Missing Data Problems in Machine Learning

by

Benjamin M. Marlin

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Missing Data Problems in Machine Learning

Benjamin M. Marlin
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2008

Learning, inference, and prediction in the presence of missing data are pervasive problems in machine learning and statistical data analysis. This thesis focuses on the problems of collaborative prediction with non-random missing data and classification with missing features. We begin by presenting and elaborating on the theory of missing data due to Little and Rubin. We place a particular emphasis on the missing at random assumption in the multivariate setting with arbitrary patterns of missing data. We derive inference and prediction methods in the presence of random missing data for a variety of probabilistic models including finite mixture models, Dirichlet process mixture models, and factor analysis.

Based on this foundation, we develop several novel models and inference procedures for both the collaborative prediction problem and the problem of classification with missing features. We develop models and methods for collaborative prediction with non-random missing data by combining standard models for complete data with models of the missing data process. Using a novel recommender system data set and experimental protocol, we show that each proposed method achieves a substantial increase in rating prediction performance compared to models that assume missing ratings are missing at random.

We describe several strategies for classification with missing features including the use of generative classifiers, and the combination of standard discriminative classifiers with single imputation, multiple imputation, classification in subspaces, and an approach based on modifying the classifier input representation to include response indicators. Results on real and synthetic data sets show that in some cases performance gains over baseline methods can be achieved by methods that do not learn a detailed model of the feature space.

Acknowledgements

I've been privileged to enjoy the support and encouragement of many people during the course of this work. I'll start by thanking my thesis supervisor, Rich Zemel. I've learned a great deal of machine learning from Rich, and have benefitted from his skill and intuition at modelling difficult problems. I'd also like to thank Sam Roweis, who essentially co-supervised much of my PhD research. His enthusiasm for machine learning is insatiable, and his support of this work has been greatly appreciated.

I have benefitted from the advice of a terrific PhD committee including Geoff Hinton and Brendan Frey, as well as Rich and Sam. Rich, Sam, Geoff, and Brendan were all instrumental in helping me to pare down a long list of interesting problems to arrive at the present contents of this thesis. I've appreciated their helpful comments and thoughtful questions throughout the research and thesis writing process. I would like to extend a special thanks to my external examiner, Zoubin Ghahramani, for his thorough reading of this thesis. His detailed comments, questions, and suggestions have helped to significantly improve this thesis.

During the course of this work I have also been very fortunate to collaborate with Malcolm Slaney at Yahoo! Research. I'm very grateful to Malcolm for championing our projects within Yahoo!, and to many other people at Yahoo! who were involved in our work including Sandra Barnat, Todd Beaupre, Josh Deinsen, Eric Gottschalk, Matt Fukuda, Kristen Jower-Ho, Brian McGuiness, Mike Mull, Peter Shafton, Zack Steinkamp, and David Tseng. I would like to thank Dennis DeCoste, who co-supervised me at Yahoo! for a short time, for his continuing interest in this work. Malcolm also helped to coordinate the release of the Yahoo! data set used in this thesis. Malcolm, Rich, and I would like to extend our thanks to Ron Brachman, David Pennock, John Langford, and Lauren McDonnell at Yahoo!, as well as Fred Zhu from the University's Office of Intellectual Property for their efforts in approving the data release and putting together a data use agreement.

I would like to acknowledge the generous funding of this work provided by the University of Toronto Fellowships program, the Ontario Graduate Scholarships program, and the Natural Sciences and Engineering Research Council Canada Graduate Scholarships program. This work wouldn't have been possible without the support of these programs.

On the personal side, I'd like to thank all my lab mates and friends at the University for good company and interesting discussions over the years including Matt Beal, Miguel Carreira-Perpinan, Stephen Fung, Inmar Givoni, Jenn Listgarten, Ted Meeds, Roland Memisevic, Andriy Mnih, Quaid Morris, Rama Natarajan, David Ross, Horst Samulowitz, Rus Salakhutdinov, Nati Srebro, Liam Stewart, Danny Tarlow, and Max Welling. I'm very grateful to Bruce and Maura Rowat, for providing me with a home away from home during my final semester of courses in Toronto. I'm also grateful to Horst Samulowitz, Nati Srebro and Eli Thomas, Sam Roweis, and

Ted Meeds for the use of spare rooms/floor space on numerous visits to the University.

I'd like to thank my Mom for never giving up on trying to understand exactly what this thesis is all about, and my Dad for teaching me that you can fix anything with hard work and the right tools. I'd like to thank the whole family for providing a great deal of support, and for their enthusiasm at the prospect of me finishing the 22^{nd} grade. Finally, I'm incredibly grateful to my wife Krisztina for reminding me to eat and sleep when things were on a roll, for love and encouragement when things weren't going well, for always being ready to drop everything and get away from it all when I needed a break, and for understanding all the late nights and weekends that went into finishing this thesis.

Contents

Chapter 1

Introduction

Missing data occur in a wide array of application domains for a variety of reasons. A sensor in a remote sensor network may be damaged and cease to transmit data. Certain regions of a gene microarray may fail to yield measurements of the underlying gene expressions due to scratches, finger prints, dust, or manufacturing defects. Participants in a clinical study may drop out during the course of the study leading to missing observations at subsequent time points. A doctor may not order all applicable tests while diagnosing a patient. Users of a recommender system rate an extremely small fraction of the available books, movies, or songs, leading to massive amounts of missing data.

Abstractly, we may consider a random process underlying the generation of incomplete data sets. This generative process can be decomposed into a complete data process that generates complete data sets, and a missing data process that determines which elements of the complete data set will be missing. In the examples given above, the hypothetical complete data set would include measurements from every sensor in a remote sensor network, the result of every medical test relevant to a particular medical condition for every patient, and the rating of every user for every item in a recommender system. The missing data process is sometimes referred to as the missing data mechanism, the observation process, or the selection process. We might imagine that a remote sensor is less likely to transmit data if its operational temperate range is exceeded, that a doctor is less likely to order a test that is invasive, and that a user of a recommender system is less likely to rate a given item if the user does not like that item.

The analysis of missing data processes leads to a theory of missing data in terms of its impact on learning, inference, and prediction. This theory draws a distinction between two fundamental categories of missing data: data that is missing at random and data that is not missing at random. When data is missing at random, the missing data process can be ignored and inference can be based on the observed data only. The resulting computations are tractable in many common generative models. When data is not missing at random, ignoring the missing

data process leads to a systematic bias in standard algorithms for unsupervised learning, inference, and prediction. An intuitive example of a process that violates the missing at random assumption is one where the probability of observing the value of a particular feature depends on the value of that feature. All forms of missing data are problematic in the classification setting since standard discriminative classifiers do not include a model of the feature space. As a result, most discriminative classifiers have no natural ability to deal with missing data.

1.1 Outline and Contributions

The focus of this thesis is the development of models and algorithms for learning, inference, and prediction in the presence of missing data. The two main problems we study are collaborative prediction with non-random missing data, and classification with missing features. We begin Chapter 2 with a discussion of decision theory as a framework for understanding different learning and inference paradigms including Bayesian inference, maximum a posteriori estimation, maximum likelihood estimation, and regularized function approximation. We review particular algorithms and principles including the Metropolis-Hastings algorithm, the Gibbs sampler, and the Expectation Maximization algorithm. We also discuss procedures for estimating the performance of prediction methods.

Chapter 3 introduces the theory of missing data due to Little and Rubin. We present formal definitions of the three main classes of missing data. We present a detailed investigation of the missing at random assumption in the multivariate case with arbitrary patterns of missing data. We argue that the missing at random assumption is best understood in terms of a set of symmetries imposed on the missing data process. We review the impact of random and non-random missing data on probabilistic inference. We present a study of the effect of data model misspecification on inference in the presence of random missing data. We demonstrate that using an incorrect data model can lead to biased inference and learning even when data is missing at random in the underlying generative process.

Chapter 4 introduces unsupervised learning models in the random missing data setting including finite multinomial mixtures, Dirichlet Process multinomial mixtures, factor analysis, and probabilistic principal component analysis. We present maximum a posteriori learning in finite mixture models with missing data. We derive conjugate and collapsed Gibbs samplers for the Dirichlet Process multinomial mixture model with missing data. We derive complete expectation maximization algorithms for factor analysis, probabilistic principal components analysis, mixtures of factor analyzers, and mixtures of probabilistic principal components analyzers with missing data.

Chapter 5 focuses on the problem of unsupervised learning for collaborative prediction when

missing data may violate the missing at random assumption. Collaborative prediction problems like rating prediction in recommender systems are typically solved using unsupervised learning methods. As discussed in Chapter 3, the results of learning and prediction will be biased if the missing at random assumption is violated. We discuss compelling new evidence in the form of a novel user study and the analysis of a new collaborative filtering data set which strongly suggests that the missing at random assumption does not hold in the recommender system domain.

We present four novel models for unsupervised learning with non-random missing data that build on the models and inference procedures for random missing data presented in Chapter 4. These models include the combination of the finite multinomial mixture model and the Dirichlet Process multinomial mixture model with a simple missing data mechanism where the probability that a rating is missing depends only on the value of that rating. We refer to this mechanism as CPT-v since it is parameterized using a simple conditional probability table. We prove that the parameters of the CPT-v missing data mechanism are conditionally identifiable even though the mixture data models are not identifiable. We also combine the finite multinomial mixture model with a more flexible missing data model that we refer to as Logit-vd. The Logit-vd model allows for response probabilities that differ depending on both the underlying rating value, and the identity of the item. The name Logit-vd derives from the fact that the missing data mechanism is represented using an additive logistic model. We review modified contrastive divergence learning for restricted Boltzmann machines with missing data, and offer a new derivation of these learning methods as standard contrastive divergence in an alternative model. The final model we consider is a conditional Restricted Boltzmann Machine that includes energy terms that can account for non-random missing data effects similar to the CPT-v model.

We show that traditional experimental protocols and testing procedures for collaborative prediction implicity assume missing ratings are missing at random. We show that these procedures fail to detect the effects of non-random missing ratings. To correct this problem we introduce novel experimental protocols and testing procedures specifically designed for collaborative prediction with non-random missing data. Our empirical results show that rating prediction methods based on models that incorporate an explicit non-random missing data mechanism achieve 25% to 40% lower error rates than methods that assume the missing at random assumption holds. To put these results in perspective, the best models studied in our previous work on collaborative filtering achieve approximately 15% lower error rates relative to the simplest models we considered [52, p. 107-108]. We also compare the methods studied in terms of ranking performance, and again show that methods that model the missing data mechanism achieve better ranking performance than methods that treat missing data as if it is

missing at random.

In Chapter 6 we consider the problem of classification with missing features. We begin with a discussion of general strategies for dealing with missing data in the classification setting. We consider the application of generative classifiers where missing data can be analytically integrated out of the model. We derive a variation of Fisher's linear discriminant analysis for missing data that uses a factor analysis model for the covariance matrix. We then derive a novel discriminative learning procedure for the classifier based on maximizing the conditional probability of the labels given the observed data.

We study the application of logistic regression, multi-layer neural networks, and kernel classifiers in conjunction with several frameworks for converting a discriminative classifier into a classifier for incomplete data cases. We consider the use of various imputation methods including multiple imputation. For data sets with a limited number of patterns of missing data, we consider a reduced model approach that learns a separate classifier for each pattern of missing data. Finally, we consider an approach based on modifying the input representation of a discriminative classifier in such a way that the classification function depends only on the observed feature values, and which features are observed. Results on real and synthetic data sets show that in some cases performance gains over baseline methods can be achieved without learning detailed models of the input space.

1.2 Notation

We use capital letters to denote random variables, and lowercase letters to denote instantiations of random variables. We use a bold typeface to indicate vector and matrix quantities, and a plain typeface to indicate scalar quantities.

When describing data sets we denote the total number of feature dimensions by D, and the total number of data cases by N. We denote the feature vector for data case n by $\mathbf{x}_n$, and individual feature values by x_{dn}. In the classification setting we denote the total number of classes by C. We denote the class variable for data case n by y_n, and assume it takes the values $\{1, -1\}$ in the binary case, and $\{1, ..., C\}$ in the multi-class case.

We use square bracket notation $[s]$ to represent an indicator function that takes the value 1 if the statement s is true, and 0 if the statement s is false. For example, $[x_{dn} = v]$ would take the value 1 if x_{dn} is equal to v, and 0 otherwise.

1.2.1 Notation for Missing Data

Following the standard representation for missing data due to Little and Rubin [49], we introduce a companion vector of response indicators for data case n denoted $\mathbf{r}_n$. r_{dn} is 1 if x_{dn} is

observed, and r_{dn} is 0 if x_{dn} is not observed. We denote the number of observed data dimensions in data case n by D_n. In addition to the response indicator vector, we introduce a vector $\mathbf{o}_n$ of length D_n listing the dimensions of x_n that are observed. We define $o_{in} = d$ if $\sum_{j=1}^{d} r_{jn} = i$ and $r_{dn} = 1$. In other words, $o_{in} = d$ if d is the i^{th} observed dimension of $\mathbf{x}_n$. We introduce a corresponding vector $\mathbf{m}_n$ of length $D - D_n$ listing the dimensions of $\mathbf{x}_n$ that are missing. We define $m_{in} = d$ if $\sum_{j=1}^{d}(1 - r_{jn}) = i$ and $(1 - r_{dn}) = 1$. In other words, $m_{in} = d$ if d is the i^{th} missing dimension of x_n.

We use superscripts to denote sub-vectors and sub-matrices. For example, $\mathbf{x}_n^{\mathbf{o}_n}$ denotes the sub-vector of $\mathbf{x}_n$ corresponding to the observed elements of $\mathbf{x}_n$. The element-wise definition of $\mathbf{x}_n^{\mathbf{o}_n}$ is $x_{in}^{\mathbf{o}_n} = x_{o_{in}n}$. Similarly, if $\mathbf{\Sigma}$ is a $D \times D$ matrix then, for example, $\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{m}_n}$ is the sub-matrix of $\mathbf{\Sigma}$ obtained by selecting the rows corresponding to the observed dimensions of $\mathbf{x}_n$, and the columns corresponding to the missing dimensions of $\mathbf{x}_n$. The element-wise definition of $\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{m}_n}$ is $\Sigma_{ij}^{\mathbf{o}_n\mathbf{m}_n} = \Sigma_{o_{in}m_{jn}}$. For simplicity we will often use the notation $\mathbf{x}^o$ and $\mathbf{\Sigma}^{om}$ in place of $\mathbf{x}_n^{\mathbf{o}_n}$ or $\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{m}_n}$ when it is clear which pattern of observed or missing entries is intended.

Projection matrices are another very useful tool for dealing with sub-vectors and sub-matrices induced by missing data. We define the projection matrix $\mathbf{H}_n^o$ where $\mathbf{H}_{ijn}^o = [\mathbf{o}_{jn} = i]$. The matrix $\mathbf{H}_n^o$ projects a vector from the D_n dimensional space corresponding to the observed dimensions of $\mathbf{x}_n$ to the full D dimensional feature space. The missing dimensions are filled with zeros. Similarly, we define the projection matrix $\mathbf{H}_n^m$ such that $\mathbf{H}_{ijn}^m = [\mathbf{m}_{jn} = i]$. The matrix $\mathbf{H}_n^m$ projects a vector from the $(D - D_n)$ dimensional space corresponding to the missing dimensions of $\mathbf{x}_n$ to the full D dimensional feature space. The observed dimensions are filled with zeros. As we will see later, these projection matrices arise naturally when taking matrix and vector derivatives of the form $\partial\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{m}_n}/\partial\mathbf{\Sigma}$.

1.2.2 Notation and Conventions for Vector and Matrix Calculus

Throughout this work we will be deriving optimization algorithms that require the closed-form or iterative solution of a set of gradient equations. The gradient equations are derived using matrix calculus. In this section we review the matrix calculus conventions used in this work.

First, we assume that all vectors are column vectors unless explicitly stated otherwise. We will follow the convention that the gradient of a scalar function f with respect to a matrix-valued function $\mathbf{g}$ of dimension $A \times B$ is a matrix of size $A \times B$ as seen in Equation 1.2.1. We adopt this convention since it avoids the need to transpose the matrix of partial derivatives when solving gradient equations, and performing iterative gradient updates.

$$\frac{\partial f}{\partial \mathbf{g}} = \begin{bmatrix} \frac{\partial f}{\partial g_{11}} & \frac{\partial f}{\partial g_{12}} & \cdots & \frac{\partial f}{\partial g_{1B}} \\ \frac{\partial f}{\partial g_{21}} & \frac{\partial f}{\partial g_{22}} & \cdots & \frac{\partial f}{\partial g_{2B}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial g_{A1}} & \frac{\partial f}{\partial g_{A2}} & \cdots & \frac{\partial f}{\partial g_{AB}} \end{bmatrix} \tag{1.2.1}$$

We will follow the convention that the matrix of partial derivatives of a vector-valued function $\mathbf{f}$ of A dimensions with respect to a vector-valued function $\mathbf{g}$ of B dimensions has size $B \times A$ as seen in Equation 1.2.2. In the case where $\mathbf{f}$ and $\mathbf{g}$ are both multi-dimensional functions, we adopt the convention of expressing the matrix of partial derivatives element-wise.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{g}} = \begin{bmatrix} \frac{\partial f_1}{\partial g_1} & \frac{\partial f_2}{\partial g_1} & \cdots & \frac{\partial f_A}{\partial g_1} \\ \frac{\partial f_1}{\partial g_2} & \frac{\partial f_2}{\partial g_2} & \cdots & \frac{\partial f_A}{\partial g_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial g_B} & \frac{\partial f_2}{\partial g_B} & \cdots & \frac{\partial f_A}{\partial g_B} \end{bmatrix} \tag{1.2.2}$$

Let $\mathbf{f}$ be an A dimensional vector-valued function, $\mathbf{g}$ be a B dimensional vector-valued function, and $\mathbf{h}$ be a C dimensional vector-valued function. Assuming that $\mathbf{f}$ is a function of $\mathbf{g}$ and $\mathbf{g}$ is a function of $\mathbf{h}$, we define the chain rule for vector calculus in Equation 1.2.3. It is important to note that the order of multiplication of terms in Equation 1.2.3 is reversed from the ordering usually used in univariate calculus. This is necessary since matrix multiplication is non-commutative. Also note that the result of applying the chain rule in this form respects our convention that the matrix of partial derivatives should have size $C \times A$ since $\mathbf{f}$ has A dimensions and $\mathbf{h}$ has C dimensions.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{h}} = \frac{\partial \mathbf{g}}{\partial \mathbf{h}} \frac{\partial \mathbf{f}}{\partial \mathbf{g}} = \begin{bmatrix} \frac{\partial g_1}{\partial h_1} & \frac{\partial g_2}{\partial h_1} & \cdots & \frac{\partial g_B}{\partial h_1} \\ \frac{\partial g_1}{\partial h_2} & \frac{\partial g_2}{\partial h_2} & \cdots & \frac{\partial g_B}{\partial h_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial h_C} & \frac{\partial g_2}{\partial h_C} & \cdots & \frac{\partial g_B}{\partial h_C} \end{bmatrix} \begin{bmatrix} \frac{\partial f_1}{\partial g_1} & \frac{\partial f_2}{\partial g_1} & \cdots & \frac{\partial g_A}{\partial g_1} \\ \frac{\partial f_1}{\partial g_2} & \frac{\partial f_2}{\partial g_2} & \cdots & \frac{\partial g_A}{\partial g_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial g_B} & \frac{\partial f_2}{\partial g_B} & \cdots & \frac{\partial g_A}{\partial g_B} \end{bmatrix} \tag{1.2.3}$$

Now assume that $\mathbf{f}$ is an $A \times B$ dimensional matrix-valued function, $\mathbf{g}$ is a $C \times D$ dimensional matrix-valued function, and $\mathbf{h}$ is an $E \times F$ dimensional matrix-valued function. Again assuming that $\mathbf{f}$ is a function of $\mathbf{g}$ and $\mathbf{g}$ is a function of $\mathbf{h}$, we define the chain rule for matrix calculus element-wise: $\frac{\partial f_{ij}}{\partial h_{mn}} = \sum_{k=1}^{C} \sum_{l=1}^{D} \frac{\partial f_{ij}}{\partial g_{kl}} \frac{\partial g_{kl}}{\partial h_{mn}}$

Chapter 2

Decision Theory, Inference, and Learning

This chapter introduces the learning and inference frameworks used in this thesis. We adopt a decision-theoretic perspective based on a general prediction problem where we are given a set of pairs $\{\mathbf{y}_n, \mathbf{x}_n\}$, $n = 1, ..., N$ along with a loss function $l(\mathbf{y}, \hat{\mathbf{y}})$. The goal is to estimate a prediction function or decision rule $f(\mathbf{x})$ that achieves the lowest possible loss on future examples. In the collaborative prediction case, $\mathbf{y}_n$ corresponds to a subset of the values in $\mathbf{x}_n^m$, and $\mathbf{x}_n$ is replaced by $\mathbf{x}_n^o$. In the classification with missing features case, $\mathbf{y}_n$ takes a single categorical value, and $\mathbf{x}_n$ is replaced with $\mathbf{x}_n^o$.

We begin by reviewing decision making and the Bayes optimal prediction function. We then turn to the Bayesian inference framework and introduce Markov chain Monte Carlo (MCMC) methods including the Metropolis Hastings algorithm and the Gibbs sampler. We present the Maximum a Posteriori principle as an approximation to Bayesian inference, and review the Expectation Maximization algorithm. We discuss a direct function approximation framework that includes neural networks, and logistic regression. We briefly review empirical evaluation procedures for estimating expected loss.

2.1 Optimal Prediction and Minimizing Expected Loss

We assume that there is a fixed but unknown generative probability distribution $p_G(\mathbf{y}, \mathbf{x})$ over pairs $(\mathbf{y}, \mathbf{x})$. The goal of the prediction problem can be formally stated as selecting a prediction function $f(\mathbf{x})$ that achieves the lowest possible expected loss on examples drawn from $p_G(\mathbf{y}, \mathbf{x})$. The expected loss of a prediction function $f(\mathbf{x})$ under the generative distribution $p_G(\mathbf{y}, \mathbf{x})$ is defined in equation 2.1.1.

$$E_{p_G}[l(y, f(\mathbf{x}))] = \int\int l(y, f(\mathbf{x}))p_G(\mathbf{y}, \mathbf{x})d\mathbf{y}d\mathbf{x} \tag{2.1.1}$$

The theoretical minimum value of the expected loss is known as the Bayes optimal loss or the Bayes error rate. The Bayes optimal loss is achieved by the Bayes optimal prediction function shown in Equation 2.1.2 [57, p. 174]. The Bayes optimal prediction $f_O(\mathbf{x})$ given a vector $\mathbf{x}$ is equal to the value $\hat{y}$ that minimizes the expectation of the loss taken with respect to the conditional distribution $p_G(\mathbf{y}|\mathbf{x})$. The Bayes optimal loss is expressed in Equation 2.1.3.

$$f_O(\mathbf{x}) = \arg\min_{\hat{\mathbf{y}}} \int l(\mathbf{y}, \hat{\mathbf{y}})p_G(\mathbf{y}|\mathbf{x})d\mathbf{y} \tag{2.1.2}$$

$$E_{p_G}[l(y, f_O(\mathbf{x}))] = \int\int l(y, f_O(\mathbf{x}))p_G(\mathbf{y}, \mathbf{x})d\mathbf{y}d\mathbf{x} \tag{2.1.3}$$

Prediction frameworks differ in how they approximate the Bayes optimal prediction function. Bayesian methods are closest in spirit to the Bayes optimal prediction rule and replace $p_G(y|\mathbf{x})$ in Equation 2.1.2 with the posterior distribution over a set of models distributions. Maximum a posteriori approximations replace $p_G(y|\mathbf{x})$ in Equation 2.1.2 with the single model distribution that attains the highest posterior probability among a given set of model distributions. The classical maximum likelihood principle replaces $p_G(y|\mathbf{x})$ in Equation 2.1.2 with the single model distribution with the highest likelihood given the training data. Direct function approximation frameworks including neural networks and logistic regression avoid approximating the predictive distribution $p_G(y|\mathbf{x})$ by directly approximating the optimal prediction function.

2.2 The Bayesian Framework

The Bayesian solution to the prediction problem consists of specifying a family of probability distributions (a model), and assigning a prior probability to each distribution in the family. Once the sample of data is observed, the posterior predictive distribution is computed and substituted for the unknown $p_G(y|\mathbf{x})$. To make this concrete, assume that the model is parametric with parameter θ. Each distribution in the model family has the form $p_M(\mathbf{y}|\mathbf{x}, \theta)$ for some value of θ. The prior probability of each distribution in the set can then be given as a prior probability $q(\theta)$ on the parameter θ.

2.2.1 Bayesian Approximation to the Prediction Function

The posterior distribution of θ is found using Bayes rule as shown below in Equation 2.2.1. The posterior predictive distribution is shown in Equation 2.2.2.

$$p_M(\theta|\{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q) = \frac{q(\theta)\prod_{n=1}^{N} p_M(\mathbf{y}_n|\mathbf{x}_n, \theta)}{\int q(\theta)\prod_{n=1}^{N} p_M(\mathbf{y}_n|\mathbf{x}_n, \theta)d\theta} \tag{2.2.1}$$

$$p_M(\mathbf{y}|\mathbf{x}, \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q) = \int p_M(\mathbf{y}|\mathbf{x}, \theta)p_M(\theta|\{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q)d\theta \tag{2.2.2}$$

The Bayesian prediction function given in Equation 2.2.3 is obtained by substituting the model posterior distribution shown in equation 2.2.2 for the true conditional distribution $p_G(\mathbf{y}|\mathbf{x})$ in Equation 2.1.2.

$$f_B(\mathbf{x}) = \arg\min_{\hat{\mathbf{y}}} \int l(\mathbf{y}, \hat{\mathbf{y}})p_M(\mathbf{y}|\mathbf{x}, \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q)d\mathbf{y} \tag{2.2.3}$$

2.2.2 Bayesian Computation

The Bayesian approximation strategy relies on the ability to analytically compute the integrals in Equations 2.2.1 and 2.2.2. Practical applications of the Bayesian approach rely on an additional layer of approximations provided by Markov chain Monte Carlo methods [51, p. 357-381]. Monte Carlo methods compute integrals and expectations by reducing them to sums over a finite number of sample points. In Markov chain Monte Carlo methods, the sample points are provided by Markov chain methods like the Metropolis Hastings algorithm and the Gibbs sampler.

Suppose, for the moment, that we have a method for drawing independent samples θ_s from the parameter posterior given in Equation 2.2.1. The Monte Carlo approximation to the posterior predictive distribution is given in Equation 2.2.4. The corresponding prediction function is given in Equation 2.2.5.

$$p_M(\mathbf{y}|\mathbf{x}, \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q) \approx \frac{1}{S}\sum_{s=1}^{S} p_M(\mathbf{y}|\mathbf{x}, \theta_s) \tag{2.2.4}$$

$$f_B(\mathbf{x}) \approx \arg\min_{\hat{\mathbf{y}}} \int l(\mathbf{y}, \hat{\mathbf{y}})\frac{1}{S}\sum_{s=1}^{S} p_M(\mathbf{y}|\mathbf{x}, \theta_s)d\mathbf{y} \tag{2.2.5}$$

The validity of the Monte Carlo approximation relies on asymptotic theory which states that as the number of independent samples goes to infinity, the sampling approximation becomes arbitrarily close to the true distribution. For some posterior expectations, the Monte Carlo

approximation can be quite accurate using only a small number of samples [51, p. 357].

In practice it is not possible to draw independent samples θ_s from an arbitrary posterior distribution. Markov chain sampling methods define a Markov chain where the state at time t is the parameter vector θ_t, and the equilibrium distribution is the posterior distribution of θ. Samples are obtained by starting the Markov chain in a random initial state, and running it for a sufficient number of steps to reach equilibrium. If sufficient iterations are allowed between samples at equilibrium, the samples will be approximately independent draws from the model posterior [25, p. 287].

The Metropolis-Hastings Algorithm

For simplicity we will drop the dependence on the data and write the posterior distribution on θ as $p(\theta)$. The Metropolis-Hastings algorithm requires the specification of a proposal distribution $p'(\theta_{new}|\theta_t)$ that gives the probability of transitioning to the state θ_{new} from the current state θ_t. The algorithm proceeds by sampling a candidate value θ_{new} from $p'(\theta_{new}|\theta_t)$, and setting θ_{t+1} to θ_{new} with probability a_{t+1}, and θ_t with probability $1 - a_{t+1}$. The probability a_{t+1} is called the acceptance probability and is defined below [25, p. 291].

$$a_{t+1} = \max\left(1, \frac{p(\theta_{new})p'(\theta_t|\theta_{new})}{p(\theta_t)p'(\theta_{new}|\theta_t)}\right) \tag{2.2.6}$$

$$\theta_{t+1} \leftarrow \begin{cases} \theta_{new} & \text{With probability } a_{t+1} \\ \theta_t & \text{otherwise} \end{cases} \tag{2.2.7}$$

A useful property of the Metropolis-Hastings method is that the normalizing factor in the posterior cancels out in the acceptance ratio. As a result, it suffices to compute the posterior up to a constant of proportionality. Asymptotic convergence of the Markov chain defined by the Metropolis-Hastings updates to the posterior distribution $p(\theta)$ is guaranteed if $p'(\theta'|\theta) > 0$ for all θ', θ [51, p. 366].

The Gibbs Sampler

The Gibbs sampler is a particular form of Metropolis-Hastings algorithm based on updating each parameter θ_k in the parameter vector θ according to its posterior distribution given the remaining parameters. The utility of the Gibbs sampler stems from the fact that in structured models it is often possible to sample efficiently from the posterior distribution of a single parameter or a small group of parameters, even when it is not possible to sample efficiently from the complete posterior. A single round of Gibbs updates is described below [25, p. 287].

$$\theta_{1t+1} \sim p(\theta_1|\theta_{2t}, ..., \theta_{Kt}, \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N})$$
$$\vdots$$
$$\theta_{Kt+1} \sim p(\theta_K|\theta_{1t+1}, ..., \theta_{K-1t+1}, \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N})$$

Since the proposal distribution for each update is based on the actual conditional distribution for each parameter, the acceptance ratio for the Gibbs proposal distribution is always equal to 1, and the proposal is always accepted.

2.2.3 Practical Considerations

Assuming the required regularity conditions hold for the Markov chain implementing a given posterior distribution, one still needs to know both how long to simulate the Markov chain before equilibrium is reached and how long to wait between samples. As MacKay indicates, lower bounds on time to equilibrium can sometimes be established, but precisely predicting time to equilibrium is a difficult problem [51, p. 379]. In general, it is not possible to guarantee that equilibrium has been reached in a running simulation. It could always be the case that if the simulation were run for additional steps, a new set of states with high posterior probability could be found. Nevertheless, the use of approximate Bayesian methods based on pragmatic choices for the number of samples and the length of the simulation can lead to good results in practice.

2.3 The Maximum a Posteriori Framework

The maximum a posteriori (MAP) approach to the prediction problem is based on selecting the single distribution with highest posterior probability from a family of probability distributions given a set of observations and a prior distribution. The classical maximum likelihood framework developed by Fisher [19] is based on selecting the single distribution with the highest likelihood given the data. We restrict our discussion to the more general case of MAP estimation.

2.3.1 MAP Approximation to The Prediction Function

Again assume that we have a family of distributions indexed by a parameter θ such that each distribution in the family has the form $p_M(\mathbf{y}|\mathbf{x}, \theta)$ with prior probability $q(\theta)$. The posterior distribution of θ is again found using Bayes rule as shown below. θ_{MAP} is set to a value of θ

that obtains maximum posterior probability. Note that in general several different parameter vectors θ could attain the same maximum value of the posterior probability.

$$p_M(\theta|\{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q) = \frac{q(\theta)\prod_{n=1}^{N} p_M(\mathbf{y}_n|\mathbf{x}_n, \theta)}{\int q(\theta)\prod_{n=1}^{N} p_M(\mathbf{y}_n|\mathbf{x}_n, \theta)d\theta} \tag{2.3.1}$$

$$\theta_{MAP} = \arg\max_{\theta} p_M(\theta|\{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1:N}, q) \tag{2.3.2}$$

The maximum a posteriori prediction function is obtained by substituting the single distribution $p_M(\mathbf{y}|\mathbf{x}, \theta_{MAP})$ for $p_G(\mathbf{y}|\mathbf{x})$ in Equation 2.1.2. The maximum a posterior prediction function f_{MAP} is shown in Equation 2.3.3.

$$f_{MAP}(\mathbf{x}) = \arg\min_{\hat{\mathbf{y}}} \int l(\mathbf{y}, \hat{\mathbf{y}}) P(\mathbf{y}|\mathbf{x}, \theta_{MAP}) d\mathbf{y} \tag{2.3.3}$$

2.3.2 MAP Computation

Computation of the maximum a posteriori parameters θ_{MAP} is accomplished through optimization of the posterior probability. Assuming that all first order partial derivatives of the posterior distribution with respect to the parameters exist, the value of θ_{MAP} can be found by solving the following set of gradient equations. Note that the normalization term is constant with respect to θ, and has been omitted below.

$$\frac{\partial}{\partial \theta_k} q(\theta) \prod_{n=1}^{N} p_M(\mathbf{y}_n|\mathbf{x}_n, \theta) = 0 \ \ldots \text{ for all } K \tag{2.3.4}$$

For more complex models, the system of equations must be solved using non-linear numerical optimization techniques such as gradient ascent, conjugate gradient, or Newton methods [62]. It is important to note that these are all local optimization methods in the sense that they return a solution consisting of a parameter vector θ_* that is only guaranteed to be optimal in a local neighbourhood. In many practical situations it is not possible to fully implement the MAP principle. Instead, the best locally optimal set of parameters found using random restarts of the optimization procedure is substituted for θ_{MAP}. The error in such an approximation is impossible to determine in general. Nevertheless, these methods are often found to work well in practice.

The Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm due to Dempster, Laird and Rubin is an iterative numerical procedure for finding maximum a posteriori parameter estimates [17]. It is useful

when the probability $p_M(\mathbf{y}_n|\mathbf{x}_n, \theta)$ is obtained by integrating over latent or unobserved variables $\mathbf{Z}$. For example, a finite mixture distribution has the form $\sum_{k=1}^{K} p(\mathbf{y}_n|\mathbf{x}_n, Z = k, \theta)p(Z = k|\theta)$.

The EM algorithm proceeds by randomly initializing the parameters to θ_0. On each iteration t, the posterior probability of the missing variables $\mathbf{Z}$ is computed for each data case n given the values of the observed variables and the current parameters. In the maximization step, θ_{t+1} is set to the value which maximizes the expected complete log posterior. These two updates are iterated as shown below until the posterior converges.

$$\begin{aligned} \text{E-Step:} \quad q_n(\mathbf{z}) &\leftarrow \frac{p(\mathbf{y}|\mathbf{z}, \mathbf{x}, \theta_t)p(\mathbf{z}|\theta_t)}{\sum_{\mathbf{z}} p(\mathbf{y}|\mathbf{z}, \mathbf{x}, \theta_t)p(z|\theta_t)} \\ \text{M-Step:} \quad \theta_{t+1} &\leftarrow \arg\max_{\theta} \sum_{n=1}^{N} \sum_{\mathbf{z}} q_n(\mathbf{z}) \log\left(p(\mathbf{y}|\mathbf{z}, \mathbf{x}, \theta)p(\mathbf{z}|\theta)\right) + \log(q(\theta)) \end{aligned}$$

The primary advantage of the Expectation Maximization algorithm over other general iterative optimization procedures is that each iteration is guaranteed to improve the value of the log posterior. General optimization techniques require the use of line search or backtracking to provide a similar guarantee. Of course, like any general optimization technique, the Expectation Maximization procedure returns a locally optimal solution.

2.4 The Direct Function Approximation Framework

In the Bayesian and MAP frameworks, the main idea is to approximate the true conditional distribution $p_G(\mathbf{y}|\mathbf{x})$ using an alternative predictive distribution inferred from a sample of data. The alternative predictive distribution is then used to derive a prediction function. The alternative approach is to circumvent the estimation of a predictive distribution $p_M(\mathbf{y}|\mathbf{x}, \theta)$, and instead directly approximate the prediction function.

Direct function approximation is used with models like perceptrons [63], neural networks [7], and support vector machines [10]. The direct function approximation approach to the prediction problem requires defining a family of functions, and specifying a rule for choosing the best function given a sample of data.

2.4.1 Function Approximation as Optimization

Suppose that the set of functions is parametric with typical elements of the form $f_\theta(\mathbf{x})$, where θ is the parameter vector. The rule for choosing the best value of θ given the sample of N observed pairs $(\mathbf{y}_n, \mathbf{x}_n)$ is normally described as the solution to a minimization problem as shown in Equation 2.4.1. The objective function L maps each function f_θ to a real-valued score.

$$\theta^* = \arg\min_{\theta} L(f_\theta, \{\mathbf{y}_n, \mathbf{x}_n\}_{n=1:N}) \tag{2.4.1}$$

Given a loss function l, the most straightforward objective function is the empirical loss given by Equation 2.4.2. Optimizing the empirical loss function selects a function f_θ with the minimum loss on the training set. Note that this function need not be unique.

$$L_E(f_\theta, \{\mathbf{y}_n, \mathbf{x}_n\}_{n=1:N}) = \frac{1}{N}\sum_{n=1}^{N} l(\mathbf{y_n}, f_\theta(\mathbf{x}_n)) \tag{2.4.2}$$

2.4.2 Function Approximation and Regularization

The empirical loss function L_E is often a poor estimator of the expected loss $E_{p_G}[l(y, f_\theta(\mathbf{x}))]$ due to overfitting the training data. Regularization methods modify the empirical loss in a way that penalizes complex functions. Let $g(\theta)$ be a function that assigns a real value to each f_θ depending on its complexity for a given notion of complexity. We assume that $g(\theta)$ decreases as the complexity of f_θ increases. Equation 2.4.3 gives a regularized objective function L_R with parameter λ controlling the trade-off between fitting the training data and penalizing the complexity of the functions.

$$L_R(f_\theta, \{\mathbf{y}_n, \mathbf{x}_n\}_{n=1:N}) = \frac{1}{N}\sum_{n=1}^{N} l(\mathbf{y_n}, f_\theta(\mathbf{x}_n)) + \lambda g(\theta) \tag{2.4.3}$$

Subset selection, more commonly known as feature selection in the machine learning community, is a form of regularization that trades off between loss on the training set and the number of features used by the model. Hoerl's ridge regression uses a quadratic regularization function of the form $g(\theta) = \theta^T\theta$ [38]. Ridge regression is more commonly referred to as "weight decay" or $L2$ regularization in the machine learning literature [35]. Tibshirani's "lasso" corresponds to the penalty function $g(\theta) = \sum_{k=1}^{K} |\theta_k|$ [72]. This regularization function is known in machine learning research as $L1$ regularization. $L2$ regularization has the advantage that the regularized objective function remains continuous and differentiable if the original loss function is continuous and differentiable. Subset selection requires a search over subsets of features that can be costly to perform. $L1$ regularization introduces discontinuities in the regularized objective function, which require specialized optimization methods.

2.5 Empirical Evaluation Procedures

While theoretical justifications can be given for each of the prediction frameworks discussed in this chapter, in practice, the underlying assumptions are almost never known to hold with certainty. As a result, the performance of prediction methods must be established through the use of empirical evaluation procedures that estimate expected loss. There are several possible estimates for expected loss including training set loss, loss on a held out validation set, and cross validation loss.

2.5.1 Training Loss

The training loss can sometimes give an accurate estimate of the expected prediction loss, but in most cases will be overly optimistic. Denote the training set by $D_T = \{\mathbf{y}_n, \mathbf{x}_n | n = 1, ..., N\}$, and a function estimated based on D_T by $f_{\theta|D_T}$. The training loss of the function $f_{\theta|D_T}$ is given in Equation 2.5.1.

$$L_T(f_{\theta|D_T}, D_T) = \frac{1}{N} \sum_{n=1}^{N} l(\mathbf{y_n}, f_{\theta|D_T}(\mathbf{x}_n)) \tag{2.5.1}$$

In a classification setting with a limited number of input patterns $\mathbf{x}$, the training loss can be an accurate estimate of the expected loss if all patterns are seen during training. In a classification setting where the number of possible input patterns is infinite, a method that achieves zero training loss can have expected loss arbitrarily close to that of random guessing.

2.5.2 Validation Loss

The most straightforward, unbiased estimate of the expected loss is obtained by randomly partitioning the available data into a training set $D_T = \{\mathbf{y}_n, \mathbf{x}_n | n = 1, ..., N_T\}$ and a validation set $D_V = \{\mathbf{y}_n, \mathbf{x}_n | n = 1, ..., N_V\}$. The function $f_{\theta|D_T}$ is selected based on the training set D_T only. The loss is assessed on the validation set D_V.

$$L_V(f_{\theta|D_T}, D_V) = \frac{1}{N_V} \sum_{n=1}^{N_V} l(\mathbf{y_n}, f_{\theta|D_T}(\mathbf{x}_n)) \tag{2.5.2}$$

The variance of validation loss as an estimate of expected loss can be lowered by increasing the size of the validation set. This exposes the obvious drawback of validation loss: using a separate validation set necessitates a reduction in the available training data. When large

amounts of training data are available, this may not be problematic. With small and medium sized samples, we might hope to make more effective use of the available data.

2.5.3 Cross Validation Loss

Cross validation loss provides an estimate of the expected loss that can be nearly unbiased [32, p. 215]. To compute the cross validation loss, the available data is randomly partitioned into K equally sized blocks $D_1, ..., D_K$. We define the data set $D_{-k} = \bigcup_{l \neq k} D_l$. For each k, a function $f_{\theta|D_{-k}}$ is selected based on the training set D_{-k}, and evaluated on the validation set D_k. The loss values obtained by testing on each D_k are averaged together to obtain the cross validation loss [71]. The cross validation loss is defined in Equation 2.5.3. For convenience we define $f^k_{\theta|D} = f_{\theta|D_{-k}}$, and introduce an indexing function $\kappa(n)$, which maps data case indices $1, ..., N$ to the corresponding block indices $1, ..., K$.

$$L_{CV}(f_{\theta|D}, D) = \frac{1}{N} \sum_{n=1}^{N} l(\mathbf{y_n}, f^{\kappa(n)}_{\theta|D}(\mathbf{x}_n)) \tag{2.5.3}$$

By alternately holding out several small subsets of data for testing, we are able to use more data for training each model. Cross validation can also be used as a method for selecting free parameters in learning algorithms by evaluating the cross validation loss of several alternative parameter sets.

Chapter 3

A Theory of Missing Data

In this chapter we review the theory of missing data due to Little and Rubin [49]. We formally introduce the classes of missing data. We present a detailed analysis of the missing at random assumption in the case of multivariate missing data. We review the effect of different types of missing data on probabilistic inference. Finally, we highlight the impact of data model misspecification on inference with missing data.

3.1 Categories of Missing Data

There are essentially two ways to factorize the joint distribution of the data, the latent variables, and the response indicators as far as the response indicators are concerned. In the first case the joint distribution is factorized as $P(\mathbf{R}|\mathbf{X},\mathbf{Z},\mu)P(\mathbf{X},\mathbf{Z}|\theta)$. $P(\mathbf{R}|\mathbf{X},\mathbf{Z},\mu)$ is referred to as the missing data model, and $P(\mathbf{X},\mathbf{Z}|\theta)$ is referred to as the complete data model. The intuition is that the response probability depends on the true values of the data vector and latent variables. In the second factorization the conditioning is reversed, and the joint distribution is written as $P(\mathbf{X},\mathbf{Z}|\mathbf{R},\vartheta)P(\mathbf{R}|\nu)$. The intuition is that each pattern of missing data specifies a different distribution for the data and latent variables. [49, p. 219]. In this chapter and throughout most of this work, we adopt the first factorization, but it is important to note that the alternative factorization is equally valid.

In the classification scheme proposed by Little and Rubin, the first category of missing data is called "missing completely at random" or MCAR. Data is MCAR when the response indicator variables $\mathbf{R}$ are independent of the data variables $\mathbf{X}$ and the latent variables $\mathbf{Z}$. The MCAR condition can be succinctly expressed by the relation $P(\mathbf{R}|\mathbf{X},\mathbf{Z},\mu) = P(\mathbf{R}|\mu)$.

The second category of missing data is called "missing at random" or MAR. The MAR condition is frequently written as $P(\mathbf{R}=\mathbf{r}|\mathbf{X}=\mathbf{x},\mathbf{Z}=\mathbf{z},\mu) = P(\mathbf{R}=\mathbf{r}|\mathbf{X}^o=\mathbf{x}^o,\mu)$ for all $\mathbf{x}^m$, $\mathbf{z}$, and μ [49, p. 12]. It is intended to express the fact that the probability of response can not

depend on missing data values or latent variables. A precise definition of the MAR condition requires the introduction of additional notation. Define a function $f(\mathbf{x}, \mathbf{r}) = \mathbf{v}$ such that $v_d = x_d$ if $r_d = 1$, and $v_d = \varnothing$ if $r_d = 0$. The function $f()$ maps a data vector and response indicator vector into an alternative representation where the missing dimensions of $\mathbf{x}$ are replaced with the symbol $\varnothing$. Given $\mathbf{x}$ and $\mathbf{r}$ we compute $\mathbf{v}$, and consider the set $C = \{(\mathbf{x}', \mathbf{z}') | P(\mathbf{X} = \mathbf{x}', \mathbf{Z} = z') > 0, f(\mathbf{x}', \mathbf{r}) = \mathbf{v}\}$. The set C contains all pairs $(\mathbf{x}', \mathbf{z}')$ with non-zero probability where $\mathbf{x}'$ agrees with $\mathbf{x}$ on the observed dimensions specified by $\mathbf{r}$.

The missing dimensions of $\mathbf{x}$ are missing at random for a fixed response vector $\mathbf{r}$ if $P(\mathbf{R} = \mathbf{r} | \mathbf{X} = \mathbf{x}', \mathbf{Z} = z', \mu)$ is constant for all $(\mathbf{x}', \mathbf{z}')$ in the set C, and all values of the parameter μ [64, p. 582]. In other words, given the values of the observed dimensions, the probability of the response vector is the same regardless of the values we fill in for the missing dimensions and latent variables.

The last class of missing data is referred to by several names including "non-ignorable" (NI) and "not missing at random" (NMAR). Data is NMAR when the response indicator variable $\mathbf{R}$ depends on either the unobserved values contained in $\mathbf{x}^m$, or the unobserved values of the latent variables $\mathbf{z}$. It may also depend on the observed data variables $\mathbf{x}^o$. A simple example of NMAR occurs when the probability that a particular data dimension is missing depends on the value of that data dimension.

3.2 The Missing at Random Assumption and Multivariate Data

The interpretation and implications of the missing at random assumption can be quite subtle in the multivariate setting with arbitrary patterns of missing data. In many of the examples considered by Little and Rubin it is assumed that the data vector can be partitioned into two parts where the first part is subject to non-response, and the second part is always observed. If we define the always observed sub-vector to be $\mathbf{x}^a$, then a missing data model of the form $P(\mathbf{R} = \mathbf{r} | \mathbf{X}^a = \mathbf{x}^a, \mu)$ satisfies the missing at random condition.

In the multivariate setting with arbitrary patterns of missing data, we find it more intuitive to think of the missing at random condition as imposing constraints on the parameters of the conditional distribution $P(\mathbf{R} = \mathbf{r} | \mathbf{X} = \mathbf{x}, \mu)$. Recall that the precise definition of the missing at random assumption states that for a fixed value of $\mathbf{r}$, $P(\mathbf{R} = \mathbf{r} | \mathbf{X} = \mathbf{x}', \mu)$ must take the same value for any complete vector $\mathbf{x}'$ that agrees with the observed dimensions of $\mathbf{x}$ as specified by $\mathbf{r}$.

In the case where $\mathbf{x}$ is a vector of discrete values, we can parameterize the conditional distribution as $P(\mathbf{R} = \mathbf{r} | \mathbf{X} = \mathbf{x}', \mu) = \mu_{\mathbf{r}\mathbf{x}'}$. The missing at random assumption then specifies constraints on the parameters $\mu_{\mathbf{r}\mathbf{x}'}$. In particular, if $\mathbf{u}$ and $\mathbf{w}$ are complete vectors that both

$X\backslash R$	0 0	0 1	1 0	1 1
0 0	α	β	γ	$1-\alpha-\beta-\gamma$
0 1	α	δ	γ	$1-\alpha-\delta-\gamma$
1 0	α	β	λ	$1-\alpha-\beta-\lambda$
1 1	α	δ	λ	$1-\alpha-\delta-\lambda$

Table 3.1: Restrictions imposed on the parameterization of $P(\mathbf{R} = \mathbf{r}|\mathbf{X} = \mathbf{x})$ by the MAR condition.

agree with the observed dimensions of $\mathbf{x}$ as specified by $\mathbf{r}$, the MAR condition requires $\mu_{\mathbf{ru}} = \mu_{\mathbf{rw}}$.

It is important to note that the MAR condition applies to individual response vectors $\mathbf{r}$ and corresponding data vectors $\mathbf{x}$. It may be the case that the missing data in one data case generated from a particular missing data mechanism is missing at random, while the missing data in another data case is not missing at random. In order for a missing data model to only generate random missing data, the MAR parameter constraints must hold for every vector $\mathbf{r}$ and corresponding instantiation of $\mathbf{x}^o$.

To help understand the restrictions placed on the parameterization of $P(\mathbf{R} = \mathbf{r}|\mathbf{X} = \mathbf{x})$ by the missing at random assumption, we elaborate on an example presented by Little and Rubin [50, p. 18]. Consider two dimensional binary data vectors subject to non-response on both dimensions. The missing at random assumption places a set of constraints on the parameters of $P(\mathbf{R} = \mathbf{r}|\mathbf{X} = \mathbf{x}, \mu)$ that requires certain elements of the conditional distribution to be equal. We illustrate this in Table 3.1.

When $\mathbf{R} = [0, 0]$, neither X_1 nor X_2 is observed, and the missing at random condition stipulates that $P(\mathbf{R} = [0, 0]|\mathbf{X} = \mathbf{x})$ must be equal for all $\mathbf{x}$. In the case where $\mathbf{R} = [0, 1]$, X_2 is observed. We get the restriction that $P(\mathbf{R} = [0, 1]|X_1 = x_1, X_2 = 0)$ must be equal for all x_1, and $P(\mathbf{R} = [0, 1]|X_1 = x_1, X_2 = 1)$ must be equal for all x_1. In the case where $\mathbf{R} = [1, 0]$, X_1 is observed. We get the restriction that $P(\mathbf{R} = [1, 0]|X_1 = 0, X_2 = x_2)$ must be equal for all x_2, and $P(\mathbf{R} = [1, 0]|X_1 = 1, X_2 = x_2)$ must be equal for all x_2. When all dimensions are observed, the missing at random assumption puts no restrictions on $P(R = [1, 1]|X = x)$, and its value is simply determined by the normalization constraint.

The essence of the missing at random condition is the symmetry imposed on the missing data model, which make it possible to determine $P(\mathbf{R} = \mathbf{r}|\mathbf{X} = \mathbf{x}, \mu)$ given only $\mathbf{r}$, and the observed values of $\mathbf{x}$. For example, suppose that $x_2 = 0$, and x_1 is unobserved so that $\mathbf{r} = [0, 1]$. Even though we do not know the value of x_1 we can still determine that the value of $P(\mathbf{R} = \mathbf{r}|\mathbf{X} = \mathbf{x}, \mu)$ must be β, since it is β both when $x_1 = 0$, and when $x_1 = 1$. When the MAR condition is satisfied, the observed values in $\mathbf{x}$ always contain exactly the information needed to determine $P(\mathbf{R} = \mathbf{r}|\mathbf{X} = \mathbf{x}, \mu)$.

From a modeling perspective, it makes more sense to assert that the random variable R_1 either depends on the random variable X_1, or is independent of the random variable X_1. If R_1 is always dependent on X_1, and X_1 is missing in a given data case, then X_1 will not be missing at random. This makes parameter estimation more difficult. On the other hand, the MAR assumption allows R_1 to depend on X_1 for some values of R_1, and not for others. This is convenient, but unnatural and difficult to justify. Little and Rubin acknowledge that assuming the missing at random condition in a multivariate setting with arbitrary patterns of missing data is not very realistic, but can sometimes lead to reasonable results if there are sufficient observed covariates to do a reasonable job of predicting the response patterns [50, p. 19].

3.3 Impact of Incomplete Data on Inference

Rubin considered the impact of incomplete data on Bayesian inference [64, p. 587]. These results are also contained in the more recent text by Schafer [67, p. 17]. In this section we describe the effect of random missing data on Bayesian inference, and contrast it with the effect of non-random missing data. The results for maximum likelihood and maximum a posteriori inference are analogous, and discussed at length by Little and Rubin [49, p. 89].

Consider a joint parametric model over data variables, latent variables, and response indicators of the form $P(\mathbf{R}|\mathbf{X},\mathbf{Z},\mu)P(\mathbf{X},\mathbf{Z}|\theta)$, and assume the prior distribution is factorized as $P(\theta|\omega)P(\mu|\eta)$. The posterior distribution $P(\theta|\{\mathbf{x}_n,\mathbf{r}_n\}_{1:N},\omega,\eta)$ on θ given a sample of incomplete data vectors $\mathbf{x}_n, n = 1,...,N$, and the prior parameters is given in Equation 3.3.1.

$$P(\theta|\{\mathbf{x}_n,\mathbf{r}_n\}_{1:N},\omega,\eta) \quad \propto P(\theta|\omega)\int P(\mu|\eta)\prod_{n=1}^{N}\int\int P(\mathbf{x}_n^o,\mathbf{x}^m,\mathbf{z}|\theta)P(\mathbf{r}_n|\mathbf{x}_n^o,\mathbf{x}^m,\mathbf{z},\mu)d\mathbf{x}^m d\mathbf{z} d\mu \tag{3.3.1}$$

Without making any simplifying assumptions, the missing data and complete data models are coupled together by the integration over both the missing data values, and the latent variable values. Under the missing at random or missing completely at random conditions, $P(\mathbf{r}_n|\mathbf{x}_n^o,\mathbf{x}^m,\mathbf{z},\mu)$ is constant for all values of $\mathbf{x}^m$ and $\mathbf{z}$ when we fix $\mathbf{r}_n$ and $\mathbf{x}_n^o$. As a result, the posterior on θ is completely independent of $\mathbf{r}_n$, μ and η, and the missing data model can be removed from the posterior. The integration over missing data values then reduces to marginalization within the complete data model only. The result of these simplifications is the observed data posterior shown in Equation 3.3.2.

$$P^{obs}(\theta|\{\mathbf{x}_n,\mathbf{r}_n\}_{1:N},\omega) \propto P(\theta|\omega)\prod_{n=1}^{N}\int P(\mathbf{x}_n^o,\mathbf{z}|\theta)d\mathbf{z} \tag{3.3.2}$$

$\mathbf{x}$	$P(\mathbf{x})$	$P(\mathbf{R}=[0,0]\|\mathbf{x})$	$P(\mathbf{R}=[0,1]\|\mathbf{x})$	$P(\mathbf{R}=[1,0]\|\mathbf{x})$	$P(\mathbf{R}=[1,1]\|\mathbf{x})$
0 0	a	α	β	γ	$1-\alpha-\beta-\gamma$
0 1	b	α	δ	γ	$1-\alpha-\delta-\gamma$
1 0	c	α	β	λ	$1-\alpha-\beta-\lambda$
1 1	d	α	δ	λ	$1-\alpha-\delta-\lambda$

Table 3.2: Specification of true $P(X=x)$ and $P(R=r|X=x)$ satisfying MAR.

