A Probabilistic Framework for Multi-Task Learning

Jian Zhang

August 16, 2006

CMU-LTI-06-006

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Yiming Yang, Chair
Jaime Carbonell
Zoubin Ghahramani
Larry Wasserman
Tong Zhang, Yahoo! Research

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language Technologies

A Probabilistic Framework for Multi-Task Learning

jian.zhang@cs.cmu.edu

Aug 16, 2006

Abstract

An important problem in statistical machine learning is how to effectively model the predictions of multiple related tasks, which is known as *multi-task learning*. Different from *single-task learning* where tasks are learned separately, multi-task learning aims to jointly model those tasks. The main benefit of multi-task learning is that it can more effectively use training resources from all tasks and achieve better generalization performance when tasks are related. To be more specific, successfully addressing multi-task learning can not only allay the data paucity problem given many tasks, but also generalize to future tasks by transferring knowledge learned from existing tasks. Multiple tasks naturally exist in many applications, such as text classification, email anti-spam filtering, image classification, etc.

We present a novel probabilistic framework for multi-task learning where task relatedness is modeled using a shared structure through latent variables. Within such a framework, we study a series of important multi-task learning scenarios and propose suitable models accordingly, and show that the flexibility of the framework is achieved by allowing different assumptions about latent variables and the shared structure. In particular, we present sparsity models which are parsimonious and suitable for high-dimensional tasks; we propose the $l_1 \circ l_p$ regularization method which is suitable for joint feature selection; we propose to use mixture models as the solution of the clusters of tasks scenario; we also extend our framework to unsupervised learning and show its connection to existing topic models. Furthermore, model selection techniques for multi-task learning are investigated since they play important roles in choosing the best joint model and generalizing to future tasks. Experiments are conducted to support our methods using both simulated datasets and real datasets from text classification, anti-spam filtering, handwritten letter recognition and collaborative filtering.

Acknowledgments

My first thanks go to my committee memebers, not just for their insightful comments on my thesis. Yiming Yang brought me into the research field of information retrieval and machine learning and gave me the freedom to choose this thesis topic. I worked with Jaime Carbonell right after I came to CMU and learned a lot from his analytical skills and insightful ways of looking at problems. Zoubin Ghahramani is a great teacher and collaborator. I have enjoyed a lot from his Bayesian way of thinking problems and wish I would work with him for more than just a couple of days every year. My non-Bayesian statistical thinking is largely influenced by Larry Wasserman and Tong Zhang. I have benefited a lot from both talking to Larry and taking his wonderful classes. Tong answered my every emails and his words greatly improved my understanding of machine learning.

I spent five years (2001-2006) at Carnegie Mellon University. During that period I received tremendous help from faculties, staffs, students and friends, who made my life much easier and enjoyable: Tom Ault, Paul Bennett, Jamie Callan, Yi Chang, Kevyn Collins-Thompson, Lie Gu, Benjamin Han, Chun Jin, Rong Jin, Brian Junker, Bryan Kisiel, Leonid Kontorovich, Abhimanyu Lad, Guy Lebanon, Fan Li, Lucian Vlad Lita, Han Liu, Yan Liu, Jie Lu, Paul Oglivie, Jiazhi Ou, Katharina Probst, Yanjun Qi, Rahda Rao, Kathryn Roeder, Monica Rogati, Mark Shervish, Luo Si, Adele Weitz, Eric Xing, Rong Yan, Jun Yang, Shinjae Yoo, Stacey Young, Rong Zhang, Yi Zhang, Ying Zhang, Jerry Zhu.

Finally, I would like to thank my parents for their early education as well as support during my time at Carnegie Mellon. Last but not least I want to thank my wife, for being patient with me many years. Without her love and support I would not be where I am now, not even close.

Notation

Symbol	Meaning
italic letters: $x, y, z; f, g, h$	x, y, z : scalars; f, g, h : functions
bold letters: $\mathbf{x}, \mathbf{y}, \mathbf{z}$	vectors (column vectors by default)
capital **BOLD** letters: $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	matrices
Greeks: $\mu, \theta, \alpha, \beta, \Theta, \Lambda$	model parameters
$\langle \mathbf{X}, \mathbf{Y} \rangle$ and $\langle \mathbf{x}, \mathbf{y} \rangle$	$\mathbf{X}^T\mathbf{X}$ and $\mathbf{x}^T\mathbf{y}$
K	number of tasks
N	number of data instances
F	number of features/predictors
H	number of hidden sources (latent variables)
$\mathsf{Bernoulli}(\mu)$	Bernoulli distribution with mean μ
$\mathsf{Multinomial}(n; \theta_1, \ldots, \theta_F)$	Multinomial distribution with parameter n and proportional parameters $\theta_1, \ldots, \theta_F$
$\mathsf{Normal}(\mathbf{m}, \mathbf{V})$	Gaussian distribution with mean $\mathbf{m}$ and covariance $\mathbf{V}$
$\mathsf{Laplace}(m, v)$	Laplace distribution with mean m and variance v
$\mathsf{InvGamma}(\nu_\alpha, \nu_\beta)$	Inverse Gamma distribution
$\mathsf{DP}(\nu, G_0)$	Dirichlet process with precision ν and base distribution G_0
$\mathsf{GP}(f, K)$	Gaussian processes with mean function $f(.)$ and covariance function $K(.,.)$
$\mathbb{E}[\mathbf{x}]$	Expectation of random variable $\mathbf{x}$
$\mathbb{V}[\mathbf{x}]$	Variance-covariance matrix of random variable $\mathbf{x}$
$\mathbb{C}[X, Y]$	Covariance between random variables X and Y

Contents

Chapter 1

Introduction

1.1 Why Multi-Task Learning?

An important problem in statistical machine learning is how to generalize among multiple related prediction tasks. This problem has been called "Multi-Task Learning" [Caruana, 1997], "Learning to Learn" and "Transfer Learning" [Thrun and Pratt, 1998], and sometimes "Predictions of Multivariate Responses" [Breiman and Friedman, 1997] in the machine learning and statistics literature. Multi-task learning has many potential applications, and in the following we give several important examples which can be recast as multi-task learning problems:

- *Multi-label Text Classification:* Text classification is one fundamental problem in information retrieval, whose objective is to automatically classify documents into pre-defined categories. Multi-label text classification refers to the situation where a document is assigned to a subset of K possible categories, and many of the existing text collections are multi-labeled by nature. Most studies in text classification decompose this problem into K binary classification problems and solve them independently. However, since it is often the case that categories are related to each other (in terms of both semantics and statistical correlations), it would be beneficial to treat this problem as a multi-task learning problem. Furthermore, the existence of multiple taxonomies also leads to multi-task learning problems where each task is a binary classification problem with respect to some category in one of the taxonomies.

- *Anti-spam Filtering:* Email anti-spam filtering has been an important research topic as people get more and more disturbing spams in their daily emails. Typically this problem is treated as a binary classification problem [Brutlag and Meek, 2000, Zhang, 2002] to distinguish spams from non-spams. In a more realistic situation, the system will serve many users for anti-spam filtering. This provides a good opportunity for multi-task learning, where we could treat the anti-spam filtering for a particular user as one task and borrow information among users. Viewing in this way has the advantage that both user-specific and user-independent preferences are effectively captured in the model.

- *Multi-user Prediction Problems:* Essentially many prediction problems involved with multiple users can be treated as multi-task learning problems, such as adaptive filtering [Roberson and Hull, 2001, Zhang, 2004, Yang et al., 2005] w.r.t. multiple users, collaborative filtering [Breese et al., 1998] with auxiliary information about movies, etc. Similar to the case of anti-spam filtering, prediction functions for each user are often closely related to each other and thus joint inference can capture such dependencies and work more effectively with all training resources.

- *Predicting Many Stocks:* Consider the problem where we would like to predict the future stock prices of several companies in one industry or several related industries. Often predictions of individual company's stock price are made using models trained with each company's previous stock data. However, due to their possible competitive or cooperative relations and cross-industry effects, those prediction tasks could be very related. Consequently, this problem can be more effectively solved as a multi-task learning problem.

Multi-task learning simply generalizes single-task learning to a higher level and as a result, it is able to capture the dependencies among tasks. Compared to single-task learning, multi-task learning has the following benefits: (1) It can provide better generalization performance especially when the amount of training data is limited; (2) It can provide meta-level knowledge (which is not available in single-task learning) which is useful to generalize to future tasks; (3) It can provide a joint, succinct representation of all task structures. Multi-task learning is particularly applicable in the following situations:

- In many existing datasets instances are naturally associated with multiple responses (e.g., multi-labeled document collections) and thus it is

beneficial to use available resource and borrow information from other related tasks.

- From the data annotation viewpoint it is more convenient to get responses of multiple related tasks simultaneously if possible (e.g., assigning documents or web pages to multiple categories after reading) as opposed to obtain them in separate steps.

- There are situations (one such example is anti-spam filtering) where for some of the tasks it is more difficult to get training resource than others, and multi-task learning can be especially beneficial for those tasks with limited training data.

Many approaches have been proposed in the machine learning literature on how to effectively learn multiple tasks, such as [Baxter, 1996, 1997, Breiman and Friedman, 1997, Caruana, 1997, Minka and Picard, 1997, Baxter, 2000, Heskes, 2000, Ando and Zhang, 2004, Evgeniou et al., 2005, Teh et al., 2005, Yu et al., 2005, Zhang et al., 2005]. Generally speaking, existing approaches share the basic assumption that tasks are related to each other. Based on how task relatedness is handled we summarize existing methods into several categories, such as artificial neural networks, hierarchical Bayesian models, regularization methods, etc. Details can be found in Chapter 2. The reason why multi-task learning works can be seen from several aspects. Given the assumption is that the outcomes in multiple tasks are related, it would be beneficial to borrow information from other tasks as opposed to learning each task independently (e.g., single-task learning). The simplest example is that, if task parameters - which index their corresponding prediction functions - are partly shared, then from a statistical estimation viewpoint we could obtain a more reliable estimation by using all training resources together and better generalization performance can be achieved.

This thesis is aimed at developing models for multi-task learning problems. By presenting a unified probabilistic framework, we gain insights in task relatedness and can systematically explore important multi-task scenarios, which are key components in order to successfully address multi-task learning. The resulting multi-task learning models can provide better generalization performance than conventional single-task learning methods when tasks are related, and efficient algorithms are achievable through (approximate) inference and opitmization techniques.

1.2 Roadmap

In this thesis we present a novel probabilistic framework for multi-task learning. Unlike previous approaches, our framework is flexible and can support models for a series of multi-task learning scenarios, and task dependencies are captured by using a shared structure through latent variables. The flexibility comes from the statistical assumptions on latent variables and structural assumptions on the shared components, and the concept of task relatedness can now be better explained by the underlying statistical assumption. Motivated from this exploration, we develop suitable models for several task scenarios that have not been studied before, as well as investigate model selection techniques. The rest of the thesis is organized as follows:

Chapter 2 first provides a brief introduction to statistical models and algorithms which are the building blocks for the rest of the thesis, and then reviews the literature on multi-task learning.

Chapter 3 presents the unified probabilistic framework for multi-task learning, and analyzes a series of important task scenarios with associated models.

Chapter 4 presents learning and inference algorithms for the generic models supported by the framework.

Chapter 5 presents two types of sparsity models for multi-task learning, and demonstrates their effectiveness on multi-label text classification and email anti-spam filtering.

Chapter 6 presents the $l_1 \circ l_p$ regularization for joint feature selection in multi-task learning. We show that it is a generalization of lasso-style algorithms under the multi-task learning setting, and our results support the theoretical claims.

Chapter 7 proposes to use mixture model for the "cluster of tasks" scenario. We present efficient EM algorithm for the inference, and apply it to the collaborative filtering problem.

Chapter 8 investigates the model selection problem in multi-task learning using cross-validation techniques.

Chapter 9 extends our framework to unsupervised multi-task learning. We show its connection to existing unsupervised topic models, and apply it to the novelty detection problem.

Chapter 10 summarizes the thesis work.

Chapter 2

Background and Literature Review

This thesis relates to a broad set of current statistical techniques. To make sure our terminology is clearly defined, we first give a brief introduction to the related statistical models, which our proposed framework is based upon, including regression, classification and dimensionality reduction techniques. Then we briefly outline some background knowledge on general algorithms for learning and inference. The second part of this chapter presents a literature review for multi-task learning research.

2.1 Background

2.1.1 Modeling using Linear Regression

Linear regression is the simplest but probably the most important model for regression problems. The linear regression model [Wasserman, 2005] assumes that

$$\begin{aligned} y_i &= \boldsymbol{\theta}^T \mathbf{x}_i + e_i \\ e_i &\sim \mathsf{Normal}(0, \sigma^2) \end{aligned} \tag{2.1}$$

or equivalently

$$y_i \sim \mathsf{Normal}(\boldsymbol{\theta}^T \mathbf{x}_i, \sigma^2) \tag{2.2}$$

where $\mathbf{x}_i \in \mathbb{R}^{F\times 1}$ is the i-th input data vector, $\boldsymbol{\theta} \in \mathbb{R}^{F\times 1}$ is the model parameter vector, $i = 1, 2, \ldots, N$ is the data index, and here we assume that variances of random noises e_i's are isotropic. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, the Maximum Likelihood Estimation (MLE) of $\boldsymbol{\theta}$ is equivalent to the least square estimation due to the Gaussian noise assumption:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 \right\} \tag{2.3}$$

which has the analytical solution $\hat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ by using the succinct matrix/vector notation $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T \in \mathbb{R}^{N\times F}$ and $\mathbf{y} = (y_1, \ldots, y_N)^T \in \mathbb{R}^{N\times 1}$.

Note that the above optimization problem in equation (2.3) may not have a unique solution if the matrix $\mathbf{X}^T\mathbf{X}$ is singular. By adding a regularization term $\Omega(\boldsymbol{\theta})$ we are guaranteed to get a more stable solution:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} (y_i - \boldsymbol{\theta}^T \mathbf{x}_i)^2 + \lambda\Omega(\boldsymbol{\theta}) \right\} \tag{2.4}$$

where $\lambda > 0$ is known as the regularization coefficient which trades off between empirical loss and model complexity. Note that when $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2^2$ or $||\boldsymbol{\theta}||_1$ the above model is known as ridge regression or lasso regression respectively. Ridge regression has a L_2 regularization while lasso has a L_1 regularization, and they can both be interpreted as Maximum A Posterior (MAP) estimators of a Bayesian model by assuming a Gaussian or Laplace prior over $\boldsymbol{\theta}$ respectively. Finally, it is well understood that L_1 regularization tends to give a sparse solution where most of the elements of $\boldsymbol{\theta}$ are zero values; while L_2 regularization shrinks all coefficients to zero smoothly [Hastie et al., 2001].

2.1.2 Modeling using Logistic Regression

Classification problems need to be handled differently from regression due to its binary output y as well as the 0/1 or cross entropy loss which is rather different from the squared loss used by default in regression. Logistic regression can be thought as a discriminative classifier (as opposed to generative classifiers like Naive Bayes [McCallum and Nigam, 1998], see also discussions on those two kinds of classifiers in [Rubinstein and Hastie, 1997, Ng and Jordan, 2002]), and it is often preferred to generative classifiers by following Vapnik's

philosophy [Vapnik, 1998, 1999] that "`When solving a given problem one should avoid solving a more general problem as an intermediate step`". Using probabilistic modeling language, the logistic regression model can be formulated as

$$\begin{aligned} y_i &\sim \mathsf{Bernoulli}(\mu(\boldsymbol{\theta}^T\mathbf{x}_i)) \\ \mu(t) &= \int_{-\infty}^{t} p(z)dz \end{aligned} \tag{2.5}$$

where y_i is generated from a Bernoulli distribution with mean $\mu(\boldsymbol{\theta}^T\mathbf{x}_i)$ and its value is either 0 or 1, and $p(z)$ is the *probability density function* (pdf) of the standard logistic distribution $p(z) = \frac{\exp(-z)}{(1+\exp(-z))^2}$. In this case, equation (2.5) can be simplified as

$$\begin{aligned} y_i &\sim \mathsf{Bernoulli}(\mu(\boldsymbol{\theta}^T\mathbf{x}_i)) \\ \mu(t) &= (1+\exp(-t))^{-1} \end{aligned} \tag{2.6}$$

Note also that by plugging in different random variable Z with its pdf $p(z)$ we are able to get several popular probabilistic classifiers. As an example, when $p(z)$ is the pdf of standard Gaussian distribution the above model becomes the so-called *probit regression*. Given a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the MLE solution of logistic regression can be derived using equation (2.6):

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \left\{ \prod_{i=1}^{n} \left(\frac{1}{1+\exp(-\boldsymbol{\theta}^T\mathbf{x}_i)} \right)^{y_i} \left(\frac{\exp(-\boldsymbol{\theta}^T\mathbf{x}_i)}{1+\exp(-\boldsymbol{\theta}^T\mathbf{x}_i)} \right)^{1-y_i} \right\} \\ &= \arg\max_{\boldsymbol{\theta}} \left\{ -\sum_{i=1}^{n} y_i \log\left(1+\exp(-\boldsymbol{\theta}^T\mathbf{x}_i)\right) - \sum_{i=1}^{n}(1-y_i)\log\left(1+\exp(\boldsymbol{\theta}^T\mathbf{x}_i)\right) \right\} \\ &\overset{y_i=0 \text{ or } 1}{=} \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{n} \log\left(1+\exp(-(2y_i-1)\boldsymbol{\theta}^T\mathbf{x}_i)\right) \right\} \end{aligned} \tag{2.7}$$

If we re-define $y_i = -1$ instead of $y_i = 0$ for negative class label, then the MLE solution of logistic regression can be represented as the solution of the following optimization problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} \log(1+\exp(-y_i\boldsymbol{\theta}^T\mathbf{x}_i)) \right\} \tag{2.8}$$

Similar to linear regression we can also formulate L_1 or L_2 regularized logistic regression, and they can both be interpreted as MAP estimators as well.

2.1.3 Modeling using Factor Analysis

Factor Analysis (FA) is a technique originated from social science [Spearman, 1904, Gorsuch, 1983] which is used to discover underlying factors from associated data. The intrinsic assumption in FA is that the hidden factors are distributed as Gaussian

$$\begin{aligned} \mathbf{y} &= \Lambda\mathbf{s} + \boldsymbol{\mu} + \mathbf{e} \\ \mathbf{s} &\sim \mathsf{Normal}(\mathbf{0}, \mathbf{I}) \\ \mathbf{e} &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{2.9}$$

where we have the observed variable $\mathbf{y} \in \mathbb{R}^{F\times 1}$, hidden source $\mathbf{s} \in \mathbb{R}^{H\times 1}$, and mixing matrix $\Lambda \in \mathbb{R}^{F\times H}$. Often $\boldsymbol{\mu}$ is set to the zero vector by centering the observed variable $\mathbf{y}$ since we have $\mathbb{E}[\mathbf{y}] = \Lambda\mathbb{E}[\mathbf{s}]+\boldsymbol{\mu} = \boldsymbol{\mu}$. So the data vector $\mathbf{y}$ can be thought as a weighted combination of factors (columns of Λ) where the weights are randomly generated from standard multivariate Gaussian, plus some random Gaussian noise. By integrating out the hidden sources $\mathbf{s}$ we get $\mathbf{y}$ again a multivariate Gaussian $\mathbf{y} \sim \mathsf{Normal}(\boldsymbol{\mu}, \Psi + \Lambda\Lambda^T)$. Alternatively, FA can also be viewed as a way to represent the covariance matrix $\mathbb{V}[\mathbf{y}]$ with two components: a low rank matrix $\Lambda\Lambda^T$ and Ψ which is often assumed to be diagonal and correspond to the contribution of common factors and individual factors [Gorsuch, 1983], respectively. Also note that probabilistic Principal Component Analysis (PCA) can be treated as a special case of FA [Tipping and Bishop, 1999].

