EFFICIENT OPTIMIZATION ALGORITHMS FOR LEARNING

by

Ruslan Salakhutdinov

A thesis submitted in conformity with the requirements
for the degree of Master's Degree
Graduate Department of Computer Science
University of Toronto

Abstract

Efficient Optimization Algorithms for Learning

Ruslan Salakhutdinov

Master's Degree

Graduate Department of Computer Science

University of Toronto

2003

Many problems in machine learning and pattern recognition ultimately reduce to the optimization of a scalar valued function. A variety of general techniques exist for optimizing such objective functions. We study the general class of *bound optimization* algorithms – including Expectation-Maximization, Iterative Scaling, Non-negative Matrix Factorization, Concave-Convex Procedure – and their relationship to direct optimization algorithms such as gradient-based methods for parameter learning. We also provide a theoretical analysis of the convergence properties of bound optimization algorithms and identify analytic conditions under which these optimizers exhibit quasi-Newton behavior, and conditions under which they possess poor, first-order convergence. Motivated by these analyses, we interpret and analyze their convergence properties and provide some recipes for preprocessing input to these algorithms to yield faster convergence behavior. Our presented analysis also allows us to design several algorithms for practical optimization, that possess superior convergence over standard existing methods.

Dedication

To my loving parents Nailya and Ravil

for inspiring, helping and letting me pursue my dreams

so far away from home

&

To my dear sister Olya and brother-in-law Steve

for their invaluable advice and support

&

To my adoring nephew Nathan.

Acknowledgements

I would like to thank my supervisor Sam Roweis for his guidance, support and for providing me with a warm and outstanding intellectual environment over the last two years at the University of Toronto.

I would also like to thank Zoubin Ghahramani for helping me in the accomplishment of this research and for being my second reader. Many members of the Toronto Machine Learning group have also been an inspiration to me: Geoff Hinton, Rich Zemel, Radford Neal, Max Welling, and Yee Whye Teh. Their discussions and presentations during weekly group meetings have been one of my best learning experiences. Very special thanks to Yoshua Bengio for nominating and helping me in getting Precarn scholarship.

My special gratitude goes to Morris Wray, the person who provided me with the great opportunity to pursue my educational goals in one of the best and most exciting schools in the United States – High Point University. I would also like to thank several professors: Roger Shore, Rob Harger, and Jeff Butera, for introducing me into the area of computer science and providing me with a great academic environment during my undergraduate years.

Contents

Chapter 1

Introduction

1.1 Optimization Algorithms

Many problems in machine learning and pattern recognition ultimately reduce to the optimization of a scalar valued function $L(\Theta)$ of a free parameter vector Θ. For example, in supervised and unsupervised probabilistic modeling the objective function may be the (conditional) data likelihood or the posterior over parameters. In discriminative learning we may use a classification or regression score; in reinforcement learning we may use average discounted reward. Optimization may also arise during inference; for example we may want to reduce the cross entropy between two distributions or minimize a function such as the Bethe and Kikuchi free energy.

A variety of general techniques exist for optimizing such objective functions. Broadly, they can be placed into one of two categories: direct optimization (DO) algorithms and what we will refer to as bound optimization (BO) algorithms.

Direct optimization methods for the parameter learning work directly with the objective function and its derivatives (or estimates thereof), trying to maximize or minimize it by adjusting the free parameters in a local search. This category of algorithms includes standard gradient-based algorithms, line search methods such as popular conjugate gradient, and more computationally intensive second-order methods, such as Newton-Raphson. An example of a particular direct optimization method; the Fletcher-Reeves-Polak-Ribiere flavor of conjugate gradient algorithm [21]; is given below:

Fletcher-Reeves-Polak-Ribiere flavor of Conjugate Gradient Algorithm for optimizing $L(\Theta)$:

- **Initialize:** $g_0 = \nabla L(\Theta_0), \quad d_0 = g_0, \quad k = 0$
- **Compute sequentially:**

  1. α_k, the value of α that maximizes $L(\Theta_k + \alpha d_k)$ line search
  2. $\Theta_{k+1} = \Theta_k + \alpha_k d_k$ new parameter
  3. $g_{k+1} = \nabla L(\Theta_{k+1})$ new gradient
  4. $\beta_k = \frac{g_{k+1}^T \cdot (g_{k+1} - g_k)}{g_k^T \cdot g_k}$ blending factor
  5. $d_{k+1} = g_{k+1} - \beta_k d_k, \quad k = k + 1$ new search direction

Direct optimization methods can be applied, in principle, to any deterministic function of the parameters. These algorithms have been extensively studied and are rather well-understood.

Bound optimization (BO) algorithms, on the other hand, take advantage of the fact that many objective functions arising in practice have a special structure. We can often exploit this structure to obtain a bound on the objective function and proceed by optimizing this bound. Ideally, we seek a bound that is valid everywhere in parameter space, easily optimized, and equal to the true objective function at one (or more) point(s). A general form of a bound *maximizer* which iteratively *lower bounds* the objective function is given below:

General Bound Optimizer for maximizing $L(\Theta)$:

- **Assume**: $\exists\ G(\Theta, \Psi)$ such that for any Θ' and Ψ':

  1. $G(\Theta', \Theta') = L(\Theta')$ & $L(\Theta) \geq G(\Theta, \Psi')\ \forall\ \Psi' \neq \Theta$
  2. $\arg\max_{\Theta} G(\Theta, \Psi')$ can be found easily for any Ψ'.

- **Iterate**: $\Theta^{t+1} = \arg\max_{\Theta} G(\Theta, \Theta^t)$
- **Guarantee**: $L(\Theta^{t+1}) = G(\Theta^{t+1}, \Theta^{t+1}) \geq G(\Theta^{t+1}, \Theta^t) \geq G(\Theta^t, \Theta^t) = L(\Theta^t)$

A bound optimizer does nothing more than coordinate ascent in the functional $G(\Theta, \Psi)$, alternating between maximizing G with respect to Ψ for fixed Θ and with respect to Θ for fixed Ψ. These algorithms enjoy a strong guarantee; they never worsen the objective function. This concept is graphically illustrated in figure 1.1.

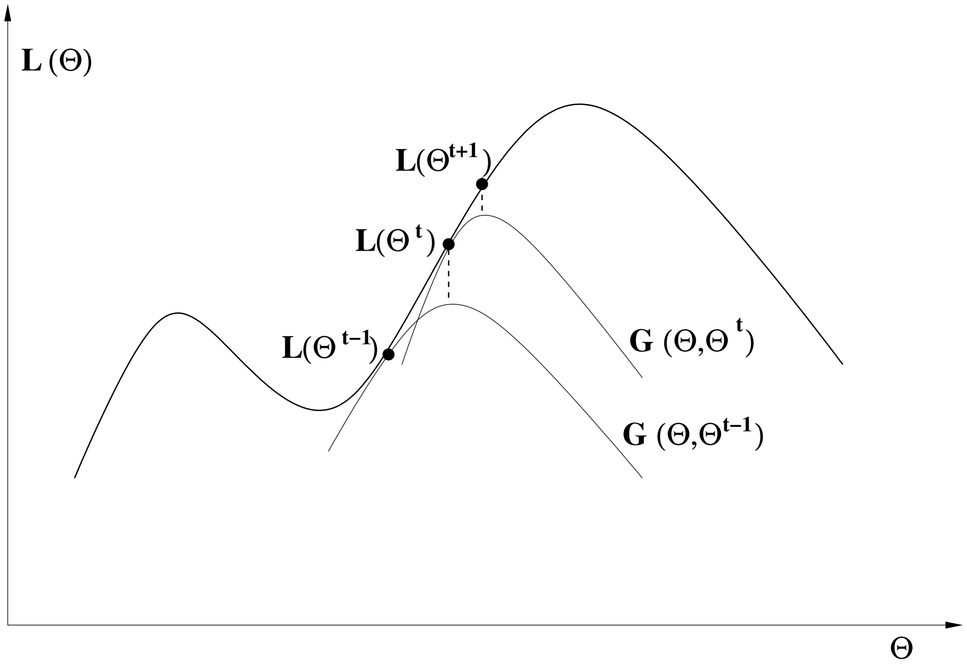

Figure 1.1: Consider iteration of the bound optimizer with $\Psi = \Theta^t$ as the currect fit for the parameter vector Θ. Maximizing the bound function $G(\Theta, \Theta^t)$ with respect to Θ yields $\Theta^{t+1} = \arg\max_{\Theta} G(\Theta, \Theta^t)$, which guarantees $G(\Theta^{t+1}, \Theta^t) \geq G(\Theta^t, \Theta^t) = L(\Theta^t)$. On the other side, since $G(\Theta, \Theta^t)$ is a lower bound on the objective function, we have $L(\Theta^{t+1}) = G(\Theta^{t+1}, \Theta^{t+1}) \geq G(\Theta^{t+1}, \Theta^t)$. Both inequalities immediately imply $L(\Theta^{t+1}) \geq L(\Theta^t)$.

Many popular iterative algorithms are bound optimizers, including the EM algorithm for maximum likelihood learning in latent variable models[4], iterative scaling (IS) algorithms for parameter estimation in maximum entropy models[3], non-negative matrix factorization (NMF)[14] and the recent Concave-Convex Procedure (CCCP) algorithm for minimizing the Bethe free energy in approximate inference problems[32].

We will study the general class of bound optimization algorithms and their relationship to direct optimization algorithms to determine conditions under which one technique can be expected to outperform another. Our general results apply to any model for which a bound optimizer can be constructed.

1.2 Thesis Overview

We will begin by introducing and analyzing a class of bound optimization algorithms such as EM, Iterative Scaling, Non-negative Matrix Factorization, CCCP and their relationship to the gradient and second-order methods. In chapter 2 we will describe analytic conditions under which bound optimization algorithms exhibit quasi-Newton behavior, and conditions under which they possess poor, first-order convergence.

Based on this analysis, in chapter 3 we will consider several specific algorithms, interpret and analyze their convergence properties and provide some recipes for preprocessing input to these algorithms to yield faster convergence behavior.

In chapter 4 we will study a popular Expectation - Maximization (EM) algorithm for learning in latent variable models. We will also show a close relationship between the Expectation - Maximization (EM) algorithm and direct optimization algorithms such as gradient-based methods for parameter learning. Based on our analysis, we will propose two novel algorithms for maximum likelihood estimation of latent variable models. To support the robustness of our conclusions, we will report empirical results showing that the proposed new algorithms can substantially outperform standard EM in terms of speed of convergence.

In chapter 5 we will describe a class of *overrelaxed* bound optimization algorithms, that we call BO(η) algorithms with η being the learning rate, and their relationship to standard bound optimizers. We will provide a theoretical analysis of the convergence properties of these optimizers and identify analytic conditions under which they are expected to outperform the standard versions. Based on this analysis, we will propose a novel, simple adaptive overrelaxed scheme for practical optimization, and report empirical results on several synthetic and real-world data sets showing that these new adaptive methods exhibit much superior performance.

We will finish with general discussion section summarizing our main contributions and describing future research directions.

Chapter 2

Convergence Analysis of Bound Optimization Methods

In this chapter we study the general class of *bound optimization* algorithms – including EM, Iterative Scaling, Non-negative Matrix Factorization, CCCP – and their relationship to direct optimization algorithms such as gradient-based methods for parameter learning. We derive a general relationship between the updates performed by bound optimization methods and those of gradient and second-order methods and identify analytic conditions under which bound optimization algorithms exhibit quasi-Newton behavior, and conditions under which they possess poor, first-order convergence.

2.1 Introduction

In chapter 1 we defined a general form of bound optimization algorithms. In this chapter we will explore two questions of theoretical and practical interest: when will bound optimization be fast or slow relative to other standard approaches, and what can be done to improve convergence rates of these algorithms when they are slow.

2.2 Linear Convergence of Bound Optimizers

Any bound optimizer implicitly defines a mapping: $M : \Theta \to \Theta'$ from parameter space to itself, so that $\Theta^{t+1} = M(\Theta^t)$. If iterates Θ^t converge to a fixed point Θ^* then $\Theta^* = M(\Theta^*)$. If $M(\Theta)$ is continuous and differentiable, we can Taylor expand it in the neighborhood of the fixed point Θ^*:

$$\Theta^{t+1} - \Theta^* \approx M'(\Theta^*)(\Theta^t - \Theta^*) \tag{2.1}$$

where $M'(\Theta^*) = \frac{\partial M}{\partial \Theta}|_{\Theta=\Theta^*}$. Since $M'(\Theta^*)$ is typically nonzero, a bound optimizer can essentially be seen as a linear iteration algorithm with a "convergence rate matrix" $M'(\Theta^*)$. Intuitively, $M'(\Theta^*)$ can be viewed as an operator that forms a contraction mapping around Θ^*. In general, we would expect $\frac{\partial^2 L(\Theta)}{\partial \Theta^2}|_{\Theta=\Theta^*}$ to be negative semidefinite,[1] or negative definite, and thus the eigenvalues of $M'(\Theta^*)$ all lie in $[0,1]$ or $[0,1)$ respectively [18]. Exceptions to the convergence of the bound optimizer to a local optimum of $L(\Theta)$ occur if $M'(\Theta^*)$ has eigenvalues that exceed unity.

Near a local optimum, the convergence rate matrix $M'(\Theta^*)$ is related to the curvature of the functional $G(\Theta, \Psi)$ as follows:

Lemma 2.1: *If bound optmizer's iterates Θ^t converge to Θ^* and $M(\Theta)$ is differentiable in the parameter space Θ, then:*

$$\lim_{\Theta^t \to \Theta^*} M'(\Theta^t) = -\left[\nabla^2_G(\Theta^*, \Psi^*)\right]\left[\nabla^2_G(\Theta^*)\right]^{-1} \tag{2.2}$$

where we define $\nabla^2_G(\Theta^*, \Psi^*) \equiv \left[\frac{\partial^2 G(\Theta,\Psi)}{\partial\Theta\partial\Psi^T}\Big|_{\substack{\Theta\,=\,\Theta^* \\ \Psi\,=\,\Theta^*}} \right]$; $\quad \nabla^2_G(\Theta^*) \equiv \left[\frac{\partial^2 G(\Theta,\Psi)}{\partial\Theta\partial\Theta^T}\Big|_{\substack{\Theta\,=\,\Theta^* \\ \Psi\,=\,\Theta^*}} \right]$,

Proof sketch: By using Taylor series expansion of $\nabla_G(\Theta^2, \Theta^1) = \frac{\partial G(\Theta,\Theta^1)}{\partial \Theta}|_{\Theta=\Theta^2}$ around (Θ^*, Θ^*), we have:

$$\nabla G(\Theta^2, \Theta^1) \;=\; \nabla_G(\Theta^*, \Theta^*) + (\Theta^2 - \Theta^*)^T \nabla^2_G(\Theta^*) + (\Theta^1 - \Theta^*)^T \nabla^2_G(\Theta^*, \Psi^*) + \dots$$

By noting that $\nabla_G(\Theta^*, \Theta^*) = \frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^*} = 0$, and substituting Θ^1 with Θ^t, and Θ^2 with

[1] $L(\Theta)$ is the objective function

$M(\Theta^t)$ results:

$$0 \quad = \quad (M(\Theta^t) - \Theta^*)^T \nabla^2_G(\Theta^*) + (\Theta^t - \Theta^*)^T \nabla^2_G(\Theta^*, \Psi^*) + \ldots$$

Assuming that higher order terms are negligible, in the limit, using $\Theta^* = M(\Theta^*)$, we have:

$$0 = \left[\lim_{\Theta^t \to \Theta^*} M'(\Theta^t) \right] \nabla^2_G(\Theta^*) + \nabla^2_G(\Theta^*, \Psi^*) \tag{2.3}$$

We therefore get:

$$\lim_{\Theta^t \to \Theta^*} M'(\Theta^t) = -\left[\nabla^2_G(\Theta^*, \Psi^*)\right] \left[\nabla^2_G(\Theta^*)\right]^{-1} \tag{2.4}$$

In general, the bound function $\nabla^2_G(\Theta^*)$ is negative definite, or invertible. Indeed, let Θ^* be the local maximum of $\nabla^2_G(\Theta^*)$, and consider $\Theta^{'}$ being some point that lies within some small neighborhood of Θ^*. We can then use the following quadratic approximation:

$$G(\Theta^{'}) \quad = \quad G(\Theta^*) + (\Theta^* - \Theta^{'})^T \nabla_G(\Theta^*) + (\Theta^* - \Theta^{'})^T \nabla^2_G(\Theta^*)(\Theta^* - \Theta^{'}) \tag{2.5}$$

Since Θ^* is the local maximum, we have:

$$(\Theta^* - \Theta^{'})^T \nabla^2_G(\Theta^*)(\Theta^* - \Theta^{'}) = G(\Theta^{'}) - G(\Theta^*) \leq 0 \tag{2.6}$$

for any $\Theta^{'}$, which implies that $\nabla^2_G(\Theta^*)$ is negative definite. For many models, however, the bound function $\nabla^2_G(\Theta^*)$ is concave, which immediately suggests that $\nabla^2_G(\Theta^*)$ is negative definite, or invertible.

2.3 Gradient and Newton behaviors of bound optimization

What directions do bound optimizers move in parameter space? For most objective functions, the BO step $\Theta^{(t+1)} - \Theta^{(t)}$ in parameter space and true gradient can be trivially related by *transformation matrix* $P(\Theta^t)$,[2] that changes at each iteration:

$$\Theta^{(t+1)} - \Theta^{(t)} = P(\Theta^t) \nabla_L(\Theta^t) \tag{2.7}$$

[2] The transformation matrix $P(\Theta^t)$ is non-unique, i.e. many $P(\Theta^t)$ matrices exist for which 2.7 holds (see appendix A).

where we define $\nabla_L(\Theta^t) = \frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^t}$.

Proposition 2.1: *Under certain conditions, the transformation matrix $P(\Theta^t)$ is guaranteed to be positive definite with respect to the gradient. In particular,* **if**

C1: *$G(\Theta, \Theta^t)$ is well-defined, and differentiable everywhere in Θ.* **and**

C2*: For any fixed $\Theta^t \neq \Theta^{(t+1)}$, along any direction that passes through Θ^{t+1}, $G(\Theta, \Theta^t)$ has only a single critical point, located at the maximum Θ^{t+1};* **then**

$$\nabla_L^\top(\Theta^t) P(\Theta^t) \nabla_L(\Theta^t) > 0 \quad \forall \Theta^t \tag{2.8}$$

Proof sketch: For $\nabla_G^\top(\Theta^t)(\Theta^{(t+1)} - \Theta^t)$, we note that $\nabla_G^\top(\Theta^t) = \frac{\partial G(\Theta, \Theta^t)}{\partial \Theta}|_{\Theta=\Theta^t}$ is the directional derivative of function $G(\Theta, \Theta^t)$ in the direction of $\Theta^{(t+1)} - \Theta^t$. C1 and C2 together imply that this quantity is positive, otherwise by the Mean Value Theorem (C1) $G(\Theta, \Theta^t)$ would have a critical point along some direction, located at a point other than Θ^{t+1} (C2). By using the the identity $\nabla_L(\Theta^t) = \frac{\partial G(\Theta, \Theta^t)}{\partial \Theta}|_{\Theta=\Theta^t}$, we have:

$$\nabla_L^\top(\Theta^t) P(\Theta^t) \nabla_L(\Theta^t) = \nabla_G^\top(\Theta^t)(\Theta^{(t+1)} - \Theta^t) > 0. \tag{2.9}$$

The second condition may seem very strong, however, it is satisfied in many practical cases. For example, for the EM algorithm, it is satisfied whenever the M-step has a single unique solution (in particular, it holds for exponential family models due to concavity of $G(\Theta, \Theta^t)$); for GIS, NMF, CCCP, and many others, it is satisfied due to concavity of $G(\Theta, \Theta^t)$ (although C2 does not imply concavity).

The important consequence of the above analysis is that when the bound function has a unique optimum, BO has the appealing quality of always taking a step $\Theta^{(t+1)} - \Theta^t$ having positive projection onto the true gradient of the objective function $L(\Theta^t)$. This makes BO similar to first order methods operating on the gradient of a locally reshaped cost function.

For maximum likelihood learning of a mixture of Gaussians model using the EM-algorithm, this positive definite transformation matrix $P(\Theta^t)$ was first described by Xu and Jordan[31]. In the appendix A we extend their results by deriving the explicit form of the transformation matrix for several other latent variables models such as Factor Analysis (FA), Probabilistic Principal Component Analysis (PPCA), mixture of PPCAs, mixture of FAs, and Hidden Markov Models; we also derive the general form of $P(\Theta^t)$ matrix for exponential family models in terms of natural parameters.

One can further study the structure of the transformation matrix $P(\Theta^t)$ and relate it to the convergence rate matrix M'.

Proposition 2.2: *If BO algorithm converges to Θ^* and $P(\Theta)$ and $M(\Theta)$ are well-defined, differentiable functions in the neighborhoods of the iterates Θ^t in the parameter space Θ, then under a weak assumption that the inverse of the Hessian of the objective function $\big[-S(\Theta^*)\big]^{-1}$ exists ($S(\Theta^t) = \frac{\partial^2 L(\Theta)}{\partial \Theta^2}|_{\Theta=\Theta^t}$), we have:*

$$\lim_{\Theta^t \to \Theta^*} P(\Theta^t) = \Bigg[I - M'(\Theta^*)\Bigg]\Bigg[-S(\Theta^*)\Bigg]^{-1} \tag{2.10}$$

Proof sketch: Taking negative derivatives of (2.7) with respect to Θ^t yields

$$I - M'(\Theta^t) = -P'(\Theta^t)\nabla_L(\Theta^t) - P(\Theta^t)S(\Theta^t) \tag{2.11}$$

where $M'_{ij}(\Theta^t) = \partial\Theta_i^{t+1}/\partial\Theta_j^t$ is the input-output derivative matrix for the BO mapping and $P'(\Theta^t) = \frac{\partial P(\Theta^t)}{\partial \Theta}|_{\Theta=\Theta^t}$ is the tensor derivative of $P(\Theta^t)$ with respect to Θ^t. In the limit, near a fixed point, the first term will vanish since the gradient is going to zero (assuming $P'(\Theta^t)$ does not become infinite). Therefore the equality (2.10) readily follows.