The MCAR and MAR conditions effectively allow us to ignore the missing data model without impacting the validity of inference. When the MCAR and MAR conditions fail to hold, the missing data is not missing at random. Ignoring the missing data model and basing inference for θ on the observed data posterior will result in biased inference for data model parameters, and biased predictions.

3.4 Missing Data, Inference, and Model Misspecification

It is important to note that when then MAR condition fails to hold it is not sufficient to use an arbitrary missing data model. Inference will also be biased if an incorrect missing data model is used. A somewhat more surprising result is that even if the MAR assumption holds for the true generative model, inference for the parameters of a simpler data model can still be biased. This can be a problem in areas like machine learning where structured models are often used to obtain tractable learning methods, and to avoid overfitting.

To illustrate this issue again consider the simple setting of binary data vectors $\mathbf{x}$ in two dimensions subject to missing data. The parameters of the missing data mechanism $P(\mathbf{R}=\mathbf{r}|\mathbf{X}=\mathbf{x})$ are given in Table 3.2, along with the data distribution $P(\mathbf{X}=\mathbf{x})$. Note that the missing data model satisfies the MAR condition as described in Section 3.2. The missing data contained in any data case sampled from the generative model is missing at random.

The data variables X_1 and X_2 will not be independent in general under this generative process. However, as is often the case in machine learning, we consider learning a simpler, factorized model of the form $P^M(\mathbf{X}=\mathbf{x}|\theta)=\prod_d^D P^M(X_d=x_d|\theta_d)$ where $\theta_{vd}=P^M(X_d=v)$. Given the generative parameters in Table 3.2, we know that the correct model parameter values are $\theta_{11}=c+d$ and $\theta_{12}=b+d$. Given a sample of data from the generative process, we may be tempted to estimate the parameters θ_{vd} using the model's observed data likelihood function since the missing data is missing at random.

The observed data likelihood function is given in Equation 3.4.1. We analytically solve for the maximum likelihood parameter estimates using Lagrange multipliers to enforce the normalization constraints, obtaining the solution shown in Equation 3.4.2.

$$\begin{aligned}
\mathcal{L}^{obs} &= \sum_n \sum_d \sum_v r_{dn}[x_{dn} = v] \log \theta_{vd} && (3.4.1)\\
\frac{\partial \mathcal{L}^{obs}}{\partial \theta_{vd}} &= \sum_n \frac{r_{dn}[x_{dn} = v]}{\theta_{vd}} - \lambda = 0 \\
\lambda \theta_{vd} &= \sum_n r_{dn}[x_{dn} = v] \\
\sum_v \lambda \theta_{vd} &= \sum_v \sum_n r_{dn}[x_{dn} = v] \\
\lambda &= \sum_n r_{dn} \\
\theta_{vd} &= \frac{\sum_n r_{dn}[x_{dn} = v]}{\sum_n r_{dn}} && (3.4.2)
\end{aligned}$$

This maximum likelihood estimation procedure is asymptotically equivalent to computing $P(X_d = v|R_d = 1)$ using the true generative parameters. Consider the specific case of the asymptotic estimate of $P^M(X_1 = 1)$, or equivalently $P(X_1 = 1|R_1 = 1)$. As seen in Equation 3.4.4, the asymptotic estimate for $P^M(X_1 = 1)$ is not equal to the true value $c + d$ of the marginal distribution of X_1 unless $\beta = \delta$ in the missing data model. This corresponds to the more stringent assumption that X_2 is missing completely at random.

$$\begin{aligned}
P(X_1 = 1|R_1 = 1) &= \frac{P(R_1 = 1, |X_1 = 1)P(X_1 = 1)}{P(R_1 = 1)} && (3.4.3)\\
&= \frac{c(1-\alpha-\beta) + d(1-\alpha-\delta)}{a(1-\alpha-\beta) + b(1-\alpha-\delta) + c(1-\alpha-\beta) + d(1-\alpha-\delta)} \\
&= \frac{(c+d)(1-\alpha) - c\beta - d\delta}{(1-\alpha) - (a+c)\beta - (b+d)\delta} \\
&= c + d \quad \text{... only if } \beta = \delta && (3.4.4)
\end{aligned}$$

In general, computing the marginal distribution on dimension d from the observed data only will be biased unless all the other dimensions are missing completely at random. When the stronger missing completely at random condition is not believed to hold, the only correct solution is to estimate the full joint distribution, recover the data model parameters, and use these to estimate the parameters of the marginal distributions. Computing the parameters a, b, c, d of the true data generating distribution $P(X = x)$ can be accomplished using the Expectation Maximization algorithm. We redefine the parameters of the true distribution to $P(X_1 = i, X_2 = j) = \phi_{ij}$ for convenience. Define the count variables:

$\beta - \delta$	True $P(X_1 = 1)$	Est. $P(X_1 = 1)$	True $P(X_1 = 1 \mid R_1 = 1)$	Est. $P^M(X_1 = 1)$
0.5	0.8000	0.7999 ± 0.0007	0.7961	0.7961 ± 0.0007
1.0	0.8000	0.8004 ± 0.0006	0.7917	0.7923 ± 0.0006
1.5	0.8000	0.7996 ± 0.0006	0.7868	0.7860 ± 0.0007
2.0	0.8000	0.8011 ± 0.0007	0.7812	0.7826 ± 0.0008
2.5	0.8000	0.7990 ± 0.0007	0.7750	0.7737 ± 0.0008
3.0	0.8000	0.8000 ± 0.0007	0.7679	0.7679 ± 0.0007
3.5	0.8000	0.7994 ± 0.0008	0.7596	0.7582 ± 0.0009
4.0	0.8000	0.7999 ± 0.0009	0.7500	0.7501 ± 0.0010
4.5	0.8000	0.7992 ± 0.0010	0.7386	0.7379 ± 0.0010
5.0	0.8000	0.7986 ± 0.0010	0.7250	0.7241 ± 0.0010

Table 3.3: Results of a simulation experiment on learning marginal parameters.

$$\begin{aligned} C^1_{ij} &= \sum_n [r_1 = 1][r_2 = 1][x_1 = i][x_2 = j] \\ C^2_i &= \sum_n [r_1 = 1][r_2 = 0][x_1 = i] \\ C^3_j &= \sum_n [r_1 = 0][r_2 = 1][x_2 = j] \end{aligned}$$

One iteration of the EM algorithm can be defined in terms of the current estimate for the parameters ϕ and the observed count variables. We give the EM iteration in Equation 3.4.5 [67, p. 44].

$$\phi_{ij} \leftarrow \frac{1}{N}\left(C^1_{ij} + C^2_i \frac{\phi_{ij}}{\sum_j \phi_{ij}} + C^3_j \frac{\phi_{ij}}{\sum_i \phi_{ij}}\right) \tag{3.4.5}$$

We present a simulation study to empirically verify the claim that the estimation of the parameters in the factorized model is subject to bias even when the missing data is missing at random. We generate incomplete data cases using the model specified in Table 3.2. We set the true parameters as seen below. We fix $\delta = 0.1$, and vary β from 0.15 to 0.6 by varying t from 1 to 10.

$$\begin{aligned} a = \phi_{00} = 0.1 \qquad b = \phi_{01} = 0.1 \qquad c = \phi_{10} = 0.7 \qquad d = \phi_{11} = 0.1 \\ \alpha = 0.1 \qquad \beta = 0.1 + t0.05 \qquad \delta = 0.1 \qquad \gamma = 0.2 \end{aligned}$$

For each value of t, we sample 5000 data cases according to the true model. We also compute the true value of $P(X_1 = 1) = c + d$, which is 0.8 for all t. We use the EM iteration given in Equation 3.4.5 to estimate all the parameters of the true model from the observed data likelihood

under the true model. The estimated parameters $\hat{c}$ and $\hat{d}$ are then used to estimate $P(X_1 = 1)$ as $\hat{c} + \hat{d}$ for each t. Next, the true value of the observed data marginal $P(X_1 = 1|R_1 = 1)$ is computed for each t according to Equation 3.4.3. Finally, we use the observed data likelihood under the factorized model to compute an estimate $\hat{\theta}_{11}$ of $P^M(X_1 = 1)$ according to Equation 3.4.2. The entire experiment is repeated 100 times for each value of t. The average results are reported in Table 3.3, along with the standard error of the mean for estimated quantities.

The results show that as the true value of the β parameter diverges from the true value of the δ parameter, the estimated parameter $\hat{\theta}_{11}$ as given by Equation 3.4.2 diverges from the true value of $P(X_1 = 1)$. In addition, the estimate $\hat{\theta}_{11}$ is approximately equal to the true value of $P(X_1 = 1|R_1 = 1)$ as claimed. Finally, the results also show that estimating all the parameters of the true data model using EM and then estimating $P(X_1 = 1)$ as $\hat{c} + \hat{d}$ is not subject to bias as claimed.

It is important to realize that in order for parameter estimation to be unbiased, it is not sufficient for missing data to be missing at random with respect to a true underlying data model. Inference in models that make independence assumptions not present in the underlying generative process for complete data may be biased unless more stringent assumptions on the missing data process hold.

Chapter 4

Unsupervised Learning With Random Missing Data

In this chapter we present unsupervised learning methods under the missing at random assumption. We discuss several types of probabilistic mixture models for categorical data including finite mixture models and Dirichlet process mixture models. We present a collapsed Gibbs sampler for the Multinomial/Dirichlet process mixture model with random missing data. We review linear Gaussian latent variable models including probabilistic principal components analysis and factor analysis. We present expectation maximization algorithms for single factor analysis models and mixtures of factor analysis models with randomly missing data.

The models and methods presented in this chapter provide a starting point for the development of novel models and methods for both unsupervised learning with non-random missing data, and classification with missing features. Experimental results using the methods described in this chapter are presented in Chapters 5 and 6.

4.1 Finite Mixture Models

A Bayesian network representation of the finite mixture model is given in figure 4.1. We assume that the data random variables $\mathbf{X}_n$ are vectors of length D that are subject to random missing data. We assume that there are K mixture components. The random variables Z_n are mixture component indicator variables. They indicate which mixture component is associated with each data case, and take values on the discrete set $\{1, ..., K\}$. In practice the random variables Z_n are not observed, and are referred to as latent variables. The parameters of the mixture component distributions are denoted by β_k. The mixing proportions θ_k give the prior probability of observing a data case from each of the K mixture components. The mixing proportions satisfy the constraints $\sum_k \theta_k = 1$, and $\theta_k > 0 \quad \forall k$.

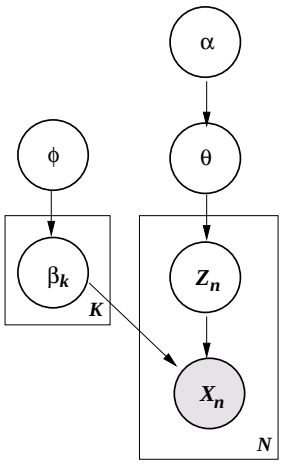


Figure 4.1: Bayesian network for the finite mixture model. θ_k gives the prior probability that a data case belongs to mixture component k. z_n is a latent variable indicating which mixture component data case n belongs to. β_k are the mixture component distribution parameters. α are the parameters of the prior distribution on θ, and ϕ are the parameters of the prior distribution on β_k.

We denote the prior distribution on the mixing proportions θ with hyperparameters α by $P(\theta|\alpha)$. We denote the prior distribution on the mixture component distributions β_k with hyperparameters ϕ by $P(\beta_k|\phi)$. A conjugate prior is often used for the mixture proportion parameters [59, p. 3]. The conjugate prior for the multinomial distribution is the Dirichlet distribution defined in equation 4.1.1. The parameters α_k simultaneously give the scale and location of the Dirichlet distribution. The scale or concentration of the distribution is given by $\sum_k \alpha_k$. The location in the simplex is equal to the expectation of θ, as given in equation 4.1.2.

$$\mathcal{D}(\theta|\alpha) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k \theta_k^{\alpha_k - 1} \tag{4.1.1}$$

$$E[\theta_k|\alpha] = \frac{\alpha_k}{\sum_{k=1}^{K} \alpha_k} \tag{4.1.2}$$

In a multinomial mixture model the component distributions are a product of independent multinomial distributions. A conjugate Dirichlet prior is usually also assumed for the multinomial component distributions [59, p. 3]. This allows for tractable maximum a posteriori learning, as well as tractable Bayesian inference. We give the probability model for the Bayesian multinomial mixture model with independent Dirichlet prior distributions in equations 4.1.3 to 4.1.6.

$$P(\theta|\alpha) = \mathcal{D}(\theta|\alpha) \tag{4.1.3}$$

$$P(\beta_{dk}|\phi_{dk}) = \mathcal{D}(\beta_{dk}|\phi_{dk}) \tag{4.1.4}$$

$$P(Z_n = k|\theta) = \theta_k \tag{4.1.5}$$

$$P(\mathbf{X}_n = \mathbf{x}_n|Z_n = k, \beta) = \prod_{d=1}^{D}\prod_{v=1}^{V} \beta_{vdk}^{[x_{dn}=v]} \tag{4.1.6}$$

We give the posterior probability of the parameters β and θ given the prior and a set of N data cases in Equation 4.1.7. We assume the missing at random condition holds.

$$\begin{aligned} P(\beta, \theta|\{\mathbf{x}_n, \mathbf{r}_n\}_{1:N}, \alpha, \phi) \propto & \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_k^{\alpha_k-1} \cdot \prod_{d=1}^{D}\frac{\Gamma(\sum_{v=1}^{V}\phi_{vdk})}{\prod_{v=1}^{V}\Gamma(\phi_{vdk})}\prod_{v=1}^{V}\beta_{vdk}^{\phi_{vdk}-1} \\ & \cdot \prod_{n=1}^{N}\sum_{k=1}^{K}\theta_k\prod_{d=1}^{D}\prod_{v=1}^{V}\beta_{vdk}^{[r_{dn}=1][x_{dn}=v]} \end{aligned} \tag{4.1.7}$$

4.1.1 Maximum A Posteriori Estimation

The maximum a posteriori principle states that we should select the parameters θ and β with maximum posterior probability. The optimal parameters of the Bayesian multinomial mixture model can not be found analytically, but an efficient expectation maximization algorithm can be derived assuming missing data is missing at random. We give the derivation in equations 4.1.8 to 4.1.11. We first find the posterior distribution over the latent variable z_n for each data case n. Note that the posterior only depends on the observed data dimensions as specified by $[r_{dn} = 1]$.

$$P(Z_n = k|\mathbf{x}_n, \mathbf{r}_n, \beta, \theta) = q_n(k) = \frac{\theta_k\prod_{d=1}^{D}\prod_{v=1}^{V}\beta_{vdk}^{[r_{dn}=1][x_{dn}=v]}}{\sum_{k'=1}^{K}\theta_{k'}\prod_{d=1}^{D}\prod_{v=1}^{V}\beta_{vdk'}^{[r_{dn}=1][x_{dn}=v]}} \tag{4.1.8}$$

Next we form the expected complete log posterior function $E[\log\mathcal{P}(\theta, \beta|\{\mathbf{x}_n, \mathbf{r}_n\}_{1:N}, \alpha, \phi)]$ using the posterior probability of the latent variables, and the complete log posterior of the parameters. The expectation is taken with respect to the latent mixture indicators only. The missing data is analytically integrated out of the complete log posterior under the MAR assumption.

$$
\begin{aligned}
E[\log \mathcal{P}(\theta, \beta | \{\mathbf{x}_n, \mathbf{r}_n\}_{n=1:N}, \alpha, \phi)] &= \log \Gamma(\sum_{k=1}^{K} \alpha_k) - \sum_{k=1}^{K} \log \Gamma(\alpha_k) + \sum_{k=1}^{K} (\alpha_k - 1) \log \theta_k \\
&+ \sum_{k=1}^{K} \sum_{d=1}^{D} \log \Gamma(\sum_{v=1}^{V} \phi_{vdk}) - \sum_{v=1}^{V} \log \Gamma(\phi_{vdk}) + \sum_{v=1}^{V} (\phi_{vdk} - 1) \log \beta_{vdk} \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} q_n(k) \left(\log \theta_k + \sum_{d=1}^{D} \sum_{v=1}^{V} [r_{dn} = 1][x_{dn} = v] \log \beta_{vdk} \right)
\end{aligned} \tag{4.1.9}
$$

Finally, we find the partial derivatives of the expected complete log posterior with respect to the multinomial parameters θ, and β. We solve the resulting gradient equations using Lagrange multipliers to enforce the normalization constraint on the multinomial parameters. The update for θ is given in Equation 4.1.10.

$$
\begin{aligned}
\frac{\partial E[\log \mathcal{P}]}{\partial \theta_k} &= \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{\theta_k} - \lambda = 0 \\
\theta_k \lambda &= \alpha_k - 1 + \sum_{n=1}^{N} q_n(k) \\
\sum_{k=1}^{K} \theta_k \lambda &= \sum_{k=1}^{K} (\alpha_k - 1) + \sum_{n=1}^{N} q_n(k) \\
\lambda &= N - K + \sum_{k=1}^{K} \alpha_k \\
\theta_k &= \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{N - K + \sum_{k=1}^{K} \alpha_k}
\end{aligned} \tag{4.1.10}
$$

The update for β_{dk} is given in Equation 4.1.11.

$$
\begin{aligned}
\frac{\partial E[\log \mathcal{P}]}{\partial \beta_{vdk}} &= \frac{\phi_{vnk} - 1 + \sum_{n=1}^{N} q_n(k)[r_{dn} = 1][x_{dn} = v]}{\beta_{vdk}} - \lambda = 0 \\
\beta_{vdk} \lambda &= \phi_{vnk} - 1 + \sum_{n=1}^{N} q_n(k)[r_{dn} = 1][x_{dn} = v] \\
\sum_{v=1}^{V} \beta_{vdk} \lambda &= \sum_{v=1}^{V} (\phi_{vnk} - 1) + \sum_{n=1}^{N} q_n(k)[r_{dn} = 1][x_{dn} = v] \\
\lambda &= \sum_{n=1}^{N} q_n(k)[r_{dn} = 1] - V + \sum_{v=1}^{V} \phi_{vdk}
\end{aligned}
$$

$$\beta_{vdk} = \frac{\phi_{vdk} - 1 + \sum_{n=1}^{N} q_n(k)[r_{dn} = 1][x_{dn} = v]}{\sum_{n=1}^{N} q_n(k)[r_{dn} = 1] - V + \sum_{v=1}^{V} \phi_{vdk}} \tag{4.1.11}$$

We give the expectation maximization algorithm for the Bayesian multinomial mixture model under the missing at random assumption in Equations 4.1.12 to 4.1.14.

$$\text{E-Step: } q_n(k) \leftarrow \frac{\theta_k \prod_{d=1}^{D} \prod_{v=1}^{V} \beta_{vdk}^{[r_{dn}=1][x_{dn}=v]}}{\sum_{k'=1}^{K} \theta_{k'} \prod_{d=1}^{D} \prod_{v=1}^{V} \beta_{vdk'}^{[r_{dn}=1][x_{dn}=v]}} \tag{4.1.12}$$

$$\text{M-Step: } \theta_k \leftarrow \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{N - K + \sum_{k=1}^{K} \alpha_k} \tag{4.1.13}$$

$$\beta_{vdk} \leftarrow \frac{\phi_{vdk} - 1 + \sum_{n=1}^{N} q_n(k)[r_{dn} = 1][x_{dn} = v]}{\sum_{n=1}^{N} q_n(k)[r_{dn} = 1] - V + \sum_{v=1}^{V} \phi_{vdk}} \tag{4.1.14}$$

Efficient Markov chain Monte Carlo inference methods can also be derived for the finite mixture model [59, p. 3]. However, with very little additional computational effort these same methods can be used to perform inference in a mixture model with an unbounded number of mixture components, as we describe in Section 4.2.

4.1.2 Predictive Distribution

Once the model parameters β and θ are learned, the finite mixture model can be used to make predictions for the unobserved values in a data case. The posterior predictive distribution is given in Equation 4.1.15.

$$\begin{aligned} P(x_{dn} = v|\mathbf{x}_n, \mathbf{r}_n, \beta, \theta) &= \sum_{k=1}^{K} P(x_{dn} = v|z_n = k, \beta) P(z_n = k|\mathbf{x}_n, \mathbf{r}_n, \beta, \theta) \\ &= \sum_{k=1}^{K} \beta_{vdk} \frac{\theta_k \prod_{d=1}^{D} \prod_{v=1}^{V} \beta_{vdk}^{[r_{dn}=1][x_{dn}=v]}}{\sum_{k'=1}^{K} \theta_{k'} \prod_{d=1}^{D} \prod_{v=1}^{V} \beta_{vdk'}^{[r_{dn}=1][x_{dn}=v]}} \end{aligned} \tag{4.1.15}$$

4.2 Dirichlet Process Mixture Models

The Dirichlet process mixture model is represented as a Bayesian network in figure 4.2. To sample a complete data set from a DP mixture model we begin by sampling a random distribution ϕ from $\mathcal{DP}(\alpha\phi_0)$. For each data case n we sample a value β_n from ϕ, and a value of $\mathbf{x}_n$ from $P(\mathbf{X}|\beta_n)$. Unlike the finite mixture model, the number of mixture components is not fixed a priori. As more data is sampled from the model, an unlimited number of mixture components may appear. Data cases n and n' are said to belong to the same mixture component if $\beta_n = \beta_{n'}$. We give the probability model for the DP mixture in equations 4.2.1 to 4.2.3.

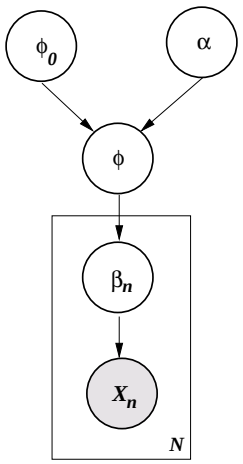


Figure 4.2: Graphical representation of the Dirichlet process mixture model. α is the concentration of the Dirichlet Process prior while ϕ_0 is the base measure. ϕ is a random distribution sampled from the Dirichlet Process prior. β_n are the mixture component parameters associated with data case n. The mixture component parameters are sampled from the random distribution ϕ, which is discrete with probability one. This leads to clustering of the data $\mathbf{x}_n$.

$$P(\phi|\phi_0, \alpha) = \mathcal{DP}(\alpha\phi_0) \tag{4.2.1}$$

$$P(\beta_n|\phi) = \phi(\beta_n) \tag{4.2.2}$$

$$P(\mathbf{X}_n = \mathbf{x}_n|\beta) = P(\mathbf{x}_n|\beta_n) \tag{4.2.3}$$

In this section we first review the properties of the Dirichlet process, and its use as a prior distribution in Bayesian mixture models. As with the finite mixture model, we assume that the data variables $\mathbf{X}_n$ are subject to random missing data, and derive inference algorithms under the missing at random condition. We present both conjugate and collapsed Gibbs samplers for the multinomial Dirichlet Process mixture model with random missing data.

4.2.1 Properties of The Dirichlet Process

We begin with Ferguson's definition of the Dirichlet process: Let S be a continuous space and let A be a σ-field of subsets of S. Let f be a non-null, finite measure on (S, A). A process P indexed by elements A_i of A is a Dirichlet process on (S, A) with parameter f if for any $k = 1, 2, ...$ and any measurable partition $(A_1, A_2, ..., A_k)$ of S, the random vector $(P(A_1), ..., P(A_k))$ has a Dirichlet distribution with parameters $(f(A_1), ..., f(A_k))$. That is to say, $(P(A_1), ..., P(A_k)) \sim Dirichlet(f(A_1), ..., f(A_k))$ [18, p. 214].

Interpreting the definition requires several basic concepts from measure theory which we briefly review. A σ-field on a set S is a set A of subsets of S that is closed under the operations

of complementation, and countably infinite unions. A measure f is a function mapping any element of the σ-field A to the positive real line which satisfies $f(\emptyset) = 0$ and for every collection $A_1, A_2, ...$ of non-overlapping elements of A, $f(\cup_{i=0}^{\infty} A_i) = \sum_{i=0}^{\infty} f(A_i)$.

Suppose that β is a continuous random variable, and the space of β is S. Normally the base measure f consists of the product of a scalar variable α, and a probability distribution ϕ_0 on β [60]. The definition states that a random probability distribution ϕ on β is distributed according to the Dirichlet process $\mathcal{DP}(\alpha\phi_0)$ if for any partition $(A_1, A_2, ..., A_k)$ of S, $(\phi(\beta \in A_1), ..., \phi(\beta \in A_K)) \sim Dirichlet(\alpha\phi_0(A_1), ..., \alpha\phi_0(A_K))$.

In the case of a Dirichlet process mixture model, we assume that the observed data variable $\mathbf{X}$ is generated according to a parametric distribution $P(\mathbf{X}|\beta)$. The parameter β is in turn distributed according to a random distribution $P(\beta|\phi) = \phi(\beta)$. The random distribution ϕ is drawn from a Dirichlet process prior $\mathcal{DP}(\alpha\phi_0)$.

Ferguson's definition of the Dirichlet process is useful for establishing some important properties of the Dirichlet process. For example, suppose that we observe a set of draws $\beta_1, ..., \beta_{N-1}$ from ϕ. Let $A_1, ..., A_K$ be any partition of the space S of β for any K. Let δ_{β_n} be a distribution concentrated at location β_n. The posterior distribution of ϕ given $\beta_1, ..., \beta_{N-1}$ is shown in equation 4.2.4, as derived by Ferguson [18, p. 217]. From the posterior distribution on ϕ we can find the posterior distribution on β_{N+1} given the values of the previous draws as seen in equation 4.2.5 [60, p. 3].

$$P(\phi|\alpha, \phi_0, \beta_1, ..., \beta_{N-1}) = \mathcal{DP}(\alpha\phi_0 + \sum_{n=1}^{N-1} \delta_{\beta_n}) \tag{4.2.4}$$

$$P(\beta_N|\alpha, \phi_0, \beta_1, ..., \beta_{N-1}) = \frac{\alpha\phi_0(\beta_N)}{N-1+\alpha} + \frac{\sum_{n=1}^{N} \delta_{\beta_n}(\beta_N)}{N-1+\alpha} \tag{4.2.5}$$

This last equation shows that when we draw a series of values for β from the Dirichlet process prior, we may either draw a value equal to one of the previous draws, or draw a new value from the base distribution ϕ_0. Further, it can be shown that a draw ϕ from the Dirichlet process is discrete with probability one [18, p. 219]. This means that ϕ consists of a countably infinite collection of point masses in the space of β, and there is a non-zero probability of drawing the same value β multiple times. In terms of the Dirichlet process mixture, this property implies that a series of data points $\mathbf{x}_n$ drawn from the DP mixture will tend to form clusters.

Suppose we introduce a set of indicator variables z_n along with β_n and $\mathbf{x}_n$. We begin by setting $z_1 = 0$. During sampling we set $z_n = z_j$ if $\beta_n = \beta_j$, $1 < j < n$. If β_n is not equal to any of $\beta_1, .., \beta_{n-1}$, we set z_n to the next available value $1 + \max(z_1, ..., z_{n-1})$. The sampling process on the indicator variables induced by the Dirichlet process is known as the Chinese Restaurant

Process or CRP [2, p. 92]. The probability distribution of z_N given the $N-1$ previous samples can be obtained from equation 4.2.5.

$$\text{If } z_j = z_N = k \text{ for some } j \neq N \; P(z_N = k|\alpha, z_1, ..., z_{N-1}) = \frac{\sum_{i=1}^{N-1}[z_i = k]}{N-1+\alpha} \tag{4.2.6}$$

$$P(z_N \neq z_j \;\; \forall \;\; j \neq N \;\; |\alpha, z_1, ..., z_{N-1}) = \frac{\alpha}{N-1+\alpha} \tag{4.2.7}$$

Using the CRP, we can draw a sample of N data points from the Dirichlet process prior in two stages. First we sample a partition of the data points from the CRP as represented by the indicator variables z_n. Let the partition consists of K clusters. Next, for each cluster k, a single parameter vector β_k is sampled from the base distribution ϕ_0 for that cluster.

We can also use the definition of the CRP, and its relationship to the Dirichlet process, to derive the joint distribution of a sample of indicator variables $z_1, ..., z_N$, as well as a sample of parameter values $\beta_1, ..., \beta_N$. To simplify the notation suppose again that the partition has K blocks or clusters. Let the unique values of z be $z_{u(1)}, ..., z_{u(K)}$, and the corresponding unique values of β be $\beta_{u(1)}, ..., \beta_{u(K)}$. We introduce count variables c_k to indicate the number of z_n equal to $z_{u(k)}$.

$$\begin{aligned} P(z_1, ..., z_N|\alpha) &= P(z_1|\alpha) \cdots P(z_N|\alpha, z_1, ..., z_{N-1}) \\ &= \frac{\prod_{k=1}^{K} \alpha \prod_{i=1}^{c_k - 1} i}{\prod_{n=1}^{N} n - 1 + \alpha} = \frac{\alpha^K \Gamma(\alpha) \prod_{k=1}^{K} \Gamma(c_k)}{\Gamma(N+\alpha)} \end{aligned} \tag{4.2.8}$$

$$\begin{aligned} P(\beta_1, ..., \beta_N|\alpha, \phi_0) &= P(\beta_1|\alpha, \phi_0) \cdots P(\beta_N|\alpha, \phi_0, \beta_1, ..., \beta_{N-1}) \\ &= \frac{\alpha^K \Gamma(\alpha) \prod_{k=1}^{K} \Gamma(c_k)}{\Gamma(N+\alpha)} \prod_{k=1}^{K} \phi_0(\beta_{u(k)}) \end{aligned} \tag{4.2.9}$$

It is clear from the form of equations 4.2.8 and 4.2.9 that the joint probability of the z's and the joint probability of the β's are both independent of their ordering. This property is called exchangeability. Under exchangeability we can assume that any element in the sampled sequence is the last element, and thus use 4.2.6 or 4.2.5 to compute the conditional probabilities needed for Gibbs sampling [2, p. 6].

4.2.2 Bayesian Inference and the Conjugate Gibbs Sampler

As we have seen, the Dirichlet Process provides a very flexible prior distribution for mixture models. Unlike finite mixture models, the Dirichlet process mixture model can produce an unlimited number of clusters as the amount of data increases. Conditioned on a fixed sample

of data, the Dirichlet process mixture model is able to automatically adapt the number of clusters. The Dirichlet process mixture model can better reflect uncertainty about the number of components underlying the data, and eliminates the need to perform an explicit search for a single, optimal number of components. In this section we derive a Gibbs sampler for the Dirichlet process mixture model assuming conjugate priors [60, p. 4].

To complete our definition of the multinomial Dirichlet Process mixture model, we must specify the base distribution ϕ_0. A conjugate Gibbs sampler can be derived using an independent product of Dirichlet distributions as the base distribution. Recall that this base distribution is the same as the conjugate prior on mixture component parameters used in the finite multinomial mixture model.

$$P(\beta|\phi_0) = \prod_{d=1}^{D} \mathcal{D}(\beta_d|\phi_{d0}) \tag{4.2.10}$$

The state of the Gibbs sampler consists of the mixture component indicators $z_1, ..., z_N$ and the parameters of the occupied mixture component distributions $\beta_1, ..., \beta_K$. Of course, the mixture contains an infinite number of components, but we do not explicitly represent components that have no data cases assigned to them. The number of occupied components K changes as new components are occupied, and old components are abandoned.

We begin by deriving the posterior probability of the mixture indicator z_n, given the remaining indicator variables z_{-n}, and the observed data. This conditional distribution can be derived from the Chinese Restaurant Process and the observed data likelihood obtained by analytically summing over missing data under the missing at random assumption.

$$\begin{aligned}
\text{If } z_j = z_n = k \text{ for some } j \neq n \; P(z_n = k|z_{-n}, \mathbf{x}_n, \beta, \alpha, \phi_0) & \\
&\propto P(z_n = k|z_{-n}, \alpha) P(\mathbf{x}_n|r_n, \beta_k) \\
&\propto \frac{\sum_{i\neq n}^{N} [z_i = k]}{N - 1 + \alpha} \prod_{d=1}^{D} \prod_{v=1}^{V} \beta_{vdk}^{[r_{dn}=1][x_{dn}=v]} \qquad (4.2.11) \\
P(z_n \neq z_j \;\; \forall \;\; j \neq n|z_{-n}, \mathbf{x}_n, \alpha, \phi_0) &\propto P(z_n \neq z_j \forall \;\; j \neq n|z_{-n}, \alpha) P(\mathbf{x}_n|\mathbf{r}_n, \phi_0) \\
&\propto \frac{\alpha}{N - 1 + \alpha} \int P(\mathbf{x}_n|\mathbf{r}_n, \beta) P(\beta|\phi_0) d\beta \\
&\propto \frac{\alpha}{N - 1 + \alpha} \prod_{d=1}^{D} \prod_{v=1}^{V} \left(\frac{\phi_{vd0}}{\sum_{v=1}^{V} \phi_{vd0}} \right)^{[r_{dn}=1][x_{dn}=v]} \qquad (4.2.12)
\end{aligned}$$

The probabilities in equations 4.2.11 and 4.2.12 have very intuitive interpretations. Equation 4.2.11 simply states that z_n is assigned to a component k, and there exists another data case j different from n that is also assigned to component k. The probability of this event depends

on the number of data cases assigned to component k, and the probability of the observed data $\mathbf{x}_n^o$ under component k.

The event expressed in equation 4.2.12 as $(z_n \neq z_j \ \forall \ j \neq n)$ is the event that data case n is assigned to a previously unoccupied component. The probability of this event depends only on the concentration parameter α, and the marginal probability of $\mathbf{x}_n^o$ under the base distribution. Increasing the concentration parameter increases the prior probability of forming new clusters. The marginal probability of assigning data case n to an unoccupied component is given by the integral of $P(\mathbf{x}_n^o|\beta)$ with respect to the base distribution. This integral is tractable when the base distribution is conjugate to the parameter β. If a new component is created while updating the z_n, it is randomly initialized by sampling from the base distribution.

Once all of the data cases have been assigned to mixture components, the update for the occupied component distribution parameters is identical to the finite mixture case. Here we use the count variables $c_{vdk} = \sum_{n=1}^{N}[z_n = k][r_{dn} = 1][x_{dn} = v]$ to simplify the notation.

$$\begin{aligned} P(\beta_{vdk}|\mathbf{z}, \mathbf{x}, \phi_0) &\propto \prod_{n=1}^{N} P(x_{dn}|\beta_{dk})P(\beta_{dk}|\phi_{d0}) \\ &\propto \prod_{n=1}^{N}\prod_{v=1}^{V} \beta_{vdk}^{[r_{dn}=1][x_{dn}=v][z_n=k]} \cdot \prod_{v=1}^{V} \beta_{vdk}^{\phi_{vd0}-1} \\ &= \mathcal{D}(c_{1dk} + \phi_{1d0}, ..., c_{Vdk} + \phi_{Vd0}) \end{aligned} \tag{4.2.13}$$

The conjugate Gibbs sampler for the multinomial/Dirichlet process mixture model alternates between sequentially assigning each data case to a mixture component, creating new components as needed, and updating the parameters associated with each occupied mixture component.

4.2.3 Bayesian Inference and the Collapsed Gibbs Sampler

In this section we derive a collapsed Gibbs sampler for the multinomial Dirichlet Process mixture model under the missing at random assumption. To form the collapsed Gibbs sampler we analytically integrate the β parameters out of the model, leaving only the mixture indicators [59, p. 4]. This integration is tractable in the conjugate multinomial/Dirichlet setting. The primary advantage of the collapsed Gibbs sampler is that the size of the state of the underlying Markov chain is greatly reduced. We give the Gibbs updates below. For convenience we introduce the count variables $c_k^{-n} = \sum_{i\neq n}^{N}[z_i = k]$ and $c_{vdk}^{-n} = \sum_{i\neq n}^{N}[z_i = k][r_{di} = 1][x_{di} = v]$. Note that the count variables must be recomputed after each Gibbs update.

$$\text{If } z_j = z_n = k \text{ for some } j \neq n \; P(z_n = k | z_{-n}, \mathbf{x}, \alpha, \phi_0) \propto \frac{c_k^{-n}}{N-1+\alpha} \prod_{d=1}^{D} \prod_{v=1}^{V} \left(\frac{c_{vdk}^{-n} + \phi_{vd0}}{\sum_{v=1}^{V} c_{vdk}^{-n} + \phi_{vd0}} \right)^{[r_{dn}=1][x_{dn}=v]} \tag{4.2.14}$$

$$P(z_n \neq z_j \;\; \forall j \neq n \;\; | z_{-n}, x_n, \alpha, \phi_0) \propto \frac{\alpha}{N-1+\alpha} \prod_{d=1}^{D} \prod_{v=1}^{V} \left(\frac{\phi_{vd0}}{\sum_{v=1}^{V} \phi_{vd0}} \right)^{[r_{dn}=1][x_{dn}=v]} \tag{4.2.15}$$

To run the collapsed Gibbs sampler we simply iterate through the data cases, updating each mixture indicator in turn. A careful implementation where the count variable are cached and updated instead of recomputed from scratch can lead to significant computational savings.

4.2.4 Predictive Distribution and the Conjugate Gibbs Sampler

Running the conjugate Gibbs sampler produces a set of samples z_n^s, β_k^s for each data case n and each occupied mixture component k. Denote the number of occupied mixture components in sample s by K^s. The posterior predictive distribution for data case n is obtained by averaging the predictions made by the mixture component data case n belongs to in each sample s. Note that we assume the data dimension d we make a prediction for is not one of the observed data dimensions.

$$P(x_{dn} = v | \{\mathbf{x}_i^o, \mathbf{r}_i\}_{i=1:N}, \phi_0, \alpha) = \frac{1}{S} \sum_{s=1}^{S} \sum_{k=1}^{K^s} [z_n^s = k] \beta_{vdk}^s \tag{4.2.16}$$

If a data case is not included in the Dirichlet process mixture simulation, it is still possible to make predictions for that data case. This is a useful procedure if we need to make predictions for additional data cases and wish to avoid re-running a complete simulation. One method to accomplish this is to convert each sample from the Dirichlet process mixture into a finite mixture model with parameters $\hat{\theta}$ and $\hat{\beta}$ as shown below. This procedure is closely related to the finite mixture model approximations to the Dirichlet Process mixture model studied by Ishwaran and Zarepour [41].

$$\hat{\theta}_k^s = \begin{cases} \frac{\sum_{n=1}^{N} [z_n = k]}{N+\alpha} & \ldots \; k \leq K^s \\ \frac{\alpha}{N+\alpha} & \ldots \; k = K^s + 1 \end{cases} \qquad \hat{\beta}_{vdk}^s = \begin{cases} \beta_{vdk}^s & \ldots \; k \leq K^s \\ \frac{\phi_{vd0}}{\sum_{v=1}^{V} \phi_{vd0}} & \ldots \; k = K^s + 1 \end{cases}$$

To compute the predictive distribution for a novel data case $\mathbf{x}_*$, we average the predictive distribution under each of the finite mixture models as seen in Equation 4.2.17. The predictive distribution within a single finite mixture model is computed according to Equation 4.1.15.

$$P(x_{d*} = v|\mathbf{x}_*^o, \mathbf{r}_*, \beta, \theta) = \frac{1}{S}\sum_{s=1}^{S}\sum_{k=1}^{K^s+1}\hat{\beta}_{vdk}^s \frac{\hat{\theta}_k \prod_{d=1}^{D}\prod_{v=1}^{V}\hat{\beta}_{vdk}^{[r_{d*}=1][x_{d*}=v]}}{\sum_{k'=1}^{K}\hat{\theta}_{k'}\prod_{d=1}^{D}\prod_{v=1}^{V}\hat{\beta}_{vdk'}^{[r_{d*}=1][x_{d*}=v]}} \tag{4.2.17}$$

If our original simulation was based on a large number of data cases, the present approximation will be quite close to the predictive distribution obtain by including the novel data point in the simulation. The reason is that a single data point can not have a large effect on either the partitioning of the data points, or the sampling of the parameter values. If we use this scheme to independently make predictions for a large number of novel data points, the results may differ significantly from including all such novel data points in the simulation.

4.2.5 Predictive Distribution and the Collapsed Gibbs Sampler

Computing the predictive distribution for a training case using samples from the collapsed Gibbs sampler requires additional steps. This is because the state of the collapsed Gibbs sampler consists only of mixture indicator variables $\mathbf{z}_n$. The posterior distribution on β_{dk} for an occupied cluster with index k in sample s is Dirichlet, and its parameters are trivial to compute given the training data and mixture indicators. We again use count variables $c_{vdk} = \sum_{n=1}^{N}[z_n = k][r_{dn} = 1][x_{dn} = v]$.

$$P(\beta_{dk}|\{z_n^s, \mathbf{x}_n, \mathbf{r}_n\}_{n=1:N}, \phi_0) = \mathcal{D}(\phi_{10} + c_{1dk}, ..., \phi_{V0} + c_{Vdk}) \tag{4.2.18}$$

A particular value β_k^s could be sampled from this posterior distribution in order to compute the predictive distribution. However, this introduces unnecessary variance. A better option is to set β_k^s to the mean of the posterior. This is a form of Rao-Blackwellization in the sense of Gelfand and Smith [24].

$$\beta_{vdk}^s = \frac{\phi_{v0} + c_{vdk}}{\sum_{v=1}^{V}\phi_{v0} + c_{vdk}} \tag{4.2.19}$$

The predictive distribution for training cases can now be calculated as seen in Equation 4.2.20. The predictive distribution for novel data cases can be found using the same approximation scheme as described for the conjugate Gibbs sampler in Equation 4.2.17.

$$P(x_{dn} = v|\{\mathbf{x}_i^o, \mathbf{r}_i\}_{i=1:N}, \phi_0, \alpha) = \frac{1}{S}\sum_{s=1}^{S}\sum_{k=1}^{K^s}[z_n^s = k]\beta_{vdk}^s \tag{4.2.20}$$

4.3 Factor Analysis and Probabilistic Principal Components Analysis

The generative process underlying factor analysis asserts that the observed data cases $\mathbf{x}_1, ..., \mathbf{x}_N$ are generated using a two step process. First, a continuous Q-dimensional latent vector $\mathbf{t}_n$ is sampled for each data case n from a zero-mean Gaussian with unit variance. Next, $\mathbf{x}_n$ is sampled from a D-dimensional Gaussian distribution with mean $\mu + \Lambda\mathbf{t}_n$, and covariance matrix Ψ [73, p. 612]. Λ is a $D \times Q$ matrix referred to as the factor loading matrix. Each dimension q of the latent space is referred to as a factor.

The main feature of factor analysis is that the covariance matrix Ψ is constrained to be diagonal. This means that conditioned on the latent vector $\mathbf{t}_n$, the dimensions of $\mathbf{x}_n$ are independent. We summarize the factor analysis model below.

$$P(\mathbf{t}_n) = \frac{1}{(2\pi)^{Q/2}} \exp(-\frac{1}{2}\mathbf{t}_n^T\mathbf{t}_n) \tag{4.3.1}$$

$$P(\mathbf{x}_n|\mathbf{t}_n, \Lambda, \Psi, \mu) = \frac{1}{|2\pi\Psi|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)^T\Psi^{-1}(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)) \tag{4.3.2}$$

Probabilistic principal components analysis is a linear Gaussian generative model for continuous data vectors [73]. The model asserts that each D dimensional observed data vector $\mathbf{x}_n$ is generated by first sampling a Q dimensional vector $\mathbf{t}_n$ from a unit-variance, zero-mean Gaussian distribution. The observed data vector $\mathbf{x}_n$ is then sampled from a Gaussian distribution with mean $\mu + \Lambda\mathbf{t}_n$ and covariance matrix $\sigma^2 I$. We summarize the probabilistic model below.

$$P(\mathbf{t}_n) = \frac{1}{(2\pi)^{Q/2}} \exp(-\frac{1}{2}\mathbf{t}_n^T\mathbf{t}_n) \tag{4.3.3}$$

$$P(\mathbf{x}_n|\mathbf{t}_n, \Lambda, \sigma, \mu) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp(-\frac{1}{2\sigma^2}(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)^T(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)) \tag{4.3.4}$$

Probabilistic PCA is thus subsumed by factor analysis since the spherical covariance matrix $\sigma^2 I$ is a special case of the diagonal covariance matrix used in factor analysis. Fitting the two models will, of course, lead to different latent factors, and different factor loading matrices. It is insightful to compare the two models based on what data transformations each model is invariant to. Probabilistic PCA is invariant to rotations of the observed data since the covariance matrix is assumed to be spherical. By contrast, factor analysis is invariant to independent scaling along different input dimensions since the covariance matrix is diagonal [73, p. 615].

Probabilistic principal components analysis is also closely related to classical principal com-

ponents analysis. Classical PCA finds a linear projection from D dimensions to Q dimensions that retains the maximum amount of variance [43]. Given a set of N data vectors $\mathbf{x}_1, ..., \mathbf{x}_N$ in D dimensions, the goal of PCA is to obtain a set of vectors $\mathbf{t}_1, ..., \mathbf{t}_N$ in Q dimensions satisfying the relationship $\mathbf{t}_n = W(\mathbf{x}_n - \bar{\mathbf{x}})$, where $\bar{\mathbf{x}}$ is the mean of the data vectors, and $\mathbf{t}_1, ..., \mathbf{t}_N$ retain the maximum amount of variance. The matrix W is called the projection matrix, and is of size $Q \times D$. The rows of W are constrained to be orthonormal. It can be shown that the matrix W that yields the desired projection is given by the Q leading eigenvectors of the empirical covariance matrix [73, p. 611].

Probabilistic PCA and classical PCA are related by the fact that classical PCA is the zero-noise limit of probabilistic PCA. As a result, the optimal probabilistic PCA factor loading matrix Λ is the transpose of classical PCA's optimal W matrix, and can be found using eigen-decomposition of the empirical covariance matrix if there is no missing data. The optimal factor loading matrix in factor analysis can not be found in this way [73, p. 619]. For the remainder of this section we develop the more general case of a diagonal covariance matrix Ψ.

4.3.1 Joint, Conditional, and Marginal Distributions

The joint probability of $\mathbf{x}_n$ and $\mathbf{t}_n$ is Gaussian. This means that all marginal and conditional distributions are also Gaussian, and can be obtained from the joint distribution using standard Gaussian marginalization and conditioning formulas. The parameters of the joint distribution can be deduced from the Gaussian conditioning formula that computes $P(\mathbf{x}_n|\mathbf{t}_n)$ from $P(\mathbf{x}_n, \mathbf{t}_n)$.

$$P\left([\mathbf{x}_n, \mathbf{t}_n]^T|\mu, \Lambda, \Psi\right) = \mathcal{N}\left(\begin{bmatrix}\mu_x \\ \mu_t\end{bmatrix}, \begin{bmatrix}\Sigma_{xx} & \Sigma_{xt} \\ \Sigma_{tx} & \Sigma_{tt}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mu \\ 0\end{bmatrix}, \begin{bmatrix}\Psi + \Lambda\Lambda^T & \Lambda \\ \Lambda^T & I\end{bmatrix}\right) \tag{4.3.5}$$

Both probabilistic PCA and factor analysis are similar to classical PCA in that they can also be used for dimensionality reduction. Dimensionality reduction is achieved by inferring the most likely latent vector $\mathbf{t}_n$ for each observed data vector $\mathbf{x}_n$ using Bayes rule. However, unlike classical PCA, probabilistic PCA and factor analysis can infer latent factors vectors given incomplete input vectors, assuming the missing at random condition holds. The posterior probability of $\mathbf{t}_n$ given $\mathbf{x}_n$ can be found by applying the Gaussian conditioning and marginalization formulas to Equation 4.3.5.

$$\begin{aligned} P(\mathbf{t}_n|\mathbf{x}_n^o, \mu, \Lambda, \Psi) &= \mathcal{N}(\mu_{t_n|x_n^o}, \Sigma_{t_n|x_n^o}) \\ \mu_{t_n|x_n^o} &= \Lambda^{oT}(\Sigma_{xx}^{oo})^{-1}(\mathbf{x}_n^o - \mu^o) \\ \Sigma_{t_n|x_n^o} &= I - \Lambda^{oT}(\Sigma_{xx}^{oo})^{-1}\Lambda^o \end{aligned} \tag{4.3.6}$$

Probabilistic PCA and factor analysis can also be thought of as providing a structured approximation to the empirical covariance matrix. This view is obtained by integrating over the unobserved latent vectors $\mathbf{t}_n$ to obtain the marginal distribution on $\mathbf{x}_n$. This operation is trivial given the joint distribution of $\mathbf{x}_n$ and $\mathbf{t}_n$.

$$P(\mathbf{x}_n|\Lambda, \sigma, \mu) = \mathcal{N}(\mathbf{x}_n; \mu_x, \Sigma_{xx}) = \mathcal{N}(\mathbf{x}_n; \mu, \Psi + \Lambda\Lambda^T)$$

The estimated covariance matrix $\Sigma_{xx} = \Psi + \Lambda\Lambda^T$ can be used in place of the empirical covariance matrix, and can result in significant savings in both computation time, and the space required to store model parameters. The factor analysis model parameters Ψ, Λ, and μ can be estimated using and expectation maximization algorithm as discussed in the next section.

4.3.2 Maximum Likelihood Estimation

In this section we derive an expectation maximization algorithm for factor analysis and probabilistic principal components analysis assuming the missing at random condition holds. The maximum likelihood principle states that we should select the parameters μ, Λ, Ψ that have the highest likelihood given the observed data. Two distinct maximum likelihood EM algorithms can be obtained depending on whether the missing data is integrated out analytically before forming the expected complete log likelihood, or treated along with the latent vectors $\mathbf{t}_n$ within the EM algorithm itself. Neither approach has a clear advantage. Solving the gradient equations in the M-Step is significantly easier when EM is used to deal with the missing data; however, the E-step is more difficult since joint expectations of missing data values and latent variables are required.