2.1.4 Modeling using Independent Component Analysis

Independent Component Analysis (ICA) [Bell and Sejnowski, 1995, Girolami, 2000, Roberts and Everson, 2001] assumes the observed data $\mathbf{y}$ is generated by the following model

$$\begin{aligned} \mathbf{y} &= \Lambda\mathbf{s} + \boldsymbol{\mu} + \mathbf{e} \\ \mathbf{s} &\sim p(.|\Phi) \\ \mathbf{e} &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{2.10}$$

from which we can see that ICA can be thought as generalization of FA by the fact that hidden source $\mathbf{s}$ is no longer restricted to be Gaussian. This generalization has significant consequences, which serves as the basis of ICA applications in signal processing [Roberts and Everson, 2001]. Briefly speaking, non-Gaussian hidden source $\mathbf{s}$ makes it possible to identify independent

components instead of just modeling the correlation among elements of $\mathbf{s}$ as in the Gaussian case.

One closely related technique of ICA is called Projection Pursuit (PP) [Kruskal, 1969, Friedman and Tukey, 1974, Huber, 1985], which seeks one projection at a time such that the extracted signal is as non-Gaussian as possible. This contrasts with ICA, which typically extracts H signals simultaneously from the observed mixtures. One practical advantage of PP over ICA is that the extraction process is incremental and can be stopped as needed; on the other hand, the parallel way of extract hidden sources makes ICA more robust than PP.

2.1.5 Some Extensions

Both linear regression and logistic regression can be seen as special cases of the Generalized Linear Models (GLM) [McCullagh and Nelder, 1989], which can be expressed as $y \sim P(g(\boldsymbol{\theta}^T\mathbf{x}))$ with $g(\boldsymbol{\theta}^T\mathbf{x})$ as the mean. Simply speaking, GLM generalizes linear regression in two ways: (1) allowing the response variable y to follow a distribution in the exponential family instead of just Gaussian; (2) introducing a link function $g(\mu)$ other than the identity function. Typical choices of the distribution $P(.)$ are normal, Bernoulli, Poisson and gamma, and details can be found in [McCullagh and Nelder, 1989] or [Dobson, 2001].

Linear models can be extended to the "nonlinear" case by first applying a feature mapping function $\phi : \mathbf{x} \mapsto \phi(\mathbf{x})$, and this can also be achieved by using the so-called "kernel trick": $K(\mathbf{x}_i, \mathbf{x}_j) = \langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle_{\mathcal{H}}$ in the Reproducing Kernel Hilbert Space. For example, Gaussian Process (GP) for regression [Williams, 1998] and classification [Gibbs and MacKay] can be thought as kernelized linear regression and logistic regression, respectively. This particular view has both conceptual and computational advantages. Conceptually, a prior over the parameters (e.g., regression coefficients) can often be treated as special cases of GP with properly chosen mean function and covariance function. Computationally we can directly compute a kernel function without the explicit computation of the mapping $\phi(\mathbf{x})$ which can be high or even infinite dimensional.

Many of the regression and classification methods can be reformulated as optimization problems, where we are trying to minimize some loss function. Viewing in this angle also has certain advantages. For example, this particular view can also be thought as the M-estimators [van der Vaart,

2000] in statistics which is a generalization of MLE estimators [Lehmann and Casella, 1998]. Popular loss functions for regression include squared error loss, ε-insensitive loss, absolute error loss, huber loss, etc. Popular loss functions for classification include exponential loss, logistic loss, hinge loss, asymmetric squared error loss and asymmetric huber loss, etc.

It is also possible to extend the basic FA and ICA models mentioned above. Note that both FA and ICA disregard any temporal or structural information in modeling the hidden sources, and as a result they are unable to capture the temporal relationship among $\mathbf{s}$ if any, for example. This limitation comes from the fact that we assume hidden sources $\mathbf{s}_k$'s are IID from some underlying distribution, which in fact is not necessary and its relaxation can be very helpful in some situations. In Chapter 3 we will present one model which is able to incorporate temporal information in the context of multi-task learning. Other possible extensions of ICA include non-linear ICA, which generalizes the linear relation $\Lambda\mathbf{s}$ into a non-linear relation $\Lambda(\mathbf{s})$.

2.1.6 Algorithms for Point Estimation

Point estimation is used everywhere in estimating parameters of non-Bayesian models as well as hyper-parameters in the empirical Bayes approach, which are all treated as fixed but unknown quantities (as opposed to be considered as random variables). Let us consider the MLE as an example, which is the most frequently used point estimation method. Given a training set $\mathcal{D}$, the likelihood can be written in general as $p(\mathcal{D}|\Theta)$, and the objective of MLE is to find the parameter Θ by maximizing the likelihood (or equivalently, the log-likelihood):

$$\hat{\Theta} \quad = \quad \arg\max_{\Theta} p(\mathcal{D}|\Theta) \tag{2.11}$$

which might accompany constraints like the non-negativity of some components of Θ. This is essentially a numerical optimization problem, and in many cases it can be converted into a convex optimization problem [Boyd and Vandenberghe, 2004] which is easy to solve for even large-scale systems. Algorithms for solving convex optimization problems are mature, and popular ones are gradient descent, conjugate gradient, Newton method and quasi-Newton method, etc [Nocedal and Wright, 1999, Luenberger, 2003, Boyd and Vandenberghe, 2004].

2.1.7 Algorithms for Bayesian Inference

We start with directed graphical models as they are great tools to represent and visualize hierarchical Bayesian models [Jordan, 2002]. Given a graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{X_1, X_2, \ldots, X_N\}$ composed of N random variables and $\mathcal{E}$ expressing the set of conditional dependence[1] among those variables, we are interested in the inference of $p(H|E)$ where both H and E are subsets of $\mathcal{V}$, correspond to unobserved nodes (hidden) and observed nodes (evidence). The computation of $p(H|E)$ can be difficult because of either a complicated graph structure or the existence of non-exponential link function among variables. Generally speaking, inference algorithms can be classified as either deterministic approximation algorithms or non-deterministic ones (e.g., sampling methods). Note that inference in Bayesian classification or regression is just a special case of the above general inference, in which the parameter θ is a random variable we want to do inference on. For classification tasks, future predictions are computed by integrating out the θ over its posterior distribution, e.g. $p(y|\mathbf{x}) = \int p(\theta|\mathcal{D})p(y|\theta, \mathbf{x})d\theta$, and confidence intervals can also be computed in a straightforward way. However, the likelihood function of classification is not within the exponential family and thus approximation is needed in the computation.

Variational Methods As an approach to function approximation, variational methods [Jaakkola and Jordan, 1997, Ghahramani and Beal, 2000, Jordan, 2002, Beal, 2003] convert the inference problem into an optimization problem by the application of appropriate inequalities. The approximation is usually done by optimizing some variational parameters so that the distance to the true quantity is minimized. It is deterministic and usually efficient, and desirable lower/upper bounds can often be obtained. Mean Field method [Saul et al., 1996] is one of the commonly used variational methods which constrains the candidate distribution to be factorized into individual components, and generalized Mean Field [Winn, 2003, Xing et al., 2003] allows the factorization into clusters of variables instead of singletons. A disadvantage of variational methods is that they may yield suboptimal solutions due to overly greedy assumptions.

[1] Formally speaking it only specifies the set of conditional independent relations. Two random variables can be independent even if the graphical model indicates their dependence.

Laplace Method Laplace method [Kass and Raftery, 1993] is one of the oldest methods used in physics and statistics to approximate posterior distributions. It is a very simple method which approximates the posterior distribution with a scaled Gaussian distribution that matches the true posterior through its mode, first and second derivatives at the mode, where the second derivative Hessian matrix is the covariance of the target Gaussian distribution. The disadvantage of Laplace method lies in the fact that it only considers up to second order derivative within a local range near the mode, which may not be good enough in some situations.

Belief Propagation Belief Propagation (BP) [Pearl, 1998] was first introduced for exact computation of inference in Bayesian networks, and later it is extended to loopy BP [Murphy et al., 1999] and generalized BP [Yedidia et al., 2002] for more complicated graphical models. For directed acyclic graphical models, BP is defined as a message passing protocol that converges after two operations: "collecting evidence" and "distributing evidence", which are implemented by a set of sum and product operations [Jordan, 2002]. BP is a fairly good method in general, and can be thought as a special case of the following Expectation Propagation method.

Expectation Propagation Expectation Propagation (EP) [Minka, 2001] is another approximate Bayesian inference algorithm which can be thought as an improvement over the Assumed Density Filtering (ADF). ADF tries to approximate the posterior distribution using a distribution within the exponential family $\mathcal{F}$ by minimizing the KL-divergence $\mathsf{KL}(p(x)||q(x))$, where $p(x)$ is the true distribution and $q(x)$ is the approximate one. It turns out that for $q(x) \in \mathcal{F}$, the exponential family, the minimization of KL-divergence is equivalent to the established moment matching method used in statistics to find approximations to distributions [Satterthwaite, 1946]. ADF does the moment matching in a sequential order, while EP performs iterative approximation using three steps: deletion, projection and updating. EP is a generalization of BP since it allows the use of exponential functions as an approximation to non-exponential messages. The advantage of EP is that moment matching is sensible in many aspects, but the disadvantage is that it does not guarantee convergence in general.

Sampling Methods As opposed to the above deterministic approximate Bayesian inference algorithms, sampling methods [Neal, 1993, Robert and

Casella, 2005] are non-deterministic approximate algorithms for Bayesian inference. The development of high-speed computers in the last decade makes sampling methods very popular for Bayesian methods. Furthermore, Markov Chain Monte Carlo techniques further push the popularity of Bayesian modeling, and there even exists a generic software package BUGS implementing Gibbs sampling. The advantages of sampling methods include its theoretical limiting properties and its relatively easy implementation; disadvantages include the difficulty in choosing convergence criteria (e.g., mixing and burn-in time, multiple chains), as well as slow convergence especially for high-dimensional problems.

2.2 Literature Review

Next we review some of the literature with respect to multi-task learning. Our review is by no means exhaustive on such a burgeoning area of research. The hope is to give readers a global picture of what are the problems that have already been explored and what are left, as well as the relative strengths and possible connections of the methods.

2.2.1 Basic Concepts

Multi-task learning is the problem which tries to estimate models for K tasks in a joint manner. Traditional learning, on the other hand, only considers one task at a time and solves them separately. Multi-task learning can be better understood by answering the following questions:

- *What is task relatedness?*
  Although most methods in multi-task learning assume some *relatedness* among tasks, the definition of *relatedness* varies. For example, model parameters may be partly shared among tasks, models may be transformation related or probabilistically related. Implicitly or explicitly, mathematically or procedurely, task relatedness must be specified under certain representation in order to play its role in multi-task learning. The main difference among existing methods lie in their assumptions and formulation of task relatedness, pretty much like the parametric form assumption in parametric models.

- *Why would multi-task learning methods work?*
  There is more than one way to explain why multi-task learning methods

would work. From the statistical estimation viewpoint, if some of the task parameters are shared, then they can be better estimated (e.g. with a smaller variance) given a lot of tasks. From the hierarchical Bayesian viewpoint, multi-task learning is essentially trying to learn a good prior (a.k.a. inductive bias in some context) over all tasks to capture task dependencies, which is often not applicable in single-task learning. In other context, it can be thought as trying to learn a set of features that are informative for all tasks.

- *When would multi-task learning be advantages?*
  Relatively speaking, multi-task learning methods will work better under the following conditions:

  1. When each task has limited amount of training resources;
  2. When the number of tasks is large;
  3. When the assumption about task relatedness is close to the truth.

  However, good multi-task learning methods should be robust in the sense that when some of the above conditions are violated the performance will not severely degrade.

- *When would multi-task learning fail?*
  If tasks are not related to each other at all, then it suffices to learn them separately (see also the answer to the previous question). If the assumption of task relatedness is inaccurate, then multi-task learning could even hurt performance by introducing undesirable biases. Note that even if the assumption is correct, multi-task learning might give slightly worse results than single-task learning for some individual task. This is not unexpected as all the arguments hold probabilistically and the overall performance should still be boosted when evaluating over all the tasks.

- *Is MTL computationally more expensive than STL?*
  On one hand, multi-task learning algorithms are often more complicated than the corresponding single-task learning algorithms because they often use the latter as components and need to do joint inference over all tasks parameters. Consequently, it can be more expensive to solve them due to the joint inference/coupling among task parameters. However, the computation should still be in the same order of magnitude as each iteration call to single-task learning modules does not require a full-blown solution. On the other hand, there are also

cases where multi-task learning can be computationally cheaper. For example, if we are able to select a joint subset of relevant features over given tasks then the cost to learn a future task is greatly reduced in the newly learned representation.

- *What form of data is required for multi-task learning?*
  In order to model the information sharing, task parameters need to share the same metric space (or at least partly, through transformation, etc). As a result, it often requires the input data space for each task to be the same, i.e. $\mathcal{X}^{(k)} = \mathcal{X}\,(k = 1, 2, \ldots, K)$. If the training data of those tasks are not in the same metric space, certain transformations are need in the pre-processing steps. How to transform data from multiple tasks into a unified representation is usually guided by human at the current stage.

Next, we describe the main approaches to multi-task learning in four categories: *artificial neural networks*, *shrinkage methods*, *regularized learning methods* and *hierarchical Bayesian models*, respectively. Our goal is to present a global preface to the reader, with the aspects which we see as being most fundamental for multi-task learning. Of course, we do not intend to cover every method in the literature, and our classification is not necessarily perfect in the sense that some boundaries may be blurred.

2.2.2 Artificial Neural Networks

Artificial neural networks are originally motivated from brain studies [Rosenblatt, 1959] and the simple perceptron is still one of the most widely used algorithms in machine learning. Generally speaking, neural networks consist of three types of units: *input units*, *hidden units* and *output units*. The set of input units take information about the example to be propagated through the network. By propagation, we mean that the information from the input will be passed through the network and reach the output units. Hidden units take input as the weighted sum of outputs from input units. Often that the number of hidden units is smaller than the number of input units. A weighted sum of outputs from the hidden units is then taken as the input to the output units. The training of a neural network is often achieved through the back-propagation algorithm [Rumelhart et al., 1986]. Neural networks are very powerful mathematical tools for machine learning, and they are known to be universal approximators in the sense that they

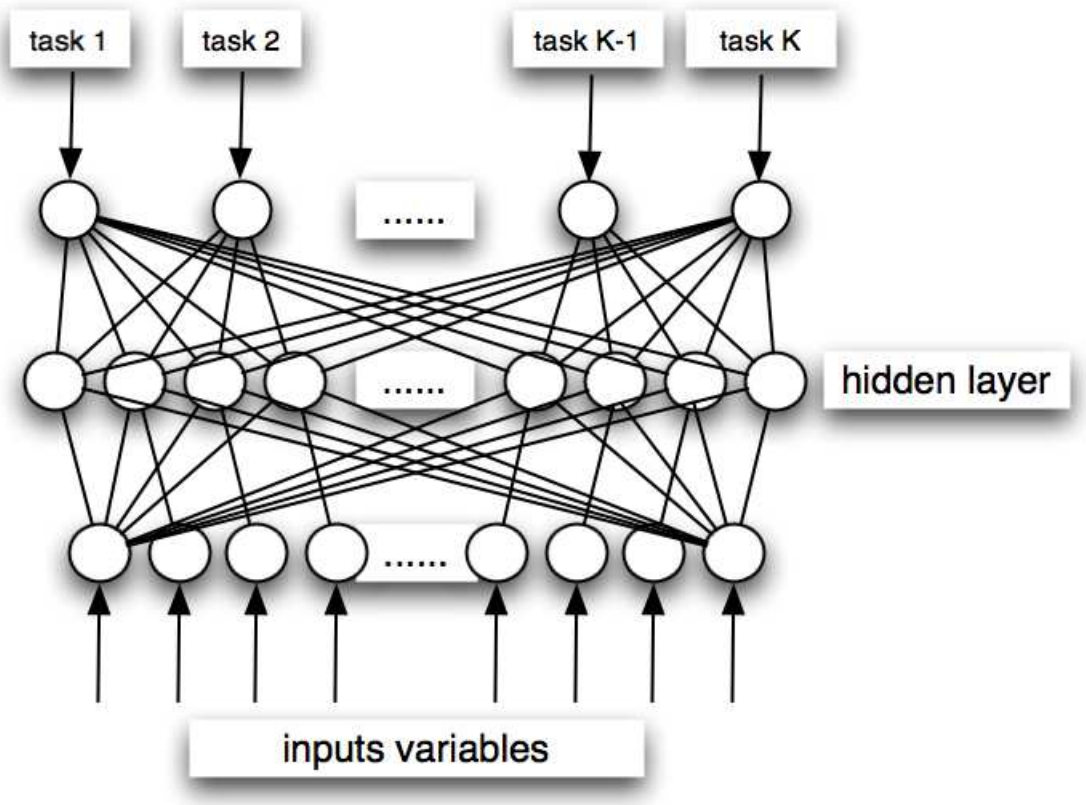

Figure 2.1: Multi-Task Learning using Neural Networks

are flexible enough to approximate any continuous function up to any given precision [Ivakhnenko, 1971].

Early work on multi-task learning [Thrun, 1996] [Caruana, 1997] [Silver and Mercer, 2001] uses neural network as the learning machine. Figure 2.1 shows a typical setting of multi-task learning with a two-layer neural network. Each hidden unit can be thought as a function of input variables and the shared components among tasks. The links in the first layer define the mappings from input variables to the shared components and the second layer links correspond to mappings from the shared components to tasks. Both levels of the mapping are jointly learned for all the tasks through back-propagation.

2.2.3 Shrinkage Methods

Shrinkage is an approach to obtain estimators that have smaller risks. Stein [1955] first showed that for the many normal means problem (e.g. given $X_i \sim \mathsf{Normal}(\mu_i, \sigma^2)$, estimate μ_i's, $i = 1, 2, \ldots, m$), the maximum likelihood estimator $\hat{\mu}_i = X_i, i = 1, 2, \ldots, m$ is not admissible[2] with respect to the total square error risk when $m > 2$. In other words, there exist other estimators which are uniformly better, such as the James-Stein estimator [James and Stein, 1961].

The basic idea of shrinkage methods is to trade bias for variance so that the overall risk is reduced compared to the original unbiased estimator. One of the simplest forms of shrinkage methods is to do proportional shrinkage, which defines a new estimator $\tilde{\mu} = b\hat{\mu}$ with $0 < b < 1$. To get a flavor of why this can help to reduce the risk, notice that in our example

$$\begin{aligned} \mathsf{bias}(\tilde{\mu}) &= (1-b)\mu \\ \mathsf{var}(\tilde{\mu}) &= b^2\mathsf{var}(\hat{\mu}) \end{aligned} \tag{2.12}$$

From equation (2.12) it is obviously that there always exists some $0 < b < 1$ such that $\mathsf{risk} = \mathsf{bias}^2 + \mathsf{var}$ is reduced, although the optimal amount of shrinkage depends on factors such as sample size. Shrinkage methods have broad applications and are related to regularization methods, as well as hierarchical Bayesian models.