We conclude that if bound optimization algorithm iterates converge to a local optima at Θ^*, then near this point (i.e. for sufficiently large t) BO may exhibit Quasi-Newton convergence behavior. This is also true in "plateau" regions where the gradient is very small even if they are not near a local optimum.

The nature of the Quasi-Newton behavior is controlled by the convergence matrix $M'(\Theta^*)$. We can now study the form or the properties of this matrix, for example by examining its individual entries, its eigenvalues, or the ratio of its two top eigenvalues. In particular, if the top eigenvalue of $M'(\Theta^*)$ tends to zero, then BO becomes a true Newton method, rescaling the gradient by exactly the negative inverse Hessian:

$$\Theta^{t+1} = \Theta^t - S(\Theta^t)^{-1}\nabla_L(\Theta^t) \tag{2.12}$$

As the eigenvalues increase and tend to unity, BO takes smaller and smaller stepsizes, giving poor, first-order, convergence.

In the next section we will discuss some of the widely used bound optimization methods.[3] We will show how one can obtain the convergence rate matrix of a particular BO algorithm and thus identify conditions under which it is expected to have fast or slow convergence behavior.

[3] One of the popular bound optimization algorithms for maximum likelihood learning of parameters in the presence of latent variables is the Expectation-Maximization. We will discuss in greater details the nature of this algorithm in chapter 4 .

Chapter 3

Bound Optimization Methods

In this chapter, we consider several specific algorithms, including Iterative Scaling, Nonnegative Matrix Factorization, Concave-Convex Procedure; interpret and analyze their convergence properties and provide some recipes for preprocessing input to these algorithms to yield faster convergence behavior. We report empirical results supporting our analysis and showing that simple data preprocessing can result in dramatically improved performance of bound optimizers in practice.

3.1 Generalized Iterative Scaling (GIS) Algorithm

In this section we consider the Generalized Iterative Scaling algorithm [3], widely used for the parameter estimation in maximum entropy models. Its goal is to determine the parameters Θ^* of an exponential family distribution:

$$p(x|\Theta) = \frac{1}{Z(\Theta)} \exp\left(\Theta^T F(x)\right) \tag{3.1}$$

such that certain generalized marginal constraints are preserved: $\sum_x p(x|\Theta^*)F(x) = \sum_x \bar{p}(x)F(x)$, where $Z(\Theta)$ is the normalizing factor, $\bar{p}(x)$ is a given empirical distribution and $F(x) = [f_1(x), ..., f_d(x)]^T$ is a given feature vector on the input.

The GIS algorithm requires that $f_i(x) > 0 \ \forall i$, but we will not require $\sum_i f_i(x) = 1$[20].

The log-likelihood is:

$$L(\Theta) = \sum_x \bar{p}(x) \ln p(x|\Theta) = \sum_x \bar{p}(x)\Theta^T F(x) - \ln Z(\Theta) \tag{3.2}$$

We note that $\ln Z(\Theta) \leq Z(\Theta)/Z(\Psi) + \ln Z(\Psi) - 1$ for any Ψ, and $\exp \sum_i \Theta_i f_i(x) \leq \sum_i f_i(x) \exp \Theta_i + [1 - \sum_i f_i(x)]$, with $\sum_i f_i(x) \leq 1$. Defining $s = \max_x \sum_i f_i(x)$, we can construct a lower bound on $L(\Theta)$:

$$\begin{aligned} L(\Theta) \;\; \geq & \;\; \sum_x \bar{p}(x) \sum_i \Theta_i f_i(x) - \ln Z(\Psi) + \sum_i \frac{f_i(x)}{s} - \\ & \sum_x p(x|\Psi) \sum_i \frac{f_i(x)}{s} \exp\big[s(\Theta_i - \Psi_i)\big] = G(\Theta, \Psi) \end{aligned} \tag{3.3}$$

This lower bound has the useful property that its maximization is decoupled across the parameters Θ_i. The GIS algorithm is then given by:

$$\Theta_i^{t+1} = \Theta_i^t + \frac{1}{s} \ln \frac{\sum_x \bar{p}(x) f_i(x)}{\sum_x p(x|\Theta^t) f_i(x)} \tag{3.4}$$

Define $\bar{F}(\Theta^*) \equiv \sum_x p(x|\Theta^*)F(x)$ to be the mean of the feature vectors, $\mathrm{D}(\Theta^*) \equiv \ \mathrm{diag}\big[\bar{F}(\Theta^*)\big]$ to be the corresponding diagonal matrix, and $\mathrm{Cov}(\Theta^*)$ to be covariance of the feature vectors under model distribution $p(x|\Theta^*)$. We can compute second order statistics using (2.2):

$$\nabla_G^2(\Theta^*) \;\; = \;\; -s\,\mathrm{diag}\big[\bar{F}(\Theta^*)\big] = -s\mathrm{D}(\Theta^*) \tag{3.5}$$

$$\begin{aligned} \nabla_G^2(\Theta^*, \Psi^*) \;\; &= \;\; s\,\mathrm{diag}\big[\bar{F}(\Theta^*)\big] - \left[\sum_x p(x|\Theta^*)F(x)F(x)^T - \big[\bar{F}(\Theta^*)\big]\big[\bar{F}(\Theta^*)\big]^T\right] \\ &= \;\; s\mathrm{D}(\Theta^*) - \mathrm{Cov}(\Theta^*) \end{aligned} \tag{3.6}$$

Due to the concavity of $G(\Theta, \Psi')$ for any fixed Ψ', the step a GIS algorithm takes in parameter space always has positive projection onto the true gradient of the objective function. From equations 3.4-3.6, the convergence rate matrix $M'(\Theta^*)$ can also be calculated:

$$\frac{\partial M(\Theta)}{\partial \Theta}|_{\Theta=\Theta^*} = I - \frac{1}{s}\mathrm{Cov}(\Theta^*)\mathrm{D}(\Theta^*)^{-1} \tag{3.7}$$

and depends on the covariance and the mean of the feature vectors.

According to (2.10), in the neighborhood of a solution (for sufficiently large t), the step GIS takes in parameter space and true gradient are related by the matrix:

$$P(\Theta^t) \approx \left[\frac{1}{s}\mathrm{Cov}(\Theta^t)\mathrm{D}(\Theta^t)^{-1}\right]\left[-S(\Theta^t)\right]^{-1} \tag{3.8}$$

We can interpret this result as follows: *when feature vectors become less correlated and closer to the origin, GIS exhibits faster convergence in the neighborhood of* Θ^*. If features are highly dependent, then GIS will exhibit extremely slow convergence.

3.2 Non-Negative Matrix Factorization (NMF)

Given a non-negative matrix V, the NMF algorithm[14] tries to find matrices W and H, such that $V \approx WH$. Posed as an optimization problem, we are interested in minimizing a divergence $L(W,H) = D(V||WH)$, subject to $(W,H) \geq 0$ elementwise:

$$L(W,H) = \sum_{ij}\left(V_{ij}\ln\frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij}\right) \tag{3.9}$$

We use the convexity of the log function:

$$-\ln\sum_{c} W_{ic}H_{cj} \leq -\sum_{c}\alpha_{ij}(c,c)\ln\frac{W_{ic}H_{cj}}{\alpha_{ij}(c,c)} \tag{3.10}$$

where $\alpha_{ij}(a,b) = W^t_{ia}H^t_{bj}/\sum_r W^t_{ir}H^t_{rj}$, so that $\alpha_{ij}(c,c)$ sum to one. Defining $\Theta = (W,H)$ and $\Psi = (W^t,H^t)$, we can construct the upper bound on the cost function:

$$\begin{aligned} L(\Theta) \;\leq\; & \sum_{ij} V_{ij}\ln V_{ij} - V_{ij} + \sum_{ijc} W_{ic}H_{cj} - \\ & \sum_{ijc} V_{ij}\alpha_{ij}(c,c)\left[\ln\frac{W_{ic}H_{cj}}{\alpha_{ij}(c,c)}\right] = G(\Theta,\Psi) \end{aligned} \tag{3.11}$$

Maximizing this bound with respect to elements of W and H, while holding $\Psi = (W^t, H^t)$ fixed (i.e.elements of α), we obtain the update equations:

$$W^{t+1}_{ic} = W^t_{ic}\frac{\sum_j H^t_{cj}V_{ic}/(WH)^t_{ic}}{\sum_v H^t_{cv}} \tag{3.12}$$

$$H^{t+1}_{cj} = H^t_{cj}\frac{\sum_i W^t_{ic}V_{ic}/(WH)^t_{ic}}{\sum_w W^t_{wc}} \tag{3.13}$$

Employing (2.2), we can compute the second order statistics to derive an explicit form of the convergence rate M' matrix:

$$\frac{\nabla^2_G(\Theta^*)}{\partial W^*_{ic}\partial W^*_{kp}} = \delta_{ik}\delta_{cp}\sum_j \frac{V_{ij}}{\bar{V}^*_{ij}}\frac{H^*_{cj}}{W^*_{ic}} \quad \frac{\nabla^2_G(\Theta^*)}{\partial W^*_{ic}\partial H^*_{pl}} = \delta_{cp}$$
$$\frac{\nabla^2_G(\Theta^*)}{\partial H^*_{cj}\partial H^*_{pl}} = \delta_{cp}\delta_{jl}\sum_i \frac{V_{ij}}{\bar{V}^*_{ij}}\frac{W^*_{ic}}{H^*_{cj}} \quad \frac{\nabla^2_G(\Theta^*)}{\partial H^*_{cj}\partial W^*_{kp}} = \delta_{cp}$$

$$\begin{aligned}
\frac{\nabla^2_G(\Theta^*,\Psi^*)}{\partial W^*_{ic}\partial W^*_{kp}} &= -\left[\delta_{ik}\delta_{cp}\sum_j \frac{V_{ij}}{\bar{V}^*_{ij}}\frac{H^*_{cj}}{W^*_{ic}} - \delta_{ik}\sum_j \frac{V_{ij}}{\bar{V}^*_{ij}}\frac{H^*_{cj}}{W^*_{ic}}\alpha_{ij}(c,p)\right] \\
\frac{\nabla^2_G(\Theta^*,\Psi^*)}{\partial H^*_{cj}\partial H^*_{pl}} &= -\left[\delta_{jl}\delta_{cp}\sum_i \frac{V_{ij}}{\bar{V}^*_{ij}}\frac{W^*_{ic}}{H^*_{cj}} - \delta_{jl}\sum_i \frac{V_{ij}}{\bar{V}^*_{ij}}\frac{W^*_{ic}}{H^*_{cj}}\alpha_{ij}(c,p))\right] \\
\frac{\nabla^2_G(\Theta^*,\Psi^*)}{\partial W^*_{ic}\partial H^*_{pl}} &= -\frac{V_{ij}}{\bar{V}^*_{ij}}(\delta_{cp} - \alpha_{il}(c,p)) \\
\frac{\nabla^2_G(\Theta^*,\Psi^*)}{\partial H^*_{cj}\partial W^*_{kp}} &= -\frac{V_{kj}}{\bar{V}^*_{kj}}(\delta_{cp} - \alpha_{kj}(c,p))
\end{aligned}$$

where we define $\bar{V}^*_{ij} = \sum_c W^*_{ic}H^*_{cj}$, and $\delta_{ij} = 1$ if $i = j$; 0 – otherwise. The convergence rate matrix M' is therefore of the form:

$$\frac{\partial M(\Theta)}{\partial\Theta}|_{\Theta=\Theta^*} = -\left[\nabla^2_G(\Theta^*,\Psi^*)\right]\left[\nabla^2_G(\Theta^*)\right]^{-1} \tag{3.14}$$

We note that the convergence matrix of NMF much resembles the convergence matrix of GIS, since both algorithms make use of the bound that comes from Jensen's inequality.

3.3 Concave-Convex Procedure

A CCCP [32] optimizer seeks to minimize an energy function $E(\Theta)$, which can be decomposed into a convex $E_{vex}(\Theta)$ and a concave $E_{cave}(\Theta)$ function:

$$E(\Theta) = E_{vex}(\Theta) + E_{cave}(\Theta) \tag{3.15}$$

CCCP algorithm is given by:

$$\nabla E_{vex}(\Theta^{t+1}) = -\nabla E_{cave}(\Theta^t) \tag{3.16}$$

It is easy to see that CCCP belongs to the class of bound optimization algorithms, and therefore can be analyzed as a first order iterative algorithm. Its bound function is of the form:

$$E(\Theta) \quad \leq \quad E_{vex}(\Theta) + E_{cave}(\Psi) + (\Theta - \Psi)^T \nabla E_{cave}(\Psi) = G(\Theta, \Psi) \tag{3.17}$$

Employing (2.2), we have:

$$\nabla^2_G(\Theta^*) \quad = \quad \frac{\partial^2 E_{vex}(\Theta)}{\partial \Theta \partial \Theta^T}|_{\Theta=\Theta^*} \tag{3.18}$$

$$\nabla^2_G(\Theta^*, \Psi^*) \quad = \quad \frac{\partial^2 E_{cave}(\Psi)}{\partial \Psi \partial \Psi^T}|_{\Psi=\Theta^*} \tag{3.19}$$

The convergence rate matrix is therefore given by:

$$M'(\Theta^*) = -\left[\frac{\partial^2 E_{cave}(\Psi)}{\partial \Psi \partial \Psi^T}|_{\Psi=\Theta^*}\right]\left[\frac{\partial^2 E_{vex}(\Theta)}{\partial \Theta \partial \Theta^T}|_{\Theta=\Theta^*}\right]^{-1} \tag{3.20}$$

which can be interpreted as a ratio of concave curvature to convex curvature. According to (2.10) in the neighborhood of a solution (for sufficiently large t) the gradient and step are related:

$$P(\Theta^t) \approx \left[I - \left(\frac{\partial^2 E_{cave}(\Psi)}{\partial \Psi \partial \Psi^T}|_{\Psi=\Theta^t}\right)\left(\frac{\partial^2 E_{vex}(\Theta)}{\partial \Theta \partial \Theta^T}|_{\Theta=\Theta^t}\right)^{-1}\right]\left[-S(\Theta^t)\right]^{-1} \tag{3.21}$$

Of course, the step CCCP takes in parameter space has positive projection onto the true gradient of the original energy function $E(\Theta)$. The above view of CCCP has an interesting interpretation: *If the concave energy function has small curvature compared to the convex energy term in the neighborhood of* Θ^**, CCCP will exhibit a quasi-Newton behavior and will possess fast, typically superlinear convergence.* As the fraction of concave-convex curvature information approaches one, CCCP will exhibit extremely slow, first order convergence behavior. Figure 3.2 illustrates exactly such an example.

As a confirmation of the above analysis we consider GIS as a reformulated CCCP algorithm. We can express GIS as CCCP [32] by letting $r_i = \exp \Theta_i$ and minimizing the cost function:

$$E(r) = \ln Z(\ln r) - \sum_x \bar{p}(x)(\ln r)^T F(x) \tag{3.22}$$

by defining $E_{vex}(r) = -\sum_x \bar{p}(x)(\ln r)^T F(x)$ and $E_{cave}(r) = \ln Z(\ln r)$. One can easily verify that CCCP update equation $\nabla E_{vex}(r^{t+1}) = -\nabla E_{cave}(r^t)$ results in GIS update equation (3.4). Differentiating convex and concave functions with respect to r twice, and substituting back $r_i = \exp \Theta_i$, we obtain the same exact equations for the convergence matrix as they are given by (3.5) and (3.6). Identical analysis can be performed for the EM algorithm.

3.4 Improving Convergence Rates

The above analyses helped to answer the question: when and why will bound optimizers converge slowly? They can also help to answer the more practical question: what can we do to speed up convergence?

One could use alternative optimization techniques in the regime where the convergence rate matrix has large eigenvalues, and a bound optimizer is likely to perform very poorly. For example, in chapter 4, we will see that for the case of EM, it is possible to estimate the key quantity controlling convergence (fraction of missing information) and switch to direct (gradient-based) optimization when we predict slow behavior of EM. For other bound optimizers, similar hybrid algorithms may be possible.

But there is another, intriguing approach to improving convergence speed: modify the original input to the algorithms based on our analysis of convergence rates. In the case of GIS this involves transforming features, in the case of NMF, this requires translating data vectors, and for CCCP this comes down to designing different convex-concave decompositions of the objective.

Beginning with GIS, we can show the following:

Corollary 3.1 *Homogeneously rescaling all feature vectors by a single constant does not affect convergence.*

Proof sketch: Consider setting $F_{new}(x) = \sigma F(x)\ \forall x$. Corollary 3.1 becomes obvious by analyzing ”scaled” convergence rate matrix $M'_{new}(\Theta^*)$. Indeed:

$$\begin{aligned} M'_{new}(\Theta^*) &= I - \frac{1}{\sigma s}\sigma^2 \text{Cov}(\Theta^*)(\sigma D(\Theta^*))^{-1} \\ &= \frac{1}{\sigma^2 s}\sigma^2 P(\Theta^*) = M'(\Theta^*) \end{aligned} \tag{3.23}$$

Corollary 3.2 *Translating feature vectors to bring them closer to the origin speed up the convergence of GIS. the convergence of GIS. In particular, the optimal translation of features is given by $F_{new}(x) = F(x) - V$ with a vector V containing elements $V_i = \min_x f_i(x)\ \forall i$.*

Proof sketch Consider setting $F_{new}(x) = F(x) - V\ \forall x$ as above. We have

$$M'_{new}(\Theta^*) = I - \frac{1}{s_{new}} \text{Cov}(\Theta^*)\text{D}_{new}(\Theta^*)^{-1} \tag{3.24}$$

with $\text{D}_{new}(\Theta^*) = D(\Theta^*) - \text{diag}(V)$, and $s_{new} = s - \sum_i V_i$. Let us denote $Q(\Theta^*) \equiv \text{Cov}(\Theta^*)\text{D}(\Theta^*)^{-1}$, $Q_{new}(\Theta^*) \equiv \text{Cov}(\Theta^*)\text{D}_{new}(\Theta^*)^{-1}$, and $\lambda_{max}(A) \equiv$ the largest eigenvalue of A. We can now show that this translation forces the top eigenvalue of $M'(\Theta^*)$ to decrease:

$$\lambda_{max}(M'_{new}(\Theta^*)) \leq \lambda_{max}(M'(\Theta^*)) \tag{3.25}$$

where we derived (3.7): $M'(\Theta^*) = I - \frac{1}{s}Q(\Theta^*)$. Note that: $\lambda_{max}(M'_{new}(\Theta^*)) = 1 - \lambda_{min}\big(\frac{1}{s_{new}}Q_{new}(\Theta^*)\big)$. Hence, our task reduces to showing:

$$\begin{aligned} \lambda_{min}\Big(\frac{1}{s_{new}}Q_{new}(\Theta^*)\Big) &\geq \lambda_{min}\Big(\frac{1}{s}Q(\Theta^*)\Big) \\ \Rightarrow \lambda_{max}\big(s_{new}Q^{-1}_{new}(\Theta^*)\big) &\leq \lambda_{max}\big(sQ^{-1}(\Theta^*)\big) \end{aligned} \tag{3.26}$$

Taking into account that $s_{new} \leq s$, the above inequality is obvious by examining:

$$\begin{aligned} \lambda_{max}\big(s_{new}Q^{-1}_{new}(\Theta^*)\big) &= s_{new}\lambda_{max}\big(\big[\text{D}(\Theta^*) - \text{diag}(V)\big]\text{Cov}^{-1}(\Theta^*)\big) \\ &\leq s\lambda_{max}\big(\text{D}(\Theta^*)\text{Cov}^{-1}(\Theta^*)\big) = s\lambda_{max}\big(Q^{-1}(\Theta^*)\big) \end{aligned} \tag{3.27}$$

It is now clear that the optimal translation of features is given by $F_{new}(x) = F(x) - V\ \forall x$ with $V_i = \min_x f_i(x)\ \forall i$.

Corollary 3.3: Decorrelating (whitening) feature vectors speeds up convergence of GIS In particular, the optimal linear transformation $F_{new}(x) = AF(x)$ is that which makes $A\text{Cov}(\Theta^*)A^T$ equal to identity matrix.

