothly varying inner product $<,>_p$ defined at each point $p \in S$ for every vector pair at that point. The mapping $g : p \mapsto <,>_p$ is called the Riemannian metric tensor and the manifold S with such a metric is a called a Riemannian manifold. The exact form of this inner product is given later in this section in Equation (3.7).

For the space of probability distributions $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, the most popular Riemannian metric tensor is given by the Fisher information [56, 1]

$$I_{ij}(\boldsymbol{\xi}) = g_{ij}(\boldsymbol{\xi}) = E\left\{\frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i}\frac{\partial \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_j}\right\} = E\left\{-\frac{\partial^2 \ln q(\boldsymbol{\theta}|\boldsymbol{\xi})}{\partial \xi_i \partial \xi_j}\right\}, \quad (3.2)$$

where the last equality is valid given certain regularity conditions [44]. It is also possible to define many other Riemannian metrics for the space of probability distributions, e.g. metrics based on the concept of observed information, called yokes [10]. However, Fisher information is a unique metric for probability distributions in the sense that it is the only metric which is both invariant under transformations of the random variables and covariant under reparametrisations [12, 5].

Finally, it should be noted that information geometry is closely related to the geometries used in the general theory of relativity, where the space-time is modelled as a four-dimensional manifold with Lorentzian metric and many of the concepts presented in this chapter such as metric connections are used, albeit the terminology in general relativity is different [44].

3.1.1 Tangent Spaces and Vector Fields

The straightforward intepretation of vectors as straight lines connecting two different points in Euclidian space does not make sense on Riemannian manifolds. The curvature of the space means there is no global notion of straightness. Because of this, vectors on Riemannian manifolds are defined as tangent vectors, local entities that are free of the global coordinate system [1].

The tangent vector $\mathbf{v}$ at a point $p \in S$ to a curve $\gamma(t)$ for which $\gamma(0) = p$ is defined by

$$\mathbf{v} = \frac{d\gamma}{dt}|_{t=0}. \quad (3.3)$$

The tangent space $T_p \sim \mathbb{R}^n$ at point $p \in S$ is the vector space obtained by combining the tangent vectors (i.e. local linearisations) of all the smooth curves passing through the point. For each coordinate system ϕ there exists a special set of curves $\{\phi_i\}$ along which only one coordinate changes. Such curves are known as coordinate curves and the corresponding functions are known as the coordinate functions. The tangent vectors of coordinate curves at any given point p form the natural basis of the tangent space T_p, and any tangent vector $\mathbf{v} \in T_p$ can be written as a linear combination of the basis vectors [1]. The concept of a tangent space and coordinate curves on Riemannian manifolds is illustrated in Figure 3.1.

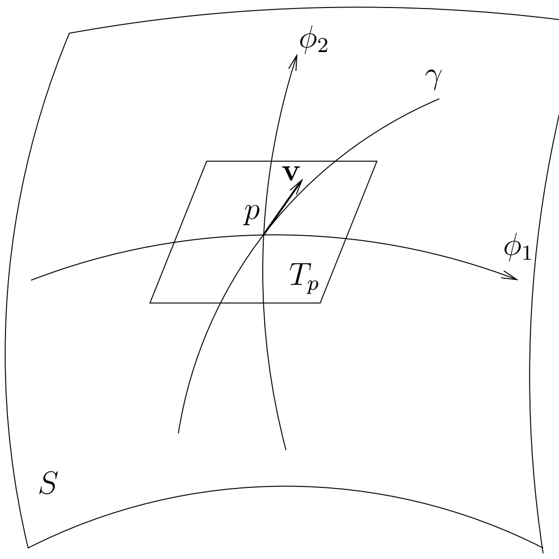


Figure 3.1: Visual presentation of a two dimensional Riemannian manifold. Displayed in the figure are the manifold S, tangent space T_p of the point p, coordinate curves ϕ_1 and ϕ_2 at point p and a curve γ and it is tangent v.

In an Euclidian space $S = \{\mathbf{w} \in \mathbb{R}^n\}$ with orthonormal coordinate system the squared length (also known as the Euclidean norm) of a vector $\mathbf{v}$ is given by

$$\|\mathbf{v}\|^2 = \sum_i v_i^2 = \mathbf{v}^T\mathbf{v}. \tag{3.4}$$

In the case of curved manifold there exists no orthonormal linear coordinates, and (Equation 3.4) is no longer valid. In Riemannian space the squared length of a tangent vector $\mathbf{v} \in T_p$ at point $p \in S$ is given by the quadratic form

$$\|\mathbf{v}\|^2 = \sum_{i,j} g_{ij} v_i v_j = \mathbf{v}^T\mathbf{G}\mathbf{v}, \tag{3.5}$$

where $\mathbf{G} = (g_{ij})$ is the Riemannian metric tensor at point p [44].

In addition to the norm of a tangent vector, we can also define an inner product between two vectors $\mathbf{v} \in T_p$ and $\mathbf{u} \in T_p$. In Euclidean orthonormal space the inner product is given by

$$< \mathbf{v}, \mathbf{u} >= \mathbf{v} \cdot \mathbf{u} = \sum_i v_i u_i = \mathbf{v}^T \mathbf{u}, \tag{3.6}$$

which is independent of the point p. In the general case of Riemannian geometry the inner product is given by

$$< \mathbf{v}, \mathbf{u} >_p = \mathbf{v} \cdot \mathbf{u} = \sum_{i,j} g_{ij} v_i u_i = \mathbf{v}^T \mathbf{G} \mathbf{u}, \tag{3.7}$$

which unlike the Euclidian equivalent also depends on the point p. In the Euclidian orthonormal case $\mathbf{G} = \mathbf{I}$ and Equations 3.5 and 3.7 simplify to the Equations 3.4 and 3.6, as should be expected [1]. Since inner product is conjugate symmetric, $\mathbf{v} \cdot \mathbf{u} = \mathbf{u} \cdot \mathbf{v}$ for real-valued vectors also in Riemannian space.

In addition to single vectors on manifolds, it also useful to define vector fields, i.e. vector valued functions. Formally, a vector field $A(p) \in T_p$ is a mapping from the manifold S to T_p, which assigns a vector $A(p) \in T_p$ to each point $p \in S$.

3.1.2 Connections and Parallel Transport

Given a curve $\gamma : [0, 1] \mapsto S$, its length d is given by

$$d = \int dt \sqrt{\sum_{i,j} g_{ij} \frac{d\phi_i(\gamma(t))}{dt} \frac{d\phi_j(\gamma(t))}{dt}}, \tag{3.8}$$

where ϕ_i are the coordinate functions. The minimiser of this distance over all curves connecting two points

$$d_{\min} = \min_{\gamma} \int dt \sqrt{\sum_{i,j} g_{ij} \frac{d\phi_i(\gamma(t))}{dt} \frac{d\phi_j(\gamma(t))}{dt}}. \tag{3.9}$$

is the (Riemannian) distance between the two points, and the corresponding curve γ is a metric connection, as discussed later in this section [44].

In addition to the simple concept of length, it is often useful to measure the rate of change in vector fields along a curve. There is one major complication, however. In Riemannian space it is meaningless to directly compare two tangent vectors $\mathbf{v}_p$ and $\mathbf{v}_{p'}$ if the points p and p' are different, as the basis vectors for the two points are normally not the same. However, it is possible to derive a linear mapping Φ that allows the comparison of two tangent vectors from different tangent spaces. Let $\{\gamma_\mu\}$ be the set of curves passing through point $p \in S$ and $\mathbf{e}_\mu$ the tangent vector of curve γ_μ at point p. Furthermore, let $\{p'\}$ be the points near p which satisfy $p' = \gamma_\mu(\delta t)$ for some curve γ_μ and small $\delta t > 0$. Now we can define $\Phi^p_{\mu,\delta t}$ as the linear mappings from p' to p which reduce to identity as $\delta t \to 0$. Because of linearity, these mappings are determined by their actions on coordinate vectors in points p and p'

$$\Phi_{\mu,\delta t} : e^{\mu,\delta t}_\rho \mapsto \Phi^\nu_{\mu,\delta t}(e^{\mu,\delta t}_\rho)e_\nu, \tag{3.10}$$

for each $\nu = 1 \ldots n$ where $\{e^{\mu,\delta t}_\rho\}$ and $\{e_\nu\}$ are the coordinate basis vectors at points p' and p, respectively, and $\Phi^\nu_{\mu,\delta t}$ is the νth component of the linear mapping. Because of the property that these mappings reduce to identity as $\delta t \to 0$, we can also write for small δt

$$\Phi_{\mu,\delta t}(e^{\mu,\delta t}_\nu) - e_\nu = \delta t \Gamma^\rho_{\mu\nu} e_\rho, \tag{3.11}$$

where the constants $\Gamma^\rho_{\mu\nu}$ are known as the Christoffel symbols or the coefficients of the affine connection [44, 1].

Analogous to a scalar derivative, we can now define the covariant derivatives [1] of e_ν as

$$\nabla_\mu e_\nu = \lim_{\delta t \to 0} \frac{\Phi^p_{\mu,\delta t}(e^{\mu,\delta t}_\nu) - e_\nu}{\delta t} = \Gamma^\rho_{\mu\nu} e_\rho. \tag{3.12}$$

For a scalar function f, the covariant derivative is simply the ordinary derivative

$$\nabla_\mu f = \partial_\mu f. \tag{3.13}$$

After some manipulation, the covariant derivative of a vector field A is given by

$$\nabla_\mu A = (\partial_\mu A^\rho + \Gamma^\rho_{\mu\nu} A^\nu) e_\rho. \tag{3.14}$$

Using the definition of covariant derivative, we can now define a process known as parallel transport along a curve, which can be used to compare vectors from different tangent spaces along a curve. Formally, a vector field $A(p) \in T_p$ is said to be parallelly transported along a curve γ with tangent vector field $B(p)$ if

$$\nabla_B A = 0. \tag{3.15}$$

A curve γ which parallelly transports tangent vector field to itself is called an affine geodesic. Formally, curve γ is an affine geodesic if

$$\nabla_A A = 0, \tag{3.16}$$

for some parametrisation of the curve for all the points along the curve [1].

The process of parallel transport is illustrated in Figure 3.2. In this work a parallelly transported version of vector $\mathbf{v}$ is denoted by $\tau\mathbf{v}$, where the two tangent spaces are assumed to be defined by the context.

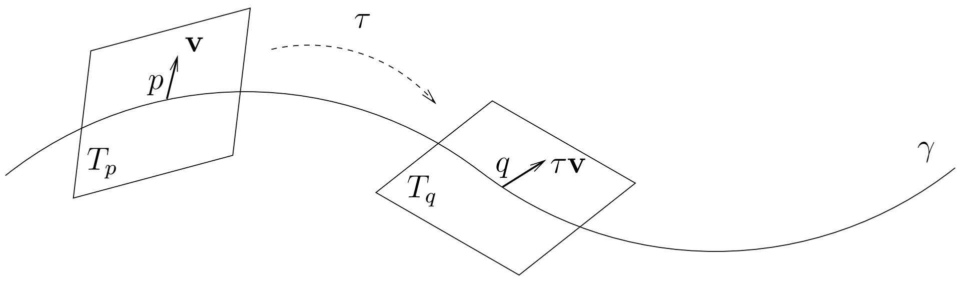


Figure 3.2: The concept of parallel transport. Vector $\mathbf{v}$ is translated from point p to point q along a curve γ on a two-dimensional Riemannian manifold.