Shrinkage methods have been applied to multi-task learning setting by Breiman and Friedman [1997], where they developed the Curds & Whey method for multivariate responses linear regression. The C&W procedure is a form of multivariate shrinkage. Its basic idea is to first transform the response variables into the canonical coordinate system, then conduct a proportional shrinkage estimation in this new coordinate system, and finally it transforms back into the original coordinate system. The optimal shrinkage in the transformed coordinate system can be determine by cross-validation techniques. According to the authors, the power of C&W method is to shrink in the right coordinate system and it can be viewed as a multivariate generalization of proportional shrinkage based on cross-validation. From the stand point of multi-task learning, the transformation into the new coordinate system is a key step which leverages information among multiple tasks.

[2] An estimator is said to be admissible with respect to a loss function for a class of distributions if there is no other estimator which has less than or equal to its loss for all distributions in the class, with the strict inequality holding for at least one distribution.

Shrinkage methods are intuitively simple and very effective methods for regression tasks. Furthermore, due to the fact that they are motivated by reducing the risk, they can achieve very good performance. However, shrinkage methods are often post-processing methods and it is not straightforward to generalize them to new tasks (compared to generative models, for example). Furthermore, the exact amount of shrinkage depends on the form of the risk function.

2.2.4 Regularized Learning Methods

In standard setting of supervised learning, we aim to find a function mapping f which maps an input vector $\mathbf{x} \in \mathcal{X}$ to an output $y \in \mathcal{Y}$. Usually we are given a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ which are identically and independently sampled from an unknown probability distribution $\mathcal{P}$: $(\mathbf{x}_i, y_i) \sim \mathcal{P}$. The objective is to find the best mapping function $f \in \mathcal{H}$ in the sense that the expected loss (with respect to $\mathcal{P}$) is minimized:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \mathbb{E}_{\mathcal{P}} L(f(\mathbf{X}), Y). \tag{2.13}$$

The most popular method is the *empirical risk minimization* [Vapnik, 1998] approach which replaces the unknown distribution $\mathcal{P}$ with the observed empirical distribution:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}_i), y_i) \tag{2.14}$$

However, empirical risk minimization is prone to overfitting. Regularization techniques [Tikhonov, 1963] were proposed to avoid overfitting in empirical risk minimization. They often have the form

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum L(f(\mathbf{x}_i), y_i) + \lambda \Omega(f) \tag{2.15}$$

where $\Omega(f)$ measures the model complexity or roughness of the prediction function f. Regularized learning methods have been widely and successfully used in statistics and machine learning including ridge regression, lasso regression, regularized logistic regression, SVM, etc., where the major differences lie in the choice of loss function $L(.,.)$ and penalty function $\Omega(.)$.

Regularized learning methods have recently been applied for multi-task learning problems in [Evgeniou and Pontil, 2004] and [Evgeniou et al., 2005]. In

their work, multi-task learning is achieved by using a joint regularization:

$$\hat{\boldsymbol{\theta}}_1, \dots \hat{\boldsymbol{\theta}}_K = \underset{\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_K}{\arg\min} \left\{ \sum_{k=1}^{K} \sum_{i=1}^{N_k} L(y_i^{(k)}, \langle \boldsymbol{\theta}_k, \mathbf{x}_i^{(k)} \rangle) + \lambda \Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K) \right\} \quad (2.16)$$

where $L(.,.)$ is taken to be the hinge loss and $\Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$ is taken to be a particular form

$$\Omega(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_K) = \left(\boldsymbol{\theta}_1^T, \boldsymbol{\theta}_2^T, \dots, \boldsymbol{\theta}_K^T\right) \mathbf{D} \begin{pmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_K \end{pmatrix} \quad (2.17)$$

in [Evgeniou et al., 2005]. The method proposed in [Evgeniou and Pontil, 2004] is similar to our "nosiy tasks" scenario under some specific settings, such as assuming both $\boldsymbol{\mu}$ and $\mathbf{e}_k$ following a multivariate Gaussian and performing point esitmation to obtain $\hat{\boldsymbol{\mu}}$ and $\hat{\mathbf{e}}_k$.

Ando and Zhang [2004] proposed a structure learning framework for multi-task learning. In their method, the predictive function for the k-th task is assumed to be

$$\begin{aligned} f^{(k)}(\mathbf{x}) &= \langle \mathbf{w}^{(k)} + \Theta^T \mathbf{v}^{(k)}, \mathbf{x} \rangle \\ &= \langle \mathbf{w}^{(k)}, \mathbf{x} \rangle + \langle \mathbf{v}^{(k)}, \Theta \mathbf{x} \rangle \end{aligned} \quad (2.18)$$

where the parameter Θ can be thought as the shared structure for a set of tasks. When learning those parameters, regularization is put on $\mathbf{w}^{(k)}$'s and $\mathbf{v}^{(k)}$'s. Alternatively, $\Theta\mathbf{x}$ can be thought as a set of good features that are learned from many tasks. This method is similar to one special case of our framework, if we assume the latent variables $\mathbf{s}_k$'s are multivariate Gaussian distributed and perform point estimations over $\mathbf{s}_k$'s.

2.2.5 Hierarchical Bayesian Models

Hierarchical Bayesian models [Box and Tiao, 1973, Bernardo and Smith, 1993, Gelman et al., 2003] are natural ways to model parameters that are related by the structure of the problem. In particular, the hierarchical structure can provide a flexible yet compact representation of the structure in the data, and thus produce models that can both fit the data well and generalize well on unseen, future data. As a result, we argue that hierarchical Bayesian

models are natural choices for representing the relatedness among tasks and modeling the task dependencies.

Baxter [1996] discussed the usage of hierarchical Bayesian models for studying multi-task learning problems. Parameters that are shared among tasks are treated as hyper-parameters at a higher level as opposed to the task-specific model parameters. Analysis are given from a Bayesian/information theoretical viewpoint.

Heskes [2000] presented a model for multi-task learning by assuming that response variables of each task follow a normal distribution. The mean of the normal distribution is learned using a two-layer neural network, and the variance is composed of a task specific component and a task independent component. Empirical Bayes method is used to learn the model hyper-parameters.

Teh et al [2005] proposed a semi-parametric model for multi-task learning. Their model uses Gaussian processes as the non-parametric components, and the predictive function of each task is a linear transformation of a set of basis Gaussian processes.

In [Yu et al., 2005] Gaussian processes is applied to learn multiple tasks. In particular, the predictive function of each task is assumed to be $f^{(k)} \sim \mathsf{GP}(m, K)$ where $m(.)$ and $K(.,.)$ are the *mean function* and *covariance function* of the Gaussian process, defined as:

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= m(\mathbf{x}) \\ \mathbb{C}(f(\mathbf{x}_i), f(\mathbf{x}_j)) &= K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{2.19}$$

Models proposed in this thesis can be generally seen as belonging to this category. Unlike previous work, we present a unified probabilistic framework and establish the connection between task relatedness and the underlying statistical assumptions. This also allows us to systematically explore important multi-task learning scenarios which are natural components of multi-task learning research. For example, our mixture model in Chapter 7 is a generalization of the work in [Yu et al., 2005].

2.2.6 Other Issues

2.2.6.1 Theoretical Analysis on Error Bounds

In standard supervised learning, generalization error bounds (a.k.a. large deviation bounds in statistics) can be obtained through the concept of VC

dimension (which measures the size of the hypothesis space). Naively speaking, good generalization (at certain accuracy ϵ with probability at least $1-\delta$) can be obtained as the number of examples is greater than a quantity that is a function of ϵ, δ and VC dimension (for quantitative results see [Vapnik, 1998], [Blumer et al., 1989] and [Ehrenfeucht et al., 1988]).

Early work on theoretical analysis of multi-task learning has been established in [Baxter, 1997, 2000]. Compared to the result of standard supervised learning, the basic statement is that, under mild conditions, the number of examples required of each task for good generalization will decrease as the number of tasks increases (and again the actual number depends on the accuracy and capacity of the hypothesis space). This result clearly justifies the benefit of "borrow information" from a theoretical viewpoint. In [Ben-David and Schuller, 2003] and [Ando and Zhang, 2004] the authors also develop specific generalization error bounds for multi-task learning under their formulation.

2.2.6.2 Parametric vs. Non-parametric Methods

Conventional statistical methods can be classified as *parametric* or *non-parametric* methods based on whether we restrict $f(\mathbf{x})$ to be of a particular functional form[3]. There is also a intermediate category called *semi-parametric* methods, and examples include models which have both parametric and non-parametric components. Generally speaking, parametric models are much more efficient when the assumption is correct, while nonparametric methods are free from model misspecification errors at the cost of a much slower convergence rate.

In multi-task learning we often have (in terms of hierarchical Bayesian model)

$$f^{(k)} \sim \mathcal{P}(.|\Theta), \quad k = 1, 2, \ldots, K. \tag{2.20}$$

We are not only required to model each predictive function $f^{(1)}, f^{(2)}, \ldots,$ but also facing the problem of how to model the distribution $\mathcal{P}(f|\Theta)$. Here the model $\mathcal{P}(f|\Theta)$ itself could be parametric, non-parametric, or semi-parametric. Our framework can be thought as having a flavor of semiparametric where the task sharing part $\mathbf{\Lambda s}$ plays a parametric role and the task specific part $\mathbf{e}_k$ works as a non-parametric component which allows $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$ to be flexible enough as K goes to infinity.

[3] As stated in [Wasserman, 2006], it is difficult to give a precise definition for "non-parametric". It means making as few assumptions as possible and can be understood as infinite-dimensional parametric in most cases.

2.2.6.3 Online vs. Retrospective Learning

The phases "learning to learn" [Thrun and Pratt, 1998] and "transfer learning" [Silver and Mercer, 2001] often refer to the situation that, given the fact we have learned K tasks, will the learning of the $(K+1)$-th task be easier? We can see that the emphasis is slightly different from the standard multi-task learning setting, i.e. after learning K tasks simultaneously, we want to do a better job with new tasks. Thus, the difference between multi-task learning and transfer learning is analogous to the difference between online learning and retrospective learning in conventional supervised learning.

Some of the multi-task learning methods cannot be directly applied to the transfer learning setting, due to the fact that they are post-processing and/or retrospective methods, such as the C&W method. On the other hand, for hierarchical Bayesian models it is straightforward to extend to the transfer learning setting as we have a generative model for $f^{(1)}, f^{(2)}, \ldots, f^{(K)}$ and learning a new task is easier given a better description of $\mathcal{P}(f)$.

Chapter 3

Probabilistic Models for MTL

In this chapter we present a unified probabilistic framework for multi-task learning, which is based on the assumption that tasks are related by sharing common structure through latent variables. The framework allows flexible modeling of both the common structure as well as statistical distributions of latent variables. Furthermore, we show that a series of important multi-task learning scenarios can be supported within the framework and present suitable models for them.

3.1 A Unified Probabilistic Framework with Latent Variables

Let us assume that we have K related tasks, and suppose we use $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ to represent the model parameters of K tasks (for example, to index their prediction functions $f(\mathbf{x}; \boldsymbol{\theta}_k)$) where $\boldsymbol{\theta}_k \in \mathbb{R}^{F \times 1}$ is the parameter vector of the k-th individual task. If we assume the existence of some latent variables which relate those $\boldsymbol{\theta}_k$'s, then we can represent those model parameters $\boldsymbol{\theta}_k$'s using a general latent variable model [Everitt, 1984]

$$p(\boldsymbol{\theta}) = \int f_1(\boldsymbol{\theta}|\mathbf{z}) f_2(\mathbf{z}) d\mathbf{z} \tag{3.1}$$

where $p(\boldsymbol{\theta})$ is the density of $\boldsymbol{\theta}$ and $\mathbf{z}$ stands for the underlying latent variable vector. It is clearly impossible to infer $f_1(.)$ and $f_2(.)$ uniquely from $p(.)$ just based on this definition; further assumptions are needed to achieve such a goal.

Let us return to the multi-task learning setting, and consider the following generative framework of $\boldsymbol{\theta}_k$'s:

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \boldsymbol{\mu} + \mathbf{e}_k \\ \mathbf{s}_1, \ldots, \mathbf{s}_K &\sim p(\mathbf{s}_1, \ldots, \mathbf{s}_K | \Phi) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{3.2}$$

where $\mathbf{s}_k \in \mathbb{R}^{H \times 1}$ and $p(.|\Phi)$ is assumed to be the hidden source model with Φ denoting its general distribution parameter; $\Lambda \in \mathbb{R}^{F \times H}$ is a linear transformation matrix on $\mathbf{s}_k$'s; $\boldsymbol{\mu} \in \mathbb{R}^{F \times 1}$ can be thought as the mean of the parameter vectors of multiple tasks, and the "noise" vector $\mathbf{e}_k \in \mathbb{R}^{F \times 1}$ is usually assumed to be multivariate Gaussian with diagonal covariance matrix $\Psi = \mathsf{diag}(\psi_{11}, \ldots, \psi_{FF})$ or even $\Psi = \sigma^2 \mathbf{I}$. In other words, we assume that the entries of $\mathbf{e}_k$ are independent from each other. Note that in general we can use any member of the exponential families to model $p(\mathbf{e}_k)$, however in most applications the $\mathbf{e}_k$ is taken to be a multivariate Gaussian distribution for convenience. The prior $p(\mathbf{s}_1, \ldots, \mathbf{s}_K | \Phi)$ is usually assumed to be $p(\mathbf{s}_1, \ldots, \mathbf{s}_K | \Phi) = \prod_{k=1}^{K} p(\mathbf{s}_k | \Phi)$.

Furthermore, we can also assume the parameters Λ and Ψ to be random variables by putting prior distributions over them to model particularly interesting structures, which we will discuss in detail in later chapters:

$$\begin{aligned} \Lambda &\sim p(\Lambda | \Delta) \\ \Psi &\sim p(\Psi | \Upsilon) \end{aligned} \tag{3.3}$$

The above framework in equation (3.2) is clearly a special case of equation (3.1) by decomposing

$$f(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \int p(\mathbf{s}_1, \ldots, \mathbf{s}_K | \Phi) \prod_{k=1}^{K} p(\boldsymbol{\theta}_k | \mathbf{s}_k) d\mathbf{s}_1 \ldots d\mathbf{s}_K \tag{3.4}$$

where $p(\boldsymbol{\theta}_k | \mathbf{s}_k) = \mathsf{Normal}(\Lambda \mathbf{s}_k + \boldsymbol{\mu}, \Psi)$. This framework can be thought as a generative process of how the $\boldsymbol{\theta}_k$'s are generated from a low dimensional space as we often have $H < F$, by an unknown linear transformation plus some random noise.

Even though the above framework is more specified than the general latent variable model in equation (3.1), it is still flexible enough to incorporate many models by specifying the distribution assumptions of $\mathbf{s}_k$'s and/or putting certain structural constraints on Λ and Ψ. Furthermore, the linear

transformation $\Lambda \mathbf{s}_k$ is robust and can capture the first order dependency, and it can be generalized to non-linear mapping function $\phi(\mathbf{s}_k; \Lambda)$ to enrich the framework. Note that the model specified in 3.1 is not identifiable due the the coupling between Λ and $\mathbf{s}_k$. However, this can be solved trivially by either putting a constraint on the variance of $p(\mathbf{s}_k)$ or the scale of Λ.

One major difference between our framework in equation (3.2) and the general latent variable model [Everitt, 1984] lies in the fact that $\boldsymbol{\theta}_k$'s are not observed but need to be inferred from observed data (i.e. latent variables). For example, in multi-task classification problems $\boldsymbol{\theta}_k$ is the parameter vector of the k-th classification task, assuming we use some linear classifier $f^{(k)}(\mathbf{x}) = \langle \boldsymbol{\theta}_k, \mathbf{x} \rangle = \boldsymbol{\theta}_k^T \mathbf{x}$. The default probabilistic model we will use for classification is the logistic regression model introduced in Chapter 2:

$$\begin{aligned} y^{(k)} &\sim \mathsf{Bernoulli}(\mu(\boldsymbol{\theta}_k^T \mathbf{x})) \\ \mu(t) &= \int_{-\infty}^{t} f(z)dz = \frac{1}{1+\exp(-t)} \end{aligned} \tag{3.5}$$

where $\mathsf{Bernoulli}(\mu)$ denotes the Bernoulli distribution with mean μ and $f(z)$ is the probability density function of standard logistic distribution.

The overall graphical model by combining equation (3.2) and (3.5) for the above learning framework is shown in Figure 3.1. In Figure 3.1 the observed variables are $\mathcal{D} = \mathcal{D}^{(1)} \cup \ldots \cup \mathcal{D}^{(K)}$ where $\mathcal{D}^{(k)} = \{(\mathbf{x}_1^{(k)}, y_1^{(k)}), \ldots, (\mathbf{x}_{n_k}^{(k)}, y_{n_k}^{(k)})\}$, the set of unknown random variables are $\mathcal{Z} = \{(\boldsymbol{\theta}_1, \mathbf{s}_1), \ldots, (\boldsymbol{\theta}_K, \mathbf{s}_K)\}$, and the set of parameters are $\Omega = \{\Phi, \Lambda, \boldsymbol{\mu}, \Psi\}$. It is worth mentioning that in Figure 3.1 all tasks do not need to share the same set of input instances (although that is the case for some of our experiments). The only requirement is that the input space for those tasks are the same, i.e. $\mathcal{X}^{(1)} = \ldots = \mathcal{X}^{(K)} \triangleq \mathcal{X}$.

Finally, it is interesting to point out that our model in equation (3.2) has a dual viewpoint. That is, if we construct $\tilde{\boldsymbol{\theta}}_f \in \mathbb{R}^{K \times 1}$ by taking the f-th corresponding coordinate of $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$, then those $\tilde{\boldsymbol{\theta}}_1, \ldots, \tilde{\boldsymbol{\theta}}_F$ can also be interpreted as a latent variable model where the meaning of the mixing matrix Λ and hidden sources $\mathbf{s}$ would be different and rather interesting. However, the associated difficulty is that given the model parameters Ω, those $\tilde{\boldsymbol{\theta}}_f$'s cannot be separately estimated since they will involve all the tasks.