Below, we derive an EM algorithm where the missing data is handled along with the latent factors. This approach is closely related to the handling of missing data when learning a full covariance Gaussian or mixture of full covariance Gaussians [27, 28]. The alternative EM algorithm for factor analysis is given by Canny [11]. We begin by forming the complete log-likelihood.

$$\begin{aligned}\mathcal{L}(\Lambda, \Psi, \mu|\mathbf{x}_n, \mathbf{t}_n) &= \log P(\mathbf{t}_n) + \log P(\mathbf{x}_n|\mathbf{t}_n, \Lambda, \Psi, \mu) \\ &= -\frac{1}{2}\log(|2\pi\Psi|) - \frac{1}{2}(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)^T\Psi^{-1}(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n) \\ &\quad - \frac{Q}{2}\log(2\pi) - \frac{1}{2}\mathbf{t}_n^T\mathbf{t}_n \end{aligned} \tag{4.3.7}$$

The E-step requires the conditional distribution of the unobserved variables $\mathbf{t}_n$ and $\mathbf{x}_n^m$ given the observed variables $\mathbf{x}_n^o$. This distribution is obtained from the joint using the Gaussian conditioning formula as seen below.

$$q_n(\mathbf{t}, \mathbf{x}^m) = P([\mathbf{t}_n, \mathbf{x}_n^m]^T | \mathbf{x}_n^o, \mu, \Lambda, \Psi) = \mathcal{N}\left(\begin{bmatrix} \mu_{t_n|x_n^o} \\ \mu_{x_n^m|x_n^o} \end{bmatrix}, \begin{bmatrix} \Sigma_{t_n|x_n^o} & \Sigma_{t_n x_n^m|x_n^o} \\ \Sigma^T_{t_n x_n^m|x_n^o} & \Sigma_{x_n^m|x_n^o} \end{bmatrix} \right) \tag{4.3.8}$$

$$\begin{aligned} \mu_{t_n|x_n^o} &= \Lambda^{oT}(\Sigma_{xx}^{oo})^{-1}(\mathbf{x}_n^o - \mu^o) & \mu_{x_n^m|x_n^o} &= \mu^m + \Sigma_{xx}^{mo}(\Sigma_{xx}^{oo})^{-1}(\mathbf{x}_n^o - \mu^o) \\ \Sigma_{t_n|x_n^o} &= \Sigma_{tt} - \Sigma_{tx}^o(\Sigma_{xx}^{oo})^{-1}\Sigma_{xt}^o & \Sigma_{x_n^m|x_n^o} &= \Sigma_{xx}^{mm} - \Sigma_{xx}^{mo}(\Sigma_{xx}^{oo})^{-1}\Sigma_{xx}^{om} \\ \Sigma_{t_n x_n^m|x_n^o} &= \Sigma_{tx}^m - \Sigma_{tx}^o(\Sigma_{xx}^{oo})^{-1}\Sigma_{xx}^{om} \end{aligned}$$

We form the expected complete log likelihood function using the posterior distribution $q_n(\mathbf{t}, \mathbf{x}^m)$, and the complete log likelihood function.

$$\begin{aligned} E[\mathcal{L}(\Lambda, \Psi, \mu | \mathbf{x}_n, \mathbf{t}_n, z_n)] &= \sum_{n=1}^{N} \int\int q_n(\mathbf{t}, \mathbf{x}^m)(\log P(\mathbf{t}_n) + \log P(\mathbf{x}_n | \mathbf{t}_n, \Lambda, \Psi, \mu)) d\mathbf{x}_n^m d\mathbf{t}_n \\ &= \sum_{n=1}^{N} -\frac{1}{2}\log(|2\pi\Psi|) - \frac{1}{2}E_{q_n}[(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)^T\Psi^{-1}(\mathbf{x}_n - \mu - \Lambda\mathbf{t}_n)] \\ &\quad - \frac{Q}{2}\log(2\pi) - \frac{1}{2}E_{q_n}[\mathbf{t}_n^T\mathbf{t}_n] \end{aligned} \tag{4.3.9}$$

We derive the maximum likelihood parameter estimates for μ, and Λ, which are identical for both factor analysis and probabilistic PCA. We derive the update for both the diagonal factor analysis covariance matrix Ψ, and the spherical probabilistic PCA covariance matrix $\sigma^2 I$. We assume the missing at random condition holds throughout the derivation. We introduce the notation $A_{d:}$ to refer to the d^{th} row vector of a matrix A. The element-wise definition of $A_{d:}$ is $(A_{d:})_j = A_{dj}$.

$$\begin{aligned} \frac{\partial E[\mathcal{L}(\Lambda, \Psi, \mu | \mathbf{x}_n, \mathbf{t}_n)]}{\partial \mu} &= \sum_{n=1}^{N} -2\Psi^{-1}E_{q_n}[\mathbf{x}_n] + 2\Psi^{-1}\Lambda E_{q_n}[t_n] + 2\Psi^{-1}\mu = 0 \\ \mu &= \frac{1}{N}\sum_{n=1}^{N} E_{q_n}[\mathbf{x}_n] - \Lambda E_{q_n}[t_n] \end{aligned} \tag{4.3.10}$$

$$\frac{\partial E[\mathcal{L}(\Lambda,\sigma,\mu|\mathbf{x}_n,\mathbf{t}_n)]}{\partial \Lambda} = \sum_{n=1}^{N} -2\Psi^{-1}E_{q_n}[(\mathbf{x}_n-\mu)\mathbf{t}_n^T] + 2\Psi^{-1}\Lambda E_{q_n}[\mathbf{t}_n\mathbf{t}_n^t]$$

$$\Lambda\sum_{n=1}^{N} E_{q_n}[\mathbf{t}_n\mathbf{t}_n^t] = \sum_{n=1}^{N} E_{q_n}[\mathbf{x}_n\mathbf{t}_n^T] - \mu E_{q_n}[\mathbf{t}_n^T]$$

$$\Lambda = \left(\sum_{n=1}^{N} E_{q_n}[\mathbf{x}_n\mathbf{t}_n^T] - \mu E_{q_n}[\mathbf{t}_n]^T\right)\left(\sum_{n=1}^{N} E_{q_n}[\mathbf{t}_n\mathbf{t}_n^T]\right)^{-1} \tag{4.3.11}$$

$$\frac{\partial E[\mathcal{L}(\Lambda,\Psi,\mu|\mathbf{x}_n,\mathbf{t}_n)]}{\partial \Psi_{dd}} = \sum_{n=1}^{N} -\frac{1}{2\Psi_{dd}} + \frac{1}{2\Psi_{dd}^2}E_{q_n}[(\mathbf{x}_{dn}-\mu_d-\Lambda_{d:}t_n)^T(\mathbf{x}_{dn}-\mu_d-\Lambda_{d:}t_n)]$$

$$\Psi_{dd} = \frac{1}{N}\sum_{n=1}^{N} E_{q_n}[\mathbf{x}_n\mathbf{x}_n^T]_{dd} + \mu_d^2 + \Lambda_{d:}E_{q_n}[\mathbf{t}_n\mathbf{t}_n^T]\Lambda_{d:}^T$$
$$- 2\mu_d E_{q_n}[\mathbf{x}_n]_d - 2\Lambda_{d:}E_{q_n}[\mathbf{x}_n\mathbf{t}_n^T]_{d:}^T + 2\mu_d\Lambda_{d:}E_{q_n}[\mathbf{t}_n] \tag{4.3.12}$$

$$\frac{\partial E[\mathcal{L}(\Lambda,\sigma,\mu|\mathbf{x}_n,\mathbf{t}_n)]}{\partial \sigma^2} = \sum_{n=1}^{N} -\frac{1}{2\sigma^2} + \frac{1}{2\sigma^4}E_{q_n}[(\mathbf{x}_n-\mu-\Lambda t_n)^T(\mathbf{x}_n-\mu-\Lambda t_n)]$$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N} \mathrm{tr}(E_{q_n}[\mathbf{x}_n\mathbf{x}_n^T]) + \mu^T\mu + \mathrm{tr}(\Lambda^T\Lambda E_{q_n}[\mathbf{t}_n\mathbf{t}_n^T])$$
$$- 2\mu^T E_{q_n}[\mathbf{x}_n] - 2\mathrm{tr}(\Lambda E_{q_n}[\mathbf{t}_n\mathbf{x}_n^T]) + 2\mu^T\Lambda E_{q_n}[\mathbf{t}_n] \tag{4.3.13}$$

The M-step updates can be expressed in terms of only five expectations derived from Equation 4.3.8. Namely, $E_{q_n}[\mathbf{x}_n] = [\mu_{x_n^m|x_n^o}, \mathbf{x}_n^o]$, $E_{q_n}[\mathbf{t}_n] = \mu_{t_n|x_n^o}$, $E_{q_n}[\mathbf{t}_n\mathbf{t}_n^T] = \Sigma_{t_n|x_n^o} + \mu_{t_n|x_n^o}\mu_{t_n|x_n^o}^T$, as well as $E_{q_n}[\mathbf{x}_n\mathbf{x}_n^T]$ and $E_{q_n}[\mathbf{x}_n\mathbf{t}_n^T]$ shown below.

$$E_{q_n}[\mathbf{x}_n\mathbf{x}_n^T] = \begin{bmatrix} (\Sigma_{x_n^m|x_n^o} + \mu_{x_n^m|x_n^o}\mu_{x_n^m|x_n^o}^T) & \mu_{x_n^m|x_n^o}(\mathbf{x}_n^o)^T \\ \mathbf{x}_n^o(\mu_{x_n^m|x_n^o})^T & \mathbf{x}_n^o(\mathbf{x}_n^o)^T \end{bmatrix} \quad E_{q_n}[\mathbf{x}_n\mathbf{t}_n^T] = \begin{bmatrix} \Sigma_{x_n^m t_n|x_n^o} + \mu_{x_n^m|x_n^o}\mu_{t_n|x_n^o}^T \\ \mathbf{x}_n^o\mu_{t_n|x_n^o}^T \end{bmatrix}$$

4.3.3 Predictive Distribution

A learned factor analysis model can be used to calculate the posterior predictive distribution on the missing data dimensions $\mathbf{x}_n^m$ given the observed data dimension $\mathbf{x}_n^o$. The posterior predictive distribution is Gaussian as seen in equation 4.3.14. The parameters are $\mu_{x_n^m|x_n^o} = \mu^m + \Sigma_{xx}^{mo}(\Sigma_{xx}^{oo})^{-1}(\mathbf{x}_n^o - \mu^o)$, $\Sigma_{x_n^m|x_n^o} = \Sigma_{xx}^{mm} - \Sigma_{xx}^{mo}(\Sigma_{xx}^{oo})^{-1}\Sigma_{xx}^{om}$, and $\Sigma_{xx} = \Psi + \Lambda\Lambda^T$.

$$P(\mathbf{x}_n^m|\mathbf{x}_n^o,\Lambda,\mu,\Psi) = \mathcal{N}(\mu_{x_n^m|x_n^o}, \Sigma_{x_n^m|x_n^o}) \tag{4.3.14}$$

4.4 Mixtures of Factor Analyzers

A mixture of factor analyzers is a finite mixture model where the mixture component distributions are factor analysis models [37, 26, 45, 74]. The generative process for the mixture of factor analyzers starts by selecting one of the K factor analysis models in the mixture according to the discrete probability distribution $P(z_n = k) = \theta_k$. Once a mixture component is selected, the generative process is identical to the generative process in a single factor analysis model. A low dimensional latent vector $\mathbf{t}_n$ is sampled from a zero mean, unit variance Gaussian distribution. A data case $\mathbf{x}_n$ is then sampled according to a Gaussian distribution with mean given by $\mu_k + \Lambda_k t_n$, and covariance matrix is given by Ψ_k. We summarize the mixture of factor analyzers model below.

$$P(z_n = k) = \theta_k \tag{4.4.1}$$

$$P(\mathbf{t}_n) = \frac{1}{(2\pi)^{Q/2}} \exp(-\frac{1}{2}\mathbf{t}_n^T\mathbf{t}_n) \tag{4.4.2}$$

$$P(\mathbf{x}_n|z_n = k, \mathbf{t}_n, \Lambda, \Psi, \mu) = \frac{1}{|2\pi\Psi_k|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}_n - \mu_k - \Lambda_k\mathbf{t}_n)^T\Psi_k^{-1}(\mathbf{x}_n - \mu_k - \Lambda_k\mathbf{t}_n)) \tag{4.4.3}$$

4.4.1 Joint, Conditional, and Marginal Distributions

The joint probability of $\mathbf{x}_n$ and $\mathbf{t}_n$ given $z_n = k$ is Gaussian as in a single factor analysis model. We give this probability in Equation 4.4.4.

$$P\left([\mathbf{x}_n, \mathbf{t}_n]^T|z_n = k, \mu, \Lambda, \Psi\right) = \mathcal{N}\left(\begin{bmatrix}\mu_{xk}\\ \mu_t\end{bmatrix}, \begin{bmatrix}\Sigma_{xxk} & \Sigma_{xtk}\\ \Sigma_{txk} & \Sigma_{ttk}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mu_k\\ 0\end{bmatrix}, \begin{bmatrix}\Psi_k + \Lambda_k\Lambda_k^T & \Lambda_k\\ \Lambda_k^T & I\end{bmatrix}\right) \tag{4.4.4}$$

The probability of a data case $\mathbf{x}_n$ given a choice of mixture component $z_n = k$ is obtained by marginalizing over $\mathbf{t}_n$ with z_n fixed.

$$P(\mathbf{x}_n|z_n = k, \Lambda, \sigma, \mu) = \mathcal{N}(\mathbf{x}_n; \mu_{xk}, \Sigma_{xxk}) = \mathcal{N}(\mathbf{x}_n; \mu_k, \Psi_k + \Lambda_k\Lambda_k^T)$$

4.4.2 Maximum Likelihood Estimation

The parameters of the mixture of factor analyzers model are found using an Expectation Maximization algorithm. The E-step for the mixture of factor analyzers first computes the probability that each data case belongs to each mixture component. The remainder of the E-step

consists of estimating expectations within each component model. The computation within each component model is identical to the E-step in a single factor analysis model. The M-step in the mixture of factor analyzers model updates the parameters of each component model according to that component's responsibility for each data case.

Below, we derive an EM algorithm where the missing data is handled along with the latent factors, and mixture indicators. The derivation for mixtures of factor analyzers with missing data is closely related to the derivation for mixtures of factor analyzers with complete data [26], as well as the derivation of full covariance mixtures of Gaussians with missing data [28]. We begin by forming the complete log-likelihood.

$$\begin{aligned}\mathcal{L}(\Lambda,\Psi,\mu,\theta|\mathbf{x}_n,\mathbf{t}_n,z_n) &= \sum_{k=1}^{K}[z_n=k](\log P(z_n=k|\theta)+\log P(\mathbf{t}_n)+\log P(\mathbf{x}_n|\mathbf{t}_n,\Lambda_k,\Psi_k,\mu_k)) \\ &= \sum_{k=1}^{K}[z_n=k]\left(-\frac{1}{2}(\mathbf{x}_n-\mu_k-\Lambda_k\mathbf{t}_n)^T\Psi_k^{-1}(\mathbf{x}_n-\mu_k-\Lambda_k\mathbf{t}_n)\right. \\ &\quad \left.+\log\theta_k-\frac{1}{2}\log(|2\pi\Psi_k|)-\frac{Q}{2}\log(2\pi)-\frac{1}{2}\mathbf{t}_n^T\mathbf{t}_n\right) \qquad (4.4.5)\end{aligned}$$

The E-step requires the distribution of the unobserved variables z_n, $\mathbf{t}_n$ and $\mathbf{x}_n^m$ given the observed variables $\mathbf{x}_n^o$. This distribution can be factorized as seen in Equation 4.4.6.

$$\begin{aligned}q_n(\mathbf{t}_n,\mathbf{x}_n^m,z_n) &= P(\mathbf{t}_n,\mathbf{x}_n^m|z_n,\mathbf{x}_n^o,\mu,\Lambda,\Psi)P(z_n|\mathbf{x}_n^o,\mu,\Lambda,\Psi,\theta) \qquad (4.4.6)\\ &= q_{nz_n}(\mathbf{t}_n,\mathbf{x}_n^m)q_n(z_n) \\ q_n(k) &\propto \theta_k\frac{1}{|2\pi(\Psi_k^o+\Lambda_k^o\Lambda_k^{oT})|}\exp(-\frac{1}{2}(\mathbf{x}_n^o-\mu_k^o)^T(\Psi_k^o+\Lambda_k^o\Lambda_k^{oT})^{-1}(\mathbf{x}_n^o-\mu_k^o)) \\ q_{nk}(\mathbf{t}_n,\mathbf{x}_n^m) &= \mathcal{N}\left(\begin{bmatrix}\mu_{t_n|x_n^ok}\\ \mu_{x_n^m|x_n^ok}\end{bmatrix},\begin{bmatrix}\Sigma_{t_n|x_n^ok} & \Sigma_{t_nx_n^m|x_n^ok}\\ \Sigma_{t_nx_n^m|x_n^ok}^T & \Sigma_{x_n^m|x_n^ok}\end{bmatrix}\right)\end{aligned}$$

The parameters of the normal distribution specified by $q_{nk}(\mathbf{t}_n,\mathbf{x}_n^m)$ are given below.

$$\begin{aligned}\mu_{t_n|x_n^ok} &= \Lambda_k^{oT}(\Sigma_{xxk}^{oo})^{-1}(\mathbf{x}_n^o-\mu_k^o) & \mu_{x_n^m|x_n^ok} &= \mu_k^m+\Sigma_{xxk}^{mo}(\Sigma_{xxk}^{oo})^{-1}(\mathbf{x}_n^o-\mu_k^o)\\ \Sigma_{t_n|x_n^ok} &= \Sigma_{tt}-\Sigma_{txk}^o(\Sigma_{xxk}^{oo})^{-1}\Sigma_{xtk}^o & \Sigma_{x_n^m|x_n^ok} &= \Sigma_{xxk}^{mm}-\Sigma_{xxk}^{mo}(\Sigma_{xxk}^{oo})^{-1}\Sigma_{xxk}^{om}\\ \Sigma_{t_nx_n^m|x_n^ok} &= \Sigma_{txk}^m-\Sigma_{txk}^o(\Sigma_{xxk}^{oo})^{-1}\Sigma_{xxk}^{om}\end{aligned}$$

We form the expected complete log likelihood function using the posterior distribution $q_n(z_n)q_{nz_n}(\mathbf{t},\mathbf{x}^m)$, and the complete log likelihood function.

$$E[\mathcal{L}(\Lambda,\Psi,\mu,\theta|\mathbf{x},\mathbf{t},\mathbf{z})] = \sum_{n=1}^{N}\sum_{z_n=1}^{K}\int\int q_n(z_n)q_{nz_n}(\mathbf{t},\mathbf{x}^m)\mathcal{L}(\Lambda,\Psi,\mu,\theta|\mathbf{x}_n,\mathbf{t}_n,\mathbf{z}_n)d\mathbf{x}_n^m d\mathbf{t}_n$$
$$= \sum_{n=1}^{N}\sum_{k=1}^{K} E_{q_n}[z_n=k]\left(-\frac{1}{2}E_{q_{nk}}[(\mathbf{x}_n-\mu_k-\Lambda_k\mathbf{t}_n)^T\Psi_k^{-1}(\mathbf{x}_n-\mu_k-\Lambda_k\mathbf{t}_n)]\right.$$
$$\left.+\log\theta_k-\frac{1}{2}\log(|2\pi\Psi_k|)-\frac{Q}{2}\log(2\pi)-\frac{1}{2}E_{q_{nk}}[\mathbf{t}_n^T\mathbf{t}_n]\right) \tag{4.4.7}$$

We give the maximum likelihood updates for θ, μ, Λ, and Ψ below. The derivation for θ is identical to the general finite mixture case. The derivations for μ, Λ, and Ψ are the same as for a single factor analysis model, except that each data case is weighted by the factor $q_n(k)$.

$$\theta_k = \frac{\sum_{n=1}^{N} q_n(k)}{N} \tag{4.4.8}$$

$$\mu_k = \frac{1}{\sum_{n=1}^{N} q_n(k)}\sum_{n=1}^{N} q_n(k)(E_{q_{nk}}[\mathbf{x}_n]-\Lambda_k E_{q_{nk}}[t_n]) \tag{4.4.9}$$

$$\Lambda_k = \left(\sum_{n=1}^{N} q_n(k)(E_{q_{nk}}[\mathbf{x}_n\mathbf{t}_n^T]-\mu_k E_{q_{nk}}[\mathbf{t}_n]^T)\right)\left(\sum_{n=1}^{N} q_n(k)E_{q_{nk}}[\mathbf{t}_n\mathbf{t}_n^T]\right)^{-1} \tag{4.4.10}$$

$$\Psi_{dd} = \frac{1}{\sum_{n=1}^{N} q_n(k)}\sum_{n=1}^{N} q_n(k)\left(E_{q_{nk}}[\mathbf{x}_n\mathbf{x}_n^T]_{dd}+\mu_{dk}^2+\Lambda_{d:k}E_{q_{nk}}[\mathbf{t}_n\mathbf{t}_n^T]\Lambda_{d:k}^T\right.$$
$$\left.-2\mu_{dk}E_{q_{nk}}[\mathbf{x}_n]_d-2\Lambda_{d:k}E_{q_{nk}}[\mathbf{x}_n\mathbf{t}_n^T]_{d:}^T+2\mu_{dk}\Lambda_{d:k}E_{q_{nk}}[\mathbf{t}_n]\right) \tag{4.4.11}$$

$$\sigma^2 = \frac{1}{\sum_{n=1}^{N} q_n(k)}\sum_{n=1}^{N} q_n(k)\left(\mathrm{tr}(E_{q_{nk}}[\mathbf{x}_n\mathbf{x}_n^T])+\mu_k^T\mu_k+\mathrm{tr}(\Lambda_k^T\Lambda_k E_{q_{nk}}[\mathbf{t}_n\mathbf{t}_n^T])\right.$$
$$\left.-2\mu_k^T E_{q_{nk}}[\mathbf{x}_n]-2\mathrm{tr}(\Lambda E_{q_{nk}}[\mathbf{t}_n\mathbf{x}_n^T])+2\mu_k^T\Lambda_k E_{q_n}[\mathbf{t}_n]\right) \tag{4.4.12}$$

The M-step updates can again be expressed in terms of only five expectations. Namely, $E_{q_{nk}}[\mathbf{x}_n]$, $E_{q_{nk}}[\mathbf{t}_n]$, $E_{q_{nk}}[\mathbf{x}_n\mathbf{x}_n^T]$, $E_{q_{nk}}[\mathbf{t}_n\mathbf{t}_n^T]$, and $E_{q_{nk}}[\mathbf{x}_n\mathbf{t}_n^T]$. These expectations can be derived from Equation 4.4.6 as shown below.

$$E_{q_{nk}}[\mathbf{t}_n]=\mu_{t_n|x_n^o k} \qquad E_{q_{nk}}[\mathbf{x}_n]=[\mu_{x_n^m|x_n^o k},\mathbf{x}_n^o] \qquad E_{q_{nk}}[\mathbf{t}_n\mathbf{t}_n^T]=\Sigma_{t_n|x_n^o k}+\mu_{t_n|x_n^o k}\mu_{t_n|x_n^o k}^T$$

$$E_{q_{nk}}[\mathbf{x}_n\mathbf{x}_n^T] = \begin{bmatrix} (\Sigma_{x_n^m|x_n^o k} + \mu_{x_n^m|x_n^o k}\mu_{x_n^m|x_n^o k}^T) & \mu_{x_n^m|x_n^o k}(\mathbf{x}_n^o)^T \\ \mathbf{x}_n^o(\mu_{x_n^m|x_n^o k})^T & \mathbf{x}_n^o(\mathbf{x}_n^o)^T \end{bmatrix}$$

$$E_{q_{nk}}[\mathbf{x}_n\mathbf{t}_n^T] = \begin{bmatrix} \Sigma_{x_n^m t_n|x_n^o k} + \mu_{x_n^m|x_n^o k}\mu_{t_n|x_n^o k}^T \\ \mathbf{x}_n^o\mu_{t_n|x_n^o k}^T \end{bmatrix}$$

4.4.3 Predictive Distribution

A learned mixture of factor analyzers model can be used to calculate the posterior predictive distribution of the missing data dimensions $\mathbf{x}_n^m$ given the observed data dimensions $\mathbf{x}_n^o$. The posterior predictive distribution is a mixture Gaussians, and can be computed in two steps as seen in Equation 4.4.13 where $\mu_{x_n^m|x_n^o k} = \mu_k^m + \Sigma_{xxk}^{mo}(\Sigma_{xxk}^{oo})^{-1}(\mathbf{x}_n^o - \mu_k^o)$, $\Sigma_{x_n^m|x_n^o k} = \Sigma_{xxk}^{mm} - \Sigma_{xxk}^{mo}(\Sigma_{xxk}^{oo})^{-1}\Sigma_{xxk}^{om}$, and $\Sigma_{xxk} = \Psi_k + \Lambda_k\Lambda_k^T$.

$$P(\mathbf{x}_n^m|\mathbf{x}_n^o, \Lambda, \mu, \Psi) = \sum_{k=1}^{K} q_n(k)\mathcal{N}(\mu_{x_n^m|x_n^o k}, \Sigma_{x_n^m|x_n^o k}) \tag{4.4.13}$$

$$q_n(k) \propto \theta_k \frac{1}{|2\pi\Sigma_{xxk}^{oo}|}\exp(-\frac{1}{2}(\mathbf{x}_n^o - \mu_k^o)^T(\Sigma_{xxk}^{oo})^{-1}(\mathbf{x}_n^o - \mu_k^o))$$

Chapter 5

Unsupervised Learning with Non-Random Missing Data

This chapter presents work on unsupervised learning with non-random missing data. We use collaborative prediction as a motivating example throughout this chapter. In a typical collaborative filtering system users assign ratings to items, and the system uses information from all users to recommend previously unseen items that each user might like or find useful. One approach to recommendation is to predict the ratings for all unrated items, and then recommend the items with the highest predicted ratings. Collaborative filtering research has focused almost exclusively on developing new models and new learning procedures to improve rating prediction performance [16, 30, 33, 39, 52].

A critical assumption behind both learning methods and testing procedures is that the missing ratings are missing at random. As discussed in Section 3.1, one way to violate the missing at random condition is for the value of a variable to affect the probability that the variable is missing. In the collaborative filtering setting this translates into the statement that the rating value for a particular item affects the probability that the rating value will be missing. In an internet based movie recommendation system, for example, users may be less likely to enter ratings for movies they do not like since they avoid seeing such movies in the first place. This would create a systematic bias towards observing ratings with higher values.

As discussed in Section 3.3, the presence of non-random missing data can introduce a systematic bias into the learned parameters of parametric and semi-parametric models such as mixture models [9], matrix factorization models [16], and other specialized probabilistic models [52]. It is important to note that the presence of non-random missing data introduces a complementary bias into the standard testing procedure for rating prediction experiments [9] [33] [52, p.90]. Models are typically learned on one subset of the observed data, and tested on a different subset of the observed data. If the distribution of the observed data is different from

the distribution of the complete data for any reason, the estimated error on observed test data can be an arbitrarily poor estimate of the error on the complete data. Marlin, Roweis, and Zemel confirm this using experiments on synthetic data [54].

In this chapter we begin by describing an online user study and rating data collection experiment based on Yahoo! Music's LaunchCast internet radio service. The data collection experiment was designed to gather information about how users choose which items they rate. The initial empirical work performed by Marlin et al. used a proprietary data set derived from this data collection experiment [53]. This work uses a slightly different data set that is publicly available through the Yahoo! Research Alliance Webscope data sharing program under the name *ydata-ymusic-rating-study-v1_0*.

We present an analysis of this new data set, which includes survey responses, and ratings for randomly chosen songs. We describe methods for learning and prediction with non-random missing data based on Bayesian mixture models, Dirichlet Process mixture models, and restricted Boltzmann machines. We introduce a new experimental protocol for rating prediction based on training models using user-selected items, and testing models using randomly selected items. Experimental results show that incorporating a simple, explicit model of the missing data mechanism can lead to significant improvements in prediction error as well as rank loss compared to. Experiments are performed both on data from the Yahoo! LaunchCast study, and synthetic data based on the Jester data set [30].

5.1 The Yahoo! Music Data Set

To properly assess the impact of the missing at random assumption on rating prediction, we require a test set consisting of ratings that are a random sample of the ratings contained in the complete data matrix for a given set of users. In this section we describe a study conducted in conjunction with Yahoo! Music's LaunchCast Radio service to collect such a data set.

LaunchCast radio is a free, customizable streaming music service where users can influence the music played on their personal station by supplying ratings for songs. The LaunchCast Radio player interface allows the user enter a rating for the currently playing song using a five point scale. Users can also enter ratings for artists and albums.

Data was collected from LaunchCast Radio users between August 22, 2006 and September 12, 2006. Users based in the US were able to join the study by clicking on a link in the LaunchCast player. Both the survey responses and rating data were collected through the study's web site. A total of $35,786$ users participated in the study. The results reported in this paper are based on a random subset of 5400 survey participants and $10,000$ non-survey participants who had at least 10 existing ratings in the LaunchCast rating database.

Rating Frequency	Preference Level				
	Hate it	Don't like it	Neutral	Like it	Love it
Never	6.76%	4.69%	2.33%	0.11%	0.07%
Very Infrequently	1.59%	4.17%	9.46%	0.54%	0.35%
Infrequently	1.63%	4.44%	24.87%	1.48%	0.20%
Often	12.46%	22.50%	26.83%	25.30%	5.46%
Very Often	77.56%	64.20%	36.50%	72.57%	93.91%

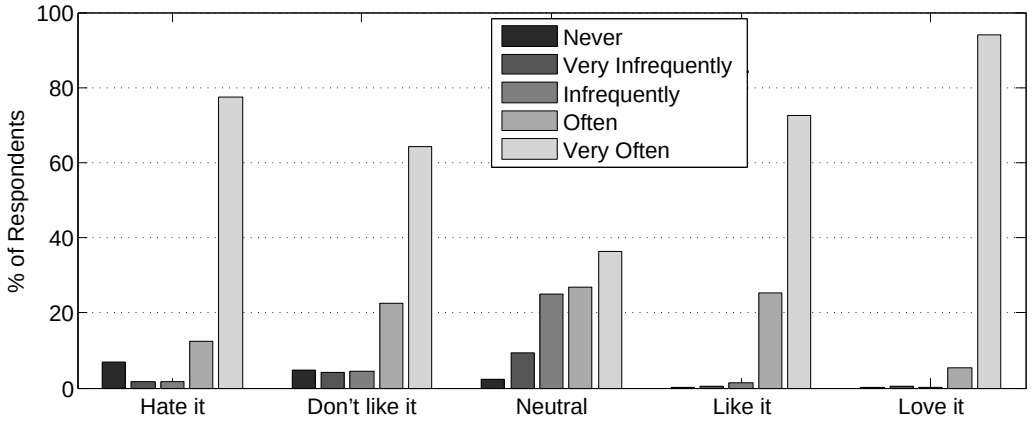
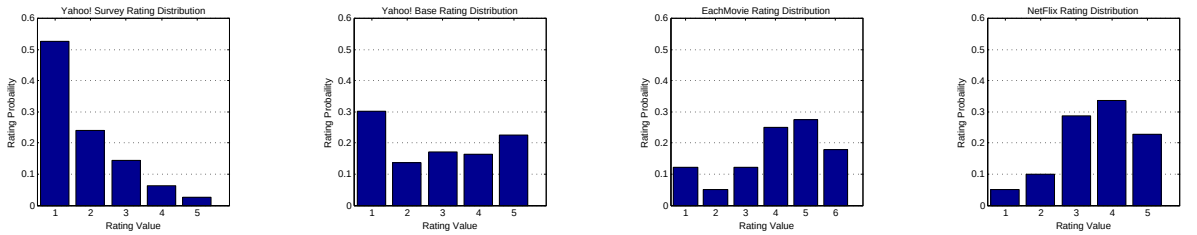


Figure 5.1: Yahoo! LaunchCast users were asked to report the frequency with which they choose to rate a song given their preference for that song. The data above show the distribution over rating frequencies given several preference levels. Users could select only one rating frequency per preference level.

5.1.1 User Survey

The first part of the study consisted of a user survey. The questions relevant to this work asked users to report on how their preferences affect which songs they choose to rate. The question was broken down by asking users to estimate how often they rate a song given the degree to which they like it. The results are summarized in the table and accompanying chart shown in Figure 5.1. Each column in the table gives the results for a single survey question. For example, the column labeled "neutral" corresponds to the question "If I hear a song I feel neutral about I choose to rate it:" with the possible answers being "never", "very infrequently", "infrequently", "often", and "very often".

The results indicate that the choice to rate a song does depend on the user's opinion of that song. Most users tend to rate songs that they love more often than songs they feel neutral about, and somewhat more often than songs that they hate. Users were also directly asked if they thought their preferences for a song *do not* affect whether they choose to rate it. 64.85% of users responded that their preferences *do* affect their choice to rate a song. By contrast, the missing at random assumption requires that the underlying ratings not influence a user's choice to rate a song.



(a) Yahoo! Survey Rating Distribution

(b) Yahoo! Base Rating Distribution

(c) EachMovie Rating Distribution

(d) NetFlix Rating Distribution

Figure 5.2: Distribution of rating values in the Yahoo! survey set and base set compared to several popular collaborative filtering data sets including EachMovie, and Netflix.

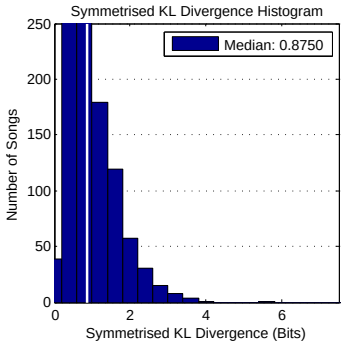
5.1.2 Rating Data Analysis

Following the survey, users were presented with a set of ten songs to rate. The artist name and song title were given for each song, along with a thirty second audio clip, which the user could play before entering a rating. Ratings were entered on the standard five point scale used by Yahoo! Music. The set of ten songs presented to each user was chosen at random without replacement from a fixed set of 1000 songs. The fixed set of 1000 songs used in the survey were chosen at random from all the songs in the LaunchCast play list having at least 500 existing ratings in the LaunchCast rating database.

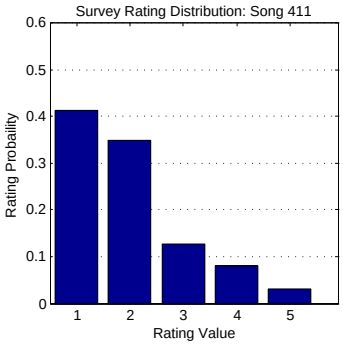
We refer to ratings collected during the survey as "survey ratings." In addition, each survey participant's existing ratings on the set of 1000 survey songs was extracted from the LaunchCast database. We refer to these existing ratings as the "base ratings." The survey ratings represent ratings for a random sample of songs, while the base ratings represent ratings for songs the survey participant *chose* to enter.

Figures 5.2(a) and 5.2(b) show the empirical distribution of survey ratings and base ratings for the 5400 survey participants. These figures show a dramatic difference between the two distributions. The number of four and five star rating values is many times lower in the survey set than the base set. The difference between the survey and base distributions is not surprising given that users can influence the LaunchCast system to play songs reflecting their preferences. Figures 5.2(c) and 5.2(d) give the rating distributions for EachMovie, and Netflix. These distributions show a much higher proportion of high rating values than are present in the random sample we collected during the survey.

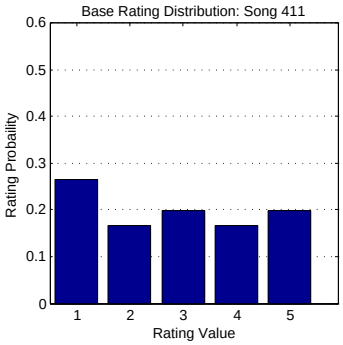
To further analyze the difference between the base ratings and the survey ratings, we computed the rating distribution separately for each item. For a particular item d let $P^S(X_d = v)$ be the empirical probability of rating value v in the survey set, and $P^B(X_d = v)$ be the empirical probability of rating value v in the base set. We smooth the empirical probabilities by one count



(a) Histogram of number of songs vs symmetrised KL divergence. The median value is 0.8750.



(b) Survey marginal distribution for song 411 with symmetrised KL divergence 0.8749.



(c) Base marginal distribution for song 411 with symmetrised KL divergence 0.8749.

Figure 5.3: Figures (a) to (c) give an indication of the distribution of differences between survey and base marginal distributions for each song.

per rating value to avoid zeros. We use the symmetrised Kullback−Leibler divergence (SKL) shown in Equation 5.1.1 to measure the difference between the $P^S(X_d = v)$ and $P^B(X_d = v)$ distributions for each item d.

$$SKL_d = \sum_{v=1}^{V} P^S(X_d = v) \log \left(\frac{P^S(X_d = v)}{P^B(X_d = v)} \right) + P^B(X_d = v) \log \left(\frac{P^B(X_d = v)}{P^S(X_d = v)} \right) \tag{5.1.1}$$

Figure 5.3(a) shows a histogram of the symmetrised Kullback−Leibler divergence values. The thick vertical line in the plot indicates the median SKL value of 0.8750 bits. Song 411 has an SKL value of 0.8749 bits, the largest SKL value less than the median. Figures 5.3(b) and 5.3(c) illustrate the marginal rating distributions for song 411. These distributions are qualitatively quite different, and half of the songs in the survey set exhibit a more extreme difference according to the SKL measure.

A pertinent question is whether users' ratings reported during the survey were consistent with ratings recorded during normal use of the LaunchCast system. Marlin et al. showed very good agreement on the approximately 1700 instances where the same user rated the same item both during normal use of the LaunchCast system, and during the survey [53]. This comparison was based on the complete set of $35,786$ survey participants.

It is important to note that the observed discrepancy between the survey set marginal distributions and the base set marginal distributions is not conclusive evidence that the missing data in the base set is NMAR. This is due to the fact that the MAR assumption can hold for the true underlying generative process, while inference for parameters of a simpler model can still be biased as discussed in Section 3.4. Nevertheless, we believe that the results of this

analysis combined with the results of the user survey provide compelling evidence against the MAR assumption.

5.1.3 Experimental Protocols for Rating Prediction

The rating data set used in the experiments is based on the 1000 survey songs and a set of $15,400$ users. The set of $15,400$ users consists of a random selection of $10,000$ non-survey participants with at least 10 existing ratings on the set of 1000 survey songs, and a random selection of 5400 survey participants with at least 10 existing ratings on the set of 1000 survey songs. We chose to enforce a minimum number of existing ratings per user so that rating prediction methods would have at least 10 observations on which to base predictions. Non-survey users were included to provide more training data to the learning methods.

In this work we follow an experimental protocol based on a five fold cross validation assessment of both the weak and strong generalization performance of rating prediction methods [52, p. 14]. We consider the prediction of ratings for both user-selected and randomly selected songs. Preparing the data set for this empirical protocol involves several steps. We begin by randomly partitioning the 5400 survey users into five blocks of 1080 users each. Each block is used as a set of test users in turn. The ratings for the test users are further divided into observed ratings, test ratings for user selected items, and test ratings for randomly selected items. The observed ratings consist of all but one of the user selected ratings for each user. The test ratings for user selected items consist of exactly one rating for a user selected item per user. The observed ratings consist of the remaining user selected ratings for each user. The test ratings for randomly selected items consist of the 10 ratings for randomly selected items collected for each user during the survey. Any overlapping ratings in the user selected and randomly selected sets were removed from the user selected set before the Yahoo! data was publicly released. Note that these ratings should have been preserved in both rating sets, but the effect of removing them from the user selected set is small.

Of the 4320 survey users from the remaining four blocks, 400 users are selected at random to form a hold out set. The set of training users consists of the remaining 3920 survey users, and all of the $10,000$ non-survey users. The training user ratings are divided into observed ratings, test ratings for user selected items, and test ratings for randomly selected items. The survey users in the training set have one test rating for a user selected item, and 10 test ratings for randomly selected items. The non-survey users have one test rating for a user selected item, and no test ratings for randomly selected items. Each set of training users has approximately $250,000$ observed ratings, while each set of test users has approximately $25,000$ observed ratings.

Unless otherwise specified, models are trained using the training users' observed ratings. We use normalized mean absolute error as the error measure for collaborative prediction [52, p. 15].

Given a predictive distribution over the V rating values, we predict the median rating value. The weak generalization error for user-selected items is computed by predicting the value of each training user's user-selected test items. The weak generalization error for randomly selected items is computed by predicting the value of each training user's randomly-selected test items. The strong generalization error for user-selected items is computed by predicting the value of each test user's user-selected test items. The strong generalization error for randomly selected items is computed by predicting the value of each test user's randomly-selected test items.

The novel aspect of this protocol stems from the division of the rating data into three sets of ratings. The observed ratings for each user are ratings for user-selected items. Thus, the user selected test items come from the same distribution as the observed ratings, and both rating sets are subject to the same unknown missing data mechanism. The ratings for randomly selected test items come from a known selection mechanism where missing data is missing completely at random.

If missing data is missing at random in the user selected items, we would not expect to see a large difference in prediction performance on the randomly selected and user selected test items. If missing data is not missing at random, we would expect methods that incorrectly assume the MAR condition to perform poorly on the randomly selected test items compared to the user-selected test items.

5.2 The Jester Data Set

The Jester recommender system was established as part of a research project on collaborative filtering methods [30]. The system allows a user to rate jokes, and then recommends other jokes the user might like. Jester is unique among the available collaborative filtering data sets in that it contains a dense subset of ten items that all users were required to rate upon joining the system. We extracted 15000 completely observed ratings vectors from the Jester gauge set. Each rating vector consists of ratings for all ten of the gauge set items. The ratings were mapped from the original continuous rating scale to 5 discrete rating values.

5.2.1 Experimental Protocols for Rating Prediction

Artificial missing data was added to the Jester data set using a synthetic missing data mechanism. The missing data model we use has the form $P(r_{dn} = 1|x_{dn} = v, \mu_v(s)) = \mu_v(s)$. The function $\mu_v(s) = s(v - 3) + 0.5$. As s increases from 0 to 0.25, high rating values have an increasing chance of being observed, and low rating values have an increasing chance of being missing. We applied this missing data mechanism to the Jester ratings using the five sets of missing data model parameters pictured in Figure 5.4, and obtained five data sets with artifi-

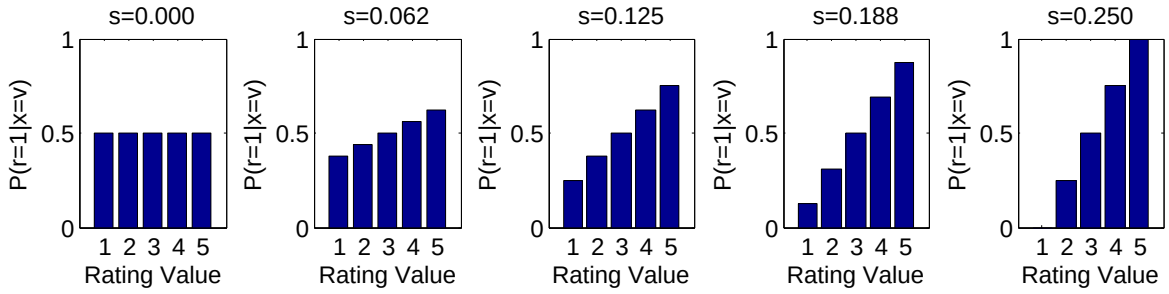


Figure 5.4: Selection probabilities for the effect $\mu_v(s) = s(v-3)+0.5$. The parameter s controls the strength of the missing data effect. Here we show $\mu_v(s)$ at five equally spaced values on the interval $0 \leq s \leq 0.250$. These are the parameter sets used to create artificial non-random missing data in the Jester data set.

cial non-random missing data. The ratings selected by our synthetic non-random missing data mechanism simulate the ratings for user selected items in the Yahoo! data set. To simulate the ratings for randomly selected items, we also selected one item at random for each user from the set of 10 items in the base data.

As in the Yahoo! data set, the experimental protocol used to evaluate predictions on the Jester data set is based on a five fold cross validation assessment of both strong and weak generalization performance on test ratings from the simulated user selection process, and test ratings from the simulated random selection process. As in the Yahoo! data set we split the available users into test users, training users, and hold out users. We use a test set of 3000 users, a training set of $10,000$ users, and a hold out set of 2000 users. Within the training and test sets we select one simulated user selected item to act as the user selected test item. The simulated randomly selected test item is removed from the remaining user selected items if required. However, we retain the fact that the randomly selected test item was originally observed in this case. The remaining user selected items form the observed ratings. We apply the same training and testing procedures as described for the Yahoo! data set resulting in estimates of weak generalization error on user selected items, weak generalization error on randomly selected items, strong generalization error on user-selected items, and strong generalization error on randomly-selected items.

The main benefit of using synthetic missing data is that we can study the behaviour of models when the missing data has different properties. At $s = 0$ in the synthetic missing data mechanism, the observation probabilities are the same for each rating value so the missing data is missing at random. In this case we would expect methods that ignore the missing data mechanism, and methods that model the missing data mechanism to give similar performance. As s increases, models that ignore the missing data mechanism will be increasingly mislead about the true underlying data distribution.

5.3 Test Items and Additional Notation for Missing Data

The act of removing user selected items from the training data to use as test items effectively introduces an additional source of missing data. Missing training data that results from held out test items must be treated differently than the missing data that results from the natural missing data process. For example, suppose item d was originally selected and rated by user n. The training data would contain $r_{dn} = 1$ and $x_{dn} = v$. Now suppose that item d is chosen as a test item for user n. We must remove the value $x_{dn} = v$ from the training data. However, it is not correct to set $r_{dn} = 0$ in the training data since user n originally selected item d.

Keeping track of the two sources of missing data in the training data requires the use of additional notation. We define $r_{dn} = 1$ to mean that user n originally selected and rated item d. We define $a_{dn} = 1$ to mean that the rating value x_{dn} is available for use during training or inference. If $r_{dn} = 1$ and $a_{dn} = 1$ this implies that user n originally selected and rated item d, and that the rating for item d is available during training and inference. This is the normal situation during learning. If $r_{dn} = 1$ and $a_{dn} = 0$ this implies that user n originally selected and rated item d, but the rating for item d is not available during training or inference. This situation occurs when x_{dn} is removed from the training set because item d is used as a test rating. If $r_{dn} = 0$ and $a_{dn} = 1$ this implies that user n originally did not select and rate item d, but the rating for item d is available. This situation generally arises when we predict the value of a randomly selected test item. It also occurs in some of the specialized training procedures we consider. Finally, $r_{dn} = 0$ and $a_{dn} = 0$ implies that item d was not originally selected and rated by user n, and that the rating value is not available for learning.

5.4 The Finite Mixture/CPT-v Model

The CPT-v missing data model was proposed by Marlin, Roweis, and Zemel as a highly simplified non-random missing data model for collaborative filtering [54]. The CPT-v missing data model captures the intuition that a user's preference for an item affects whether they choose to rate that item or not. The model assumes that the choice to rate each item is independent, and the probability of rating a single item, given the user's rating for the item is v, is Bernoulli distributed with parameter μ_v. Marlin et al. combine the CPT-v missing data model with a finite mixture data model [54, 53]. Figure 5.5 shows a graphical representation of the *CPT-v* missing data model combined with the finite multinomial mixture data model. We give the corresponding probabilistic model in equations 5.4.1 to 5.4.6.

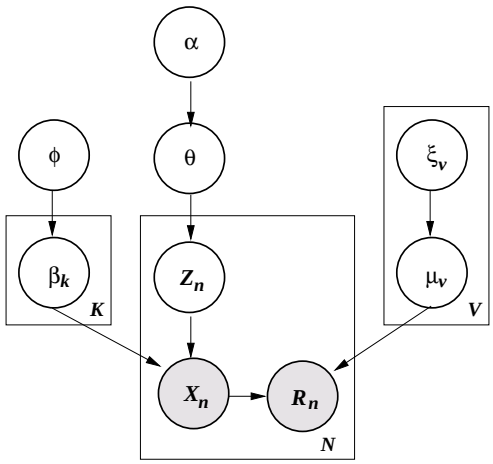


Figure 5.5: Bayesian network for the Bayesian mixture/CPT-v model combination.

$$P(\theta|\alpha) = \mathcal{D}(\theta|\alpha) \tag{5.4.1}$$

$$P(\beta|\phi) = \prod_{k=1}^{K}\prod_{d=1}^{D} \mathcal{D}(\beta_{dk}|\phi_{dk}) \tag{5.4.2}$$

$$P(\mu|\xi) = \prod_{v} \mathcal{B}(\mu_v|\xi_v) \tag{5.4.3}$$

$$P(Z_n = k|\theta) = \theta_k \tag{5.4.4}$$

$$P(\mathbf{X} = \mathbf{x}_n|Z_n = k, \beta) = \prod_{d=1}^{D}\prod_{v=1}^{V} \beta_{vdk}^{[x_{dn}=v]} \tag{5.4.5}$$

$$P(\mathbf{R} = \mathbf{r}_n|\mathbf{X} = \mathbf{x}_n, \mu) = \prod_{d=1}^{D}\prod_{v=1}^{V} \mu_v^{[r_{dn}=1][x_{dn}=v]}(1-\mu_v)^{[r_{dn}=0][x_{im}=v]} \tag{5.4.6}$$

The Beta prior $P(\mu|\xi)$ is the conjugate prior for the Bernoulli parameters μ_v. We give the form of the Beta prior distribution in Equation 5.4.7.

$$\mathcal{B}(\mu_v|\xi_v) = \frac{\Gamma(\xi_{0v}+\xi_{1v})}{\Gamma(\xi_{0v})\Gamma(\xi_{1v})}\mu_v^{\xi_{1v}-1}(1-\mu_v)^{\xi_{0v}-1} \tag{5.4.7}$$

Equation 5.4.8 gives the complete data likelihood assuming the values of mixture indicator z_n and missing rating values $\mathbf{x}_n^m$ are known. Computing the observed data likelihood involves summing out over all joint configurations of the missing rating values $\mathbf{x}_n^m$. This sum has a number of terms that is exponential in the number of missing data values. The independence structure of the finite mixture/CPT-v model combination allows this sum to be computed

efficiently as seen in Equation 5.4.9. Unlike the missing at random case, both the missing rating values and response indicators contribute to the likelihood.

$$P(z_n, \mathbf{x}_n, \mathbf{r}_n|\beta, \theta, \mu) = \prod_{k=1}^{K}\left(\theta_k \prod_{d=1}^{D}\prod_{v=1}^{V}\left(\mu_v^{[r_{dn}=1]}(1-\mu_v)^{[r_{dn}=0]}\beta_{vdk}\right)^{[x_{dn}=v]}\right)^{[z_n=k]} \tag{5.4.8}$$

$$\begin{aligned}
P(\mathbf{x}_n^o, \mathbf{r}_n|\beta, \theta, \mu) &= \sum_{\mathbf{x}^m}\sum_{k=1}^{K}\theta_k \prod_{d=1}^{D}\prod_{v=1}^{V}\left(\mu_v^{[r_{dn}=1]}(1-\mu_v)^{[r_{dn}=0]}\beta_{vdk}\right) \\
&= \sum_{k=1}^{K}\theta_k\left(\prod_{d=1}^{D}\prod_{v=1}^{V}(\mu_v\beta_{vdk})^{[r_{dn}=1][x_{dn}=v]}\right) \\
&\qquad \left(\sum_{x_1^m=1}^{V}\cdots\sum_{x_D^m=1}^{V}\prod_{d=1}^{D}\prod_{v=1}^{V}((1-\mu_v)\beta_{vdk})^{[r_{dn}=0][x_d^m=v]}\right) \\
&= \sum_{k=1}^{K}\theta_k\left(\prod_{d=1}^{D}\prod_{v=1}^{V}(\mu_v\beta_{vdk})^{[r_{dn}=1][x_{dn}=v]}\right)\left(\prod_{d=1}^{D}\sum_{v=1}^{V}((1-\mu_v)\beta_{vdk})^{[r_{dn}=0]}\right) \\
&= \sum_{k=1}^{K}\theta_k\prod_{d=1}^{D}\left(\prod_{v=1}^{V}(\mu_v\beta_{vdk})^{[x_{dn}=v]}\right)^{[r_{dn}=1]}\left(\sum_{v=1}^{V}(1-\mu_v)\beta_{vdk}\right)^{[r_{dn}=0]}
\end{aligned} \tag{5.4.9}$$

5.4.1 Conditional Identifiability

The CPT-v missing data model has the interesting property that its parameters are conditionally identifiable even when the multinomial mixture model parameters are not identifiable. The proof builds on the work of Glonek, who studied the problem of non-random missing data for a single binary variable in the classification setting [29]. We first review Glonek's proof, and then extend the proof method to the CPT-v missing data mechanism.

Binary Classification with Non-Random Missing Labels

Consider a binary classification task with fully observed binary predictors $\mathbf{x}$, and a single binary class variable y subject to nonignorable nonresponse. The data model is a conditional model parameterized by $\pi_{\mathbf{x}} = P(Y = 1|\mathbf{X} = \mathbf{x})$, while the selection model is parameterized by $\mu_y = P(R = 1|Y = y)$ where R is a binary response indicator. Thus, the model consists of the predictor or feature variables $\mathbf{X}$, the binary class variable Y, and the binary response indicator variable R.

Glonek defines the variable w to be categorical with values from the set $\{0, 1, \varnothing\}$ where $\varnothing$ indicates that the value of y was not observed [29]. Thus, the single variable w replaces the combination of variables y, and r. Note that regardless of the data and selection models in question, all that is ever observed in the data are the values of the random variables w, and the

predictors $\mathbf{x}$. We can now define the distribution over the observed variables $\phi_{w\mathbf{x}} = P(W = w|\mathbf{X} = \mathbf{x})$.

A necessary condition for identifiability is that the number of degrees of freedom in the observed variable distributions exceed the number of degrees of freedom in the model parameters. If the total number of different configurations of the predictor variables $\mathbf{x}$ is M, then the number of degrees of freedom in the observed variable distributions will be $2M$ since w is a three-valued variable with the constraint that the values sum to one. The number of degrees of freedom in the model parameters will be $M + 2$, and the condition is satisfied if $M \geq 2$.

Glonek formulates a precise characterization of global identifiability, which we now recount. Suppose we have at least two distinct settings of the predictor variables, and for two particular settings of the predictor variables $\mathbf{x}_1$ and $\mathbf{x}_2$ we have $\pi_{\mathbf{x}_1} \neq \pi_{\mathbf{x}_2}$. We relate the observed distributions to the model parameters as follows:

$$\phi_{0\mathbf{x}_1} = \mu_0(1-\pi_{\mathbf{x}_1}) \quad \phi_{1\mathbf{x}_1} = \mu_1(\pi_{\mathbf{x}_1}) \quad \phi_{0\mathbf{x}_2} = \mu_0(1-\pi_{\mathbf{x}_2}) \quad \phi_{1\mathbf{x}_2} = \mu_1(\pi_{\mathbf{x}_2})$$

Noting that $\phi_{0\mathbf{x}_1}/\mu_0+\phi_{1\mathbf{x}_1}/\mu_1 = (1-\pi_{\mathbf{x}_1})+\pi_{\mathbf{x}_1} = 1$ and $\phi_{0\mathbf{x}_2}/\mu_0+\phi_{1\mathbf{x}_2}/\mu_1 = (1-\pi_{\mathbf{x}_2})+\pi_{\mathbf{x}_2} = 1$ we can reduce the above set of equations to the following linear system for $1/\mu_0$ and $1/\mu_1$.

$$\begin{bmatrix} \phi_{0\mathbf{x}_1} & \phi_{1\mathbf{x}_1} \\ \phi_{0\mathbf{x}_2} & \phi_{1\mathbf{x}_2} \end{bmatrix} \begin{bmatrix} \frac{1}{\mu_0} \\ \frac{1}{\mu_1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.4.10}$$

Subject to the conditions $\pi_{\mathbf{x}_1} \neq \pi_{\mathbf{x}_2}$ and $\mu_0, \mu_1 > 0$, the above matrix is invertible. As a result, the above system has a unique solution for μ_0, μ_1, and all of the $\pi' s$.

CPT-v and Conditional Identifiability

In the collaborative filtering case the observed variables are denoted w_d, and take values on the extended set $\{1, ..., V, \varnothing\}$ where $\varnothing$ again denotes the observation that x_d is missing. Thus w_d replaces x_d and r_d. We denote the parameters of the joint distribution of the observed variables by $\phi_{\mathbf{w}}$. This distribution has $V^D - 1$ free parameters. We relate the parameters of the joint distribution to the parameters of the model distribution as seen in Equation 5.4.11.

$$\phi_{\mathbf{w}} = \sum_{k=1}^{K} \theta_k \prod_{d=1}^{D} \left(\prod_{v=1}^{V} (\mu_v \beta_{vdk})^{[w_d=v]} \right)^{[w_d \neq \varnothing]} \left(\sum_{v=1}^{V} (1-\mu_v)\beta_{vdk} \right)^{[w_d=\varnothing]} \tag{5.4.11}$$

Before describing the general case, we consider a specific instance where $V = 2, D = 2, K = 2$. We list the ϕ functions for several configuration of the observed variables.

$$\phi_{11} = \sum_{k=1}^{K} \theta_k \mu_1 \beta_{11k} \mu_1 \beta_{12k} \qquad \phi_{21} = \sum_{k=1}^{K} \theta_k \mu_2 \beta_{21k} \mu_1 \beta_{12k}$$
$$\phi_{12} = \sum_{k=1}^{K} \theta_k \mu_1 \beta_{11k} \mu_2 \beta_{22k} \qquad \phi_{22} = \sum_{k=1}^{K} \theta_k \mu_2 \beta_{21k} \mu_2 \beta_{22k}$$
$$\phi_{1\varnothing} = \sum_{k=1}^{K} \theta_k \mu_1 \beta_{11k} (\sum_{v=1}^{V} (1-\mu_v)\beta_{v2k}) \qquad \phi_{2\varnothing} = \sum_{k=1}^{K} \theta_k \mu_2 \beta_{21k} (\sum_{v=1}^{V} (1-\mu_v)\beta_{v2k})$$
$$\phi_{\varnothing 1} = \sum_{k=1}^{K} \theta_k (\sum_{v=1}^{V} (1-\mu_v)\beta_{v1k}) \mu_1 \beta_{12k} \qquad \phi_{\varnothing 2} = \sum_{k=1}^{K} \theta_k (\sum_{v=1}^{V} (1-\mu_v)\beta_{v1k}) \mu_2 \beta_{22k}$$

We show that μ_1 and μ_2 are uniquely determined using an argument similar to that used by Glonek. Consider the system of linear equations derived from the full set of ϕ values given previously.

$$\Phi \begin{bmatrix} \frac{1}{\mu_1} \\ \frac{1}{\mu_2} \end{bmatrix} = \begin{bmatrix} (\phi_{11} + \phi_{12} + \phi_{1\varnothing}) & (\phi_{21} + \phi_{22} + \phi_{2\varnothing}) \\ (\phi_{11} + \phi_{21} + \phi_{\varnothing 1}) & (\phi_{12} + \phi_{22} + \phi_{\varnothing 2}) \end{bmatrix} \begin{bmatrix} \frac{1}{\mu_1} \\ \frac{1}{\mu_2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5.4.12}$$

This system will have a unique solution for μ_1 and μ_2 assuming both are greater than 0, and the matrix Φ of sums of ϕ coefficients is non-singular. We note that the entry $\Phi(d, v)$ is equal to the marginal observed probability $P(w_d = v)$. This result is easily extended to the general case when $D \geq V$ by defining $\Phi(d, v)$ as follows:

$$\Phi(d, v) = \sum_{x_1} \cdots \sum_{x_{d-1}} \sum_{x_{d+1}} \cdots \sum_{x_D} \phi_{x_1 \cdots x_{d-1} v x_{d+1} \cdots x_D} \tag{5.4.13}$$

As before the relation $\sum_v \Phi(d, v)/\mu_v = 1$ holds. Suppose that for some set of V data dimensions, the corresponding matrix Φ is nonsingular. Without loss of generality we can assume it is the first V dimensions of the data. Under these conditions, the following system has a unique solution, and the CPT-v selection model parameters are uniquely identified.

$$\begin{bmatrix} \Phi_{11} & \Phi_{12} & \cdots & \Phi_{1V} \\ \Phi_{21} & \Phi_{22} & \cdots & \Phi_{2V} \\ \vdots & & \ddots & \vdots \\ \Phi_{V1} & \Phi_{V2} & \cdots & \Phi_{VV} \end{bmatrix} \begin{bmatrix} \frac{1}{\mu_0} \\ \frac{1}{\mu_1} \\ \vdots \\ \frac{1}{\mu_V} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{5.4.14}$$

The matrix Φ may have an interesting interpretation as far as inferring the selection model parameters from data is concerned. Consider the case where data is generated exactly from a CPT-v selection model paired with a multinomial mixture model. In this case we can directly estimate the Φ_{dv} coefficients from the data, and they will be correct up to sampling error. We can select a suitable set of V items and compute estimate for the μ_v parameters. However, in

the presence of deviations from the CPT-v model, the estimates obtained from this procedure may not be probabilities.