Parallel transport has several important and quite intuitive properties, which make it useful for generalising many algorithms and concepts to Riemannian manifolds. First of all, tangent vectors of the geodesic curve remain tangent vectors under parallel transport, as the entire tangent vector space is translated. Moreover, inner product of vectors is invariant under parallel transport for metric connections, which also means that the length of a vector does not change when it is transported parallelly [1].

A curve is a geodesic if it locally minimises the distance between the points of its path. A geodesic is said to be metric if it also gives the shortest distance between two points in the sense of the Equation (3.8). There is a subclass of metric geodesics that are also affine geodesics, these geodesics are known as metric connections. Metric connections that are in addition symmetric have a very important role in differential geometry and they are known as Riemannian connections or Levi-Civita connections [1, 44]. In the case of Fisher metric, Amari's $\alpha = 0$-connections are also Riemannian [5]. The important property of metric connections is the fact that they directly impose a metric. The distance between two points in Riemannian space is given by the length of the shortest path between them, and this path is equal to the metric connection [44, 1].

In addition to Riemannian (metric) connections, there are two more classes of connections that have special importance. These are the e-connection (the exponential connection or the $\alpha = 1$-connection of Amari) and the m-connection (mixture connection or $\alpha = -1$-connection of Amari). The importance of these connections derives from the fact that the canonical parametrisations of exponential family and mixture family distributions are flat with respect to e- and m-connection, respectively [5].

3.1.3 Variational Approximation as a Geometric Projection

The variational approximation has a natural interpretation in information geometry. The approximation of the posterior distribution with another tractable distribution corresponds to finding an approximation of the true posterior $p \in S$ in a submanifold $S_0 \subset S$. Optimal approximation is the projection of p on S_0. In Riemannian space there are multiple such projections, the most important are the e-projection

$$q_e(\boldsymbol{\theta}|\boldsymbol{\xi}) = \arg\min_{q \in S_0} D_{KL}(q(\boldsymbol{\theta}|\boldsymbol{\xi})||p(\boldsymbol{\theta}|\mathbf{X})) \tag{3.17}$$

and the m-projection

$$q_m(\boldsymbol{\theta}|\boldsymbol{\xi}) = \arg\min_{q \in S_0} D_{KL}(p(\boldsymbol{\theta}|\mathbf{X})||q(\boldsymbol{\theta}|\boldsymbol{\xi})), \tag{3.18}$$

which are defined by the e- and m-connections, respectively. Both of these projections correspond to minimising the Kullback-Leibler divergence, but with the order of the distributions reversed. The m-projection is the unbiased maximum likelihood estimator, but unfortunately its computation involves integration over the posterior and it is therefore intractable in most cases. Variational approximation uses the biased e-projection instead [67, 27].

3.2 Natural Gradient

The problem of optimising a scalar function arises in many different fields. In the case of variational Bayes, the goal is to maximise the lower bound on marginal log-likelihood (or alternatively, minimise the Kullback-Leibler divergence). A simple solution to this problem is given by the method of steepest descent. Let $\mathcal{F}(\boldsymbol{\xi})$ be a scalar function defined on the manifold $S = \{\boldsymbol{\xi} \in \mathbf{R}^n\}$. The direction of steepest descent is defined to be the vector $\mathbf{w}$ which minimises $\mathcal{F}(\boldsymbol{\xi} + \mathbf{w})$ under the constraint $|\mathbf{w}|^2 = \epsilon^2$ for sufficiently small constant ϵ.

In the case of Euclidian space, the direction of steepest descent is equal to negative gradient, and the method of steepest descent can be written as follows

$$\boldsymbol{\xi}_n = \boldsymbol{\xi}_{n-1} - \mu \nabla \mathcal{F}(\boldsymbol{\xi}_{n-1}), \tag{3.19}$$

where $\nabla\mathcal{F}(\boldsymbol{\xi}_n)$ is the current gradient and μ is the step size, which can be computed with line search or adaptively adjusted. The iteration is repeated until satisfactory convergence has been reached. However, in the case of Riemannian geometry, negative gradient is no longer the direction of steepest descent; it is replaced by natural gradient [3]

$$\tilde{\nabla}\mathcal{F}(\boldsymbol{\xi}) = \mathbf{G}^{-1}(\boldsymbol{\xi})\nabla\mathcal{F}(\boldsymbol{\xi}), \tag{3.20}$$

where $\mathbf{G}$ is the Riemannian metric tensor and $\nabla\mathcal{F}(\boldsymbol{\xi})$ is the normal gradient. Therefore, natural gradient descent algorithm is given by

$$\boldsymbol{\xi}_n = \boldsymbol{\xi}_{n-1} - \mu\tilde{\nabla}\mathcal{F}(\boldsymbol{\xi}_{n-1}). \tag{3.21}$$

In theory, there are some additional details that should be taken into account. Most importantly, if line search is used, it should use the geodesics of the Riemannian manifold instead of the Euclidian straight lines as discussed in Section 4.3 where Riemannian conjugate gradient method is presented. However, many of the implementations and much of the theoretical work on natural gradient ignores these complications since the derivation of the geodesics can be a very difficult problem.

Natural gradient descent typically converges much faster than normal gradient descent in non-Euclidian spaces. In particular, natural gradient algorithm is able to avoid many of the plateau phases encountered in normal gradient descent. It has also been shown that online natural gradient learning is Fisher-efficient [3, 57, 36].

3.2.1 Efficient Implementation

The computation of the full $\mathbf{G}$ matrix is a very involved process, and in the case of nonlinear state-space models where the dimensionality of the problem space can be very high, even the inversion of the full matrix required for the computation of the natural gradient can be prohibitively costly. Luckily with parametric distributions, parameters associated with different variables are often assumed independent, which results in a block diagonal $\mathbf{G}$. Such a matrix can be inverted efficiently as long as the block sizes remain relatively small.

Additionally, it is possible to simply ignore some of the dependencies between different parameters while computing the matrix $\mathbf{G}$. This results in an approximation of $\mathbf{G}$, but in many cases even this approximation can result in significant speedups compared to gradient descent with very small computational overhead.

3.2.2 Normal Family

As an example, we derive some basic properties of the univariate normal distribution in Riemannian geometry. The canonical parametrisation of the normal distribution is given by

$$p(\theta_1, \theta_2) = \exp(x^2\theta_1 + x\theta_2 - K(\theta_1, \theta_2)), \tag{3.22}$$

where $\theta_1 = \frac{-1}{2\sigma^2}$, $\theta_2 = \frac{\mu}{\sigma^2}$ and $K(\theta_1, \theta_2) = \frac{1}{2}\log(\frac{-\pi}{\theta_1}) - \frac{\theta_2^2}{4\theta_1}$. Even though the canonical coordinates imposed by this parametrisation have some important geometric properties [44], we concentrate on the more traditional parametrisation of the normal distribution

$$p(x|\mu, v) = \frac{1}{\sqrt{2\pi v}} \exp\left(\frac{-(x-\mu)^2}{2v}\right). \tag{3.23}$$

For this parametrisation $N[x, \mu, v]$, we have

$$\ln p(x|\mu, v) = -\frac{1}{2v}(x-\mu)^2 - \frac{1}{2}\ln(v) - \frac{1}{2}\ln(2\pi). \tag{3.24}$$

Further,

$$E\left\{-\frac{\partial^2 \ln p(x|\mu, v)}{\partial \mu^2}\right\} = E\left\{\frac{1}{v}\right\} = \frac{1}{v}, \tag{3.25}$$

$$E\left\{-\frac{\partial^2 \ln p(x|\mu, v)}{\partial v \partial \mu}\right\} = E\left\{\frac{m-x}{v^2}\right\} = 0, \tag{3.26}$$

and

$$E\left\{-\frac{\partial^2 \ln p(x|\mu, v)}{\partial v^2}\right\} = E\left\{\frac{(x-\mu)^2}{v^3} - \frac{1}{2v^2}\right\} = \frac{1}{2v^2}, \tag{3.27}$$

where identity $E\{(x-\mu)^2\} = v$ is used.

The resulting Fisher information matrix is diagonal and its inverse is given simply by

$$\mathbf{G}^{-1} = \begin{pmatrix} v & 0 \\ 0 & 2v^2 \end{pmatrix}. \tag{3.28}$$

Another important parametrisation is given by parametrising variance on log-scale. For the repametrisation $N[x, m, \exp(2v)]$, we have

$$\ln p(x|m, v) = -\frac{1}{2}(x-m)^2 \exp(-2v) - v - \frac{1}{2}\ln(2\pi). \tag{3.29}$$

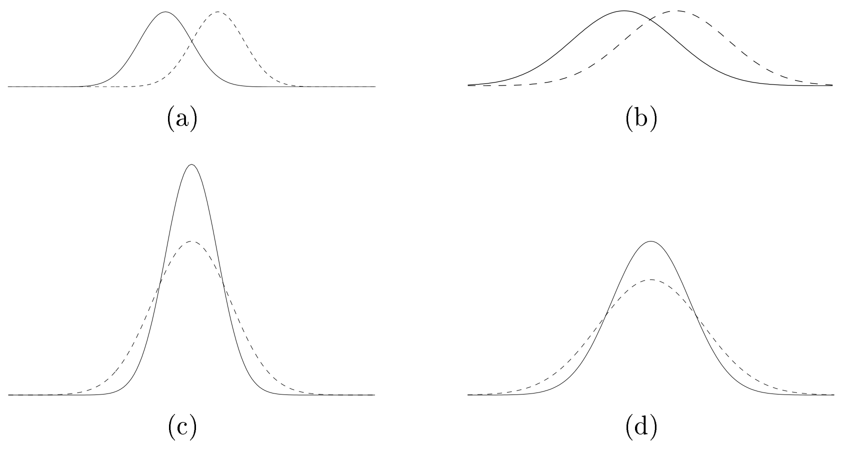


Figure 3.3: The amount of change in mean in figures (a) and (b) and the amount of change in variance in figures (c) and (d) is the same. However, the relative effect is much larger when the variance is small as in figures (a) and (c) compared to the case when variance is high as in figures (b) and (d) [69]

.

and

$$E\left\{-\frac{\partial^2 \ln p(x|m,v)}{\partial m \partial m}\right\} = E\{\exp(-2v)\} = \exp(-2v), \tag{3.30}$$

$$E\left\{-\frac{\partial^2 \ln p(x|m,v)}{\partial v \partial m}\right\} = E\left\{2(x-m)\exp(-2v)\right\} = 0, \tag{3.31}$$

and

$$E\left\{-\frac{\partial^2 \ln p(x|m,v)}{\partial v \partial v}\right\} = E\left\{2(x-m)^2\exp(-2v)\right\} = 2. \tag{3.32}$$

For normal distribution with log-scale variance the Fisher information matrix is again diagonal and its inverse is given by

$$\mathbf{G}^{-1} = \begin{pmatrix} \exp(-2v) & 0 \\ 0 & 2 \end{pmatrix}. \tag{3.33}$$

Intuitively, these results can be interpreted as follows. When the variance of a Gaussian distribution is large, the relative effect of a change in the mean is smaller than when the variance is small as shown in Figure 3.3 [69]. Likewise, when the variance of the Gaussian distribution is large, the relative effect of the change in the variance is much smaller than when the variance is small.