Proof sketch Consider spectral decomposition: $\text{Cov}(\Theta^*) = WHW^T$, with H being the diagonal matrix of the eigenvalues, and W being the orthogonal matrix of the corresponding eigenvectors. Let $A = WH^{-1/2}W^T$. The linear transformation becomes $F_{new}(x) = AF(x)$[1], in which case $A\text{Cov}(\Theta^*)A^T = I$. Then:

$$\begin{aligned} M'_{new}(\Theta^*) &= I - \frac{1}{s_{new}} A\text{Cov}(\Theta^*)A^T \text{D}_{new}(\Theta^*)^{-1} \\ &= I - \frac{1}{s_{new}} \text{D}_{new}(\Theta^*)^{-1} \end{aligned} \tag{3.28}$$

with $s_{new} = \max_x \sum_i \big[AF(x)\big]_i$, and $\text{D}_{new}(\Theta^*) = \text{diag}\big[A\sum_x p(x|\Theta^*)F(x)\big] = \text{diag}\big[A\bar{F}(\Theta^*)\big]$. We now show that, in general, $\lambda_{max}(M'_{new}(\Theta^*)) \leq \lambda_{max}(M'(\Theta^*))$. This task reduces to showing (see eq (3.26)):

$$\lambda_{max}\big(s_{new}D_{new}(\Theta^*)\big) \leq \lambda_{max}\big(sQ^{-1}(\Theta^*)\big) \tag{3.29}$$

First note that:

$$\begin{aligned} \lambda_{max}\big(sQ^{-1}(\Theta^*)\big) &= s\lambda_{max}\big(D(\Theta^*)\text{Cov}^{-1}(\Theta^*)\big) = s\lambda_{max}\big(D(\Theta^*)A\big[A\text{Cov}(\Theta^*)A^T\big]^{-1}A^T\big) \\ &= s\lambda_{max}\big(D(\Theta^*)AA^T\big) \end{aligned} \tag{3.30}$$

On the other side:

$$\begin{aligned} s_{new}\lambda_{max}\big(D_{new}(\Theta^*)\big) &= s_{new} \parallel A\bar{F}(\Theta^*) \parallel_\infty \\ &\leq s_{new} \parallel D(\Theta^*)A \parallel_\infty \end{aligned} \tag{3.31}$$

with $\bar{F}(\Theta^*) = \sum_x p(x|\Theta^*)F(x)$ and $D(\Theta^*) = \text{diag}\big[\bar{F}(\Theta^*)\big]$. It can further be shown that $s_{new} \leq s\lambda_{max}(A) = s \parallel A \parallel_2$. By using above facts, slightly more relaxed bound holds:

$$\parallel D(\Theta^*)A \cdot s_{new} \parallel_2 \leq \parallel D(\Theta^*)A \cdot sA \parallel_2 \tag{3.32}$$

Therefore in general, ”whitening” feature vectors, pushes down the top eigenvalue of the convergence rate matrix, which according to our analysis, results in its faster rate of convergence.

[1] Here we are assuming that the new feature vector AF(x) has only positive entries. If AF(x) has negative entries it might be necessary to decorrelate and add a translation, which trades off the advantage of Corollary 3.2 and Corollary 3.3

Of course, the covariance $\mathrm{Cov}(\Theta^*)$ cannot be evaluated until the optimal parameters are known, but it can be approximated by using the sample covariance of features on the training set.

For NMF, similar to GIS, *translating data vectors to bring them closer to the origin* speeds up convergence, whereas homogeneously rescaling all data by a single constant does not affect convergence.

For CCCP, it is well-known that any energy function with bounded curvature has many convex-concave decompositions but no clear principle for finding a good one has been known. Our analysis provides guidance in this regard: we should *minimize the ratio of curvatures between the convex and concave parts of the energy*.

In the next section we illustrate that appropriate preprocessing of the input to these various bound optimization algorithms *does* result in significantly faster rate of convergence.

3.5 Experimental Results

We now present empirical results to support the validity of our analysis for several bound optimization algorithms. We first analyze and apply Generalized Iterative Scaling (GIS) to a logistic regression model. We then show the effect of data translation on the convergence properties of NMF. Finally, we finish by describing and analyzing the effect of various energy function decompositions on the convergence behavior of the CCCP algorithm. Though not shown, we confirmed that the convergence results presented below do not vary significantly for different random initial starting points in the parameter space.

To confirm our analysis of GIS, we applied iterative scaling algorithm to a simple 2-class logistic regression model: $p(y = \pm 1|x, w) = 1/(1 + \exp(-yw^T x))$ [20]. In our first experiment, N feature vectors of dimensionality d were drawn from normal: $x \sim \mathcal{N}(0, 2I_d)$, with the true parameter vector w^* being randomly chosen on the surface of the d-dimensional sphere with radius $\sqrt{2}$. To make features positive, the data set was modified by adding 20 to all

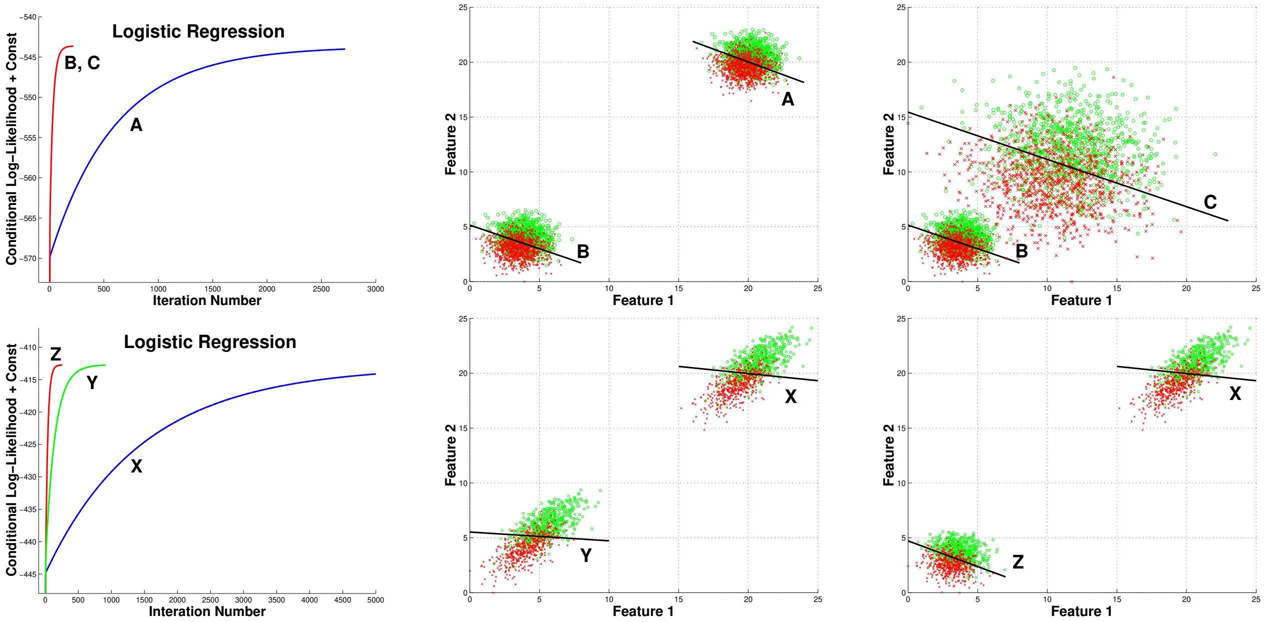

Figure 3.1: Learning curves (left panels) of Iterative Scaling algorithm for logistic regression model, showing the effect that translation and whitening of the feature vectors have on the IS convergence behavior, with letters corresponding to the respective data sets. Top panels show an experiment with 2,000 2-dimensional feature vectors drawn from standard normal, bottom panels display an identical experiment with 2,000 feature vectors drawn from normal with oriented covariance. Top, right panel shows that scaling feature vectors by constant does not affect the convergence of IS.

feature values. Figure 3.1 shows that for $N = 2000$ and $d = 2$, naive IS, that runs on the original unpreprocessed features, takes over 2500 iterations to converge. When feature vectors are translated closer to the origin, IS converges to exactly the same maximum likelihood solution, but beats naive IS by a factor of almost twelve.

Our second experiment was similar, but feature vectors of dimensionality d were drawn from a Gaussian with oriented covariance. Figure 3.1 shows that for N=2,000 and d=2, translating features improves the convergence of IS by a factor of over 4, whereas translating and whitening feature vectors results in speedup by factor of over twenty. Similar results are obtained if dimensionality of the data is increased.

Next, we experimented with the NMF algorithm. Data vectors were drawn from standard normal: $x \sim \mathcal{N}(0, I_{16})$. To make features positive, the data set was modified by adding 20 to all data values, forming non-negative matrix V. We then applied NMF to perform non-negative factorization: $V \approx WH$. Figure 3.2 reveals that naive NMF, that runs on the original

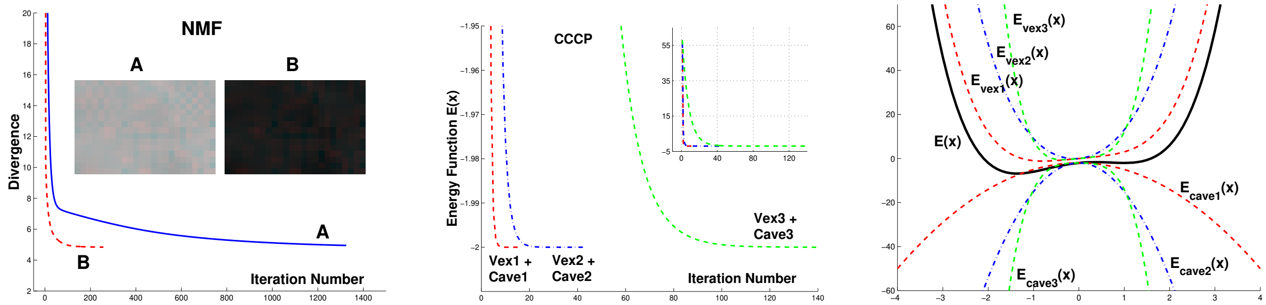

Figure 3.2: Learning curves of NMF and CCCP algorithms. For NMF, we show the effect that data translation has on the convergence behavior of NMF (in our case black pixels correspond to 0, white to 30). Applying CCCP to minimize a simple energy function $E(x) = x^4 - 3x^2 + 2x - 2$, we display the effect that different energy decompositions (left panel) have on the convergence. behavior of CCCP.

unpreprocessed data (data set A), takes over 1,300 iterations to converge. Once data vectors are translated closer to the origin (data set B), NMF converges to exactly the same value of the cost function in about 230 iterations, outperforming naive NMF by a factor of over five.

Finally, we experimented with the CCCP algorithm. We considered a simple energy function $E(x)$=x^4-3x^2+2x-2, which has many decompositions (fig.3.2). A decomposition which minimizes the ratio of concave-convex curvature is: $E_{cave1}(x)$=-3x^2-2 and $E_{vex1}(x)$=x^4+2x. Other decompositions: $E_{cave2}(x)$=-13x^2-2 and $E_{vex2}(x)$=x^4+10x^2+2x; $E_{cave3}(x)$=-9x^4-3x^2-2 and $E_{vex3}(x)$=10x^4+2x; clearly increase the proportion of concave-convex curvature. In our experiment, all runs of CCCP were started from the same initial point in the parameter space. Figure 3.2 reveals that as the proportion of the local concave-convex curvature increases, the convergence rate of CCCP significantly slows down, by several orders of magnitude.

3.6 Discussion

In this chapter we have analyzed a large class of bound optimization algorithms and their relationship to direct optimization algorithms such as gradient-based methods. We have also analyzed and determined conditions under which BO algorithms exhibit local-gradient and fast quasi-Newton convergence behaviors. Based on this analysis and interpretation, we have also provided some recommendations for how the input to these algorithms can be preprocessed to yield faster convergence.

The analysis and experiments motivate the use of alternative optimization techniques in the regime where the convergence rate matrix has large eigenvalues, and a bound optimizer is likely to perform poorly. In particular, slow convergence is expected when missing information is high while learning with EM; when feature vectors are highly dependent while estimating parameters with GIS or NMF; or when the ratio of concave-convex curvature is large when minimizing energy function with CCCP. As we will see in the next chapter, in these cases, direct optimization algorithms such as conjugate-gradient are likely to have far superior performance. Either such alternatives should be employed or else the input should be preprocessed to speed convergence.

Chapter 4

EM and Expectation-Conjugate-Gradient Algorithms

In this chapter we show a close relationship between the Expectation - Maximization (EM) algorithm and direct optimization algorithms such as gradient-based methods for parameter learning. We identify analytic conditions under which EM exhibits Quasi-Newton behavior, and conditions under which it possesses poor, first-order convergence. Based on this analysis, we propose two novel algorithms for maximum likelihood estimation of latent variable models, and report empirical results showing that, as predicted by the theory, the proposed new algorithms can substantially outperform standard EM in terms of speed of convergence in certain cases.

4.1 Introduction

The problem of Maximum Likelihood (ML) parameter estimation for latent variable models is an important problem in the area of machine learning and pattern recognition. ML learning with unobserved quantities arises in many probabilistic models such as density estimation, where one seeks to find a descriptive model of data; dimensionality reduction, where one tries to discover a compact representation of data, or classification, and generally reduces to a relatively hard optimization problem in terms of the model parameters after the hidden quantities have been integrated out.

A common technique for ML estimation of model parameters in the presence of latent variables is Expectation-Maximization (EM) algorithm [4]. The EM algorithm alternates between estimating the unobserved variables given the current model and refitting the model given the estimated, complete data. As such it takes discrete steps in parameter space similar to first order method operating on the gradient of a locally reshaped likelihood function.

In spite of tremendous success of the EM algorithm in practice due to its simplicity and fast initial progress, some authors [23] have argued that the speed of EM convergence can be extremely slow, and that more complicated second-order methods should generally be favored to EM. Many methods have been proposed to enhance the convergence speed of the EM algorithm, mostly based on conventional optimization theory [10, 12]. Louis[16] proposed an approximate Newton's method, known as *Turbo EM*, that first produces Θ_{EM}^{t+1} using the EM iteration with Θ^t as the current fit for Θ, and then uses Θ_{EM}^{t+1} in the conventional Aitken's acceleration method to yield the final iterate Θ^{t+1}. Jamshidian and Jennrich[9] proposed accelerating the EM algorithm by applying generalized conjugate gradient algorithm. where generalized gradient is computed by the change in Θ after performing each EM iterate. Several authors [23, 1] have also proposed hybrid approaches for ML learning, advocating switching to a Newton or Quasi-Newton method after performing several EM iterations. All of these approaches, although sometimes successful in terms of convergence, are much more complex than EM, and difficult to analyze; thus they have not been popular in practice.

The goal of this chapter is to contrast the EM algorithm with a direct gradient-based optimization approach. As a concrete alternative, we present an Expectation-Conjugate-Gradient (ECG) algorithm for maximum likelihood estimation in latent variable models, and show that it can outperform EM in terms of convergence in certain cases. However, in other cases the performance of EM is superior. To understand these behaviors, we study the convergence properties of the EM algorithm and identify analytic conditions under which EM algorithm exhibits Quasi-Newton convergence behavior, and conditions under which it possesses extremely poor, first-order convergence. Based on this analysis, we introduce a simple hybrid EM-ECG algorithm that switches between EM and ECG based on estimated quantities suggested by our analysis. We report empirical results on synthetic as well as real-world data sets, showing that, as predicted by the theory, this simple algorithm almost never performs worse than standard EM and can substantially outperform EM's convergence.

4.2 Linear and Newton Convergence of EM

We now turn to Expectation-Maximization (EM) algorithm and bring the analysis for models which use EM to adjust their parameters. Consider a probabilistic model of observed data $\mathbf{x}$ which uses latent variables $\mathbf{y}$. The log-likelihood (objective function) can be written as:

$$L(\Theta) \;=\; \ln p(\mathbf{x}|\Theta) = \int p(\mathbf{y}|\mathbf{x},\Theta)\ln p(\mathbf{x}|\Theta)d\mathbf{y} \tag{4.1}$$

For any value of Ψ, we can construct a lower bound on the objective function $L(\Theta)$:

$$\begin{aligned} L(\Theta) &= \ln p(\mathbf{x}|\Theta) = \int \ln p(\mathbf{x},\mathbf{y}|\Theta)d\mathbf{y} \\ &\geq \int p(\mathbf{y}|\mathbf{x},\Psi)\ln\frac{p(\mathbf{x},\mathbf{y}|\Theta)}{p(\mathbf{y}|\mathbf{x},\Psi)}d\mathbf{y} \\ &= \int p(\mathbf{y}|\mathbf{x},\Psi)\ln p(\mathbf{x},\mathbf{y}|\Theta)d\mathbf{y} - \int p(\mathbf{y}|\mathbf{x},\Psi)\ln p(\mathbf{y}|\mathbf{x},\Psi)d\mathbf{y} \\ &= Q(\Theta,\Psi) - H(\Psi,\Psi) = G(\Theta,\Psi) \end{aligned} \tag{4.2}$$

The EM algorithm is nothing more than coordinate ascent in the functional $G(\Theta,\Psi)$, alternating between maximizing G with respect to Ψ for fixed Θ (E-step) and with respect to Θ for fixed Ψ (M-step).

As discussed in chapter 2, EM algorithm implicitly defines a mapping: $M : \Theta \rightarrow \Theta$ from parameter space to itself, such that $\Theta^{t+1} = M(\Theta^t)$. If iterates Θ^t converge to Θ^* and $M(\Theta)$ is continuous, then $\Theta^* = M(\Theta^*)$, and in the neighborhood of Θ^*, by Taylor series expansion:

$$\Theta^{t+1} - \Theta^* = M'(\Theta^*)(\Theta^t - \Theta^*) \tag{4.3}$$

where $M'(\Theta^*) = \frac{\partial M}{\partial \Theta}|_{\Theta=\Theta^*}$. Since $M'(\Theta^*)$ is typically nonzero, then EM is essentially a linear iteration algorithm with a convergence rate matrix $M'(\Theta^*)$.

Employing Lemma 2.1, if EM iterates converge to Θ^*, then we can easily establish:

$$\nabla^2_G(\Theta^*) = \frac{\partial^2 G(\Theta, \Psi)}{\partial\Theta\partial\Theta^T}|_{\substack{\Theta = \Theta^* \\ \Psi = \Theta^*}} = \frac{\partial^2 Q(\Theta, \Theta^*)}{\partial\Theta^2}|_{\Theta=\Theta^*} \tag{4.4}$$

$$\nabla^2_G(\Theta^*, \Psi^*) = \left[\frac{\partial^2 G(\Theta, \Psi)}{\partial\Theta\partial\Psi^T}|_{\substack{\Theta = \Theta^* \\ \Psi = \Theta^*}} = -\frac{\partial^2 H(\Theta, \Theta^*)}{\partial\Theta^2}|_{\Theta=\Theta^*} \tag{4.5}\right.$$

And therefore we have:

$$\begin{aligned} \frac{\partial M(\Theta)}{\partial\Theta}|_{\Theta=\Theta^*} &= -\left[\nabla^2_G(\Theta^*, \Psi^*)\right]\left[\nabla^2_G(\Theta^*)\right]^{-1} \\ &= \left[\frac{\partial^2 H(\Theta, \Theta^*)}{\partial\Theta^2}|_{\Theta=\Theta^*}\right]\left[\frac{\partial^2 Q(\Theta, \Theta^*)}{\partial\Theta^2}|_{\Theta=\Theta^*}\right]^{-1} \end{aligned} \tag{4.6}$$

This can be interpreted as the ratio of missing information to complete information near the local optimum [4]. According to Proposition 2.2, in the neighborhood of a solution (for sufficiently large t):

$$\begin{aligned} P(\Theta^t) &\approx \left[I - M'(\Theta^t)\right]\left[-S(\Theta^t)\right]^{-1} \\ &= \left[I - \left(\frac{\partial^2 H}{\partial\Theta^2}\right)\left(\frac{\partial^2 Q}{\partial\Theta^2}\right)^{-1}|_{\Theta=\Theta^t}\right]\left[-S(\Theta^t)\right]^{-1} \end{aligned} \tag{4.7}$$

where $S(\Theta^t) = \frac{\partial^2 L(\Theta)}{\partial\Theta^2}|_{\Theta=\Theta^t}$ is the Hessian of the objective function.

This formulation of the EM algorithm has a very interesting interpretation which is applicable to any latent variable model: *When the missing information is small compared to the complete information, EM exhibits Quasi-Newton behavior and enjoys fast, typically superlinear convergence in the neighborhood of* Θ^*. If fraction of missing information approaches unity, the eigenvalues of the first term (4.7) approach zero and EM will exhibit extremely slow convergence.

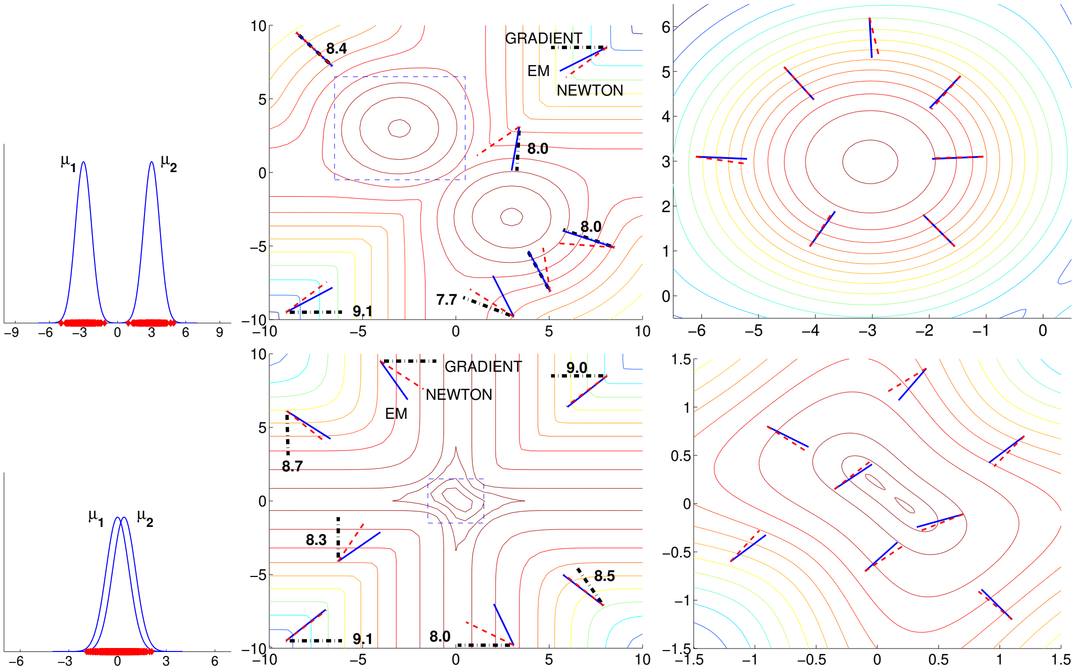

Figure 4.1: Contour plots of the likelihood function $L(\Theta)$ for MoG examples using well-separated (upper panels) and not-well-separated (lower panels) one-dimensional datasets. Axes correspond to the two means. The dashdot line shows the direction of the true gradient $\nabla_L(\Theta)$, the solid line shows the direction of $P(\Theta)\nabla_L(\Theta)$ and the dashed line shows the direction of $(-S)^{-1}\nabla_L(\Theta)$. Right panels are blowups of dashed regions on the left. The numbers indicate the log of the l_2 norm of the gradient. Note that for the "well-separated" case, in the vicinity of the maximum, vectors $P(\Theta)\nabla_L(\Theta)$ and $(-S)^{-1}\nabla_L(\Theta)$ become identical.