It is a well known fact of linear algebra that the condition number of a linear system, the ratio of largest to smallest singular value of the coefficient matrix, is an indication of the sensitivity of the system. If the coefficient matrix has high condition number, then small changes to the coefficients can result is large changes in the solution to the system. If we construct a non-singular matrix Φ from a sample of data and see that the condition number of the matrix is high, this is an indication that the selection model parameters could be highly sensitive to the particular data sample used to estimate them. On the other hand, if the condition number is low, the selection model parameters should be fairly invariant to the particular data sample used to estimate them.

5.4.2 Maximum A Posteriori Estimation

The principal of maximum a posteriori probability states that we should select the parameters θ, β, and μ with maximum posterior probability. As in the case of the finite Bayesian mixture model, maximum a posteriori estimation can be accomplished using an expectation maximization algorithm. To derive the expectation maximization algorithm we first require the posterior distribution of the mixture indicator variable z_n and the missing ratings $\mathbf{x}_n^m$ given the observed data $\mathbf{x}_n^o$, the response indicators $\mathbf{r}_n$ and the parameters θ, β, μ. At this point, we also introduce the available rating indicators $\mathbf{a}_n$ to properly account for held out test data during inference and learning. To help simplify the notation, we introduce the auxiliary variables γ_{dkn} in Equation 5.4.15. The posterior distribution is denoted $q_n(k, \mathbf{x}_n^m)$ and given in Equation 5.4.16.

$$\begin{aligned}\gamma_{dkn} = &\left(\prod_{v=1}^{V}(\beta_{vdk}\mu_v)^{[x_{dn}=v]}\right)^{[r_{dn}=1][a_{dn}=1]}\left(\prod_{v=1}^{V}(\beta_{vdk}(1-\mu_v))^{[x=v]}\right)^{[r_{dn}=0][a_{dn}=1]} \\ &\cdot\left(\sum_{v=1}^{V}\beta_{vdk}\mu_v\right)^{[r_{dn}=1][a_{dn}=0]}\left(\sum_{v=1}^{V}\beta_{vdk}(1-\mu_v)\right)^{[r_{dn}=0][a_{dn}=0]}\end{aligned} \tag{5.4.15}$$

$$\begin{aligned}q_n(k,\mathbf{x}_n^m) &= P(z_n=k,\mathbf{x}_n^m|\mathbf{x}_n^o,\mathbf{r}_n,\mathbf{a}_n,\theta,\beta,\mu) = \frac{P(z_n=k,\mathbf{x}_n^m,\mathbf{x}_n^o,\mathbf{r}_n|\mathbf{a}_n,\theta,\beta,\mu)}{P(\mathbf{x}_n^o,\mathbf{r}_n|\mathbf{a}_n,\theta,\beta,\mu)} \\ &= \frac{\theta_k\prod_{d=1}^{D}\prod_{v=1}^{V}(\mu_v\beta_{vdk})^{[r_{dn}=1][x_{dn}=v]}((1-\mu_v)\beta_{vdk})^{[r_{dn}=0][x_{dn}=v]}}{\sum_{k=1}^{K}\theta_k\prod_{d=1}^{D}\gamma_{dkn}}\end{aligned} \tag{5.4.16}$$

This posterior distribution has a number of terms that is again exponential in the number of missing data values. We form the expected complete log posterior shown below based on the missing data posterior distribution $q_n(k, \mathbf{x}_n^m)$ given in Equation 5.4.16, the complete likelihood given in Equation 5.4.8, and the prior distribution. As we will see, the required expectations are functions of at most two variables, and can be efficiently computed.

$$
\begin{aligned}
E[\log \mathcal{P}(\beta, \theta, \mu | \{\mathbf{x}_n, \mathbf{r}_n, \mathbf{a}_n\}_{n=1:N}, \alpha, \phi, \xi)] &= \log \Gamma(\sum_{k=1}^{K} \alpha_k) - \sum_{k=1}^{K} \log \Gamma(\alpha_k) + \sum_{k=1}^{K} (\alpha_k - 1) \log \theta_k \\
&+ \sum_{k=1}^{K} \sum_{d=1}^{D} \log \Gamma(\sum_{v=1}^{V} \phi_{vdk}) - \sum_{v=1}^{V} \log \Gamma(\phi_{vdk}) + \sum_{v=1}^{V} (\phi_{vdk} - 1) \log \beta_{vdk} \\
&+ \sum_{v=1}^{V} \log \Gamma(\xi_{0v} + \xi_{1v}) - \log \Gamma(\xi_{0v}) - \log \Gamma(\xi_{1v}) + (\xi_{1v} - 1) \log \mu_v + (\xi_{0v} - 1) \log(1 - \mu_v) \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{\mathbf{x}_n^m} q_n(k, \mathbf{x}_n^m)[z_n = k] \log \theta_k \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{\mathbf{x}_n^m} \sum_{d=1}^{D} \sum_{v=1}^{V} q_n(k, \mathbf{x}_n^m)[z_n = k][x_{dn} = v][r_{dn} = 1][a_{dn} = 1](\log \beta_{vdk} + \log \mu_v) \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{\mathbf{x}_n^m} \sum_{d=1}^{D} \sum_{v=1}^{V} q_n(k, \mathbf{x}_n^m)[z_n = k][x_{dn} = v][r_{dn} = 1][a_{dn} = 0](\log \beta_{vdk} + \log \mu_v) \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{\mathbf{x}_n^m} \sum_{d=1}^{D} \sum_{v=1}^{V} q_n(k, \mathbf{x}_n^m)[z_n = k][x_{dn} = v][r_{dn} = 0][a_{dn} = 0](\log \beta_{vdk} + \log(1 - \mu_v))
\end{aligned}
\tag{5.4.17}
$$

The expected complete log posterior can be simplified significantly. First, we note that the term $\sum_{\mathbf{x}_n^m} q_n(k, \mathbf{x}_n^m)[z_n = k] \log \theta_k$ depends only on k and not on $\mathbf{x}_n^m$. This is also true for the term $q_n(k, \mathbf{x}_n^m)[z_n = k][x_{dn} = v][r_{dn} = 1][a_{dn} = 1](\log \beta_{vdk} + \log \mu_v)$ since x_{dn} is not missing when $r_{dn} = 1$ and $a_{dn} = 1$. We can sum $q_n(k, \mathbf{x}_n^m)$ over $\mathbf{x}_n^m$ to obtain the simplified posterior term $q_n(k)$ as seen below. This yields the simplified factors of the form $q_n(k)[z_n = k] \log \theta_k$, and $q_n(k)[z_n = k][x_{dn} = v][r_{dn} = 1][a_{dn} = 1](\log \beta_{vdk} + \log \mu_v)$.

$$
q_n(k) = P(z_n = k | \mathbf{x}_n^o, \mathbf{r}_n, \mathbf{a}_n, \theta, \beta, \mu) = \frac{P(z_n = k, \mathbf{x}_n^o, \mathbf{r}_n | \mathbf{a}_n, \theta, \beta, \mu)}{P(\mathbf{x}_n^o, \mathbf{r}_n | \mathbf{a}_n, \theta, \beta, \mu)} = \frac{\theta_k \prod_{d=1}^{D} \gamma_{dkn}}{\sum_{k=1}^{K} \theta_k \prod_{d=1}^{D} \gamma_{dkn}}
$$

Terms where $[a_{dn} = 0]$ can also be simplified since they depend on exactly one missing data value. Assume that dimension d is missing, we can sum out over the remaining missing data dimensions to obtain a simplified posterior factor of the form $q_n(k, v, d)$ as seen in Equation 5.4.18. The value of $q_n(k, v, d)$ is simple to compute given the value of $q_n(k)$.

$$
\begin{aligned}
q_n(k,v,d) &= P(z_n = k, x_{dn} = v, \mathbf{x}_n^o, \mathbf{r}_n | \mathbf{a}_n, \theta, \beta, \mu) \\
&= q_n(k) \left(\frac{\mu_v \beta_{vdk}}{\sum_{v'=1}^{V} \mu_{v'} \beta_{v'dk}} \right)^{[r_{dn}=1][a_{dn}=0]} \left(\frac{(1-\mu_v)\beta_{vdk}}{\sum_{v'=1}^{V} (1-\mu_{v'})\beta_{v'dk}} \right)^{[r_{dn}=0][a_{dn}=0]}
\end{aligned} \tag{5.4.18}
$$

Another useful posterior probability is the probability that a missing rating takes value v. We denote this distribution by $q_n(v,d)$. This distribution can be obtained from $q_n(k,v,d)$ by marginalizing over k.

$$
q_n(v,d) = P(x_{dn} = v, \mathbf{x}_n^o, \mathbf{r}_n | \mathbf{a}_n, \theta, \beta, \mu) = \sum_{k=1}^{K} q_n(k,v,d)
$$

We rewrite the expected complete log posterior using the simplified posterior distributions. It is clear from this form that the expected complete log posterior depends only on a relatively small set of local posterior factors, all of which can be efficiently computed.

$$
\begin{aligned}
E[\log \mathcal{P}(\beta, \theta, \mu | \{\mathbf{x}_n, \mathbf{r}_n, \mathbf{a}_n\}_{n=1:N}, \alpha, \phi, \xi)] &= \log \Gamma(\sum_{k=1}^{K} \alpha_k) - \sum_{k=1}^{K} \log \Gamma(\alpha_k) + \sum_{k=1}^{K} (\alpha_k - 1) \log \theta_k \\
&+ \sum_{k=1}^{K} \sum_{d=1}^{D} \log \Gamma(\sum_{v=1}^{V} \phi_{vdk}) - \sum_{v=1}^{V} \log \Gamma(\phi_{vdk}) + \sum_{v=1}^{V} (\phi_{vdk} - 1) \log \beta_{vdk} \\
&+ \sum_{v=1}^{V} \log \Gamma(\xi_{0v} + \xi_{1v}) - \log \Gamma(\xi_{0v}) - \log \Gamma(\xi_{1v}) + (\xi_{1v} - 1) \log \mu_v + (\xi_{0v} - 1) \log(1 - \mu_v) \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} q_n(k) [z_n = k] \log \theta_k \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} q_n(k) \sum_{d=1}^{D} \sum_{v=1}^{V} [z_n = k][x_{dn} = v][r_{dn} = 1][a_{dn} = 1](\log \beta_{vdk} + \log \mu_v) \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{d=1}^{D} \sum_{v=1}^{V} q_n(k,v,d) [z_n = k][r_{dn} = 1][a_{dn} = 0](\log \beta_{vdk} + \log(\mu_v)) \\
&+ \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{d=1}^{D} \sum_{v=1}^{V} q_n(k,v,d) [z_n = k][r_{dn} = 0][a_{dn} = 0](\log \beta_{vdk} + \log(1 - \mu_v))
\end{aligned} \tag{5.4.19}
$$

Finally, we find the partial derivatives of the expected complete log posterior with respect to the multinomial parameters θ, β, and the Bernoulli parameters μ_v. We solve the result-

ing gradient equations using Lagrange multipliers to enforce normalization constraints. The derivation for θ is identical to the finite mixture case.

$$\begin{aligned}
\frac{\partial E[\log \mathcal{P}]}{\partial \theta_k} &= \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{\theta_k} - \lambda = 0 \\
\theta_k \lambda &= \alpha_k - 1 + \sum_{n=1}^{N} p_n(k) \\
\sum_{k=1}^{K} \theta_k \lambda &= \sum_{k=1}^{K} (\alpha_k - 1) + \sum_{n=1}^{N} q_n(k) \\
\lambda &= N - K + \sum_{k=1}^{K} \alpha_k \\
\theta_k &= \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{N - K + \sum_{k=1}^{K} \alpha_k}
\end{aligned} \tag{5.4.20}$$

The update for β_{vdk} takes into account both the expected number of observed ratings of value v explained by component k, and the expected number of missing ratings of value v explained by component k.

$$\begin{aligned}
\frac{\partial E[\log \mathcal{P}]}{\partial \beta_{vdk}} &= \frac{\phi_{vnk} - 1}{\beta_{vdk}} + \frac{\sum_{n=1}^{N} q_n(k)[a_{dn} = 1][x_{dn} = v]}{\beta_{vdk}} \\
&\quad + \frac{\sum_{n=1}^{N} q_n(v, d, k)[a_{dn} = 0]}{\beta_{vdk}} - \lambda = 0
\end{aligned} \tag{5.4.21}$$

$$\begin{aligned}
\beta_{vdk} \lambda &= \phi_{vnk} - 1 + \sum_{n=1}^{N} q_n(k)[a_{dn} = 1][x_{dn} = v] + q_n(k, v, d)[a_{dn} = 0] \\
\lambda &= \sum_{v=1}^{V} (\phi_{vnk} - 1) + \sum_{n=1}^{N} q_n(k) \\
\beta_{vdk} &= \frac{\phi_{vdk} - 1 + \sum_{n=1}^{N} q_n(k)[a_{dn} = 1][x_{dn} = v] + q_n(k, v, d)[a_{dn} = 0]}{\sum_{n=1}^{N} q_n(k) - V + \sum_{v=1}^{V} \phi_{vdk}}
\end{aligned} \tag{5.4.22}$$

The update for μ_v is given by the ratio of the number of observed ratings with value v to the total number of observed and expected ratings with value v.

$$\begin{aligned}
\frac{\partial E[\log \mathcal{P}]}{\partial \mu_v} &= \frac{\xi_{1v} - 1}{\mu_v} + \frac{\sum_{n=1}^{N} \sum_{d=1}^{D} [r_{dn} = 1][a_{dn} = 1][x_{dn} = v] + q_n(v, d)[r_{dn} = 1][a_{dn} = 0]}{\mu_v} \\
&\quad - \frac{\xi_{0v} - 1}{(1 - \mu_v)} - \frac{\sum_{n=1}^{N} \sum_{d=1}^{D} q_n(v, d)[r_{dn} = 0][a_{dn} = 0]}{(1 - \mu_v)} = 0
\end{aligned}$$

$$= (1-\mu_v)\left(\xi_{1v} - 1 + \sum_{n=1}^{N}\sum_{d=1}^{D}[r_{dn}=1][x_{dn}=v] + q_n(v,d)[r_{dn}=1][a_{dn}=0]\right)$$
$$- \mu_v\left(\xi_{0v} - 1 + \sum_{n=1}^{N}\sum_{d=1}^{D} q_n(v,d)[r_{dn}=0][a_{dn}=0]\right) = 0$$
$$\mu_v = \frac{\xi_{1v} - 1 + \sum_{n=1}^{N}\sum_{d=1}^{D}[r_{dn}=1][x_{dn}=v] + q_n(v,d)[r_{dn}=1][a_{dn}=0]}{\xi_{1v} + \xi_{0v} - 2 + \sum_{n=1}^{N}\sum_{d=1}^{D}[r_{dn}=1][x_{dn}=v] + q_n(v,d)[a_{dn}=0]} \quad (5.4.23)$$

5.4.3 Rating Prediction

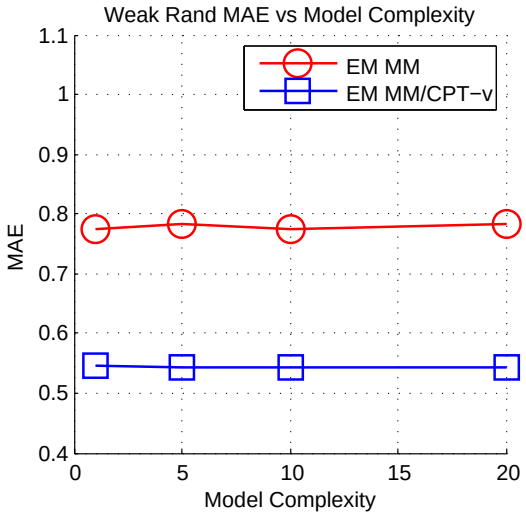
The posterior predictive distribution is given below assuming that x_{dn} is missing. Note that the predictive distribution differs depending on whether x_{dn} was originally selected and rated by the user or not.

$$P(X_{dn}=v|\mathbf{x}_n^o, \mathbf{r}_n, \mathbf{a}_n, \theta, \beta, \mu) = \sum_{k=1}^{K} \frac{\theta_k \prod_{d=1}^{D}\gamma_{dkn}}{\sum_{k=1}^{K}\theta_k\prod_{d=1}^{D}\gamma_{dkn}} \left(\frac{\mu_v\beta_{vdk}}{\sum_{v'=1}^{V}\mu_{v'}\beta_{v'dk}}\right)^{[r_{dn}=1]}$$
$$\cdot \left(\frac{(1-\mu_v)\beta_{vdk}}{\sum_{v'=1}^{V}(1-\mu_{v'})\beta_{v'dk}}\right)^{[r_{dn}=0]} \quad (5.4.24)$$

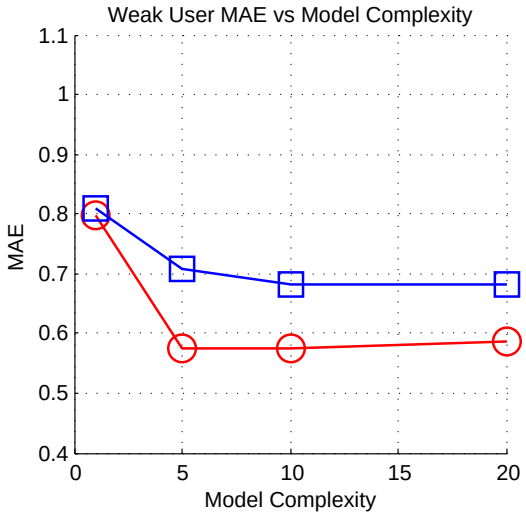
5.4.4 Experimentation and Results

The standard training and testing protocols described in Sections 5.2.1 and 5.1.3 were applied to both the finite multinomial mixture model learned using EM assuming MAR (EM MM), and the finite multinomial mixture/CPT-v model learned using EM (EM MM/CPT-v). A maximum of 5000 EM iterations were used. General smoothing priors were used for both models. The prior parameters on β_{dk} were set to $\phi_{vkd} = 2$ for all d and k in both models. The prior parameters on θ were set to $\alpha_k = 2$ for all k in both models. The prior parameters on μ_v were set to $\xi_{0v} = \xi_{1v} = 2$ for all v. Models with 1, 5, 10, and 20 components were trained on the Yahoo! data. The Jester results are based on models with 5 components.

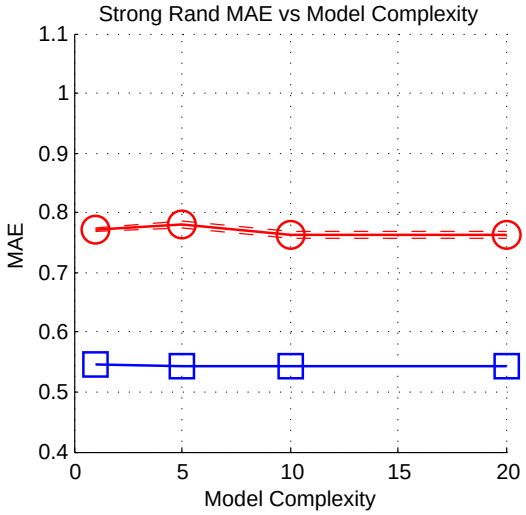
Figures 5.6(a) to 5.6(d) show the performance of the finite Bayesian multinomial mixture model (EM MM) compared to the performance of the combined finite multinomial mixture/CPT-v model (EM MM/CPT-v) on the Yahoo! data set. The horizontal axis represents the number of mixture components. The vertical axis represents normalized mean absolute error. The solid lines represent the mean of the error over the five cross validation folds for each model. The standard error of the mean is shown using dashed lines. The rating prediction results show that the MM/CPT-v model outperforms the MM model on randomly selected items over the complete range of model complexity values, while the MM model out-performs the MM/CPT-v on user-selected items.



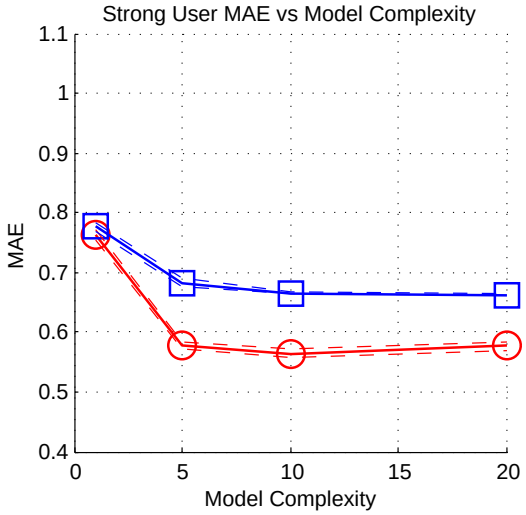
(a) Weak generalization on randomly selected items from the Yahoo! data set.



(b) Weak generalization on user selected items from the Yahoo! data set.

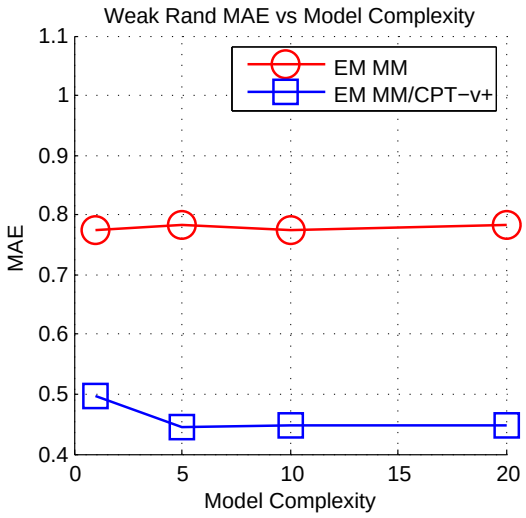


(c) Strong generalization on randomly selected items from the Yahoo! data set.

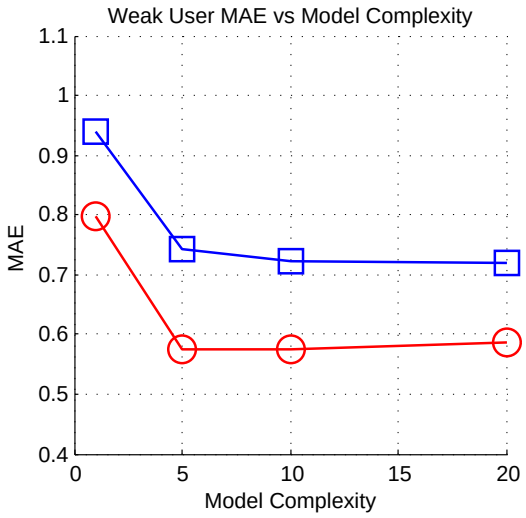


(d) Strong generalization on user selected items from the Yahoo! data set.

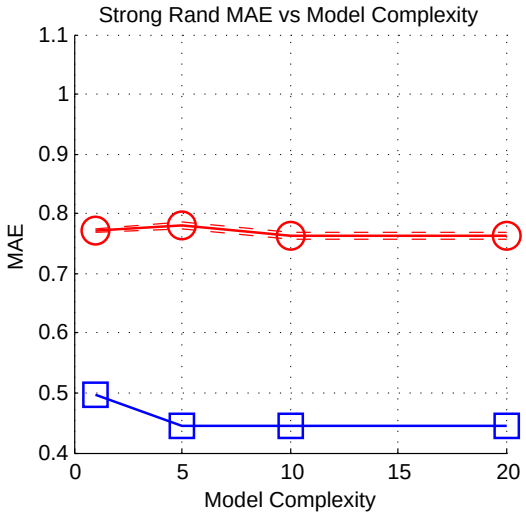
Figure 5.6: Figures 5.6(a) to 5.6(d) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/CPT-v model learned using EM (EM MM/CPT-v) on the Yahoo! data set. The horizontal axis represents the number of mixture components. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/CPT-v model vastly outperforms the MM model on randomly selected items over a range of model complexity values.



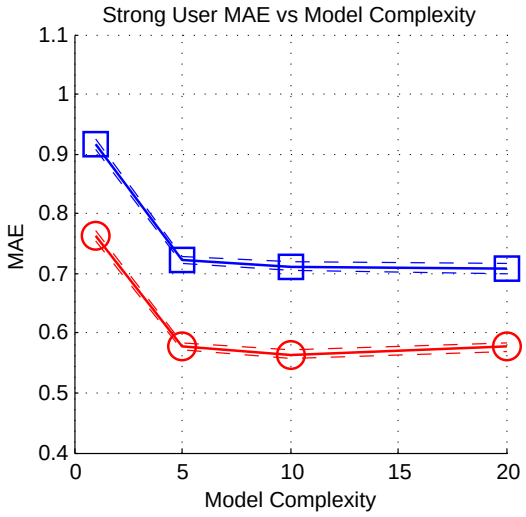
(a) Weak generalization on randomly selected items from the Yahoo! data set.



(b) Weak generalization on user selected items from the Yahoo! data set.

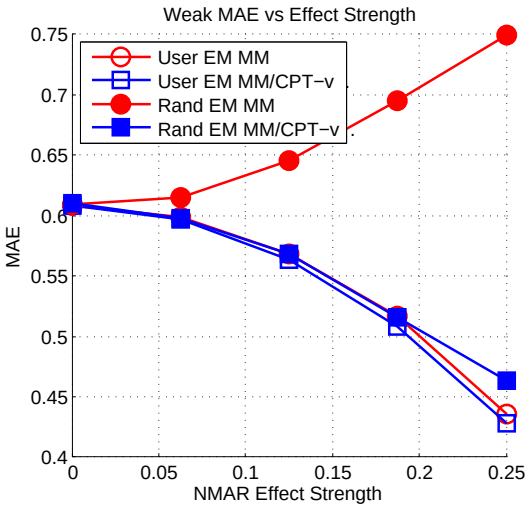


(c) Strong generalization on randomly selected items from the Yahoo! data set.

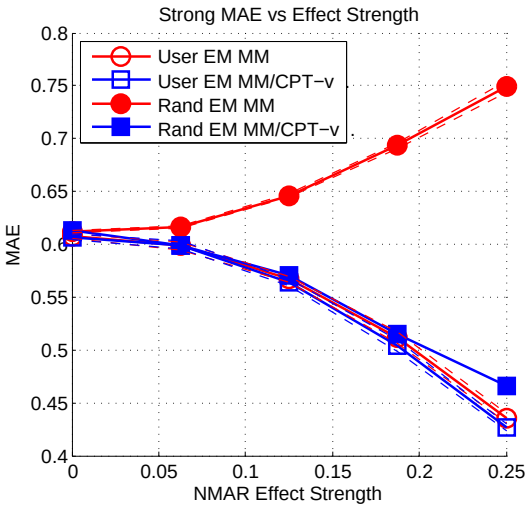


(d) Strong generalization on user selected items from the Yahoo! data set.

Figure 5.7: Figures 5.7(a) to 5.7(d) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/CPT-v model learned using EM (EM MM/CPT-v+) on the Yahoo! data set. In this case the CPT-v μ parameters are learned using an additional held-out sample of ratings for user and randomly selected item. The horizontal axis represents the number of mixture components. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/CPT-v+ model vastly outperforms the MM model on randomly selected items over a range of model complexity values.



(a) Weak generalization on user and randomly selected items from the Jester data set.



(b) Strong generalization on user and randomly selected items from the Jester data set.

Figure 5.8: Figures 5.8(a) and 5.8(b) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/CPT-v model learned using EM (EM MM/CPT-v) on the Jester data. The horizontal axis represents the strength of the synthetic non-random missing data effect applied to the Jester data set. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/CPT-v model vastly outperforms the MM model on randomly selected items when there is a strong non-random missing data effect.

Recall that our goal is to obtain an accurate estimate of rating prediction performance over the whole data matrix. The error on randomly selected items provides an unbiased estimate of the error on the complete data matrix. Under this criteria the MM/CPT-v model performs significantly better than the MM model under the MAR assumption. By contrast the error on user-selected items reflects the selection bias in the user-selected ratings, and is clearly not a reliable estimate of prediction performance on the complete data matrix.

To asses the value of a small sample of ratings for randomly selected items, ratings from the set of holdout users described in section 5.1.3 were used to obtain a direct estimate of the CPT-v μ parameters. The MM/CPT-v model was then trained with the μ parameters fixed to their estimated values. The estimator for μ_v is given below. Recall that the hold out set consists both of ratings for user-selected items, and ratings for randomly selected items so that the required probabilities can be computed directly.

$$\mu_v = \frac{P(x = v|r = 1)P(r = 1)}{P(x = v|r = 1)P(r = 1) + P(x = v|r = 0)P(r = 0)} \tag{5.4.25}$$

$$P(r=1) \approx (1/DN) \sum_{n=1}^{N} \sum_{d=1}^{D} [r_{dn}=1] \tag{5.4.26}$$

$$P(r=0) \approx (1/DN) \sum_{n=1}^{N} \sum_{d=1}^{D} [r_{dn}=0] \tag{5.4.27}$$

$$P(x=v|r=1) \approx \frac{\sum_{n=1}^{N} \sum_{d=1}^{D} [x_{dn}=v][r_{dn}=1][a_{dn}=1]}{\sum_{n=1}^{N} \sum_{d=1}^{D} [r_{dn}=1][a_{dn}=1]} \tag{5.4.28}$$

$$P(x=v|r=0) \approx \frac{\sum_{n=1}^{N} \sum_{d=1}^{D} [x_{dn}=v][r_{dn}=0][a_{dn}=1]}{\sum_{n=1}^{N} \sum_{d=1}^{D} [r_{dn}=1][a_{dn}=0]} \tag{5.4.29}$$

Figures 5.7(a) to 5.7(d) give a comparison between CPT-v trained using this modified protocol (EM MM/CPT-v+), and the baseline multinomial mixture model assuming MAR (EM MM). The rating prediction results again show that the MM/CPT-v+ model vastly outperforms the MM model on randomly selected items. Learning the CPT-v parameters on held-out data also yields significantly better performance than jointly learning the CPT-v parameters and mixture model parameters using the EM algorithm. This demonstrates that a small number of ratings for randomly selected items can yield a significant benefit.

Figures 5.8(a) and 5.8(b) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/CPT-v model learned using EM (EM MM/CPT-v) on the Jester data. The standard training protocol where both missing data parameters and mixture model parameters are jointly learned was used for the MM/CPT-v model. The horizontal axis represents the strength of the synthetic non-random missing data effect applied to the Jester data set. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/CPT-v model vastly outperforms the MM model on randomly selected items when there is a strong non-random missing data effect. The two models perform similarly when missing data is missing at random.

It is interesting to note that the error on randomly-selected items actually goes down in all of these experiments as the effect strength increases. This is due to the fact that we condition on the value of r_{dn} when making predictions for x_{dn}. As the effect strength increases the prediction problem become easier because the value of r_{dn} carries a significant amount of information about the value of x_{dn}. The error on user-selected items shows a similar trend for the MM model, which assuming MAR . The explanation for the MM model is that the distribution of ratings for user-selected items becomes increasingly concentrated on high rating values as the effect strength increases. The user-selected rating values thus become easier to predict.

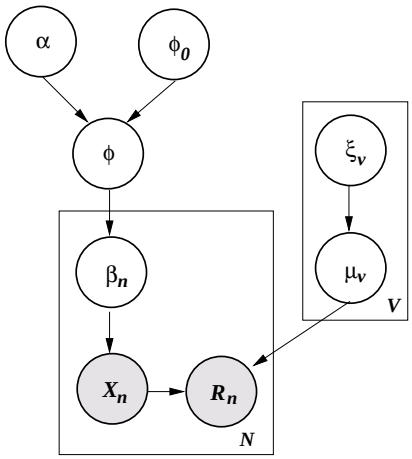


Figure 5.9: Bayesian network for the Dirichlet Process mixture/CPT-v model combination.

5.5 The Dirichlet Process Mixture/CPT-v Model

One issue with the Bayesian Mixture/CPT-v model combination is the difficulty in choosing the number of clusters K. In this section we show how the CPT-v missing data model can be combined with a Dirichlet Process mixture model. The combined model is illustrated in figure 5.9. The probability model is given in Equations 5.5.1 to 5.5.5.

$$P(\phi|\phi_0,\alpha) = \mathcal{DP}(\alpha\phi_0) \tag{5.5.1}$$

$$P(\beta_n|\phi) = \phi(\beta_n) \tag{5.5.2}$$

$$P(\mu|\xi) = \prod_v \mathcal{B}(\mu_v|\xi_v) \tag{5.5.3}$$

$$P(\mathbf{X}=\mathbf{x}_n|\beta_n) = \prod_{d=1}^{D}\prod_{v=1}^{V} \beta_{vdk}^{[x_{dn}=v]} \tag{5.5.4}$$

$$P(\mathbf{R}=\mathbf{r}_n|\mathbf{X}=\mathbf{x}_n,\mu) = \prod_{d=1}^{D}\prod_{v=1}^{V} \mu_v^{[r_{dn}=1][x_{dn}=v]}(1-\mu_v)^{[r_{dn}=0][x_{dn}=v]} \tag{5.5.5}$$

Unlike the Dirichlet process mixture model under the missing at random assumption, summing over the missing data in the Dirichlet Process mixture/CPT-v model renders the parameter updates non-conjugate. Given a partition consisting of K clusters, the posterior distribution on β_{dk} is shown in Equation 5.5.6. It is not difficult to see that this posterior distribution is non-Dirichlet.

$$
\begin{aligned}
&P(\beta_{dk} | \{\mathbf{x}_n, \mathbf{r}_n, \mathbf{a}_n, \mathbf{z}_n\}_{n=1:N}, \mu, \phi_0) \\
&\quad \propto \prod_{n=1}^{N} \left(\prod_{v=1}^{V} \beta_{vdk}^{[x_{dn}=v][r_{dn}=1][a_{dn}=1][z_n=k]} \right) \cdot \left(\sum_{v=1}^{V} \mu_v \beta_{vdk} \right)^{[r_{dn}=1][a_{dn}=0][z_n=k]} \\
&\quad \cdot \left(\sum_{v=1}^{V} (1-\mu_v) \beta_{vdk} \right)^{[r_{dn}=0][a_{dn}=0][z_n=k]} \left(\prod_{v=1}^{V} \beta_{vdk}^{\phi_{vd0}-1} \right)
\end{aligned} \tag{5.5.6}
$$

The straightforward approach to avoiding non-conjugate parameter sampling is to sample all of the missing data values given the current partition and parameter values. With the missing data values included in the state of the Gibbs sampler, the Gibbs updates consist of sampling from Dirichlet, Beta, or discrete distributions. While simple, this sampling scheme is not practical in the collaborative filtering case. First, sampling all of the missing data means that the space complexity of the inference algorithm scales with the size of the complete data matrix instead of the number of observed ratings. Second, introducing millions of additional state variables into the Gibbs sampler has a severely detrimental affect on mixing time, rendering the naive sampler effectively useless.

5.5.1 An Auxiliary Variable Gibbs Sampler

In this section we describe an exact auxiliary variable sampler for the Dirichlet Process mixture/CPT-v model. The sampler exploits two key properties of the combined model. First, the missing data can be analytically summed over when assigning a data case to a mixture component. This includes the case of assignment to a previously unrepresented component. Second, the parameter updates for β_{vdk} and μ_v depend only on counts of observed and missing rating values. It is possible to exactly sample the missing rating counts as auxiliary variables without simultaneously storing the missing ratings for each user.

We give the cluster assignment updates in Equations 5.5.7 and 5.5.9. Equation 5.5.7 gives the probability that data case n is assigned to occupied cluster k. This probability can be viewed as the product of a prior term, and a likelihood term. The prior component of the update probability follows from the Chinese Restaurant Process, and depends only on the concentration parameter, and the number of data cases assigned to each occupied component. The likelihood component of equation 5.5.7 is obtained by summing over missing data values as in the finite mixture case.

$$
\begin{aligned}
&\text{If } z_j = z_n = k \text{ for some } j \neq n \; P(z_n = k | z_{-n}, \mathbf{x}_n^o, \mathbf{r}_n, \mathbf{a}_n, \beta, \alpha, \mu) \\
&\quad \propto P(z_n = k | z_{-n}, \alpha) P(\mathbf{x}_n^o, \mathbf{r}_n | \mathbf{a}_n, \beta_k, \mu) \\
&\quad \propto P(z_n = k | z_{-n}, \alpha) \sum_{\mathbf{x}_n^m} P(\mathbf{x}_n^o, \mathbf{x}_n^m, \mathbf{r}_n | \mathbf{a}_n, \beta_k, \mu) \\
&\quad \propto \frac{\sum_{i \neq n}^{N} [z_i = k]}{N - 1 + \alpha} \prod_{d=1}^{D} \left(\prod_{v=1}^{V} \beta_{vdk}{}^{[x_{dn}=v]} \right)^{[r_{dn}=1][a_{dn}=1]} \left(\sum_{v=1}^{V} \mu_v \beta_{vdk} \right)^{[r_{dn}=1][a_{dn}=0]} \\
&\qquad \cdot \left(\sum_{v=1}^{V} (1 - \mu_v) \beta_{vdk} \right)^{[r_{dn}=0][a_{dn}=0]}
\end{aligned} \tag{5.5.7}
$$

$$
\begin{aligned}
&P(z_n \neq z_j \;\; \forall \;\; j \neq n | z_{-n}, \mathbf{x}_n^o, \mathbf{r}_n, \mathbf{a}_n, \alpha, \mu, \phi_0) \\
&\quad \propto P(z_n \neq z_j \;\; \forall \;\; j \neq n \;\; | z_{-n}, \alpha) \int \sum_{\mathbf{x}_n^m} P(\mathbf{x}_n^o, \mathbf{x}_n^m, \mathbf{r}_n | \mathbf{a}_n, \beta, \mu) P(\beta | \phi_0) d\beta \qquad (5.5.8) \\
&\quad \propto \frac{\alpha}{N - 1 + \alpha} \prod_{d=1}^{D} \left(\prod_{v=1}^{V} \frac{\phi_{vd0}}{\sum_v \phi_{vd0}}{}^{[x_{dn}=v]} \right)^{[r_{dn}=1][a_{dn}=1]} \left(\sum_{v=1}^{V} \mu_v \frac{\phi_{vd0}}{\sum_v \phi_{vd0}} \right)^{[r_{dn}=1][a_{dn}=0]} \\
&\qquad \cdot \left(\sum_{v=1}^{V} (1 - \mu_v) \frac{\phi_{vd0}}{\sum_v \phi_{vd0}} \right)^{[r_{dn}=0][a_{dn}=0]} \qquad (5.5.9)
\end{aligned}
$$

Equation 5.5.9 gives the probability that data case n is assigned to a new cluster. This probability can again be viewed as the product of a prior term, and a likelihood term. The prior component of the update probability again follows from the Chinese Restaurant process, and depends only on the concentration parameter. The likelihood component of equation 5.5.9 is obtained by simultaneously summing over missing data values, and integrating over β with respect to the base distribution ϕ_0. The multi-dimensional integral in Equation 5.5.8 is an expectation with respect to the base distribution, which is itself factorized over data dimensions d. Due to independence of the data dimensions, the expectation of the product over dimensions d is equal to the product of expectations. If the value of dimension d is observed to be v, then the expected value simplifies to $E_{\phi_0}[\beta_{vd}]$. If the value of dimension d is not observed, but dimension d was originally rated, the expected value simplifies to $E_{\phi_0}[\sum \mu_v \beta_{vd}]$. By linearity of expectation this gives $\sum \mu_v E_{\phi_0}[\beta_{vd}]$. If the value of dimension d is not observed, and dimension d was originally not rated then the expected value simplifies to $E_{\phi_0}[\sum (1 - \mu_v) \beta_{vd}]$. By Linearity of expectation this gives $\sum (1 - \mu_v) E_{\phi_0}[\beta_{vd}]$. Equation 5.5.9 is obtained by using the Dirichlet expectation $E_{\phi_0}[\beta_{vd}] = \phi_{vd0} / \sum_v \phi_{vd0}$.

The updates given in Equations 5.5.7 and 5.5.9 clearly show that the cluster assignments

can be updated without sampling the missing data values. As mentioned at the start of this section, the updates for β and μ are much easier if we sample the missing data values for each data case. The drawback is that the space complexity of the inference method then depends on the size of the data matrix instead of the number of observed entries.

To overcome this problem we introduce an exact auxiliary variable Gibbs sampling scheme. The auxiliary variables consist of counts C_{vdk00} and C_{vdk10}. C_{vdk00} represents the number of data cases where $x_{dn} = v$, $z_n = k$, $r_{dn} = 0$, and $a_{dn} = 0$. This is the number of data cases assigned to cluster k where dimension d was not rated by the user and the rating is not available, but the underlying value is believed to be v. C_{vdk10} represents the number of data cases where $x_{dn} = v$, $z_n = k$, $r_{dn} = 1$, and $a_{dn} = 0$. This is the number of data cases assigned to cluster k where dimension d was rated by the user, the rating is not available, but the underlying value is believed to be v. Given the current partition and parameter values, the variables C_{vdk00} and C_{vdk10} are multinomial-distributed as seen in Equations 5.5.10 and 5.5.11. We define N_{dk00} to be the total number of data cases assigned to cluster k where dimension d was not rated, and is not observed. N_{dk10} is defined to be the total number of data cases assigned to cluster k where dimension d was rated by the user, but the rating is not available.

$$\begin{aligned} P(C_{dk00} = c_{dk00}|\mathbf{z}, \mathbf{x}^o, \mathbf{r}, \mathbf{a}, \beta, \mu) &= \frac{N_{dk00}!}{\prod_{v=1}^{V} c_{vdk00!}} \prod_{v=1}^{V} P(x_d = v|r_d = 0, a_d = 0, z_n = k)^{c_{vdk00}} \\ &= \frac{N_{dk00}!}{\prod_{v=1}^{V} c_{vdk00!}} \prod_{v=1}^{V} \left(\frac{(1-\mu_v)\beta_{vmk}}{\sum_{v=1}^{V}(1-\mu_v)\beta_{vmk}} \right)^{c_{vdk00}} \end{aligned} \tag{5.5.10}$$

$$\begin{aligned} P(C_{dk10} = c_{dk10}|\mathbf{z}, \mathbf{x}^o, \mathbf{r}, \mathbf{a}, \beta, \mu) &= \frac{N_{dk10}!}{\prod_{v=1}^{V} c_{vdk10!}} \prod_{v=1}^{V} P(x_d = v|r_d = 1, a_d = 0, z_n = k)^{c_{vdk10}} \\ &= \frac{N_{dk10}!}{\prod_{v=1}^{V} c_{vdk10!}} \prod_{v=1}^{V} \left(\frac{\mu_v\beta_{vmk}}{\sum_{v=1}^{V}\mu_v\beta_{vmk}} \right)^{c_{vdk10}} \end{aligned} \tag{5.5.11}$$

Sampling a value for c_{1dk00} to c_{Vdk00} is accomplished by independently drawing N_{dk00} samples from the distribution $P(x_d = v|r_d = 0, a_d = 0, z_n = k)$, and counting the number of samples with each rating value v. Drawing values for c_{1dk10} to c_{Vdk10} is similar. It is clear that the storage space required to sample all of the auxiliary variables is independent of the number of missing data values, while the amount of computation is proportional to the number of missing data values. We note that drawing M samples from the same discrete distribution can be more efficient than drawing one sample from each of M different discrete distributions. Thus, it is preferable to sample missing values corresponding to a particular dimension and cluster at the same time, instead of sampling all missing values for a particular data case.

To derive the updates for β, we assume that all missing data values have been sampled. As we see in Equation 5.5.12, the update for β_{dk} reduces to a function of the auxiliary count variables c_{vdk00}, c_{vdk10}, and the observed counts $c_{vdk11} = \sum_n [z_n = k][x_{dn} = v][r_{dn} = 1][a_{dn} = 1]$.

$$\begin{aligned} P(\beta_{vdk}|\mathbf{z}, \mathbf{x}^o, \mathbf{x}^m, \mathbf{r}, \phi_0) &\propto \prod_{n=1}^{N} P(x_{dn}|\beta_{dk}) P(\beta_{dk}|\phi_{d0}) \\ &\propto \prod_{n=1}^{N} \prod_{v=1}^{V} \beta_{vdk}{}^{[r_{dn}=1][x_{dn}=v][z_n=k]+[r_{dn}=0][x_{dn}=v][z_n=k]} \cdot \prod_{v=1}^{V} \beta_{vdk}^{\phi_{vd0}-1} \\ &\propto \prod_{n=1}^{N} \prod_{v=1}^{V} \beta_{vdk}{}^{c_{vdk11}+c_{vdk10}+c_{vdk00}+\phi_{vd0}-1} \\ &= \mathcal{D}(c_{vdk11} + c_{vdk10} + c_{vdk00} + \phi_{vd0}) \end{aligned} \tag{5.5.12}$$

We assume that all missing values have been sampled, and derive the update for μ_v. We find that the updates depend on the counts variables as seen in Equation 5.5.13.

$$\begin{aligned} P(\mu_v|\mathbf{x}^o, \mathbf{x}^m, \mathbf{r}, \xi) &\propto \prod_{n=1}^{N} \prod_{d=1}^{D} \mu_v^{[r_{dn}=1][x_{dn}=v]} (1-\mu_v)^{[r_{dn}=0][x_{dn}=v]} \cdot \mu_v^{\eta_{1v}-1} (1-\mu_v)^{\eta_{0v}-1} \\ &\propto \mu_v^{\eta_{1v}-1+\sum_{n=1}^{N}\sum_{d=1}^{D}[r_{dn}=1][x_{dn}=v]} (1-\mu_v)^{\eta_{0v}-1+\sum_{n=1}^{N}\sum_{n=1}^{N}[r_{dn}=0][x_{dn}=v]} \\ &\propto \mu_v^{\eta_{1v}-1+\sum_{k=1}^{K}\sum_{d=1}^{D} c_{vdk11}+c_{vdk10}} (1-\mu_v)^{\eta_{0v}-1+\sum_{k=1}^{K}\sum_{d=1}^{D} c_{vdk00}} \\ &= \mathcal{B}(\eta_{1v} + \sum_{k=1}^{K}\sum_{d=1}^{D} c_{vdk11} + c_{vdk10}, \eta_{0v} + \sum_{k=1}^{K}\sum_{d=1}^{D} c_{vdk00}) \end{aligned} \tag{5.5.13}$$

The complete Gibbs sampler consists of updating the cluster assignment for each data case n. These updates are based on observed data values, response indicator values, and current parameter values. Next, the auxiliary count variables are sampled from their exact posterior distribution given the current clustering and parameter values. The parameter values β and μ are updated using the auxiliary count variables and the auxiliary count variables are then discarded. The fact that the cluster assignment step does not depend on the auxiliary count variables allows the Gibbs sampler to mix much more efficiently than the naive Gibbs sampler.

5.5.2 Rating Prediction for Training Cases

Rating prediction for training cases is straightforward. Assuming that x_{dn} is missing, the posterior predictive distribution is obtained by averaging across samples. The desired posterior predictive distribution is $P(x_{dn} = v|\mathbf{r}_n, \mathbf{x}_n^o, \mathbf{a}_n, \alpha, \phi_0, \eta)$. We assume that a total of S samples

have been obtained from the above Gibbs sampler. We denote the parameter samples by β^s_{vmk}, μ^s_v, the cluster indicator samples by z^s_n, and the number of cluster by K^s. As in the finite mixture case, the posterior predictive distribution contains two cases depending on whether x_{dn} was originally rated by the user or not. Note that we assume the data dimensions we make predictions for are not observed dimensions.

$$P(x_{dn} = v|\{\mathbf{x}^o_i, \mathbf{r}_i, \mathbf{a}_i\}_{i=1:N}, \alpha, \phi_0, \eta) \approx \frac{1}{S}\sum_{s=1}^{S} P(x_{dn} = v|r_{dn}, z^s_n, \beta^s, \mu^s)$$
$$= \frac{1}{S}\sum_{s=1}^{S}\sum_{k=1}^{K^s}[z^s_n = k]\left(\frac{\mu_v\beta_{vdk}}{\sum_{v=1}^{V}\mu_v\beta_{vdk}}\right)^{[r_{dn}=0]}\left(\frac{(1-\mu_v)\beta_{vdk}}{\sum_{v=1}^{V}(1-\mu_v)\beta_{vdk}}\right)^{[r_{dn}=0]} \quad (5.5.14)$$

5.5.3 Rating Prediction for Novel Cases

To make predictions for novel data cases we introduce a set of pseudo mixture proportions $\hat{\theta}^s$ for each sample s. $\hat{\theta}^s_k$ is equal to the probability under the Chinese Restaurant Process of assigning a new data point to occupied cluster k for $k \leq K^S$. $\hat{\theta}^s_{K+1}$ is the probability under the Chinese Restaurant process of assigning a new data point to a new cluster.

We also introduce a set of pseudo parameters $\hat{\beta}^s$ for the mixture component distributions. For $k \leq K^s$, we set $\hat{\beta}^s_{vdk} = \beta^s_{vdk}$. For $k = K^s + 1$ we set $\hat{\beta}^s_{vdk}$ to the probability that $x_{dn} = v$ under the base distribution ϕ_0, $P(x_{dn} = v|\phi_0)$. This is the likelihood term used to compute the posterior probability that a data case should be assigned to a new component.

$$\hat{\theta}^s_k = \begin{cases} \frac{\sum_{n=1}^{N}[z_n=k]}{N+\alpha} & \ldots\ k \leq K^s \\ \frac{\alpha}{N+\alpha} & \ldots\ k = K^s+1 \end{cases} \qquad \hat{\beta}^s_{vdk} = \begin{cases} \beta^s_{vdk} & \ldots\ k \leq K^s \\ \frac{\phi_{vd0}}{\sum_{v=1}^{V}\phi_{vd0}} & \ldots\ k = K^s+1 \end{cases} \quad (5.5.15)$$

The parameters $\hat{\theta}^s$, and $\hat{\beta}^s$, and μ^s define a finite mixture/CPT-v model with $K^s + 1$ components for each sample s. We can compute the approximate predictive distribution for novel data cases by averaging the predictive distribution under each model. Computing the predictive distribution under each model is identical to the finite mixture/CPT-v case.

$$P(x_{d*} = v|\mathbf{r}_*, \mathbf{x}^o_*, \mathbf{a}_*, \{\mathbf{x}_n, \mathbf{r}_n, \mathbf{a}_n\}_{n=1:N}, \alpha, \phi, \eta) \approx \frac{1}{S}\sum_{s=1}^{S} P(x_{d*} = v|\mathbf{r}_{d*}, \mathbf{x}_{d*}, \mathbf{a}_{d*}, \hat{\theta}^s, \hat{\beta}^s, \mu^s) \quad (5.5.16)$$

	Weak Rand	Weak User	Strong Rand	Strong User
MCMC DP	0.7658 ± 0.0031	0.5735 ± 0.0004	0.7624 ± 0.0063	0.5767 ± 0.0077
MCMC DP/CPT-v	0.5548 ± 0.0037	0.6798 ± 0.0049	0.5549 ± 0.0026	0.6670 ± 0.0071
MCMC DP/CPT-v+	0.4421 ± 0.0008	0.7814 ± 0.0082	0.4428 ± 0.0027	0.7537 ± 0.0026

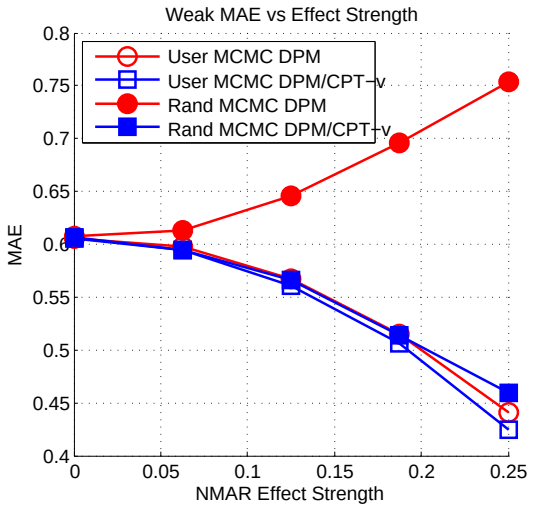
Table 5.1: This table compares rating prediction performance for methods based on the Dirichlet process mixture model. Results are given in terms of normalized mean absolute error. Model variations include the basic DP multinomial mixture model assuming MAR (MCMC DP), the combined DP multinomial mixture/CPT-v model (MCMC DP/CPT-v), and the combined DP multinomial mixture/CPT-v model with CPT-v μ parameters estimated based on a held-out sample of ratings for user-selected and randomly selected items (MCMC DP/CPT-v+). Results on randomly selected items show that the DP/CPT-v combination significantly outperforms the basic DP mixture, which assumes MAR. Further, DP/CPT-v+ significantly out-performs DP/CPT-v indicating the benefit of a held-out sample of ratings for randomly selected items.