In addition to the Riemannian metric tensor, some other important results can also be derived for the normal distribution. Only the results are given here, for detailed derivation see e.g. [64]. For a normal distribution $N[x, \mu, \sigma^2]$ the Riemannian distance $d(\theta_1, \theta_2)$ between two distributions $\theta_1 = (\mu_1, \sigma_1^2)$ and $\theta_2 = (\mu_2, \sigma_2^2)$ is given by

$$d(\theta_1, \theta_2) = \sqrt{2} \cosh^{-1}(((\mu_1 - \mu_2)^2 + 2(\sigma_1^2 + \sigma_2^2))/4\sigma_1\sigma_2). \tag{3.34}$$

The geodesic curve connecting the two distributions is given by

$$\begin{aligned} \mu(t) &= c_1 + 2c_2 \tanh(t/\sqrt{2} + c_3) \\ \sigma(t) &= \sqrt{2}c_2 \cosh^{-1}(t/\sqrt{2} + c_3) \end{aligned} \tag{3.35}$$

when $\mu_1 \neq \mu_2$, where $\{c_i\}$ are constants that satisfy $\mu(0) = \mu_1$ and $\sigma(0) = \sigma_1(0) = \sigma_1$ and that for some value of the geodesic length t $\mu(t) = \mu_2$ and $\sigma(t) = \sigma_2$. Likewise when $\mu_1 = \mu_2 = \mu$, the geodesic is given by

$$\begin{aligned} \mu(t) &= \mu_1 \\ \sigma(t) &= exp(t/\sqrt{2} + c), \end{aligned} \tag{3.36}$$

where c is a constant that satisfies the same conditions [1].

These results can also be extended to multivariate Gaussian distributions, detailed results and derivations can be found in e.g. [64]. The presence of geodesics in simple analytic form is important for practical implementation of optimisation algorithms. One such example is explored in Section 4.3, where the Riemannian conjugate gradient is introduced.

3.2.3 Related Work

Natural gradient learning has been applied to a wide variety of problems such as independent component analysis (ICA) [4, 3] and MLP networks [3] as well as to analyze the properties of general EM [2], mean-field variational learning [67], and online variational Bayesian EM [60]. Riemannian conjugate gradient has also been applied to a variety of different problems, in particular different eigen-like problems [17, 16]. However, in all these works the geometry is based on the true posterior $p(\boldsymbol{\theta}|\mathbf{X})$ whereas this work uses the geometry of the approximation of the posterior $q(\boldsymbol{\theta}|\boldsymbol{\xi})$, which can often result in greatly simplified computations.

Another alternative to the traditional EM algorithm is expectation-conjugate-gradient (ECG) algorithm [59]. It is rather interesting that ECG algorithm has several similarities with the Riemannian conjugate gradient method presented in Section 4.3, even though the theoretical background of the two algorithms is quite different.

Chapter 4

Conjugate Gradient Methods

Natural gradient algorithm presented in Section 3.2 typically converges much faster than the normal gradient descent algorithm. Unfortunately, in high dimensional problems both algorithms tend to take multiple consecutive steps in almost the same direction. Natural gradient algorithm alleviates this problem to some extent, but much better solution to the problem is given by conjugate gradient method. The seminal paper on nonlinear conjugate gradient is [18], and textbook introductions to conjugate gradient method include [61, 22]. A more intuitive description of the algorithm can be found in [63].

This chapter starts by reviewing the concepts of conjugate directions and the conjugate gradient method in Section 4.1. Some important implementation details are discussed in Section 4.2. In Section 4.3 conjugate gradient method is extended to Riemannian space resulting in the natural conjugate gradient method, also known as the Riemannian conjugate gradient method. Finally, some alternative algorithms with superlinear convergence are presented in Section 4.4.

4.1 Introduction to Conjugate Gradient Algorithm

Even though the gradient descent and natural gradient descent algorithms presented in Section 3.2 can find a local minimum for almost any optimisation problem, they have some shortcomings that make them impractical for many real world optimisation problems. First of all, they only make use of the first order information of the function $f(x)$, and their convergence is therefore quite slow compared to more advanced methods, especially near the local minimum. Additionally, gradient descent algorithms often tend to take multiple steps in almost the same direction, slowing down the convergence. The conjugate gradient and the Riemannian conjugate gradient methods try to solve both these problems.

The conjugate gradient algorithm [25, 22] is the standard tool in numerical optimisation for solving high dimensional systems of linear equations of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.1}$$

where $\mathbf{b}$ is a known vector, $\mathbf{A}$ is a known square, symmetric, positive-definite matrix, and $\mathbf{x}$ is the unknown vector to be solved. For a symmetric positive definitive matrix, this problem is equal to the problem of minimising the quadratic form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}. \tag{4.2}$$

The conjugate gradient method can also be generalised to nonlinear problems where $f(\mathbf{x})$ is no longer quadratic [18], but the performance of nonlinear gradient methods is typically best when $f(\mathbf{x})$ is close to quadratic.

4.1.1 Conjugate Directions

Given a matrix $\mathbf{A}$, two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be $\mathbf{A}$-orthogonal or conjugate (with respect to $\mathbf{A}$) if

$$\mathbf{u}^T\mathbf{A}\mathbf{v} = 0\ . \tag{4.3}$$

It should be noted that this notion of conjugacy has no connection to complex conjugates. Before proceeding to conjugate gradient method itself, the method of conjugate directions is explored. Even though there is no way to efficiently compute a sequence of orthogonal search directions and step sizes, it is possible to generate a sequence of $\mathbf{A}$-orthogonal search directions by a process known as Gram-Schmidt conjugation.

Given a sequence of n conjugate directions $\{\mathbf{p}_k\}$, the solution to the Equation (4.1) is simply given by

$$\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{p}_i, \tag{4.4}$$

where

$$\alpha_i = \frac{\mathbf{p}_i^T\mathbf{b}}{\mathbf{p}_i^T\mathbf{A}\mathbf{p}_i}. \tag{4.5}$$

4.1.2 Conjugate Gradient Method

The conjugate gradient method uses a clever way to construct a sequence of conjugate directions. The current search direction is generated by conjugation of the residuals. With this choice the search directions form a Krylov subspace and only the previous search direction and the current gradient are required for the conjugation process, greatly reducing both the time and space complexity of the algorithm [48].

The conjugate gradient method starts out by searching in the direction of the negative gradient during the first iteration. The optimum in the search direction is determined by line search. On subsequent iterations the search direction $\mathbf{p}_k$ is determined by

$$\mathbf{p}_k = -\mathbf{g}_k + \beta \mathbf{p}_{k-1}, \tag{4.6}$$

where $\mathbf{g}_k = \nabla f(\boldsymbol{\xi}_k)$ is the current gradient and $\mathbf{p}_{k-1}$ is the search direction from the previous iteration. For nonlinear conjugate gradient method, there are several different ways, however, to choose the multiplier β_k. These include the Fletcher-Reeves formula [18]

$$\beta_k = \frac{\mathbf{g}_k \cdot \mathbf{g}_k}{\mathbf{g}_{k-1} \cdot \mathbf{g}_{k-1}} \tag{4.7}$$

and the Polak-Ribiére formula [50]

$$\beta_k = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1}) \cdot \mathbf{g}_k}{\mathbf{g}_{k-1} \cdot \mathbf{g}_k}, \tag{4.8}$$

where $\mathbf{g}_k$ is the current gradient and $\mathbf{g}_{k-1}$ is the gradient from the previous iteration. In most problems the performance with Polak-Ribiére formula is superior to Fletcher-Reeves formula [48], and it is also exclusively used in all the experiments in this work. There is however a minor complication with Polak-Ribiére formula. β_k may become negative and thus the algorithm is not guaranteed to converge. Luckily, there is a simple solution to this problem. The global convergence of the algorithm to a local minimum can be guaranteed by setting $\beta_k = \max(\beta_k, 0)$, which effectively means that the search direction is reverted back to the negative gradient whenever a non-positive value of β_k is encountered. Another way to ensure the global convergence of the Polak-Ribière conjugate method is to use a line search algorithm that satisfies stronger conditions than the usual Wolfe conditions [23], the conditions typically used to ensure the efficient convergence of line search subroutines.

4.2 Implementation

Some care must be taken while implementing a nonlinear conjugate gradient algorithm. This section discusses some potential problems and their solutions. In particular, the search directions tend to lose conjugacy after too many iterations, which can significantly slow down the convergence rate of the algorithm.

4.2.1 Resetting the Search Direction

When applied to a linear problem and assuming infinite precision floating point arithmetic, conjugate gradient algorithm will converge in at most n steps, where n is the number of dimensions of the problem [63]. Unfortunately this property no longer holds when the problem is nonlinear or numeric errors caused by finite floating point precision are taken into account. In practice the algorithm may have to be iterated many more than n times. Over time the search directions tend to lose conjugacy and it is therefore recommended to periodically reset the search direction to the negative of the gradient to improve convergence. This can done at fixed intervals, values of n or $\sqrt{n}$ are typically suggested in literature [63] depending on the size of the problem.

Another solution is to monitor the orthogonality of the subsequent gradients and adaptively decide when the search direction should be reset. This solution is known as Powell-Beale restarts [51] and one such possible restart condition is given by

$$|\mathbf{g}_{k-1} \cdot \mathbf{g}_k| \geq 0.2\|\mathbf{g}_k\|^2, \tag{4.9}$$

where $\mathbf{g}_k$ is the current gradient and $\mathbf{g}_{k-1}$ the gradient from the previous iteration.

4.2.2 Complex Models

For complex models such as high dimensional nonlinear state-space models, it is often beneficial to update the different types of parameters separately from each other, as this is easier to implement and may even speed up convergence in some cases. Unfortunately, the conjugate gradient method relies on information from the previous iteration. Unless all the parameters are updated in a single conjugate gradient step, this information is no longer valid, as there have been changes to the model between conjugate gradient iterations.

The simple solution of updating all the model parameters in a single conjugate gradient iteration can be somewhat problematic however. First of all, this approach can even lead to slower overall convergence caused by scaling issues between different parameters. Finally, it may be useful to use more simple or even exact update formulas for some types of parameters in the model, further discouraging the use of a single conjugate gradient update step. Additionally, if the Riemannian conjugate gradient algorithm presented in Section 4.3 is used, it can be a rather involved process to compute the natural gradients of all the model parameters.

4.3 Riemannian Conjugate Gradient

Up to this point, natural gradient learning and conjugate gradient method have been studied separately. Natural gradient learning works quite well on its own, avoiding most of the shortcomings of the normal gradient descent. However, when only approximations of the natural gradient can be computed, it can be quite beneficial to combine natural gradient and the conjugate gradient methods, as is later shown experimentally. The resulting "natural conjugate gradient" algorithm is known as the Riemannian conjugate gradient [65].