Figure 4.1 illustrates the above results in the simple case of fitting a mixture of Gaussians model to well-clustered data – for which EM exhibits Quasi-Newton convergence – and not-well-clustered data, for which EM is slow. As we will see from the empirical results of the later sections, many other models also show this same effect. For example, when Hidden Markov Models or Aggregate Markov Models [26] are trained on very structured sequences, EM exhibits Quasi-Newton behavior, in particular when the state transition matrix is sparse and the output distributions are almost deterministic at each state.

The above analysis and experiments motivates the use of alternative optimization techniques in the regime where missing information is high and EM is likely to perform poorly. In the following section, we analyze exactly such an alternative, the Expectation-Conjugate Gradient (ECG) algorithm, a simple direct optimization method for learning the parameters of latent variables models.

4.3 Expectation Conjugate Gradient (ECG) Algorithm

The key idea of the ECG algorithm is to note that if we can easily compute the derivative $\frac{\partial}{\partial\Theta}\ln p(\mathbf{x},\mathbf{y}|\Theta)$ of the *complete* log likelihood, then knowing the posterior $p(\mathbf{y}|\mathbf{x},\Theta)$ we can compute the exact gradient $\nabla_L(\Theta)$. In particular:

$$\nabla_L(\Theta) = \int_{\mathbf{y}} p(\mathbf{y}|\mathbf{x},\Theta)\frac{\partial}{\partial\Theta}\log p(\mathbf{x},\mathbf{y}|\Theta)d\mathbf{y} \tag{4.8}$$

This exact gradient can then be utilized in any standard manner, for example to do gradient (as)descent or to control a line search technique. (Note that if one can derive exact EM for a latent variable model, then one can always derive ECG by computing the above integral over hidden variables.) As an example, we describe a conjugate gradient algorithm:

Expectation-Conjugate-Gradient algorithm:
Apply a conjugate gradient optimizer to $L(\Theta)$, performing an "EG" step whenever the value or gradient of $L(\Theta)$ is requested (e.g. during a line search).
The gradient computation is given by

- **E-Step:** Compute posterior $p(\mathbf{y}|\mathbf{x},\Theta^t)$ and log-likelihood $L(\Theta)$ as normal.
- **G-Step:** $\nabla_L(\Theta^t) = \int p(\mathbf{y}|\mathbf{x},\Theta^t)\frac{\partial}{\partial\Theta}\log p(\mathbf{x},\mathbf{y}|\Theta)d\mathbf{y}$

When certain parameters must obey positivity or linear constraints, we can either modify our optimizer to respect the constraints, or we can reparameterize to allow unconstrained optimization. In our experiments, we use simple reparameterizations of model parameters that allow our optimizers to work with arbitrary values. For example, in the MoG model we use a "softmax" parameterization of the mixing coefficients $\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^{M}\exp(\gamma_j)}$, for covariance matrices to be symmetric positive definite, we use the Choleski decomposition (or log variances for diagonal covariance matrices). In HMMs, we reparameterize probabilities via softmax functions as well. In the appendix B we derive and describe ECG algorithm for several latent variable models.

Of course, the choice of initial conditions is very important for the EM algorithm or for ECG. Since EM is based on optimizing a convex lower bound on the likelihood, once EM is trapped in a poor basin of attraction, it can never find a better local optimum. Algorithms such as split and merge EM [29] were developed to escape from such configurations. It turns out that direct optimization methods such as ECG may also avoid this problem because of the nonlocal nature of the line search. In many of our experiments, ECG actually converges to a better local optimum than EM; figure 4.2 illustrates exactly such case.

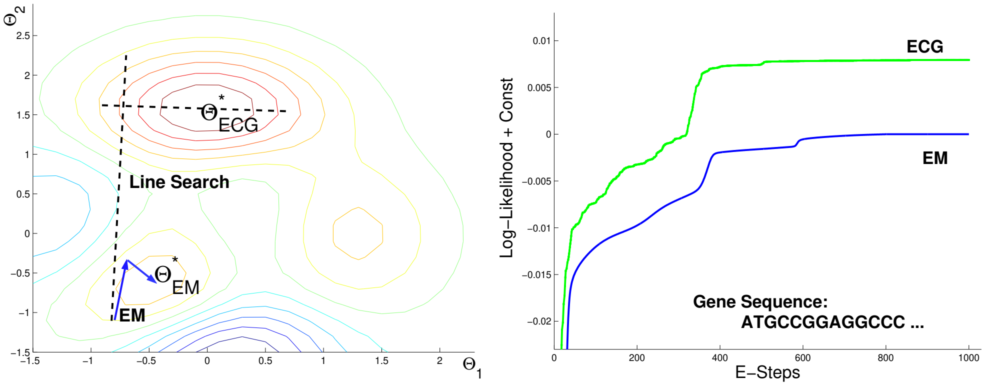

Figure 4.2: Left panel pictorially illustrates why ECG may converge to a better local optimum. Right panel displays the learning curves for EM and ECG algorithms of training fully connected 7-state HMM to model human DNA sequences. Both algorithms started from the same initial parameter values, but converged to the different local optimum.

4.4 Hybrid EM-ECG Algorithm

As we have seen, the relative performance of EM versus direct optimization depends on the missing information ratio for the given model and data set. The key to practical speedups is the ability to design a hybrid algorithm that can estimate the local missing information ratio $M'(\Theta^t)$ to detect whether to use EM or a direct approach such as ECG. Some authors have attacked this problem by finding the top eigenvector of $\frac{\partial M(\Theta)}{\partial \Theta}|_{\Theta=\Theta^t}$ as Θ^t approaches Θ^* using conventional numerical methods, such as finite-difference approximations, or power methods [5]. These approaches are computationally intensive and difficult to implement, and thus they have not been popular in practice.

Here, we propose using the *entropy of the posterior over hidden variables*, which can be computed after performing an E-step, as a crude estimate of the local missing information ratio. This entropy has a natural interpretation as the uncertainty about missing information, and thus can serve as a guide between switching regimes of EM and ECG. For many models with discrete hidden variables this quantity is quite easy to compute. In particular, we define the Normalized Entropy term:

$$\bar{H}_t = \tfrac{-1}{N \ln M} \textstyle\sum_n^N \sum_i^M p(\mathbf{y}=i|\mathbf{x}_n, \Theta^t) \ln p(\mathbf{y}=i|\mathbf{x}_n, \Theta^t) \tag{4.9}$$

with $\mathbf{y}$ being discrete hidden variable taking on M values, and N observed data vectors $\mathbf{x}_n$. We now simply switch between EM and ECG based on thresholding this quantity:

Hybrid EM-ECG algorithm:

- Perform EM iterations, evaluating $\bar{H}_t$ after each E-step
- If $\bar{H}_t \geq \tau$ Then[a] Switch to ECG
- Perform ECG, evaluating $\bar{H}_t$ at the end of each line search
- If $\bar{H}_t < \tau$ Then Switch back to EM
- Exit at either phase **IF**:
    1. $(L(\Theta^t) - L(\Theta^{t-1}))$/abs$(L(\Theta^t)) <$ tol **OR**
    2. t > Tmax

[a] We are near the optimum or in plateau region with high entropy

As we will see from the experimental results, this simple hybrid EM-ECG algorithm performs no worse, and often far better than either EM or ECG.

4.5 Experimental Results

We now present empirical results comparing the performance of EM, ECG, and hybrid EM-ECG for learning the parameters of three latent variable models: Mixtures of Gaussians (MoG), Hidden Markov Models (HMM), and Aggregate Markov Models. In many latent variable models, performing inference (E-step) is significantly more expensive compared to either the parameter updates (M-step) or the line search overhead in the CG step of ECG. To compare the performance of the algorithms, we therefore simply compare the number of E-steps each algorithm executes until its convergence.

We first show results on synthetic data sets, whose properties we can control to verify certain aspects of our theoretical analysis. We also report empirical results on several real world data sets, showing that our algorithms do work well in practice. Though we show examples of single runs, we have confirmed that the convergence results presented in all our experiments do not vary significantly for different initial parameter conditions. For all of the reported experiments, we used tol = 10^{-8} and $\tau = 0.5$.

4.5.1 Synthetic Data Sets

First, consider a mixture of Gaussians (MoG) model. We considered two types of data sets, one in which the data is "well-separated" into distinct clusters and another "not-well-separated" case in which the data overlaps in one contiguous region. Figure 4.3 shows that ECG and Hybrid EM-ECG outperform standard EM in the poorly separated cases. For the well-separated case, the hybrid EM-ECG algorithm never switches to ECG due to the small normalized entropy term, and EM converges very quickly. This is predicted by our analysis: in the vicinity of the local optima Θ^* the directions of the vectors $P(\Theta)\nabla_L(\Theta)$ and $(-S)^{-1}\nabla_L(\Theta)$ become identical (fig. 4.1), suggesting that EM will have Quasi-Newton convergence behavior.

We then applied our algorithms to the training of Hidden Markov Models (HMMs). Missing information in this model is high when the observed data do not well determine the underlying state sequence (given the parameters). We therefore generated two data sets from a 5-state HMM, with an alphabet size of 5 characters. The first data set ("aliased" sequences) was generated from a HMM where output parameters were set to uniform values plus some small noise. The second data set ("very structured sequences") was generated from a HMM with sparse transition and output matrices. For the ambiguous or aliased data, ECG and hybrid EM-ECG outperform EM substantially. For the very structured data, EM performs well and

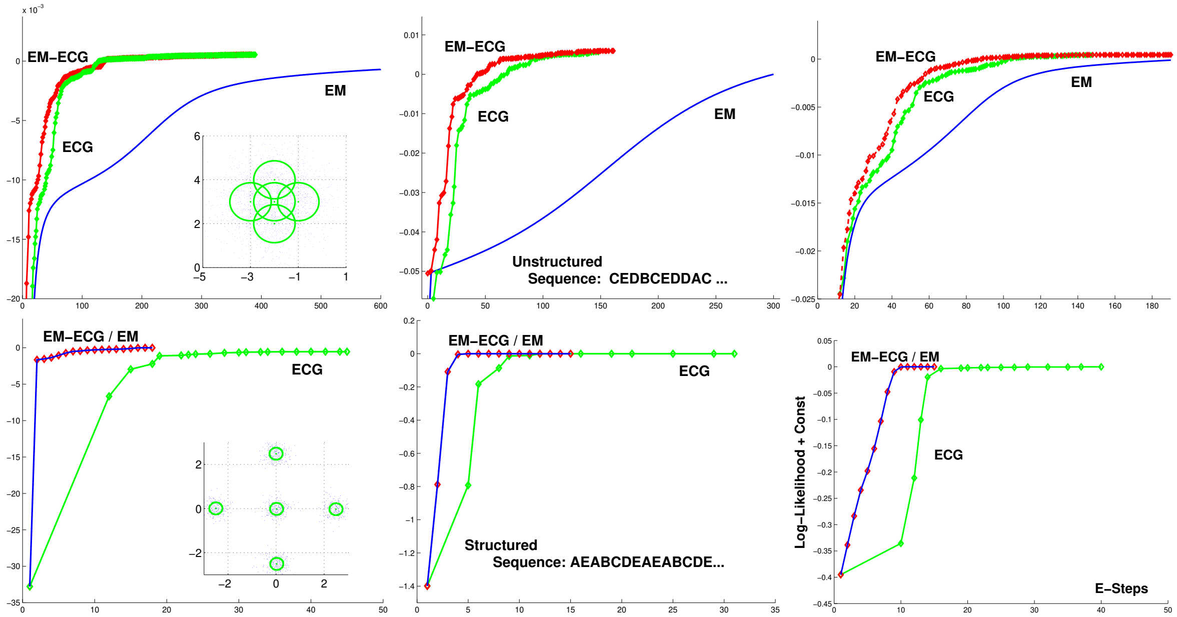

Figure 4.3: Learning curves for ECG, EM-ECG, and EM algorithms, showing superior (upper panels) and inferior (lower panels) performance of ECG under different conditions for three models: MoG (left), HMM (middle), and Aggregate Markov Models (right). The number of E-steps taken by either algorithm is shown on the horizontal axis, and log likelihood is shown on the vertical axis. For ECG and EM-ECG, diamonds indicate the maximum of each line search, and the zero-level likelihood corresponds to the converging point of the EM algorithm. The bottom panels use "well-separated", or "structured" data for which EM possesses Quasi-Newton convergence behavior. All models in this case converge in 10-15 iterations with stopping criterion: $[L(\Theta^{t+1}) - L(\Theta^{t})]/\text{abs}(L(\Theta^{t+1})) < 10^{-15}$. The upper panels use "overlapping", "aliased", or "unstructured" data for which proposed algorithms performs much better.

exhibits second order convergence in the vicinity of the local optimum.[1]

Finally, we experimented with Aggregate Markov Models (AMMs) [26]. AMMs model define a discrete conditional probability table $p_{ij} = p(y = j|x = i)$ using a low rank approximation. In the context of n-gram models for word sequences, AMMs are class-based bigram models in which the mapping from words to classes is probabilistic. In particular, the class-based bigram model predicts that word w_1 is followed by word w_2 with probability: $P(w_2|w_1) = \sum_{c=1}^{C} P(w_2|c)P(c|w_1)$ with C being the total number of classes. Here, the concept of missing information corresponds to how well or poor a set of words determine the class

[1] ECG performs better on hard cases, whereas EM does better on easy cases. One does not seem to loose much by just applying ECG instead of EM.

labels C based on the observation words that follow them. We therefore generated two data sets for 50-state AMM model. For the first data set ("ambiguous", "aliased") the transition matrix was set to uniform values plus some small noise. In this case words do not well determine class labels. For the second data set ("structured") the transition matrix was set to be very sparse, in which case the proportion of missing information is very small. The right panels of figure 4.3 show training of a 2-class 50-state AMM model on ambiguous data, and on more structured data. ECG and hybrid EM-ECG are superior to EM by at least a factor of two for ambiguous data; for structured data EM shows the expected Quasi-Newton convergence behavior.

4.5.2 Real World Data Sets

In our first experiment, we cluster a set of 50,000 8×8 grayscale pixel image patches[2] using a mixture of Gaussians model. The patches were extracted from 768×512 natural images, described in [30] (see fig 4.4 for an example of a natural image, and sample patches). To speed-up the experiments, the patch data was projected with PCA down to a 10-dimensional linear subspace and the mixing proportions and covariances of the model were held fixed. The means were initialized by performing K-means. We experimented with mixtures having M=2 up to M=65 clusters.

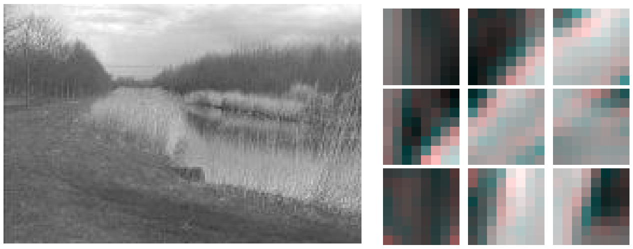

Figure 4.4: An example of a natural image and some samples of 8×8 gray pixel image patches, used in the clustering experiment.

Figure 4.5 displays the convergence of EM, ECG, and Hybrid EM-EC algorithms for M=5, M=50 and M=65. The experimental results show that with fewer mixture components EM

[2] The data set used was the **imlog** data set publicly available at ftp://hlab.phys.rug.nl/pub/samples/imlog

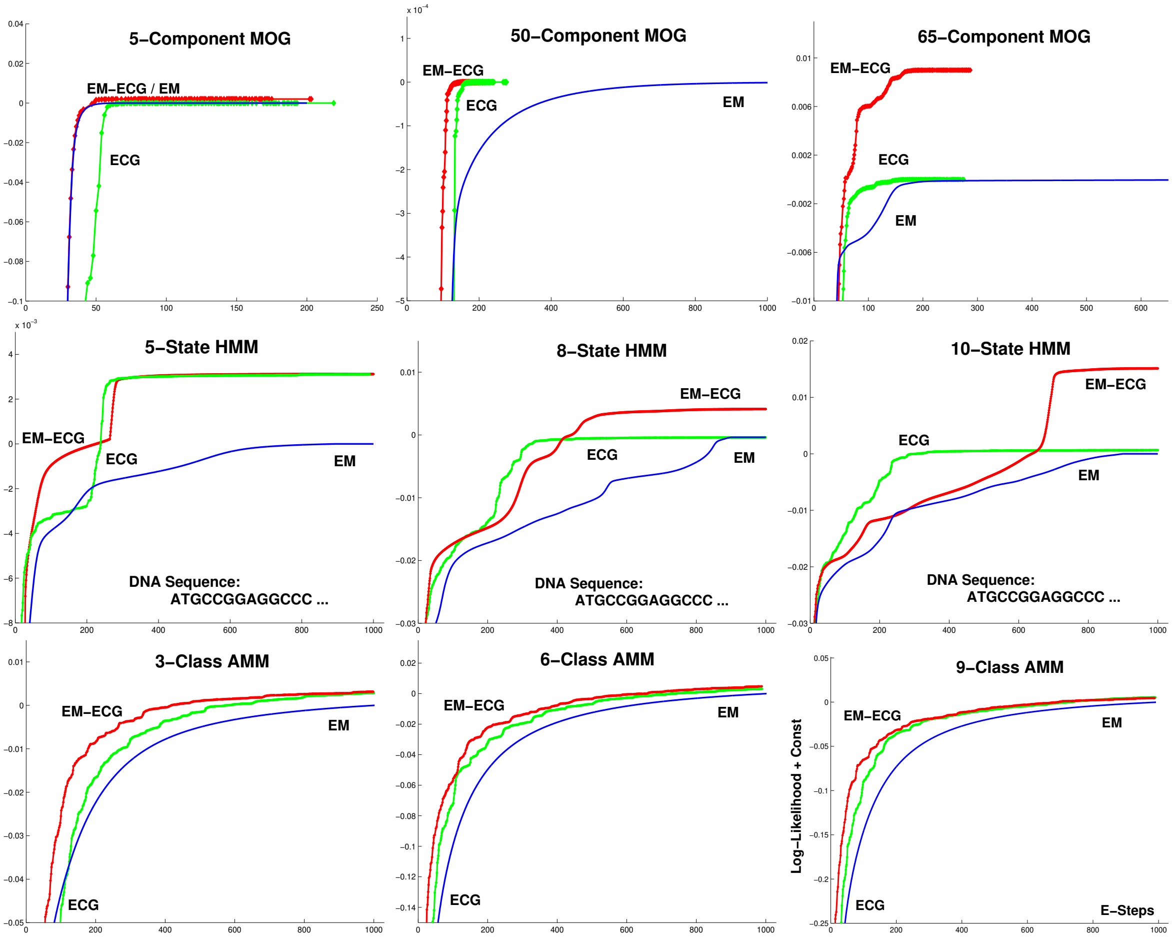

Figure 4.5: Learning curves for ECG, EM-ECG, and EM algorithms, displaying convergence performance under different conditions for three models: MoG (upper), HMM (middle), and Aggregate Markov Models (bottom). The number of E-steps taken by either algorithm is shown on the horizontal axis, and log likelihood is shown on the vertical axis. For ECG and EM-ECG, diamonds indicate the maximum of each line search, and the zero-level likelihood corresponds to the converging point of the EM algorithm. The number of learned clusters for MoG model were 5 (left), 50 (middle), and 65 (right). For HMM model, the number of states were 5 (left), 8 (middle), and 10 (right). The number of learned themes for the AMM model were 3 (left), 6 (middle), and 9 (right).

outperforms ECG, since the components generally model the data with fairly distinct, non-contiguous clusters. As the number of mixtures components increases, clusters overlap in contiguous regions and the normalized entropy term grows, suggesting a relatively high proportion of the missing information. In this case ECG outperforms EM by several times.

Our second experiment consisted of training a fully connected HMM to model DNA sequences. For the training, we used publicly available ”GENIE gene finding data set”, provided by UCSC and LBNL [6], that contains 793 unrelated human genomic DNA sequences. We applied our different algorithms on 66 DNA sequences with length varying anywhere between 200 to 3000 multiple exon and single exon genes per sequence. The number of states ranged from M=5 to M=10 and all the parameter values were randomly initialized. Figure 4.5 shows the convergence of EM, ECG, and Hybrid EM-ECG algorithms for M=5,8,10. This data set contains very complex structure which is not easily modeled by HMMs, resulting in a very high proportion of missing information. As a result, hybrid EM-ECG and ECG substantially outperform EM in terms of convergence.

In our last experiment, we applied Aggregate Markov Models to the data set consisting of 2,037 NIPS authors and corresponding counts of the top 1,000 most frequently used words of the NIPS conference proceedings, volumes 1 to 12.[3] The goal was to model the probability that an author A will use word W using a small number of "soft" classes (t): $P(W|A) = \sum_{t=1}^{T} P(W|t)P(t|A)$. Once again, we observe that for this simple model, this data set has a large fraction of missing information. Figure 4.5 displays the convergence of EM, ECG, and EM-ECG algorithms for T=3,6,9. with hybrid EM-ECG and ECG having superior convergence over EM.