5.5.4 Experimentation and Results

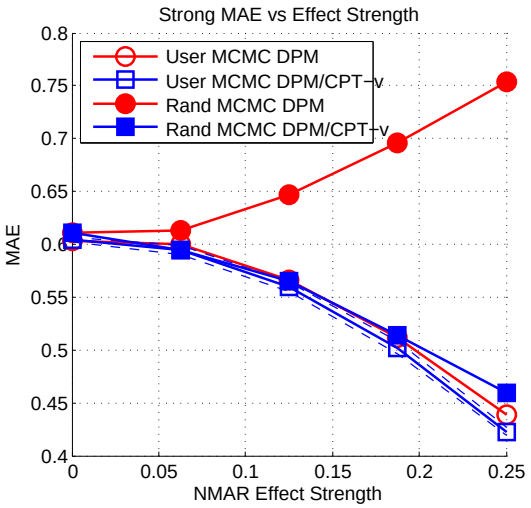
The training and testing protocols described in Sections 5.1.3 and 5.2.1 were applied to both the Dirichlet process multinomial mixture model assuming MAR (MCMC DP), and the Dirichlet process multinomial mixture/CPT-v combination (MCMC DP/CPT-v). Sampling in the basic Dirichlet process mixture was performed using the collapsed Gibbs sampler described in Section 4.2.3. Sampling in the DP/CPT-v model was performed using the auxiliary variable method described in this section. A total of 1000 Gibbs iterations were performed in each model and a fixed burn-in of 100 iterations was used. Following the burn in period, one sample was recorded every ten Gibbs iterations. As in the finite mixture experiments, general smoothing priors were used for both models. The prior parameters on β_{dk} were set to $\phi_{vkd} = 2$ for all d and k in both models. The concentration parameter was set to $\alpha_k = 2$ for all k in both models. The prior parameters on μ_v were set to $\xi_{0v} = \xi_{1v} = 2$ for all v. Strong generalization performance was assessed using the approximate prediction rule for novel data cases.

Table 5.1 summarizes the performance of the DP/CPT-v model compared to the basic DP model assuming MAR on the Yahoo! data set in terms of normalized mean absolute error. The results again show a substantial decrease in error using the CPT-v missing data model. We again used ratings from the set of holdout users to estimate the μ_v parameters directly as described in the finite mixture case. The μ_v parameters were then fixed, and sampling was carried out for the remaining parameters in the DP/CPT-v model. The results are also summarized in Table 5.1 (MCMC DP/CPT-v+), and again show a significant increase in performance when the μ parameters are estimated using held out ratings for randomly selected items relative to sampling for μ along with the rest of the model parameters.

Figures 5.10(a) and 5.10(b) show the performance of the multinomial DP mixture model assuming MAR compared to the performance of the combined finite DP multinomial mixture/CPT-



(a) Weak generalization on user and randomly selected items from the Jester data set.



(b) Strong generalization on user and randomly selected items from the Jester data set.

Figure 5.10: Figures 5.10(a) and 5.10(b) show the performance of the DP multinomial mixture model learned using the collapsed Gibbs sampler assuming MAR (MCMC DP) compared to the performance of the combined DP multinomial mixture/CPT-v model learned using the Gibbs sampler (MCMC DP/CPT-v) on the Jester data set. The vertical axis represents normalized mean absolute error. The rating prediction results show that the DP/CPT-v+ model outperforms the basic DP model when there is a strong non-random missing data effect.

v model on the Jester data. The standard training protocol where both missing data parameters and mixture model parameters are jointly sampled was used for the DP/CPT-v model. The horizontal axis represents the strength of the synthetic non-random missing data effect applied to the Jester data set. The vertical axis represents normalized mean absolute error. The rating prediction results again show that the DP/CPT-v model vastly outperforms the standard DP model on randomly selected items when there is a strong non-random missing data effect. The two models perform similarly when missing data is missing at random.

5.6 The Finite Mixture/Logit-vd Model

The Dirichlet Process mixture/CPT-v model demonstrates the combination of the CPT-v missing data model with the more flexible Dirichlet Process mixture data model. In this section we consider combining the finite mixture model with a more flexible missing data model, the Logit-vd model, a variation of the Logit model introduced by Marlin, Roweis, and Zemel for collaborative prediction [54]. The main idea behind the Logit-vd model is to allow the missing data probability for an item to depend on both the value of the underlying rating, and the identity of the item. The Logit-vd model specifies a restricted form for this relationship as seen

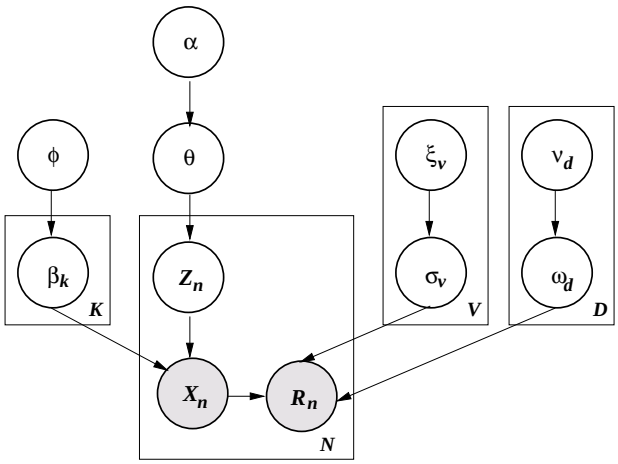


Figure 5.11: Bayesian network for the finite mixture/Logit-vd model combination.

in Equation 5.6.1. The σ_v factor models a non-random missing data effect that depends on the underlying rating value. This effect is constrained to be the same across all items. The ω_d factor models a per-item missing data effect. This effect can be useful if all items do not have the same exposure in a recommender system. This situation can arise when some items are more heavily promoted than others. The Logit-vd model with ω_d constrained to be equal for all d is equivalent to the CPT-v missing data model.

$$P(r_{dn} = 1|x_{dn} = v) = \mu_{vd} = \frac{1}{1 + \exp(-(\sigma_v + \omega_d))} \tag{5.6.1}$$

The remaining model components are identical to the finite mixture/CPT-v case, except that σ_v and ω_d are given Gaussian priors. The graphical model for the combined finite mixture/Logit-vd model is given in Figure 5.11. We give the probabilistic model in Equations 5.6.2 to 5.6.7.

$$P(\theta|\alpha) = \mathcal{D}(\theta|\alpha) \tag{5.6.2}$$

$$P(\beta|\phi) = \prod_{k=1}^{K} \prod_{d=1}^{D} \mathcal{D}(\beta_{dk}|\phi_{dk}) \tag{5.6.3}$$

$$P(\sigma, \omega|\xi, \tau, \nu, \rho) = \prod_{v=1}^{V} \mathcal{N}(\sigma_v|\xi_v, \tau) \cdot \prod_{d=1}^{D} \mathcal{N}(\omega_d|\nu_d, \rho) \tag{5.6.4}$$

$$P(Z_n = k|\theta) = \theta_k \tag{5.6.5}$$

$$P(\mathbf{X} = \mathbf{x}_n|Z_n = k, \beta) = \prod_{d=1}^{D} \prod_{v=1}^{V} \beta_{vdk}^{[x_{dn}=v]} \tag{5.6.6}$$

$$P(\mathbf{R} = \mathbf{r}_n|\mathbf{X} = \mathbf{x}_n, \mu) = \prod_{d=1}^{D} \prod_{v=1}^{V} \mu_{vd}^{[r_{dn}=1][x_{dn}=v]} (1 - \mu_{vd})^{[r_{dn}=0][x_{dn}=v]} \tag{5.6.7}$$

Equation 5.6.8 gives the complete data likelihood assuming the values of mixture indicator z_n and missing rating values $\mathbf{x}^m$ are known. Equation 5.6.9 gives the observed data likelihood after integrating over both missing data values, and mixture indicators.

$$P(z_n, \mathbf{x}_n, \mathbf{r}_n|\beta, \theta, \mu) = \prod_{k=1}^{K}\left(\theta_k \prod_{d=1}^{D}\prod_{v=1}^{V}\left(\mu_{vd}^{[r_{dn}=1]}(1-\mu_{vd})^{[r_{dn}=0]}\beta_{vdk}\right)^{[x_{dn}=v]}\right)^{[z_n=k]} \tag{5.6.8}$$

$$\begin{aligned} P(\mathbf{x}_n^o, \mathbf{r}_n|\mathbf{a}_n, \beta, \theta, \mu) = \sum_{k=1}^{K}\theta_k \prod_{d=1}^{D}\left(\prod_{v=1}^{V}(\mu_{vd}\beta_{vdk})^{[x_{dn}=v]}\right)^{[r_{dn}=1][a_{dn}=1]}\left(\sum_{v=1}^{V}\mu_{vd}\beta_{vdk}\right)^{[r_{dn}=1][a_{dn}=0]} \\ \cdot\left(\sum_{v=1}^{V}(1-\mu_{vd})\beta_{vdk}\right)^{[r_{dn}=0][a_{dn}=0]} \end{aligned} \tag{5.6.9}$$

5.6.1 Maximum A Posteriori Estimation

The principal of maximum a posteriori probability states that we should select the parameters θ, β, σ, and ω with maximum posterior probability. We again derive an Expectation Maximization algorithm to estimate the maximum a posteriori parameters. However, closed form updates for σ and ω are not possible, necessitating the use of a generalized Expectation Maximization approach.

The majority of the derivation for the Bayesian mixture/Logit-vd model is identical to the Bayesian mixture/CPT-v case. We begin by defining the auxiliary variables shown in Equation 5.6.10 to simplify the likelihood.

$$\begin{aligned} \gamma_{dkn} = \left(\prod_{v=1}^{V}(\beta_{vdk}\mu_{vd})^{[x_{dn}=v]}\right)^{[r_{dn}=1][a_{dn}=1]}\left(\prod_{v=1}^{V}(\beta_{vdk}(1-\mu_{vd}))^{[x=v]}\right)^{[r_{dn}=0][a_{dn}=1]} \\ \cdot\left(\sum_{v=1}^{V}\beta_{vdk}\mu_{vd}\right)^{[r_{dn}=1][a_{dn}=0]}\left(\sum_{v=1}^{V}\beta_{vdk}(1-\mu_{vd})\right)^{[r_{dn}=0][a_{dn}=0]} \end{aligned} \tag{5.6.10}$$

We give the posterior distribution of the mixture indicator variable z_n and the missing ratings $\mathbf{x}_n^m$ given the observed data $\mathbf{x}_n^o$, the response indicators $\mathbf{r}_n$ and the parameters θ, β, μ. We denote this distribution by $q_n(k, \mathbf{x}_n^m)$.

$$q_n(k, \mathbf{x}_n^m) = \frac{\theta_k \prod_{d=1}^{D}\prod_{v=1}^{V}(\mu_{vd}\beta_{vdk})^{[r_{dn}=1][x_{dn}=v]}((1-\mu_{vd})\beta_{vdk})^{[r_{dn}=0][x_{dn}=v]}}{\sum_{k=1}^{K}\theta_k\prod_{d=1}^{D}\gamma_{dkn}} \tag{5.6.11}$$

As in the CPT-v case, the full distribution $q_n(k, \mathbf{x}_n^m)$ is not explicitly required to compute the expectations needed in the E-step. Instead, a relatively small number of marginal distributions of the form $q_n(k)$, $q_n(v, d, k)$, and $q_n(v, d)$ suffice. We list these distributions below.

$$
\begin{aligned}
q_n(k) &= \frac{\theta_k \prod_{d=1}^{D} \gamma_{dkn}}{\sum_{k=1}^{K} \theta_k \prod_{d=1}^{D} \gamma_{dkn}} \\
q_n(k,v,d) &= q_n(k) \left(\frac{\mu_{vd}\beta_{vdk}}{\sum_{v'=1}^{V} \mu_{v'd}\beta_{v'dk}} \right)^{[r_{dn}=1][a_{dn}=0]} \left(\frac{(1-\mu_{vd})\beta_{vdk}}{\sum_{v'=1}^{V}(1-\mu_{v'd})\beta_{v'dk}} \right)^{[r_{dn}=0][a_{dn}=0]} \\
q_n(v,d) &= \sum_{k=1}^{K} q_n(k,v,d)
\end{aligned}
$$

We write the expected complete log posterior using the posterior marginal distributions. This again shows that the expected complete log posterior depends only on a relatively small set of local posterior factors, all of which can be efficiently computed.

$$
\begin{aligned}
&E[\log \mathcal{P}(\beta,\theta,\omega,\sigma|\{\mathbf{x}_n,\mathbf{r}_n,\mathbf{a}_n\}_{n=1:N},\alpha,\phi,\xi,\tau,\nu,\rho)] \qquad (5.6.12)\\
&= \log\Gamma(\sum_{k=1}^{K}\alpha_k) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) + \sum_{k=1}^{K}(\alpha_k-1)\log\theta_k \\
&\quad + \sum_{k=1}^{K}\sum_{d=1}^{D}\log\Gamma(\sum_{v=1}^{V}\phi_{vdk}) - \sum_{v=1}^{V}\log\Gamma(\phi_{vdk}) + \sum_{v=1}^{V}(\phi_{vdk}-1)\log\beta_{vdk} \\
&\quad - \sum_{v=1}^{V}\frac{1}{2}\log(2\pi\tau^2) - \frac{1}{2\tau^2}(\sigma_v-\xi_v)^2 - \sum_{d=1}^{D}\frac{1}{2}\log(2\pi\rho^2) - \frac{1}{2\rho^2}(\omega_d-\nu_d)^2 \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K} q_n(k)[z_n=k]\log\theta_k \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K} q_n(k)\sum_{d=1}^{D}\sum_{v=1}^{V}[z_n=k][x_{dn}=v][r_{dn}=1][a_{dn}=1](\log\beta_{vdk}+\log\mu_{vd}) \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{d=1}^{D}\sum_{v=1}^{V} q_n(k,v,d)[z_n=k][r_{dn}=1][a_{dn}=0](\log\beta_{vdk}+\log(\mu_{vd})) \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{d=1}^{D}\sum_{v=1}^{V} q_n(k,v,d)[z_n=k][r_{dn}=0][a_{dn}=0](\log\beta_{vdk}+\log(1-\mu_{vd}))
\end{aligned}
$$

Finally, we find the partial derivatives of the expected complete log posterior with respect to the multinomial parameters θ, β, and the real-valued parameters σ, and ω. The solution for β and θ is identical to the CPT-v case. As noted previously, the gradient equations for σ and ω can not be solved analytically.

$$\frac{\partial E[\log \mathcal{P}]}{\partial \theta_k} = \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{\theta_k} - \lambda = 0$$

$$\theta_k = \frac{\alpha_k - 1 + \sum_{n=1}^{N} q_n(k)}{N - K + \sum_{k=1}^{K} \alpha_k} \tag{5.6.13}$$

$$\frac{\partial E[\log \mathcal{P}]}{\partial \beta_{vdk}} = \frac{\phi_{vnk} - 1}{\beta_{vdk}} + \frac{\sum_{n=1}^{N} q_n(k)[a_{dn} = 1][x_{dn} = v]}{\beta_{vdk}} + \frac{\sum_{n=1}^{N} q_n(v,d,k)[a_{dn} = 0]}{\beta_{vdk}} - \lambda = 0$$

$$\beta_{vdk} = \frac{\phi_{vdk} - 1 + \sum_{n=1}^{N} q_n(k)[a_{dn} = 1][x_{dn} = v] + q_n(k,v,d)[a_{dn} = 0]}{\sum_{n=1}^{N} q_n(k) - V + \sum_{v=1}^{V} \phi_{vdk}} \tag{5.6.14}$$

$$\frac{\partial E[\log \mathcal{P}]}{\partial \mu_{vd}} = \frac{\sum_{n=1}^{N} [r_{dn} = 1][a_{dn} = 1][x_{dn} = v] + q_n(v,d)[r_{dn} = 1][a_{dn} = 0]}{\mu_{vd}} - \frac{\sum_{n=1}^{N} q_n(v,d)[r_{dn} = 0][a_{dn} = 0]}{(1 - \mu_{vd})} = 0 \tag{5.6.15}$$

$$\frac{\partial E[\log \mathcal{P}]}{\partial \sigma_v} = \sum_{d=1}^{D} \frac{\partial E[\log \mathcal{P}]}{\partial \mu_{vd}} \frac{\partial \mu_{vd}}{\partial \sigma_v} - \frac{1}{\tau^2}(\sigma_v - \xi_v) = 0$$

$$= \sum_{d=1}^{D} \sum_{n=1}^{N} \frac{\partial E[\log \mathcal{P}]}{\partial \mu_{vd}} \mu_{vd}(1 - \mu_{vd}) - \frac{1}{\tau^2}(\sigma_v - \xi_v) \tag{5.6.16}$$

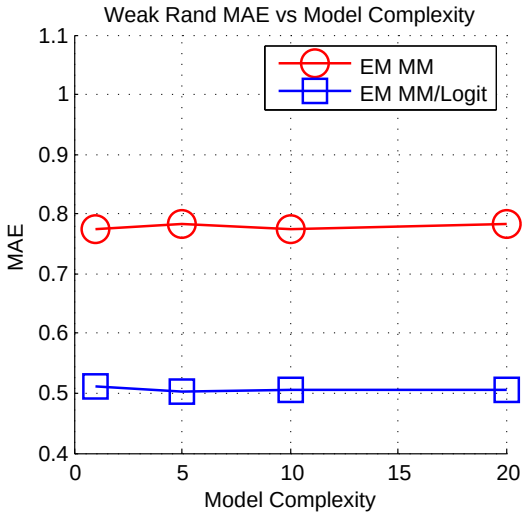
$$\frac{\partial E[\log \mathcal{P}]}{\partial \omega_d} = \sum_{v=1}^{V} \frac{\partial E[\log \mathcal{P}]}{\partial \mu_{vd}} \frac{\partial \mu_{vd}}{\partial \omega_d} - \frac{1}{\gamma^2}(\omega_d - \nu_d) = 0$$

$$= \sum_{v=1}^{V} \sum_{n=1}^{N} \frac{\partial E[\log \mathcal{P}]}{\partial \mu_{vd}} \mu_{vd}(1 - \mu_{vd}) - \frac{1}{\rho^2}(\omega_d - \nu_d) \tag{5.6.17}$$

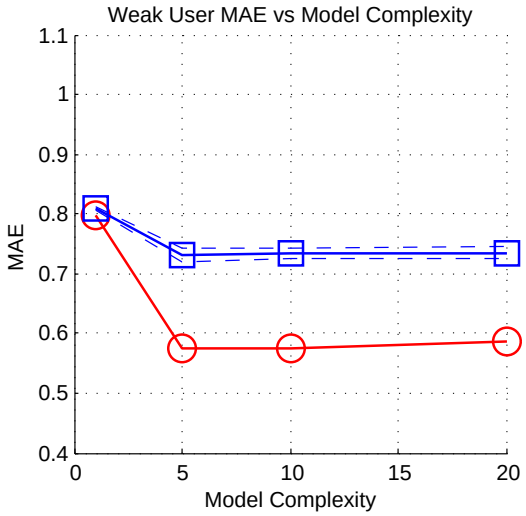
5.6.2 Rating Prediction

The posterior predictive distribution for the finite mixture/Logit-vd model is identical to the finite mixture/CPT-v case. The only difference is the definition of the μ_{vd} and γ_{vdk} auxiliary variables.

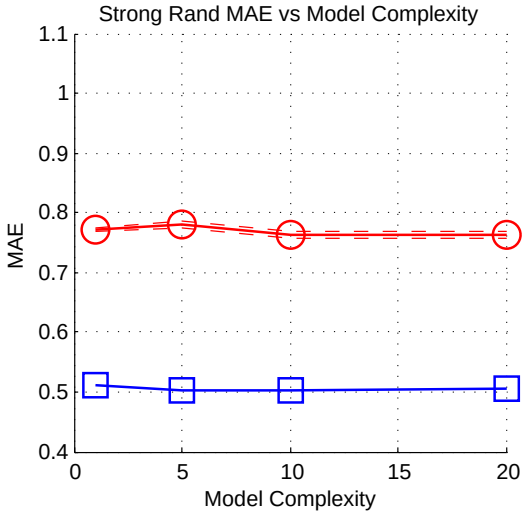
$$P(X_{dn} = v | \mathbf{x}_n, \mathbf{r}_n, \mathbf{a}_n, \theta, \beta, \mu) = \sum_{k=1}^{K} \frac{\theta_k \prod_{d=1}^{D} \gamma_{dkn}}{\sum_{k=1}^{K} \theta_k \prod_{d=1}^{D} \gamma_{dkn}} \left(\frac{\mu_{vd}\beta_{vdk}}{\sum_{v'=1}^{V} \mu_{v'd}\beta_{v'dk}} \right)^{[r_{dn}=1]} \cdot \left(\frac{(1 - \mu_{vd})\beta_{vdk}}{\sum_{v'=1}^{V} (1 - \mu_{v'd})\beta_{v'dk}} \right)^{[r_{dn}=0]} \tag{5.6.18}$$



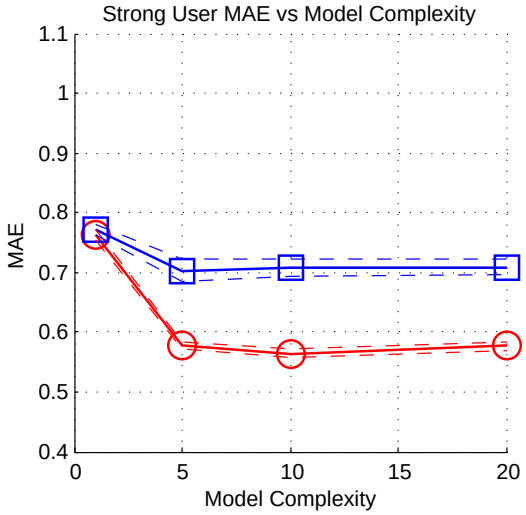
(a) Weak generalization on randomly selected items for the Yahoo! data set.



(b) Weak generalization on user selected items for the Yahoo! data set.

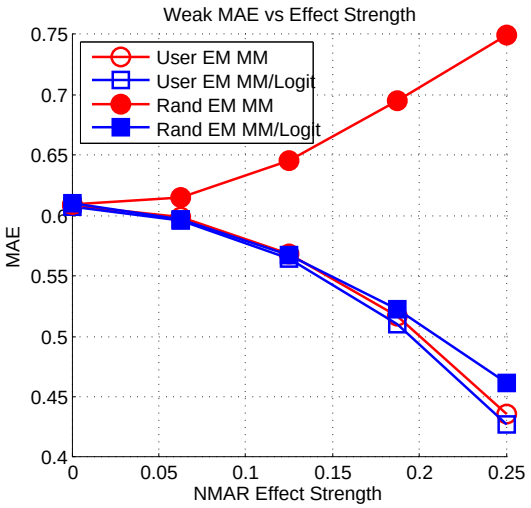


(c) Strong generalization on randomly selected items for the Yahoo! data set.

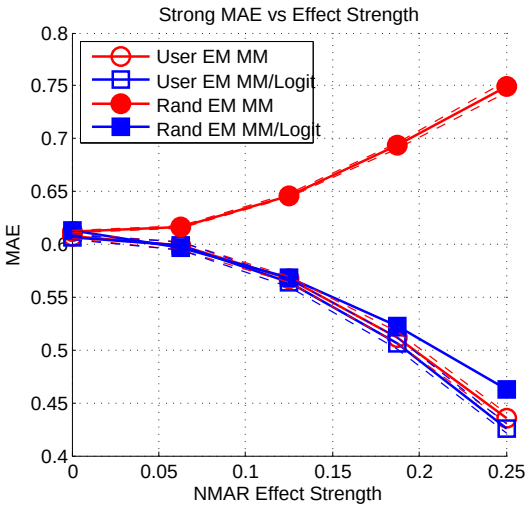


(d) Strong generalization on user selected items for the Yahoo! data set.

Figure 5.12: Figures 5.12(a) to 5.12(d) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/Logit-vd model learned using EM (EM MM/Logit-vd) on the Yahoo! data set. The horizontal axis represents the number of mixture components. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/Logit-vd model vastly outperforms the MM model on randomly selected items over a range of model complexity values.



(a) Weak generalization on randomly selected items for the Jester data set.



(b) Strong generalization on randomly selected items for the Jester data set.

Figure 5.13: Figures 5.12(a) to 5.12(d) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/Logit-vd model learned using EM (EM MM/Logit-vd). The horizontal axis represents non-random missing data effect strength. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/Logit-vd model vastly outperforms the MM model on randomly selected items as the effect strength increases.

5.6.3 Experimentation and Results

The standard training and testing protocols described in Sections 5.2.1 and 5.1.3 were applied to both the finite multinomial mixture model learned using EM assuming MAR (EM MM), and the finite multinomial mixture/Logit-vd model learned using EM (EM MM/Logit). A maximum of 5000 EM iterations were used to train each model. General smoothing priors were used for both models. The prior parameters on β_{dk} were set to $\phi_{vkd} = 2$ for all d and k in both models. The prior parameters on θ were set to $\alpha_k = 2$ for all k in both models. A zero mean Gaussian prior was used on both σ_v and ω_d with a standard deviation of 10. The σ_v parameters were all initialized to the same small random value. The ω_d parameters were each initialized to a different small random value. Models with 1, 5, 10, and 20 components were trained on the Yahoo! data. The Jester results are based on models with 5 components.

Figures 5.12(a) to 5.12(d) show the performance of the finite multinomial mixture model assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/Logit-vd model (EM MM/Logit) on the Yahoo! data set. The horizontal axis represents the number of mixture components. The vertical axis represents normalized mean absolute error. The solid lines represent the mean of the error over the five cross validation

folds for each model. The standard error of the mean is shown using dashed lines. The rating prediction results show that the MM/Logit-v model outperforms the MM model on randomly selected items over the complete range of model complexity values.

Figures 5.13(a) and 5.13(b) show the performance of the finite multinomial mixture model learned using EM assuming MAR (EM MM) compared to the performance of the combined finite multinomial mixture/Logit-vd model learned using EM (EM MM/Logit) on the Jester data. The horizontal axis represents the strength of the synthetic non-random missing data effect applied to the Jester data set. The vertical axis represents normalized mean absolute error. The rating prediction results show that the MM/Logit-vd model significantly outperforms the MM model on randomly selected items when there is a strong non-random missing data effect. The two models perform similarly when missing data is missing at random.

5.7 Restricted Boltzmann Machines

In this section we present restricted Boltzmann machines (RBMs) for complete data, and two types of conditional RBMs that can efficiently cope with missing data. Unlike the other models considered to this point, the conditional RBM models do not follow the data model/selection model factorization $P(X)P(R|X)$. Instead, they follow the alternative factorization $P(R)P(X|R)$. Since our only interest is making predictions for missing data values given the corresponding response indicators, the distribution $P(R)$ is not needed. Therefore, the conditional RBM models specify only the distribution $P(X|R)$.

We begin this section by reviewing RBMs for complete data. We then present a new interpretation of the conditional RBM model for missing data due to Salakhutdinov, Minh, and Hinton [65]. We extend the basic conditional RBM model and devise a new training protocol to better deal with prediction for non user-selected items. Finally, we present an empirical comparison of the two conditional RBM models for missing data.

5.7.1 Restricted Boltzmann Machines and Complete Data

Boltzmann machines are a sub-class of Markov Random fields. A Boltzmann machine is typically organized into one layer of visible units or observed variables, and multiple layers of hidden units or latent variables. A Boltzmann machine typically has connections between units in subsequent layers, as well as between units in the same layer [34]. Restricted Boltzmann Machines are a sub-class of Boltzmann Machines with one layer of hidden units, and no connections between units in the same layer [70].

The joint probability distribution over the data $\mathbf{x}_n$ and the hidden units $\mathbf{z}_n$ defined by a Restricted Boltzmann machine is specified in terms of an energy function $E(\mathbf{x}_n, \mathbf{z}_n)$ as seen in

Equation 5.7.1. The denominator of Equation 5.7.1 requires summing over all joint configurations of the visible and hidden units, and is referred to as the *partition function*. The probability of a data case is obtained from the joint distribution by summing over all configurations of the hidden units as seen in Equation 5.7.2.

$$P(\mathbf{x}_n, \mathbf{z}_n) = \frac{\exp(-E(\mathbf{x}_n, \mathbf{z}_n))}{\sum_{\mathbf{x}} \sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}))} \tag{5.7.1}$$

$$P(\mathbf{x}_n) = \sum_{\mathbf{z}} \frac{\exp(-E(\mathbf{x}_n, \mathbf{z}))}{\sum_{\mathbf{x}} \sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}))} \tag{5.7.2}$$

Restricted Boltzmann Machines can be defined for a variety of visible and hidden unit types. In this section we focus on restricted Boltzmann machines with binary hidden units and categorical visible units. We denote the number of hidden units by K. Equation 5.7.3 shows a typical energy function consisting of weights between the visible and hidden units parameterized by W, as well as biases on the visible and hidden units parameterized by b and c respectively.

$$E(\mathbf{x}_n, \mathbf{z}_n) = -\sum_{d=1}^{D} \sum_{v=1}^{V} \sum_{k=1}^{K} W_{vdk}[x_{dn} = v][z_{kn} = 1] - \sum_{d=1}^{D} \sum_{v=1}^{V} b_{vd}[x_{dn} = v] - \sum_{k=1}^{K} c_k[z_{kn} = 1] \tag{5.7.3}$$

The conditional distribution of x_{dn} given the hidden units is shown in Equation 5.7.4. The conditional distribution takes the form of a softmax function since x_{dn} is a categorical variable. The conditional distribution of z_{kn} given the complete vector of visible units is shown in equation 5.7.5. The conditional distribution takes the form of a logistic distribution since z_{kn} is a binary variable.

$$P(x_{dn} = v|\mathbf{z}_n, W, b) = \frac{\exp(\sum_{k=1}^{K} W_{vdk}[z_{kn} = 1] + b_{vd})}{\sum_{v=1}^{V} \exp(\sum_{k=1}^{K} W_{vdk}[z_{kn} = 1] + b_{vd})} \tag{5.7.4}$$

$$P(z_{kn} = 1|\mathbf{x}_n, W, c) = \frac{1}{1 + \exp(-(\sum_{d=1}^{D} \sum_{v=1}^{V} W_{vdk}[x_{dn} = v] + c_k))} \tag{5.7.5}$$

Maximum Likelihood Estimation

The maximum likelihood principle states that we should select the parameters W, b, and c with the highest likelihood given the data. The log likelihood of the parameters given a data set consisting of N completely observed data vectors is given in Equation 5.7.6.

$$\mathcal{L}(W, c, b|\{\mathbf{x}_n\}_{n=1:N}) = \sum_{n=1}^{N} \log P(\mathbf{x}_n|W, b, c) = \sum_{n=1}^{N} \log \left(\sum_{\mathbf{z}} \frac{\exp(-E(\mathbf{x}_n, \mathbf{z}))}{\sum_{\mathbf{x}} \sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}))} \right) \quad (5.7.6)$$

To fit the parameters by maximum likelihood we use gradient ascent on the log likelihood function $\mathcal{L}(W, c, b|\{\mathbf{x}_n\}_{n=1:N})$. We derive the gradient of the log likelihood with respect to the weight W_{vdk} below. The derivation for the two bias terms is similar.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W_{vdk}} &= \sum_{n=1}^{N} \frac{1}{P(\mathbf{x}_n|W, b, c)} \frac{\partial P(\mathbf{x}_n|W, b, c)}{\partial W_{vdk}} \\
&= -\sum_{\mathbf{z}} [z_k = 1][x_{dn} = v] P(\mathbf{x_n}, \mathbf{z}|W, c, b) \frac{\partial E(\mathbf{x}_n, \mathbf{z})}{\partial W_{vdk}} \\
&\quad + P(\mathbf{x}_n|W, c, b) \sum_{\mathbf{z}} \sum_{\mathbf{x}} [z_k = 1][x_d = v] P(\mathbf{x}, \mathbf{z}|W, c, b) \frac{\partial E(\mathbf{x}, \mathbf{z})}{\partial W_{vdk}} \\
\frac{\partial E(\mathbf{x}, \mathbf{z})}{\partial W_{vdk}} &= -[x_d = v][z_k = 1] \\
\frac{\partial \mathcal{L}}{\partial W_{vdk}} &= \sum_{n=1}^{N} [x_{dn} = v] P(z_k = 1|\mathbf{x}_n, W, c, b) - \sum_{\mathbf{z}} \sum_{\mathbf{x}} [z_k = 1][x_d = v] P(\mathbf{x}, \mathbf{z}|W, c, b) \quad (5.7.7)
\end{aligned}$$

The final form of the derivative with respect to W_{vdk} is given in Equation 5.7.7. The first term in the derivative is the conditional expectation of number of times that $z_k = 1$ and $x_d = v$ given the data. The second term in the derivative is the unconditional expectation of number of times that $z_k = 1$ and $x_d = v$. While the derivative has a simple form, it can not be computed exactly for large D and K due to the fact that computing $P(\mathbf{x}, \mathbf{z})$ involves a sum over the $V^D 2^K$ joint configurations of $\mathbf{z}$ and $\mathbf{x}$.

The straightforward approach to overcoming this difficulty is to use a Monte Carlo estimate of the second term in the derivative. The Restricted Boltzmann Machine admits a very simple Gibbs sampler based on alternately conditioning on the hidden units and sampling the visible units according to Equation 5.7.4, and conditioning on the visible units and sampling the hidden units according to Equation 5.7.5. Once the Gibbs sampler has reached equilibrium, a total of S samples $\mathbf{x}^s$, $\mathbf{z}^s$ are drawn. The second term in the derivative is approximated as seen in Equation 5.7.8.

$$\sum_{\mathbf{z}} \sum_{\mathbf{x}} [z_k = 1][x_d = v] P(\mathbf{x}, \mathbf{z}|W, b, c) \approx \frac{1}{S} \sum_{s=1}^{S} [z_k^s = 1][x_d^s = v] \quad (5.7.8)$$

Contrastive Divergence Learning

The main difficulty with the Gibbs approximation to Maximum Likelihood estimation in the Restricted Boltzmann machine is the time required to reach the equilibrium distribution using the Gibbs sampler. Suppose that for each data case n we initialize a different Gibbs sampler by setting the visible units to $\mathbf{x}_n$. We can estimate the second term in the derivative by running each Gibbs sampler for T steps, taking one sample from each, and using these samples in Equation 5.7.8. The main insight behind the contrastive divergence learning algorithm is that a small number of steps, such as $T = 1$, can yield sufficient information about the gradient for maximum likelihood learning to succeed [36].

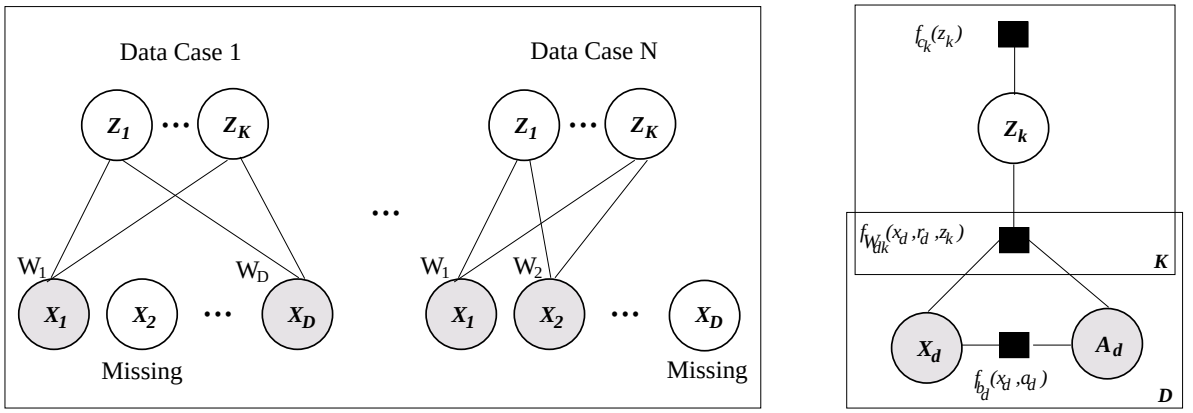
Let Q^0 to be the empirical distribution of the data, and Q^t to be the distribution of t-step reconstructions of the data using the Gibbs sampler. The standard Gibbs approach to maximum likelihood learning can be thought of as minimizing the Kullback-Leibler divergence $KL(Q^0||Q^\infty)$ since $Q^t = Q^\infty$ for any t once equilibrium is reached. The T-step contrastive divergence objective function is $\mathcal{C}(W, c, b|\{\mathbf{x}_n\}_{n=1:N}) = KL(Q^0||Q^\infty) - KL(Q^T||Q^\infty)$.

When we take the derivative of the Contrastive Divergence with respect to the parameter W_{vdk} in the case of the Restricted Boltzmann Machine, the intractable expectations involving $P(\mathbf{x}, \mathbf{z}|W, b, c)$ cancel out. However, we are left with the intractable partial derivative $(\partial KL(Q^T||Q^\infty)/\partial Q^T)(\partial Q^T/\partial w_{vdk})$. The work of Hinton indicates that this term can safely be ignored, and the derivative approximated by Equation 5.7.9 [36]. Note that x^T_{dn} is the value of visible unit d in the n^{th} Gibbs sampler (the Gibbs sampler with visible units initialized to $\mathbf{x}_n$) after T Gibbs steps.

$$\begin{aligned}
\frac{\partial \mathcal{C}}{\partial W_{vdk}} &= \sum_{n=1}^{N}[x_{dn} = v]P(z_k = 1|\mathbf{x}_n, W, b, c) - \sum_{\mathbf{z}}\sum_{\mathbf{x}}[z_k = 1][x_d = v]P(\mathbf{x}, \mathbf{z}|W, b, c) \\
&\quad - \sum_{n=1}^{N}[x^T_{dn} = v]P(z_k = 1|\mathbf{x}^T_n, W, b, c) + \sum_{\mathbf{z}}\sum_{\mathbf{x}}[z_k = 1][x_d = v]P(\mathbf{x}, \mathbf{z}|W, b, c) \\
&\quad - \frac{\partial KL(Q^T||Q^\infty)}{\partial Q^T}\frac{\partial Q^T}{\partial w_{vdk}} \\
&\approx \sum_{n=1}^{N}[x_{dn} = v]P(\mathbf{z}_k = 1|\mathbf{x}_n, W, b, c) - \sum_{n=1}^{N}[x^T_{dn} = v]P(\mathbf{z}_k = 1|\mathbf{x}^T_n, W, b, c) \qquad (5.7.9)
\end{aligned}$$

5.7.2 Conditional Restricted Boltzmann Machines and Missing Data

Unlike Bayesian network models such as factor analysis and mixture models, Markov random field models such as restricted Boltzmann machines can not efficiently deal with missing data. Both the standard Gibbs/Maximum Likelihood algorithm and the contrastive divergence



(a) Missing data RBM as described by Salakhutdinov et al.

(b) Factor graph representing missing data RBM.

Figure 5.14: Figure 5.14(a) shows the missing data RBM model as described by Salakhutdinov et al. The connectivity between visible and hidden units for each data case is determined by which data dimensions are observed in that data case. The weight parameters W are shared between different data cases. Figure 5.14(b) shows a factor graph representation of the same model where the dependence on the response indicators is made explicit. The factors are $f_{W_{dk}}(x_{dn}, a_{dn}, z_k) = \exp(\sum_{v=1}^{V} W_{vdk}[x_{dn} = v][a_{dn} = 1][z_k = 1])$, $f_{b_d}(x_{dn}, a_{dn}) = \exp(\sum_{v=1}^{V} b_{vd}[x_{dn} = v][a_{dn} = 1])$, and $f_{c_k}(z_k) = \exp(c_k[z_k = 1])$.

algorithm must sample missing visible units as well as latent hidden units. This makes standard RBM learning inefficient in terms of both computation time and storage space relative to Bayesian network models such as mixtures.

Salakhutdinov, Minh, and Hinton introduced a variant of the contrastive divergence learning algorithm for Restricted Boltzmann Machines that can efficiently deal with missing data [65]. The learning algorithm is based on the idea that each data case has a unique model structure in terms of the subset of visible to hidden edges it includes, while the parameters on each edge in different models are constrained to be equal. The structure associated with data case $\mathbf{x}_n$ has a connection between hidden unit z_k and visible unit x_d only when the value of x_{dn} is observed. On the other hand, the weights W_{vdk} on the connection from z_k to x_{dn} are equal to the weights from z_k to $x_{dn'}$ if dimension d is observed in both data case n, and data case n'. An illustration of the resulting set of models is given in Figure 5.14(a). Data case n contributes to the gradient of the contrastive divergence with respect to w_{vdk} only if x_{dn} is observed. This learning algorithm maintains the sparsity of the data set since samples are only generated for observed data values.

The algorithm introduced by Salakhutdinov et al. can also be derived as standard contrastive divergence learning in a conditional Restricted Boltzmann Machine. Salakhutdinov et al. discuss a particular type of conditional Restricted Boltzmann Machine with an energy term

of the form $V_{dk}[a_{dn} = 1][z_k = 1]$. However, the basic learning algorithm discussed above also implicitly conditions on the available rating indicators a_{dn}. Making this conditioning explicit clarifies the description of the model and the learning algorithm.

We provide a factor graph representation of the basic conditional RBM model for missing data in Figure 5.14(b). The key difference between the conditional RBM for missing data and the standard RBM is the definition of the energy function. The energy function for the conditional RBM is given in Equation 5.7.10. Note that $[x_{dn} = v]$ only appears in the energy function multiplied by $[a_{dn} = 1]$. Whenever x_{dn} is not available $a_{dn} = 0$, and the value of $[x_{dn} = v][a_{dn} = 1]$ is zero for all v. This is equivalent to removing all edges starting at x_{dn} in a standard RBM. Note that the basic conditional RBM model does not differentiate between user-selected and non-user selected ratings.

$$E(\mathbf{x}_n^o, \mathbf{z}_n, \mathbf{a}_n) = -\sum_{d=1}^{D}\sum_{v=1}^{V}\sum_{k=1}^{K} W_{vdk}[x_{dn} = v][z_{kn} = 1][a_{dn} = 1] - \sum_{d=1}^{D}\sum_{v=1}^{V} b_{vd}[x_{dn} = v][a_{dn} = 1] - \sum_{k=1}^{K} c_k[z_{kn} = 1] \tag{5.7.10}$$

Equation 5.7.11 shows the joint probability of the visible and hidden units conditioned on the response indicators. Equation 5.7.12 shows the probability of the visible units conditioned on the response indicators.

$$P(\mathbf{x}_n^o, \mathbf{z}_n|\mathbf{a}_n) = \sum_{\mathbf{x}_n^m} \frac{\exp(-E(\mathbf{x}_n, \mathbf{z}_n, \mathbf{a}_n))}{\sum_{\mathbf{x}}\sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}, \mathbf{a}_n))} = \frac{\exp(-E(\mathbf{x}_n^o, \mathbf{z}_n, \mathbf{a}_n))}{\sum_{\mathbf{x}^o}\sum_{\mathbf{z}} \exp(-E(\mathbf{x}^o, \mathbf{z}, \mathbf{a}_n))} \tag{5.7.11}$$

$$P(\mathbf{x}_n^o) = \sum_{\mathbf{z}} \frac{\exp(-E(\mathbf{x}_n^o, \mathbf{z}, \mathbf{a}_n))}{\sum_{\mathbf{x}^o}\sum_{\mathbf{z}} \exp(-E(\mathbf{x}^o, \mathbf{z}, \mathbf{a}_n))} \tag{5.7.12}$$

It is important to note that missing values x_{dn} can be analytically summed out of Equation 5.7.11 since they make no contribution to the energy function. As a result, the normalization in the partition function is taken with respect to the observed variables $\mathbf{x}^o$ only. The conditional RBM is similar to mixture models under the missing at random assumption in the sense that we can ignore missing data values during learning and inference. However, the conditional RBM model does not satisfy the missing at random condition.

Assume for the moment that we are in the simple setting where all observed ratings are available during inference so that $\mathbf{a} = \mathbf{r}$. The conditional RBM has the property that for any response pattern $\mathbf{r}$, and any two data vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\mathbf{x}_1^o = \mathbf{x}_2^o$, $P(\mathbf{x}_1|\mathbf{r}) = P(\mathbf{x}_2|\mathbf{r})$. This is precisely the property needed to ignore missing data during inference and learning.

Recall from Section 3.1 that the missing at random condition corresponds to the assertion that for any response pattern $\mathbf{r}$, and any two data vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ such that $\mathbf{x}_1^o = \mathbf{x}_2^o$, $P(\mathbf{r}|\mathbf{x}_1) = P(\mathbf{r}|\mathbf{x}_2)$. Equation 5.7.13 gives the conditional probability $P(\mathbf{r}|\mathbf{x})$ for an arbitrary response distribution $P(\mathbf{r})$.

$$P(\mathbf{r}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{r})P(\mathbf{r})}{\sum_{\mathbf{r}'} P(\mathbf{x}|\mathbf{r}')P(\mathbf{r}')} = \frac{P(\mathbf{x}|\mathbf{r})P(\mathbf{r})}{P(\mathbf{x})} \tag{5.7.13}$$

It is easy to see that $P(\mathbf{r}|\mathbf{x}_1)$ will only be equal to $P(\mathbf{r}|\mathbf{x}_2)$ if the marginal $P(\mathbf{x}_1) = P(\mathbf{x}_2)$. In general this condition will not hold, and missing data generated from the model will not be missing at random.

Contrastive Divergence Learning

The conditional distribution of x_d given the hidden units is shown in Equation 5.7.14. This equation is the same as in the standard RBM model. The conditional distribution of z_k given the observed visible units and available rating indicators is shown in equation 5.7.15.

$$P(x_d = v|\mathbf{z}_n, W, b) = \frac{\exp(\sum_{k=1}^K W_{vdk}[z_{kn} = 1] + b_{vd})}{\sum_{v=1}^V \exp(\sum_{k=1}^K W_{vdk}[z_k = 1] + b_{vd})} \tag{5.7.14}$$

$$P(z_k = 1|\mathbf{x}_n^o, \mathbf{a}_n, W, c) = \frac{1}{1 + \exp(-(\sum_{d=1}^D \sum_{v=1}^V W_{vdk}[x_{dn} = v][a_{dn} = 1] + c_k))} \tag{5.7.15}$$

It is straightforward to construct a Gibbs sampler for our conditional RBM based on Equations 5.7.14, and 5.7.15. Note that Equation 5.7.14 specifies the distribution of each data dimension given a configuration of the hidden units. However, we only draw samples for data dimensions where $a_{dn} = 1$, since only these dimensions contribute to Equation 5.7.15.

The approximate derivative of the contrastive divergence function with respect to W_{vdk} is given in Equation 5.7.16. The samples $\mathbf{x}_n^T$ are drawn using the Gibbs sampler for the conditional RBM. Note that this equation is identical to the one implied by Salakhutdinov et al. [65], and captures the same notion that data cases where x_{dn} is not observed do not contribute to the contrastive divergence gradient.

$$\frac{\partial \mathcal{C}}{\partial W_{vdk}} \approx \sum_{n=1}^N [x_{dn} = v][a_{dn} = 1]P(\mathbf{z}_k = 1|\mathbf{x}_n) - \sum_{n=1}^N [x_{dn}^T = v][a_{dn} = 1]P(\mathbf{z}_k = 1|\mathbf{x}_n^T) \tag{5.7.16}$$

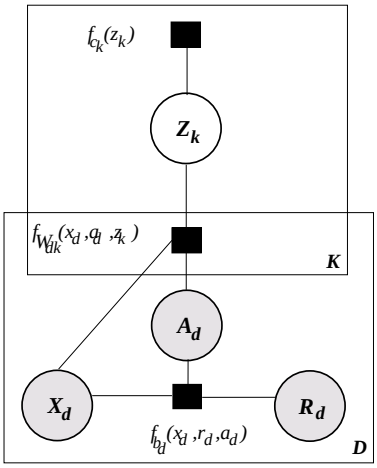


Figure 5.15: Factor graph representing the conditional RBM model that we refer to as cRBM/E-v. This model treats user-selected and non user-selected items differently. The factors are $f_{W_{dk}}(x_{dn}, a_{dn}, z_k) = \exp(\sum_{v=1}^{V} W_{vdk}[x_{dn} = v][a_{dn} = 1][z_k = 1])$, $f_{b_d}(x_{dn}, r_{dn}, a_{dn}) = \exp(\sum_{v=1}^{V} b^1_{vd}[x_{dn} = v][r_{dn} = 1][a_{dn} = 1] + b^0_v[x_{dn} = v][r_{dn} = 0][a_{dn} = 1])$, and $f_{c_k}(z_k) = \exp(c_k[z_k = 1])$.

Predictive Distribution

Making prediction using a trained conditional RBM is efficient assuming we predict one missing data dimension at a time. Suppose we are given a data vector $\mathbf{x}_n$ with response indicator vector $\mathbf{a}_n$ with $a_{dn} = 0$. To obtain the predictive distribution for data dimension d we add $x_{dn} = v$ to $\mathbf{x}^o_n$ for each value of v with a_{dn} set to 1 and normalize. This computation involves summing out over all joint configurations of the hidden units, but can be done in linear time as seen in Equation 5.7.17 [65].

$$\begin{aligned} P(x_{dn} = v | a_{dn} = 1, \mathbf{x}^o_n, \mathbf{a}_{-dn}, W, b, c) &\propto P(x_{dn} = v, \mathbf{x}_{-dn} | a_{dn} = 1, \mathbf{a}_{-dn}, W, b, c) \\ &\propto \exp(b_{vd}) \prod_{k=1}^{K} \left(1 + \exp\left(W_{vdk} + \sum_{d' \neq d}^{D} \sum_{v'=1}^{V} w_{v'd'k}[x_{d'n} = v'][a_{d'n} = 1] + c_k\right)\right) \end{aligned} \tag{5.7.17}$$

5.7.3 Conditional Restricted Boltzmann Machines and Non User-Selected Items

A key issue with the basic conditional RBM model is that it treats user selected and non-user selected items in the same way during learning and when making predictions. In the original application of the model by Salakhutdinov, Minh, and Hinton, predictions are only made for items selected by the user so this issue does not arise [65]. In this section we present a modification of the basic conditional RBM model that we call the cRBM/E-v model. The cRBM/E-v

learning procedure requires training ratings for both user-selected items and randomly selected items. Unlike the basic conditional RBM model, the cRBM/E-v model treats user-selected and non user-selected items differently using energy terms that depend on the response indicators $\mathbf{r}_n$ as well as the available rating indicators $\mathbf{a}_n$. The trained cRBM/E-v model makes different predictions for user-selected and non user-selected items in a way that is similar to the mixture/CPT-v model. This allows the cRBM/E-v model to better deal with prediction for non user-selected items than the basic cRBM model.

Equation 5.7.18 defines cRBM/E-v model. We have a contribution to the energy of W_{vdk} if $z_k = 1$, $x_{dn} = 1$ and $a_{dn} = 1$. This term is included both when $r_{dn} = 1$ and $r_{dn} = 0$. We have a contribution to the energy of b^1_{vd} if $x_{dn} = v$, $a_{dn} = 1$, and $r_{dn} = 1$. We have a contribution to the energy of b^0_v if $x_{dn} = v$, $a_{dn} = 1$, and $r_{dn} = 0$. To train the model we require data that includes observations of the form $a_{dn} = 1$ and $r_{dn} = 0$. This situation corresponds to having rating observations for items that the user did not select. Both the Yahoo! Rating Study data set and the Jester data set with synthetic missing data include observations of this form.

$$\begin{aligned} E(\mathbf{x}^o_n, \mathbf{z}_n, \mathbf{r}_n, \mathbf{a}_n) = &- \sum_{d=1}^{D}\sum_{v=1}^{V}\sum_{k=1}^{K} W_{vdk}[x_d = v][z_k = 1][a_{dn} = 1] - \sum_{k=1}^{K} c_k[z_k = 1] \\ &- \sum_{d=1}^{D}\sum_{v=1}^{V} b^1_{vd}[x_d = v][r_{dn} = 1][a_{dn} = 1] + b^0_v[x_d = v][r_{dn} = 0][a_{dn} = 1] \end{aligned} \tag{5.7.18}$$

The additional bias term b^0_v has an equivalent role to the μ_v parameter in the CPT-v model. This particular model form was selected since we anticipate a training regime where ratings for user selected items are abundant, while ratings for randomly selected items are significantly more scarce. If this were not the case the model could be easily enhanced by including additional factors in the energy function. Note that we do not include a factor of the from $V_{dk}[r_{dn} = 1][a_{dn} = 1][z_k = 1]$ as used by Salakhutdinov et al. since we have a relatively small number of observations where $r_{dn} = 1$ and $a_{dn} = 0$ [65]. The focus of our model is the complementary case where $r_{dn} = 0$ and $a_{dn} = 1$.

Contrastive Divergence Learning

The conditional distribution of x_{dn} given the hidden units is shown in Equation 5.7.19. The conditional distribution of z_k given the observed visible units and response indicators is shown in equation 5.7.20.

$$P(x_{dn} = v|r_{dn}, a_{dn}, \mathbf{z}, W, b) = \frac{\exp(\sum_{k=1}^{K} W_{vdk}[z_k = 1] + b_{vd}^1[r_{dn} = 1] + b_v^0[r_{dn} = 0])}{\sum_{v=1}^{V} \exp(\sum_{k=1}^{K} W_{vdk}[z_k = 1] + b_{vd}^1[r_{dn} = 1] + b_v^0[r_{dn} = 0])} \tag{5.7.19}$$

$$P(z_k = 1|\mathbf{x}_n^o, \mathbf{r}_n, \mathbf{a}_n, W, c) = \frac{1}{1 + \exp(-(\sum_{d=1}^{D} \sum_{v=1}^{V} W_{vdk}[x_{dn} = v][a_{dn} = 1] + c_k))} \tag{5.7.20}$$

It is straightforward to construct a Gibbs sampler for the cRBM/E-v model based on Equations 5.7.19, and 5.7.20. Note that the predictive distribution for x_{dn} in Equation 5.7.19 is different depending on whether the user selected the item or not.

The approximate derivative of the contrastive divergence function with respect to W_{vdk} is given in Equation 5.7.21. The samples $\mathbf{x}_n^T$ are drawn using the Gibbs sampler for the cRBM/E-v specified in Equations 5.7.19, and 5.7.20. Note that items rated under each rating source contribute to the contrastive divergence gradients. For completeness, we also give the gradients with respect to b_{vd}^1, b_v^0, and c_k.

$$\begin{aligned}\frac{\partial \mathcal{C}(\mathbf{x}|\mathbf{r}, \mathbf{a}, W, b, c)}{\partial W_{vdk}} \approx & \sum_{n=1}^{N} [x_{dn} = v][a_{dn} = 1] P(\mathbf{z}_{kn} = 1|\mathbf{x}_n, W, b, c) \\ & - \sum_{n=1}^{N} [x_{dn}^T = v][a_{dn} = 1] P(\mathbf{z}_{kn} = 1|\mathbf{x}_n^T, W, b, c)\end{aligned} \tag{5.7.21}$$

$$\frac{\partial \mathcal{C}(\mathbf{x}|\mathbf{r}, \mathbf{a}, W, b, c)}{\partial c_k} \approx \sum_{n=1}^{N} \left(P(\mathbf{z}_{kn} = 1|\mathbf{x}_n, W, b, c) - P(\mathbf{z}_{kn} = 1|\mathbf{x}_n^T, W, b, c) \right) \tag{5.7.22}$$

$$\frac{\partial \mathcal{C}(\mathbf{x}|\mathbf{r}, \mathbf{a}, W, b, c)}{\partial b_{vd}^1} \approx \sum_{n=1}^{N} [r_{dn} = 1][a_{dn} = 1] \left([x_{dn} = v] - P(x_{dn}^T = v|W, b, c) \right) \tag{5.7.23}$$

$$\frac{\partial \mathcal{C}(\mathbf{x}|\mathbf{r}, \mathbf{a}, W, b, c)}{\partial b_v^0} \approx \sum_{n=1}^{N} \sum_{d=1}^{D} [r_{dn} = 0][a_{dn} = 1] \left([x_{dn} = v] - P(x_{dn}^T = v|W, b, c) \right) \tag{5.7.24}$$

Rating Prediction

Making predictions using a trained cRBM/E-v model remains efficient assuming we predict one missing rating at a time. The predictive distribution again includes two cases corresponding to whether the item was originally selected by the user or not. The difference between the two cases comes down to whether the bias term b_{vd}^1 is used, corresponding to an item that was selected by the user, or whether the bias term b_v^0 is used, corresponding to an item not selected by the user. This conditional RBM model can learn and exploit rating patterns such as items not selected by the user tend to have lower ratings than items selected by the user.