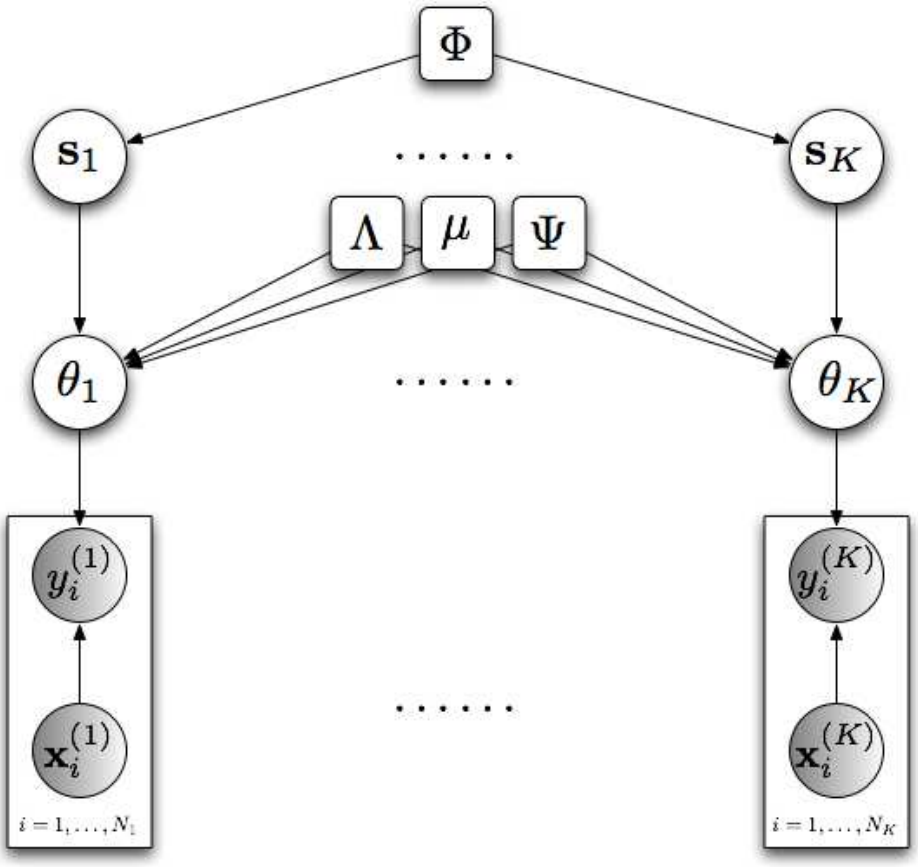

Figure 3.1: Graphical Model for Multi-task Learning

3.2 MTL Scenarios and Associated Probabilistic Models

For multi-task learning, it is important to identify different scenarios[1] – *how tasks are related to each other* – and use appropriate assumptions for each scenario explicitly. Here we analyze a series of important and interesting multi-task learning scenarios, and demonstrate how to use the generic framework presented above as a basis for the specific probabilistic models to leverage their dependencies in those multi-task learning scenarios. To be more specific we will show that the generality and flexibility mainly come from how to model the underlying sources $\mathbf{s}_k$'s, as well as whether special restrictions are put on the parameters Λ, Ψ, etc.

3.2.1 Independent Tasks

Our joint learning framework is clearly a generalization of single-task learning. By setting the parameters $\Lambda = \mathbf{0}_{F\times H}$ and $\boldsymbol{\mu} = \mathbf{0}_{F\times 1}$ we totally ignore the connections among $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ in the learning framework and have:

$$\boldsymbol{\theta}_k = \mathbf{e}_k \sim \mathsf{Normal}(\mathbf{0}, \Psi) \tag{3.6}$$

As a result it simply degenerates to learning K individual tasks separately. For example, if the classification model is logistic regression, then by doing a point estimation on θ_k we will get the standard Maximum A Posterior (MAP) estimation, and similarly we will get a Bayesian logistic regression model by inferring the posterior distribution of $\boldsymbol{\theta}_k$ given the observed data.

This simple degeneralization is illuminating to show different roles of $\Lambda\mathbf{s}_k$ and $\mathbf{e}_k$ played in modeling $\boldsymbol{\theta}_k$ in equation (3.2). $\Lambda\mathbf{s}_k$ is supposed to capture the shared information among tasks which does not need to be exclusive or perfect; $\mathbf{e}_k$ contributes to the remaining part that is task specific. In this viewpoint multi-task learning is a full-spectrum while single-task learning is just one end point!

3.2.2 Noisy tasks

Suppose our K tasks are all some noisy representations or versions of a single underlying task $\boldsymbol{\theta}_0 \in \mathbb{R}^{F\times 1}$. Then the generic framework simplifies

[1] By "scenario" we mean how tasks are related to each other. Formally it can be thought as the choice of parametric form in parametric density estimation.

with $\Lambda = \mathbf{0}_{F\times H}$ and $\boldsymbol{\mu} = \boldsymbol{\theta}_0$. That is, one may think of the application as to use different equipments to measure the same physical quantity of some object, the differences among those equipments can be modeled as noise, and the underlying model can be thought as the theoretically correct model. In other words we have

$$\boldsymbol{\theta}_k = \boldsymbol{\mu} + \mathbf{e}_k \sim \mathsf{Normal}(\boldsymbol{\mu}, \Psi) \tag{3.7}$$

where the covariance Ψ of $\mathbf{e}_k$ reflects our prior knowledge about how noisy those tasks are with respect to the centroid.

3.2.3 Clusters of tasks

This scenario can be thought as a generalization of the "noisy tasks" case, with the prior knowledge telling us that tasks should be grouped into certain number of clusters. One can simply use our framework in equation (3.2) to subsume this as a special case by specifying

$$\mathbf{s}_k \sim \mathsf{Multinomial}(1; p_1, p_2, \ldots, p_H) \tag{3.8}$$

where $\mathsf{Multinomial}(1; p_1, \ldots, p_H)$ stands for the Multinomial distribution with parameter $n = 1$ and proportional parameters $p_1, \ldots, p_H$ satisfying $p_h \geq 0$ and $\sum_{h=1}^{H} p_h = 1$.

So $\mathbf{s}_k$ will take the form $[0, \ldots, 0, 1, 0, \ldots, 0]$ where only one element is 1 and the rest are 0's. This means that each $\boldsymbol{\theta}_k$ randomly picks up one and only one column of the matrix Λ. As a result, the generated $\boldsymbol{\theta}_k$ will be clustered around the individual column vectors of Λ, $\Lambda_{\cdot j}$'s. This model resembles a Gaussian mixture model over the task space. In Chapter 7 we focus on this model and compare it to the simplified scenario "*noisy tasks*".

3.2.4 Tasks sharing a linear subspace

In this scenario tasks are assumed to be generated from a linear subspace for which we would think of each column of Λ as a basis and $\mathbf{s}_k$ stores the corresponding coordinates. In other words, the K tasks are sharing a common linear subspace. By assuming the hidden sources

$$\mathbf{s}_k \sim \mathsf{Normal}(\mathbf{0}, \mathbf{I}) \tag{3.9}$$

to be standard multivariate Gaussian distribution, this generative model for $\boldsymbol{\theta}_k$'s becomes the standard factor analysis model. In other words, those K

tasks share a linear subspace whose basis are the columns of the mixing matrix Λ, since we have $\boldsymbol{\theta}_k = \sum_{h=1}^{H} s_{k,h}\Lambda_{.h}$ where $\Lambda_{.h}$ is the h-th column of Λ and $s_{k,h}$ is the h-th element of $\mathbf{s}_k$.

More generally, we would assume the hidden source $\mathbf{s}_k$ to come from a product of generalized Gaussian distributions [Hyvarinen et al., 2001], whose probability density function is defined as

$$p(z) = \frac{\lambda\nu}{2\Gamma(1/\nu)} \exp\left(-\lambda|z|^{\nu}\right) \tag{3.10}$$

where ν denotes the shape parameter and λ relates to the variance, and $\Gamma(.)$ is the Gamma function defined as

$$\Gamma(z) = \int_0^{\infty} t^{z-1}e^{-t}dt \ \ (z > 0). \tag{3.11}$$

When $\nu = 2$ this reduces to standard Gaussian distribution and $\nu = 1$ it reduces to Laplace distribution. Furthermore, it is known that $\nu > 2$ will lead to sub-Gaussian distributions and $\nu < 2$ will lead to super-Gaussian distributions[2]. Figure 3.2 shows several plots of members in this distribution family.

3.2.5 Tasks with sparse representation

Sparsity has become one of the most important concepts in modern learning theory, and many algorithms are successful at least partially due to this property, including *winnow*, *lasso*, *SVM*, *wavelet*, etc. Sparsity usually means that only a small number of components of the solution are non-zero. In terms of distribution, sparsity could be generally explained as that the majority of the mass is distributed around zero.

Sparsity is a nice property since theoretically it is often related to the generalization capability if the assumption is close to truth (e.g. the relevant dimension is small), and practically it is often associated with computational advantages especially for high-dimension problems. Here we are interested in several types of sparsities:

[2]Formally, super-Gaussian (leptokurtic) are distributions which have positive *kurtosis* while sub-Gaussian (platykurtic) are distributions which have negative *kurtosis*. For a zero-mean random variable X, *kurtosis* is defined as $K(X) \triangleq \mathbb{E}[X^4] - 3(\mathbb{E}[X^2])^2$.

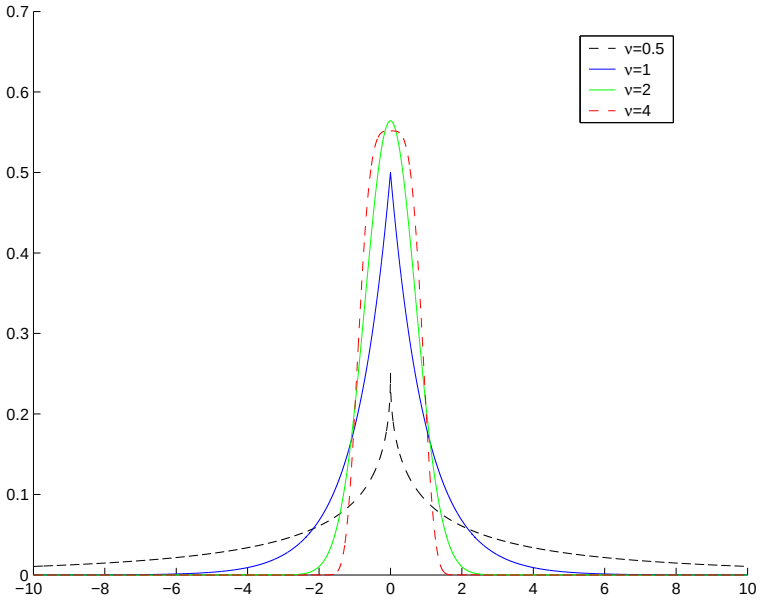

Figure 3.2: Generalized Gaussian Distributions

- The first sparsity can be specified by putting a super Gaussian distribution (e.g., Laplace distribution) over the hidden source $\mathbf{s}_k$, which essentially means that we assume that the target prediction functions of those K tasks are sparse linear combinations of basis prediction functions. The generative model corresponds to this scenario can be written as:

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \mathbf{e}_k \\ \mathbf{s}_k &\sim \prod_{h=1}^{H} \mathsf{Laplace}(0,1) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{3.12}$$

  Moreover, this model is of particular interest if we have an over-complete basis [Lewicki and Sejnowski, 2000] (e.g., Λ has a relatively large column basis), since in that situation sparsity could be crucial to a key property to have a reliable estimation.

- Alternatively, we could assume the matrix Λ is sparse by itself. This assumption will lead to a natural sparse solution since $\boldsymbol{\theta}_k$'s are linear combinations of columns of Λ and thus will also be sparse. This could

be achieved, for example, if we put a Laplace prior over each column of Λ in addition to the above model assumptions:

$$\Lambda_{\cdot h} \sim \prod_{f=1}^{F} \mathsf{Laplace}(0, 1) \tag{3.13}$$

As a result, a point estimation of Λ could lead to a natural sparse matrix.

In Chapter 6 we will show how to conduct joint feature selection for multi-task learning, which can be thought as another type of sparsity.

3.2.6 Tasks sharing a single component

There are situations when we have multiple tasks sharing a single component. Although this is very similar to "*noisy tasks*" and can be thought as a special case of treating the shared component as one column of Λ (or as μ), here we emphasize the point that different priors can be put on both the task-specific component and the task-independent component. Consider the following generative model with $k = 1, \ldots, K$:

$$\begin{aligned} \boldsymbol{\theta}_k &= \boldsymbol{\beta}_0 + \boldsymbol{\beta}_k \\ \boldsymbol{\beta}_0 &\sim \mathsf{Normal}(\mathbf{0}, \mathbf{V}_0) \text{ or } \prod_{f=1}^{F} \mathsf{Laplace}(0, \nu_0) \\ \boldsymbol{\beta}_k &\sim \mathsf{Normal}(\mathbf{0}, \mathbf{V}) \text{ or } \prod_{f=1}^{F} \mathsf{Laplace}(0, \nu_0) \end{aligned} \tag{3.14}$$

The shared component $\boldsymbol{\beta}_0$ among $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K$ has the same contribution to them, while $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_K$ can be thought as task specific preferences. One major difference with the model in equation (3.2) is that we also put a prior over the shared component $\boldsymbol{\beta}_0$. By using the product of Laplace as the prior distribution of $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_k$ we are able to achieve two types of sparse solutions. Having a sparse solution over β_0 means that we would like the shared component to be significantly supported by evidence from data if exists; while having a sparse solution over $\boldsymbol{\beta}_k$ have the effect that each individual task is assumed to only deviate from the shared community (all K tasks) when it is necessary.

3.2.7 Tasks sharing common relevant dimensions

Another interesting scenario is that we have the K tasks sharing a similar set of features (which is only small subset of the original set of features), although the contributions of those features to the K prediction tasks can be quite different (e.g., some of them can be positive contribution while others being negative contribution). This scenario can be thought as a variant of "*sharing a linear subspace*", while the linear subspace is aligned with the original feature space. We would have the following generative model to capture the scenario:

$$\begin{aligned} \boldsymbol{\theta}_k &\sim \mathsf{Normal}(\mathbf{0}, \mathsf{diag}(\boldsymbol{\alpha})) \\ \alpha_f &\sim \mathsf{InvGamma}(\nu_\alpha, \nu_\beta) \end{aligned} \tag{3.15}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_F)^T \in \mathbb{R}^{F \times 1}$ is a non-negative vector which specifies the variance of each dimension of $\boldsymbol{\theta}_k$ and its each dimension follows a prior distribution such as Inverse-Gamma distribution (or any other sensible distribution over $\mathbb{R}^+$):

$$\mathsf{InvGamma}(z \mid \nu_\alpha, \nu_\beta) = \frac{\nu_\beta^{\nu_\alpha}}{\Gamma(\nu_\alpha)} z^{-\nu_\alpha - 1} \exp(-\frac{\nu_\beta}{z}) \tag{3.16}$$

By sharing the same $\boldsymbol{\alpha}$ among all K tasks, we are able to reflect the assumption that those K tasks tend to share the same relevant dimensions, although positive/negative effects on each dimension could vary depending on the data likelihood.

Equivalently, this model can be represented as a special case of the framework in equation (3.2) as follows:

$$\begin{aligned} \boldsymbol{\theta}_k &= \mathbf{e}_k \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \\ \Psi_{f,f} &\sim \mathsf{InvGamma}(\nu_\alpha, \nu_\beta) \end{aligned} \tag{3.17}$$

where $\Psi = \mathsf{diag}(\Psi_{1,1}, \ldots, \Psi_{F,F})$. That is, $\boldsymbol{\theta}_k$ equals $\mathbf{e}_k$ whose covariance matrix Ψ is assumed to be diagonal and random. Figure 3.3 shows the graphical model for this scenario. In Chapter 6 we will discuss how to do joint feature selection for multi-task learning and its connection to this scenario.

3.2.8 Duplicated tasks

In reality it may happen that we need to solve exactly the same task which we have already solved previously, although there is no indicator telling us

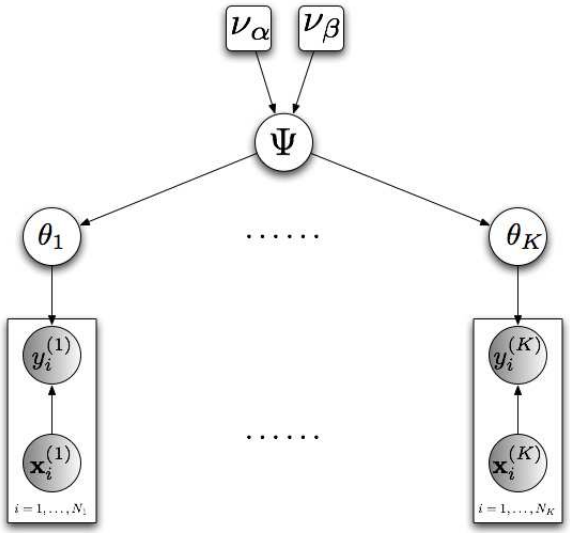

Figure 3.3: Graphical Model for Tasks Sharing Relevant Dimensions

which task it is unless we infer from the training data. Formally, we want to consider the situation where it is likely that we have $\boldsymbol{\theta}_k$ identical to one of the previous models $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{K-1}\}$. In other words, we assume that the probability that we will meet previously solved tasks again in the future is positive and bounded away from zero (as opposed to the probability that a continuous variable takes a particular value, which equals zero). Having this in mind, we can use non-parametric Bayesian techniques like Dirichlet Process [Ferguson, 1973] to model the generation process of the $\boldsymbol{\theta}_k$'s:

$$\begin{aligned} G &\sim \mathsf{DP}(\alpha, G_0) \\ \boldsymbol{\theta}_k &\sim G \end{aligned} \tag{3.18}$$

where α is the precision parameter of Dirichlet Process and G_0 is its base distribution. By using Dirichlet Process to model the generation of $\boldsymbol{\theta}_k$ we will have non-zero prior mass on previous seen tasks.

Alternatively we could use Dirichlet Processes to model the generation of θ_k through $\mathbf{s}_k$ as in our model in Figure 3.1. The advantage of that is we could still have "exact" tasks subject to some random noise $\mathbf{e}_k$, and those different tasks are still related through the shared linear subspace. The hierarchical model to capture this scenario can be summarized as:

$$G \sim \mathsf{DP}(\alpha, G_0)$$

$$\begin{aligned} \mathbf{s}_k &\sim G \\ \theta_k &= \Lambda \mathbf{s}_k + \mathbf{e}_k \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{3.19}$$

where any appropriate distribution over $\mathbf{s}_k$ could be the candidate of the base distribution G_0. Due to the property of DP, given $\mathbf{s}_1, \ldots, \mathbf{s}_{k-1}$, the probability that $\mathbf{s}_k$ equals one of them is strictly greater than zero. Consequently $\boldsymbol{\theta}_k$ is identical to one previous model subject to some random noise, and the generative model is able to capture the scenario of duplicated tasks. It is worth mentioning that although this model could be approximated by a finite mixture model as in the "*clusters of tasks*" scenario, DPs provide a natural way of handling increasing number of clusters as the number of tasks grows.

3.2.9 Evolving tasks

For all previous discussions we assume that tasks are exchangeable, which means that the order of those task parameters does not matter in our generative model. However, there are cases of multi-task learning where tasks are evolving one after another. For this scenario, the model should reflect that fact that $\boldsymbol{\theta}_k$'s are evolving. One of the simplest choices, for example, is to assume that there is a first-order Markov chain over $\boldsymbol{\theta}_k$'s:

$$\boldsymbol{\theta}_{k-1} \rightarrow \boldsymbol{\theta}_k \tag{3.20}$$

which can be fully specified using the transition probability $p(\boldsymbol{\theta}_k | \boldsymbol{\theta}_{k-1})$.