4.6 Discussion

Although we have focused here on discrete latent variables, the ECG and hybrid algorithms can also be derived for latent variable models with continuous hidden variables. As an example figure 4.6 illustrates convergence behavior of the Probabilistic Principal Component Analysis (PPCA) latent variable model[24, 28], which has continuous rather than discrete hidden variables. Here the concept of missing information is related to the ratios of the leading eigenvalues of the sample covariance, which corresponds to the ellipticity of the distribution. For "low-rank" data with a large ratio EM performs well; for nearly circular data ECG converges

[3] NIPS corpus used in the experiments is publicly available at http://www.cs.toronto.edu/~roweis/data.html

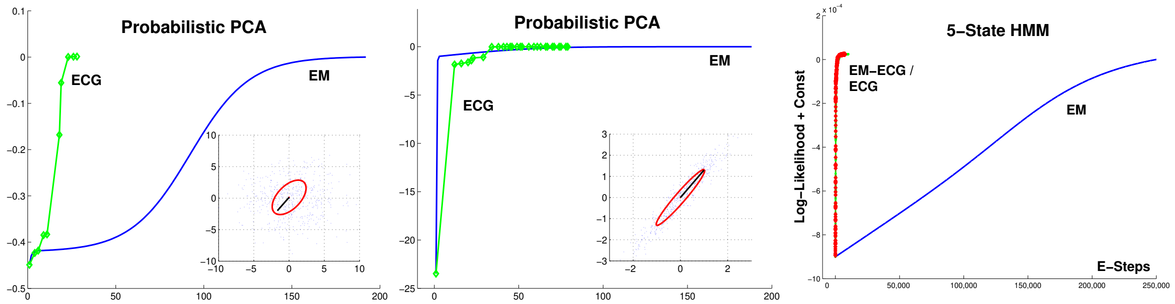

Figure 4.6: Learning curves for ECG (dots) and EM (solid lines) algorithms, showing superior (left) and inferior (middle) performance of ECG. The left panel uses ”ill-conditioned” data for which ECG converges quickly; the middle panel uses “low-rank” data for which EM performs better. Right panel displays ”non-converging” case of the EM. Very unstructured data (30 sequences, each of length 50) was generated from a full 5-state HMM with alphabet size of 5. Parameter values were set to be uniform plus some small uniform noise. ECG and EM-ECG converge in about 7,000 iterations, whereas after even 250,000 iterations, EM is only approaching to the ML estimate.

faster.

As a confirmation that this behavior is in accordance with our analysis, in figure 4.7 we show the evolution of the eigenvalues of the matrix $\left(\frac{\partial^2 H}{\partial \Theta^2}\right)\left(\frac{\partial^2 Q}{\partial \Theta^2}\right)^{-1}$ during learning of the shown datasets, generated from known parameters for which we can compute this missing information matrix exactly. For the well-separated MoG case the eigenvalues of the matrix approach zero, and the ratio of missing information to the complete information becomes very small, driving $P(\Theta)$ toward the negative of the inverse Hessian. Interestingly, in the case of PPCA, even though the determinant of the matrix approaches zero, one of its eigenvalues remains nonzero even in the low-rank data case (fig. 4.7). This suggests that the convergence of the EM algorithm for PPCA can still be slow very close to the optimum in certain directions in parameter space, even for “nice” data.

The slow convergence of EM in PPCA is also true for FA and especially for linear dynamic systems. In these models, there is large amount of missing information due to the fact that latent variables are continuous and they can be rotated without affecting the likelihood as long as the parameters are rotated accordingly.

In some degenerate cases, where the proportion of missing information is very high, i.e.

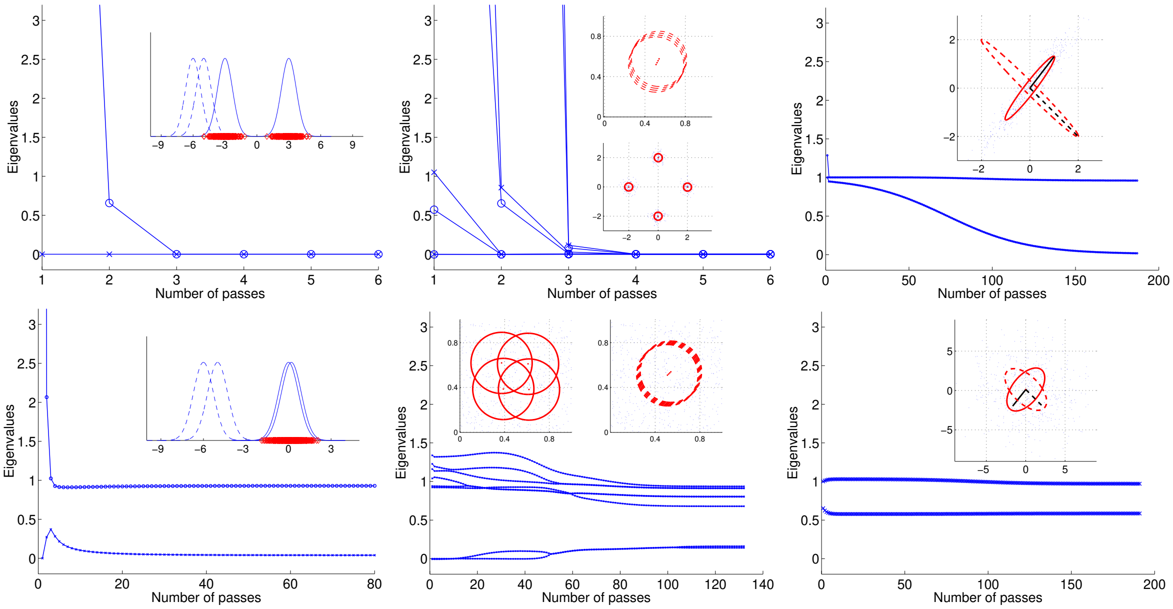

Figure 4.7: Eigenvalues of matrix in eq.(4.6) for MoG (left, centre) and PPCA (right) using well-separated or low-rank (top) and not-well-separated or circular (bottom) datasets of 3000 points. Insets show data and initial/final conditions: initial conditions are shown as dashed circles at the one-σ contour of each model; final converged model is shown by solid circles.

$M'(\Theta^*)$ approaches identity, EM convergence can be exponentially slow. Figure 4.6 (right panel) illustrates such example for the case of HMM training using almost random sequences. It takes about 7,000 iterations for ECG and EM-ECG to converge to the ML estimate, whereas even after 250,000 iterations EM is still only approaching the local optimum.

In this chapter we have presented comparative analysis of EM and direct optimization algorithms for latent variable models, and developed a theoretical connection between these two approaches. We have also analyzed and determined conditions under which EM algorithm can demonstrate local-gradient and Quasi-Newton convergence behaviors. Our results extend those of Xu and Jordan[31] who analyzed the convergence properties of the EM algorithm in the special case of Gaussian mixtures, and apply to any exponential family model.

Motivated by these analyses, we have proposed an alternative hybrid optimization method that can significantly outperform EM in many cases and is almost never inferior. We tested the proposed algorithms by training several basic latent variable models on several synthetic as well as real world data sets, reporting convergence behavior and explaining the results with reference to our analysis.

Chapter 5

Adaptive Overrelaxed Bound Optimization Methods

In this chapter we study a class of *overrelaxed* bound optimization algorithms, and their relationship to standard bound optimizers, such as Expectation-Maximization, Iterative Scaling, CCCP and Non-Negative Matrix Factorization. We provide a theoretical analysis of the convergence properties of these optimizers and identify analytic conditions under which they are expected to outperform the standard versions. Based on this analysis, we propose a novel, simple adaptive overrelaxed scheme for practical optimization and report empirical results on several synthetic and real-world data sets showing that these new adaptive methods exhibit superior performance (in certain cases by several times) compared to their traditional counterparts. Our "drop-in" extensions are simple to implement, apply to a wide variety of algorithms, almost always give a substantial speedup, and do not require any theoretical analysis of the underlying algorithm.

5.1 Overrelaxed Bound Optimization: BO(η)

In chapter 1 we described a general form of a bound maximizer which iteratively lower bounds the objective function. These standard algorithms enjoy a strong guarantee: they never worsen the objective function. However, to guarantee an increase in the objective function at each iteration, BO methods must sometimes construct very conservative bounds, resulting in extremely slow convergence behavior. Below, we analyze a family of *overrelaxed* BO algorithms called BO(η) algorithms with η denoting the overrelaxation learning rate. The basic idea that lies behind overrelaxed bound optimization algorithms is very simple: produce a new parameter vector Θ^{t+1} by performing standard bound optimization with Θ^t and go a little further in the direction of $\Theta^{t+1} - \Theta^t$.

Naive Overrelaxed BO(η) algorithm for maximizing $L(\Theta)$:
• **Assume**: $\exists$ function $G(\Theta, \Psi)$ such that for
any given Θ' and Ψ':

1. $G(\Theta', \Theta') = L(\Theta')$ & $L(\Theta) \geq G(\Theta, \Psi') \; \forall \; \Theta \neq \Psi'$
2. $\arg\max_{\Theta} G(\Theta, \Psi')$ can be found easily.

• **Iterate**: $\Theta^{t+1} = \Theta^t + \eta\big(\arg\max_{\Theta} G(\Theta, \Theta^t) - \Theta^t\big)$
• **No convergence guarantee.**
• **Difficult to set η in practice.**

Clearly, for $\eta = 1$ BO(η) algorithms become just regular bound optimizers. Several authors have studied a particular variant of this idea as applied to Expectation Maximization. In particular, Helmbold et al. (1995) [8] investigated the problem of estimating the component priors for a mixture of given densities, and discovered that an EM(η) update rule can be viewed as a first order approximation to the exponentiated gradient EG(η) update. Following this, Bauer et al. (1997) [2] presented an analysis of EM(η) similar to [8] and derived the update rules for parameter estimation in discrete Bayesian networks. However, more general BO(η) methods have not been widely used for several reasons. First, only one particular variant of BO(η) is well studied: EM(η), and update rules have been published only in the special case of discrete Bayesian networks. Second, if a learning rate larger than optimal is used, BO(η) algorithms cannot guarantee convergence to even a local optimum of their objective function, in contrast to standard bound optimizers. Finally, it is computationally very difficult to obtain the optimal learning rate η^*.

In this chapter, we analyze a broad class of BO(η) algorithms beyond EM(η) and show how one can design a simple adaptive algorithm that, in general, will possess superior convergence rates over standard BO(1) methods while at the same time guaranteeing convergence and avoiding the need to calculate an optimal learning rate η^*.

5.2 Convergence Properties of BO(1) and BO(η)

In chapter 2, we showed that standard bound optimization methods implicitly define a mapping: $M : \Theta \rightarrow \Theta'$ from parameter space to itself, such that $\Theta^{t+1} = M(\Theta^t)$. In particular, if the iterates Θ^t converge to Θ^* and $M(\Theta)$ is continuous, then $\Theta^* = M(\Theta^*)$, and in the neighborhood of Θ^*, by Taylor series expansion:

$$\Theta^{t+1} - \Theta^* = M'(\Theta^*)(\Theta^t - \Theta^*) \tag{5.1}$$

where $M'(\Theta^*) = \frac{\partial M}{\partial \Theta}|_{\Theta=\Theta^*}$. Since $M'(\Theta^*)$ is typically nonzero, then any bound optimizer is essentially a linear iteration algorithm with a convergence rate matrix $M'(\Theta^*)$.

For multidimensional vector Θ, a measure of the actual observed convergence rate is the "global" rate, defined as:

$$r = \lim_{t\rightarrow\infty} \frac{\|\Theta^{t+1}-\Theta^*\|}{\|\Theta^t-\Theta^*\|} \tag{5.2}$$

with $\| \textbf{.} \|$ being Euclidean norm[19]. Intuitively, the "global" rate of convergence measures by how much the new fit Θ^{t+1} of the parameter vector Θ is getting closer to the local optimum compared to the previous fit Θ^t. It is also well-known that under some regularity conditions $r = \lambda_{max}(M') \equiv$ the largest eigenvalue of $M'(\Theta^*)$. All of the eigenvalues of the convergence rate matrix $M'(\Theta^*)$ lie in the interval $[0, 1)$. Larger values of λ_{max} (as they approach unity) imply slower convergence.

BO(η) methods, just as standard bound optimizers, implicitly define a mapping: $\Phi : \Theta \rightarrow \Theta$ from parameter space to itself, such that $\Theta^{t+1} = \Phi(\Theta^t)$. In particular,

$$\Phi(\Theta^t) = \Theta^t + \eta(\Theta^{t+1}_{BO} - \Theta^t) = \Theta^t + \eta(M(\Theta^t) - \Theta^t) \tag{5.3}$$

We can now analyze convergence behavior of the BO(η) methods as well as their relationship to the standard BO(1) algorithms.

Lemma 4.1 *If BO(η) iterates converge to Θ^*, then for any value of η, $\Phi(\Theta^*) = \Theta^*$.*

Proof: This follows from (5.3) due to the necessity of a fixpoint of the mapping M: $M(\Theta^*) = \Theta^*$.

Lemma 4.2 *If BO(η) iterates converge to Θ^* and $\Phi(\Theta)$ and $M(\Theta)$ are differentiable in the parameter space Θ, then:*

$$\Phi'(\Theta^t) \;=\; I - \eta(I - M'(\Theta^t)) \tag{5.4}$$

with I being identity matrix and $\Phi'_{ij}(\Theta^t) = \frac{\partial \Theta_i^{t+1}}{\partial \Theta_j^t}$ is the input-output derivative matrix for the BO(η) mapping.

Proof: The derivatives of both sides of (5.3) with respect to Θ are well-defined, and therefore we have:

$$\Phi'(\Theta^t) \;=\; I + \eta(M'(\Theta^t) - I) = I - \eta(I - M'(\Theta^t)) \tag{5.5}$$

Equation (5.4) shows a very interesting relationship between convergence properties of BO(η) and its standard BO(1) counterparts. If the eigenvalues of $M'(\Theta^t)$ approach unity in the neighborhood of Θ^*, BO(1) algorithm will exhibit extremely slow convergence. In this case, larger values of η will in fact force the eigenvalues of $\Phi'(\Theta^t)$ to decrease, and thus result in faster global rate of convergence of the BO(η) algorithm.

Proposition 4.1 *If BO(1) iterates converge to Θ^*, then within some neighborhood of Θ^*, BO(η) algorithm will converge to the local maximum of the objective function for any*
$0 < \eta < 2$.

Proof:[1] First, by using Taylor series expansion of $\Phi(\Theta^t)$ around Θ^*, we have:

$$\Phi(\Theta^t) = \Phi(\Theta^*) + \Phi'(\Theta^*)(\Theta^t - \Theta^*) + ... \tag{5.6}$$

For sufficiently large t, we have the following linear approximation:

$$\Theta^{t+1} - \Theta^* = \Phi(\Theta^t) - \Phi(\Theta^*) \approx \Phi'(\Theta^*)(\Theta^t - \Theta^*) \tag{5.7}$$

[1] Our proof is similar in spirit to [2]. Nevertheless, we feel that it is worth going through the details since it will play an important role in our further analysis.

For a fixed η, consider $\gamma_1, ..., \gamma_k$ being the non-zero eigenvalues of $\Phi'(\Theta^*)$. The corresponding eigenvectors $v_1, ..., v_k$ form an orthonormal basis for the real k-subspace Ω.[2] Assume that for sufficiently large t, $(\Theta^t - \Theta^*) \in \Omega$, in which case a vector $(\Theta^t - \Theta^*)$ can be represented uniquely as $(\Theta^t - \Theta^*) = \sum_{i=1}^{k} c_i v_i$. Moreover

$$\Phi'(\Theta^*)(\Theta^t - \Theta^*) = \sum_{i=1}^{k} \gamma_i c_i v_i \tag{5.8}$$

The application of $\Phi'(\Theta^*)$ to $(\Theta^t - \Theta^*)$ results in linear coefficients c_i to be scaled by γ_i. Therefore the rate at which different components of $(\Theta^t - \Theta^*)$ shrink or stretch depends on the size of the eigenvalues γ_i. To guarantee the shrinkage of each component of $(\Theta^t - \Theta^*)$, we require $|\gamma_i| < 1$ for i=1,...k. In this case:

$$\| \Phi'(\Theta^*)(\Theta^t - \Theta^*) \| = \| \sum_{i=1}^{k} \gamma_i c_i v_i \| < \| \sum_{i=1}^{k} c_i v_i \| = \| \Theta^t - \Theta^* \| \tag{5.9}$$

with $\| \cdot \|$ denoting Euclidean norm.[3] Indeed:

$$\| \Phi'(\Theta^*)(\Theta^t - \Theta^*) \| = \sqrt{\sum_{i=1}^{k} \gamma_i^2 c_i^2 v_i^T v_i} \leq \rho_\eta \sqrt{\sum_{i=1}^{k} c_i^2 v_i^T v_i} = \rho_\eta \| \Theta^t - \Theta^* \| \tag{5.10}$$

with ρ_η being the spectral radius of $\Phi'(\Theta^*)$: ρ_η=max$|\lambda|$. Clearly, the spectral radius of $\Phi'(\Theta^*)$ is defined as:

$$\rho_\eta = \max\{|1 - \eta(1 - \lambda_{max})|, |1 - \eta(1 - \lambda_{min})|\}$$

with λ_{max} and λ_{min} being the largest and smallest eigenvalues of $M'(\Theta^*)$. And thus for any $0 < \eta < 2$ we have $\rho_\eta < 1$. We can now analyze the global rate of convergence of $\Phi'(\Theta^t)$ in the neighborhood of Θ^*. In particular, for sufficiently large t and any $0 < \eta < 2$:

$$r = \frac{\| \Theta^{t+1} - \Theta^* \|}{\| \Theta^t - \Theta^* \|} = \frac{\| \Phi'(\Theta^*)(\Theta^t - \Theta^*) \|}{\| \Theta^t - \Theta^* \|} \leq \frac{\rho_\eta \| \Theta^t - \Theta^* \|}{\| \Theta^t - \Theta^* \|} = \rho_\eta < 1 \tag{5.11}$$

[2] We note that $M'(\Theta^*)$ has exactly the same eigenvectors with eigenvalues defined as $\lambda_i = 1 - (1 - \gamma_i)/\eta$ for $i = 1, ..., k$.

[3] This argument is valid for any norm defined on the k-dimensional Euclidean space. However, for the sake of simplicity of the proof, we reserve to the Euclidean norm.

Therefore within some neighborhood of Θ^* for any $0 < \eta < 2$, BO(η) algorithm is guaranteed to converge to the local optimum of the objective function.[4] This concept is graphically illustrated in figure 5.1.

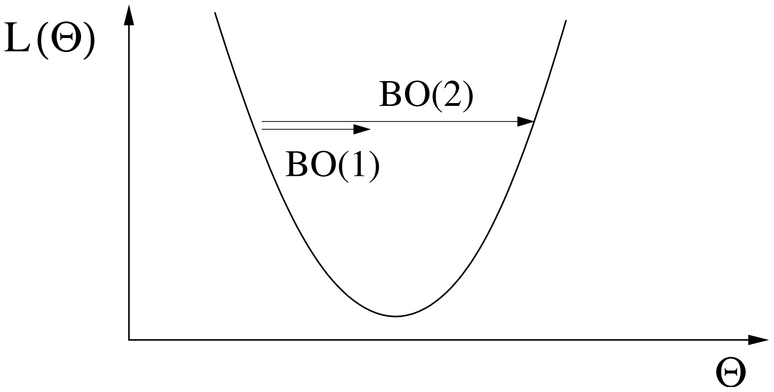

Figure 5.1: In the vicinity of local optimum we have quadratic approximation of the objective function $L(\Theta)$. In this case going twice as far as a standard bound optimization algorithm guarantees not to decrease $L(\Theta)$.

Corollary 4.1 *The optimal learning rate η^* is:*

$$\eta^* = 2/(2 - \lambda_{max} - \lambda_{min}) \tag{5.12}$$

with λ_{max} and λ_{min} being the largest and smallest eigenvalues of $M'(\Theta^*)$. Moreover, $\eta^* \geq 1$.

Proof: We have established that the global rate of convergence of the BO(η) algorithms is:

$$r = \max\{|1 - \eta(1 - \lambda_{max})|, |1 - \eta(1 - \lambda_{min})|\}$$

Clearly, the fastest uniform global rate of convergence is obtained when $|1 - \eta(1 - \lambda_{max})|$ = $|1 - \eta(1 - \lambda_{min})|$. We can now easily derive that $\eta^* = 2/(2 - \lambda_{max} - \lambda_{min})$. Since $0 \leq \lambda_{min} \leq \lambda_{max} < 1$, we have

$$\eta^* = 2/(2 - \lambda_{max} - \lambda_{min}) \geq 1 \tag{5.13}$$

The important consequence of the above analysis is that for the typical real problems with $\lambda_{max} > 0$, the optimal learning rate is $\eta^* > 1$. This implies that, within some neighborhood

[4] The above analysis is only valid in the vicinity of Θ^* as opposed to standard bound optimizers BO(1), that guarantee to converge to some local optimum Θ^* from any point in the parameter space.

of Θ^*, BO(η) methods can significantly outperform standard BO(1) algorithms in terms of convergence. Indeed, after M iterations BO(η) will shrink the distance $\| \Theta - \Theta^* \|$ by a factor of ρ_η^M, whereas standard BO(1) will shrink it by ρ_1^M. This clearly constitutes exponential gain of $(\rho_\eta/\rho_1)^M$ in the vicinity of the local optimum.