The Riemannian conjugate gradient uses a similar iteration as the conventional conjugate gradient. There are few key differences, however. First of all, the gradient $\nabla f(\mathbf{w})$ must be replaced by the natural gradient $\tilde{\nabla} f(\mathbf{w}) = \mathbf{G}^{-1}\nabla f(\mathbf{w})$. In addition, the vector norms and inner products in Equations (4.8) and (4.9) must be replaced by their generalised counterparts in Riemannian space. Finally, line search must be performed along geodesic curves, which is discussed in more detail in the next section. Many of the formulas used in conjugate gradient method involve vectors from tangent spaces at different points in Riemannian space. To evaluate these formulas, parallel transport must be used to transform the vectors to the same tangent space [65].

In conclusion, the Equations (4.6), (4.8), and (4.9) must be rewritten as follows. The search direction $\mathbf{p}_k$ for Riemannian conjugate gradient method is therefore given by

$$\mathbf{p}_k = -\tilde{\mathbf{g}}_k + \beta\tau\mathbf{p}_{k-1}, \tag{4.10}$$

where $\tilde{\mathbf{g}}_k = \tilde{\nabla} f(\boldsymbol{\xi}_k)$ is the natural gradient and β in the case of Polak-Ribiére formula is given by

$$\beta_k = \frac{(\tilde{\mathbf{g}}_k - \tau\tilde{\mathbf{g}}_{k-1}) \cdot \tilde{\mathbf{g}}_k}{\tau\tilde{\mathbf{g}}_{k-1} \cdot \tilde{\mathbf{g}}_k}, \tag{4.11}$$

and the Powell-Beale restart condition by

$$|\tau\tilde{\mathbf{g}}_{k-1} \cdot \tilde{\mathbf{g}}_k| \geq 0.2\|\tilde{\mathbf{g}}_k\|^2, \tag{4.12}$$

In all these three equations τ denotes parallel transport of the vector from the previous search point to the current search point along the geodesic curve. Additionally, all inner products are taken based on the Riemannian norm. An illustration of the operation of the Riemannian conjugate gradient algorithm can be seen in Figure 4.1 [16, 65].

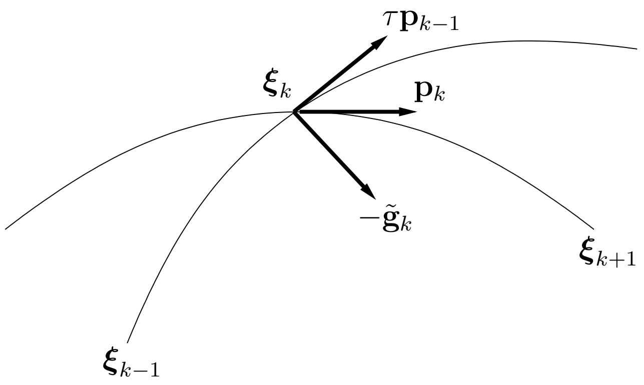


Figure 4.1: Riemannian conjugate gradient algorithm on a curved manifold. Geodesics from two successive iterations and the current gradient $\tilde{\mathbf{g}}_k$, previous search direction (translated using parallel transport) $\tau\mathbf{p}_{k-1}$ and the current search direction $\mathbf{p}_k$ are displayed [16, 65]
.

4.3.1 Line Search Along Geodesics

For an exact Riemannian conjugate gradient algorithm, the line search subroutine also requires certain changes. Even though traditional line search is used in the experiments of this work, the process is reviewed for the sake of completeness. As mentioned earlier, the line search in Riemannian conjugate gradient algorithm is performed along a geodesic curve, the analogue of a straight line in Riemannian space. As long as the geometry of the problem space is such that geodesics can be derived in analytic form, this simply means that the points used in line search subroutine are taken along the geodesic [65].

Unfortunately, even though using geodesics for line search is simple in theory, in practice geodesics and parallel transport may be hard to compute efficiently for many problem spaces. In certain special cases such as normal distribution with suitable parametrisation there exists relatively simple formulas for both geodesics and parallel transport in closed form. However, for more general distributions this is often not the case and various approximations have to be used for implementation.

4.3.2 Limitations

Riemannian conjugate gradient method assumes that the Fisher information matrix, geodesic curves and parallel transport can be computed for the Riemannian manifold of the problem space. Unfortunately, for some problems these may be very time-consuming to derive and compute.

Additionally, the superlinear convergence of Riemannian conjugate gradient algorithm is only guaranteed when exact line search is used. In most cases this is not practical, since in general using exact line search may require infinite computation time. Inexact line search typically leads to good results as well, but such algorithm may converge slowly in certain special cases [65].

4.4 Other Superlinear Algorithms

Conjugate gradient methods have been very successful in solving a large variety of different problems and they are widely used to solve large scale real world problems. However, there are also many other superlinear algorithms that are better suited to certain problems. This chapter gives an overview of some competing superlinear algorithms and compares their strengths and weaknesses with the conjugate gradient method. It is also interesting to note that many of the algorithms presented in this section have a relatively straightforward extension to Riemannian manifolds.

An overview of the different algorithms discussed in this chapter is presented in Table 4.1. It is important to note that many of the superlinear optimisation algorithms require specific conditions to reach their stated convergence rate, and may exhibit linear convergence or fail to converge entirely when these conditions are not met. The listed time and space complexities are only for each step of the algorithm itself, in some cases the computation of the gradients and Hessians can exceed these limits. Finally, when the algorithms are extended to Riemannian space, additional computation is required. This overhead is heavily dependant on the geometry of problem space.

Table 4.1: Optimisation algorithm summary

Method	Convergence	Time complexity	Space complexity
Gradient descent	$O(n)$	$O(n)$	$O(n)$
Conjugate gradient	$O(n^2)$	$O(n)$	$O(n)$
Memory-gradient	$O(n^2)$	$O(n)$	$O(n)$
Scaled conjugate gradient	$O(n^2)$	$O(n)$	$O(n)$
Quasi-Newton	$O(n^2)$	$O(n^2)$	$O(n^2)$
Newton	$O(n^2)$	$O(n^3)$	$O(n^2)$

4.4.1 Scaled Conjugate Gradient

The traditional conjugate gradient algorithm offers fast convergence, but if the computation of the cost function requires significant time, the line search can be quite time consuming. An alternative way to determine the step size is to use a so-called trust region or Levenberg-Marquardt approach. Such variant of the conjugate gradient method is known as the scaled conjugate gradient method. The algorithm itself is rather complex and introduces some new parameters, full details can be found in [43].

The Levenberg-Marquardt approach introduces a new scale term λ_k which forces the approximation of the Hessian to remain positive definite. After the update the quality of the approximation is evaluated, and the parameter is adjusted accordingly. When the λ_k is zero, the algorithm is equal to the traditional Conjugate Gradient method.

The main benefit of the Scaled Conjugate Gradient method is the fact that it requires only constant number of cost function and gradient evaluations per iteration. In the optimal case, the line search in the traditional Conjugate Gradient method requires similar run time as the Scaled Conjugate Gradient method. In practice, standard conjugate gradient method with good line search subroutine requires two to three times more cost function and gradient evaluations compared to the Scaled Conjugate Gradient method.

There are some issues with the Scaled Conjugate Gradient method, however. First of all, some scaled conjugate gradient iterations are spent adjusting the scale parameter without any reduction in the cost function even though full gradient and cost function evaluations are required for these iterations as well. In addition, the step sizes are less optimal than with line search, which leads to faster loss of conjugacy of the search directions. Finally, whereas a conjugate gradient algorithm is easy to implement, the scaled-conjugate gradient algorithm is relatively complex and relies on certain parameter values that must be chosen during the implementation.

4.4.2 Memory Gradient

Conjugate gradient algorithm uses information from two iterations to approximate the Hessian matrix. It is also possible to store and utilise gradient and search direction information from more than two iterations to better approximate the higher order information of the optimised function.

Based on this idea, a class of algorithms has been developed that try to improve the performance of gradient based algorithms without significantly increasing the computational complexity. These algorithms include memory gradient [40] and the three-term-recurrence algorithm [45], both of which take into account search direction information from several past iterations.

Compared to conjugate gradient methods, these algorithms require more memory overhead, and are more difficult to implement than the simple conjugate gradient. Even though they provide some performance advantages over conjugate gradient, neither has been studied as widely nor enjoys the same popularity as conjugate gradient method.

4.4.3 Newton's Method

The algorithms presented so far in this chapter do not directly use the higher order information of the function. There also exists a wide class of algorithms that directly use this higher order information, however. The most popular of these algorithms are Newton's method and its various approximations, known as quasi-Newton algorithms. All these algorithms provide superlinear convergence near the local minimum. Unfortunately, these algorithms often have rather limited region of convergence, and typically other methods such as gradient descent are used to initialise the iteration. Another alternative is the Levenberg-Marquardt method, a robust algorithm that combines Newton's method and gradient descent [38].

Newton's method has also been generalised to Riemannian manifolds [65, 66, 73]. Newton-like algorithms have one typical problem while solving high-dimensional problems, however. They require matrix operations with $n \times n$

matrices, where n is the dimension of the problem space. In many high-dimensional problems, this is not computationally feasible, as for example the problem space of a NSSM may well have dimensionality of $n > 10000$. Matrix operations during each optimisation step with matrices of this size are typically not feasible even with state-of-the-art algorithms and hardware.

When the dimensionality of the problem space is slightly smaller, Newton-based algorithms can provide a viable alternative to conjugate gradient methods. Of particular interest are limited memory Newton algorithms [48], which have partially replaced conjugate gradient methods in problems with slightly lower dimensionality. Conjugate gradient methods, however, are still the best choice for very high dimensional problems because of their modest computation and memory demands. Conjugate gradient methods are also relatively easy to implement and more suitable to parallel computation than many competing algorithms.

Chapter 5

Nonlinear State-Space Models

Nonlinear state-space models (NSSM) are one particularly important class of probabilistic models. In this chapter NSSMs are presented as a case study for natural gradient learning, and in particular the NSSM from [71] is discussed in more detail.

General NSSM structure and the building blocks of the model are discussed in Section 5.1. The NDFA model from [71] is presented as an example of an NSSM in Section 5.2. Finally, implementation details of the conjugate gradient and Riemannian conjugate gradient methods for the NDFA model are discussed in Section 5.3.

5.1 Model Structure

State-space models are one popular way to model dynamical systems. Instead of modelling the dynamics of the observed time-series $\mathbf{X} = \{\mathbf{x}(t)\}$ directly, state-space models use a set of hidden states $\mathbf{S} = \{\mathbf{s}(t)\}$ to model the dynamics. Furthermore, the mapping that maps the states back to the actual observations is modelled. The states form a so-called state-space, hence the name of the model.

5.1.1 Linear State-space Model

The simplest state-space model is the linear state-space model

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t), \tag{5.1}$$

$$\mathbf{s}(t) = \mathbf{B}\mathbf{s}(t-1) + \mathbf{m}(t), \tag{5.2}$$

where $\mathbf{x}(t)$ are the observations and $\mathbf{s}(t)$ are the hidden internal states of the system. The vectors $\mathbf{m}(t)$ and $\mathbf{n}(t)$ are the process and observation noise, respectively. $\mathbf{A}$ and $\mathbf{B}$ define the linear observation and dynamic mappings. The observations $\mathbf{X}$ and the states $\mathbf{S}$ are assumed to be real-valued and the time t is discrete.

In practice, linear model for the dynamics is too restrictive. The behaviour of a linear dynamical system is defined by the eigenvalues of the matrix $\mathbf{A}$, and there is only a very restricted set of possible outcomes. This is insufficient for modelling any but the most basic real-world systems [7].