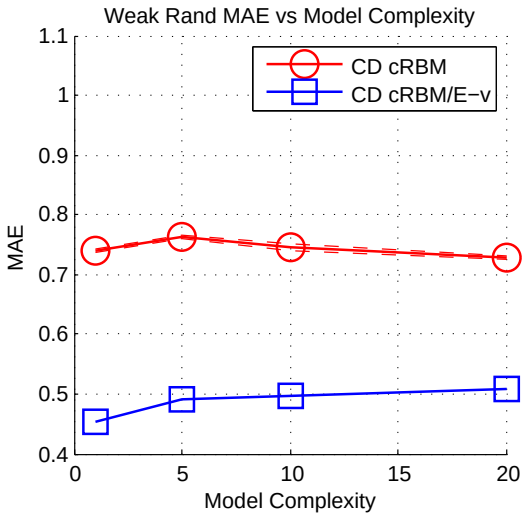
$$
\begin{aligned}
P(x_{dn} = v|\mathbf{x}_n^o, \mathbf{r}_n, \mathbf{a}_n, W, b, c) &\propto P(x_{dn} = v, \mathbf{x}_n^o|\mathbf{r}_n, \mathbf{a}_n, W, b, c) \\
&\propto \exp(b_{vd}^1[r_{dn} = 1]) \exp(b_v^0[r_{dn} = 0]) \\
&\cdot \prod_{k=1}^{K} \left(1 + \exp\left(W_{vdk} + \sum_{d' \neq d}^{D} \sum_{v'=1}^{V} W_{v'd'k}[x_{d'n} = v'][a_{dn} = 1] + c_k\right)\right)
\end{aligned} \tag{5.7.25}
$$

5.7.4 Experimentation and Results

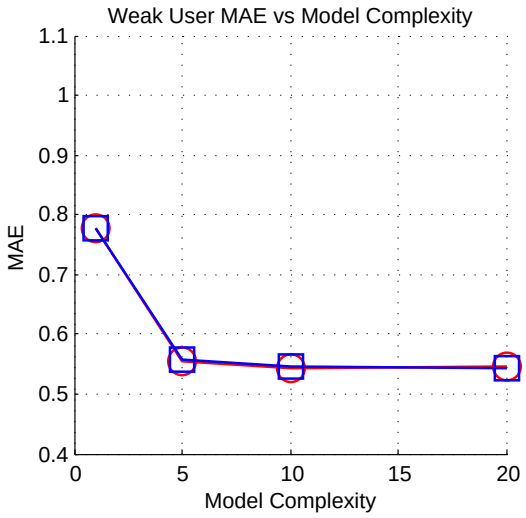
The training and testing protocols described in Sections 5.2.1 and 5.1.3 were applied to both the basic conditional RBM model (cRBM), and the conditional RBM/E-v model (cRBM/E-v). A total of 1000 iterations of contrastive divergence learning was performed for each model. Iteration i of the contrastive divergence algorithm used a Gibbs burn-in of $\lceil i/200 \rceil$ steps followed by the collection of $\lceil i/200 \rceil$ samples. A small amount of weight decay was used with regularization parameter set to 0.0001. The cRBM/E-v model was trained on the set of training users combined with the set of held out users. Models with 1, 5, 10, and 20 hidden units were trained on the Yahoo! data. The Jester results are based on models with 5 hidden units.

Figures 5.16(a) to 5.16(d) show the performance of the basic conditional RBM model (cRBM) compared to the conditional RBM/E-v model (cRBM/E-v). The horizontal axis represents the number of hidden units. The vertical axis represents normalized mean absolute error. The solid lines represent the mean of the error over the five cross validation folds for each model. The standard error of the mean is shown using dashed lines. The results show a substantial decrease in error on randomly selected items using the cRBM/E-v model. However, the error on randomly selected items increases as the number of components increases, while the error on user-selected items decreases. This is likely due to the fact a relatively small number of ratings for randomly selected items was added to a much larger data set of ratings for user-selected items. The gradient for the W parameter is thus dominated by contributions from user selected ratings, and appears to over fit these ratings as the number of hidden units increases.

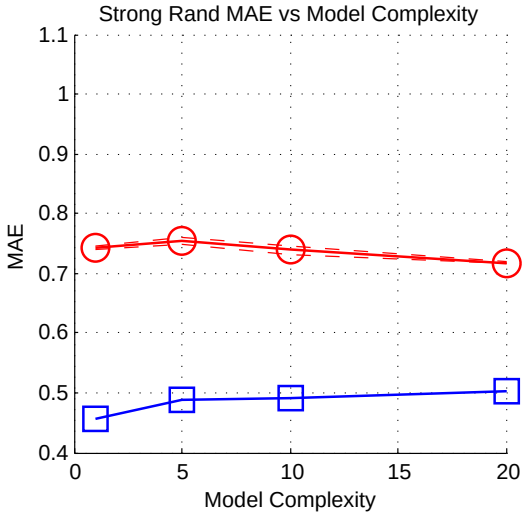
Figures 5.17(a) and 5.17(b) show the performance of the basic conditional RBM model (cRBM) compared to the performance of the conditional RBM/E-v model (cRBM/E-v) on the Jester data. The horizontal axis represents the strength of the synthetic non-random missing data effect applied to the Jester data set. The vertical axis represents normalized mean absolute error. The rating prediction results show that the cRBM/E-v model out performs the basic cRBM model when the non-random missing data effect strength is significant.



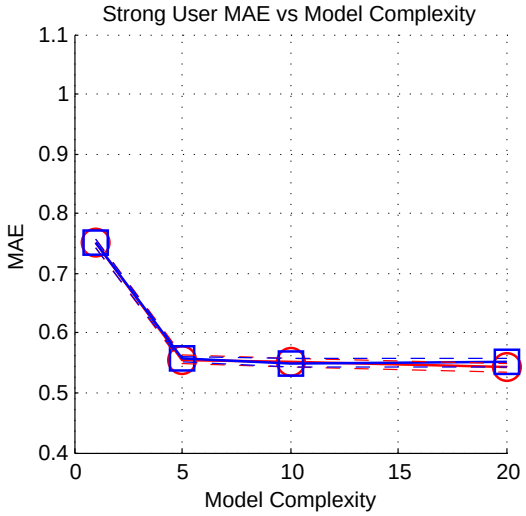
(a) Weak generalization on randomly selected items for the Yahoo! data set.



(b) Weak generalization on user selected items for the Yahoo! data set.

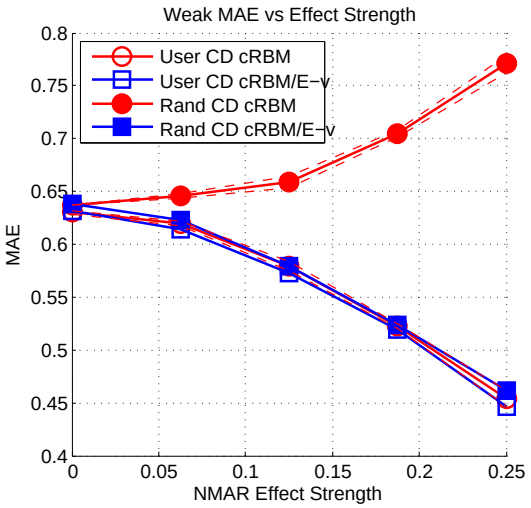


(c) Strong generalization on randomly selected items for the Yahoo! data set.

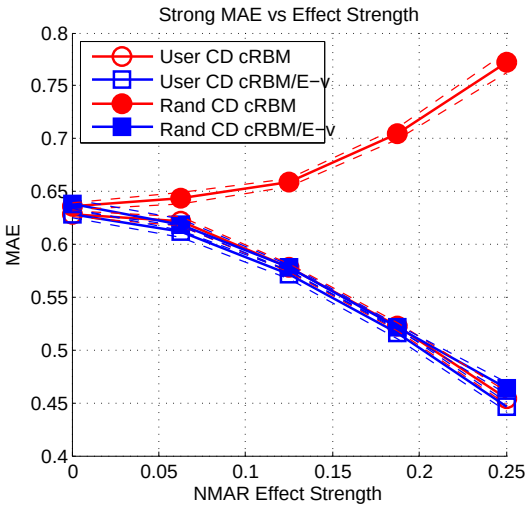


(d) Strong generalization on user selected items for the Yahoo! data set.

Figure 5.16: Figures 5.16(a) to 5.16(d) show the performance of the conditional RBM learned using contrastive divergence (CD cRBM) compared to the performance of the cRBM/E-v model (CD cRBM/E-v) on the Yahoo! data. The cRBM/E-v model is trained with an additional set of ratings of both user selected and randomly selected items. Not surprisingly, the cRBM and cRBM/E-v models have nearly identical performance on the user selected ratings. On the randomly selected ratings the cRBM/E-v model out-performs the basic cRBM model, which does not differentiate between user-selected and non user-selected items.



(a) Weak generalization on randomly selected items for the Jester data set.



(b) Strong generalization on randomly selected items for the Jester data set.

Figure 5.17: Figures 5.17(a) to 5.17(b) show the performance of the conditional Restricted Boltzmann Machine learned using contrastive divergence (CD cRBM) compared to the performance of the cRBM/E-v model (CD cRBM/E-v) on the Jester data. The cRBM/E-v model is trained with an additional set of ratings of both user selected and randomly selected items. The horizontal axis represents non-random missing data effect strength. The vertical axis represents normalized mean absolute error.

5.8 Comparison of Results and Discussion

The results presented in this chapter show that all of the models that include a non-random missing data mechanism perform well on the synthetic missing data in the Jester data set as the non-random effect strength is increased. The error on both randomly selected and user-selected items decreases as the effect strength increases. This results from the fact that as the effect strength increases, the response indicators contain an increasing amount of information about the value of missing ratings. Since the prediction function condition on the response indicators, the prediction error decreases. Models that treat the missing data in the Jester data set as if it were missing at random are increasingly mislead about the data distribution as the effect strength increases. The error on randomly selected items for all of the models that ignore the missing data mechanism increases as the non-random missing data effect strength increases. Of course, the Jester data set is quite limited since we know that the underlying missing data model is exactly a CPT-v mechanism. All of the methods that include a missing data model have the capacity to model this type of missing data.

The Yahoo! data provides a true test of rating prediction performance in the presence of an unknown missing data mechanism. Throughout this chapter we have observed that methods which incorporate even a simple non-random missing data mechanism vastly outperform

	Complexity	Rand MAE	Complexity	User MAE
EM MM	1	0.7725 ± 0.0024	5	0.5779 ± 0.0066
EM MM/CPT-v	20	0.5431 ± 0.0012	10	0.6661 ± 0.0025
EM MM/Logit	5	0.5038 ± 0.0030	5	0.7029 ± 0.0186
EM MM/CPT-v+	5	0.4456 ± 0.0033	20	0.7088 ± 0.0087
MCMC DP	N/A	0.7624 ± 0.0063	N/A	0.5767 ± 0.0077
MCMC DP/CPT-v	N/A	0.5549 ± 0.0026	N/A	0.6670 ± 0.0071
MCMC DP/CPT-v+	N/A	0.4428 ± 0.0027	N/A	0.7537 ± 0.0026
CD RBM	20	0.7179 ± 0.0025	10	0.5513 ± 0.0077
CD cRBM/E-v	1	0.4553 ± 0.0031	20	0.5506 ± 0.0085

Table 5.2: Summary of strong generalization error results on the Yahoo! data set. Results are reported for the model complexity that was found to be optimal according to the weak generalization estimate of the error on randomly selected and user selected items respectively. The optimal complexity for randomly selected items is shown in the first column. The second column shows the average mean absolute error on randomly selected items. The third column shows the optimal complexity for user-selected items. The fourth column shows the average mean absolute error on user selected items.

methods that ignore the missing data process on the random item prediction task. Table 5.2 summarizes the performance of the nine model variations we have experimented with in this chapter. We report the strong generalization mean absolute error on both randomly selected items and user selected items. Results are reported for the model complexity that achieved the lowest weak generalization error on randomly selected items and user-selected items, respectively.

Our empirical results show that rating prediction methods based on models that incorporate an explicit non-random missing data mechanism achieve 25% to 40% lower error rates on the Yahoo! data compared to methods that assume the missing at random condition holds. To put these results in perspective, the best models studied in our previous work on collaborative filtering achieve approximately 15% lower error rates relative to the simplest models we considered [52, p. 107-108].

The best methods overall are MCMC DP/CPT-v+, the Dirichlet process multinomial mixture model/CPT-v combination where the missing data parameters are set using held out ratings for randomly selected items; and EM MM/CPT-v+, the finite multinomial mixture model/CPT-v combination where the missing data parameters are set using held out ratings for randomly selected items. There is no significant difference in the performance of these methods on randomly selected items.

The CD cRBM/E-v model trained using a combination of ratings for user-selected and randomly selected items performs somewhat worse than the CPT-v+ variants. As mentioned previously, this is likely due to the imbalance between randomly selected and user selected item

	K=1	K=5	K=10	K=20
EM MM	0.8153 ± 0.0007	0.8135 ± 0.0006	0.8106 ± 0.0005	0.8073 ± 0.0006
EM MM/CPT-v	0.8257 ± 0.0006	0.8325 ± 0.0006	0.8353 ± 0.0006	0.8356 ± 0.0008
EM MM/Logit	0.8251 ± 0.0005	0.8385 ± 0.0003	0.8384 ± 0.0005	0.8381 ± 0.0010
EM MM/CPT-v+	0.8282 ± 0.0003	0.8337 ± 0.0007	0.8355 ± 0.0008	0.8367 ± 0.0007
MCMC DP	0.8167 ± 0.0007	0.8167 ± 0.0007	0.8167 ± 0.0007	0.8167 ± 0.0007
MCMC DP/CPT-v	0.8259 ± 0.0010	0.8259 ± 0.0010	0.8259 ± 0.0010	0.8259 ± 0.0010
MCMC DP/CPT-v+	0.8320 ± 0.0011	0.8320 ± 0.0011	0.8320 ± 0.0011	0.8320 ± 0.0011
CD cRBM	0.8104 ± 0.0007	0.8154 ± 0.0012	0.8174 ± 0.0010	0.8183 ± 0.0011
CD cRBM/E-v	0.8211 ± 0.0007	0.8185 ± 0.0010	0.8220 ± 0.0011	0.8210 ± 0.0009

Table 5.3: Summary of weak generalization ranking results on the Yahoo! data set in terms of average NDCG score on randomly selected items. Results are reported for model complexity values from $K = 1$ to $K = 20$. Note that the performance of the Dirichlet process models is independent of K.

during training. It may be possible to improve the performance of the cRBM/E-v model by altering the gradients to balance the contribution from randomly selected and user-selected items. It may also be interesting to consider an energy function with more flexibility in terms of modeling the missing data process.

Of the models that are trained only on user-selected items, the MM/Logit model obtained significantly lower error than the models incorporating a CPT-v type missing data model. It appears that the additional flexibility in the MM/Logit model helps to learn a better model. We note that some instability in the MM/Logit model was noticed when the σ_v parameters are not all initialized to the same value. Initializing all the σ_v to the same small random value appears to lead to good performance. We also experimented with a DP/Logit model combination. However, the complete lack of conjugacy created by the logistic function makes inference much more difficult when compared to the DP/CPT-v case. The development of a suitable DP/Logit sampling framework is an interesting area for future work.

The MM/CPT-v and DP/CPT-v models again have approximately equal performance when trained only on user selected items. As was observed by Marlin et al. [53], the MM/CPT-v model converges to a solution where almost all of the missing data is explained as having been generated by the second rating value. It was hoped that giving the model more flexibility during training by moving to a Dirichlet process based mixture and full Bayesian inference might alleviate this problem. However, the DP/CPT-v model appears to be prone to the same type of problem on this data set. The development of the auxiliary variable Gibbs sampler for the DP/CPT-v model is still quite interesting. The naive approach of sampling missing data values and performing cluster assignments on the basis of the completed data cases is completely ineffective since it vastly increases the state space of the Gibbs chain. An interesting

	Complexity	Rand NDCG
EM MM	1	0.8162 ± 0.0022
EM MM/CPT-v	20	0.8352 ± 0.0023
EM MM/Logit	5	0.8398 ± 0.0012
EM MM/CPT-v+	20	0.8377 ± 0.0012
MCMC DP	N/A	0.8167 ± 0.0025
MCMC DP/CPT-v	N/A	0.8248 ± 0.0020
MCMC DP/CPT-v+	N/A	0.8319 ± 0.0011
CD cRBM	20	0.8207 ± 0.0011
CD cRBM/E-v	10	0.8244 ± 0.0017

Table 5.4: Summary of strong generalization ranking loss results on the Yahoo! data set. Results are reported for the model complexity that was found to be optimal according to the weak generalization estimate of the average NDCG score on randomly selected items. The optimal complexity for randomly selected items is shown in the first column. The second column shows the average strong generalization NDCG score on randomly selected items.

area for future work is the development of a more flexible CPT-type missing data models with a hierarchical prior distribution.

The models developed in this chapter were designed to address the problem of accurate rating prediction in the presence of non-random missing data. However, we began this chapter by describing how rating prediction methods can be used to solve the recommendation task by predicting ratings for all unrated items, and then recommending the top-rated items. One question that has been left open to this point is the effect of non-random missing data on ranking. Tables 5.3 and 5.4 present the results of an experiment to test the weak and strong generalization ranking performance of the methods studied in this chapter.

The empirical protocol used for ranking is identical to that used for the mean absolute error rating prediction experiments. Each learned model is used to compute the predictive distribution over the test items for each user. The test items for each user are then ranked according to the mean of their predictive distributions, and a Normalized Discounted Cumulative Gain (NDCG) score is computed for each user given their true rating values. The per-user NDCG score is computed as seen in Equation 5.8.1 where x_{in}^t is the true value of test rating i for user n, $\hat{x}_{in}^t$ is the mean of the predictive distribution for test rating i and user n, $\pi(i, n)$ is the rank of item i when test items are sorted in descending order by true rating x_{in}^t, $\hat{\pi}(i, n)$ is the rank of test item i when items are sorted in descending order according to their predicted ratings $\hat{x}_{in}^t$, and T denotes the number of test items. Note that the best value of the NDCG score is 1, indicating that the estimated ranking agrees with an optimal ranking.

$$NDCG(n) = \frac{\sum_{i=1}^{T}(2^{x_{in}^t} - 1)/\log(1 + \hat{\pi}(i,n))}{\sum_{i=1}^{T}(2^{x_{in}^t} - 1)/\log(1 + \pi(i,n))} \tag{5.8.1}$$

In this experiment we perform mean prediction to rank the ten randomly selected test items for each survey participant in the Yahoo! data set. The weak generalization results are reported in Table 5.4 for model complexity values from 1 to 20. The best complexity level for each model based on the weak generalization NDCG score was determined from Table 5.3. The complexity level and corresponding strong generalization NDCG score is reported in Table 5.4. These results show small but significant improvements in average NDCG when the underlying rating prediction methods learn a model of the missing data mechanism.

An interesting avenue for future research is to consider models that both take into account a model of the missing data process, and directly optimize a rank loss measure. This is closely related to the recent work of Weimer, Karatzoglou, Le, and Smola on training matrix factorization methods for collaborative filtering based on a relaxation of the NDCG score [77].

Chapter 6

Classification With Missing Data

This chapter explores a variety of strategies for performing classification with missing feature values. The classification setting is particularly affected by the presence of missing feature values since most discriminative learning approaches including logistic regression, support vector machines, and neural networks have no natural ability to deal with missing input features. Our main interest is in classification methods that can both learn from data cases with missing features, and make predictions for data cases with missing features.

We begin with an overview of strategies for dealing with missing data in classification. Generative classifiers learn a joint model of labels and features. Generative classifier do have a natural ability to learn from incomplete data cases and to make predictions when features are missing. We then discuss several strategies that can be applied to any discriminative classifier including case deletion, imputation, and classification in subspaces. Finally, we discuss a framework for classification from incomplete data based on augmenting the input representation of complete data classifiers with a vector of response indicators.

We consider Linear Discriminant Analysis as an example of a generative classifier. We present both maximum likelihood and maximum conditional likelihood learning methods for a regularized Linear Discriminant Analysis model with missing data. We review several discriminative classification methods including logistic regression, multi-layer neural networks, and kernel logistic regression. We consider applying these methods to classification with missing data using imputation, reduced models, and the response indicator framework.

6.1 Frameworks for Classification With Missing Features

Generative classifiers have a natural ability to deal with missing data through marginalization. This makes them well suited for dealing with random missing data. The most well known methods for dealing with missing data in discriminative classifiers are case deletion, imputation,

and learning in subspaces. All of these methods can be applied in conjunction with any classifier that operates on complete data. In this section we discuss these methods for dealing with missing data. We also discuss a different strategy for converting a complete data classifier into a classifier that can operate on incomplete data cases by augmenting the input representation with response indicators.

6.1.1 Generative Classifiers

Generative classifiers model the joint distribution of labels and features. If any feature values are missing they can be marginalized over when classifying data cases. In class conditional models like the Naive Bayes classifier and Linear Discriminant Analysis, the marginalization operation can be performed efficiently. Missing data must also be dealt with during learning. This typically requires an application of the Expectation Maximization algorithm. However, generative classifiers require making explicit assumptions about the feature space distribution, while discriminative classifiers do not.

6.1.2 Case Deletion

In case deletion any data case with missing features values is discarded from the data set, and standard methods for complete data are run on the remaining data cases. Case deletion discards valuable information about features that are observed, and its use is generally not recommended [49]. Case deletion can be appropriate if there is a relatively small amount of missing data, and only complete data cases are permitted at test time. The particular missing data scenario we are interested in requires the ability to classify incomplete data vectors, so case deletion can not be applied.

6.1.3 Classification and Imputation

Imputation is a strategy for dealing with missing data that is widely used in the statistical community. In unconditional mean imputation, the mean of feature d is computed using the data cases where feature d is observed [49, p.44]. The mean value for feature d is then used as the value for feature d in data cases where feature d is not observed. In regression imputation, a set of regression models of missing features given observed features is learned. Missing features are filled in using predicted values from the learned regression model [49, p. 61].

Regression and mean imputation belong to the class of single imputation methods. In both cases a single completion of the data set is formed by imputing exactly one value for each unobserved variable. Multiple imputation is an alternative to single imputation procedures [49, p. 255]. As the name implies, multiple completions of a data set are formed by imputing several

values for each missing variable. In its most basic form, the imputed values are sampled from a simplified imputation model and standard methods are used on each complete data set. The principal advantage of multiple imputation over single imputation is that multiple imputation better reflects the variability due to missing values. Sophisticated forms of multiple imputation are closely related to approximate Bayesian techniques like Markov chain Monte Carlo methods, and can be viewed as an approximation to integrating out the missing data with respect to an auxiliary distribution over the feature space.

The key to imputation techniques is selecting an appropriate model of the input space to sample from. This is rarely the case in single imputation where imputing zeros is common. A standard practice in multiple imputation is to fit a gaussian distribution to each class, and sample multiple completions of the missing features conditioned on the observed features. More flexible imputation models for real valued data are often based on mixtures of Gaussians [75]. In high dimensions, learning a mixture of probabilistic principal components analysis or factor analysis models may be more appropriate.

The advantage of imputation methods is that they can be used in conjunction with any complete data classifier. The main disadvantage is that learning one or more imputation models can be a costly operation. In addition, using multiple imputations leads to maintaining an ensemble of classifiers at test time. Combining multiple imputation with cross validation requires training and evaluating many individual classifiers.

6.1.4 Classification in Sub-spaces: Reduced Models

Perhaps the most straightforward method for dealing with missing data is to learn a different classifier for each pattern of observed values. Sharpe and Solly studied the diagnosis of thyroid disease with neural networks under this framework, which they refer to as the network reduction approach [69]. The advantage of this approach is that standard discriminative learning methods can be applied to learn each model. Sharpe and Solly found that learning one neural network classifier for each subspace of observed features led to better classification performance than using neural network regression imputation combined with a single neural network classifier taking all features as inputs.

As Tresp et al. point out, the main drawback of the reduced model approach is that the number of different patterns of missing features is exponential in the number of features [75]. In Sharpe and Solly's case, the data set contained four inputs, and only four different patterns of missing features, making the entire approach feasible [69, p. 74].

6.1.5 A Framework for Classification with Response Indicators

An alternative to imputation and subspace classification is to augment the input to a standard classifier with a vector of response indicators. The input representation $\widetilde{\mathbf{x}}_n = [\mathbf{x}_n \odot \mathbf{r}_n, \mathbf{r}_n]$ can be thought of as an encoding for $\mathbf{x}_n^o$. Here $\odot$ signifies elementwise multiplication. A trained classifier can be thought of as computing a decision function of the form $f(\mathbf{x}_n^o)$. In logistic regression, multi-layer neural networks, and some kernel-based classifiers, substituting $\widetilde{\mathbf{x}}_n$ for $\mathbf{x}_n$ is the only modification required. This framework was studied in conjunction with certain SVM models by Chechik et al. [13], although they focus on the problem of structurally incomplete data cases. Structurally incomplete data cases arise when certain feature values are undefined for some data cases. This is semantically different than the missing data we consider here.

6.2 Linear Discriminant Analysis

In this section we present Linear Discriminant Analysis, and its application to classification with missing features. We begin by reviewing Fisher's original conception of Linear Discriminant Analysis. We then describe the relationship between Fisher's view and a view based on maximum probability classification in a class conditional Gaussian model. We discuss several extensions of LDA including Quadratic Discriminant Analysis (QDA), and Regularized Discriminant Analysis (RDA). We introduce a new method for missing data classification based on generative training of a linear discriminant analysis model with a factor analysis-style covariance matrix. Finally, we present a discriminative training method for the same model that maximizes the conditional probability of labels given features.

6.2.1 Fisher's Linear Discriminant Analysis

In the original binary classification setting, Fisher motivates the objective function for linear discriminant analysis by arguing for two criteria. The first criteria is that the means of the two classes $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_{-1}$ should be maximally separated by the choice of the discriminant direction $\mathbf{w}$ as expressed in Equation 6.2.1. The second criteria is that the chosen $\mathbf{w}$ should minimize the within-class scatter $\mathbf{S}$ of the two classes as expressed in Equation 6.2.2. The definition of the scatter matrix $\mathbf{S}$ is given in Equation 6.2.3 [20] [55, p. 63].

$$f_1(\mathbf{w}) = \mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1}) \tag{6.2.1}$$

$$f_2(\mathbf{w}) = \mathbf{w}^T\mathbf{S}\mathbf{w} \tag{6.2.2}$$

$$\mathbf{S} = \sum_{n=1}^{N} \sum_{c \in \{-1,1\}} [y_n = c](\mathbf{x}_n - \boldsymbol{\mu}_c)(\mathbf{x}_n - \boldsymbol{\mu}_c)^T \tag{6.2.3}$$

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n=1}^{N} [y_n = c]\mathbf{x}_n \tag{6.2.4}$$

Fisher proposes choosing $\mathbf{w}$ to maximize the objective function shown in Equation 6.2.5, which is the ratio of the square of mean separation to scatter in the direction $\mathbf{w}$.

$$f(\mathbf{w}) = \frac{f_1(\mathbf{w})^2}{f_2(\mathbf{w})} = \frac{\mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\mathbf{w}}{\mathbf{w}^T\mathbf{S}\mathbf{w}} \tag{6.2.5}$$

To find the best $\mathbf{w}^*$ we must differentiate $f(\mathbf{w})$ with respect to $\mathbf{w}$ and solve for $\mathbf{w}$. However, it is useful to note that the objective function $f(\mathbf{w})$ is invariant to the rescaling of $\mathbf{w}$. As a result we can always rescale an optimal $\mathbf{w}$ such that $f_2(\mathbf{w}) = 1$. We can then transform the optimization of $f(\mathbf{w})$ into a constrained problem with a simplified objective function as seen in Equation 6.2.6.

$$\begin{aligned} \mathbf{w}^* = \max_{\mathbf{w}} \ & \mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\mathbf{w} \\ \text{st } & \mathbf{w}^T\mathbf{S}\mathbf{w} = 1 \end{aligned} \tag{6.2.6}$$

To solve the constrained optimization problem we take the derivative of the objective function, and introduce a Lagrange multiplier along with the derivative of the constraint leading to equation 6.2.7. Isolating $\lambda\mathbf{w}$ under the assumption that $\mathbf{S}$ is invertible shows that $\mathbf{w}$ is given by the solution to a generalized eigenvalue problem.

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\mathbf{w} - \lambda\mathbf{S}\mathbf{w} = 0 \tag{6.2.7}$$

$$\mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\mathbf{w} = \lambda\mathbf{w} \tag{6.2.8}$$

This problem can be solved by inspection if we note that the result of multiplying any vector $\mathbf{w}$ by the matrix on the left hand side of Equation 6.2.8 is a vector proportional to $\mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})$. As a result, $\mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})$ is the only eigenvector, and the only possible solution to the generalized eigenvalue problem. We can thus conclude that $\mathbf{w}^* = \mathbf{S}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})$.

If we assume that the midpoint between the projections of the two means should separate the two classes, we can derive a simple formula for the optimal bias term b^* given the optimal weight vector $\mathbf{w}^*$as seen in Equation 6.2.9. A data case $\mathbf{x}_n$ is assigned to class 1 if $\mathbf{w}^{*T}\mathbf{x}_n + b > 0$.

$$b^* = \frac{1}{2}(\mathbf{w}^{*T}\boldsymbol{\mu}_1 + \mathbf{w}^{*T}\boldsymbol{\mu}_{-1}) = \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\mathbf{S}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_{-1}) \tag{6.2.9}$$

6.2.2 Linear Discriminant Analysis as Maximum Probability Classification

Welch was the first to note that Fisher's linear discriminant can be derived from the class posterior distribution in a class conditional Gaussian model with shared covariance, and different means [78]. This approach is more clearly motivated by the idea of maximum probability classification, and has become the standard derivation for linear discriminant analysis [55, p. 59]. The model is described in Equations 6.2.10 and 6.2.11. The posterior class probability is given in equation 6.2.12, which we work into a form that depends only on a linear function of the data.

$$P(Y_n = c) = \theta_c \tag{6.2.10}$$

$$P(\mathbf{X}_n = \mathbf{x}_n|Y_n = c) = |2\pi\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_c)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_c)\right) \tag{6.2.11}$$

$$P(Y_n = 1|\mathbf{X}_n = \mathbf{x}_n) = \frac{\theta_1|2\pi\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1)\right)}{\sum_{c\in-1,1}\theta_c|2\pi\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_c)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_c)\right)} \tag{6.2.12}$$

$$= \frac{1}{1 + \exp\left(\mathbf{w}^T\mathbf{x}_n + b\right)} \tag{6.2.13}$$

$$\mathbf{w} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\boldsymbol{\Sigma}^{-1} \tag{6.2.14}$$

$$b = \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{-1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_{-1}) + \frac{1}{2}\log(\theta_1/\theta_{-1}) \tag{6.2.15}$$

It is now easy to see that the discriminant function $P(Y_n = c|\mathbf{X}_n = \mathbf{x}_n) > 0.5$ is exactly equivalent to Fisher's discriminant function in the case where $\theta_1 = \theta_{-1} = 0.5$. The fact that Fisher's discriminant function is defined in terms of the scatter matrix S as opposed to the shared covariance matrix Σ makes no difference since the two are related by a scalar constant.

6.2.3 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis is the natural extension of Fisher's Linear Discriminant to the case where each class has a different covariance matrix. In this case the class posterior distribution obviously does not reduce to a linear function of the data. Instead, the decision surface is quadratic [55, p. 52]. We give the probability model for Quadratic Discriminant Analysis in equations 6.2.16 and 6.2.17. The class posterior is given in equation 6.2.18. As in the linear case, the classification rule is to assign a vector $\mathbf{x}_n$ to the class under which it has maximum posterior probability.

$$P(Y_n = c) = \theta_c \tag{6.2.16}$$

$$P(\mathbf{X}_n = \mathbf{x}_n | Y_n = c) = |2\pi\boldsymbol{\Sigma}_c|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_c)\right) \tag{6.2.17}$$

$$P(Y_n = 1 | \mathbf{X}_n = \mathbf{x}_n) = \frac{\theta_1 |2\pi\boldsymbol{\Sigma}_c|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1)\right)}{\sum_{c\in -1,1} \theta_c |2\pi\boldsymbol{\Sigma}_c|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_c)\right)} \tag{6.2.18}$$

6.2.4 Regularized Discriminant Analysis

In this section we discuss regularization strategies for Linear and Quadratic discriminant analysis. The main issue is that computing an accurate estimate of the required covariance matrices can be difficult when data is high dimensional and scarce. It is well known that the sampling variation of the standard covariance estimate goes up as the number of data cases goes down, and that when $N_c < D$ all of the covariance parameters are not identifiable for class c [55, p. 130].

Friedman presents an insightful demonstration of the effect this has on the discrimination function based on spectral analysis of the covariance matrix [23]. Assume that ϵ_d are the eigenvalues of $\boldsymbol{\Sigma}$, and that $\mathbf{v}_d$ are the corresponding eigenvectors. We have that $\boldsymbol{\Sigma}^{-1} = \sum_{d=1}^{D} \mathbf{v}_d \mathbf{v}_d^T / \epsilon_d$. This shows that the spectral decomposition of $\boldsymbol{\Sigma}^{-1}$ is dominated by the smallest eigenvalues of $\boldsymbol{\Sigma}^{-1}$.

Unsupervised Covariance Shrinkage

As Friedman claims, the eigenvalues of the sample covariance matrix tend to have a systematic bias, with the small eigenvalues biased low, and the large eigenvalues biased high [23]. Not surprisingly, the first techniques used to regularize covariance matrix estimates were based on attempts to correct the bias in the eigenvalue spectrum. Regularized estimates of the form $\alpha\hat{\boldsymbol{\Sigma}}_c + \beta\mathbf{I}$ were introduced to effectively shrink the eigenvalues towards a central value [58]. However, the optimal trade-off between the empirical covariance $\hat{\boldsymbol{\Sigma}}_c$ and the identity matrix $\mathbf{I}$ was sometimes left as an open problem, or set by optimizing a loss function unrelated to classification performance [55, p. 131].

Supervised Covariance Shrinkage

Friedman introduced the first covariance regularization method for the quadratic case based on minimizing a cross-validation estimate of prediction error [23] [55, p. 144]. The basic idea is to shrink the per-class covariance matrices towards the pooled covariance matrix by introducing a trade-off parameter α. That estimate is further shrunk towards a multiple of the identity matrix

through the introduction of a second trade-off parameter γ. We give the definition of Friedman's per-class covariance matrix estimate in equation 6.2.19 where $\hat{\mathbf{\Sigma}}_c$ is the standard estimate of the per-class covariance matrix, and $\hat{\mathbf{\Sigma}}$ is the standard estimate of the pooled covariance matrix [23].

$$\mathbf{\Sigma}_c = \gamma(\alpha\hat{\mathbf{\Sigma}}_c + (1-\alpha)\hat{\mathbf{\Sigma}}) + \frac{(1-\gamma)}{D}\mathrm{tr}(\hat{\mathbf{\Sigma}})\mathbf{I} \tag{6.2.19}$$

Friedman performs leave-one-out cross-validation over a grid of possible α and γ values with classification error as the objective function to choose the optimal α and γ. A specialized updating procedure is used during training to avoid the otherwise quadratic cost of computing a new set of covariance estimates when classifying each data case for each value of α and γ.

Restricted Eigen-Decomposition Covariance Models

Bensmail and Celeux introduced a regularization method called Eigenvalue Decomposition Discriminant Analysis that is based on a group of 14 different covariance approximations [6]. Each approximation is formed by imposing different symmetries on the eigen-decompositions of the class covariance matrices. Bensmail and Celeux propose a slightly refined eigen-decomposition of the form $\mathbf{\Sigma}_c = \lambda_c \mathbf{V}_c \boldsymbol{\epsilon}_c \mathbf{V}_c^T$ where $\mathbf{V}_c$ is the matrix of eigenvectors, $\lambda_c = |\mathbf{\Sigma}_c|^{1/D}$, and $\boldsymbol{\epsilon}_c$ is the matrix of eigenvalues normalized such that $|\boldsymbol{\epsilon}_c| = 1$. Under this decomposition, λ_c controls the volume of the covariance matrix, $\mathbf{V}_c$ controls the orientation, and $\boldsymbol{\epsilon}_c$ controls the shape [6]. The group of models proposed by Bensmail and Celeux includes simpler models of the form $\mathbf{\Sigma}_c = \lambda_c \mathbf{B}_c$ for $\mathbf{B}_c$ diagonal, and $\mathbf{\Sigma}_c = \lambda_c \mathbf{I}$

Regularization is achieved by tying the parameters of the decompositions across classes. For example, the model $\mathbf{\Sigma}_c = \lambda_c \mathbf{V} \boldsymbol{\epsilon}_c \mathbf{V}^T$ allows the volume and shape to differ across classes, but restricts the orientation parameters to be the same for all classes. Bensmail and Celeux, like Friedman, select the single model that minimizes a cross-validation estimate of classification error [6].

It is interesting to note that Bensmail and Celeux do not consider covariance models based on retaining only K of the D eigenvalue-eigenvector pairs. It is well known that retaining only the largest K eigenvalues and eigenvectors of a covariance matrix leads to an optimal rank K approximation in the squared error sense. This is essentially classical Principal Components Analysis [43]. Of course, classical PCA can not be directly applied as a covariance approximation since it results in a singular matrix for $K < D$. However, this is easily overcome by adding in a multiple of the identity matrix.

The Probabilistic Principal Components Analysis framework advanced by Tipping and Bishop and described in Section 4.3 provides just such a covariance approximation. Indeed,

Tipping and Bishop consider the application of the PPCA covariance approximation to the discriminant analysis problem with complete data, and show that it leads to improved classification error relative to spherical, diagonal, and full covariance matrices [73, p. 618].

6.2.5 LDA and Missing Data

As a generative model, Linear Discriminant Analysis has a natural ability to deal with missing input features. The class conditional probability of a data vector with missing input features is given in Equation 6.2.20. The posterior probability of each class given a data case with missing features is shown in Equation 6.2.21.

$$P(\mathbf{X}_n^o = \mathbf{x}_n^o | Y_n = c) = |2\pi\Sigma^{oo}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n^o - \mu_c^o)^T(\Sigma^{oo})^{-1}(\mathbf{x}_n^o - \mu_c^o)\right) \tag{6.2.20}$$

$$P(Y = c | \mathbf{X}_n^o = \mathbf{x}_n^o) = \frac{\theta_c |2\pi\Sigma^{oo}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n^o - \mu_c^o)^T(\Sigma^{oo})^{-1}(\mathbf{x}_n^o - \mu_c^o)\right)}{\sum_c \theta_c |2\pi\Sigma^{oo}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n^o - \mu_c^o)^T(\Sigma^{oo})^{-1}(\mathbf{x}_n^o - \mu_c^o)\right)} \tag{6.2.21}$$

Maximum Likelihood Estimation

The maximum likelihood estimate of the mean parameters is computed from incomplete data as shown in Equation 6.2.22.

$$\mu_{dc} = \frac{\sum_{n=1}^N [y_n = c][r_{dn} = 1] x_{dn}}{\sum_{n=1}^N [y_n = c][r_{dn} = 1]} \tag{6.2.22}$$

The parameters of the full covariance matrix can be estimated using the Expectation Maximization algorithm. However, when data vectors are high dimensional and there are a relatively small number of data cases, it is preferable to use a structured covariance approximation. A variety of structured covariance approximations were discussed in Section 6.2.4. We choose to use a factor analysis-like covariance matrix of the form $\Lambda\Lambda^T + \Psi$ with Ψ diagonal. We call this model LDA-FA for Linear Discriminant Analysis with Factor Analysis covariance. The factor analysis covariance model is slightly more general than the PPCA covariance model used by Tipping and Bishop in their LDA experiments [73, p. 618]. Note that while Tipping and Bishop also consider learning PPCA models with missing data, they do not consider the simultaneous application of PPCA to linear discriminant analysis with missing data.

The factor analysis covariance parameters are learned by first centering the training data by subtracting off the appropriate class mean as seen in Equation 6.2.23, and then applying the Expectation Maximization algorithm for factor analysis with missing data as derived in Section 4.3.2. The dimensionality of the latent factors Q is a free parameter that must be set using cross validation.

$$\hat{x}_{dn} = \sum_{c=1}^{C} [y_n = c][r_{dn}](x_{dn} - \mu_{dc}) \tag{6.2.23}$$

6.2.6 Discriminatively Trained LDA and Missing Data

One of the main drawbacks of generatively trained classification methods is that they tend to be very sensitive to violations of the underlying modeling assumptions. In this section we consider a discriminative training procedure for the LDA-FA model described in the previous section. The main insight is that we can fit the LDA-FA model parameters by maximizing the conditional probability of the labels given the incomplete features instead of maximizing the joint probability of the labels and the incomplete features. This training procedure is closely related to the minimum classification error factor analysis algorithm introduced by Saul and Rahim for complete data [66].

The posterior class probabilities given an incomplete feature vector are again given by Equation 6.2.21. The objective function used during training is the conditional log likelihood as shown in Equation 6.2.24.

$$\begin{aligned} \mathcal{L} &= \sum_{n=1}^{N} \sum_{c=1}^{C} [y_n = c] \log(P_n^c) \\ P_n^c &= \frac{A_n^c}{\sum_{c'} A_n^{c'}} \\ A_n^c &= \theta_c |2\pi\Sigma^{oo}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_n^o - \mu_c^o)^T (\Sigma^{oo})^{-1} (\mathbf{x}_n^o - \mu_c^o)\right) \end{aligned} \tag{6.2.24}$$

Conditional Maximum Likelihood Estimation

We derive a maximum conditional likelihood learning algorithm for the LDA-FA model in this section. We optimize the average conditional log likelihood with respect to the parameters θ, μ, Λ, and Ψ using non-linear optimization. We first transform the parameters θ and Ψ to eliminate constraints. θ represent the parameters of a discrete distribution with normalization and positivity constraints, while ψ_{ii} simply has to be positive since it is a variance parameter. We use the mappings shown below.

$$\Psi_{ii} = \exp(\phi_{ii}) \qquad\qquad \theta_c = \frac{\exp(\omega_c)}{\sum_c \exp(\omega_c)}$$

We begin by computing the partial derivative of the conditional log likelihood with respect

to the current posterior class probabilities P_n^k, and the partial derivative of the posterior class probability with respect to A_n^k.

$$\frac{\partial \mathcal{L}}{\partial P_n^k} = [y_n = k]\frac{1}{P_n^k}\partial P_n^k \qquad \frac{\partial P_n^k}{\partial A_n^l} = \left(\frac{[k=l]}{\sum_k A_n^k} - \frac{A_n^k}{(\sum_k A_n^k)^2}\right)\partial A_n^l \tag{6.2.25}$$

We compute the partial derivative of A_n^l with respect to θ_j, and the partial derivative of θ_j with respect to ω_c. We use the chain rule to find the partial derivative of the conditional log likelihood with respect to ω_c.

$$\frac{\partial A_n^l}{\partial \theta_j} = [l = j]\frac{A_n^l}{\theta_j} \tag{6.2.26}$$

$$\frac{\partial \theta_j}{\partial \omega_c} = [j = c]\theta_c - \theta_j\theta_c \tag{6.2.27}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \omega_c} &= \sum_{n=1}^{N}\sum_{k=1}^{C}\sum_{l=1}^{C}\sum_{j=1}^{C}\frac{\partial \mathcal{L}}{\partial P_n^k}\frac{\partial P_n^k}{\partial A_n^l}\frac{\partial A_n^l}{\partial \theta_j}\frac{\partial \theta_j}{\partial \omega_c} \\ &= \sum_{n=1}^{N}\sum_{k=1}^{C}\sum_{l=1}^{C}\sum_{j=1}^{C}[y_n = k]\frac{1}{P_n^k}\left(\frac{[k=l]}{\sum_k A_n^k} - \frac{A_n^k}{(\sum_k A_n^k)^2}\right)([j=c]\theta_c - \theta_j\theta_c) \\ &= \sum_{n=1}^{N}\sum_{c=1}^{C}([y_n = c] - P_n^c)\end{aligned} \tag{6.2.28}$$

We compute the partial derivative of A_n^l with respect to μ_c, and use the chain rule to find the partial derivative of the conditional log likelihood with respect to μ_c. The projection matrix $\mathbf{H}_n^o$ was introduced in Section 1.2.1. Recall that $\mathbf{H}_n^o$ projects the observed dimensions of $\mathbf{x}_n^o$ back into D dimension such that the missing dimensions are filled in with zeros. These projection matrices arise naturally when taking the derivative of a sub-matrix or sub-vector with respect to a full dimensional matrix or vector. Also recall that $\mathbf{o}_n$ refers to the vector of observed dimensions for data case $\mathbf{x}_n$ such that $o_{in} = d$ if d is the i^{th} observed dimension of $\mathbf{x}_n$.

$$\frac{\partial A_n^l}{\partial \boldsymbol{\mu}_c^{\mathbf{o}_n}} = [l = c]A_n^c(-\frac{1}{2})\left(2(\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1}(\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})\right) \tag{6.2.29}$$

$$\frac{\partial \boldsymbol{\mu}_c^{\mathbf{o}_n}}{\partial \boldsymbol{\mu}_c} = \mathbf{H}_n^o,\ H_{ijn}^o = [o_{jn} = i] \tag{6.2.30}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mu_c} &= \sum_{n=1}^{N}\sum_{k=1}^{C}\frac{\partial \mathcal{L}}{\partial P_n^k}\sum_{l=1}^{C}\frac{\partial P_n^k}{\partial A_n^l}\frac{\partial \boldsymbol{\mu}_c^{\mathbf{o}_n}}{\partial \boldsymbol{\mu}_c}\frac{\partial A_n^l}{\partial \boldsymbol{\mu}_c^{\mathbf{o}_n}} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{C}[y_n = k]\frac{1}{P_n^k}\sum_{l=1}^{C}\left(\frac{[k=l]}{\sum_k A_n^k} - \frac{A_n^k}{(\sum_k A_n^k)^2}\right)[l=c](-A_n^c)\mathbf{H}_n^o(\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1}(\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n}) \\
&= \sum_{n=1}^{N}\frac{1}{P_n^{y_n}}\left(\frac{[y_n=c]}{\sum_k A_n^k} - \frac{A_n^{y_n}}{(\sum_k A_n^k)^2}\right)\mathbf{H}_n^o(-A_n^c)(\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1}(\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n}) \\
&= \sum_{n=1}^{N}\left([y_n=c] - P_n^c\right)\mathbf{H}_n^o(\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1}(\mathbf{x}_n^{\mathbf{o}_n} - \boldsymbol{\mu}_c^{\mathbf{o}_n}) \qquad (6.2.31)
\end{aligned}$$

We compute the partial derivative of A_n^l with respect to $\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n}$, the partial derivative of $\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n}$ with respect to $\boldsymbol{\Sigma}$, and the partial derivative of $\boldsymbol{\Sigma}$ with respect to $\boldsymbol{\Lambda}$. We apply the chain rule to obtain the partial derivative of the conditional log likelihood with respect to $\boldsymbol{\Lambda}$.

$$\frac{\partial A_n^l}{\partial \boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n}} = \frac{1}{2}A_n^l\left((\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1}(\boldsymbol{\mu}_l^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})(\boldsymbol{\mu}_l^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})^T(\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} - (\boldsymbol{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1}\right) \qquad (6.2.32)$$

$$\frac{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}{\partial \boldsymbol{\Sigma}_{st}} = [\mathbf{o}_{in} = s][\mathbf{o}_{jn} = t] \qquad (6.2.33)$$

$$\frac{\partial \boldsymbol{\Sigma}_{st}}{\partial \boldsymbol{\Lambda}_{ab}} = [s=a]\boldsymbol{\Lambda}_{tb} + [t=a]\boldsymbol{\Lambda}_{sb} \qquad (6.2.34)$$

$$\begin{aligned}
\frac{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}{\partial \boldsymbol{\Lambda}_{ab}} &= \sum_{s=1}^{D}\sum_{t=1}^{D}[\mathbf{o}_{in}=s][o_{jn}=t]\left([s=a]\boldsymbol{\Lambda}_{tb} + [t=a]\boldsymbol{\Lambda}_{sb}\right) \\
&= [\mathbf{o}_{in}=a]\sum_{t=1}^{D}[\mathbf{o}_{jn}=t]\boldsymbol{\Lambda}_{tb} + [\mathbf{o}_{jn}=a]\sum_{s=1}^{D}[\mathbf{o}_{in}=s]\boldsymbol{\Lambda}_{sb} \qquad (6.2.35)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial A_n^l}{\partial \boldsymbol{\Lambda}_{ab}} &= \sum_{i=1}^{D_n}\sum_{j=1}^{D_n}\frac{\partial A_n^l}{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}\frac{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}{\partial \boldsymbol{\Lambda}_{ab}} \\
&= \sum_{i=1}^{D_n}\sum_{j=1}^{D_n}\frac{\partial A_n^l}{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}\left([\mathbf{o}_{in}=a]\sum_{t=1}^{D}[\mathbf{o}_{jn}=t]\boldsymbol{\Lambda}_{tb} + [\mathbf{o}_{jn}=a]\sum_{s=1}^{D}[\mathbf{o}_{in}=s]\boldsymbol{\Lambda}_{sb}\right) \\
&= 2\sum_{i=1}^{D_n}\sum_{j=1}^{D_n}\frac{\partial A_n^l}{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}[\mathbf{o}_{in}=a]\sum_{t=1}^{D}[\mathbf{o}_{jn}=t]\boldsymbol{\Lambda}_{tb} \\
&= 2\sum_{t=1}^{D}\left(\sum_{i=1}^{D_n}\sum_{j=1}^{D_n}[\mathbf{o}_{in}=a]\frac{\partial A_n^l}{\partial \boldsymbol{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}[\mathbf{o}_{jn}=t]\right)\boldsymbol{\Lambda}_{tb}
\end{aligned}$$

$$\frac{\partial A_n^l}{\partial \mathbf{\Lambda}} = 2\mathbf{H}_n^o \frac{\partial A_n^l}{\partial \mathbf{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}} (\mathbf{H}_n^o)^T \mathbf{\Lambda} \tag{6.2.36}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{\Lambda}} &= \sum_{n=1}^{N}\sum_{k=1}^{C} \frac{\partial \mathcal{L}}{\partial P_n^k} \sum_{l=1}^{C} \frac{\partial P_n^k}{\partial A_n^l} \frac{\partial A_n^l}{\partial \mathbf{\Lambda}} \\
&= \sum_{n=1}^{N}\sum_{k=1}^{C} [y_n = k] \frac{1}{P_n^k} \sum_{l=1}^{C} \left(\frac{[k=l]}{\sum_k A_n^k} - \frac{A_n^k}{\left(\sum_k A_n^k\right)^2} \right) \qquad (6.2.37) \\
&\quad \cdot \mathbf{H}_n^o \left(A_n^l \left((\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\boldsymbol{\mu}_l^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})(\boldsymbol{\mu}_l^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})^T (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} - (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} \right) \right) (\mathbf{H}_n^o)^T \mathbf{\Lambda} \\
&= \sum_{n=1}^{N}\sum_{c=1}^{C} ([y_n = c] - P_n^c) \Big(\mathbf{H}_n^o (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})(\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})^T (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\mathbf{H}_n^o)^T \\
&\quad - \mathbf{H}_n^o (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\mathbf{H}_n^o)^T \Big) \mathbf{\Lambda} \qquad (6.2.38)
\end{aligned}$$

Finally, we compute the partial derivative of the conditional log likelihood with respect to ϕ by computing the partial derivative of $\mathbf{\Sigma}$ with respect to ϕ and applying the chain rule.

$$\frac{\partial \mathbf{\Sigma}_{st}}{\partial \phi_{aa}} = [s = a][t = a]\Psi_{aa} \tag{6.2.39}$$

$$\frac{\partial \mathbf{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}{\partial \phi_{aa}} = \sum_{s=1}^{D}\sum_{t=1}^{D} [\mathbf{o}_{in} = s][o_{jn} = t][s = a][t = a]\Psi_{aa} = [\mathbf{o}_{in} = a][\mathbf{o}_{jn} = a]\Psi_{aa} \tag{6.2.40}$$

$$\frac{\partial A_n^l}{\partial \phi_{aa}} = \sum_{i=1}^{D_n}\sum_{j=1}^{D_n} \frac{\partial A_n^l}{\partial \mathbf{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}} \frac{\partial \mathbf{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}}{\partial \Phi_{aa}} = \sum_{i=1}^{D_n}\sum_{j=1}^{D_n} \frac{\partial A_n^l}{\partial \mathbf{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}} ([\mathbf{o}_{in} = a][\mathbf{o}_{jn} = a])\Psi_{aa} \tag{6.2.41}$$

$$\frac{\partial A_n^l}{\partial \phi} = \mathrm{diag}\Big(\mathbf{H}_n^o \frac{\partial A_n^l}{\partial \mathbf{\Sigma}_{ij}^{\mathbf{o}_n\mathbf{o}_n}} (\mathbf{H}_n^o)^T \Big) \Psi \tag{6.2.42}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \phi} &= \sum_{n=1}^{N}\sum_{k=1}^{C} \frac{\partial \mathcal{L}}{\partial P_n^k} \sum_{l=1}^{C} \frac{\partial P_n^k}{\partial A_n^l} \frac{\partial A_n^l}{\partial \phi} \\
&= \frac{1}{2} \sum_{n=1}^{N}\sum_{c=1}^{C} ([y_n = c] - P_n^c) \mathrm{diag}\Big(\mathbf{H}_n^o (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})(\boldsymbol{\mu}_c^{\mathbf{o}_n} - \mathbf{x}_n^{\mathbf{o}_n})^T (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\mathbf{H}_n^o)^T \\
&\quad - \mathbf{H}_n^o (\mathbf{\Sigma}^{\mathbf{o}_n\mathbf{o}_n})^{-1} (\mathbf{H}_n^o)^T \Big) \Psi \qquad (6.2.43)
\end{aligned}$$

In addition to controlling the complexity of the classifier through the latent dimensionality of the covariance model, it can also be useful to apply the equivalent of weight decay. The hyperplane separator for class c is given by $\boldsymbol{\mu_c}^T \Sigma$. Instead of regularizing $\boldsymbol{\mu_c}$ and Σ independently, we can regularize the norm of the hyperplane using an L2 penalty of the form $-0.5\lambda \boldsymbol{\mu_c}^T \Sigma \Sigma \boldsymbol{\mu_c}$. The additional gradient terms can easily be computed.