5.3 Adaptive Overrelaxed Bound Optimization

Computing the optimal learning rate may be very expensive, since it requires knowledge of the minimum and maximum eigenvalues λ_{min} and λ_{max} of a particular mapping matrix that depends on the algorithm details, data set, and current parameters. Furthermore, this calculation is only valid in a very small neighborhood around a local optimum. Ideally, we would like to find the optimal learning rate in an adaptive fashion that is computationally inexpensive and valid everywhere. It is possible to perform a line search at each step to determine η^* [9]; however, this is quite expensive. We now describe a very simple adaptive overrelaxed bound optimization (ABO) algorithm that is guaranteed not to decrease the objective function at each iteration and requires only a very slight overhead in computation over regular BO(1) methods yet can often be many times faster.

```
Adaptive Overrelaxed Bound Optimization (ABO) for maximizing $L(\Theta)$:
• Assume: $\exists$ function $G(\Theta, \Psi)$ such that for
any given $\Theta'$ and $\Psi'$:
    1. $G(\Theta', \Theta') = L(\Theta')$ & $L(\Theta) > G(\Theta, \Psi') \; \forall \; \Theta \neq \Psi'$
    2. $\arg\max_\Theta G(\Theta, \Psi')$ can be found easily.
• Iterate starting with $\eta = 1$:
    1. $\Theta_{BO}^{t+1} = \arg\max_\Theta G(\Theta, \Theta^t)$
    2. $\Theta^{t+1} = \Theta^t + \eta(\Theta_{BO}^{t+1} - \Theta^t)$
    3. If $L(\Theta^{t+1}) > L(\Theta^t)$ Then Increase $\eta$
       Else $\Theta^{t+1} = \Theta_{BO}^{t+1}$ and Decrease $\eta$
• Guarantee: $L(\Theta^{t+1}) = G(\Theta^{t+1}, \Theta^{t+1}) \geq G(\Theta^{t+1}, \Theta^t) \geq G(\Theta^t, \Theta^t) = L(\Theta^t)$
```

Note that for many objective functions, computing $\arg\max_\Theta G(\Theta, \Theta^t)$ also evaluates the function $L(\Theta^t)$ "for free", so that step 3 above can be efficiently interleaved between steps 1 and 2 with essentially no extra computation (except when the optimizer oversteps).

5.3.1 Reparameterization of Constrained Quantities

The description of the adaptive algorithm above assumes that the parameters being optimized are unconstrained. In many cases, parameters must remain non-negative (or positive definite), sum to unity, respect symmetries or other parameter tying constraints. In these situations, the appropriate update rules can be derived by first reparameterizing the optimization using unconstrained variables which are related to the original variables through a fixed (possibly nonlinear) mapping. As examples, we develop several cases that arise often in practice.

If parameter values Θ_j must be positive (e.g. variances), the overrelaxation step can be derived from the reparameterization $\Theta_j = \exp(\beta_j)$ and results in:

$$\Theta_j^{t+1} = \Theta_j^t \left(\frac{\Theta_{jBO}^{t+1}}{\Theta_j^t} \right)^\eta \tag{5.14}$$

For parameter values vecΘ_j that represent a discrete distribution (e.g. mixing proportions in a mixture model, conditional probability tables for discrete quantities, or state transition probabilities in dynamic models), we reparameterize vecΘ_j via softmax function vec$\Theta_{ji} = \frac{\exp{(\beta_{ji})}}{\sum_{i=1} \exp{(\beta_{ji})}}$, and perform overrelaxation in the unconstrained vecβ space. In the constrained space this corresponds to the update:

$$\text{vec}\Theta_j^{t+1} = \frac{1}{Z} \text{vec}\Theta_j^t \left(\frac{\text{vec}\Theta_{jBO}^{t+1}}{\text{vec}\Theta_j^t} \right)^\eta \tag{5.15}$$

using elementwise multiplication and division operations and with Z being an appropriate normalizing constant.

5.3.2 Adaptive Expectation Maximization

We now consider a particular bound optimizer, the popular Expectation-Maximization (EM) algorithm and present its adaptive overrelaxed version. In chapter 4 we considered a probabilistic model of continuous observed data $\mathbf{x}$ which uses continuous latent variables $\mathbf{y}$. For any value of Ψ, we showed that that the following difference of two terms is a lower bound on the likelihood:

$$\begin{aligned} G(\Theta, \Psi) &= Q(\Theta, \Psi) - H(\Psi, \Psi) \\ &= \int p(\mathbf{y}|\mathbf{x}, \Psi) \ln p(\mathbf{x}, \mathbf{y}|\Theta) d\mathbf{y} - \int p(\mathbf{y}|\mathbf{x}, \Psi) \ln p(\mathbf{y}|\mathbf{x}, \Psi) d\mathbf{y} \end{aligned}$$

The log likelihood function can be written as:

$$\begin{aligned} L(\Theta) &= \ln p(\mathbf{x}|\Theta) = \textstyle\int p(\mathbf{y}|\mathbf{x}, \Theta) \ln p(\mathbf{x}|\Theta) d\mathbf{y} \\ &= G(\Theta, \Theta) \geq G(\Theta, \Psi) \quad \forall \Psi \end{aligned}$$

The EM algorithm is nothing more than coordinate ascent in the functional $G(\Theta, \Psi)$, alternating between maximizing G with respect to Ψ for fixed Θ (E-step) and with respect to Θ for fixed Ψ (M-step). Our new adaptive overrelaxed version of EM is given below:

Adaptive Overrelaxed EM (AEM) algorithm:

```
• $\eta$=1;  $L(\Theta^0)$ = -$\infty$;  $\delta$=tol;
• While ($\delta$ $\geq$tol  and  t<Tmax)
    – Perform E-step with $\Theta^t$ and get $L(\Theta^t)$
    – $\delta = (L(\Theta^t) - L(\Theta^{t-1}))$/abs($L(\Theta^t)$)
    – If $\delta$ <tol   /* We have gone too far */
        * $\eta = 1$;  Perform E-step with $\Theta^t_{EM}$
        * Get $L(\Theta^t_{EM})$  and compute new $\delta$;
        * /* Count this as an additional step */
      Else  $\eta = \alpha * \eta$; $\alpha > 1$   EndIf
    – Perform M-step to get $\Theta^{t+1}_{EM}$
    – $\Theta^{t+1} = \Theta^t$+$\eta$($\Theta^{t+1}_{EM} - \Theta^t$)
• EndWhile
```

In chapter 4, we established (refer to eq. 4.6):

$$\frac{\partial M(\Theta)}{\partial \Theta}|_{\Theta=\Theta^*} = \left[\frac{\partial^2 H(\Theta,\Theta^*)}{\partial \Theta^2}|_{\Theta=\Theta^*}\right]\left[\frac{\partial^2 Q(\Theta,\Theta^*)}{\partial \Theta^2}|_{\Theta=\Theta^*}\right]^{-1} \tag{5.16}$$

which can be interpreted as the ratio of missing information to the complete information near the local optimum. According to Lemma 4.2, the convergence rate matrix of EM(η) algorithm can be represented as follows: In the neighborhood of a solution (for sufficiently large t):

$$\Phi'(\Theta^t) = I - \eta\Bigg[I - \left(\frac{\partial^2 H}{\partial \Theta^2}\right)\left(\frac{\partial^2 Q}{\partial \Theta^2}\right)^{-1}|_{\Theta=\Theta^t}\Bigg] \tag{5.17}$$

This view of the EM(η) algorithm has a very interesting interpretation: *An increase in the proportion of missing information corresponds to higher values of the learning rate η.* If the fraction of missing information approaches unity, standard EM will be forced to take very small, conservative steps in parameter space, therefore higher and more aggressive values of η will result in much faster convergence. When the missing information is small compared to the complete information, the potential advantage of EM(η) over EM(1) becomes much less.

5.3.3 Adaptive Generalized Iterative Scaling Algorithm

In chapter 3 we studied the Generalized Iterative Scaling algorithm, that is widely used for parameter estimation in maximum entropy models [3]. We also derived a convex lower bound on $L(\Theta)$ for GIS algorithm (eq. 3.3):

$$\begin{aligned} L(\Theta) \quad \geq \quad & \sum_x \bar{p}(x) \sum_i \Theta_i f_i(x) - \ln Z(\Psi) + \sum_i \frac{f_i(x)}{s} - \\ & \sum_x p(x|\Psi) \sum_i \frac{f_i(x)}{s} \exp\left[s(\Theta_i - \Psi_i)\right] = G(\Theta, \Psi) \end{aligned} \tag{5.18}$$

We can now derive an adaptive overrelaxed version of GIS:

Adaptive Overrelaxed GIS (AGIS) algorithm:

```
• $\eta$=1;  $L(\Theta^0)$ = -$\infty$;  $\delta$=tol;
• While  ($\delta$ $\geq$tol  and  t<Tmax)
    – $\Theta^t_{i_{GIS}} = \Theta^{t-1}_i + \frac{1}{s} \ln \frac{\sum_x \bar{p}(x) f_i(x)}{\sum_x p(x|\Theta^t) f_i(x)}$
    – $\Theta^t = \Theta^{t-1}$+$\eta(\Theta^t_{GIS} - \Theta^{t-1})$  and get $L(\Theta^t)$
    – $\delta = (L(\Theta^t) - L(\Theta^{t-1}))$/abs$(L(\Theta^t))$
    – If $\delta$ <tol   /* We have gone too far */
        * $\eta = 1$;  $\Theta^t = \Theta^t_{GIS}$
        * Get $L(\Theta^t_{GIS})$  and compute new $\delta$;
        * /* Count this as an additional step */
      Else  $\eta = \alpha * \eta$; $\alpha > 1$   EndIf
• EndWhile
```

In chapter 3 we also derived the explicit form of the convergence rate matrix (eq. 3.7):

$$\frac{\partial M(\Theta)}{\partial \Theta}|_{\Theta=\Theta^*} = I - \frac{1}{s}\text{Cov}(\Theta^*)\text{D}(\Theta^*)^{-1} \tag{5.19}$$

which depends on the covariance and the mean of the feature vectors. According to Lemma 4.2, the convergence rate matrix of GIS(η) algorithm can be represented as follows: In the neighborhood of a solution (for sufficiently large t):

$$\Phi'(\Theta^t) = I - \eta\left[\frac{1}{s}\text{Cov}(\Theta^t)\text{D}(\Theta^t)^{-1}\right] \tag{5.20}$$

This view of the GIS(η) algorithm has a very interesting interpretation: *An increase in the feature dependence corresponds to higher values of the learning rate η.* If feature vectors are highly dependent, GIS will take very small conservative steps in the parameter space. Thus higher and more aggressive values of η will result in much faster convergence.

5.3.4 Adaptive Non-Negative Matrix Factorization

In chapter 3 we also introduced Non-Negative Matrix Factorization algorithm. Given non-negative matrix V, we are interested in finding matrices W and H, such that $V \approx WH$ [15].

Posed as an optimization problem, we are interested in minimizing a divergence $L(W, H) = D(V||WH)$, subject to $(W, H) \geq 0$ elementwise. Defining $\Theta = (W, H)$ and $\Psi = (W^t, H^t)$, we derived the upper bound on the cost function (eq. 3.11):

$$\begin{aligned} L(\Theta) \;\; \leq \;\; & \sum_{ij} V_{ij} \ln V_{ij} - V_{ij} + \sum_{ijc} W_{ic} H_{cj} - \\ & \sum_{ijc} V_{ij} \alpha_{ij}(c, c) \left[\ln \frac{W_{ic} H_{cj}}{\alpha_{ij}(c, c)} \right] = G(\Theta, \Psi) \end{aligned} \tag{5.21}$$

Adaptive overrelaxed NMF algorithm is then given as:

```
Adaptive Overrelaxed NMF (ANMF) algorithm:
  • $\eta$=1;  $L(\Theta^0)$ = -$\infty$;  $\delta$=tol;
  • While  ($\delta$ $\geq$tol  and  t<Tmax)
      – $W^t_{icNMF} = W^{t-1}_{ic} \left[ \frac{\sum_j H_{cj} V_{ic}/(WH)_{ic}}{\sum_v H_{cv}} \right]^\eta$
        $H^t_{cjNMF} = H^{t-1}_{cj} \left[ \frac{\sum_i W_{ic} V_{ic}/(WH)_{ic}}{\sum_w W_{wc}} \right]^\eta$
      – $\Theta^t = \Theta^{t-1}$+$\eta(\Theta^t_{NMF} - \Theta^{t-1})$;  and get $L(\Theta^t)$
      – $\delta = (L(\Theta^t) - L(\Theta^{t-1}))$/abs$(L(\Theta^t))$
      – If $\delta$ <tol    /* We have gone too far */
          ∗ $\eta = 1$;  Update $W^t$ and $H^t$;  get $L(\Theta^t_{NMF})$
          ∗ Compute new $\delta$;
          ∗ /* Count this as an additional step */
        Else  $\eta = \alpha * \eta$; $\alpha > 1$    EndIf
  • EndWhile
```

For many models overrelaxation is straightforward to implement and does not require significant computational overhead. As we will see in the next section, it can substantially outperform standard bound optimization algorithms.

5.4 Experimental Results

We now present empirical results comparing the performance of adaptive overrelaxed bound optimizers to standard BO(1) algorithms for learning the model parameters. We begin by show-

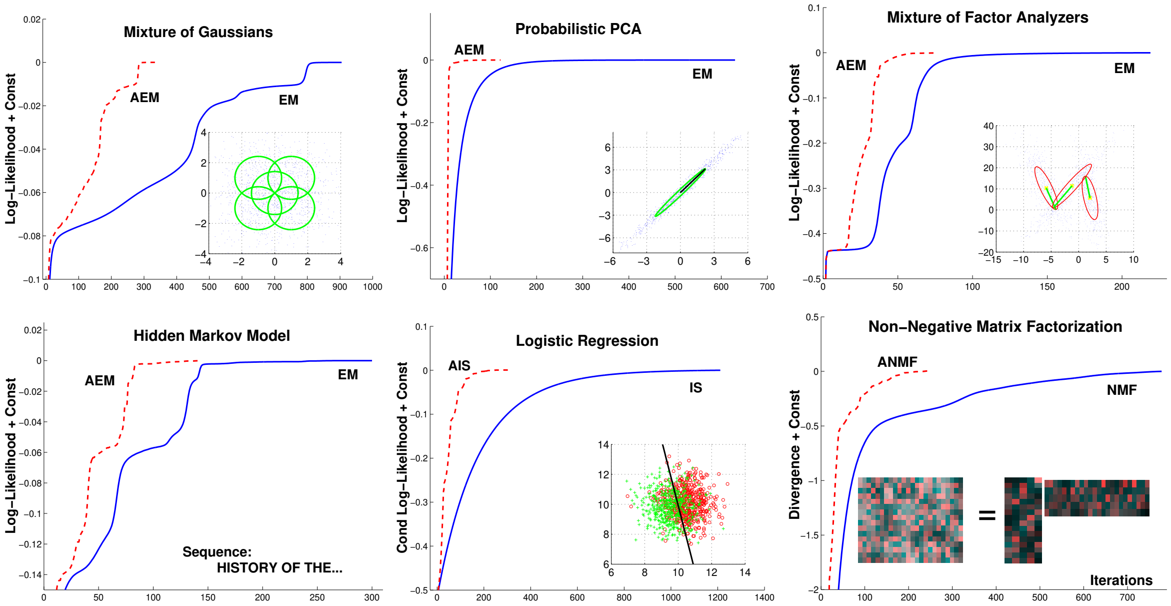

Figure 5.2: Learning curves for adaptive overrelaxed and standard bound optimization algorithms, showing convergence performance for different models. To present fair comparison, learning curves of adaptive bound optimizers include discarded steps (these curves contain flat regions). Upper panel displays MoG (left), PPCA (middle), MFA (right); and bottom panel displays HMM (left), logistic regression (middle), NMF (right). The iteration numbers are shown on the horizontal axis, and the value of the cost function is shown on the vertical axis, with zero-level corresponding to the converging point of the BO(1) algorithm.

ing convergence results on synthetic data sets, since it makes it easier to interpret, display and analyze. We then proceed to reporting similar empirical results on the real world data sets, supporting our presented analysis. Though not reported, we confirmed that the convergence results presented below do not vary significantly for different initial starting points in the parameter space. For all of the experiments reported below, we used tol = 10^{-8} and $\alpha = 1.1$.

5.4.1 Synthetic Data Sets

To compare AEM and EM algorithms, we considered several latent variable models: mixture of Gaussians (MoG), Hidden Markov Model (HMM), Probabilistic PCA (PPCA), and mixture of Factor Analyzers (MFA) models. As predicted by theory, high proportion of missing information in these models will result in slow convergence of EM, more aggressive learning rates η and thus superior performance of AEM.

First, consider a mixture of Gaussians (MoG) model. The data was generated from 5 Gaussian mixture components (see fig 5.2). In this model the proportion of missing information corresponds to how “well” or “not-well” data is separated into distinct clusters. Note that in the considered data set, mixture components overlap in one contiguous region, which constitutes the large proportion of missing information. Figure 5.2 shows that AEM outperforms standard EM algorithm by almost a factor of three.

We then applied our algorithm to the training of Hidden Markov Model. Missing information in this model is high when the observed data do not well determine the underlying state sequence (given the parameters). A simple 5-state fully-connected model was trained on 41 character sequences from the book ”Decline and Fall of the Roman Empire” by Gibbon, with an alphabet size of 30 characters (parameters were randomly initialized). We observe that even for the real, structured data AEM is superior to EM.

We also experimented with the Probabilistic Principal Component Analysis (PPCA) latent variable model[24, 28], which has continuous rather than discrete hidden variables. Here the concept of missing information is related to the ratios of the leading eigenvalues of the sample covariance, which corresponds to the ellipticity of the distribution. We observe that even for “nice” data, AEM outperforms EM by almost a factor of four. Similar results are displayed in figure 5.2 for the MFA [7] model.

As a confirmation to our analysis, in figure 5.3 we show the evolution of the adaptive learning rate η and the optimal learning rate η^* during fitting the means of the four mixture components in the MoG model, holding the mixing proportions and covariances fixed. The optimal learning rate was obtained by calculating λ_{min} and λ_{max} eigenvalues of $M'(\Theta^*)$ matrix and applying equation 5.12.

To compare IS and adaptive IS algorithms, we applied both methods to the simple 2-class logistic regression model: $p(y = \pm 1|x, w) = 1/(1 + \exp{(-yw^T x)})$ [20]. Feature vectors of dimensionality d were drawn from standard normal: $x \sim \mathcal{N}(0, I_d)$, with true parameter vector w^* being randomly chosen on surface of the d-dimensional sphere with radius $\sqrt{2}$. To make

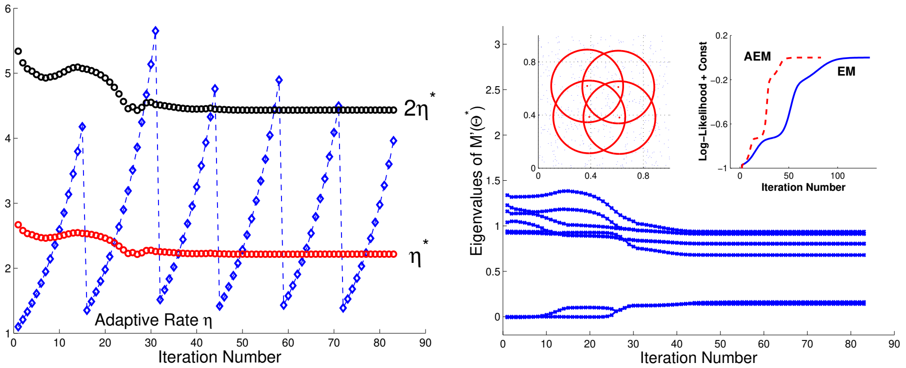

Figure 5.3: Left panel pictorially illustrates the adaptive learning rate η, the optimal rate η^*, and the approximate upper bound on the learning rate $2\eta^*$. The right panel shows the evolution of the eigenvalues of the convergence rate matrix $M'(\Theta^*)$ in eq.(5.1).

features positive, the data set was modified by adding 10 to all feature values. This in tern introduces significant correlation, and thus results in slow convergence of IS. To insure that $w^t x$ is unchanged, an extra feature was added. Figure 5.2 reveals that for d=2, AIS is superior to standard IS by at least a factor of three. Similar results are obtained if dimensionality of the data is increased.

At last, to compare ANMF and NMF, we randomly initialized the non-negative matrix $V_{16\times24}$, and applied both algorithms to perform non-negative factorization: $V_{16\times24} \approx W_{16\times5}H_{5\times24}$. Once again, results confirm the fact that overrelaxed methods can give speedup over conventional bound optimizers by several times.