5.1.2 Nonlinear models

In principle, it is relatively straightforward to extend a linear state-space model into a nonlinear one. It is simply enough to replace the linear mappings by generic nonlinear mappings, resulting in the model

$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_f) + \mathbf{n}(t) \tag{5.3}$$

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_g) + \mathbf{m}(t), \tag{5.4}$$

where $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ are the vectors containing the model parameters which define the mappings $\mathbf{f}$ and $\mathbf{g}$, respectively. The dependence of the mappings $\mathbf{f}$ and $\mathbf{g}$ on the model parameters $\boldsymbol{\theta}$ is assumed for the rest of this text, even though it is not explicitly shown for reasons of clarity. Only the observations $\mathbf{x}(t)$ are known beforehand, and both the states $\mathbf{s}$ and the mappings $\mathbf{f}$ and $\mathbf{g}$ are learned from the data.

Assuming that the mappings $\mathbf{f}$ and $\mathbf{g}$ are modelled in a generic enough way, nonlinear state-space models are generic enough to model any time-series. The addition of nonlinearity can also give rise to chaotic effects. Over long time periods, even small changes in the states can lead to complitely different outcomes.

5.1.3 Modelling Nonlinearities

One major problem while implementing a nonlinear model is the representation of the nonlinearities. Whereas linear mappings can simply be represented by matrices, there is no such easy solution for generic nonlinear functions. Luckily, there exist different function approximators that can approximate any function to a desired accuracy given enough parameters. The most well-known of these are the various series decompositions including polynomial approximations and trigonometric series. Unfortunately, trigonometric series can only be used to model periodic functions and polynomic approximations can be sensitive to very small parameter changes, which makes them a poor choice for learning purposes. In addition, high order polynomic approximations tend to generalise very poorly. Some of these problems can be solved by using splines instead of higher order polynomials [24].

In the field of neural networks, two different function approximations are widely used. These are the radial-basis function (RBF) and multilayer perceptron (MLP) network. Both of them are universal function approximators; given enough parameters (i.e. neurons), they can at least in theory model any function to a desired accuracy [31, 24]. Since the NDFA model described in Section 5.2 and used in the experiments uses MLP networks, the next section describes them in greater detail.

5.1.4 Multilayer Perceptron

A MLP network consists of several simple neurons known as perceptrons. A perceptron is a very simple computation unit that computes a single output from multiple inputs by applying a nonlinear activation function to a linear combination of the inputs. A perceptron can be presented mathematically by the equation

$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T\mathbf{x} + b), \tag{5.5}$$

where $\mathbf{w} = [w_1\ w_2\ \dots\ w_n]^T$ is the weight vector, $\mathbf{x}$ are the inputs, b is the bias and φ is the activation function [24].

In neural networks research, the most common activation functions are the logistic sigmoid $1/(1 + e^{-x})$ and the hyperbolic tangent $\tanh(x)$. These two are closely related and they share the useful property that they exhibit nearly linear behaviour near the origin but become saturated quickly farther away from the origin. This property makes them well suited for modelling both strongly and mildly nonlinear functions [24].

A single perceptron can only represent very limited linearly separable mappings. Therefore large networks of perceptrons are used, as seen in Figure 5.1. MLP networks are usually arranged in several layers with at least one so called hidden layer between the input and the output layers [24].

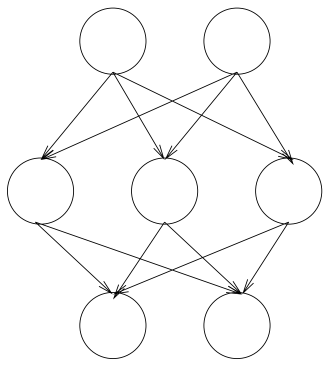


Figure 5.1: MLP network with one hidden layer.

The functional form of a nonlinear state-space model where nonlinearities are modelled with MLP networks with one hidden layer is

$$\mathbf{f}(\mathbf{s}(t)) = \mathbf{B}\tanh[\mathbf{A}\mathbf{s}(t) + \mathbf{a}] + \mathbf{b} \tag{5.6}$$

$$\mathbf{g}(\mathbf{s}(t-1)) = \mathbf{s}(t-1) + \mathbf{D}\tanh[\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}] + \mathbf{d}, \tag{5.7}$$

where $\mathbf{A}$ and $\mathbf{C}$ are the weight matrices for hidden layers, $\mathbf{B}$ and $\mathbf{D}$ are the weight matrices for output layers, and $\mathbf{a}$, $\mathbf{c}$, $\mathbf{b}$, and $\mathbf{d}$ are the corresponding biases [71].

MLP networks are most often used in supervised learning tasks, where the most commonly used learning algorithm is the back-propagation algorithm which iterates between backward and forward passes [24]. In addition, it is possible to derive a nonlinear Kalman filter known as the Extended Kalman Filter (EKF) [6, 24] which can be used to derive the hidden state-space if the observations and the nonlinear mappings are known.

The complete learning of hidden state-space models requires more complex algorithms and is usually much slower than in the case of supervised learning tasks. One such unsupervised learning algorithm is given by Dr. Valpola [71]. In this work this algorithm is extended to take into account the non-Euclidian nature of the space of probability distributions as described in Section 3.2. The algorithm uses MLP networks to model the nonlinearities and is based on variational Bayesian learning, which is discussed in more detail in Section 2.4. Other learning algorithms for nonlinear state-space models include the work of Ghahramani and Roweis [21], which uses RBF networks and standard EM algorithm where EKF is used for the E-step.

5.2 Nonlinear Dynamic Factor Analysis

As an example of a NSSM, nonlinear dynamic factor analysis (NDFA) [71] is used. This particular NSSM uses multilayer perceptron networks with one hidden layer and tanh nonlinearity to model the nonlinear mappings.

The weights of the MLP networks and the other model parameters are all assumed to be independent and they are modelled with Gaussian distributions with diagonal covariance to limit the number of parameters and keep the computation efficient. The state vectors $\mathbf{s}(t)$ are also assumed componentwise independent. The subsequent state vectors are also assumed independent with one exception: the dependence between the corresponding components of $\mathbf{s}(t-1)$ and $\mathbf{s}(t)$ is modeled with a linear dependence parameter $\hat{\mathbf{s}}(t, t-1)$. This correlation is a realistic minimal assumption for modelling a dynamic system [71]. This simple assumption also makes the derivation of a natural gradient algorithm straightforward.

This dynamic model for the parameters and the states leads to the approximation

$$q(\mathbf{S}, \boldsymbol{\theta}) = q(\mathbf{S})q(\boldsymbol{\theta}) \tag{5.8}$$

and

$$q(\boldsymbol{\theta}) = \prod_i q_i(\theta_i), \tag{5.9}$$

and finally

$$q(\mathbf{S}) = \prod_i q_i(s_i(t)|s_i(t-1)), \tag{5.10}$$

where the approximate density $q_i(s_i(t)|s_i(t-1))$ is parametrised by its mean $\overline{s}_i(t)$, linear dependence $\hat{s}_i(t, t-1)$, and variance $\widetilde{s}_i(t)$.

5.3 Riemannian Conjugate Gradient

The implementation of the Riemannian conjugate gradient algorithm is based on the NDFA package [70] presented in [71]. There are some key improvements, however. First of all, the Taylor approximation used for the nonlinearities in [71] can result in stability problems. This problem can be solved by replacing the Taylor approximation by Gauss-Hermite quadratures as described in [26, 28]. The replacement of Taylor approximations with the more complex approximation roughly doubles the computational cost of the algorithm. However, the resulting algorithm tends to converge faster and and it is almost entirely free from the stability problems of the original implementation, so this modification is quite justified.

Additionally, the heuristic update rules from [71] for the states and nonlinear mappings tend to converge slowly. A significant speedup can be attained by replacing these update rules with an efficient direct optimisation algorithm. In this case, the means of the latent states and all the network weights are updated simultaneously using the Riemannian conjugate algorithm with some simplifying assumptions as described later in this section. The local optimum in the search direction is found using a line search subroutine based on polynomic interpolation. The formulas for the gradients of the parameters $q(\mathbf{S})$ and $q(\boldsymbol{\theta})$ required in the computation of the natural gradient can be found in [71]. It is important to note that the natural gradient is computed based on the geometry of the approximating distribution q, whereas tradiationally natural gradient algorithms have been only used for the true posterior distribution.

5.3.1 Used Approximations

To simplify the implementation of the Riemannian conjugate gradient, certain approximations were used. First of all, the component-wise dependency parameter $\hat{s}$ is updated separately from the means and variances to simplify the geometry of the problem space. Typically this parameter can be updated in a single step, so the extra computational cost is not significant.

Additionally, natural gradient learning is only used for the network weights and the sources. The rest of the parameters and hyperparameters are updated by the algorithms described in [71]. It is unlikely that using Riemannian conjugate gradient for all the parameters would have resulted in a significant speedup compared to the current implementation. Usually only the weights and the sources require significant amount of iterations to converge, the other parameters and hyperparameters typically converge relatively fast.

5.3.2 Update Order

In the current implementation of the algorithm, the model parameters and hyperparameters are updated first. This is done for two reasons. First of all, parameter updates can be done separately from the feedforward and backward passes of the sources. Additionally, this update order allows taking into account any external modifications (such as pruning away dead neurons) to the model straight away.

The parameter updates are followed by feedforward and feedback passes, which also include the computation of the cost function C_{KL}. The gradient information from the backward pass is first used to update the variances of the network weights and sources based on fixed point update rule. This is followed by updating the means using a direct update algorithm, in the experiments in this thesis either conjugate gradient or Riemannian conjugate gradient algorithm. Even though variances and means can be updated in a single Riemannian conjugate gradient iteration, updating them separately resulted in a more stable algorithm.

It should be noted that the gradient information is only computed once, even though technically it should be recomputed after the variances have been updated. A full feedforward and backward pass is quite expensive in terms of computation time, and thus small loss of accuracy can be justified here. Intuitively, the change in the variance of a parameter has a smaller effect on the gradient of the mean than vice versa. This was also verified experimentally, thus justifying the chosen update order.

5.3.3 Line Search

Many optimisation algorithms alternate between finding a new search direction and finding the optimum in this direction. The procedure of finding the optimum is known as line search. For linear problems exact line search is often practical, but for nonlinear problems this is typically not the case and inexact line search methods must be used. Therefore the minimum is bracketed either by using a search procedure such as Fibonacci or golden section search or by using polynomial interpolation and extrapolation. When the function to be minimised is continuous, the performance of polynomial interpolation methods is typically superior to other alternatives [52].

In quadratic interpolation a second order polynomial of the form $p(\alpha) = a\alpha^2 + b\alpha + c$ is fitted to the available data points. The extremum of the polynomial can be found at $\frac{-b}{2a}$. Given three known data points $f(x_1)$, $f(x_2)$, and $f(x_3)$ this can be rewritten as

$$x_{min} = \frac{1}{2} \frac{\beta_{23} f(x_1) + \beta_{31} f(x_2) + \beta_{12} f(x_3)}{\gamma_{23} f(x_1) + \gamma_{31} f(x_2) + \gamma_{12} f(x_3)}, \tag{5.11}$$

where $\beta_{ij} = x_i^2 - x_j^2$ and $\gamma_{ij} = x_i - x_j$. To ensure that the extremum is a minimum and that interpolation is performed instead of extrapolation, the condition

$$f(x_2) < f(x_1) \wedge f(x_2) < f(x_3) \tag{5.12}$$

must be satisfied [52].