	Simple		Mix		Overlap	
	Loss	Err(%)	Loss	Err(%)	Loss	Err(%)
LDA-FA Gen	0.0449	1.75	0.3028	20.50	0.2902	13.50
LDA-FA Dis	0.0494	2.00	0.0992	3.25	0.2886	13.75

Table 6.1: Summary of illustrative results for generative and discriminatively trained LDA-FA models. We report the log loss (average negative log probability of the correct class), as well as the average classification error.

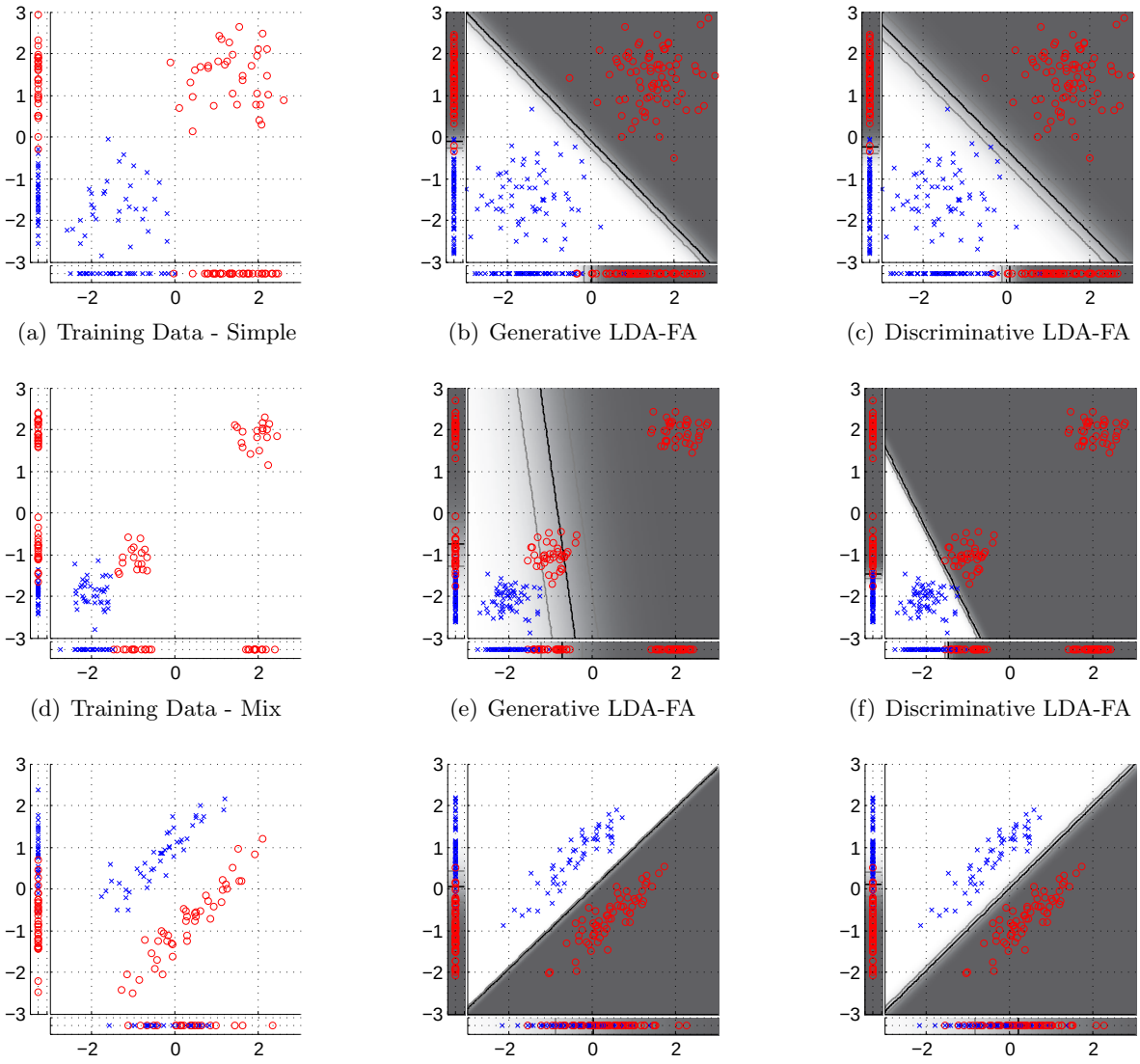
6.2.7 Synthetic Data Experiments and Results

Experiments were performed on three synthetic data sets to illustrate the properties of generative and discriminative training for the LDA-FA model. The first data set was sampled from a class conditional Gaussian model with equal class proportions, class means located at [-1.5,-1.5] and [1.5,1.5], and spherical covariance matrix with variance 0.5. 100 training cases were sampled from each class for training, and a separate set of 200 cases from each class was sampled for testing. Missing data was created by randomly sampling the three response patterns $[1, 1]$, $[0, 1]$, and $[1, 0]$ each with probability 1/3. We refer to this as the "Simple" data set. It is pictured in Figure 6.1(a).

The second data set was sampled from a class condition model where one class is Gaussian with mean $[-2, -2]$, and the other class is a mixture of Gaussians with one mean at $[-1, -1]$, and the other mean at $[2, 2]$. A spherical covariance matrix with variance 0.1 was used for both classes. A total of 100 training cases were sampled from each class for training, and a separate set of 200 cases from each class was sampled for testing. Missing data was created by randomly sampling the three response patterns $[1, 1]$, $[0, 1]$, and $[1, 0]$ each with probability 1/3. We refer to this as the "Mix" data set. It is pictured in Figure 6.1(d).

The third data set was sampled from a class condition Gaussian model with one class mean at $[-0.3, 0.8]$, and the other class mean at $[0.3, -0.8]$. A full covariance matrix was used with variance terms equal to 0.5, and covariance terms equal to 0.45. 100 training cases were sampled from each class for training, and a separate set of 200 cases from each class was sampled for testing. Missing data was created by randomly sampling the three response patterns $[1, 1]$, $[0, 1]$, and $[1, 0]$ each with probability 1/3. We refer to this as the "Overlap" data set. It is pictured in Figure 6.1(g).

The generative LDA-FA learning algorithm has one free parameter: the dimensionality of the latent space. In the experiments that follow we fix the latent dimensionality in both the generative and discriminative learning procedures to one since this setting is sufficient to learn any two dimensional covariance matrix. The regularization parameter in the discriminative training algorithm is set by a five fold cross validation search over the range 10^{-2} to 10^{-5}.



(a) Training Data - Simple (b) Generative LDA-FA (c) Discriminative LDA-FA

(d) Training Data - Mix (e) Generative LDA-FA (f) Discriminative LDA-FA

(g) Training Data - Overlap (h) Generative LDA-FA (i) Discriminative LDA-FA

Figure 6.1: The first column of figures shows the training data. The second column of figures shows the classification function learned by the generatively trained LDA-FA model. The third column of figures shows the classification function learned by the discriminatively trained LDA-FA model. Data cases missing the horizontal dimension are shown in the left panel in each figure. Data cases missing the vertical dimension are shown in the bottom panel. Complete data cases are shown in the main panel. Both discriminative and generative training perform well on the Simple data set shown in the first row of figures. The Mix data set shown in the second row of figures violates the LDA generative model assumptions. Generative training performs very poorly on the data set while discriminative training does much better. The Overlap data set shown in the third row is fundamentally difficult because the two classes overlap when either data dimension is missing.

Each method was run for a maximum of 1000 training iterations.

The results are presented in Figure 6.1 and summarized in Table 6.1. The first column in Figure 6.1 shows a plot of each training set. Data cases missing the horizontal dimension are shown in the left panel in each figure. Data cases missing the vertical dimension are shown in the bottom panel. Complete data cases are shown in the main panel. The second column in Figure 6.1 shows the classification function learned by the generatively trained LDA-FA model. The third column in Figure 6.1 shows the classification function learned by the discriminatively trained LDA-FA model. The intensity of the plot represents the conditional probability $P(y|\mathbf{x})$. The dark contour line represents $P(y|\mathbf{x}) = 0.5$. The two lighter contour lines represent $P(y|\mathbf{x}) = 0.25$ and $P(y|\mathbf{x}) = 0.75$.

The results show that both discriminative and generative training perform well on the Simple data set. The Mix data set violates the class conditional Gaussian assumption underlying the LDA-FA generative model. Generative training performs very poorly on the data set. Discriminative training obtains an error rate that is nearly six times lower than generative training, and a loss that is three times lower. The Overlap data set is fundamentally more difficult than the Simple and Mix data sets due to the fact that there is large overlap between the classes when either data dimension is missing. In such cases no classifier can perform well.

6.3 Logistic Regression

Logistic Regression is a statistical method for modeling the relationship between a binary or categorical class variable, and a vector of features variables or covariates. The name logistic regression comes from the use of the logistic function $f(x) = 1/(1+\exp(-x))$ as a map from the real line to the interval $[0, 1]$. In this section we review binary and multi-class logistic regression including model fitting, regularization, and the relationship between Logistic Regression and Linear Discriminant Analysis. We consider the application of logistic regression with missing data in conjunction with imputation, reduced models, and the response indicator framework.

6.3.1 The Logistic Regression Model

In the two class case the linear logistic regression model takes the form shown in equation 6.3.1 where $\mathbf{w}$ is a D dimensional vector referred to as the weight vector, and b is a scalar referred to as the bias [40, p. 31]. The multi-class case generalizes the binary case by introducing one weight vector $\mathbf{w}_c$ and bias b_c for each class [40, p. 260] [55, p. 255]. Without loss of generality we may assume that $\mathbf{w}_C = 0$ and $b_C = 0$ since the multiclass logistic regression model is over-parameterized when C distinct sets of parameters are used. We give the multi-class model in equation 6.3.2.

$$P(Y = 1|\mathbf{X} = \mathbf{x}_n) = \frac{1}{1 + \exp(-(\mathbf{w}^T\mathbf{x}_n + b))} \tag{6.3.1}$$

$$P(Y = c|\mathbf{X} = \mathbf{x}_n) = \frac{\exp(\mathbf{w}_c^T\mathbf{x}_n + b_c)}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T\mathbf{x}_n + b_{c'})} \tag{6.3.2}$$

In the binary case, the negative log probability of a single observation can also be expressed in terms of the logistic loss function as shown in equation 6.3.3 [68, p. 63].

$$\begin{aligned} -\log P(Y = y_n|\mathbf{X} = \mathbf{x}_n) &= -\log\left(\frac{[y_n = 1] + [y_n = -1]\exp(-(\mathbf{w}^T\mathbf{x}_n + b))}{1 + \exp(-(\mathbf{w}^T\mathbf{x}_n + b))}\right) \\ &= -\log\left(\frac{1}{1 + \exp(-y_n(\mathbf{w}^T\mathbf{x}_n + b))}\right) \\ &= \log(1 + \exp(-y_n(\mathbf{w}^T\mathbf{x}_n + b)) \end{aligned} \tag{6.3.3}$$

6.3.2 Maximum Likelihood Estimation for Logistic Regression

Maximum likelihood estimation is the standard approach to estimating parameters for logistic regression. Equation 6.3.4 shows the log likelihood function for logistic regression. Maximizing the likelihood function involves solving a set of gradient equations. We derive the gradient with respect to $\mathbf{w}_k$ and b_k in Equations 6.3.5 to 6.3.9. We introduce the notation $p_{cn} = P(Y = c|\mathbf{X} = \mathbf{x}_n)$ to simplify the derivation.

$$\mathcal{L} = \sum_{n=1}^{N}\sum_{c=1}^{C}[y_n = c]\log(p_{cn}) \tag{6.3.4}$$

$$\frac{\partial\mathcal{L}}{\partial p_{cn}} = 1/p_{cn} \tag{6.3.5}$$

$$\frac{\partial p_{cn}}{\partial \mathbf{w}_k} = (p_{cn}p_{kn} - [c = k]p_{cn})\mathbf{x}_n \tag{6.3.6}$$

$$\frac{\partial\mathcal{L}}{\partial \mathbf{w}_k} = \sum_{n=1}^{N}\sum_{c=1}^{C}[y_n = c](1/p_{cn})(p_{cn}p_{kn} - [c = k]p_{cn})\mathbf{x}_n = \sum_{n=1}^{N}(p_{kn} - [y_n = k])\mathbf{x}_n \tag{6.3.7}$$

$$\frac{\partial p_{cn}}{\partial b_k} = (p_{cn}p_{kn} - [c = k]p_{cn}) \tag{6.3.8}$$

$$\frac{\partial\mathcal{L}}{\partial b_k} = \sum_{n=1}^{N}\sum_{c=1}^{C}[y_n = c](1/p_{cn})(p_{cn}p_{kn} - [c = k]p_{cn}) = \sum_{n=1}^{N}(p_{kn} - [y_n = k]) \tag{6.3.9}$$

Maximizing the likelihood function requires the use of an iterative, non-linear optimization algorithm since a closed form solution of the gradient equations is not possible. Conjugate

gradient methods have the advantage of only needing the first order derivatives given above. A Newton-Raphson method also requires the matrix of second derivatives. The log likelihood function for linear logistic regression is strictly convex. The likelihood equations have a unique, finite-valued solution except for a small number of special cases related to a lack of overlap between the class distributions [1, p. 195] [55, p. 264]. Details can be found in Agresti [1, p. 194], Hosmer and Lemeshow [40, p. 263], and Mclachlan [55, p. 263].

6.3.3 Regularization for Logistic Regression

Regularization is an important issue when learning logistic regression parameters. The two main regularizers used with logistic regression are L2 regularization and L1 regularization. L2 regularized logistic regression has the advantage that the derivatives of the regularized objective function remain continuous. This means that standard, unconstrained optimization methods can still be used to find the optimal weight vector. On the other hand, the L1 regularized logistic regression objective function has discontinuous gradients if any component of the weight vector is 0. This means that standard, unconstrained optimization methods can not be reliably applied.

An advantage of L1 regularized logistic regression over L2 is that the L1 penalty tends to set weight parameters exactly to zero, while the L2 penalty does not. This means that the L1 regularizer is accomplishing both feature selection and regularization simultaneously. For example, Ng demonstrated that the L1 logistic regression formulation performs much better than L2 logistic regression in situations with many irrelevant features [61]. Tibshirani notes that in the linear regression case L1 performs better than L2 when there are a small number of large effects, and a moderate number of moderate effects [72]. In both cases the L1 penalty is able to ignore irrelevant or low-quality features.

These are important properties in machine learning where large-scale problems with many features of unknown quality are considered. As a result, several specialized methods for solving L1 regularized logistic regression have been proposed. Boyd has proposed a customized interior point method [8]. Lee et al. present a method based on a modification of iteratively re-weighted least squares [48]. Andrew and Gao present a modified limited-memory quasi-Newton method suitable for very large problems [4].

6.3.4 Logistic Regression and Missing Data

In this section we discuss the application of imputation, reduced models, and the response indicator framework to linear logistic regression with missing data.

Imputation

The application of zero imputation and mean imputation is straightforward. To apply multiple imputation using S imputations per data case we first learn S separate imputation models $P^s(\mathbf{x})$ from the incomplete training data set $\mathbf{x}_n^o$, $n = 1, ..., N$. For each data case n and each imputation model s we compute $P^s(\mathbf{x}_n^m|\mathbf{x}_n^o)$, and sample a value for $\mathbf{x}_{ns}^m$ to form the complete training data case $\mathbf{x}_{ns} = [\mathbf{x}_n^o, \mathbf{x}_{ns}^m]$. Note that we could also learn a single imputation model and draw S samples from that single model. We choose to learn multiple imputation models to help address the problem of multiple modes when learning imputation models such as mixtures.

For each of the S imputed data sets we learn a separate logistic regression model of the form $P^s(y|\mathbf{x})$. To classify a novel incomplete data case $\mathbf{x}_*^o$, we sample S completions of the data case $\mathbf{x}_{*s}$, $s = 1, ..., S$. We compute the predictive distribution $P^s(y|\mathbf{x}_{*s})$ under each of the S imputation models, and average them together to obtain the final imputation predictive distribution $P(y|\mathbf{x}_*^o)$.

$$P(y|\mathbf{x}_*^o) = \frac{1}{S}\sum_{s=1}^{S} P^s(y|\mathbf{x}_{*s}) \tag{6.3.10}$$

When combining multiple imputation with cross validation we learn the imputation models, and sample the imputed data sets before applying cross validation. This shortcut saves a large amount of computation. The imputation models learned before splitting the data into cross validation sets will tend to be somewhat more accurate than imputation models learned for a particular cross validation fold, since there is more training data available in the first case. However, the imputation models do not have access to labels during training, and the complete multiple imputation learning procedure could be viewed as a form of transductive learning.

Williams et al. introduced a scheme for incomplete data classification in a probit regression model that closely approximates logistic regression. The advantage of the probit link function is that is allows an approximation to analytically integrating over missing data with respect to an auxiliary Gaussian mixture feature space model [79]. However, the approach can only be applied in the binary classification setting, a limitation that does not apply to general multiple imputation.

Reduced Models

The application of reduced models to logistic regression is again straightforward. One important detail is that during training of a particular reduced model corresponding to the response pattern $\mathbf{r}$, we include data case n in the training set if for all d where $r_d = 1$, $r_{dn} = 1$. In other

words, we include as training data all cases that match the response pattern exactly, as well as all cases that match the response pattern, and have additional dimensions observed. Of course, the model for response pattern $\mathbf{r}$ is trained only on the dimensions where $r_d = 1$. At test time we compute the predictive distribution for a novel data case $\mathbf{x}_*^o$ using the reduced model that corresponds exactly to the novel data case's response pattern $\mathbf{r}_*$.

Response Indicator Framework

The response indicator framework can be applied to linear logistic regression by providing it with data vectors of the form $\widetilde{\mathbf{x}} = [\mathbf{x} \odot \mathbf{r}, \mathbf{r}]$. The resulting model for $P(Y = 1|\mathbf{x}, \mathbf{r})$ is given in equation 6.3.11 where w_d is the weight on feature d, v_d is the weight on response indicator r_d, and b is the bias. The objective function used to train the model is given in equation 6.3.12. It is easy to see that unobserved dimensions of $\mathbf{x}$ do not contribute to the classification function, or the objective function.

$$P(Y =1|\mathbf{x}, \mathbf{r}) = \frac{1}{1 + \exp(-(b + \sum_{d=1}^{D} r_d(w_d x_{dn} + v_d))} \tag{6.3.11}$$

$$F = \sum_{n=1}^{N} \log(1 + \exp(-y_n(b + \sum_{d=D}^{D} r_d(w_d x_{nd} + v_d))) + \lambda(\mathbf{w}^T\mathbf{w} + \mathbf{v}^T\mathbf{v}) \tag{6.3.12}$$

Learning the parameters of the response augmented logistic regression model is straightforward, and can be accomplished using any first or second order non-linear optimization method. Note that the optimization problem remains convex.

6.3.5 An Equivalence Between Missing Data Strategies for Linear Classification

Anderson was likely the first to note that the logistic regression model is much more general than the linear discriminant analysis model since it can also exactly represent the posterior of several other model classes [3] [55, p. 256]. Jordan showed that all exponential family models with shared dispersion parameters lead to class posteriors that can be represented exactly by linear logistic regression [44]. Banerjee has proven that the log-odds ratio of the class posteriors will be linear if and only if the class conditional distributions belong to the same exponential family, yielding a full characterization of the posterior distributions representable by logistic regression [5]. This implies that logistic regression and linear discriminant analysis are equivalent when the assumptions underlying LDA are satisfied. In fact, some early work on multivariate logistic regression models used linear discriminant estimates due to computational requirements of

obtaining direct estimates of the logistic regression model parameters [40, p. 43].

The relationship between logistic regression and linear discriminant analysis extends to the missing data setting as well. Under the class conditional gaussian/shared covariance assumption, the predictive distribution obtained from linear disclaimant analysis is asymptotically equivalent to the predictive distribution given by a combination of logistic regression and reduced models.

The true predictive distribution in a particular sub-space is given by $P(Y = y|\mathbf{x}^o)$. Under the class conditional Gaussian/shared covariance assumption the predictive distribution is calculated as seen below.

$$P(Y = y|\mathbf{x}^o) = \frac{\int_{\mathbf{x}^m} P(\mathbf{x}^o, \mathbf{x}^m|Y = y)P(Y = y)d\mathbf{x}^m}{\sum_y \int_{\mathbf{x}^m} P(\mathbf{x}^o, \mathbf{x}^m|Y = y)P(Y = y)d\mathbf{x}^m} \tag{6.3.13}$$

$$= \frac{P(\mathbf{x}^o|Y = y)P(Y = y)}{\sum_{y'} P(\mathbf{x}^o|Y = y')P(Y = y')} \tag{6.3.14}$$

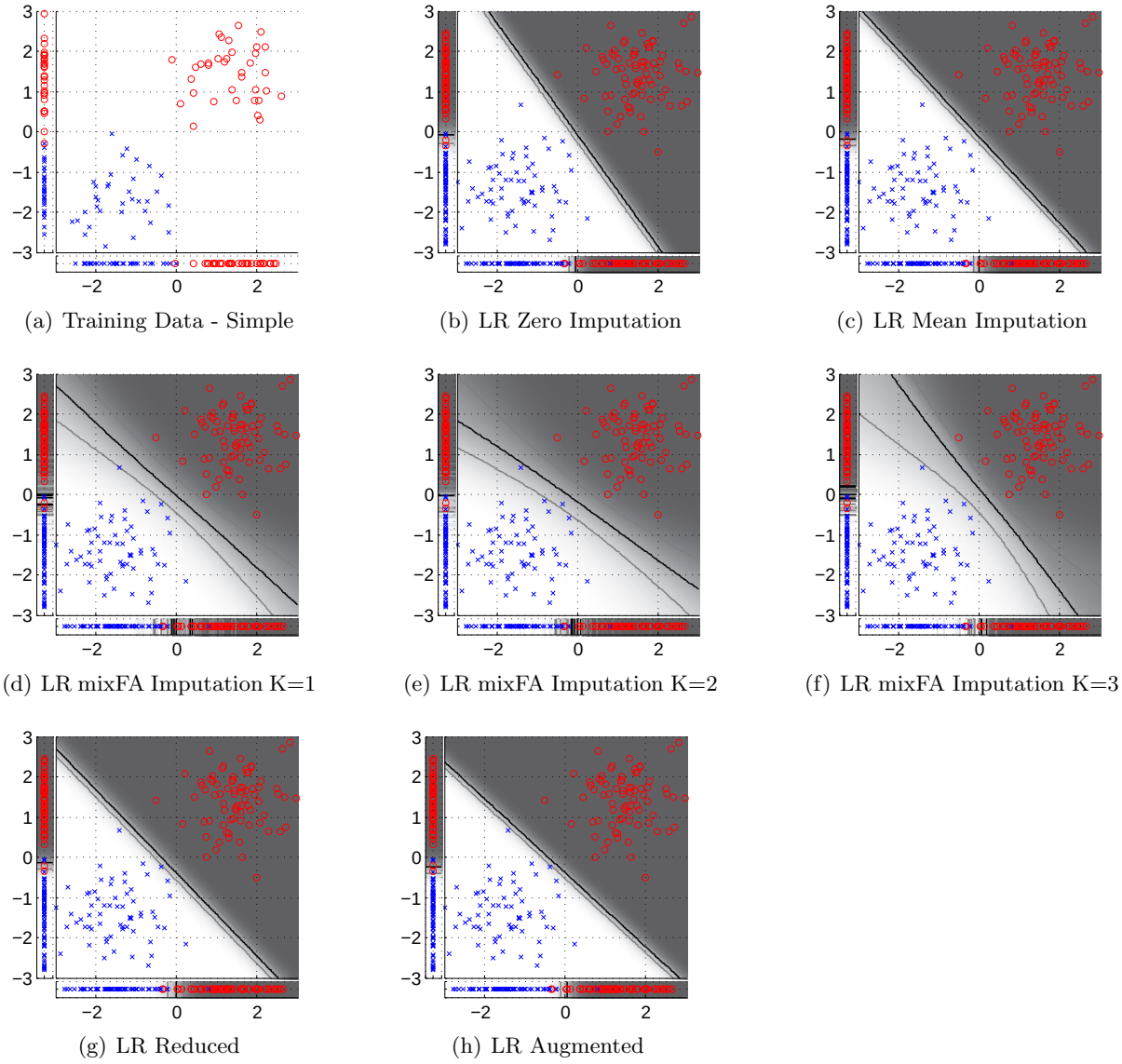
$$= \frac{\mathcal{N}(\mathbf{x}^o; \mu_y^o, \Sigma^{oo})P(Y = y)}{\sum_{y'} \mathcal{N}(\mathbf{x}^o; \mu_{y'}^o, \Sigma^{oo})P(Y = y')} \tag{6.3.15}$$

Since the LDA parameter estimates are asymptotically unbiased, the estimated predictive distribution $P^{LDA}(Y = y|\mathbf{x}^o, \hat{\mu}, \hat{\Sigma})$ will be equal to the true predictive distribution in the limit. We know that logistic regression can exactly represent the predictive distribution of any class conditional Gaussian model with shared covariance, and that the logistic regression objective function is convex. As a result, the predictive distribution $P^{LR}(Y = y|\mathbf{x}^o, \hat{\mathbf{w}}, \hat{b})$ learned in each observed data subspace will also be equal to the true predictive distribution in the limit.

Of course, when the class conditional Gaussian/shared covariance assumption is violated the two approaches can lead to very different predictive distributions. The generative linear discriminants approach makes much more stringent assumptions about the structure of the input space, while the logistic regression/reduced models approach makes no such assumptions.

6.3.6 Synthetic Data Experiments and Results

Illustrative experiments comparing different strategies for missing data classification with linear logistic regression were performed using the "Simple", "Mix", and "Overlap" data sets introduced in Section 6.2.7. We compare logistic regression combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response indicator framework. The imputation model used is a mixture of factor analyzers with latent dimensionality $Q = 1$. We present results using one, two, and three mixture components. For each data set and each number of mixture components we learn five separate imputation models using the Expecta-

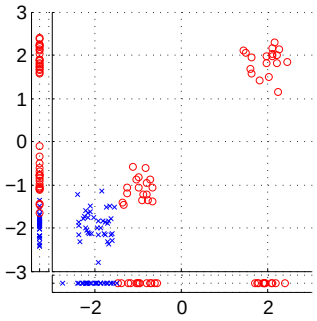


(a) Training Data - Simple (b) LR Zero Imputation (c) LR Mean Imputation

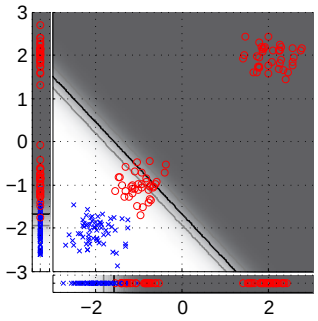
(d) LR mixFA Imputation K=1 (e) LR mixFA Imputation K=2 (f) LR mixFA Imputation K=3

(g) LR Reduced (h) LR Augmented

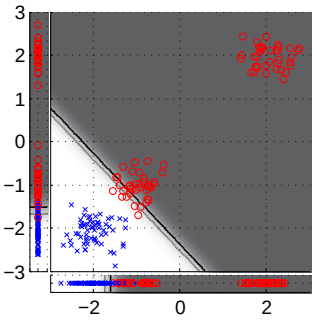
Figure 6.2: This figure shows results for the Simple data set based on several missing data methods combined with logistic regression. Missing data methods include zero imputation, mean imputation, multiple imputation, reduced models, and response indicator augmentation. The results show that zero imputation and mean imputation can perform well if they result in nearly separable configurations in the original feature space. The multiple imputation results illustrate the fact that the imputation model can be incorrect ($K = 1$), yet still give good estimates of the conditional densities $P(\mathbf{x}^m|\mathbf{x}_n^o)$.



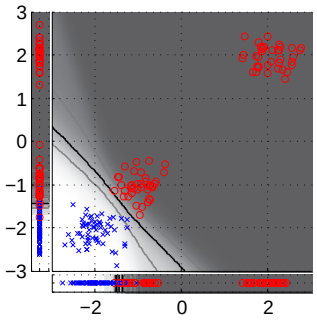
(a) Training Data - Mix



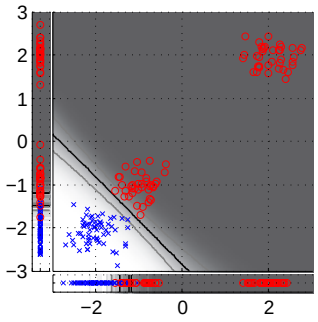
(b) LR Zero Imputation



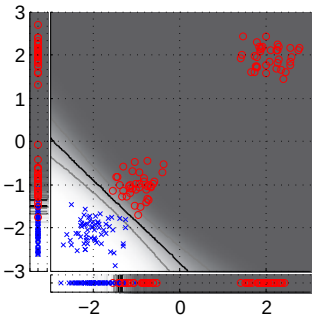
(c) LR Mean Imputation



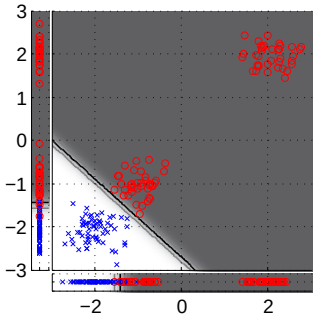
(d) LR mixFA Imputation K=1



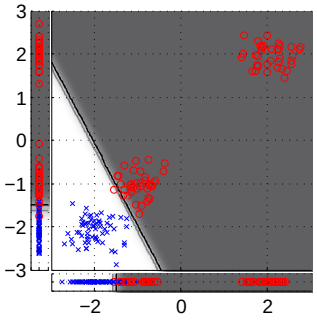
(e) LR mixFA Imputation K=2



(f) LR mixFA Imputation K=3

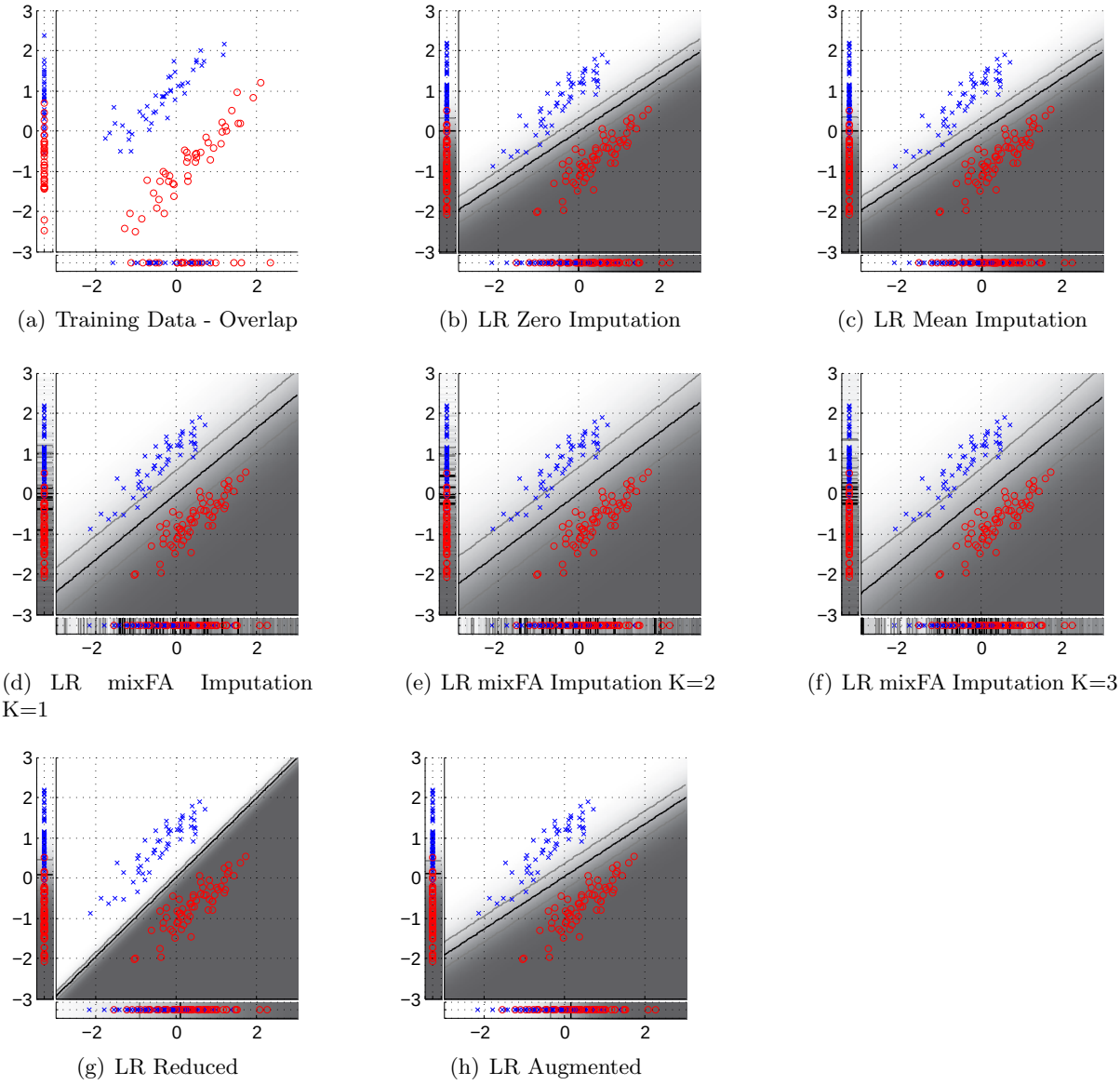


(g) LR Reduced



(h) LR Augmented

Figure 6.3: This figure shows results for the Mix data set based on several missing data methods combined with logistic regression. Missing data methods include zero imputation, mean imputation, multiple imputation, reduced models, and response indicator augmentation. The results show that zero imputation and mean imputation perform poorly when they result in non-separable configurations in the original feature space. The multiple imputation results again show that the imputation model can be incorrect ($K = 1$), yet still give good estimates of the conditional densities $P(\mathbf{x}^m|\mathbf{x}_n^o)$. The response augmented model result shows that learning an optimal axis aligned embedding can be superior to mean imputation.



(a) Training Data - Overlap (b) LR Zero Imputation (c) LR Mean Imputation

(d) LR mixFA Imputation K=1 (e) LR mixFA Imputation K=2 (f) LR mixFA Imputation K=3

(g) LR Reduced (h) LR Augmented

Figure 6.4: This figure shows results for the Overlap data set based on several missing data methods combined with logistic regression. Missing data methods include zero imputation, mean imputation, multiple imputation, reduced models, and response indicator augmentation. The results show that when the classes are highly overlapping when a data dimensions is missing, no missing data method can perform well including multiple imputation methods.

	Simple		Mix		Overlap	
	Loss	Err(%)	Loss	Err(%)	Loss	Err(%)
LR Zero	0.0463	1.75	0.3336	14.50	0.3137	13.75
LR Mean	0.0470	1.75	0.1693	6.50	0.3139	13.50
LR Mix K=1	0.0687	2.00	0.1053	3.75	0.3940	17.75
LR Mix K=2	0.0655	1.50	0.1052	2.50	0.4098	19.00
LR Mix K=3	0.0501	1.00	0.1038	3.50	0.4067	15.00
LR Reduced	0.0433	1.75	0.0746	2.75	0.2926	13.50
LR Augmented	0.0498	2.00	0.1019	3.25	0.3180	13.75

Table 6.2: Summary of illustrative results for linear logistic regression with missing data including, zero imputation, mean imputation, multiple imputation with mixtures of factor analyzers, reduced models, and response vector augmentation. We report the log loss (average negative log probability of the correct class), as well as the average classification error.

tion Maximization algorithm. We sample one value of $\mathbf{x}_n^m$ given $\mathbf{x}_n^o$ for each data case n and each imputation model. We regularize the logistic regression model using an $L2$ penalty on the weights. The regularization trade off parameter is set by a five fold cross validation over the range 10^{-2} to 10^{-5}. Each method was run for a maximum of 1000 training iterations.

The classification functions learned using each method are shown in Figures 6.2, 6.3, and 6.4 for the Simple, Mix, and Overlap data sets respectively. The results are summarized in Table 6.2. The results on the Simple data set show that all methods achieve very similar classification performance. The fact that zero imputation performs as well as the more sophisticated methods is due to the fact that when missing data is embedded along the axes, the Simple data set is nearly linearly separable in the original feature space. A similar argument applies to mean imputation for the Simple data set. The multiple imputation results show much smoother classification functions than any of the other methods. This results from a combination of noise due to sampling variation in the imputations, as well as from the fact that each classification function results from an ensemble of logistic regression classifiers. The multiple imputation results also show that multiple imputation can perform well even if the imputation model is incorrect. There is little difference in the classification functions based on a one component factor analysis model, and a two component factor analysis mixture. The reason for this behaviour is explained by the fact that if a single Gaussian is used to explain both clusters in the Simple training data set, the conditional densities $P(\mathbf{x}^m|\mathbf{x}_n^o)$ are approximately correct for most data cases, even though a two component mixture gives a much better fit to the data.

The Mix data set clearly illustrates the problems that can result when zero imputation and unconditional mean imputation are used. In the Mix data set, embedding missing data along the axes in the original feature space interferes with the data cases that are fully observed. The same problem occurs when mean imputation is used, although embedding missing data

cases along the mean values is somewhat better than embedding along the axes. Multiple imputation again works well with one, two, or three clusters. The explanation is the same as for the Simple data set: the conditional densities $P(\mathbf{x}^m|\mathbf{x}_n^o)$ are correct for most data points. Comparing the response indicator framework to unconditional mean estimation shows that the two are not equivalent. The response indicator framework essentially learns the optimal axis aligned embedding of the missing data at the same time it learns the classification function. This results in improved accuracy relative to unconditional mean estimation.

The Overlap data set again illustrates the fact that if the classes overlap significantly when some data dimensions are missing, no classification method can perform well. Any axis aligned embedding of the missing data into the original feature space results in a highly non-separable arrangement of data points. This adversely affects zero imputation, mean imputation, and the response indicator framework. Even when an appropriate number of mixture components are used, the conditional densities $P(\mathbf{x}^m|\mathbf{x}_n^o)$ are highly bimodal for most data cases. Sampling will place data points in one cluster or the other essentially at random.

6.4 Perceptrons and Support Vector Machines

Both Linear Discriminant Analysis and Logistic Regression are examples of hyperplane classifiers. A potential drawback of these methods is that neither directly optimizes classification error as an objective function. Linear discriminant analysis optimizes the joint probability of features and labels, while logistic regression optimizes the conditional probability of labels given features. In this section we review perceptrons and support vector machines. The classical perceptron is a hyperplane classifier that attempts to directly minimize classification error. The support vector machine is a hyperplane classifier that selects an optimal hyperplane by trading off classification error against a complexity penalty.

6.4.1 Perceptrons

The perceptron is a binary linear classifier that chooses a hyperplane by directly optimizing classification error, as seen in Equation 6.4.1 [63]. The original perceptron learning rule introduced by Rosenblatt is a stochastic gradient descent algorithm [63]. Data cases are presented one at a time, and the corrections to the parameters shown in Equation 6.4.2 is made if the current data case is misclassified. The parameter α is a learning rate.

$$f_{Pct}(\mathbf{w}, b) = \sum_{n=1}^{N} \frac{1}{2}|y_n - \text{sgn}(\mathbf{w}^T\mathbf{x}_n + b)| \tag{6.4.1}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha y_n \mathbf{x}_n \tag{6.4.2}$$

$$b \leftarrow b + \alpha y_n \tag{6.4.3}$$

The algorithm is run until a pass through the complete training set results in no updates to the parameters. The interpretation of the single-layer perceptron in terms of a separating hyperplane, along with the limitations this implied, was only understood later by Minsky and Papert [56].

Rosenblatt's perceptron learning algorithm is guaranteed to converge in a finite number of steps if the training data is linearly separable [32, p. 108]. However, the solution that is returned is essentially an arbitrary member of the set of all hyperplanes yielding perfect classification of the training data. The hyperplane obtained when the perceptron learning algorithm halts depends on the initial parameter values, as well as the order in which that training cases are presented. The non-uniqueness of the solution in the separable case can be eliminated by requiring that the hyperplane satisfy additional optimality constraints. This is the approach taken by the support vector machine classifier discussed in the following section.

If the training data is not linearly separable the perceptron learning algorithm will not converge, and cycles will develop in the parameter values. Several methods have been developed to deal with the problem of non-separable training sets including Freund and Schapire's voted perceptron method [21], and the soft margin support vector machine.

6.4.2 Hard Margin Support Vector Machines

As mentioned in the previous section, there is an infinite set of hyperplanes that can perfectly classify a separable data set. The hyperplane returned by the perceptron learning rule is essentially a random member of this set. The optimal separating hyperplane, as defined by Vapnik, is the one that maximizes the minimum distance of any training data point to the separating hyperplane as seen in equation 6.4.4 [76].

Since the same decision rule results if the parameters $\mathbf{w}$ and b are multiplied by a constant, any scaling of the optimal $\mathbf{w}$ can be used. In this case $\mathbf{w}$ and b are scaled such that $||\mathbf{w}|| = 1$. Equation 6.4.5 gives an alternate formulation for the optimal hyperplane problem. The utility of this formulation is that it is a standard, convex quadratic optimization problem subject to linear constraints.

$$f_H^P(\mathbf{w}, b) = \max_{\mathbf{w},b} \min_n y_n(\mathbf{w}^T\mathbf{x}_n + b) \qquad \text{... st } ||\mathbf{w}|| = 1 \tag{6.4.4}$$

$$= \min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 \qquad \text{... st } \forall n\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \tag{6.4.5}$$

6.4.3 Soft Margin Support Vector Machines

The main issue with the hard margin formulation of the support vector machine is that it is only useful in the case where the data is perfectly separable. One solution to this problem is to modify the constraints in equation 6.4.5 to the form $y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq (1 - \eta_n)$ where η_n is a non-negative slack variable [32, p. 373]. This essentially relaxes the constraints, and ensures that the primal optimization problem always has a feasible solution. Penalizing the use of slack leads to the optimization problem in Equation 6.4.6. S is a free tradeoff parameter.

$$f_S(\mathbf{w}, b, \boldsymbol{\eta}) = \min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + S\sum_{n=1}^{N} \eta_n \qquad \text{... st } \forall n\ y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq (1 - \eta_n),\ \eta_n > 0 \tag{6.4.6}$$

6.4.4 Soft Margin Support Vector Machine via Loss + Penalty

The standard derivation for the soft margin support vector machine tends to obscures the relationship between the support vector machine and logistic regression. The soft margin support vector machine can also be derived in an alternative regularization framework using a particular loss function and penalty term [32, p. 380]. As we noted previously, regularized logistic regression can be derived from a combination of the logistic loss function shown in equation 6.3.3, and an L1 or L2 penalty term on the weight vector. The soft margin support vector machine can be derived from a combination of the hinge loss shown in equation 6.4.7, and an L2 penalty on the weight vector.

$$L_H(y_n, \mathbf{w}^T\mathbf{x}_n + b) = [1 - y_n(\mathbf{w}^T\mathbf{x}_n + b)]_+ \tag{6.4.7}$$

$$[a]_+ = \max(a, 0); \tag{6.4.8}$$

The primal objective function is given in Equation 6.4.9. Since the loss term is discontinuous this optimization problem is difficult to solve directly. In the second step below we introduce a set of auxiliary variables η_n to represent the value of the hinge loss function for each data case. We constrain these variables to be exactly equal to the corresponding hinge loss values. The formulation now has a continuous objective function with the discontinuity moved into the constraints. In the final step we recognize that the equality constraints can be replaced by

linear inequality constraints, and that the larger of the two inequalities will be saturated at the minimum. If that was not the case we could lower the value of η_n for some n and reduce the objective function further, contradicting the fact that we are at a minimum.

$$f_{S'}(\mathbf{w}, b) = \min_{\mathbf{w},b} \sum_{n=1}^{N} [1 - y_n(\mathbf{w}^T\mathbf{x}_n + b)]_+ \gamma ||\mathbf{w}||^2 \tag{6.4.9}$$

$$\begin{aligned} f_{S'}(\mathbf{w}, b, \boldsymbol{\eta}) &= \min_{\mathbf{w},b,\boldsymbol{\eta}} \sum_{n=1}^{N} \eta_n + \gamma ||\mathbf{w}||^2 \\ &\quad \text{st} \quad \forall n \ \eta_n = \begin{cases} 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) & \ldots \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) > 0 \\ 0 & \ldots \quad 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b) \leq 0 \end{cases} \end{aligned} \tag{6.4.10}$$

$$= \min_{\mathbf{w},b} \sum_{n=1}^{N} \eta_n + \gamma ||\mathbf{w}||^2 \ \ldots \ \text{st} \quad \forall n \ \eta_n \geq 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b), \ \eta_n \geq 0 \tag{6.4.11}$$

It is easy to see that the final form of the optimization problem derived from Equation 6.4.9 is equivalent to the soft margin support vector machine formulation given in Equation 6.4.6 for equivalent choices of the tradeoff parameters γ and S. In this work we focus on the use of logistic regression instead of SVMs. Do to the similarity between support vector machines an logistic regression, we expect results to generalize well from logistic regression to SVMs.

6.5 Basis Expansion and Kernel Methods

The classification methods we have considered to this point allow for a very limited class of decision surfaces. Linear discriminant analysis, linear logistic regression, perceptrons, and linear support vector machines all have decision surfaces described by hyperplanes. Of the methods we have discussed, only quadratic discriminant analysis and some forms of regularized discriminant analysis have non-linear decision surfaces.

In statistics the restrictions imposed by linear models are often overcome through the use of higher order interaction terms [15, p. 191]. Consider the case when the feature vector $\mathbf{x}$ is two dimensional, for example. Linear logistic regression finds the hyperplane $w_1x_1 + w_2x_2 + b$ that maximizes the conditional probability of the labels. If there is a strong interaction between x_1 and x_2, then an interaction term of the from x_1x_2 can be included and an optimal hyperplane of the form $w_1x_1 + w_2x_2 + w_3x_1x_2 + b$ will be found. Basis expansions and kernel methods generalize this basic strategy for converting a linear classifier into a non-linear classifier. The resulting learning algorithms are often convex for a fixed basis expansion or kernel function.

6.5.1 Basis Expansion

The idea of interaction terms can be generalized by replacing the original feature vector $\mathbf{x}$ with a derived feature vector $\mathbf{z} = [\phi_1(\mathbf{x}), ..., \phi_M(\mathbf{x})]$. Each $\phi_i(\mathbf{x})$ is a function that maps a D dimensional vector to a scalar value. More generally, we can assume that there exists a mapping $\mathbf{z} = \Phi(\mathbf{x})$ that maps D dimensional feature vectors to M dimensional derived feature vectors. Normally M will be greater than D, in which case the vector $\mathbf{z}$ is referred to as a basis expansion of $\mathbf{x}$ [32, p. 115].

Under the assumption that the function $\Phi(\mathbf{x})$ is fixed, the vector $\mathbf{z}_n$ can be computed for each $\mathbf{x}_n$. Standard methods for fitting linear models can then be applied to the data in the expanded basis. The result is a model that is linear in the expanded basis. However, if any of the basis functions $\phi_i(\mathbf{x})$ are non-linear, the resulting model will be non-linear in the original features $\mathbf{x}$. This is a useful way to obtain non-linear models while retaining the ease of fitting linear models. The obvious problem with this approach is that it assumes an appropriate set of basis functions is known a priori.

6.5.2 Kernel Methods

In many linear classification models the optimal parameters can be expressed in terms of a linear combination of data cases. As a result, the corresponding classification rule depends only on inner products between data vectors. In the case of logistic regression and perceptrons, it is clear from the optimization algorithm that the resulting optimal weight parameters can be expressed as a linear combination of data vectors. In the case of support vector machines, an analysis of the dual optimization problem shows that the optimal parameters can be expressed as a linear combination of data vectors.

When a basis expansion is introduced, the optimal weight vector depends only on a linear combination of the expanded vectors $\Phi(\mathbf{x}_n)$. The fact that the optimal parameters depend only on inner products between the $\Phi(\mathbf{x}_n)$ means we never need to explicitly compute the value of $\Phi(\mathbf{x}_n)$ for a single data case in isolation. It suffices to have a kernel function $\mathcal{K}(\mathbf{x}_i, \mathbf{x_j}) = \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}_j)$ that returns the required inner products. Further, it is possible to directly specify a kernel function, and prove that it corresponds to an inner product under some mapping $\Phi(\cdot)$, without explicitly constructing that mapping $\Phi(\cdot)$.

In order for a kernel function to correspond to a valid inner product for some mapping $\Phi(\cdot)$ it suffices for the kernel function to be positive definite. A kernel $\mathcal{K}(\cdot, \cdot)$ is positive definite if for any N and any $\mathbf{x}_1, ..., \mathbf{x}_N$, the induced Gram matrix $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ is positive definite. A real-valued $N \times N$ Gram matrix $\mathbf{K}$ is positive definite if it is symmetric, and for any choice of vector $\mathbf{v} \in \mathbb{R}^N$ the quadratic form $\mathbf{v}^T \mathbf{K} \mathbf{v} \geq 0$ [68, p. 30].

6.5.3 Kernel Support Vector Machines and Kernel Logistic Regression

In the kernel support vector machine we replace the optimal weights by the optimal linear combination of expanded data vectors $\Phi(\mathbf{x}_n)$: $\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \Phi(\mathbf{x}_n)$. This leads to a decision surface based only on the corresponding kernel function $\mathcal{K}(\cdot, \cdot)$ as seen in equation 6.5.1.

$$\mathbf{w}^T \Phi(\mathbf{x}_n) + b = \sum_{n'=1}^{N} \alpha_{n'} y_{n'} \Phi(\mathbf{x}_{n'})^T \Phi(\mathbf{x}_n) + b = \sum_{n'=1}^{N} \alpha_{n'} y_{n'} \mathcal{K}(\mathbf{x}_{n'}, \mathbf{x}_n) + b \tag{6.5.1}$$

The optimal weight vector and decision surface for binary kernel logistic regression can also be expressed as seen in Equation 6.5.1. However, the optimal parameters are found by minimizing a regularized logistic loss function. In Equation 6.5.2 we shown the objective function for L2 regularized kernel logistic regression in the multiclass case.

$$f_L(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha}} - \sum_{n=1}^{N} \sum_{c=1}^{C} [y_n = c] \log \left(\frac{\exp(\sum_{n'=1}^{N} \alpha_{n'c} \mathcal{K}(\mathbf{x}_{n'}, \mathbf{x}_n) + b_c)}{\exp(\sum_{c'=1}^{C} \sum_{n'=1}^{N} \alpha_{n'c'} \mathcal{K}(\mathbf{x}_{n'}, \mathbf{x}_n) + b_{c'})} \right) + \gamma \sum_{c=1}^{C} \boldsymbol{\alpha}_c^T \mathbf{K} \boldsymbol{\alpha}_c \tag{6.5.2}$$

One of the main advantages of the kernel support vector machine is that a number of α_n are usually exactly zero. This is a result of the hinge loss being exactly equal to zero for data cases on the correct side of the margin. L2 regularized kernel logistic regression does not share this property. In general all data cases will have non-zero α_n.

An advantage of kernel logistic regression is that it naturally estimates classification probabilities while kernel support vector machines do not. Under certain conditions, kernel logistic regression also has margin maximizing properties similar to kernel support vector machines [80]. As we have shown, kernel logistic regression naturally generalizes to the multiclass setting simply by replacing the logistic loss by a loss function derived from the softmax function. We perform experiments exclusively with kernel logistic regression, although we again expect similar results using kernel support vector machines.

The primal L2 regularized kernel logistic regression optimization problem is unconstrained, and can be solved directly using first or second order non-linear optimization methods. The dual Sequential Minimal Optimization (SMO) method for kernel logistic regression developed by Keerthi et al. reportedly achieves significant increases in speed compared to standard first and second order optimization methods applied to the primal optimization problem [46]. However, the SMO method has only been developed for the binary classification case. We use a simple primal optimization method in all experiments.

6.5.4 Kernels For Missing Data Classification

The main problem in the application of kernel methods is the choice of an appropriate kernel function. In the present context this choice is complicated by the fact that the kernel function must be defined even in the presence of missing feature values. To make the dependence on the response indicators explicit we consider kernels of the form $K(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j)$.

Linear Kernels

The linear kernel K_l^o shown in equation 6.5.3 is a positive definite kernel since it corresponds to the standard inner product in $\mathbb{R}^D$ when the data vector and response indicator vector are mapped to $\mathbf{x} \odot \mathbf{r}$. $\odot$ denotes the element-wise product. This mapping simply replaces missing values by zeros, and is one of the kernel studied by Chechik et al. [13].

$$K_l^o(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \sum_d r_{di} r_{dj} x_{di} x_{dj} \tag{6.5.3}$$

The explicit representation of missing data in terms of response indicators leads us immediately to the new linear kernel K_l^{o+r} shown in equation 6.5.4. Again, this kernel can immediately be seen as positive definite since it corresponds to the standard inner product in $\mathbb{R}^{2D}$ when the data vector and response indicator vector are mapped to $[\mathbf{x} \odot \mathbf{r}, \sqrt{\gamma}\mathbf{r}]$. Performing kernel logistic regression using the linear kernel K_l^o corresponds to performing linear logistic regression combined with zero imputation. Performing kernel logistic regression using the linear kernel K_l^{o+r} corresponds to performing logistic regression under the response indicator framework.

$$K_l^{o+r}(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \sum_d r_{di} r_{dj} x_{di} x_{dj} + \gamma r_{di} r_{dj} \tag{6.5.4}$$

Polynomial Kernels

Given these two positive definite linear kernels, we can obtain a pair of more interesting positive definite inhomogeneous polynomial kernels through the usual construction [68, p. 46]. The kernels are given in equations 6.5.5 and 6.5.6. Positive definiteness of K_p^o and K_p^{o+r} follows from the positive definiteness of K_l^o and K_l^{o+r}. Chechik et al. have also studied the K_p^o kernel [13].

$$K_p^o(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \left(\kappa + \sum_d r_{di} r_{dj} x_{di} x_{dj}\right)^\delta \tag{6.5.5}$$

$$K_p^{o+r}(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \left(\kappa + \sum_d r_{di} r_{dj} x_{di} x_{dj} + \gamma r_{di} r_{dj}\right)^\delta \tag{6.5.6}$$

Gaussian Kernels

An interesting open problem is the development of a Gaussian-like kernel for use with missing data. Following the development of the observed data linear and polynomial kernels, the obvious extension of the gaussian kernel to the case of missing data is given in equation 6.5.7.

$$K_g^o(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \exp(-\frac{1}{\sigma^2} D^o(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j)^2) \tag{6.5.7}$$

$$D^o(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \left(\sum_d r_{di} r_{dj} (x_{di} - x_{dj})^2\right)^{\frac{1}{2}} \tag{6.5.8}$$

Like the linear and polynomial kernels, K_g^o only takes into account features that are observed both in $\mathbf{x}_i$ and $\mathbf{x}_j$. K_g^o can also be thought of as a distance substitution kernel where the standard euclidean distance has been replaced by euclidean distance $D^o()$ within the intersection of the observed feature spaces of $\mathbf{x}_i$ and $\mathbf{x}_j$. However, $D^o()$ is not a metric since it does not satisfy the triangle inequality. This can be shown by counter example. As a result, K_g^o can not be positive definite [31]. In addition K_g^o strongly violates even an intuitive notion of similarity since two data vectors with no overlap in their observed feature spaces have a distance of 0 under $D^o()$, and thus a similarity of 1 according to K_g^o.