5.4.2 Real World Data Sets

To compare AEM and EM, our first experiment consisted of training Aggregate Markov models AMM [26] on the ARPA North American Business News (NAB) corpus. AMMs are class-based bigram models in which the mapping from words to classes is probabilistic. The task of AMMs is to discover ”soft” word classes. The experiment used a vocabulary of sixty-thousand words, including tokens for punctuation, sentence boundaries, etc. The training data consisting of approximately 78 million words (three million sentences), with all sentences drawn without replacement from the NAB corpus. The number of classes was set C=2,4,6

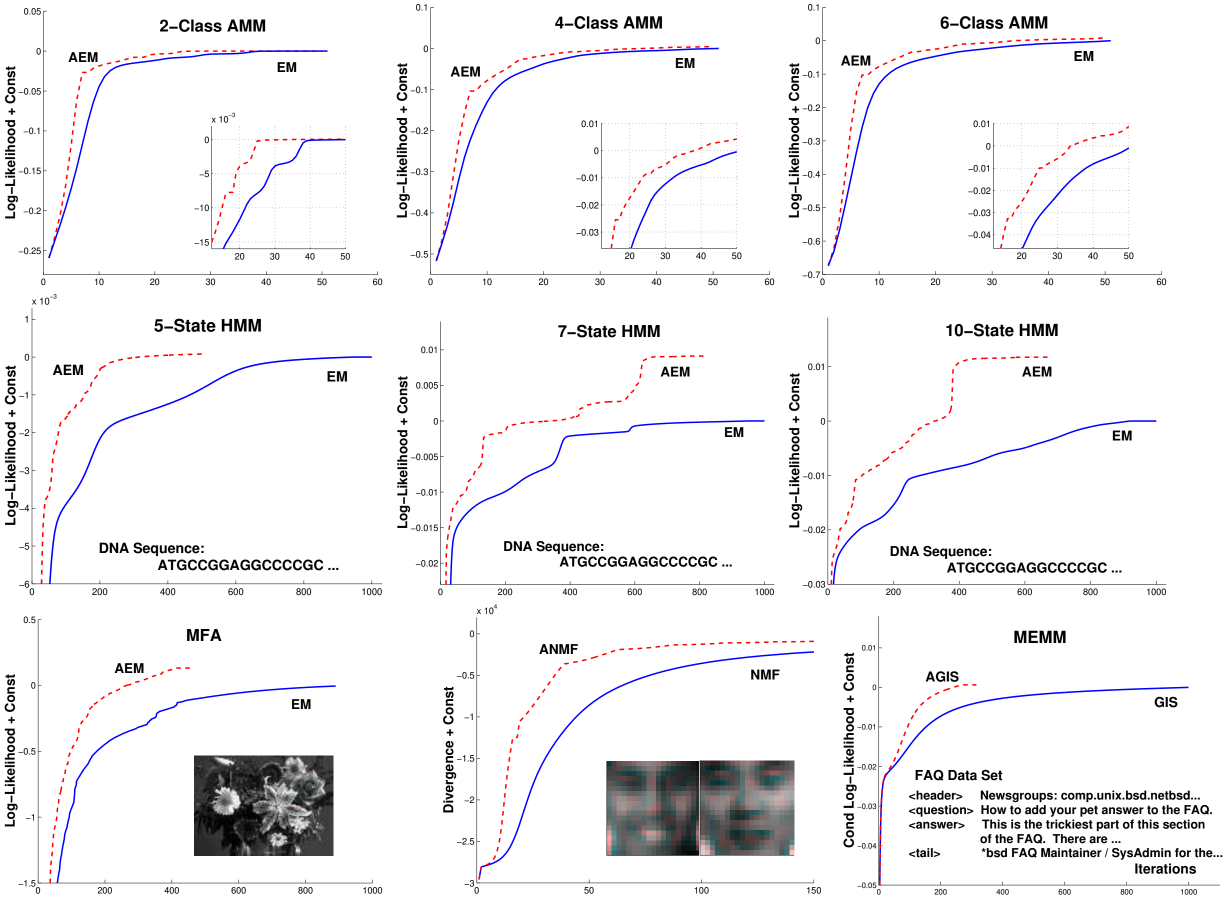

Figure 5.4: Learning curves for adaptive overrelaxed and standard bound optimization algorithms, showing convergence performance for different models. Upper panel displays AMM model with number of learned classes being: 2 (left), 4 (middle), and 6 (left). Middle panel shows HMM model with number of states being: 5 (left), 7 (middle), and 10 (right). Lower panel displays MFA (left), NMF (middle), and MEMM (right). The iteration numbers are shown on the horizontal axis, and the value of the cost function is shown on the vertical axis, with zero-level corresponding to the converging point of the BO(1) algorithm.

and all parameter values were randomly initialized.[5] Figure 5.4 (upper panels) reveals that AEM outperforms EM by at least a factor of 1.5. The considered data sets contains rather structured real data, suggesting relatively small fraction of the missing information. Nevertheless, to perform fair comparison, we have run both algorithms until the convergence criterion: $(L(\Theta^{t+1}) - L(\Theta^{t}))/\text{abs}(L(\Theta^{t+1})) \leq 10^{-8}$ is met. Setting the number of classes to 2, EM has converged in 164 iterations, whereas AEM has converged to exactly the same likelihood only

[5] For the details of the model and the data set, refer to [26].

after 72 iterations. This clearly constitutes a gain of a factor of over two.

Our second experiment consisted of training a fully connected HMM to model DNA sequences. For the training, we used publicly available ”GENIE gene finding data set”, provided by UCSC and LBNL [6], that contains 793 unrelated human genomic DNA sequences. We applied our AEM algorithm to 66 DNA sequences with length varying anywhere between 200 to 3000 multiple exon and single exon genes per sequence. The number of states were set to 5,7, and 10 and all the parameter values were randomly initialized. Figure 5.4 shows superior convergence of AEM over EM algorithm. In this case the considered data set contains very little overall structure, which constitutes high proportion of the missing information.

We have also applied the MFA model to the block transform image coding problem. A data set of 360×496 pixel images (see fig 5.4 bottom left panel) were subdivided into nonoverlapping blocks of size 8×8 pixels. Each block was regarded as a d=8×8 dimensional vector. The blocks (total of 2,790) were then compressed down to five dimensions using 10 mixture components.[6] Once again, AEM beats EM by a factor of over two, converging to the better likelihood.

To present the comparison between GIS and AGIS, we trained Maximum Entropy Markov Model [17] on the Frequently Asked Questions (FAQ) data set. The data set consisted of 38 files belonging to 7 Usenet groups. Each file contains header, followed by a series of one or more question/answer pairs, and ends with tail. The goal is to automatically label each line according to whether it is header, question, answer, or tail by using 24 boolean features of lines, like *begin-with-number, contains-http,* etc.[7] We observe that AGIS outperforms GIS by several times. We have also obtained analogous results training Conditional Random Fields [13].

In our last experiment, we trained NMF and adaptive NMF on the data set of facial images to learn part-based representation of faces [14]. The data set consisted of m=2,429 facial

[6] This experiment is similar to the one described in [29].

[7] See [17] for the description of the model and the data set.

images, each consisting of n=19×19 gray pixels, thus forming an $n \times m$ matrix V. In this experiment, the number of learned basis images were set to 49.[8] Once again, figure 5.4 reveals that ANMF substantially outperforms standard NMF algorithm. In particular, ANMF has converged in only about 3,500 iterations until the convergence criterion is met, whereas NMF converged in approximately 13,500 iterations to exactly the same value of the cost function, losing to ANMF by a factor of almost four.

5.5 Discussion

In this chapter we have analyzed the convergence properties of a large family of overrelaxed bound optimization BO(η) algorithms. Based on this analysis, we introduced a novel class of simple, adaptive overrelaxed bound optimization (ABO) methods and provided empirical results on several synthetic as well as real-world data sets showing superior convergence of the ABO methods over standard bound optimizers.

We have also experimented with models where parameter values Θ represent symmetric positive definite matrices (e.g. covariance matrices in the MoG model). Here we could use the fact that the map SPD$(n) \rightarrow \exp \mathrm{S}(n)$ is a bijection with SPD(n) and S(n) being $n \times n$ symmetric positive definite and symmetric matrices. We then can use the matrix exponential $\Theta = \exp \Lambda$ to perform overrelaxation in the unconstrained Λ space. In particular, we use a spectral decomposition: $\Theta = VDV^T$, with D being the diagonal matrix of the eigenvalues, and V being the orthogonal matrix of the corresponding eigenvectors. The matrix functions $\ln$ and $\exp$ are then defined: $\ln \Theta = V \ln{(D)} V^T$, and $\exp \Theta = V \exp{(D)} V^T$. When the matrix is diagonal, the overrelaxation corresponds to equation (5.14).

In all of our experiments with adaptive algorithms we found that the value of objective function at any iterate was better than the value at the same iterate of the standard bound optimizer: $L(\Theta^t_{ABO}) > L(\Theta^t_{BO}) : \forall t$. In other words, we have never found a disadvantage to

[8] See [14] for the detailed description of the experiment.

using adaptive methods; and often there is a large advantage, particularly for complex models with large training data sets, where due to the time constraints one could only afford to perform a few number of the BO iterations.

Chapter 6

Discussion and Future Work

In this thesis we have analyzed a large class of bound optimization algorithms and have studied an exciting question: What is the relationship between bound optimization algorithms and direct optimization algorithms such as gradient-based methods?

We have analyzed and determined conditions under which BO algorithms exhibit local-gradient and fast quasi-Newton convergence behaviors. Based on this analysis and interpretation, we considered several popular bound optimization algorithms and were able to provide some recommendations for how the input to these algorithms can be preprocessed to yield faster convergence.

We have also presented comparative analysis of Expectation-Maximization and direct optimization algorithms for learning parameters in latent variable models, and developed a theoretical connection between these two approaches. Motivated by these analyses, we have proposed an alternative hybrid optimization method that can significantly outperform EM in many cases and is almost never inferior in practice.

We have also studied and analyzed the convergence properties of a large family of over-relaxed bound optimization BO(η) algorithms, and proposed a novel, simple adaptive over-relaxed scheme for practical optimization. We showed that a novel class of simple, adaptive overrelaxed bound optimization (ABO) methods that possess superior performance (in certain cases by several orders of magnitude) to standard bound optimization algorithms.

Our results are also applicable to other bound optimization algorithms, for example Sha et. al. [27] recently introduced a multiplicative algorithm for training SVMs and provided a convergence analysis based on margins.

The overall analysis and interpretation of bound optimization algorithms, presented in the thesis, helped up to gain deeper understanding of the nature of these algorithms and conditions under which certain optimization techniques are expected to outperform others.

Many directions for future research in this area remain open. Currently, using our derivation of an explicit form of the convergence rate matrix, we are working on identifying analytic conditions under which CCCP possesses fast or extremely slow convergence in minimizing Bethe and Kikuchi free energies in approximate inference problems. We are also deriving a more compact representation and identifying a more intuitive interpretation of the convergence rate matrix for non-negative matrix factorization algorithm.

In chapter 4 we proposed to use the *entropy of the posterior over hidden variables* as a crude estimate of the local missing information ratio. Nevertheless, deriving an explicit mathematical connection between the entropy term and the fraction of missing information remains a challenge.

We are also working on identifying an intriguing relation between overrelaxed bound optimization algorithms and exponentiated gradient techniques [11] that have been very popular in the machine learning community.

Bibliography

[1] S.E. Atkinson. The performance of standard and hybrid EM algorithms for ML estimates of the normal mixture model with censoring. *J. of Stat. Computation and Simulation*, 44, 1992.

[2] Eric Bauer, Daphne Koller, and Yoram Singer. Update rules for parameter estimation in bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 3–13, August 1–3 1997.

[3] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the RS Society series B*, 39:1–38, 1977.

[5] Chris Fraley. On computing the largest fraction of missing information for the EM algorithm and the worst linear function for data augmentation. Technical report, University of Washington.

[6] GENIE gene data set. LBNL and UC Santa Cruz, `http://www.fruitfly.org/sequence`.

[7] Zoubin Ghahramani and Geoffrey Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, May 1996.

[8] D. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning*, pages 97–119, 1997.

[9] Mortaza Jamshidian and Robert I. Jennrich. Conjugate gradient acceleration of the EM algorithm. *Journal of the American Statistical Association*, 88(421):221–228, March 1993.

[10] Mortaza Jamshidian and Robert I. Jennrich. Acceleration of the EM algorithm by using quasi-newton methods. *J. of the RS Society series B*, 49, 1997.

[11] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 10 January 1997.

[12] Meng X. L. and van Dyk D. Fast EM-type implementations for mixed effects models. *J. of the Royal Statistical Society series B*, 60:559–578, 1998.

[13] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289, 2001.

[14] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Letters to Nature*, 401:788–791, 1999.

[15] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, 2001.

[16] T. A. Louis. Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society series B*, 44:226–233, 1982.

[17] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598, 2000.

[18] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.

[19] X. L. Meng and D. B. Rubin. On the global and componentwise rates of convergence of the EM algorithm. *Linear Algebra and Its Applications*, 199:413–425, 1994.

[20] Tom Minka. Algorithms for maximum-likelihood logistic regression. Technical Report 758, Dept. of Statistics, Carnegie Melon University, 2001.

[21] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 1993. ISBN: 0521431085.

[22] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *In Proceedings of IEEE*, 77(2):257–286, February 1989.

[23] Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, April 1984.

[24] S. T. Roweis. EM algorthms for PCA and SPCA. In *Advances in neural information processing systems*, volume 10, pages 626–632, Cambridge, MA, 1998. MIT Press.

[25] Sam Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2), 1999.

[26] Lawrence Saul and Fernando Pereira. Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 81–89. 1997.

[27] Fei Sha, Lawrence Saul, and Daniel Lee. Multiplicative updates for nonnegative quadratic programming in support vector machines. In *Advances in NIPS*, volume 15, 2003.

[28] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[29] Naonori Ueda, Ryohei Nakano, Zoubin Ghabramani, and Geoffrey E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[30] J. H. van Hateren and A. van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. In *Proceedings of the Royal Society of London*, pages 359–366, 1998.

[31] L. Xu and M. I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

[32] Alan Yuille and Anand Rangarajan. The convex-concave computational procedure (CCCP). In *Advances in NIPS*, volume 13, 2001.

Appendix A

Relationship between gradient and EM

In this appendix we present a close relationship between Expectation - Maximization algorithm and direct optimization approaches such as gradient-based methods for parameter learning. We show that the step EM takes in the parameter space and true gradient are related by the *symmetric positive definite* P matrix, and provide an explicit form of this matrix for several widely used latent variable models. We then go on deriving a general form of the P matrix for the regular exponential family in terms of its natural parameters.

A.1 Introduction

The problem of Maximum Livelihood (ML) learning is known to be an important problem in the area of machine learning and pattern recognition. ML learning is generally hard problem and arises in many probabilistic models with unobserved or latent variables such as density estimation, where one seeks to find a descriptive model of data, or dimensionality reduction, where one tries to discover a compact representation of data.

A variety of methods exist for ML learning of the model parameters in the presence of latent variables. A very popular technique for ML estimation is Expectation-Maximization (EM) algorithm. The EM algorithm alternates between estimating the unobserved variables given the current model and refitting the model given the estimated, complete data. As such it takes discrete steps in parameter space similar to to first order method operating on the gradient of a locally reshaped likelihood function. Direct optimization methods for the parameter learning can be viewed as alternative to the Expectation-Maximization. These algorithms work directly with the likelihood function and its derivatives (or estimates thereof), trying to maximize or minimize it by adjusting the free parameters in a local search. This category of algorithms includes random search, standard gradient-based algorithms, line search methods such as conjugate gradient, and more computationally intensive second-order methods, such as Newton-Raphson.

In this appendix we establish mathematical connection between Expectation-Maximization algorithm and direct optimization algorithms. In particular, we show that the step EM takes in the parameter space and true gradient are related by the *symmetric positive definite* matrix $P(\Theta)$, which is a function of the model parameters Θ. For a finite Gaussian mixture model this $P(\Theta)$ matrix was first described by Xu and Jordan[31]. We extend their results by deriving the explicit form of the symmetric positive definite matrix for several widely used latent variable models: Factor Analysis (FA), Probabilistic Principal Component Analysis (PPCA), mixture of FAs, mixture of PPCAs, and Hidden Markov Models (HMM). We then provide a general form of the $P(\Theta)$ matrix for the regular exponential family in terms of its natural parameters.

A.2 Connection between EM and gradient

A.2.1 Factor Analysis

Maximum likelihood Factor Analysis (FA) model seeks to specify probabilistically how a d-dimensional observed variable x is related to a p-dimensional latent variable z, where generally $p < d$. This can be viewed as a form of dimensionality reduction. The generative model is:

$$x = \Lambda z + \epsilon \tag{A.1}$$

with Λ being $d \times p$ factor loading matrix, $z \sim \mathcal{N}(0,1)$, and $\epsilon \sim \mathcal{N}(0,\Psi)$, where Ψ is diagonal matrix. In this model, the p-factors represent informative projections of the data, similar to the principal components in PCA.

The log-likelihood function for the FA model with parameters $\{\Lambda, \Psi\}$ is

$$L(\Theta) = -\frac{N}{2}\Big(d \ln 2\pi + \ln |C| + \text{tr}(C^{-1}S)\Big) \tag{A.2}$$

where C is the model covariance $C = \Lambda\Lambda^T + \Psi$, and S is a sample covariance matrix $S = \frac{1}{N}\sum_n (x_n - \mu)(x_n - \mu)^T$.

At each iteration of EM algorithm we have

$$\begin{aligned}
\text{vec}[\Lambda^{(t+1)}] - \text{vec}[\Lambda^{(t)}] &= P_\Lambda^{(t)} \frac{\partial L(\Theta)}{\partial \text{vec}[\Lambda]}|_{\Lambda=\Lambda^{(t)}} && \text{(A.3)}\\
\text{vec}[\Psi^{(t+1)}] - \text{vec}[\Psi^{(t)}] &= P_\Psi^{(t)} \frac{\partial L(\Theta)}{\partial \text{vec}[\Psi]}|_{\Psi=\Psi^{(t)}} && \text{(A.4)}
\end{aligned}$$

where vec(A) denotes the stacked columns of A, and

$$\begin{aligned}
P_\Lambda^{(t)} &= \Big(\sum_n E^{(t)}(x_n)\Big)^{-1} \otimes \Psi^{(t)}\\
P_\Psi^{(t)} &= \frac{2}{N}\text{diag}^*\Big[\big(\Lambda^{(t)}(\Lambda^{(t)})^T + \Psi^{(t)}\big) \otimes \Psi^{(t)}\Big]
\end{aligned}$$

where $E(x_n) \equiv I - \beta\Lambda + \beta(x_n - \mu)(x_n - \mu)^T\beta^T$ with $\beta \equiv \Lambda^T(\Lambda\Lambda^T + \Psi)$, diag*(A) sets all the rows of A to zero except for rows $j(d+1) - d$, $j = 1, 2, ..., d$, and "$\otimes$" denotes the Kronecker product.

Using the notation $\Theta = \left[\text{vec}[\Lambda]^T, \text{vec}[\Psi]^T\right]^T$, and $P(\Theta) = \text{diag}[P_\Lambda, P_\Psi]$ we can write

$$\Theta^{(t+1)} = \Theta^{(t)} + P(\Theta^t)\frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^{(t)}} \tag{A.5}$$

The validity of this symmetric positive definite matrix can be easily verified by multiplying it by the gradient of the log-likelihood function.

Restricting the covariance matrix Ψ to be spherical $\Psi = \sigma^2 I$, we arrive to so-called Probabilistic Principal Component Analysis (PPCA) [24, 28]. Here Λ spans p-dimensional principal subspace of the observed data. The P matrix for PPCA model can be easily derived in the similar way.

A.2.2 Mixture of Factor Analyzers

Mixture of Factor Analyzers (MFA) can be interpreted as a combination of two basic models: the standard mixture of Gaussians model together with Factor Analysis model.[1] As a result, this hybrid model simultaneously performs two tasks: clustering and local dimensionality reduction within each cluster [7].

The log-likelihood function for MFA model with parameters $\{\pi_i, \mu_i, \Lambda_i, \Psi_i\}_{i=1}^{M}$ is

$$L(\Theta) = \sum_n \ln \sum_{i=1}^{M} \pi_i \mathcal{N}(x_n|\mu_i, \Lambda_i\Lambda_i^T + \Psi_i) \tag{A.6}$$

with M denoting the number of clusters, and π_i, $i = 1, ..., M$ representing the mixing coefficients. At each iteration of EM algorithm we have

$$\begin{aligned}
\Pi^{(t+1)} - \Pi^{(t)} &= P_{\Pi}^{(t)} \frac{\partial L(\Theta)}{\partial \Pi}|_{\Pi=\Pi^{(t)}} && \text{(A.7)}\\
\mu_i^{(t+1)} - \mu_i^{(t)} &= P_{\mu_i}^{(t)} \frac{\partial L(\Theta)}{\partial \mu_i}|_{\mu_i=\mu_i^{(t)}} && \text{(A.8)}\\
\text{vec}[\Lambda_i^{(t+1)}] - \text{vec}[\Lambda_i^{(t)}] &= P_{\Lambda_i}^{(t)} \frac{\partial L(\Theta)}{\partial \text{vec}[\Lambda_i]}|_{\Lambda_i=\Lambda_i^{(t)}} && \text{(A.9)}\\
\text{vec}[\Psi_i^{(t+1)}] - \text{vec}[\Psi_i^{(t)}] &= P_{\Psi_i}^{(t)} \frac{\partial L(\Theta)}{\partial \text{vec}[\Psi_i]}|_{\Psi_i=\Psi_i^{(t)}} && \text{(A.10)}
\end{aligned}$$

where Π denotes mixing coefficients, $\Pi = [\pi_1, ..., \pi_M]^T$ and

$$\begin{aligned}
P_{\Pi}^{(t)} &= \frac{1}{N}\left[\text{diag}[\pi_1^{(t)}, ..., \pi_M^{(t)}] - \Pi^{(t)}(\Pi^{(t)})^T\right]\\
P_{\mu_i}^{(t)} &= \frac{\Psi_i^{(t)}}{\sum_n h_i^{(t)}(x_n)}\\
P_{\Lambda_i}^{(t)} &= \left(\sum_n h_i^{(t)}(x_n) E_i^{(t)}(x_n)\right)^{-1} \otimes \Psi_i^{(t)}\\
P_{\Psi_i}^{(t)} &= \frac{2}{\sum_n h_i^{(t)}(x_n)} \text{diag}^*\left[\left(\Lambda_i^{(t)}(\Lambda_i^{(t)})^T + \Psi_i^{(t)}\right) \otimes \Psi_i^{(t)}\right]
\end{aligned}$$

where $E_i(x_n) \equiv I - \beta_i\Lambda_i + \beta_i(x_n - \mu_i)(x_n - \mu_i)^T\beta_i^T$ with $\beta_i \equiv \Lambda_i^T(\Lambda_i\Lambda_i^T + \Psi_i)$, $h_i(x_n)$ are the responsibilities, diag*(A) sets all the rows of A to zero except for rows $j(d + 1) - d$, $j = 1, 2, ..., d$, where d is the dimensionality of data, and "$\otimes$" denotes the Kronecker product.