Whenever gradient information or more than three function evaluations are available, cubic interpolation can be used instead of quadratic interpolation. In cubic interpolation a third order polynomial of the form $p(\alpha) = a\alpha^3 + b\alpha^2 + c\alpha + d$ is fitted to the available data. The local extremum of the polynomial are the roots of the equation

$$3a\alpha^2 + 2b\alpha + c = 0 \tag{5.13}$$

and the local minimum is given by the root with $6a\alpha + 2b > 0$ [52].

Even higher order polynomial interpolation can be used to approximate the function $f(x)$ but the use of higher than third order polynomials often leads to problems with numerical stability, increased computational complexity, and may also result in Runge's phenomenon, the oscillation of the interpolation polynomial near the end points of the interpolation interval. This phenomenon is closely related to Gibbs' phenomenon, a similar problem with sinusoidal basis functions [52].

Before the local minimum has been bracketed, the end points must be adjusted so that the condition (5.12) holds. For a convex function this can be done in a rather simple way by doubling t_3 or halving t_2 and setting the other point to the old value of the adjusted point until both parts of the condition are satisfied.

To speed up this bracketing, a polynomic approximation can be used here as well. Given the interpolated or extrapolated minimum t_{min}, we can set $t_3 = 2t_{min}$ when adjusting the points upwards and $t_2 = t_{min}$ when adjusting the points downwards. To make the extrapolation more robust, only quadratic extrapolation is used. Additional safeguards are also used to limit the minimum and maximum relative change in the line search points.

Chapter 6

Experiments

In this chapter, the conjugate gradient method and Riemannian conjugate gradient method presented in Chapter 4 are applied to three different problems. In each experiment, the nonlinear state-space model presented in Section 5.2 is used to learn a different data set.

In Section 6.1, the method is applied to a synthetic data set generated using random MLPs. In Section 6.2 the method is used to learn the dynamics of the inverted pendulum system, an important benchmark in control theory. Finally, in Section 6.3 the method is applied to the challenging real world data set consisting of human speech.

6.1 Synthetic Data

To compare the performance of conjugate gradient and Riemannian conjugate gradient under different noise levels, the algorithms were applied to multiple randomly generated synthetic data sets.

6.1.1 Data Set

The data sets consisted of 500 samples, which were generated using the generative model defined in Equations (5.3) and (5.4). The mappings were modelled by MLPs with 10 hidden nodes, and all the weights were randomly generated from a Gaussian distribution. The state space was three dimensional, and the generated data was four dimensional.

Two groups of data sets were generated using this method. For the first group, the innovation (process noise) $\mathbf{m}(t)$ variance was kept constant $\sigma_m^2 = 10^{-4}$ and the variance of the observation noise $\mathbf{n}(t)$ was varied. For the second group, the innovation process variance was varied while the observation noise variance was constant $\sigma_n^2 = 10^{-4}$.

6.1.2 Learning

The NSSMs used in this experiment used the same parameters as the original data: three dimensional state-space and MLP networks with 10 hidden nodes.

Initial values of the means of the MLP weights were drawn randomly from the same distribution as the weights of the MLPs used to generate the data. NSSM states were initialized to all zeros. For each different noise level three different initialisations of the parameters were used and those iterations where different algorithms converged to a different local optimum from the same initialisation were ignored.

Iteration was assumed to have converged when $|\mathcal{B}^t - \mathcal{B}^{t-1}| < 10^{-4}$ for 200 consecutive iterations, where $\mathcal{B}^t$ is the bound on the marginal log-likelihood at iteration t.

6.1.3 Results

A comparison of the convergence speed of conjugate gradient and Riemannian conjugate gradient is presented in the Figure 6.1. The heuristic algorithm from [71] suffered from some stability problems with this data set and therefore it was omitted from the results.

At low levels of observation noise $\mathbf{n}(t)$ and process noise $\mathbf{m}(t)$ the performance of regular and Riemannian conjugate gradient algorithms is comparable. As the noise levels increase, the Riemannian algorithm becomes significantly faster while the regular conjugate gradient algorithm benefits less. Still, the effect of the noise variance to convergence speed is sublinear, whereas in theory it would be linear for the EM [49].

The speed difference in the methods in cases of high noise is caused by the fact that there will be more uncertainty on the values of some parameter. Hence there will be greater variation among the posterior variances that determine the inverse Fisher information matrix of Equation (3.28), which will therefore

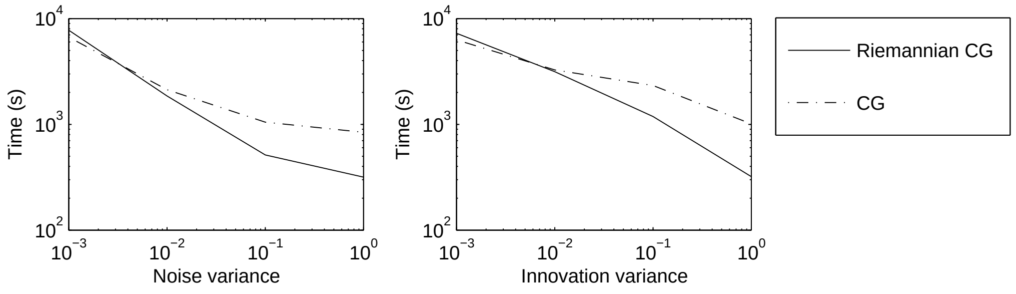


Figure 6.1: The effects of variance on the convergence speed of conjugate gradient (dash-dotted line) and Riemannian conjugate gradient (solid line). The plots show convergence speed with different levels of observation noise $\mathbf{n}(t)$ (left) and convergence speed with different levels of innovations $\mathbf{m}(t)$ (process noise, right).

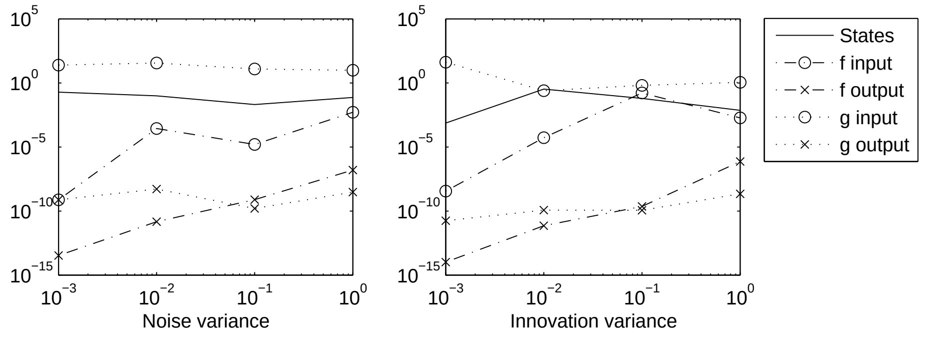


Figure 6.2: The variance of the posterior variances of the latent states and different model parameters plotted against observation noise $\mathbf{n}(t)$ (left) and innovation process $\mathbf{m}(t)$ (right). The variances are shown for the latent states (solid line), input layer weights of the observation mapping $\mathbf{f}$ (dash-dotted line with circles), output layer weights of $\mathbf{f}$ (dash-dotted line with crosses), input layer weights of the dynamical mapping $\mathbf{g}$ (dotted line with circles), and output layer weights of $\mathbf{g}$ (dotted line with crosses).

differ more from the form of constant times identity. This is illustrated in the Figure 6.2, which shows a clear increase in the variance of the estimated posterior variances in situations where regular conjugate gradient is performing badly in comparison to the Riemannian variant.

6.2 Inverted Pendulum System

One very important problem where nonlinear state-space models are used is system identification in the field of control. Typically, observed data and external inputs are available, and the goal is to learn a model for the system from this data. The learned state-space model can then be used in various different control schemes, one popular example is the nonlinear model predictive control (NMPC) method [39].

The state-space model described in Section 5.2 does not directly support control input. However, it is relatively simple to extend the model by augmenting the state matrix $\mathbf{S}$ with the control signals and ensuring that the control states remain constant during the learning process as described in [54, 68].

6.2.1 Data Set

The inverted pendulum system [34], also known as the cart-pole system, is a classic benchmark for nonlinear control and system identification. The system consists of a pole (which acts as an inverted pendulum) attached to a cart (Figure 6.3). The force applied to the cart can be controlled within certain limits. Typical control task for this system is to swing the pole to an upward position and stabilise it. This must be accomplished without the cart crashing into the walls of the track.

The observed variables of the system are the position of the cart s, angle of the pole measured from the upward position ϕ, and their first derivatives s' and ϕ'. Control input is the force F applied to the cart. The dynamics of the system are described by the following differential equations [34]

$$\theta'' = \frac{g \sin\theta + \cos\theta \left(\frac{-F - ml\theta'^2 \sin\theta + \mu_c \mathrm{sgn}(s')}{M+m} \right) - \frac{\mu_p \theta'}{ml}}{l\left(\frac{4}{3} - \frac{m\cos^2\theta}{M+m}\right)} \tag{6.1}$$

$$x'' = \frac{F + ml(\theta'^2 \sin\theta - \theta'' \cos\theta) - \mu_c \mathrm{sgn}(x')}{M+m}, \tag{6.2}$$

where $M = 1.0$ kg is the mass of the cart, $m = 0.1$ kg is the mass of the pole, $l = 0.5$ m is half the length of the pole, $g = 9.8$ m/s^2 is the acceleration of gravity, and $\mu_c = 0.05$ and $\mu_p = 0.01$ are the coefficients of the friction of the cart and the pole respectively.

In this experiment the dynamics of the system are assumed unknown, and a NSSM describing the system is learnt from a set of training data. The data

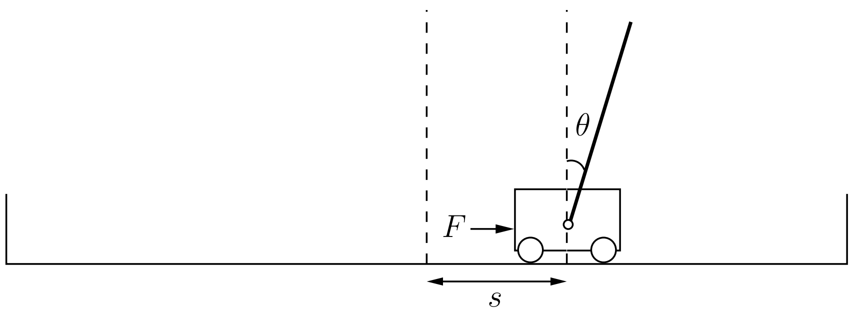


Figure 6.3: The cart-pole system

set was generated by simulating a discrete-time system with a time step of $\Delta t = 0.05$ s. Both observation noise and process noise were Gaussian with variance $\sigma = 0.001$. The possible force was constrained between -10 N and 10 N, and the position between -3 m and 3 m.

The control signal used to generate the training data set was mostly random with some hand-tuned sections to ensure that the entire state-space was sufficiently represented in the training data set. Two different data sets were used, a larger data set with 2500 samples and a smaller data set which contained the first 500 samples of the full data set.