Similar to the scenario of "*duplicated tasks*", we can make the $\mathbf{s}_k$'s not IID in Figure 3.1. That is, we assume a Markov chain over $\mathbf{s}_k$'s instead of over $\boldsymbol{\theta}_k$'s:

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \boldsymbol{\mu} + \mathbf{e}_k \\ \mathbf{s}_{k-1} &\rightarrow \mathbf{s}_k \end{aligned} \tag{3.21}$$

One particular advantage of using the latter model is that we have a Markov chain over a relatively low dimensional space of size H instead of size F. As a result, we are only responsible for the estimation of parameters involved in $p(\mathbf{s}_k | \mathbf{s}_{k-1})$ instead of $p(\boldsymbol{\theta}_k | \boldsymbol{\theta}_{k-1})$, which is quadratic in the dimensionality of $\boldsymbol{\theta}$. This model is closely related to the linear State Space Model in the literature [Ghahramani and Hinton, 1996, Minka, 1999] which is widely

used in modeling dynamic systems. By making assumptions on the transition probability $p(\mathbf{s}_t \mid \mathbf{s}_{t-1})$ it is possible to do inference on the multiple evolving tasks with observed data. For example, this model can be applied to the problem of *concept drift* [Klinkenberg and Joachims, 2000] where the underlying model for classification/regression drifts as it proceeds.

3.3 Summary

In this chapter we proposed a generic probabilistic framework for multi-task learning. The framework is a special case of hierarchical Bayesian model and latent variable model. We analyze a series of important task scenarios and present suitable models within the framework. From the exploration we can see that the flexibility of the proposed framework in Figure 3.1 comes from the fact that we can model different assumptions about the latent variables and have certain assumptions over the mixing matrix Λ, covariance matrix Ψ, etc. It should be clear that there could exist alternative ways of modeling those task scenarios, but putting them in a unified framework make the different assumptions and restrictions transparent and comparison easy.

The choice of the parametric form $p(\mathbf{s}|\Phi)$ for hidden source $\mathbf{s}$ is crucial in our framework and can make a lot of differences in terms of combining the domain knowledge into the modeling process, in pretty much the same way as the choice of parametric forms in parametric density estimation. As we already showed, when $p(\mathbf{s}|\Phi)$ is assumed to be the multinomial distribution we are essentially performing clustering over the task space; if $p(\mathbf{s}|\Phi)$ is assumed to be super Gaussian distributions like Laplace, then we are expecting to have a more sparse solution than using Gaussian instead. Furthermore, $\mathbf{s}$ can be generalized to be a mixed type random vector if needed, where some coordinates are continuous and others are discrete. Generally speaking, the choice of $p(\mathbf{s}|\Phi)$ should reflect the domain knowledge about how the tasks are associated with each other, i.e. what people refer to as "task relatedness". Furthermore, the shared parameters including Λ, μ, Ψ can also be modeled to capture different scenarios, such as a sparsity constraint on Λ.

Although we presented most of the models for simultaneously learning K tasks, it should be clear that multi-task learning can also be applied to situations where we are learning those tasks sequentially. In that case, learning a new task should be easier given the fact that we already learned the shared components from previous related tasks. On the other hand, we would like to point out that in order for the proposed framework to work effectively with

respect to learning each task individually, there are some mild conditions that need to be satisfied (refer to Chapter 2 for details):

- Task relatedness: Tasks should be related so that we can borrow information from each other when learn them jointly. If we use $\boldsymbol{\theta}_k$ to denote the parameter of the k-th task's prediction function, then it is necessary that those K tasks should share the same input space (at least partly) in order to have some common components in $\boldsymbol{\theta}_k$'s.

- Number of related tasks: In order to have a reliable estimation of the shared part, we need to have certain number of tasks in order to obtain a reliable estimation of the shared components among those tasks. Although the exact answer depends on the particular task domain, in general we prefer to have more than five tasks to apply the joint learning framework.

- Training resources of individual tasks: There are limited training resources for those related tasks. Actually as the amount of available resources grows, under regularity conditions some general principles (Maximum Likelihood Estimator, Bayes Estimator, etc) for single-task learning will lead to the same, asymptotic optimal solution. On the other hand, the joint learning framework will benefit the most when training resources are quite limited, due to the fact that the shared components will be learned using all resources from K tasks.

However, it is expected that by controlling the model complexity of the shared components, violations of some of the above conditions should not significantly deteriorate the joint learning compared to individual learning. A better question to ask is when is it appropriate to use which particular model for multi-task learning, and this is essentially a model selection problem and will be discussed in Chapter 8.

Chapter 4

Learning and Inference Algorithms

The probabilistic framework for multi-task learning presented in Chapter 3 is a hierarchical Bayesian model [Gelman et al., 2003] as well as a latent variable model [Everitt, 1984]. Compared to conventional latent variable models such as factor analysis [Gorsuch, 1983] or independent component analysis [Hyvarinen et al., 2001], the key difference is that in multi-task learning those $\boldsymbol{\theta}_k$'s are not observed (latent) and have to be estimated from the training data.

In this chapter we focus on the learning and inference algorithms for models presented in Chapter 3. Generally speaking, given a probabilistic model we can either use a full Bayesian approach, an empirical Bayes approach, or a point estimation approach to learn the model. The full Bayesian approach has the advantage of taking into consideration the uncertainties of parameters using their posterior distributions, and does not suffer from the overfitting problem. However it is computationally expensive and often intractable for high-dimensional problems, and thus certain approximation algorithms are necessary to apply it in realistic situations. The point estimation approach discards the uncertainty of parameters and just considers their point estimations instead. By doing so it can be computationally very efficient, but may suffer from overfitting. The empirical Bayes approach can be thought as in-between of these two approaches, which incorporates the uncertainty of the intermediate level parameters but tries to perform point estimation for hyper-parameters.

We present algorithms for both the empirical Bayes approach and point

estimation approach, with the former being able to capture the uncertainty in the parameters while the later more suitable for high-dimensional tasks. Here we focus on how to conduct learning and inference in the model shown in equation (4.1):

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \boldsymbol{\mu} + \mathbf{e}_k \\ \mathbf{s}_k &\sim p(.|\Phi) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \\ \mathsf{classification}: y_{i_k}^{(k)} &\sim \mathsf{Bernoulli}(\sigma(\boldsymbol{\theta}_k^T \mathbf{x}_{i_k}^{(k)})) \\ \mathsf{regression}: y_{i_i}^{(k)} &\sim \mathsf{Normal}(\boldsymbol{\theta}_k^T \mathbf{x}_{i_k}^{(k)}, \sigma^2) \end{aligned} \tag{4.1}$$

where $\sigma(t) = (1+\exp(-t))^{-1}$ is the standard logistic function, $k = 1, 2, \ldots, K$ is the task index and $i_k = 1, 2, \ldots, N_k$ is the index of data instances for task k. Superscript k on both $\mathbf{x}$ and y indicate the task that they are associated with, i.e., we have $\mathcal{D}^{(k)} = \{(\mathbf{x}_{i_k}^{(k)}, y_{i_k}^{(k)})_{i_k=1}^{N_k}\}$ as the training set for the k-th task.

4.1 Empirical Bayes Approach

The upper level of the graphical model in Figure 3.1 captures the relations among tasks. We can use an empirical Bayes approach to learn the parameters $\Omega = \{\Phi, \Lambda, \boldsymbol{\mu}, \Psi\}$ from the data while treating the variables $\mathcal{Z} = \{(\boldsymbol{\theta}_k, \mathbf{s}_k)_{k=1}^K\}$ as hidden variables (and thus will integrate them out). Because Λ and $\mathbf{s}_k$ are always coupled together in our model, we have the usual identifiability issue [Lehmann and Casella, 1998] in estimating those parameters. In particular, to get around the un-identifiability caused by the interaction between Λ and $\mathbf{s}_k$, we assume that Φ is of standard parametric form (e.g., zero mean and unit variance) and thus remove it from Ω. The goal is to learn point estimators $\hat{\Lambda}$, $\hat{\boldsymbol{\mu}}$ and $\hat{\Psi}$ as well as obtain posterior distributions over hidden variables given training data.

Given the training data $\mathcal{D} = \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \ldots \cup \mathcal{D}^{(K)}$, the log-likelihood of incomplete data $\log p(\mathcal{D} \mid \Omega)$[1] can be calculated by integrating out hidden variables

$$\mathcal{L} = \sum_{k=1}^{K} \log \left\{ \int \prod_{i_k=1}^{N_k} p(y_{i_k}^{(k)} \mid \mathbf{x}_{i_k}^{(k)}, \boldsymbol{\theta}_k)$$

[1] Here for simplicity we just use $p(\mathcal{D} \mid \Omega)$ to denote the likelihood instead of conditioning on input vectors.

$$\left(\int p(\boldsymbol{\theta}_k \mid \mathbf{s}_k, \Lambda, \boldsymbol{\mu}, \Psi) p(\mathbf{s}_k|\Phi) d\mathbf{s}_k \right) d\boldsymbol{\theta}_k \Bigg\} \tag{4.2}$$

for which the maximization over parameters $\Omega = \{\Lambda, \boldsymbol{\mu}, \Psi\}$ involves two complicated integrals on $\boldsymbol{\theta}_k$ and $\mathbf{s}_k$, respectively.

The integration over $\mathbf{s}_k$ will be easy if $p(\mathbf{s}_k|\Phi)$ is Gaussian ($p(\boldsymbol{\theta}_k|\mathbf{s}_k, \Lambda, \boldsymbol{\mu}, \Psi)$ is also Gaussian), otherwise approximation is needed. Furthermore, for classification tasks the likelihood function $p(y|\mathbf{x}, \boldsymbol{\theta})$ is typically non-exponential and thus exact calculation becomes intractable. However, we can approximate the solution by applying the Expectation Maximization (EM) algorithm [Dempster et al., 1977] to decouple the maximization process into a series of simpler E-steps and M-steps. In the EM formulation instead of maximizing the log-likelihood of the observed data $p(\mathcal{D} \mid \Omega)$, we attempt to maximize the expectation of the joint log-likelihood of both the observed data and the hidden variables in the model $\mathbb{E}[\log p(\mathcal{D}, \mathcal{Z} \mid \Omega)]$, where the expectation is taken with respect to some distribution $q(\mathcal{Z})$ over the hidden variable $\mathcal{Z}$. It is straightforward to show that this expectation is always a lower bound of the incomplete data likelihood with equality holding if $q(\mathcal{Z}) = p(\mathcal{Z} \mid \mathcal{D}, \Omega)$.

The EM algorithm for the empirical Bayes approach can be briefly stated as follows:

1. **E-step:** Given the parameter $\Omega^{t-1} = \{\Lambda, \boldsymbol{\mu}, \Psi\}^{t-1}$ calculated from the previous $(t-1)$-th step, calculate the distribution of hidden variables given Ω^{t-1} and $\mathcal{D}$:
$$p(\mathcal{Z} \mid \Omega^{t-1}, \mathcal{D}) \tag{4.3}$$

2. **M-step:** Maximize the expected log-likelihood of complete data $(\mathcal{Z}, \mathcal{D})$, where the expectation is taken over the distribution of hidden variables obtained in the E-step, and the maximization is done with respect to Ω:
$$\Omega^t = \arg\max_{\Omega} \mathbb{E}_{p(\mathcal{Z}|\Omega^{t-1}, \mathcal{D})} \left[\log p(\mathcal{D}, \mathcal{Z} \mid \Omega) \right]. \tag{4.4}$$

4.1.1 M-step

We will begin with the M-step instead since it is easier than the E-step. The log-likelihood of complete data can be written as follows:

$$\log p(\mathcal{D}, \mathcal{Z} \mid \Omega)$$

$$
\begin{aligned}
&= \sum_{k=1}^{K} \log \left\{ \left(\prod_{i_k=1}^{N_k} p(y_{i_k}^{(k)} \mid \mathbf{x}_{i_k}^{(k)}, \boldsymbol{\theta}_k) \right) p(\boldsymbol{\theta}_k \mid \mathbf{s}_k, \Lambda, \mu, \Psi) p(\mathbf{s}_k \mid \Phi) \right\} \\
&= \sum_{k=1}^{K} \left\{ \sum_{i_k=1}^{N_k} \log p(y_{i_k}^{(k)} | \mathbf{x}_{i_k}^{(k)}, \boldsymbol{\theta}_k) + \log p(\boldsymbol{\theta}_k | \mathbf{s}_k, \Lambda, \mu, \Psi) + \log p(\mathbf{s}_k | \Phi) \right\}
\end{aligned} \tag{4.5}
$$

and its expected value with respect to $q(\mathcal{Z})$ is:

$$
\begin{aligned}
&\mathbb{E}\left[\log p(\mathcal{D}, \mathcal{Z} \mid \Omega)\right] \\
= &\sum_{k=1}^{K} \left\{ \sum_{i_k=1}^{N_k} \mathbb{E}[\log p(y_{i_k}^{(k)} \mid \mathbf{x}_{i_k}^{(k)}, \boldsymbol{\theta}_k)] + \mathbb{E}[\log p(\boldsymbol{\theta}_k \mid \mathbf{s}_k, \Lambda, \mu, \Psi)] + \mathbb{E}[\log p(\mathbf{s}_k \mid \Phi)] \right\}
\end{aligned} \tag{4.6}
$$

The first and third terms in the curly bracket are the likelihood term for classification/regression and source prior and do not depend on any of the parameters Ω (since we assumed that $p(\mathbf{s}|\Phi)$ is of standard form), and thus can be dropped off in the M-step.

Consequently, the M-step can be simplified by maximizing the following expectation with respect to the parameters to be estimated, namely $\Lambda, \boldsymbol{\mu}, \Psi$:

$$
\begin{aligned}
&\arg\max \sum_{k=1}^{K} \mathbb{E}[\log p(\boldsymbol{\theta}_k \mid \mathbf{s}_k, \Lambda, \boldsymbol{\mu}, \Psi)] \\
= &\arg\max \sum_{k=1}^{K} \mathbb{E}\left[-\frac{1}{2} \log |2\pi\Psi| - \frac{1}{2} (\boldsymbol{\theta}_k - \Lambda \mathbf{s}_k - \boldsymbol{\mu})^T \Psi^{-1} (\boldsymbol{\theta}_k - \Lambda \mathbf{s}_k - \boldsymbol{\mu}) \right] \\
= &\arg\min \sum_{k=1}^{K} \Big\{ \log |\Psi| + \mathsf{Tr}\left(\Psi^{-1} (\mathbb{E}[\boldsymbol{\theta}_k \boldsymbol{\theta}_k^T] + \Lambda \mathbb{E}[\mathbf{s}_k \mathbf{s}_k^T] \Lambda^T + \boldsymbol{\mu}\boldsymbol{\mu}^T \right) \\
+ &\mathsf{Tr}\left(\Psi^{-1} (-2\mathbb{E}[\boldsymbol{\theta}_k \mathbf{s}_k^T] \Lambda^T - 2\mathbb{E}[\boldsymbol{\theta}_k] \boldsymbol{\mu}^T + 2\Lambda \mathbb{E}[\mathbf{s}_k] \boldsymbol{\mu}^T) \right) \Big\}
\end{aligned} \tag{4.7}
$$

where $\mathsf{Tr}(.)$ stands for the matrix trace operator that returns the sum of the diagonal elements. Setting the derivative with respect to Λ to zero we get

$$
\sum_{k=1}^{K} \left\{ 2\mathbb{E}[\mathbf{s}_k \mathbf{s}_k^T] \Lambda^T - 2\mathbb{E}[\mathbf{s}_k \boldsymbol{\mu}_k^T] + 2\mathbb{E}[\mathbf{s}_k] \boldsymbol{\mu}^T \right\} = 0 \tag{4.8}
$$

similarly for $\boldsymbol{\mu}$ we have

$$\sum_{k=1}^{K} \{2\boldsymbol{\mu} - 2\mathbb{E}[\boldsymbol{\theta}_k] + 2\Lambda\mathbb{E}[\mathbf{s}_k]\} = 0 \tag{4.9}$$

and for Ψ we have

$$\begin{aligned} \Psi &= \frac{1}{K}\sum_{k=1}^{K} \big\{\mathbb{E}[\boldsymbol{\theta}_k\boldsymbol{\theta}_k^T] + \Lambda\mathbb{E}[\mathbf{s}_k\mathbf{s}_k^T]\Lambda^T + \boldsymbol{\mu}\boldsymbol{\mu}^T \\ &- 2\mathbb{E}[\boldsymbol{\theta}_k\mathbf{s}_k^T]\Lambda^T - 2\mathbb{E}[\boldsymbol{\theta}_k]\boldsymbol{\mu}^T + 2\Lambda\mathbb{E}[\mathbf{s}_k]\boldsymbol{\mu}^T\big\} \end{aligned} \tag{4.10}$$

Combining the last three equations we can solve them to get the final estimations:

$$\begin{aligned} \hat{\Lambda} &= \left(\sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k\mathbf{s}_k^T] - \frac{1}{K}(\sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k])(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k])^T\right) \\ &\times \left(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k\mathbf{s}_k^T] - \frac{1}{K}(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k])(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k])^T\right)^{-1} \end{aligned} \tag{4.11}$$

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \left(K - (\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k])^T(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k\mathbf{s}_k^T])^{-1}(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k])\right)^{-1} \\ &\times \left(\sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k] - \sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k\mathbf{s}_k^T](\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k\mathbf{s}_k^T])^{-1}\mathbb{E}[\mathbf{s}_k]\right) \end{aligned} \tag{4.12}$$

Since we assume Ψ to be a diagonal matrix it is only necessary to assign the diagonal elements to $\hat{\Psi}$, which can also be verified by directly considering the constrained optimization problem. If we do not assume that we know the parametric form of $p(\mathbf{s}|\Phi)$ then we should also treat Φ as an unknown parameter and update it during the M-step as follows:

$$\hat{\Phi} = \arg\max \sum_{k=1}^{K}\mathbb{E}[\log p(\mathbf{s}_k \mid \Phi)]. \tag{4.13}$$

Below we consider the special case when $\boldsymbol{\mu} = 0$, which can greatly simplify the notation[2]. In this case,

$$\hat{\Lambda} = (\sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k\mathbf{s}_k^T])(\sum_{k=1}^{K}\mathbb{E}[\mathbf{s}_k\mathbf{s}_k^T])^{-1}$$

[2]This is usually fine as $\boldsymbol{\mu}$'s functionality can be roughly contributed by one column of Λ if there is one element of $\mathbf{s}_k$ that is const across tasks.

$$\hat{\Psi} = \frac{1}{K}\left(\sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k\boldsymbol{\theta}_k^T] - (\sum_{k=1}^{K}\mathbb{E}[\boldsymbol{\theta}_k\mathbf{s}_k^T])\hat{\Lambda}^T\right) \tag{4.14}$$

4.1.2 E-step

In the E-step we need to calculate posterior distribution $p(\mathcal{Z} \mid \mathcal{D}, \Omega)$, given the parameter Ω calculated in previous M-step. Essentially only the first and second order moments matters in the E-step, namely: $\mathbb{E}[\boldsymbol{\theta}_k]$, $\mathbb{E}[\mathbf{s}_k]$, $\mathbb{E}[\boldsymbol{\theta}_k\boldsymbol{\theta}_k^T]$, $\mathbb{E}[\mathbf{s}_k\mathbf{s}_k^T]$ and $\mathbb{E}[\boldsymbol{\theta}_k\mathbf{s}_k^T]$ are needed in the M-step.