[1] In regular Mixture of Factor Analyzers model, the the isotropic noise covariance Ψ is fixed across all component densities. In our derivation we have different noise models across different component densities.

Using the notation $\Theta = \left[\Pi^T, \mu_1^T, ..., \mu_M^T, \text{vec}[\Lambda_1]^T, ..., \text{vec}[\Lambda_M]^T, \text{vec}[\Psi_1]^T, ..., \text{vec}[\Psi_M]^T\right]^T$, and $P(\Theta) = \text{diag}[P_\Pi, P_{\mu_1}, ..., P_{\mu_M}, P_{\Lambda_1}, ..., P_{\Lambda_M}, P_{\Psi_1}, ..., P_{\Psi_M}]$ we can write

$$\Theta^{(t+1)} = \Theta^{(t)} + P(\Theta^t)\frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^{(t)}} \tag{A.11}$$

One can easily verify the validity of this symmetric positive definite matrix by multiplying it by the gradient of the log-likelihood function.

The symmetric positive definite matrix for Mixture of Probabilistic Principal Component Analyzers model [28] can be easily derived in the analogous way.

A.2.3 Hidden Markov Model

Hidden Markov Model (HMM) can be interpreted as a dynamical mixture model, or a mixture model evolving over time [22].

The log-likelihood of observing the data under this model with parameters $\Theta = \{\pi, A, H\}$:

$$L(\Theta) = \log \sum_{s_1}\sum_{s_2} ... \sum_{s_T} \pi_{s_1} \prod_{t=1}^{T-1} a_{s_t, s_{t+1}} \prod_{t=1}^{T} h_{s_t, x_t} \tag{A.12}$$

where

- π_i is the probability of state s_i at time t=1.
- A is $M \times M$ matrix with its elements a_{ij} denoting the transition probability from state s_i to state s_j, and M denoting the number of states.
- H is $M \times R$ matrix with its elements h_{ij} denoting the probability of state s_i to generate observation x_j, and R denoting the alphabet size.

At each iteration of EM algorithm we have

$$\begin{aligned}
\Pi^{(t+1)} - \Pi^{(t)} &= P_\Pi^{(t)} \frac{\partial L(\Theta)}{\partial \Pi}|_{\Pi=\Pi^{(t)}} && \text{(A.13)}\\
\text{vec}[A^{(t+1)}] - \text{vec}[A^{(t)}] &= P_A^{(t)} \frac{\partial L(\Theta)}{\partial \text{vec}[A]}|_{A=A^{(t)}} && \text{(A.14)}\\
\text{vec}[H^{(t+1)}] - \text{vec}[H^{(t)}] &= P_H^{(t)} \frac{\partial L(\Theta)}{\partial \text{vec}[H]}|_{H=H^{(t)}} && \text{(A.15)}
\end{aligned}$$

where Π denotes initial state priors, $\Pi = [\pi_1, ..., \pi_M]^T$, and

$$\begin{aligned}
P_\Pi^{(t)} &= \text{diag}[\Pi^{(t)}] - \Pi^{(t)}(\Pi^{(t)})^T \\
P_A^{(t)} &= \text{diag}\big[\text{diag}(\text{vec}(E^{(t)}))\text{vec}(A^{(t)})\big] - \frac{1}{M}\text{vec}(A^{(t)})\text{vec}(A^{(t)})^T\text{diag}(\text{vec}(E^{(t)})) \\
P_H^{(t)} &= \text{diag}\big[\text{diag}(\text{vec}(F^{(t)}))\text{vec}(H^{(t)})\big] - \frac{1}{M}\text{vec}(H^{(t)})\text{vec}(H^{(t)})^T\text{diag}(\text{vec}(F^{(t)}))
\end{aligned}$$

where we have defined the following:

- $E^{(t)}$ is $M \times M$ matrix with elements $e_{ij} = 1/\sum_{t=1}^{T-1}\gamma_t(i)$, where $\gamma_t(i)$ denotes the probability of being in state s_i at time t. (Note that $e_{i1} = e_{i2} = ... = e_{iM}$.)

- $F^{(t)}$ is $M \times R$ matrix with elements $f_{ij} = 1/\sum_{t=1}^{T}\gamma_t(i)$.

Using the notation $\Theta = \big[\Pi^T, \text{vec}[A]^T, \text{vec}[H]^T\big]^T$, and $P(\Theta) = \text{diag}[P_\Pi, P_A, P_H]$ we can rewrite:

$$\Theta^{(t+1)} = \Theta^{(t)} + P(\Theta^t)\frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^{(t)}} \tag{A.16}$$

Once again, the reader can easily verify the validity of this symmetric positive definite matrix by multiplying it by the gradient of the log-likelihood function.

A.3 Exponential Family Models

Let us assume that the exponential family takes the following form:

$$p(x, z|\Theta) = f(x, z)\exp\{\Theta^T T(x, z)\}/g(\Theta) \tag{A.17}$$

where x are the observed variables, z are the latent variables, Θ is the vector of natural parameters and T is the vector of sufficient statistics. We are seeking the general form of the transformation matrix $P(\Theta)$, as a function of Θ:

$$\Theta^{t+1} - \Theta^t = P(\Theta^t)\frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^t} \tag{A.18}$$

where $L(\Theta)$ is the log-likelihood function, and $\Theta^{t+1} - \Theta^t$ represents the step EM performs in the parameter space. We also define the expected complete log-likelihood term as:

$$Q(\Theta|\Theta^t) = \int p(z|x, \Theta^t) \log p(x, z|\Theta) dz \tag{A.19}$$

For exponential family models we get from (A.17) that

$$\begin{aligned} \frac{\partial L(\Theta)}{\partial \Theta}|_{\Theta=\Theta^t} &= \frac{\partial Q(\Theta|\Theta^t)}{\partial \Theta}|_{\Theta=\Theta^t} \\ &= \int p(z|x, \Theta^t) T(x, z) dz - \int p(z, x|\Theta^t) T(x, z) dx\, dz \end{aligned} \tag{A.20}$$

which can be interpreted as the difference in the expected sufficient statistic vector when the observed data is clamped and unclamped. Define the following vector-valued functions:

$$\bar{T}(\Theta) = \int p(z, x|\Theta) T(x, z) dx\, dz \tag{A.21}$$

$$\bar{T}_z(\Theta) = \int p(z|x, \Theta) T(x, z) dz \tag{A.22}$$

The M step of the EM algorithm for the exponential family models then solves:

$$\frac{\partial Q(\Theta|\Theta^t)}{\partial \Theta}|_{\Theta^{t+1}} = \bar{T}_z(\Theta^t) - \bar{T}(\Theta^{t+1}) = 0 \tag{A.23}$$

Since $\bar{T}(\Theta)$ is an invertible function, we can write $\Theta^{t+1} = \bar{T}^{-1}(\bar{T}_z(\Theta^t))$. We now have all the ingredients to re-write (A.18) as:

$$\bar{T}^{-1}(\bar{T}_z(\Theta^t)) - \Theta^t = P(\Theta^t)\big[\bar{T}_z(\Theta^t) - \bar{T}(\Theta^t)\big] \tag{A.24}$$

One way, out of many, to write the general form of the transformation matrix $P(\Theta^t)$ that satisfies equation (A.24) is the following:

$$P(\Theta^t) = \frac{v(\Theta^t) v(\Theta^t)^T}{v(\Theta^t)^T u(\Theta^t)} \tag{A.25}$$

where $v(\Theta^t) = \bar{T}^{-1}(\bar{T}_z(\Theta^t)) - \Theta^t$, and $u(\Theta^t) = \bar{T}_z(\Theta^t) - \bar{T}(\Theta^t)$. Note that this transformation matrix $P(\Theta^t)$ is symmetric positive definite. Indeed, the denominator of (A.25) is written as:

$$u(\Theta^t)^T v(\Theta^t) = \frac{\partial Q(\Theta|\Theta^t)}{\partial \Theta}|_{\Theta^t} (\Theta^{t+1} - \Theta) > 0 \quad \Theta^{t+1} \neq \Theta^t \tag{A.26}$$

The above term can be regarded as a directional derivative of function $Q(\Theta|\Theta^t)$ in the direction of $\Theta^{t+1} - \Theta^t$. This quantity is always positive because the expected complete log-likelihood function $Q(\Theta|\Theta^t)$ for exponential family models is well-defined and concave, attaining its maximum at the point Θ^{t+1}.

A.4 Discussion

In this appendix we have built up the link between EM algorithm and gradient based methods for ML learning by showing that the EM step in the parameter space can be obtained from the gradient via the transformation symmetric positive definite P matrix. The important consequence of the above analysis is that EM has the appealing quality of always taking a step $\Theta^{(t+1)} - \Theta^t$ having positive projection onto the true gradient of the likelihood function $L(\Theta^t)$. This makes EM similar to the first order methods operating on the gradient of a locally reshaped likelihood function.

Appendix B

ECG for Several Latent Variable Models

B.1 Continuous Latent Variable Models

A latent variable model seeks to specify probabilistically how an observed variable x is related to a set of latent variables Z. This can be viewed as a form of dimensionality reduction or feature extraction. We will discuss a linear-Gaussian continuous hidden factor model, which is a simple example of a latent variable model

Consider the generative model, where latent variables cause the observed data.

$$x = \Lambda z + \mu + \epsilon, \qquad z \backsim \mathcal{N}(0, I), \epsilon \backsim \mathcal{N}(0, \Psi)$$

The marginal distribution of x can be computed analytically:

$$\begin{aligned} p(x|\Theta) &= \int_z p(x|z, \Theta)p(z)dz = \int_z \mathcal{N}(\Lambda z + \mu, \Psi)\mathcal{N}(0, I)dz = \\ & \frac{1}{(2\pi)^{d/2}|C|^{1/2}} \exp\Bigg(-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)\Bigg) \end{aligned} \tag{B.1}$$

The log-likelihood of observing data in these models is of the form:

$$L(\Theta) = -\frac{N}{2}\Bigg(d \ln 2\pi + \ln|C| + \text{tr}(C^{-1}S)\Bigg) \tag{B.2}$$

Note that C is the model covariance $C = \Lambda\Lambda^T + \Psi$, and S is a sample covariance matrix $S = \frac{1}{N}\sum_n (x_n - \mu)(x_n - \mu)^T$

B.1.1 Factor Analysis and PPCA

Restricting the covariance matrix Ψ of the observation noise to be diagonal, we obtain a standard statistical model known as Factor Analysis. Restricting the covariance matrix Ψ to be spherical $\Psi = \epsilon I$, we arrive to so-called probabilistic principal component analysis [25]

We can directly find the derivatives of $L(\Theta)$ with respect to its parameters $\Theta = \{\Lambda, \Psi\}$, without introducing an E-step.

Expectation-Conjugate-Gradient for Factor Analysis:

- E-Step: Analytical
- CG-Step:

$$\begin{aligned}\frac{\partial L(\Theta)}{\partial \Lambda} &= N(C^{-1}SC^{-1} - C^{-1})\Lambda \\ \frac{\partial L(\Theta)}{\partial \Gamma} &= \frac{N}{2}\text{diag}\Big(C^{-1}SC^{-1} - C^{-1}\Big)\Psi\end{aligned}$$

- Note that $\Psi_{ii} = \exp \Gamma_{ii}$

Expectation-Conjugate-Gradient for Probabilistic PCA:

- E-Step: Analytical
- CG-Step:

$$\begin{aligned}\frac{\partial L(\Theta)}{\partial \Lambda} &= N(C^{-1}SC^{-1} - C^{-1})\Lambda \\ \frac{\partial L(\Theta)}{\partial \eta} &= \frac{N}{2}\text{tr}\Big(C^{-1}SC^{-1} - C^{-1}\Big)\epsilon\end{aligned}$$

- Note that $\epsilon = \exp \eta$

B.2 Mixture Models

Mixture models can be regarded as generative models, where discrete hidden variable z causes the observed data

A mixture model can be regarded as a linear combination of M component densities. Let $z = 1, .., M$ be discrete latent variable that labels these component densities.

$$\begin{aligned}p(x|\Theta) &= \sum_{i=1}^{M} p(z=i)p(x|z=i,\Theta) \\ &= \sum_{i=1}^{M} \pi_i p(x|z=i,\Theta_i) \qquad \text{(B.3)}\end{aligned}$$

The log-likelihood function for mixture models takes the following form

$$L(\Theta) = \sum_n \ln \sum_{i=1}^{M} \pi_i p(x_n | z = i, \Theta_i) \tag{B.4}$$

B.2.1 Mixture of Gaussians

Mixture of Gaussians model (MoG) is perhaps the most common density estimation technique. In this model, mixing component densities are defined to be multivariate Gaussians with symmetric positive definite matrices Σ_i and mean vectors μ_i, $p_i(x) = \mathcal{N}(\mu_i, \Sigma_i)$.

The log-likelihood function of MoG model with parameters $\{\pi_i, \mu_i, \Sigma_i\}_{i=1}^{M}$ is

$$L(\Theta) = \sum_n \ln \sum_{i=1}^{M} \pi_i \mathcal{N}(\mu_i, \Sigma_i) \tag{B.5}$$

Expectation-Conjugate-Gradient for Mixture of Gaussians:

- E-Step: Compute responsibilities $h_i(x_n)$ for all $i = 1, ..., M$
- CG-Step:

$$\frac{\partial L(\Theta)}{\partial \gamma_i} = \sum_n h_i(x_n) - N\pi_i$$

$$\frac{\partial L(\Theta)}{\partial \mu_i} = \sum_n h_i(x_n) \Sigma_i^{-1} (x_n - \mu_i)$$

$$\frac{\partial L(\Theta)}{\partial A_i} = \left[N \Sigma_i^{-1} \bar{S}_i - \sum_n h_i(x_n) I \right] \Sigma_i^{-1} A_i$$

- Note that $\bar{S}_i$ is the responsibility-weighted local sample covariance matrix $\bar{S}_i = \frac{1}{N} \sum_n h_i(x_n)(x_n - \mu_i)(x_n - \mu_i)^T$, $\pi_i = \frac{\exp{(\gamma_i)}}{\sum_{j=1}^{M} \exp{(\gamma_j)}}$, and $\Sigma = AA^T$

B.2.2 Mixture of FAs

Mixture of Factor Analyzers (MFA) [7] can be interpreted as a combination of two basic models: the standard mixture of Gaussians model together with Factor Analysis model. As a result, this hybrid model simultaneously performs two tasks: clustering and local dimensionality reduction within each cluster.

The log-likelihood function for MFA model with parameters $\{\pi_i, \mu_i, \Lambda_i, \Psi\}_{i=1}^{M}$ is:

$$L(\Theta) = \sum_n \ln \sum_{i=1}^{M} \pi_i \mathcal{N}(\mu_i, \Lambda_i \Lambda_i^T + \Psi) \tag{B.6}$$

Expectation-Conjugate-Gradient for Mixture of FAs:

- E-Step: Compute responsibilities $h_i(x_n)$ for all $i = 1, ..., M$
- CG-Step:

$$\frac{\partial L(\Theta)}{\partial \gamma_i} = \sum_n h_i(x_n) - N\pi_i$$

$$\frac{\partial L(\Theta)}{\partial \mu_i} = C_i^{-1} \sum_n h_i(x_n)(x_n - \mu_i)$$

$$\frac{\partial L(\Theta)}{\partial \Lambda_i} = \Big[N C_i^{-1} \bar{S}_i - \sum_n h_i(x_n) I \Big] C_i^{-1} \Lambda_i$$

$$\frac{\partial L(\Theta)}{\partial \Gamma} = \frac{1}{2} \mathrm{diag} \Big(\sum_{i=1}^{M} \Big[N C_i^{-1} \bar{S}_i - \sum_n h_i(x_n) I \Big] C_i^{-1} \Big) \Psi$$

- Note that $\bar{S}_i$ is the responsibility-weighted local sample covariance matrix $\bar{S}_i = \frac{1}{N} \sum_n h_i(x_n)(x_n - \mu_i)(x_n - \mu_i)^T$, $\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^{M} \exp(\gamma_j)}$, and $\Psi = \exp \Gamma$

B.2.3 Mixture of PPCA

Mixture of Probabilistic Principal Component Analyzers (MPPCA) [28], as in case of MFA, can be viewed as a basic clustering method together with local dimensionality reduction method.

The log-likelihood function for MPPCA model with: model parameters $\{\pi_i, \mu_i, \Lambda_i, \epsilon\}_{i=1}^{M}$ is written as:

$$L(\Theta) = \sum_n \ln \sum_{i=1}^{M} \pi_i \mathcal{N}(\mu_i, \Lambda_i \Lambda_i^T + \epsilon I) \tag{B.7}$$

Expectation-Conjugate-Gradient for Mixture of Probabilistic PCAs:

- E-Step: Compute responsibilities $h_i(x_n)$ for all $i = 1, ..., M$
- CG-Step:

$$\frac{\partial L(\Theta)}{\partial \gamma_i} = \sum_n h_i(x_n) - N\pi_i$$

$$\frac{\partial L(\Theta)}{\partial \mu_i} = C_i^{-1} \sum_n h_i(x_n)(x_n - \mu_i)$$

$$\frac{\partial L(\Theta)}{\partial \Lambda_i} = \left[NC_i^{-1}\bar{S}_i - \sum_n h_i(x_n) I \right] C_i^{-1}\Lambda_i$$

$$\frac{\partial L(\Theta)}{\partial \eta} = \frac{1}{2}\mathrm{tr}\left(\sum_{i=1}^{M} \left[NC_i^{-1}\bar{S}_i - \sum_n h_i(x_n) I \right] C_i^{-1} \right)\epsilon$$

- Note that $\bar{S}_i$ is the responsibility-weighted local sample covariance matrix $\bar{S}_i = \frac{1}{N}\sum_n h_i(x_n)(x_n - \mu_i)(x_n - \mu_i)^T$, $\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^{M}\exp(\gamma_j)}$, and $\epsilon = \exp\eta$

B.2.4 Hidden Markov Model

Hidden Markov Model can be interpreted as a dynamical mixture model, or a mixture model evolving over time [22]. The log-likelihood of observing the data under this model with parameters $\Theta = \{\pi, A, H\}$ is:

$$L(\Theta) = \log \sum_{s_1}\sum_{s_2} ... \sum_{s_T} \pi_{s_1} \prod_{t=1}^{T-1} a_{s_t,s_{t+1}} \prod_{t=1}^{T} h_{s_t,x_t} \tag{B.8}$$

where we define:

- π_i is the probability of state s_i at time t=1.
- A : a_{ij} is the transition probability from state s_i to state s_j.
- H : h_{ij} is the probability of state s_i to generate observation x_j.

Expectation-Conjugate-Gradient for Hidden Markov Models:

- E-Step: Compute expectation of sufficient statistics:

  $E[\delta_{s_t,i}\delta_{s_{t+1},j}],\ \ E[\delta_{s_t,i}\delta_{y_t,j}],\ \ E[\delta_{s_t,i}]$ for t=1,...,T

- CG-Step:

$$\frac{\partial L(\Theta)}{\partial \gamma_i^{(\pi)}} = E[\delta_{s_1,i}] - \pi_i$$

$$\frac{\partial L(\Theta)}{\partial \gamma_{ij}^{(a)}} = \sum_{t=1}^{T-1}\Big[E[\delta_{s_t,i}\delta_{s_{t+1},j}] - E[\delta_{s_t,i}]a_{ij}\Big]$$

$$\frac{\partial L(\Theta)}{\partial \gamma_{ij}^{(h)}} = \sum_{t=1}^{T}\Big[E[\delta_{s_t,i}\delta_{y_t,j}] - E[\delta_{s_t,i}]h_{ij}\Big]$$

- Note that $\pi_i = \frac{\exp\gamma_i^\pi}{\sum_{i'=1}^M \exp\gamma_{i'}^\pi}$, $a_{ij} = \frac{\exp\gamma_{ij}^a}{\sum_{j'=1}^M \exp\gamma_{ij'}^a}$, $h_{ij} = \frac{\exp\gamma_{ij}^h}{\sum_{j'=1}^M \exp\gamma_{ij'}^h}$

B.3 Unconstrained Optimization

In all of the considered models, we need to satisfy regularity constraints on the model parameters. To conduct unconstrained optimization, we use simple reparametarization of the discrete model parameters

- In all mixture models we use softmax parametarization of the mixing coefficients:

$$\pi_i = \frac{\exp(\gamma_i)}{\sum_{j=1}^M \exp(\gamma_j)} \tag{B.9}$$

- For covariance matrix Σ to be symmetric positive definite in MoG model, we use Choleski decomposition:

$$\Sigma = AA^T \tag{B.10}$$

- To keep diagonal entries of the noise model positive, we use:

$$\Psi_{ii} = \exp\Gamma_{ii} \quad \text{for Mixture of FAs} \tag{B.11}$$

$$\epsilon = \exp\eta \quad \text{for Mixture of PPCAs} \tag{B.12}$$

- In HMM model, we reparametarize probabilities via softmax function:

$$\pi_i = \frac{\exp\gamma_i^\pi}{\sum_{i'=1}^M \exp\gamma_{i'}^\pi}, \quad a_{ij} = \frac{\exp\gamma_{ij}^a}{\sum_{j'=1}^M \exp\gamma_{ij'}^a}, \quad h_{ij} = \frac{\exp\gamma_{ij}^h}{\sum_{j'=1}^M \exp\gamma_{ij'}^h} \tag{B.13}$$

To remove translational degeneracy, we force $\gamma_1 = 0$ and remove it from the parameter Θ.