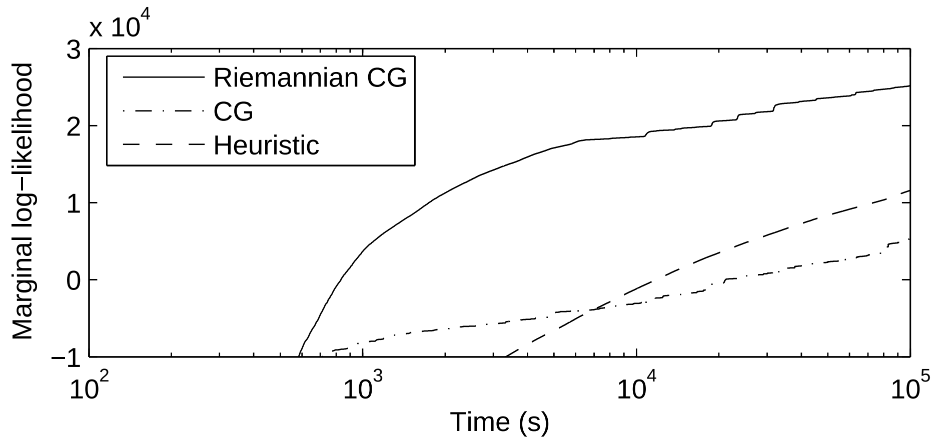
6.2.2 Learning

A six dimensional state-space model was used with the control signal as the seventh state dimension. Both the observation and dynamical mapping were modelled with MLP networks with 30 neurons. Embedding was used to initialise the sources to meaningful values as described in [71]. The sources were initialised to the 6 first principal components of the concatenated data vector $\hat{\mathbf{x}}(t) = [\mathbf{x}^T(t)\ \mathbf{x}^T(t-1)\ \mathbf{x}^T(t-2)\ \mathbf{x}^T(t-4)\ \mathbf{x}^T(t-8)\ \mathbf{x}^T(t-16)]^T$, and this 24-dimensional embedded data vector was used for the 200 first iterations, at which point the data vector and the observation mapping MLP were pruned. These iterations are not displayed in the results of the next section.

Three different initialisations for the other model parameters including MLP weights were used to avoid problems with local minima. The results in the next section are from the initialisation that converged to the best value of the cost function.

6.2.3 Results

The performance of the Riemannian conjugate gradient, the conjugate gradient and the heuristic algorithm from [71] is presented in Figure 6.4.



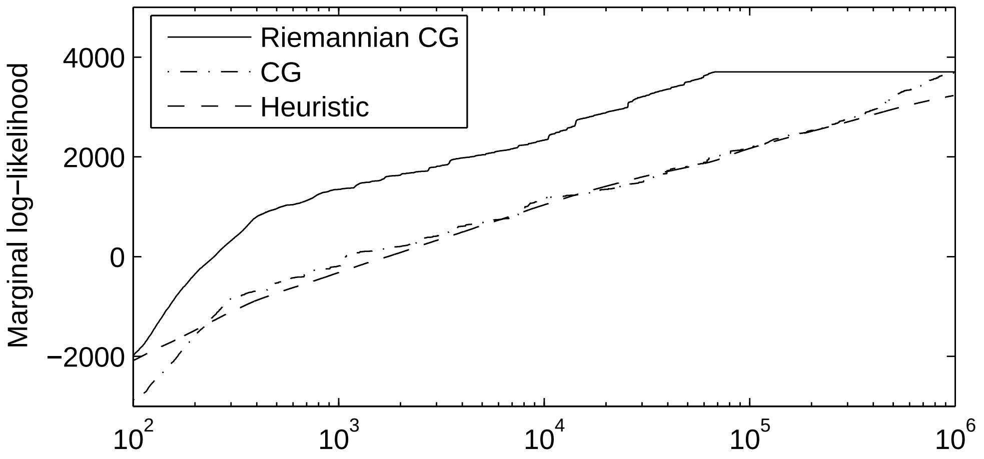


Figure 6.4: Comparison of the performance of the different algorithms with the cart-pole data set using logarithmic scale for the computation time. Results with the full data set are displayed in the top figure, results with the small data set in the bottom figure. The compared algorithms are Riemannian conjugate gradient (solid line), the heuristic algorithm from [71] (dashed line) and conjugate gradient (dash-dotted line).

With the large data set, none of the algorithms converged in reasonable time, but the relative differences between the algorithms are rather large. Riemannian conjugate gradient clearly outperformed the other algorithms in this experiment, and conjugate gradient method in particular performed very poorly. Both the model trained with Riemannian conjugate gradient and the model trained with the heuristic algorithm have also been successfully used in a difficult control task with the simulated cart-pole system as reported in [54, 68].

For the smaller data set, the performance differences between the different algorithms were slightly less pronounced. The performance of Riemannian conjugate gradient remained clearly superior to the other methods, but the performance of the conjugate gradient method and the heuristic algorithm was quite similar in this experiment. Both Riemannian conjugate gradient and conjugate gradient method converged to a similar local minimum with a slightly different values of the cost function. However, it took conjugate gradient algorithm more than 10 times longer to converge. The heuristic algorithm failed to converge in reasonable time in this experiment as well.

At least in this experiment, the smaller dimensionality of the data set reduces the performance advantages of the Riemannian conjugate gradient. A possible explanation for the larger performance difference between the algorithms when the amount of data increases is given by the intuitive intepretation of natural gradient in the space of Gaussian distributions. With a larger data set, the differences in the variances of the parameters will also likely be larger. A gradient based learning algorithm which assumes flat geometry will try to adjust the parameters with low variance too much compared to the variables with high, and this can significantly slow down the overall convergence as all the parameters must be updated in parallel. In contrast, an optimisation algorithm that takes into account the Riemannian nature of the problem space will correctly scale the step sizes so that conflicting updates are less of a problem.

6.3 Speech Data

As a final demonstration of the performance of the algorithm, the Riemannian conjugate gradient method was used to learn a state-space model for high-dimensional real-world data set with complex dynamics.

6.3.1 Data Set

The data set in this experiment consisted of 21 dimensional real world speech data. The data consisted of mel-scaled log power speech spectra. A 2000 sample portion of the original data set was used, the sample contained continuous human speech with no significant pauses. This sample size corresponds to roughly 15 seconds of real time.

It should be noted that for any reasonable dynamical model of human speech, a much larger data set should be used. However, even this relatively small data set is useful for demonstrating the convergence speed of the different algorithms.

6.3.2 Learning

In this problem a NSSM with seven sources was used. Both MLP networks of the NSSM had 30 hidden nodes.

As with inverted pendulum system, the sources were initialised to the first principal components of the embedded data vector $\hat{\mathbf{x}}(t)$. However, because of the high dimensionality of the problem space, embedded data was not used during the learning. It is likely that this made it more difficult to learn meaningful dynamics for the data. However, since the main focus of this experiment was to compare the convergence of the different algorithms, this should not significantly alter the results.

6.3.3 Results

The performance of the original heuristic algorithm presented in [71] was compared with conventional conjugate gradient learning and Riemannian conjugate gradient learning. Unfortunately a reasonable comparison with a variational EM algorithm was impossible due to the extended Kalman smoother [6] being unstable and thus failing the E-step. The results and a part of the data set can be seen in Figure 6.5. Five different initialisations were used to avoid problems with poor local optima. The results presented in Figure 6.5 are from the iterations that converged to the best local optimum.

The results with the speech data are quite similar to the inverted pendulum system results. Riemannian conjugate gradient has a clear performance advantage over the two other algorithms. In particular, conventional conjugate gradient learning converged very slowly in this problem and regardless of initialisation failed to reach a local optimum within reasonable time. Riemannian conjugate gradient also outperformed the heuristic algorithm from [71] by a factor of more than 10.

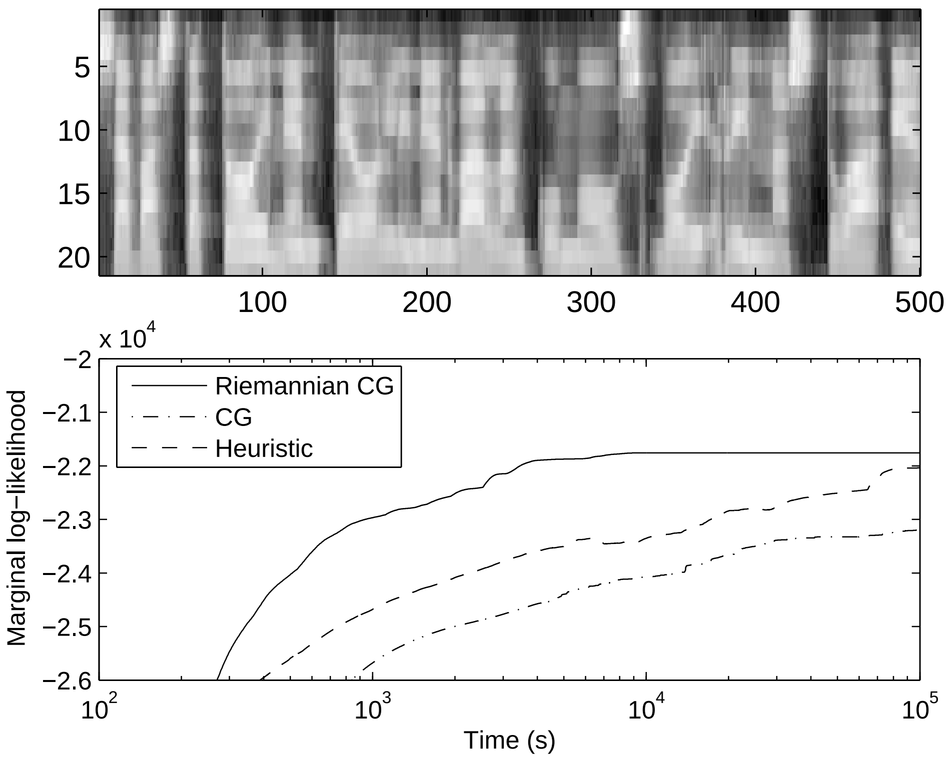


Figure 6.5: Top: Part of the speech spectrum data used in the experiments. Bottom: Comparison of the performance of the different algorithms using logarithmic scale for the computation time. The compared algorithms are Riemannian conjugate gradient (solid line), the heuristic algorithm from [71] (dashed line) and conjugate gradient (dash-dotted line).

Chapter 7

Discussion

It is important to note that the actual implementation of the Riemannian conjugate gradient method is only an approximation of the exact algorithm presented in Section 4.3. However, as the experiments in this thesis show, even this approximation can lead to very significant performance gains.

Comparison of the presented algorithm with traditional optimisation algorithms such as EM would provide valuable insight into the applicability of the presented algorithm on realistic problems. Unfortunately, there is no exact variational EM algorithm for the nonlinear state-space model used in this work, which makes direct comparison difficult.

In theory, it is possible to use an EM-like algorithm where Kalman filter based updates are used to infer the new states for each iteration. Unfortunately, some initial testing indicated that iterated extended Kalman filter (IEKS) [6] is quite unstable for at least this particular NSSM. In several simulations the NSSM ended up in such a state that IEKS did not converge to any meaningful states. One solution to this problem would be to use a filter that uses more stable methods to approximate the nonlinearity. One example of such advanced filter is backward-smoothing extended Kalman filter [53]. However, because of relatively complex implementation and concerns over increased computational cost, no comparison with this method was made.