However, the exact calculation is often intractable for several reasons. First, $p(\mathbf{s}_k|\Phi)$ may not be Gaussian or may not even be within the exponential family; second, for classification tasks the likelihood function $p(y|\boldsymbol{\theta}, \mathbf{x})$ does not belong to the exponential family and thus cannot be summarized with finite sufficient statistics as data grows. Under such situations, we need to come up with a easy-to-handle (e.g., belonging to the exponential family) approximation $q(\mathcal{Z})$ that minimizes some distance measure $\mathsf{Distance}(p(\mathcal{Z}|\mathcal{D}, \Omega), q(\mathcal{Z}))$ between the true posterior $p(\mathcal{Z}|\mathcal{D}, \Omega)$ and the approximate one $q(\mathcal{Z})$, where common distance measures include Kullback-Leibler (KL) divergence $\mathsf{KL}(p(\mathcal{Z}|\mathcal{D}, \Omega)||q(\mathcal{Z}))$ and $\mathsf{KL}(q(\mathcal{Z})||p(\mathcal{Z}|\mathcal{D}, \Omega))$ (since KL-divergence is asymmetric), which are defined as:

$$\mathsf{KL}(p(x)||q(x)) = \int p(x)\log\frac{p(x)}{q(x)}dx. \tag{4.15}$$

Fortunately, the E-step for K tasks is decoupled given the parameter Ω, its calculation can be done by conducting inference on a separate graphical model for each task, as shown in Figure 4.1.

Since the resulting task for the E-step is essentially inference in a graphical model. Inference can be carried out using general-purpose algorithms like variational methods, belief propagation or expectation propagation, as introduced in Chapter 2. For example, if we use Gaussian distributions to approximate the posterior distributions $p(\boldsymbol{\theta}_k|\Omega^{t-1}, \mathcal{D})$ and $p(\mathbf{s}_k|\Omega^{t-1}, \mathcal{D})$, one particular choice of approximation criteria (actually EP only approximates this goal) can lead to the following E-step in case of expectation propagation:

$$\begin{aligned}\{\mathbb{E}[\boldsymbol{\theta}_k], \mathbb{V}[\boldsymbol{\theta}_k]\} &\approx \arg\min_{\mathbf{m},\mathbf{V}} \mathsf{KL}\left(p(\boldsymbol{\theta}_k \mid \Omega^{t-1}, \mathcal{D}) \parallel \mathsf{Normal}(\boldsymbol{\theta}; \mathbf{m}, \mathbf{V})\right) \\ \{\mathbb{E}[\mathbf{s}_k], \mathbb{V}[\mathbf{s}_k]\} &\approx \arg\min_{\mathbf{m},\mathbf{V}} \mathsf{KL}\left(p(\mathbf{s}_k \mid \Omega^{t-1}, \mathcal{D}) \parallel \mathsf{Normal}(\mathbf{s}; \mathbf{m}, \mathbf{V})\right)\end{aligned} \tag{4.16}$$

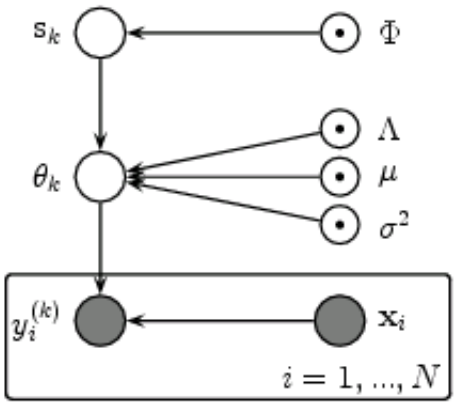

Figure 4.1: Graphical Model for E-step Inference of Task k

For $p(.)$ belonging to the exponential family it is well-known that minimizing KL-divergence is equivalent to moment matching [Minka, 2001]. If we reverse the order in the KL-divergence to minimize $\mathsf{KL}(q||p)$ then we end up using variational method for the approximate inference. For now we use the variational method which is known to be more robust with guaranteed convergence and often results in good quality approximations.

The basic idea of variational methods is to lower bound the log-likelihood using Jensen's inequality:

$$\mathcal{L} = \log p(\mathcal{D}) = \log \int p(\mathcal{D}, \mathcal{Z}) d\mathcal{Z} \geq \int q(\mathcal{Z}) \log \frac{p(\mathcal{D}, \mathcal{Z})}{q(\mathcal{Z})} d\mathcal{Z} \triangleq \mathcal{O} \qquad (4.17)$$

where the inequality is due to the concavity of the logarithm function. The RHS of the above equation is the objective we want to maximize in the variational method, and $q(\mathcal{Z})$ is usually taken to be within the exponential family so that it is easy to compute. It is straightforward to show that maximizing this lower bound is equivalent to minimizing the KL-divergence $\mathsf{KL}(q(\mathcal{Z})||p(\mathcal{Z}|\mathcal{D}))$ and the calculated $q(\mathcal{Z})$ can then be used as an approximation to the true posterior distribution $p(\mathcal{Z}|\mathcal{D}, \Omega)$.

For simplicity we will omit the task index k in the following and simply denote the classification parameter vector as θ and the hidden source vector as $\mathbf{s}$. Furthermore we also assume the $q(\mathbf{s}, \boldsymbol{\theta})$ can be factorized in the following form

$$q(\mathbf{s}, \boldsymbol{\theta}) \quad = \quad q(\boldsymbol{\theta}) q(\mathbf{s}) \qquad (4.18)$$

This is often a reasonable simplifying assumption and allows us to do the optimization iteratively. Furthermore we assume $q(\boldsymbol{\theta}) = \mathsf{Normal}(\mathbf{m}_{\boldsymbol{\theta}}, \mathbf{V}_{\boldsymbol{\theta}})$ to be a multivariate Gaussian distribution and $q(\mathbf{s}) = f(\mathbf{s}|\mathbf{m}_s, \mathbf{I})$ which follows some parametric distribution with mean $\mathbf{m}_s$ and unit covariance matrix.

Note that Gaussian distribution is usually a good and convenient choice, particularly considering the fact that we only need the first and second order statistics in the M-step. Now we have

$$\begin{aligned}\mathcal{O} &= \int\int q(\boldsymbol{\theta})q(\mathbf{s}) \log \frac{p(\mathbf{s})p(\boldsymbol{\theta}|\mathbf{s})\prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}, \mathbf{x}_i)}{q(\boldsymbol{\theta})q(\mathbf{s})} d\boldsymbol{\theta} d\mathbf{s} \\ &= \int q(\mathbf{s}) \left[\log \frac{p(\mathbf{s})}{q(\mathbf{s})} + \int q(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta}|\mathbf{s})\prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}, \mathbf{x}_i)}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \right] d\mathbf{s} \quad (4.19)\end{aligned}$$

Although the posterior distribution $q(\mathbf{s}, \boldsymbol{\theta})$ is assumed to be factorized, $\mathbf{s}$ and $\boldsymbol{\theta}$ are still coupled in above equation by the distribution $p(\boldsymbol{\theta}|\mathbf{s})$. In order to tackle the problem we propose the following iterative algorithm to solve the E-step, which optimizes $q(\mathbf{s})$ and $q(\boldsymbol{\theta})$ interchangeably:

1. Given $q(\mathbf{s}) = f(\mathbf{s}|\mathbf{m}_s, \mathbf{I})$, the first term in equation (4.19) does not involve $q(\boldsymbol{\theta})$ and thus can be dropped off. The second term can also be greatly simplified since only $\log p(\boldsymbol{\theta}|\mathbf{s})$ involves $\mathbf{s}$ and it can be easily integrated out due to the Gaussianity of $\boldsymbol{\theta}$ given $\mathbf{s}$.

$$\begin{aligned}\mathbb{E}_{q(\mathbf{s})}[\log p(\boldsymbol{\theta}|\mathbf{s})] &= \mathbb{E}_{q(\mathbf{s})}[-\frac{1}{2}\log|2\pi\Psi|] \\ &+ \mathbb{E}_{q(\mathbf{s})}[-\frac{1}{2}(\boldsymbol{\theta} - \Lambda\mathbf{s} - \boldsymbol{\mu})^T\Psi^{-1}(\boldsymbol{\theta} - \Lambda\mathbf{s} - \boldsymbol{\mu})] \quad (4.20)\end{aligned}$$

   As a result, we can obtain an estimate of $q(\theta)$.

2. Given $q(\boldsymbol{\theta}) = \mathsf{Normal}(\boldsymbol{\theta}; \mathbf{m}_{\boldsymbol{\theta}}, \mathbf{V}_{\boldsymbol{\theta}})$, for similar reason the second term in equation (4.19) can also be greatly simplified. That is, only the term $\log p(\boldsymbol{\theta}|\mathbf{s})$ involves $\mathbf{s}$ and its expectation with respect to $q(\boldsymbol{\theta})$ can be written down. So the final objective function of the optimization over $q(\mathbf{s})$ composes of two terms: a cross entropy term and a quadratic term which penalizes the distance between $q(\mathbf{s})$ and $\mathbb{E}[\boldsymbol{\theta}]$.

The detailed algorithm about the E-step is listed in Algorithm 1 for reference, and we would like to comment on several things. First, we assume the form of $q(\boldsymbol{\theta})$ to be multivariate Gaussian, which is a reasonable choice especially

Algorithm 1 An Iterative Algorithm for E-step

1. Initialize $q(\mathbf{s})$ with some standard distribution, such as

$$q(\mathbf{s}) \quad = \quad \prod_{h=1}^{H} \mathsf{Normal}(0,1) \quad or \quad \prod_{h=1}^{H} \mathsf{Laplace}(0,1). \tag{4.21}$$

2. Calculate the expected value of $\mathbf{s}$: $\mathbb{E}_{q(\mathbf{s})}[\mathbf{s}]$.

3. Solve a Bayesian logistic/linear regression with a prior $\mathsf{Normal}(\Lambda\mathbb{E}[\mathbf{s}] + \boldsymbol{\mu}, \Psi)$ on $\boldsymbol{\theta}$ (see later section for details):

$$q(\boldsymbol{\theta}) \quad \leftarrow \quad \arg\max_{q(\boldsymbol{\theta})} \left\{ \int q(\boldsymbol{\theta}) \log \frac{\mathsf{Normal}(\boldsymbol{\theta}; \Lambda\mathbb{E}[\mathbf{s}] + \boldsymbol{\mu}, \Psi) \prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}, \mathbf{x}_i)}{q(\boldsymbol{\theta})} d\boldsymbol{\theta} \right\} \tag{4.22}$$

4. Calculate the expected value of $\boldsymbol{\theta}$: $\mathbb{E}_{q(\boldsymbol{\theta})}[\boldsymbol{\theta}]$.

5. Update $q(\mathbf{s})$:

$$\begin{aligned} q(\mathbf{s}) \quad \leftarrow \quad & \arg\max_{q(\mathbf{s})} \left\{ \int q(\mathbf{s}) \left[\log \frac{p(\mathbf{s})}{q(\mathbf{s})} - \frac{1}{2} \mathsf{Tr}\left(\Psi^{-1} (\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T]) \right) \right. \right. \\ & \left. \left. - \frac{1}{2} \mathsf{Tr}\left(\Psi^{-1} ((\Lambda\mathbf{s} + \boldsymbol{\mu})(\Lambda\mathbf{s} + \boldsymbol{\mu})^T - 2\mathbb{E}[\boldsymbol{\theta}](\Lambda\mathbf{s} + \boldsymbol{\mu})^T) \right) \right] d\mathbf{s} \right\} \end{aligned} \tag{4.23}$$

6. Repeat steps 2-5 until convergence.

considering the fact that only first and second moments are needed in the M-step. Second, the prior choice of $q(\mathbf{s})$ in step 3 is significant since for each $\mathbf{s}$ we only have one associated "data point" $\boldsymbol{\theta}$. In particular using the Laplace distribution will lead to a more sparse solution of $\mathbb{E}[\mathbf{s}]$, and this will be made more clear in the corresponding point estimation algorithm. Finally, for sparsity models we could take the parametric form of $q(\mathbf{s})$ to be the product of Laplace distribution with unit variance but unknown mean, where the fixed variance is intended to remove the unidentifiability issue caused by the interaction between scales of $\mathbf{s}$ and Λ. Although using a full covariance Gaussian for $q(\mathbf{s})$ is another choice, again due to the unidentifiability reason caused by rotations of $\mathbf{s}$ and Λ we could make it a diagonal Gaussian. As a result, we argue that the product of Laplaces is better than the product of Gaussians since it has the same parametric form as the prior $q(\mathbf{s})$, and the overall goal in step 5 is to estimate the distribution mean $\mathbf{m_s} = (m_1, \ldots, m_H)$.

In general we have $q(\mathbf{s}) = f(\mathbf{s}|\mathbf{m})$ where $\mathbf{m}$ is the mean and $q(\mathbf{s})$ is assumed to have standard variance. For $p(\mathbf{s})$ a Gaussian distribution $\mathsf{Normal}(\mathbf{0}, \mathbf{I})$, step 5 becomes

$$\begin{aligned} \hat{\mathbf{m}} &= \arg\min_{\mathbf{m}} \left\{ \mathbf{m}^T\mathbf{m} + \mathbf{m}^T\Lambda^T\Psi^{-1}\Lambda\mathbf{m} - 2\mathbf{m}^T\Lambda^T\Psi^{-1}\mathbb{E}[\boldsymbol{\theta}] \right\} \\ &= \left(\mathbf{I} + \Lambda^T\Psi^{-1}\Lambda\right)^{-1}\Lambda^T\Psi^{-1}\mathbb{E}[\boldsymbol{\theta}] \end{aligned} \tag{4.24}$$

and for $p(\mathbf{s})$ product of Laplace distributions $\prod_{h=1}^{H}\mathsf{Laplace}(0,1)$ step 5 becomes

$$\begin{aligned} \hat{\mathbf{m}} &= \arg\min_{\mathbf{m}} \Big\{ 2||\mathbf{m}||_1 + 2\sum \exp(-|m_f|) \\ &+ \mathbf{m}^T\Lambda^T\Psi^{-1}\Lambda\mathbf{m} - 2\mathbf{m}^T\Lambda^T\Psi^{-1}\mathbb{E}[\boldsymbol{\theta}] \Big\} \end{aligned} \tag{4.25}$$

which need to be solved numerically. Note that similar to the sparsity property of L_1 norm, here we have when $||\mathbf{m}||_1$ is large the l_1 norm dominates $\sum_{f=1}^{F}\exp(-|m_f|)$ and thus the distribution mean achieves a more "sparse" solution in terms of the mean, e.g., more mass is around zero. Later we will show that for point estimation approach the sparsity property will be made more clear.

4.1.3 Variational Method for Bayesian Logistic Regression

In the following we restrict our discussions to logistic regression as our base classifier and present an algorithm for solving the Bayesian logistic regres-

sion, which is used in step 3 of Algorithm 1. Algorithms for other probabilistic classifiers can also be developed in principle. Our algorithm here is based on the variational method originally proposed by Jaakkola and Jordan [Jaakkola and Jordan, 1997], which is an elegant algorithm that is guaranteed to converge, and experimentally it has been verified to be stable and efficient.

Here we ignore the task index k as well. Given a Gaussian prior $\mathsf{Normal}(\mathbf{m}_0, \mathbf{V}_0)$ over the parameter vector $\boldsymbol{\theta}$ and a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, we would like to obtain an approximation to the true posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. In the following we essentially use an exponential function to approximate the non-exponential likelihood function

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-y\boldsymbol{\theta}^T \mathbf{x})} \tag{4.26}$$

which in turn makes the Bayes formula tractable.

Note that the function $\log(1/(1 + \exp(-z)))$ is a convex function in the variable z^2 (which can be verified by taking the second derivative with respect to z^2), we can use the first order Taylor series to expand at ξ^2 with respect to z^2. Due to the concavity, we have the following inequality:

$$\begin{aligned} p(y|\mathbf{x}, \boldsymbol{\theta}) &\geq \sigma(\xi) \exp\left\{(y\mathbf{x}^T\boldsymbol{\theta} - \xi)/2 - \lambda(\xi)((\mathbf{x}^T\boldsymbol{\theta})^2 - \xi^2)\right\} \\ &\triangleq p(y|\mathbf{x}, \boldsymbol{\theta}, \xi) \end{aligned} \tag{4.27}$$

where $\sigma(z) = 1/(1 + \exp(-z))$ is the logistic function and $\lambda(\xi)$ is defined as $\lambda(\xi) = \tanh(\xi/2)/4\xi$.

Our goal is to maximize the lower bound of

$$p(y|\mathbf{x}) = \int p(\boldsymbol{\theta}) p(y|\mathbf{x}, \boldsymbol{\theta}) d\boldsymbol{\theta} \geq \int p(\boldsymbol{\theta}) p(y|\mathbf{x}, \boldsymbol{\theta}, \xi) d\boldsymbol{\theta} \tag{4.28}$$

In order to maximize the RHS lower bound $\int p(\boldsymbol{\theta}) p(y|\mathbf{x}, \boldsymbol{\theta}, \xi) d\boldsymbol{\theta}$, we formulate an EM algorithm by treating ξ as the parameter in MLE and $\boldsymbol{\theta}$ as the hidden variable, and the resulting steps are:

$$\begin{aligned} \mathsf{E-step:}\ \mathbf{Q}(\xi, \xi^t) &= \mathbb{E}\left[\log\{p(\boldsymbol{\theta}) p(y|\mathbf{x}, \boldsymbol{\theta}, \xi)\} \mid \mathbf{x}, y, \xi^t\right] \\ \mathsf{M-step:}\ \xi^{t+1} &= \arg\max_{\xi} \mathbf{Q}(\xi, \xi^t) \end{aligned} \tag{4.29}$$

Since both terms in the expectation are exponential functions and the expectation is taken over a Gaussian distribution $p(\boldsymbol{\theta}|\mathbf{x}, y, \xi^t)$, the E-step can

be actually be computed as:

$$\begin{aligned}(\mathbf{V}^t)^{-1} &\leftarrow \mathbf{V}^{-1} + 2\lambda(\xi^t)\mathbf{x}\mathbf{x}^T \\ \mathbf{m}^t &\leftarrow \mathbf{V}^t(\mathbf{V}^{-1}\mathbf{m} + y\mathbf{x}/2)\end{aligned} \tag{4.30}$$

where the superscript t means the t-th step, and we assume $p(\boldsymbol{\theta}) = \mathsf{Normal}(\boldsymbol{\theta}; \mathbf{m}, \mathbf{V})$. Taking the derivative of $\mathbf{Q}(\xi, \xi^t)$ with respect to ξ and setting it to zero leads to:

$$\xi^{t+1} = \sqrt{\mathbf{x}^T\mathbf{V}^t\mathbf{x} + (\mathbf{x}^T\mathbf{m}^t)^2}. \tag{4.31}$$

Although solving this EM can give us a good lower bound of the log likelihood function $p(y|\mathbf{x}) = \int p(\boldsymbol{\theta})p(y|\mathbf{x}, \boldsymbol{\theta})d\boldsymbol{\theta}$, it involves expensive matrix inverse calculation in the E-step. Actually this EM procedure can be greatly simplified by realizing the Woodbury formula [Golub and Loan, 1996]:

$$\left(\mathbf{A} + \mathbf{B}\mathbf{C}^T\right)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{I} + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}^T\mathbf{A}^{-1}. \tag{4.32}$$

The advantage of applying the Woodbury formula is that if both $\mathbf{B}$ and $\mathbf{C}$ are low rank matrices, computing $(\mathbf{I} + \mathbf{C}^T\mathbf{A}^{-1}\mathbf{B})^{-1}$ can be much more efficient, which is exactly our case. By simplifying we can get the following results as follows:

- E-step:

$$\begin{aligned}\mathbf{V}^t &\leftarrow \mathbf{V} - \frac{2\lambda(\xi)}{1 + 2\lambda(\xi)\mathbf{x}^T\mathbf{V}\mathbf{x}}\mathbf{V}\mathbf{x}(\mathbf{V}\mathbf{x})^T \\ \mathbf{m}^t &\leftarrow \mathbf{m} - \frac{2\lambda(\xi)}{1 + 2\lambda(\xi)\mathbf{x}^T\mathbf{V}\mathbf{x}}\mathbf{V}\mathbf{x}\mathbf{x}^T\mathbf{m} \\ &\quad + \frac{y}{2}\mathbf{V}\mathbf{x} - \frac{y}{2}\frac{2\lambda(\xi)}{1 + 2\lambda(\xi)\mathbf{x}^T\mathbf{V}\mathbf{x}}\mathbf{V}\mathbf{x}\mathbf{x}^T\mathbf{V}\mathbf{x}\end{aligned} \tag{4.33}$$

- M-step: solving a one-dimensional fixed point equation iteratively ($c = \mathbf{x}^T\mathbf{V}\mathbf{x}$)

$$\begin{aligned}\xi^2 &= c - \frac{2\lambda(\xi)}{1 + 2\lambda(\xi)c}c^2 \\ &+ \left(\mathbf{x}^T\mathbf{m} - \frac{2\lambda(\xi)}{1 + 2\lambda(\xi)c}c\mathbf{x}^T\mathbf{m} + \frac{y}{2}c - \frac{y}{2}\frac{2\lambda(\xi)}{1 + 2\lambda(\xi)c}c^2\right)^2\end{aligned} \tag{4.34}$$

Note the original EM steps is simplified to first compute a fixed-point solution of a one-dimensional problem, then compute the E-step in one-shot. Furthermore, in the computation of the E-step we do not need to calculate the matrix inverse $\mathbf{V}^{-1}$ any more.