Now consider the gaussian-like kernel K_g^{o+r} given in equation 6.5.9. D^{o+r} measures the distance between data points according to euclidean distance in the intersection of the observed feature spaces, and adds a penalty term γ for each dimension where only one of the two data vectors is observed.

$$K_g^{o+r}(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \exp(-\frac{1}{\sigma^2} D^{o+r}(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j)^2) \tag{6.5.9}$$

$$D^{o+r}(\mathbf{x}_i, \mathbf{r}_i, \mathbf{x}_j, \mathbf{r}_j) = \left(\sum_d r_{di} r_{dj} (x_{di} - x_{dj})^2 + \gamma_d (r_{di} - r_{dj})^2\right)^{\frac{1}{2}} \tag{6.5.10}$$

We first note that if γ_d is set to positive infinity for all d, K_g^{o+r} will always result in a positive

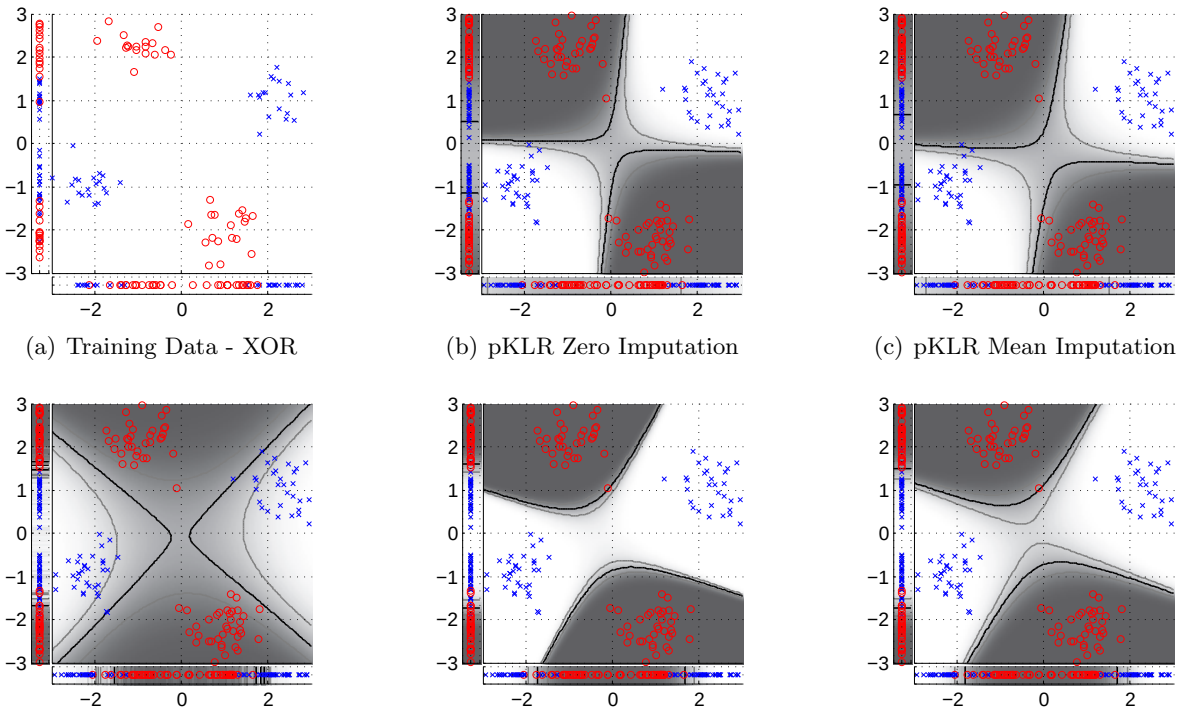
	XOR	
	Loss	Err(%)
pKLR Zero	0.3660	25.75
pKLR Mean	0.3660	26.50
pKLR Mix	0.2046	5.75
pKLR Reduced	0.1345	6.25
pKLR Augmented	0.1467	5.50
gKLR Zero	0.1932	7.00
gKLR Mean	0.2271	8.25
gKLR Mix	0.2055	5.25
gKLR Reduced	0.1402	5.75
gKLR Augmented	0.1677	5.25

Table 6.3: Summary of illustrative results for kernel logistic regression combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response indicator framework. We report the log loss (average negative log probability of the correct class), as well as the average classification error.

definite kernel matrix. For any two data points that are defined in the same feature subspace, the value of K_g^{o+r} is equal to the standard Gaussian Kernel applied in that sub-space. For any two data points that are defined in different subspaces, the value of K_g^{o+r} will be 0. Permuting the kernel matrix so that all the rows and columns corresponding to data points from the same subspace are contiguous results in a block diagonal matrix where each block is positive definite. As a result, the complete Kernel matrix is positive definite.

Performing kernel logistic regression using the K_g^{o+r} kernel with $\gamma_d = \infty$ has some interesting properties. First, since the K_g^{o+r} kernel matrix is block diagonal, using it to perform kernel logistic regression is similar to separately computing a kernel logistic regression classifier in each observed data subspace using only the data specifically from that sub-space. This is closely related to the reduced models approach of Sharp and Solly [69].

The negative side of using the K_g^{o+r} kernel with $\gamma_d = \infty$ is that there is no regularization across feature subspaces. In the extreme case of a data set with every data point in a unique subspace, no regularization would be achieved at all. The solution to this problem is to consider K_g^{o+r} for general γ, but it can again be shown that D^{o+r} fails to satisfy the triangle inequality in general. However, it may be possible to construct a positive definite kernel matrix using a small values of γ_d for a fixed data set, even though the kernel function is not positive definite. Another option is to construct the kernel matrix and explicitly test if it is positive definite. Indefinite kernel matrices can be made positive semi-definite by modifying the eigen-decomposition to eliminate negative eigen values.



(a) Training Data - XOR (b) pKLR Zero Imputation (c) pKLR Mean Imputation

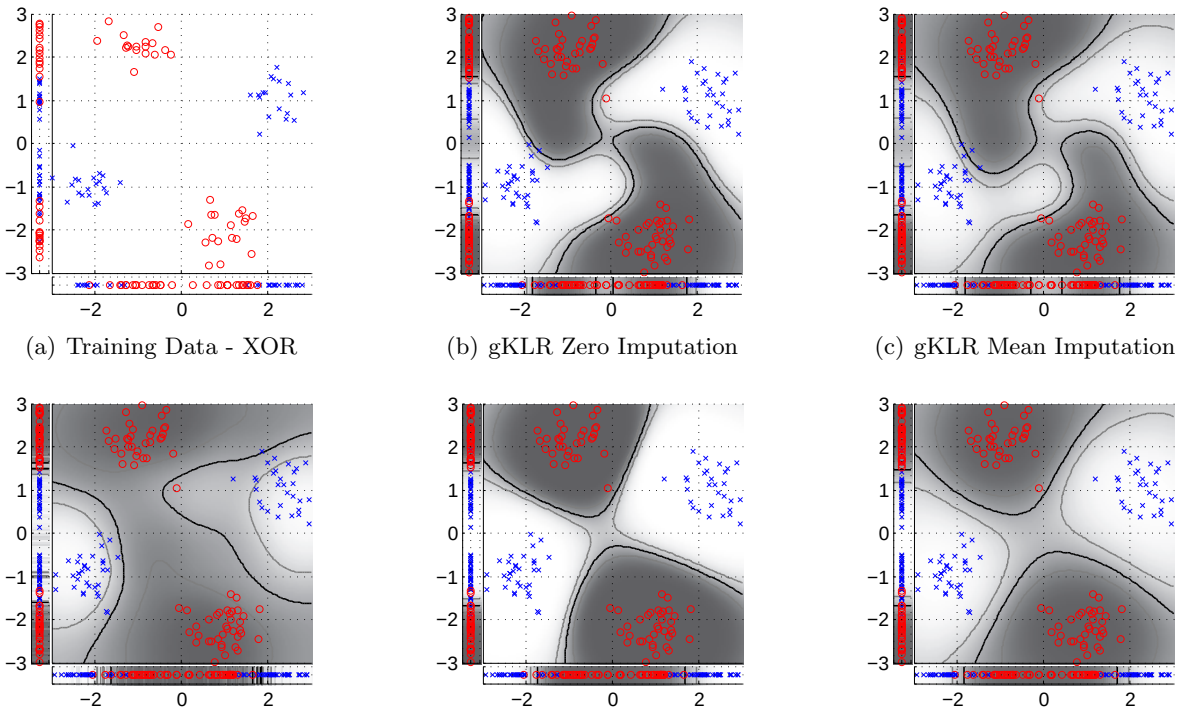
(d) pKLR mixFA Imputation (e) pKLR Reduced (f) pKLR Augmented

Figure 6.5: This figure shows results for the XOR data set based on several missing data methods combined with degree two polynomial kernel logistic regression. Missing data methods include zero imputation, mean imputation, multiple imputation, reduced models, and response indicator augmentation. The results show poor performance using zeros imputation and mean imputation. Multiple imputation, reduced models, and the response indicator framework all give similar accuracy. The multiple imputation classification function is again much smoother.

6.5.5 Synthetic Data Experiments and Results

To illustrate the performance of non-linear classification methods, a two class Gaussian mixture data set closely related to the binary exclusive or (XOR) problem was created. The first class has means located at $[2.2, 1]$ and $[-2.2, -1]$. The second class has means located at $[-1, 2.2]$ and $[1, -2.2]$. A spherical covariance matrix with variance equal to 0.2 was used for both Gaussian components in each class. A total of 100 training points and 200 test points were sampled from each class. Missing data was created by randomly assigning each data case to one of the three response patterns $[1, 1]$, $[1, 0]$, $[0, 1]$. We refer to this data set as the "XOR" data set. It is pictured in Figure 6.5(a).

A degree two polynomial kernel logistic regression method was combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response indicator framework. The standard polynomial kernel was used in all cases except for the response indicator framework where the K_p^{o+r} kernel was used with γ fixed to 1. A Gaussian kernel logistic



(a) Training Data - XOR (b) gKLR Zero Imputation (c) gKLR Mean Imputation

(d) gKLR mixFA Imputation (e) gKLR Reduced (f) gKLR Augmented

Figure 6.6: This figure shows results for the XOR data set based on several missing data methods combined with Gaussian kernel logistic regression. Missing data methods include zero imputation, mean imputation, multiple imputation, reduced models, and response indicator augmentation. Gaussian kernel logistic regression is better able to deal with zero and mean imputation than the more rigid polynomial kernel. Multiple imputation, reduced models, and the response indicator framework yield similar classification functions and similar accuracy, with multiple imputation again leading to a smoother classification function.

regression method was also combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response indicator framework. The standard Gaussian kernel with $\sigma^2 = 2$ was used in all cases except for the response indicator framework where the K_g^{o+r} kernel was used with γ_d fixed to 1 for all d, and σ^2 fixed to 2. The regularization parameter for both the polynomial and Gaussian methods was set by five fold cross validation over the range 10^{-2} to 10^{-5}. The multiple imputation results are based on a mixture of factor analyzers model with latent dimensionality $Q = 1$, and $K = 6$ mixture components. Each method was run for a maximum of 1000 training iterations.

The resulting classification functions are shown in Figures 6.5 and 6.6. Classification performance is summarized in Table 6.3. The results show that degree two polynomial kernel logistic regression is unable to deal with zero and mean imputed missing data. This is expected sice the model does not have sufficient capacity to account for missing data embedded along the axes. Gaussian kernel logistic regression is better able to cope with zero and mean im-

puted missing data, but it is clear that both of these imputation strategies adversely affect the resulting classification function. Polynomial kernel regression does have sufficient capacity to accurately classify the Mix data set when the imputations are more accurate, or reduced models are used. Polynomial kernel logistic regression also works well in this case when response indicator augmentation is used. A similar pattern holds for Gaussian kernel logistic regression where multiple imputation, reduced models, and the response indicator framework yield similar results and similar classification functions.

6.6 Neural Networks

Neural networks are a biologically inspired computational framework where a number of simple computational units, the neurons, are connected together to compute a more complex function. Neural networks can be characterized by the topology of the network, the dynamics of the network, and the activation function computed at each node. The key property of neural networks is that the activation function computed at each node is a parametric function. In the classification setting, neural networks learn to classify inputs by adapting the parameters associated with each node in the network. In this section we review sigmoid multi-layer feed-forward neural networks, and their application to classification with missing data.

6.6.1 Feed-Forward Neural Network Architecture

The network topology of a feed-forward neural network is constrained to be a directed acyclic graph. The nodes are typically arranged in one or more layers, with connections between layers, but not within layers. The first layer is called the input layer, and the last layer is called the output layer. Any intermediate layers are referred to as hidden layers. The parents of a node in the network are referred to as that node's inputs. We will assume that each layer is fully connected to the subsequent layer. At time step t each node in layer t computes its output value by applying its local activation function to its inputs. The assumption that the network topology is a directed acyclic graph insures that the input values needed at each time step have already been computed, and are not affected by computations at higher levels in the network.

The activation function computed at each node is typically a function of a linear combination of its inputs. In this case the directed edges connecting nodes in the network can be thought of as having weights. Typical functions applied to the linear combination of inputs include the identity, sign, logistic or sigmoid, and hyperbolic tangent functions.

6.6.2 One Hidden Layer Neural Networks for Classification

In this section we develop a learning algorithm for feed forward neural networks with one hidden layer. We denote the inputs by x_{dn}, the first layer activations by A^1_{kn}, the hidden unit values by h_{kn}, the second layer activations by A^2_{cn}, and the outputs by $\hat{y}_{cn}$. We denote the number of nodes in the hidden layer by K. The first layer weight from input dimension d to hidden unit k is denoted W^1_{dk}. The second layer weight from hidden unit k to output unit c is denoted by W^2_{kc}. We denote the hidden unit biases by b^1_k, and the output unit biases by b^2_c. We select a sigmoid or logistic activation function $\sigma(\cdot)$ for the hidden units as shown in Equation 6.6.1. We assume a multiclass classification setting and choose a softmax activation function at the output layer coupled with a cross-entropy objective function as shown in Equation 6.6.2.

$$\sigma(x) = \frac{1}{1 + \exp -x} \tag{6.6.1}$$

$$\mathcal{L} = -\sum_{n=1}^{N}\sum_{c=1}^{C}[y_n = c]\log(\hat{y}_{cn}) \tag{6.6.2}$$

The output of a feed forward neural network is computed using a forward pass through the network from the input layer to the output layer as seen in Equation 6.6.3.

$$A^1_{kn} = \sum_{d=1}^{D} W^1_{dk}x_{dn} + b^1_k \tag{6.6.3}$$

$$h_{kn} = \sigma(A^1_{kn}) \tag{6.6.4}$$

$$A^2_{cn} = \sum_{k=1}^{K} W^2_{kc}h_{kn} + b^2_c \tag{6.6.5}$$

$$\hat{y}_{cn} = \frac{\exp(A^2_{cn})}{\sum_{c'=1}^{C}\exp(A^2_{c'n})} \tag{6.6.6}$$

The parameters of the network are adapted using gradient descent on the objective function given in Equation 6.6.2. The back-propagation algorithm [32, p. 353] is an efficient method for computing the required gradients by exploiting the structure of the network. The weights are fixed and a forward pass is used to compute and store the activation, hidden unit, and output values. The derivatives with respect to the weights, biases, and activation values are then computed and stored using a backward pass through the network. The gradients for the current network are given in Equations 6.6.7 to 6.6.11.

$$\frac{\partial \mathcal{L}}{\partial A^2_{cn}} = \hat{y}_{cn} - [y_n = c] \tag{6.6.7}$$

$$\frac{\partial \mathcal{L}}{\partial W^2_{kc}} = \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial A^2_{cn}} \frac{\partial A^2_{cn}}{\partial W^2_{kc}} = \sum_{n=1}^{N} (\hat{y}_{cn} - [y_n = c]) h_{kn} \tag{6.6.8}$$

$$\frac{\partial \mathcal{L}}{\partial b^2_{c}} = \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial A^2_{cn}} \frac{\partial A^2_{cn}}{\partial b^2_{c}} = \sum_{n=1}^{N} (\hat{y}_{cn} - [y_n = c]) \tag{6.6.9}$$

$$\frac{\partial \mathcal{L}}{\partial W^1_{dk}} = \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{\partial \mathcal{L}}{\partial A^2_{cn}} \frac{\partial A^2_{cn}}{\partial h_{kn}} \frac{\partial h_{kn}}{\partial A^1_{kn}} \frac{\partial A^1_{kn}}{\partial W^1_{dk}} = \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{\partial \mathcal{L}}{\partial A^2_{cn}} W^2_{kc} h_{kn}(1 - h_{kn}) x_{dn} \tag{6.6.10}$$

$$\frac{\partial \mathcal{L}}{\partial b^1_{k}} = \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{\partial \mathcal{L}}{\partial A^2_{cn}} \frac{\partial A^2_{cn}}{\partial h_{kn}} \frac{\partial h_{kn}}{\partial A^1_{kn}} \frac{\partial A^1_{kn}}{\partial b^1_{k}} = \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{\partial \mathcal{L}}{\partial A^2_{cn}} W^2_{kc} h_{kn}(1 - h_{kn}) \tag{6.6.11}$$

6.6.3 Special Cases of Feed-Forward Neural Networks

Rosenblatt's perceptron discussed in Section 6.4.1 is considered to be the earliest example of a feed-forward neural network [63]. The network consists of the input layer, and one output node. The activation function at the output node is the sign of a linear combination of the inputs. The loss function used by the perceptron is classification error. Since the activation function used by the perceptron has a discontinuous first derivative, the back-propagation algorithm can not be used. Instead, the Perceptron Learning Rule adapts the weights or connection strengths in the network in response to classification errors.

Logistic regression discussed in Section 6.3 can also be interpreted in terms of a neural network. In the multi-class case logistic regression corresponds to a network with an input layer, an output layer with C nodes, and no hidden layer. The activation function used at each output is the softmax function. The loss function used is cross-entropy.

6.6.4 Regularization in Neural Networks

Several regularization techniques are commonly used to improve neural network learning. Weight decay corresponds placing an L2 penalty on the network weights $\mathbf{w}$. L1 regularization is less commonly used with neural networks due to the discontinuity in the penalty function.

Momentum is a heuristic optimization strategy that modifies the current gradient by mixing in a portion of the previous gradient. The use of momentum in gradient descent is often motivated by a physical analogy to a ball rolling down a hill. If the ball has no momentum it will be trapped by any local minimum, no matter how shallow. If the ball has momentum it can skip over shallow local minima, and continue down the hill until it falls into a deep minimum.

Early stopping is a heuristic optimization technique used to avoid over-fitting in neural net-

	XOR	
	Loss	Err(%)
NN Zero	0.3114	11.75
NN Mean	0.1781	7.25
NN Mix	0.2124	4.50
NN Reduced	0.1610	5.50
NN Augmented	0.1847	5.75

Table 6.4: Summary of illustrative results for multi-layer neural networks combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response indicator framework. We report the log loss (average negative log probability of the correct class), as well as the average classification error.

works. The optimization is typically started with all weights set to small random values. When the weights are small typical activation functions including the sigmoid function behave as if they were linear. As the optimization proceeds some weights increase in magnitude thereby introducing non-linearities into the network, and increasing its capacity. Starting the optimization process with small weights and stopping before convergence limits the capacity of the network, and provides a form of regularization.

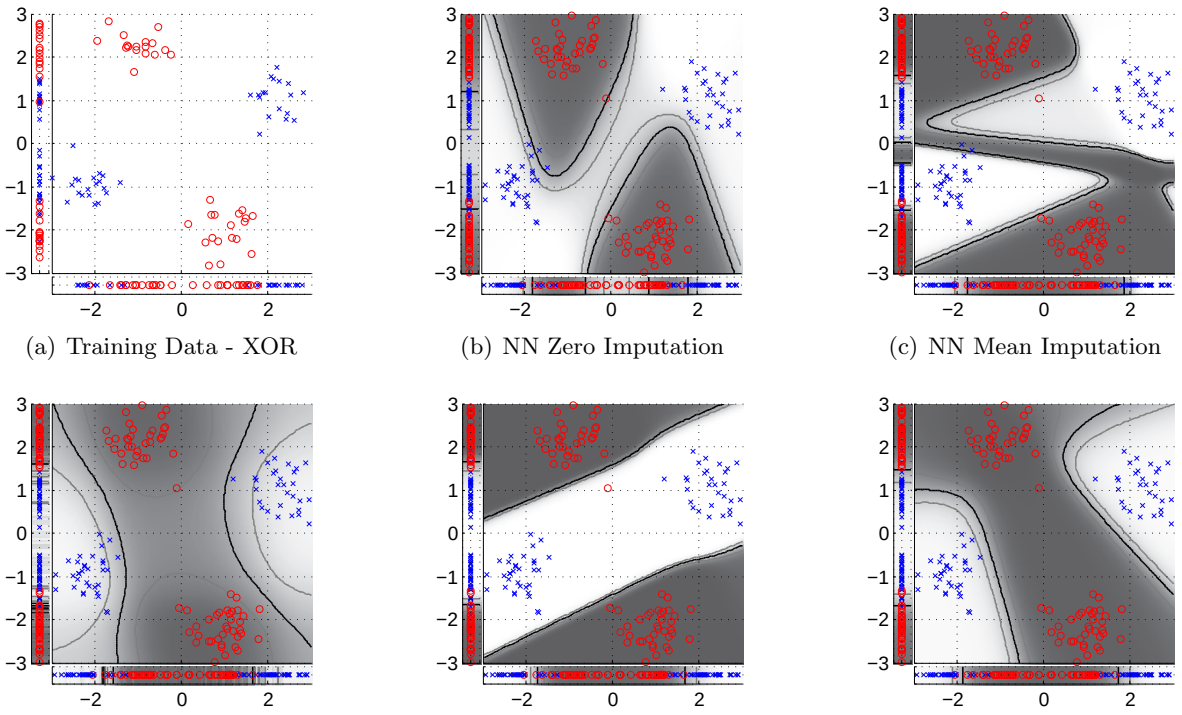
6.6.5 Neural Network Classification and Missing Data

The imputation and reduced models strategies can be combined with multi-layer neural networks in exactly the same way as logistic regression. When combining neural networks with multiple imputation we train a separate network for each imputation model. When combining neural networks with both multiple imputation and cross validation, we choose regularization parameters separately for each imputation model. Similarly, when combining neural networks and reduced models we set cross validation parameters separately for each reduced model learned. As in the logistic regression case, we use all data cases that contain the required observed dimensions when learning each reduced model.

Combining neural networks with the response indicator framework is also similar to the logistic regression case. Augmenting the input representation with the response indicator vector only affects the first hidden layer as seen in Equation 6.6.12.

$$A^1_{kn} = \frac{1}{1 + \exp(-(\sum_{d=1}^{D} r_d(w_{dk2}x_{dn} + v_{dn}) + b^1_k))} \tag{6.6.12}$$

We apply several random restarts when learning multi-layer neural networks to help mitigate the problem of local optima. We select the network achieving the lowest training loss from among the candidate networks. When combining neural networks with multiple imputation



(a) Training Data - XOR (b) NN Zero Imputation (c) NN Mean Imputation

(d) NN mixFA Imputation (e) NN Reduced (f) NN Augmented

Figure 6.7: This figure shows results for the XOR data set based on several missing data methods combined with a multi-layer neural network with one hidden layer containing 6 hidden units. Missing data methods include zero imputation, mean imputation, multiple imputation, reduced models, and response indicator augmentation. The results show that the non-linearity of the neural network is able to overcome poor embeddings resulting from zero and mean imputation to a certain degree. Multiple imputation performs the best on this data set, followed closely by the reduced models approach and the response augmented network. It is clear from inspection that the response indicator augmented network learns a more flexible embedding of the incomplete data cases than response augmented logistic regression.

or reduced models, as well as cross validation, this means learning several networks for each combination of imputation model or observed data sub-space, and each set of cross validation parameters.

6.6.6 Synthetic Data Experiments and Results

A neural network classifier with one hidden layer and six hidden units was learned on the XOR data set in combination with zero imputation, mean imputation, multiple imputation, reduced models, and the response indicator framework. Logistic activation functions were used for the hidden units as well as the output. Weight decay was applied with a regularization parameter set by five fold cross validation over the range 10^{-2} to 10^{-5}. The multiple imputation results are based on a mixture of factor analyzers model with latent dimensionality $Q = 1$, and $K = 6$

mixture components. Each method was run for a maximum of 1000 training iterations, and 5 random restarts were used.

The resulting classification functions are shown in Figure 6.7, and classification performance is summarized in Table 6.4. Zero imputation and mean imputation show two very different classification functions. Note that the overall data mean is approximately zero so the two solutions simply represent two different local optima. The mean imputation result shows that a non-linear neural network classifier with sufficient capacity can achieve reasonable accuracy using poor imputation methods. The performance of multiple imputation, reduced models, and the response indicator framework were found to be quite similar, despite learning very different classification functions. The smoothness of the multiple imputation solution is likely explained by the fact that it results from an ensemble of neural network classifiers. The response indicator augmented neural network has clearly not learned an axis aligned embedding of the missing data, showing that it is significantly more flexible than response augmented logistic regression.

6.7 Real Data Experiments and Results

In this section we report results of experiments performed on several real data sets using the classification methods described in this chapter. Experiments were performed using three medical domain data sets from the UCI machine learning repository including Hepatitis, Thyroid-Sick, and Thyroid-AllHypo. All three data sets contain natural missing data. In the final experiment we consider the problem of classifying MNIST digit images subject to artificial missing data.

6.7.1 Hepatitis Data Set

The UCI Hepatitis data set consists of a binary class label indicating whether the patient lived or died. The data set includes a total of 19 features. Twelve of the features are binary, while the remaining seven features are real valued. There are a total of 155 cases in the data set. 32 cases correspond to an outcome where the patient died, and 123 correspond to an outcome where the patient lived. Approximately 6% of the feature values are missing. There are 21 different patterns of missing data. The data set does not have a fixed test set. Due to the small number of cases, a ten fold cross validation error assessment was performed. Logistic regression was combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response augmentation framework. L2 regularization was applied in all cases with the regularization parameter set by a ten fold cross validation search over the range 2^0 to 2^{-10}. The multiple imputation procedure used 5 imputations from a factor analysis mixture with $K = 10$ mixture components and $Q = 5$ latent dimensions. The discriminatively trained LDA-FA model was also applied. A cross validation search was performed over both the number

	Hepatitis	
	Loss	Err(%)
LR Zero	0.4012 ± 0.0439	20.67 ± 2.71
LR Mean	0.4064 ± 0.0576	18.00 ± 2.82
LR MixFA	0.3517 ± 0.0506	13.33 ± 3.44
LR Reduced	0.4443 ± 0.0720	19.33 ± 3.78
LR Augmented	0.5812 ± 0.1258	19.33 ± 4.27
LDA-FA Dis	0.4312 ± 0.0720	20.00 ± 3.98

Table 6.5: Results on the UCI Hepatitis Data set.

of latent dimensions, and the value of the regularization parameter. Latent dimensions in the range 1 to 15 were tried, as were regularization parameters in the range 2^0 to 2^{-10}.

The results of this experiment are presented in Table 6.5. For the most part, the methods have approximately equal performance in terms of both the log loss, and classification error. The one interesting result in this data set is the combination of multiple imputation and logistic regression, which obtains a much lower value of the log loss as well as the classification error rate. This may seem surprising since there are a significant number of binary features in the Hepatitis data set, and the imputation model assumes features are continuous. However, most of the missing attributes in the data set are real valued.

6.7.2 Thyroid - AllHypo Data Set

The UCI Thyroid-AllHypo data set consists of four classes: Primary Hypothyroid, Compensated Hypothyroid, Secondary Hypothyroid and Negative. We perform the standard prediction task, which consists of discriminating between the negative class and the remaining classes. We used five continuous feature variables: FTI, T4U, TT4, T3, and TSH. There are a total of 2659 training cases and a total of 934 test cases. The test data set contains 863 negative cases and 71 positive cases. Approximately 7% of feature values are missing. There are 15 different patterns of missing data.

Logistic regression was combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response augmentation framework. L2 regularization was applied in all cases with the regularization parameter set by five fold cross validation search over the range 2^0 to 2^{-10}. The multiple imputation procedure used 5 imputations from a factor analysis mixture with $K = 4$ mixture components and $Q = 3$ latent dimensions. A sigmoid neural network with one hidden layer was combined with mean imputation, multiple imputation, reduced models, and the response augmentation framework. L2 regularization was applied in all cases with the regularization parameter set by five fold cross validation search over the range 2^0 to 2^{-10}. The hidden layer was fixed to 3 hidden units. The discriminatively trained LDA-FA

	Thyroid: AllHypo	
	Loss	Err(%)
LR Zero	0.1284 ± 0.0002	3.62 ± 0.02
LR Mean	0.1274 ± 0.0001	3.43 ± 0.00
LR MixFA	0.1273 ± 0.0020	3.88 ± 0.15
LR Reduced	0.1281 ± 0.0008	3.53 ± 0.06
LR Augmented	0.1246 ± 0.0003	3.49 ± 0.03
NN Mean	0.0630 ± 0.0007	2.51 ± 0.08
NN MixFA	0.0673 ± 0.0002	2.72 ± 0.03
NN Reduced	0.0650 ± 0.0004	2.55 ± 0.07
NN Augmented	0.0612 ± 0.0003	2.57 ± 0.10
LDA-FA Dis	0.1246 ± 0.0003	3.55 ± 0.02

Table 6.6: Results on the UCI Thyroid AllHypo Data set

model was also applied. A cross validation search was performed over both the number of latent dimensions, and the value of the regularization parameter. Latent dimensions in the range 0 to 5 were tested along with regularization parameters in the range 2^0 to 2^{-10}. A maximum of 1000 learning iterations was used for each method. The complete experiment was repeated 5 times using different cross validation splits during training.

The results of this experiment are presented in Table 6.6. The logistic regression-based models all achieve approximately the same log loss and classification error rates on this data set. The discriminatively trained LDA-FA model also performs about the same as the logistic regression models. Interestingly, the neural network classifiers show a significant improvement in both classification error and log loss compared to the linear classification methods. The performance of all four neural-network strategies was similar, although the response augmented neural network is significantly better than the next best method in terms of log loss.

6.7.3 Thyroid - Sick Data Set

The UCI Thyroid-Sick data set consists of two classes: Sick and Negative. The feature set is the same as for the Thyroid-AllHypo data set. We used five continuous feature variables: FTI, T4U, TT4, T3, and TSH. There are a total of 2659 training cases and a total of 934 test cases. The test data set contains 874 Negative cases and 60 Sick cases. Approximately 7% of feature values are missing. There are 15 different patterns of missing data.

Logistic regression was combined with zero imputation, mean imputation, multiple imputation, reduced models, and the response augmentation framework. L2 regularization was applied in all cases with the regularization parameter set by five fold cross validation search over the range 2^0 to 2^{-10}. The multiple imputation procedure used 5 imputations from a factor analysis mixture with $K = 4$ mixture components and $Q = 3$ latent dimensions. A sigmoid neural

	Thyroid: Sick	
	Loss	Err(%)
LR Zero	0.2123 ± 0.0005	6.75 ± 0.00
LR Mean	0.1112 ± 0.0000	5.25 ± 0.00
LR MixFA	0.1270 ± 0.0009	6.21 ± 0.11
LR Reduced	0.1263 ± 0.0000	5.35 ± 0.00
LR Augmented	0.1166 ± 0.0024	5.35 ± 0.06
NN Mean	0.1892 ± 0.0036	6.42 ± 0.00
NN MixFA	0.1118 ± 0.0012	5.03 ± 0.15
NN Reduced	0.1069 ± 0.0022	3.81 ± 0.09
NN Augmented	0.1065 ± 0.0025	4.95 ± 0.19
LDA-FA Dis	0.1092 ± 0.0011	5.16 ± 0.02

Table 6.7: Results on the UCI Thyroid Sick Data set.

network with one hidden layer was combined with mean imputation, multiple imputation, reduced models, and the response augmentation framework. L2 regularization was applied in all cases with the regularization parameter set by five fold cross validation search over the range 2^0 to 2^{-10}. The hidden layer was fixed to 3 hidden units. The discriminatively trained LDA-FA model was also applied. A cross validation search was performed over both the number of latent dimensions, and the value of the regularization parameter. Latent dimensions in the range 0 to 5 were tested along with regularization parameters in the range 2^0 to 2^{-10}. A maximum of 1000 learning iterations was used for each method. The complete experiment was repeated 5 times using different cross validation splits during training.

The results of this experiment are presented in Table 6.7. There is considerably more variation between the linear logistic methods on the Sick data set compared to the AllHypo data set. Logistic regression with zero imputation performs significantly worse than the other methods in terms of both log loss and classification error. The reduced neural network approach obtains a significantly better test error than any of the other methods on this data set, followed by the response indicator augmented neural network. However, the log loss values are quite similar for the top performing methods.

6.7.4 MNIST Data Set

The MNIST data set consists of 28×28 size images of hand written digits [47]. The standard MNIST classification task is to predict the digit represented by each image. The classification problem has ten classes corresponding to the digits 0 to 9. We consider the full ten class problem. We form a training set consisting of 100 randomly selected training images from each digit class, and a test set containing 500 randomly selected test images from each digit class. We apply a synthetic missing data process to the digit images that randomly selects and

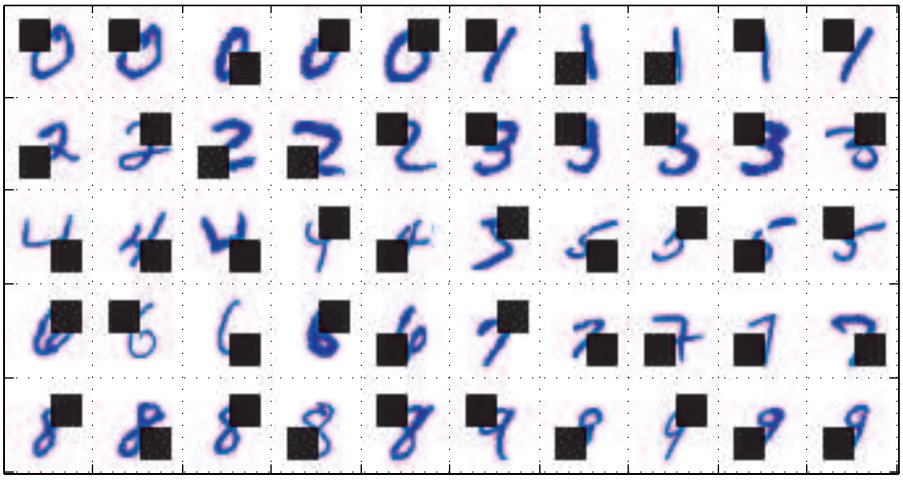


Figure 6.8: This figure shows the result of adding random missing data to MNIST digit images. One quadrant of each image was selected at random and removed. The missing regions are shown in black.

removes one quadrant of each image. This results in only four patterns of missing data so that the reduced models framework may be evaluated. Example digit images are shown in Figure 6.8. Chechik et al. consider a similar task based on the MNIST data set, but restricted to the two class problem of discriminating $5's$ and $6's$ [14].

We combine logistic regression with zero imputation, mean imputation, reduced models, and the response indicator framework. We consider learning the discriminatively trained LDA-FA model with latent dimensionality 0, 10, 20, and 50. We combine a one hidden layer neural network with mean imputation, reduced models, and the response indicator framework. The hidden layer of the neural network was fixed to 20 hidden units. Finally, we combine Gaussian kernel logistic regression with mean imputation, reduced models, and the response indicator framework. We use five fold cross validation to set the regularization parameters of all of the models. We also use five fold cross validation to set the Gaussian kernel parameters for kernel logistic regression. We search over the range $\sigma^2 = 25, 50, 100, 200$ for all Gaussian kernels. For the response augmented gaussian kernel we set $\gamma_d = \gamma s_d^2$ where s_d^2 is the empirical variance on dimension d, and $\gamma = 1, 0.01, 0.0001$ was set by cross validation. We remove negative eigenvalues from the response augmented kernel matrix to ensure that it is positive definite. All methods were trained for up to 1000 iterations.

The results of this experiment are given in Table 6.8. It is interesting to note that the worst performance is obtained using the reduced models framework. Note that there are only four patterns of missing data in the data set, but there are also only 100 training cases per class. This means that on average each reduced model is trained on less than 25 data cases per class. It is clear from the results that pooling data cases with different response patterns leads to

	MNIST Digits	
	Loss	Err(%)
LR Zero	0.6350 ± 0.0110	19.75 ± 0.41
LR Mean	0.6150 ± 0.0112	19.15 ± 0.34
LR Reduced	0.7182 ± 0.0135	22.62 ± 0.45
LR Augmented	0.6160 ± 0.0112	19.35 ± 0.36
LDA-FA Dis	0.6355 ± 0.0051	19.95 ± 0.25
NN Mean	0.6235 ± 0.0541	18.34 ± 0.42
NN Reduced	0.6944 ± 0.0088	21.51 ± 0.27
NN Augmented	0.5925 ± 0.0161	17.76 ± 0.18
gKLR Mean	0.4147 ± 0.0075	13.02 ± 0.24
gKLR Reduced	0.5694 ± 0.0079	18.32 ± 0.49
gKLR Augmented	0.3896 ± 0.0101	12.34 ± 0.46

Table 6.8: Results on the MNIST data set with synthetic missing data.

better classification performance.

Linear classification methods are known to give poor performance on the MNIST data set, so it is not surprising that the LDA-FA model and the logistic regression based solutions yield relatively low classification accuracy. The best results are obtained using the response augmented neural network, and response augmented Gaussian kernel logistic regression method. It is interesting to note that the performance of the response augmented Gaussian kernel logistic regression method is significantly better than the reduced Gaussian kernel logistic regression method. This indicates that the response augmented kernel selected by the cross validation procedure is allowing for useful smoothing between data cases with different response patterns.

Chapter 7

Conclusions

Learning, inference, and prediction in the presence of missing data are pervasive problems in machine learning and statistical data analysis. This thesis focuses on the problems of collaborative prediction with non-random missing data and classification with missing features. We have presented new experimental protocols, new data sets, and new models and algorithms for learning, inference, and prediction. In this final chapter we present concluding remarks and indicate directions for future research.

7.1 Unsupervised Learning with Non-Random Missing Data

In the first half of this thesis, we describe a framework for collaborative prediction in the presence of non-random missing data. The framework is based on combining probabilistic models for complete data with probabilistic models of the non-random missing data process. These missing data models capture simple non-random effects by allowing the response probability for each data dimension to depend on the underlying feature values. Models and prediction methods were tested both on the novel Yahoo! music data set with natural missing data and the Jester data set with synthetic missing data.

Our results show that incorporating a model of the missing data process results in substantial improvements in predictive performance on randomly selected items compared to models that ignore the missing data process. Our results show that training and testing only on ratings for user selected items can vastly overestimate prediction performance on randomly selected items. Our analysis also shows that the availability of even a small sample of ratings for randomly selected items can have a large impact on rating prediction performance. Rating prediction performance is also important if recommendations are derived from predicted ratings. Our initial results evaluating rankings produced by rating prediction methods also show that modeling the missing data mechanism results in improved ranking performance.

There are many interesting future directions for modeling and learning with non-random missing data within the collaborative filtering domain. The development of more flexible missing data models for the collaborative filtering domain remains an important and challenging problem. The key assumption of the CPT-v model is that the choice to rate an item depends only on the underlying rating value. The Logit-vd model loosens that assumption by allowing the choice to also depend on the identity of the item. Of course, one might speculate that there are any number of other factors involved in the choice to rate an item. The identity of the user is likely to be important for several reasons. First, some users may be inherently more interested in rating items than other users. Second, different users may differ in their probability of response for a given underlying preference level. This necessitates the introduction of different missing data probabilities for different users, or classes of users.

The assumption that the choice to rate each item is independent of all of the other items given the underlying rating value and item identity may also be questionable. It could be interesting to consider removing this assumption and allowing response probabilities to directly depend on higher level user tastes or preferences, as captured by latent variables. A related issue is that current models strictly assume that the user chooses not to rate all of the unrated items. This assumption is certainly flawed in the case of new users where we know that the number of ratings the user has entered is limited by the amount of time the user has spent using the recommender system. The assumption also may not hold for users who are not aware that certain items are available in the recommender system.

A further implicit assumption is that all users in the recommender system use the rating scale in the same way. This is not necessarily the case, and has the potential to introduce an additional ambiguity when interpreting rating data. How might one go about deciding whether a particular user selects items to rate completely at random and then only uses the extreme one star and five star rating values, or whether the same user only rates items he thinks are one star or five star items? In both cases all of the observations will be either one star or five star ratings, but the two situation have very different interpretations.

Relaxing the assumptions behind the CPT-v and Logit-vd models in one or more of the ways we have described here has the potential to result in more flexible selection models. However, a crucial problem is the development of suitable learning, inference, and prediction methods. Likely candidates for dealing with more flexible selection models are hierarchical and semi-parametric Bayesian modeling and inference. However, the need for practical methods to operate in an online setting raises significant questions about computational efficiency of prediction procedures.

A complimentary approach to dealing with more complex selection models is to collect additional information that will facilitate learning and inference. An approach to dealing with

the problem of new users and users who may not be aware of certain items is to explicitly record all of the items that each user views, listens to, reads, or purchases, regardless of whether ratings are supplied for the items. The fact that a user listened to a particular song and chose not to rate it is significantly more informative than simply knowing that a user did not rate a song.

An interesting alternative to the data model/selection framework is the conditional restricted Boltzmann machine framework. The key to the RBM framework is the availability of training ratings for items the user did not choose to rate. In this work, a random sample of ratings was used to provide these ratings. It is doubtless that many users would object to rating random items. However, it is certainly the case that some users are willing to provide this type of data as evidenced by the more than $35,000$ users who participated in the Yahoo! study.

The main avenues for further research with the conditional RBM model revolve around the definition of the energy function. Like the CPT-v selection model, the cRBM/E-v model specifies a fairly strict relationship between the ratings of user-selected and non user-selected items when making predictions. Any number of additional terms may be included and their effect on prediction accuracy assessed. An interesting line of inquiry would be to study the effect of the amount of ratings for non-user selected items on the ability to learn conditional RBM models with differently parameterized energy functions. We have also previously mentioned the possibility of re-weighting the contrastive divergence gradients to help balance different numbers of user-selected and non-user selected items.

7.2 Classification with Missing Features

In the second half of this thesis, we consider the problem of classification with missing features. We consider several strategies for performing incomplete data classification including the use of generative classification models, and the combination of standard discriminative methods with imputation, reduced models, and response indicator augmentation. We introduce a generative classifier for incomplete data based on linear discriminant analysis combined with a factor analysis covariance model. We compare imputation, reduced models, and response indicator augmentation combined with logistic regression, multi-layer neural networks, and kernel logistic regression. We show that response augmentation has interesting properties when combined with non-linear classifiers.

Synthetic data results illustrate the fact that zero imputation and unconditional mean imputation rarely work well with linear classifiers. With sufficient capacity, non-linear classifiers can sometimes overcome poor imputation procedures. Multiple imputation was shown to work well so long as the conditional distribution of missing data given observed data is approximately

correct. The reduced models approach was also shown to work well when there is sufficient data to reliably estimate a model for each response pattern. The synthetic data results illustrate the important fact that if classes significantly overlap in a given observed data sub-space, no classification method can perform well including multiple imputation and reduced models.

Results on real medical domain data show that many of the methods yield classification accuracy that is not significantly better than baseline methods like zero imputation and mean imputation. This may be partly due to the fact that the medical domain data sets do not contain a large amount of missing data, and partly due to the fact that the medical domain data sets are strongly imbalanced. Looking at the bigger picture; however, the best method on each data set performs quite well in absolute terms.

The most interesting classification results were seen in the MNIST experiment. The MNIST experiment used a relatively small number of high dimensional data cases with a significant amount of missing data in each data case. The novel response augmented Gaussian kernel logistic regression method obtained the best results in the MNIST experiment, followed by Gaussian kernel logistic regression combined with mean imputation. In general, the classification results show that learning a detailed model of the feature space can often be avoided and good classification performance can still be obtained.

In terms of future research directions, it may be useful to consider other approaches for dealing with the indefinite augmented Gaussian kernel. The present strategy of removing negative eigenvalues from the kernel matrix is a fairly costly operation. Another option is to consider other Gaussian-like kernels that are positive definite. Bhattacharya kernels, a special case of probability product kernels [42], can be constructed to resemble the augmented Gaussian kernel using relatively simple Gaussian or class conditional Gaussian models of the feature space, and are always positive definite. A completely different classification framework that we have not explored here, but might be amenable to dealing with missing features is boosting [22]. It would be quite interesting to consider an application of boosting using response augmented decision stumps as the weak learners.

Semi-supervised learning deals with the problem of learning classifiers from both labeled and unlabeled data [12]. The significance of semi-supervised learning stems from the fact that in some domains there is an abundance of unlabeled data while labeled data is quite scarce. The semi-supervised learning framework can be extended by simultaneously allowing missing data in the features as well as the labels. Semi-supervised learning can also be made more difficult if missing data in the labels or features is not missing at random. Semi-supervised learning along with these variations provides yet another source of very interesting missing data problems in machine learning.

Bibliography

[1] Alan Agresti. *Categorical Data Analysis (Second Edition).* John Wiley and Sons, 2002.

[2] D. Aldous. Exchangeability and Related Topics. In *Proceedings of the Ecole d'Ete de Probabilities de Saint-Flour XIII*, Pages 1–198. Springer, 1985.

[3] J. A. Anderson. Separate Sample Logistic Discrimination. *Biometrika*, 59(1):19–35, 1972.

[4] Galen Andrew and Jianfeng Gao. Scalable Training of L1-Regularized Log-Linear Models. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[5] A. Banerjee. An Analysis of Logistic Models: Exponential Family Connections and Online Performance. In *SIAM International Conference on Data Mining*, 2002.

[6] Halima Bensmail and Gilles Celeux. Regularized Gaussian Discriminant Analysis Through Eigenvalue Decomposition. *Journal of the American Statistical Association*, 91(463):1743–1748, 1996.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[8] Stephen Boyd. An Interior-Point Method for Large-Scale L1-Regularized Logistic Regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

[9] John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, Pages 43–52, July 1998.

[10] Christopher J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[11] John Canny. Collaborative Filtering with Privacy via Factor Analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pages 238–245. ACM Press, 2002.

[12] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[13] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel, and Daphne Koller. Max-Margin Classification of Incomplete Data. In *Advances in Neural Information Processing Systems 19*, 2006.

[14] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel, and Daphne Koller. Max-Margin Classification of Data with Absent Features. *Journal of Machine Learning Research*, 9:1–27, 2007.

[15] D. R. Cox and E. J. Snell. *Analysis of Binary Data*. Chapman Hall, second edition, 1989.

[16] Dennis Decoste. Collaborative Prediction Using Ensembles of Maximum Margin Matrix Factorizations. In *Proceedings of the 23rd International Conference on Machine Learning*, Pages 249–256, 2006.

[17] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum Likelihood From Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[18] Thomas S. Ferguson. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[19] R. A. Fisher. On the Mathematical Foundations of Theoretical Statistics. *Philosphical Transactions of the Royal Society of London Series A*, 1922.

[20] R.A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *The Annual Eugenics*, 7:179–188, 1936.

[21] Y. Freund and R. E. Schapire. Large Margin Classification Using the Perceptron Algorithm. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (Colt' 98)*, 1998.

[22] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[23] Jerome H. Friedman. Regularized Discriminant Analysis. *Journal of the American Statistical Association*, 84:165–175, 1989.

[24] Alan E. Gelfand and Adrian F. M. Smith. Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

[25] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis, Second Edition*. Chapman & Hall/CRC, 2003.

[26] Zoubin Ghahramani and Geoffrey E. Hinton. The EM Algorithm for Mixtures of Factor Analyzers. Technical Report CRG-TR-96-1, University of Toronto, 1996.

[27] Zoubin Ghahramani and Michael I. Jordan. Learning From Incomplete Data. Technical Report CBCL-108, Center for Biological and Computational Learning. Massachusetts Institute of Technology, 1994.

[28] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an EM approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, Volume 6, Pages 120–127, 1994.

[29] G.F.V. Glonek. On Identifiability in Models for Incomplete Binary Data. *Statistics and Probability Letters*, 41:191–197, 1999.

[30] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval Journal*, 4(2):133–151, July 2001.

[31] B. Haasdonk and C. Bahlmann. Learning with Distance Substitution Kernels. *Pattern Recognition - Proceedings of the 26th DAGM Symposium*, 2004.

[32] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[33] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pages 230–237, 1999.

[34] G. E. Hinton and T. J. Sejnowski. *Parallel Distributed Processing*, Volume 1: Foundations, Chapter: Learning and Relearning in Boltzmann Machines, Pages 194–281. MIT Press, 1986.

[35] Geoffrey E. Hinton. *Parle Parallel Architectures and Languages Europe*, Volume 258, Pages 1–13. Springer Berlin, 1987.

[36] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[37] Geoffrey E Hinton, Michael Revow, and Peter Dayan. Recognizing Handwritten Digits Using Mixtures of Linear Models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, Volume 7, Pages 1015–1022, 1995.

[38] Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970.

[39] Thomas Hofmann and Jan Puzicha. Latent Class Models for Collaborative Filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence*, 1999.

[40] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression, Second Edition*. John Wiley and Sons, 2000.

[41] Hemant Ishwaran and Mahmoud Zarepour. Exact and Approximate Sum Representations for the Dirichlet Process. *The Canadian Journal of Statistics*, 30(2):269–283, 2002.

[42] Tony Jebara, Risi Kondor, and Andrew Howard. Probability Product Kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[43] Ian T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2002.

[44] M. Jordan. Why the Logistic Function? A Tutorial Discussion on Probabilities and Neural Networks. Technical Report 9503, MIT Computational Cognitive Science, 1995.

[45] Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors. *Using Mixtures of Factor Analyzers for Segmentation and Pose Estimation*. The MIT Press, 1998.

[46] S. S. Keerthi, K. B. Duan, S. K. Shevade, and A. N. Poo. A Fast Dual Algorithm for Kernel Logistic Regression. *Machine Learning*, 61(1-3):151–165, 2005.

[47] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[48] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient L1 Regularized Logistic Regression. In *Proceedings of the Twenty-First Conference on Artificial Intelligence*, 2006.

[49] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. John Wiley and Sons, Inc., 1987.

[50] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data*. Wiley, second edition, 2002.

[51] David J. C. Mackay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[52] Benjamin Marlin. Collaborative Filtering: A Machine Learning Perspective. Master's thesis, University of Toronto, January 2004.

[53] Benjamin Marlin, Richard S. Zemel, Sam T. Roweis, and Malcom Slaney. Collaborative Filtering and the Missing at Random Assumption. In *Uncertainty in Artificial Intelligence: Proceedings of the 23rd Conference (Submitted)*, 2007.

[54] Benjamin M. Marlin, Sam T. Roweis, and Richard S. Zemel. Unsupervised Learning with Non-Ignorable Missing Data. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, 2005.

[55] Geoffrey J. Mclachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley and Sons, 1992.

[56] M. Minsky and S. Papert. *Perceptron*. MIT Press, 1969.

[57] Tom M. Mitchell. *Machine Learning*. Mcgraw-Hill, New York, 1997.

[58] A. Mkhadri, G. Celeux, and A. Nasroallah. Regularization in Discriminant Analysis: An Overview. *Computational Statistics and Data Analysis*, 23(3):403–423, 1997.

[59] Radford M. Neal. Bayesian Mixture Modeling By Monte Carlo Simulation. Technical Report CRG-TR-91-2, University of Toronto. Department of Computer Science, 1991.

[60] Radford M. Neal. Markov Chain Sampling Methods for Dirichlet Process Mixture Models. Technical Report 9815, University of Toronto. Department of Statistics, 1998.

[61] Andrew Y. Ng. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*. ACM Press, 2004.

[62] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.

[63] Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 58(6):386–408, 1958.

[64] Donald B. Rubin. Inference and Missing Data. *Biometrika*, 64(3):581–592, 1976.

[65] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning*, Pages 249–256, 2007.

[66] L. Saul and M. Rahim. Maximum Likelihood and Minimum Classification Error Factor Analysis for Automatic Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 8(2):115–125, 1999.

[67] J. L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman and Hall, 1997.

[68] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[69] P.K. Sharpe and R.J. Solly. Dealing with Missing Values in Neural Network-Based Diagnostic Systems. *Neural Computing and Applications*, 3:73–77, 1995.

[70] P. Smolensky. *Parallel Distributed Processing*, Volume 1: Foundations, Chapter: Information Processing in Dynamical Systems: Foundations of Harmony Theory, Pages 194–281. MIT Press, 1986.

[71] M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):111–147, 1974.

[72] Robert Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[73] M. E. Tipping and C. M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society*, 61(3):611–622, 1999.

[74] Michael E. Tipping and Christopher M. Bishop. Mixtures of Probabilistic Principal Component Analysers. *Neural Computation*, 11(2):443–482, 1999.

[75] Volker Tresp, Subutai Ahmad, and Ralph Neuneier. Training Neural Networks with Deficient Data. In *Advances in Neural Information Processing Systems 6*, 1994.

[76] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[77] Markus Weimer, Alexandros Karatzoglou, Quoc Le, and Alex Smola. COFI RANK - Maximum Margin Matrix Factorization for Collaborative Ranking. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Pages 1593–1600, Cambridge, MA, 2008. MIT Press.

[78] B. L. Welch. Note on Discriminant Functions. *Biometrika*, 31(1):218–220, 1939.

[79] D. Williams, X. Liao, Y. Xue, and L. Carin. Incomplete-Data Classification Using Logistic Regression. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.

[80] Ji Zhu and Trevor Hastie. Kernel Logistic Regression and the Import Vector Machine. *Journal of Computional and Graphical Statistics*, 14(1):185–205, 2005.