As mentioned earlier, the implementation of Riemannian conjugate gradient makes use of several simplifying assumptions. Most importantly, geodesic curves were not used for line search. For manifolds where geodesics are close to linear, the effects of this approximation will likely be limited. It is also worth noting that a large part of the earlier work with natural gradient makes similar assumptions, for example many of the works of Amari [4, 3].

Further experiments are required to determine how much this approximation affected the experimental results in Chapter 6. In the experiments Riemannian conjugate converged very rapidly in the beginning. However, this convergence rate tends to slow down, and it is possible that this was at least partially caused by the approximations used in the implementation of the Riemannian conjugate gradient.

It is also worth noting that superlinear convergence proofs for Riemannian conjugate gradient involve the use of exact line search [65], which is not feasible in practice except for some special cases. Therefore a realistic implementation of the Riemannian conjugate gradient algorithm will already have to make use of at least some approximations. In practice the restarting condition in the conjugate gradient algorithm will ensure that the effects of the inexact line search will not become too significant.

7.1 Other Applications

Even though nonlinear state-space models are used as a case study in this thesis, the presented algorithms can be used for almost any probabistic model where parametric approximations are used and a suitable cost function can be derived.

In practice, there are some limitations of the applicability of the algorithm, however. Most importantly, geometry of certain problem spaces can be so complex that computation of the natural gradient is not feasible. In addition, even if the natural gradient can be computed, the computation of the inverse Fisher information matrix may be too time-consuming to make the implementation useful in practice.

7.2 Future Work

The implementation of the Riemannian conjugate gradient method uses some approximations such as using the flat geometry for line search subroutine, which may slow down the convergence of the algorithm, especially in problems where the geometry of the problem space is far from flat. Comparison of the basic line search and line search along geodesics would provide valuable information how much the geometry of the problem space affects the results. As discussed earlier in this chapter, it is at least possible that this exact implementation would provide further performance gains for the experiments in this work as well.

Variational EM algorithms have been derived for many other parametric models, and one of these could be used to compare the performance of the algorithms. One interesting test case would be mixture-of-Gaussian model, where direct comparisons could be made with the EM-based variational Bayesian mixture-of-Gaussians (VB-MOG) model [8].

In the experiments with the speech data in Section 6.3 the data set is so small, that it is impossible to derive any kind of general model for speech. However, with a much larger data set, it may be possible to find a reasonable state-space representation for speech data. Such a model could then be used as a preprocessing tool by using the state-space representation of speech data in e.g. speech recognition tasks.

This kind of application requires a fast inference algorithm for quickly deriving the state-space for a given data-set. One such algorithm is presented in [55]. Further study is also required to determine how the concept of total derivatives presented in this paper works with natural gradient.

Chapter 8

Conclusions

In this thesis, a Riemannian conjugate gradient method for learning probabilistic models was presented. Traditionally natural gradient based algorithms have used the geometry of the true posterior distribution. In this thesis, however, the geometry of the variational approximations is used instead. This makes the implementation simple as the space of the approximating distributions typically has less complex geometry than the space of the true posterior distributions. It is also possible to apply the method to a wide range of different models which use the same variational approximation.

As a case study, the algorithm was used to learn nonlinear state-space models with multiple different data sets. Riemannian conjugate gradient method performed significantly better than other compared algorithms. Compared to a standard conjugate gradient method, the Riemannian conjugate gradient method was at least ten times faster with all the data sets.

Bibliography

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

[2] S. Amari. Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, 8(9):1379–1408, 1995.

[3] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[4] S. Amari, A. Cichocki, and H. Yang. A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 757–763. MIT Press, Cambridge, MA, USA, 1996.

[5] S. Amari and H. Nagaoka. *Methods of Information Geometry*. American Mathematical Society and Oxford University Press, Providence, Rhode Island, 2000.

[6] B. Anderson and J. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[7] D. K. Arrowsmith and C. M. Place. *An Introduction to Dynamical Systems*. Cambridge University Press, Cambridge, 1990.

[8] H. Attias. A variational Bayesian framework for graphical models. In S. Solla, T. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 209–215. MIT Press, Cambridge, MA, USA, 2000.

[9] 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.

[10] O. E. Barndorff-Nielsen. Likelihood and observed geometries. *Ann. Statist.*, 14:856–873, 1986.

[11] C. Bishop. *Pattern Recognition and Machince Learning.* Springer, Cambridge, 2006.

[12] J. M. Corcuera and F. Giummolè. A characterization of monotone and regular divergences. *Annals of the Institute of Statistical Mathematics*, 50(3):433–450, 1998.

[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* J. Wiley, New York, 1991.

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

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

[16] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[17] A. Edelman and S. T. Smith. On conjugate gradient-like methods for eigen-like problems. *BIT Numerical Mathematics*, 36(3):494–508, 1996.

[18] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7:149–154, 1964.

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

[20] Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 507–513. The MIT Press, Cambridge, MA, USA, 2001.

[21] Z. Ghahramani and S. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 431–437. The MIT Press, Cambridge, MA, USA, 1999.

[22] G. H. Golub and C. F. V. Loan. *Iterative Methods for Linear Systems*, chapter 10. Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[23] L. Grippo and S. Lucidi. A globally convergent version of the Polak-Ribière conjugate gradient method. *Math. Programming*, 78:375–391, 1997.

[24] S. Haykin. *Neural Networks – A Comprehensive Foundation, 2nd ed.* Prentice-Hall, 1999.

[25] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[26] A. Honkela. Approximating nonlinear transformations of probability distributions for nonlinear independent component analysis. In *Proc. 2004 IEEE Int. Joint Conf. on Neural Networks (IJCNN 2004)*, pages 2169–2174, Budapest, Hungary, 2004.

[27] A. Honkela. *Advances in Variational Bayesian Nonlinear Blind Source Separation.* PhD thesis, Helsinki University of Technology, Espoo, Finland, 2005.

[28] A. Honkela and H. Valpola. Unsupervised variational Bayesian learning of nonlinear models. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 593–600. MIT Press, Cambridge, MA, USA, 2005.

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

[30] R. Hooke and T. A. Jeeves. 'Direct search' solution of numerical and statistical problems. *J. of the ACM*, 8(2):212–229, 1961.

[31] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[32] A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. *Neural Processing Letters*, 22(2):183–204, 2005.

[33] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In M. Jordan, editor, *Learning in Graphical Models*, pages 105–161. The MIT Press, Cambridge, MA, USA, 1999.

[34] H. Kimura and S. Kobayashi. Efficient non-linear control by combining Q-learning with local linear controllers. In *Proc. ICML*, pages 210–219, 1999.

[35] H. Lappalainen and J. Miskin. Ensemble learning. In M. Girolami, editor, *Advances in Independent Component Analysis*, pages 75–92. Springer-Verlag, Berlin, 2000.

[36] S. Ma, C. Ji, and J. Farmer. An efficient EM-based training algorithm for feedforward neural networks. *Neural Computation*, 10(2):243–256, 1997.

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

[38] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal of Appl. Math.*, 11:431–441, 1963.

[39] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[40] A. Miele and J. W. Cantrell. Study on memory gradient method for the minimization of function. *Journal of Optimization Theory and Application*, 3:459–470, 1969.

[41] T. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, UAI 2001*, pages 362–369, 2001.

[42] J. Miskin and D. J. C. MacKay. Ensemble learning for blind source separation. In S. Roberts and R. Everson, editors, *Independent Component Analysis: Principles and Practice*, pages 209–233. Cambridge University Press, 2001.

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

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

[45] J. L. Nazareth. A conjugate direction algorithm for unconstrained minimization without line searches. *Journal of Optimization Theory and Application*, 23:373–387, 1977.

[46] R. M. Neal. *Bayesian Learning for Neural Networks, Lecture Notes in Statistics No. 118.* Springer-Verlag, 1996.

[47] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. The MIT Press, Cambridge, MA, USA, 1999.

[48] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1:199–242, 1991.

[49] K. B. Petersen, O. Winther, and L. K. Hansen. On the slow convergence of EM and VBEM in low-noise linear models. *Neural Computation*, 17(9):1921–1926, 2005.

[50] E. Polak and G. Ribière. Note sur la convergence de méthodes de directions conjugées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 16:35–43, 1969.

[51] M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254, 1977.

[52] M. J. D. Powell. *Approximation Theory and Method*, chapter 4. Cambridge University Press, Cambridge, 1981.

[53] M. Psiaki. Backward-smoothing extended Kalman filter. *Journal of Guidance, Control, and Dynamics*, 28(5), Sep–Oct 2005.

[54] T. Raiko and M. Tornio. Learning nonlinear state-space models for control. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'05)*, pages 815–820, Montreal, Canada, 2005.

[55] T. Raiko, M. Tornio, A. Honkela, and J. Karhunen. State inference in variational Bayesian nonlinear state-space models. In *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Source Separation (ICA 2006)*, pages 222–229, Charleston, South Carolina, USA, March 2006.

[56] C. R. Rao. Information and accuracy attainable in the estimation of statistical paramaters. *Bulletin of Calcutta Mathematical Society*, 37:81–91, 1945.

[57] M. Rattray and D. Saad. Transients and asymptotics of natural gradient learning. In *Proc. of the 8th International Conference on Artificial Neural Networks (ICANN 98)*, pages 183–188, 1998.

[58] R. Salakhutdinov and S. T. Roweis. Adaptive overrelaxed bound optimization methods. In *Proc. 20th International Conference on Machine Learning (ICML 2003)*, pages 664–671, 2003.

[59] R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani. Optimization with EM and expectation-conjugate-gradient. In *Proc. 20th International Conference on Machine Learning (ICML 2003)*, pages 672–679, 2003.

[60] M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[61] L. E. Scales. *Introduction to Non-Linear Optimization.* Springer-Verlag, New York, 1985.

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

[63] 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.

[64] L. T. Skovgaard. A Riemannian geometry of the multivariate Gaussian model. *Scandinavian Journal of Statistics*, 11(4):211–223, 1984.

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

[66] S. T. Smith. Optimization techniques on Riemannian manifolds. *Fields Institute Communications*, pages 113–146, 1994.

[67] T. Tanaka. Information geometry of mean-field approximation. In M. Opper and D. Saad, editors, *Advanced Mean Field Methods: Theory and Practice*, pages 259–273. The MIT Press, Cambridge, MA, USA, 2001.

[68] M. Tornio and T. Raiko. Variational Bayesian approach for nonlinear identification and control. In *Proceedings of the IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, NMPC FS06*, pages 41–46, Grenoble, France, 2006.

[69] H. Valpola. *Bayesian Ensemble Learning for Nonlinear Factor Analysis.* PhD thesis, Helsinki University of Technology, Espoo, Finland, 2000. Published in Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 108.

[70] H. Valpola, A. Honkela, and X. Giannakopoulos. Matlab codes for the NFA and NDFA algorithms. *http://www.cis.hut.fi/projects/bayes/software/*, 2002.

[71] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

[72] J. Winn and C. M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, April 2005.

[73] Y. Yang. Optimization on Riemannian manifold. In *Proc. of the 38th Conference on Decision and Control*, pages 888–893, 1999.