4.1.4 Variational Method for High Dimensional Task

Although the computation of the above method is affordable for medium scale problems (e.g., having several thousand features), the memory requirement and time complexity become unaffordable as the number of features grows. For high dimensional tasks we have $\boldsymbol{\theta}_k \in \mathbb{R}^{F\times 1}$ where $F \gg 1$. For example, this could happen in text or image domain where the number of features can easily go up to more than ten thousand. Given such a high dimensional vector space, approximations using full covariance Gaussian distributions are no longer applicable due to both time and memory constraints. In order to handle such cases we consider a fully factorized version of the above variational method for which we have:

$$q(\boldsymbol{\theta}_k) = \prod_{j=1}^{F} q(\boldsymbol{\theta}_{k,j}) \tag{4.35}$$

This full factorization assumption is essentially equivalent to the assumption that the approximating Gaussian distribution $q(\boldsymbol{\theta}_k)$ has the following mean and diagonal covariance matrix:

$$\begin{aligned} \mathbb{E}[\boldsymbol{\theta}_k] &= (\mu(\theta_{k,1}), \ldots, \mu(\theta_{k,F}))^T \\ \mathbb{V}(\boldsymbol{\theta}_k) &= \mathsf{diag}(\sigma^2(\theta_{k,1}), \ldots, \sigma^2(\theta_{k,F})) \end{aligned} \tag{4.36}$$

and thus for each individual component we have $q(\theta_{k,j}) = \mathsf{Normal}(\mu(\theta_{k,j}), \sigma^2(\theta_{k,j}))$. This additional assumption reduces the memory complexity from quadratic to linear in terms of F, the number of features. As a result the algorithm could be very efficient and competitive with point estimation approach. Since it is not clear how to construct a tight bound for fully factorizable approximation, we could use Laplace approximation method introduced in Chapter 2, by first obtaining the MAP estimate $\boldsymbol{\theta}^{MAP}$:

$$\begin{aligned} \boldsymbol{\theta}^{MAP} &= \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} \log(1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}_i)) \right. \\ &+ \left. \log|2\pi\mathbf{V}_0| + \frac{1}{2}(\boldsymbol{\theta} - \mathbf{m}_0)^T \mathbf{V}_0^{-1} (\boldsymbol{\theta} - \mathbf{m}_0) \right\} \end{aligned} \tag{4.37}$$

The update rule of step 3 in Algorithm 1 now becomes first solving the MAP estimation of $\boldsymbol{\theta}^{MAP}$ and then finding the Laplace approximation using a fully factorized multivariate Gaussian distribution.

Algorithm 2 Iterative Algorithm for L_1 Regularized Problem

1. Given the optimization objective

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathcal{O}(\mathbf{w}) = \arg\min_{\mathbf{w}} \{\mathbf{w}^T \mathbf{A}\mathbf{w} + \mathbf{b}^T \mathbf{w} + \lambda ||\mathbf{w}||_1\}$$

2. Initialize $\mathbf{w} = \mathbf{0} \in \mathbb{R}^{F \times 1}$.

3. Loop until convergence:

   (a) Pick up a w_f (can be sequentially or with other heuristics)

   (b) - $w_f > 0$:

$$\begin{aligned} \triangle w_f &= \frac{-A_{f.}\mathbf{w} - b_f/2 - \lambda/2}{A_{ff}} \\ w_f &\leftarrow \max(0, w_f + \triangle w_f) \end{aligned}$$

   - $w_f < 0$:

$$\begin{aligned} \triangle w_f &= \frac{-A_{f.}\mathbf{w} - b_f/2 + \lambda/2}{A_{ff}} \\ w_f &\leftarrow \min(0, w_f + \triangle w_f) \end{aligned}$$

   - $w_f = 0$: Update w_f only if $|A_{f.}\mathbf{w} + b_f/2| > \lambda/2$

$$\begin{aligned} \triangle w_f &= \frac{-A_{f.}\mathbf{w} - b_f/2 + sign(-A_{f.}\mathbf{w} - b_f/2)}{A_{ff}} \\ w_f &\leftarrow \triangle w_f \end{aligned}$$

4.2 Point Estimation

In our model if we also treat $\mathcal{Z}$ as non-random but unknown parameters as Ω, then we can obtain their MLE estimators. Unfortunately, in the following we show that straightforward application of MLE to this generative model without any constraint will lead to fully decoupled MLE estimations for each individual task, and as a result the model fails to borrow information among tasks. To see this, notice that for unconstrained MLE we have

$$\begin{aligned} \{\mathcal{Z}, \Omega\}_{MLE} &= \arg\max_{\mathcal{Z},\Omega} \mathcal{L}(\mathcal{Z}, \Omega|\mathcal{D}) \\ &= \arg\max_{\mathcal{Z},\Omega} \{p(\mathcal{D}|\mathcal{Z})p(\mathcal{Z}|\Omega)\} \end{aligned} \tag{4.38}$$

and the following solution obviously maximizes the above joint likelihood $\mathcal{L}(\mathcal{Z}, \Omega|\mathcal{D})$:

$$\hat{\Lambda} = \mathbf{0},\ \hat{\mu} = \mathbf{0},\ \hat{\Psi} \to \infty. \tag{4.39}$$

In fact as long as the variance Ψ goes to infinity, the models of K tasks will be fully decoupled and thus this generative framework fails to capture the relations among tasks. The failure of unconstrained MLE demonstrates the importance of having a finite value Ψ. Actually it is possible to assume Ψ to be fixed when optimizing other parameters while use cross-validation as a wrapper to tune the optimal value of the diagonal elements of Ψ. Given Ψ fixed, point estimations of the rest parameters become well-behaved. In Algorithm 1 if we take a limiting case by letting both $q(\boldsymbol{\theta})$ and $q(\mathbf{s})$ converging to the Dirac delta function, then step 3 can be thought as finding the MAP estimation of $\boldsymbol{\theta}$ and step 5 becomes the following optimization problem for the case of Gaussian sources

$$\hat{\mathbf{m}}_\mathbf{s} = \arg\min_{\mathbf{m}_\mathbf{s}} \{\mathbf{m}_\mathbf{s}^T \mathbf{m}_\mathbf{s} + \mathbf{m}_\mathbf{s}^T \Lambda^T \Psi^{-1} \Lambda \mathbf{m}_\mathbf{s} - 2\mathbf{m}_\mathbf{s}^T \Lambda \Psi^{-1} \mathbb{E}[\boldsymbol{\theta}]\} \tag{4.40}$$

and it becomes lasso-like optimization problem ($\mathbf{m}_\mathbf{s}$ denotes the point estimation of $\mathbf{s}$ here) for the case of Laplace sources

$$\begin{aligned} \hat{\mathbf{m}}_\mathbf{s} &= \arg\min_{\mathbf{m}_\mathbf{s}} \{2||\mathbf{m}_\mathbf{s}||_1 + \mathbf{m}_\mathbf{s}^T \Lambda^T \Psi^{-1} \Lambda \mathbf{m}_\mathbf{s} - 2\mathbf{m}_\mathbf{s}^T \Lambda \Psi^{-1} \mathbb{E}[\boldsymbol{\theta}]\} \\ &= \arg\min_{\mathbf{m}_\mathbf{s}} \{\mathcal{O}(\mathbf{m}_\mathbf{s}) + ||\mathbf{m}_\mathbf{s}||_1\} \end{aligned} \tag{4.41}$$

which can be solved numerically by Algorithm 2 below (note that the same algorithm can be used to solve problems like lasso regression with slight modification). The solution of this optimization is sparse in $\mathbf{m}_\mathbf{s}$. This is a

nice property since we would only like to consider hidden sources for which the association with tasks are significantly supported by evidence.

Another way is to assign prior distributions over (some of) the parameters Λ, $\boldsymbol{\mu}$ and Ψ and do a point estimation (e.g., MAP over Ω):

$$\{\mathcal{Z}, \Omega\} \quad \leftarrow \quad \arg\max_{\mathcal{Z},\Omega} \{p(\Omega)p(\mathcal{Z}|\Omega)p(\mathcal{D}|\mathcal{Z})\} \tag{4.42}$$

Compared with MLE estimations, MAP has similar computational cost but is usually better behaved due to its prior distribution, which can also be thought as putting some regularization term $\log p(\Omega)$ over the parameter Ω.

4.3 Summary

In this chapter we presented the algorithms of both empirical Bayes approach and point estimation approach for the generic probabilistic model of multi-task learning. So far our choice of the inference algorithm for classification, variational method, is based on the facts that they are guaranteed to converge and efficient for high dimensional problems. Also, we focused on classification problems as regression problems can be solved in the same procedure (but are much simpler). However, there are other general possibilities like expectation propagation and sampling method which might be more accurate for small-scaled tasks. Although the presented algorithms do not solve all the task scenarios, this EM-based procedure can serve as the basis for our later algorithms of other scenarios. One advantage of the proposed probabilistic learning framework is that it provides a unified view of those algorithms. Based on specific task domains, we could use the full Bayesian approach, empirical Bayes approach or point estimation approach, and tradeoffs among those algorithms are also clear.

Chapter 5

Sparsity Models for MTL

In this chapter we focus on multi-task learning scenarios which can lead to sparse solutions, as we previously discussed in Chapter 3. In particular we focus on two types of sparsity models[1] here: model that has sparse hidden source $\mathbf{s}$ and model that has sparse linear mixing matrix Λ (e.g., sparse basis). We show that they lead to different sparse solutions by reducing the joint model complexity and improve the classification performance.

5.1 Sparsity Models

Sparsity is often observed in real applications, and it is a both theoretically and practically desirable property. From the theoretical viewpoint, sparsity can greatly reduce the model complexity; from the practical viewpoint it reduces the storage and computation, especially for high-dimensional data.

As described in Chapter 3, there are two types of sparsity models that can be achieved within our probabilistic framework:

- *sparse linear combination of basis functions:*

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \mathbf{e}_k \\ \mathbf{s}_k &\sim \prod_{h=1}^{H} \mathsf{Laplace}(0, \gamma) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{5.1}$$

[1] Our joint feature selection method in Chapter 6 can be seen as the third type of sparsity model.

where $\mathbf{s}_k$ will be sparse due to the Laplace prior if conducting point estimation to obtain $\hat{\mathbf{s}}_k$. This is essentially assuming that each prediction function $f^{(k)}$ is a sparse linear combination of basis classifiers.

- *linear combination of sparse basis functions:*

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \mathbf{e}_k \\ \mathbf{s}_k &\sim p(\mathbf{s}_k|\Phi) \\ \Lambda_{\cdot,h} &\sim \prod_{f=1}^{F} \mathsf{Laplace}(0,\gamma) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0},\Psi) \end{aligned} \tag{5.2}$$

where $\Lambda_{\cdot,h}$ denotes the h-th column of matrix Λ. In other words, we assume that each column vector of Λ follows a sparse prior distribution. By performing a point estimation $\hat{\Lambda}$, this model will lead to a set of sparse basis classifiers.

In summary, the first model is more appropriate when we believe that those task prediction functions share the same set of basis, but each one is only a combination of small number of them (relatively pure tasks); in the second model we do not put restriction on how many bases are used, but instead assume that each basis function is only represented by a few features.

5.2 Algorithms

We name the first typee of sparsity model Latent ICA (LICA), for the reason that the generative model for $\boldsymbol{\theta}_k$'s is very similar to the ICA model. For the LICA, generic algorithms presented in Chapter 4 can be directly applied by using product of Laplace distributions as the prior of $\mathbf{s}_k$'s.

Here we focus on the algorithm for the second type of sparsity, where each prediction function is a linear combination of sparse basis functions. To achieve the sparse solution for high-dimensional data like text, we will perform point estimation to obtain both $\hat{\mathbf{s}}_k$ and $\hat{\Lambda}$. Similar to the algorithms presented in Chapter 4, we need to propagate information from the known ($\mathbf{x}$'s and y's) to the unknown ($\boldsymbol{\theta}_k$'s, $\mathbf{s}_k$'s and Λ) using some iterative procedure. Note that particular attention needs to be paid to the estimation of Ψ. Essentially point estimation for the covariance matrix Ψ of $\mathbf{e}_k$'s is not

well-behaved, we instead restrict Ψ to take the form $\Psi = \lambda \mathbf{I}$ in our algorithm, and use cross-validation in an outer loop to tune the scalar parameter λ, just as people usually do in traditional single-task learning to tune the regularization coefficient. Details are given in Algorithm 3.

5.3 Experiments

We show the experimental results of our models in multi-label text classification and email anti-spam filtering. Since joint learning will be most effective when we have limited training resources, in our experiments we evaluate the model by varying the number of training instances.

The model we applied here for text classification is the one in equation (5.1) where we use Laplace distribution to model the hidden source $\mathbf{s}$. In other words, we assume that the prediction function of each task is a sparse linear combination of basis classifiers that are shared among all K tasks. In the experiments of anti-spam E-mail filtering we explicitly divide the contribution to "spam" into two components for each user: the common spam component and the user-specific spam component. As a result, the multi-task learning model we used is the model for the *"noisy tasks"* scenario with sparse priors on both the $\boldsymbol{\mu}$ and $\mathbf{e}_k$. Details of the model can be found in Chapter 3.

5.3.1 Multi-label Text Classification

With the rapid growth of the Internet in recent years, people are facing an increasingly large amount of information, with the majority stored in an electronic form. As a natural result, how to automatically and selectively obtain useful information becomes a very important research challenge. Among various types of information, textual information is arguably the most important because it has a large volume and its processing is relatively easier than other media types like audio and video so far. In the field of information retrieval, *text classification*, the task of classifying natural language documents into a pre-defined set of semantic categories, has become one of the fundamental components for organizing information.

There exists a rich literature about text classification in the past several decades, which provides valuable information about individual classification methods as well as their empirical evaluation [Yang and Liu, 1999, Zhang and

Algorithm 3 Probabilistic Model for MTL with Sparse Components

Loop until convergence:

1. Learn $\hat{\boldsymbol{\theta}}_k$ given $\hat{\Lambda}$ and $\hat{\mathbf{s}}_k$ computed in previous step (conditioned on Λ and Ψ, tasks parameters will decouple and can be estimated separately) for $k = 1, \ldots, K$:

$$\hat{\boldsymbol{\theta}_k} = \arg\max_{\boldsymbol{\theta}_k} \left\{ \sum_{i=1}^{n_k} \log p(y_i^{(k)} | \mathbf{x}_i^{(k)}, \boldsymbol{\theta}_k) + \log p(\boldsymbol{\theta}_k \mid \hat{\Lambda}, \hat{\mathbf{s}}_k) \right\} \tag{5.3}$$

   This is essentially equivalent to solving the regularized linear methods for classification/regression, and we can apply any suitable optimization algorithm such as conjugate gradient.

2. Learn $\hat{\mathbf{s}}_k$ given $\hat{\Lambda}$ and $\hat{\boldsymbol{\theta}}_k$ for $k = 1, \ldots, K$:

$$\hat{\mathbf{s}}_k = \arg\max_{\mathbf{s}_k} \Big\{ \log p(\boldsymbol{\theta}_k \mid \hat{\Lambda}, \mathbf{s}_k) \Big\} \tag{5.4}$$

3. Update $\hat{\Lambda}$ given $\hat{\boldsymbol{\theta}}_k$'s and $\hat{\mathbf{s}}_k$ $(k = 1, \ldots, K)$:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\Lambda} \left\{ \sum_{k=1}^{K} \log p(\hat{\boldsymbol{\theta}}_k \mid \Lambda \hat{\mathbf{s}}_k) + \log p(\Lambda) \right\} \\ &= \arg\min_{\Lambda} \left\{ \sum_{k=1}^{K} (\hat{\boldsymbol{\theta}}_k - \Lambda \hat{\mathbf{s}}_k)^T (\hat{\boldsymbol{\theta}}_k - \Lambda \hat{\mathbf{s}}_k) + \gamma \sum_{h=1}^{H} \sum_{f=1}^{F} |\Lambda_{f,h}| \right\} \end{aligned} \tag{5.5}$$

   where γ controls how sparse the solution $\hat{\Lambda}$ is. Plugging in the prior of Λ, it can be solved as a set of Lasso-style problems.

Oles, 2001, Zhang and Yang, 2003]. Although many of the text collections are multi-labeled by their nature, most of the existing approaches will convert the problem into a set of independent binary classification problems, one for each category. Instead, here we treat multi-label text classification as a multi-task learning problem, where each label corresponds to a task.

There are several benefits of treating the multi-label text classification as a multi-task learning problem. First of all, it is more convenient that when people label documents, they simultaneously classify the documents with respect to all the categories at hand. This can also be verified from the existing text classification collections. Second, those categories for a given data collection are often related in both semantics and statistics. Third, it is known that most of the categories in existing collections obey the Power Law distribution [Yang et al., 2003], which means that we are often facing the situation that there are very limited training resources for most of the categories. And this is blessing for multi-task learning since multi-task learning will be most effective under such situation.

For evaluation we often use the F1 measure instead of error rate due to the fact that for text classification the number of positive and negative documents are often unbalanced and thus F1 is a better measure than error rate which can be insensitive. Given a two-way contingency table

	positive	**negative**
predicted positive	A	B
predicted negative	C	D

Table 5.1: A Two-way Contingency Table

the *precision* p and *recall* r are defined as

$$\begin{aligned} p &= \frac{A}{A+B} \\ r &= \frac{A}{A+C} \end{aligned} \tag{5.6}$$

and F1 is defined based on precision and recall as

$$F1 = \frac{2pr}{p+r}. \tag{5.7}$$

Furthermore, we will also use the notation of macro-F1 and micro-F1 in our experimental results. Macro-F1 is calculated by averaging over the F1

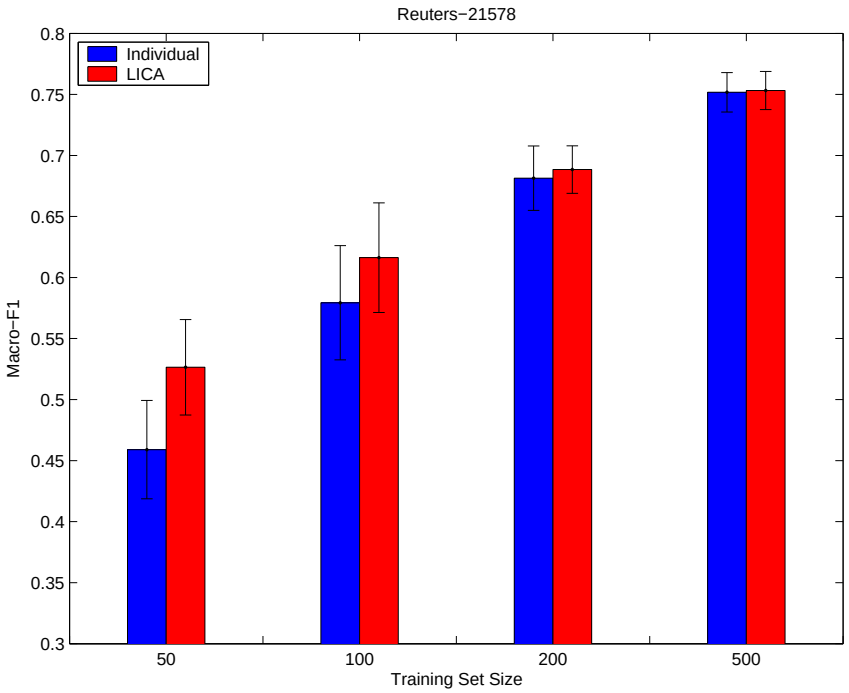

Figure 5.1: Multi-label Text Classification Results on Reuters-21578 ("Individual" refers to the STL algorithm which is the regularized logistic regression; "LICA" is our MTL algorithm)

values of all categories; while micro-F1 is the F1 value calculated by using the contingency table whose cell values are summed over all the corresponding cell values of every category's contingency table. As a result, macro-F1 will treat each category equally and thus dominated by small categories due to the Power Law category distribution; while micro-F1 will be dominated by large categories by nature.

5.3.1.1 Results on Reuters-21578

Reuters-21578 has been one of the most widely used benchmark collection for evaluating text classification algorithms in the literature [Yang and Pedersen, 1997, Yang and Liu, 1999, Zhang and Oles, 2001, Zhang and Yang, 2003]. We use a pre-processed version [Yang and Liu, 1999] which has ninety categories. Our training and test split is based on the standard ModApt split as commonly did in the literature.

Since multi-task learning will be most effective if correlations of tasks are high, we choose nine categories out of its ninety categories (those categories are *corn*, *wheat*, *grain*, *ship*, *crude*, *interest*, *money-fx*, *dlr*, *nat-gas*), which is based on the fact that those categories are often correlated by previous studies [Koller and Sahami, 1997]. In other words, the number of tasks for this data collection is $K = 9$, as we treat each category as an individual task. After stemming, stopword and rare word (words that happen less than three times) removal, we get 3,358 unique features/words. We use the empirical Bayes method in Chapter 4 to solve the problem, with Laplace priors over the hidden sources $\mathbf{s}_k$'s, and furthermore we let H, the dimensionality of hidden source $\mathbf{s}_k$, to be the same as K in this experiment. For this data collection we only report the macro-F1 results because this corpus is relatively easy and the micro-F1 results are very similar for both our model and the single-task learning algorithm (which is the regularized logistic regression classifier). Results in Figure 5.1 show that multi-task learning outperforms single task learning, especially when the amount of training resources is limited.

5.3.1.2 Results on RCV1

RCV1 is the new Reuters corpus which was intended to consist of all and only English language stories produced by Reuters journalists between August 20, 1996, and August 19, 1997. It consists of over 800,000 newswire stories that have been manually coded using three category sets. In our experiments we used the pre-processed version [Lewis et al., 2004] which is publicly available. Since there are three taxonomies for the corpus, we use the TOPIC code whose total number of categories $K = 116$ after taking into consideration some intermediate level categories suggested in [Lewis et al., 2004]. We take the standard training/test split for this collection as well. However, since the test collection is huge (more than 700k documents), we randomly select 10k as our test set in the following experiments. As in the previous experiment, we take $H = K$ in this experiment.

After some preprocessing, the total number of unique features of this data collection is $47,236$. Empirical Bayes method is not feasible here since the input space is so high-dimensional that only the memory requirement to store the covariance $\mathbb{V}[\boldsymbol{\theta}]$ is $O(F^2)$, which is clearly not affordable. Instead we take the point estimation approach, which reduces the memory requirement to $O(F)$. In Figure 5.2 the result "individual" is again obtained by using regularized logistic regression for each category individually, and our model with Laplace prior over hidden sources $\mathbf{s}_k$'s is estimated using the point

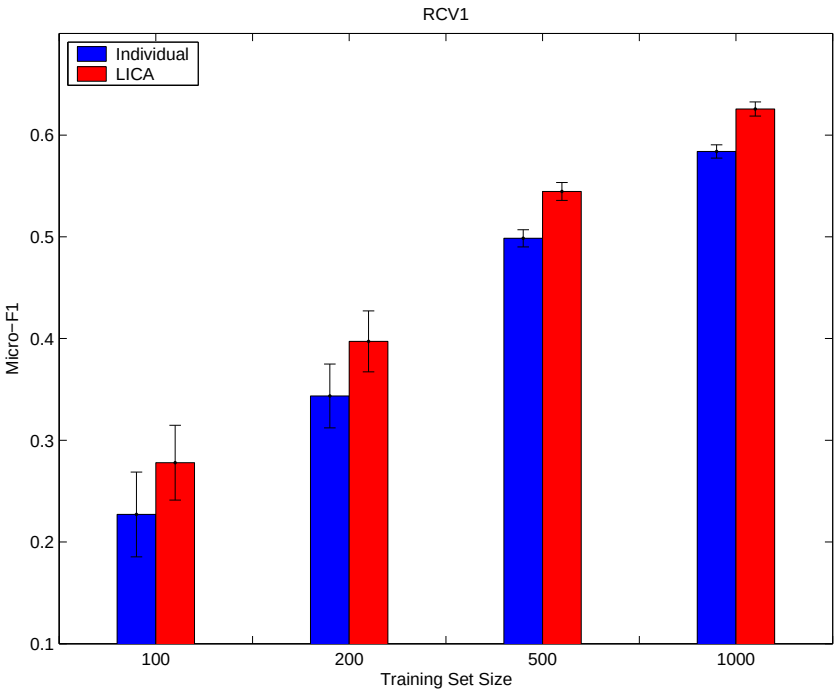

Figure 5.2: Multi-label Text Classification Results on RCV1

estimation approach. For the RCV1 collection we only report Micro-F1, and in fact we observed similar trend in Macro-F1 although values are much lower due to the large number of rare categories.

The number of non-zero $\mathbf{s}$ elements indicates how sparse the solution is, e.g., how many of the basis classifiers are actually used to form the combined classifier for each task parameter $\boldsymbol{\theta}_k$ in the joint learning framework. We take one random training set with 100 examples and count the number of non-zeros elements of $\mathbf{s}_k$ for $k = 1, \ldots, K$. It turns out that maximum number of non-zero elements is 5 (1 time), followed by 4 non-zero elements 5 times, 3 non-zero elements 76 times, 2 non-zero elements 30 times, and finally 1 non-zero element 4 times. The detailed results are shown in Table 5.2.

# of non-zero elements in $\mathbf{s}_k$	5	4	3	2	1	total
frequency	1	5	76	30	4	116

Table 5.2: Distribution of Number of Non-zero Elements in $\mathbf{s}_k$ (training size is 100)

5.3.2 Anti-Spam Filtering

Emails have become more and more important in people's daily life and the most important communication tool since the rapid growth of Internet. However, as the growth of its popularity, people are suffering from receiving "spam emails", which greatly slow down the effectiveness of emails and become quite annoying. As a result, anti-spam filtering has become a research challenge during the last several years. Anti-spam filtering can in general be treated as a binary classification problem by providing certain number of training emails - emails that are labeled "spam" or "non-spam" by users.

A simple way to build an anti-spam filtering system is to train a classifier for all users in the system based on the training data they provided, which we name as "POOLED STL". However, it is interesting to realize that users might have different definitions of what is spam based on their preference, although they do share a lot about the definition. This observation is especially important as we gather more and more judgments for a specific user, since training a separate model for that user might be beneficial. On the other hand, we usually have limited training resources for most of the users in the system and thus training a separate model for each user may not be wise especially considering the fact that users do share a lot on what is spam.

We treat the anti-spam filtering as a multi-task learning problem where each user is defined as a task. The prediction function of each task is composed of two parts: a common component and a task-specific component, as in the "*noisy tasks*" scenario in Chapter 3. This method has the advantages of both the individual learning and learning a single task using pooled data, since all training resources are used for each user's prediction while he still keeps his specific component about what is a spam.

The email corpus we used in our experiments were collected at Carnegie Mellon University. It contains personal emails from six users, and those emails were collected in around 3 months (roughly from September 2003 to November 2003 and the exact time differs across six users). Emails were originally

classified into five categories with different priorities, namely "*spam*", "*whatever*", "*keep*", "*important*", and "*very important*". Figure 5.3 shows some corpus statistics about this email collection. In our anti-spam filtering experiments we simply treat all emails labeled other than "spam" as belong to a single "non-spam" category, and thus are able to formulate a binary classification problem for this dataset.

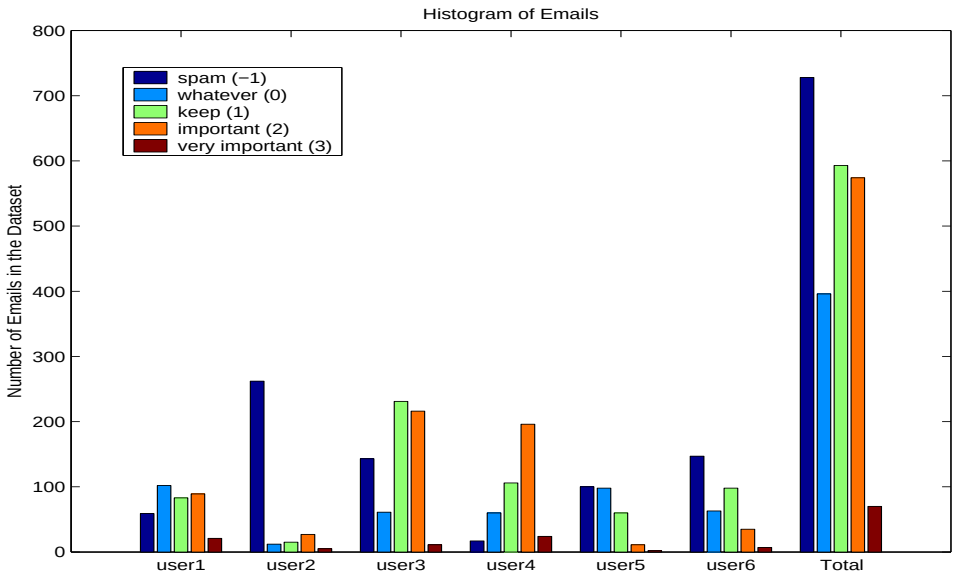

Figure 5.3: Email Corpus Statistics

From the figure we can see that even the spam populations are quite different across users, with the percentage ranging from 4.23% to 81.56% in our data set. The total number of labeled emails is around 2,300, and we randomly sample 50% as the test set[2], while the training set is sampled from the remaining 50% randomly with varying size. The experimental results are measured in F1 measure and shown in Figure 5.4.

In Figure 5.4 "MTL" refers our model for multi-task learning, "STL" refers to single task learning and is done using regularized logistic regression for each user, while "POOLED STL" is also performed by regularized logistic regression by pooling all users' training resources together. From the results we can see that our model is more effective in terms of detecting spams, due to the fact that it considers both the common factor as well as individual

[2]Here we did not consider the temporal information inside the emails for simplicity.

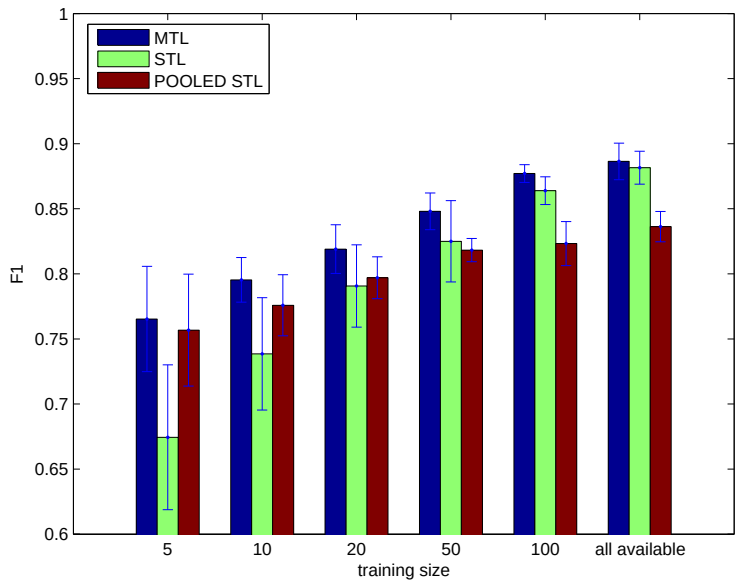

Figure 5.4: Email Anti-Spam Filtering

specific factor. On the other hand, it is not surprising to see that "STL" performs better as we get more and more training resources, while "POOLED STL" is mostly effective when the training resource is quite limited. It is important to have an anti-spam filtering system that works well for both cases, since for an email system it is unusual to have every user annotate a lot of "spam" emails, but meanwhile it is quite possible that a certain number of users could be willing to label a lot of their emails in order to achieve a better anti-spam filtering effect. The above experiment showed that with as few as six users, we are able to achieve a better system using the proposed multi-task learning framework.

It would be interesting to see what are the important features captured in the shared component and what are the features that are more effective for each individual user. To illustrate that, we rank the features based on the absolute values of their corresponding parameter values. In Table 5.3 we show the results of the feature ranking[3] for further reference. From the results we can see that all users getting spams with words like "vicodin"

[3]For privacy reasons we removed words that are related to person identification in the list.

rank	**shared**	**user1**	**user2**	**user3**	**user4**	**user5**	**user6**
1	campus	left	hi	re:	buy	title	http:
2	re:	desk	please	you	thanks	campus	thanks
3	hi	put	http:	have	question	please	re:
4	thanks	yesterday	course	time	protect	hi	qg
5	please	hi	play	vicodin	tickets	inex	please
6	you	cheers	original	do	agent	participants	free
7	vicodin	school	university	me	travel	east	kv
8	have	meeting	subject	remember	flight	garage	mwg
9	inex	student	schedule	title	http:	work	pm
10	participants	computer	available	convex	know	http:	online

Table 5.3: Informative Features for Anti-Spam Filtering as MTL (features are ranked based on their relative importance – which is measured by the absolute value of the corresponding parameter)

and "inex". User 6 is bothered about "free" and "online" things, while user 4 is probably involved with booking tickets (and those are showed as most informative features for non-spams).

5.4 Summary

We considered two types of sparsity models for multi-task learning within our framework, and conducted experiments on several text classification benchmark collections and one email corpus. The results show that our models outperform the single-task learning methods especially when the training resource for each individual task is limited, which often appears in practice. Furthermore, we verified that our models are able to achieve the claimed sparsity property.

Chapter 6

Joint Feature Selection[1]

Besides achieving better generalization performance in supervised learning problems, multi-task learning can also contribute to other important statistical machine learning problems such as feature/variable selection[2]. In this chapter we formulate the feature selection problem under the multi-task learning setting, which can be seen to naturally generalize the traditional feature selection problem in the single-task learning setting. We develop algorithms which are able to identify features that are relevant to all (or most) of the tasks and our primary goal is to show that the proposed method for multi-task feature selection can be more effective than traditional feature selection when tasks share the same subset of relevant features.

6.1 Introduction

Given a set of input variables (a.k.a. predictors, features) $X_1, X_2, \ldots, X_F$, the objective of feature/variable selection is to select a subset of features $\mathcal{R} = (r_1, r_2, \ldots, r_m) \subset \{1, 2, \ldots, F\}$ that are relevant and/or informative. Here relevance is often defined with certain applications in mind. For example, in the context of classification and regression, it usually means relevant to the response variable Y; while in an unsupervised learning setting such as

[1] An alternative name would be "Feature Selection for Multi-Task Learning".

[2] Here we use the phrase "feature selection" and "variable selection" interchangeably, as they do not differ much in our setting. Strictly speaking, feature selection is more general in the sense that each feature can possibly be a function of several variables. We assume that each input variable is a feature throughout this chapter.

density estimation, it could mean relevant to the *probability density* or *mass function* (in other words, densities change the most along relevant feature dimensions). In statistical language, most of the cases can be approximately summarized through the concept of independence:

$$\begin{aligned} \text{supervised:} \quad & f(Y|X_1, X_2, \ldots, X_F) = f(Y|X_{r_1}, X_{r_2}, \ldots, X_{r_m}) \\ \text{unsupervised:} \quad & f(X_1, X_2, \ldots, X_F) \propto f(X_{r_1}, X_{r_2}, \ldots, X_{r_m}) \end{aligned} \tag{6.1}$$

Feature selection has been an important problem in statistics [Tibshirani, 1996, Hastie et al., 2001] and machine learning [Blum and Langley, 1997, Yang and Pedersen, 1997, Liu and Setiono, 1998] for many years and is also known as "variable selection", "dimensionality reduction" in other slightly different context. Techniques for feature selection can contribute in several ways, such as:

- obtaining better predictive power
- achieving efficiency in (future) computation and storage
- providing better interpretability and scientific discovery

There have been many feature selection methods developed during the past, and they can be roughly categorized into *filter-based* methods and *wrapper-based* methods, see [Guyon and Elisseeff, 2003] for a recent survey. In the former, feature selection is done by ranking features by correlation coefficients or other criteria with respect to response variables; while in the latter subsets of features are assessed in a wrapper (such as lasso or SVM) according to their usefulness to some response variables. Generally speaking, filter-based methods treat features independently and thus are easier to conduct and more efficient; wrapper-based methods are computationally expensive but more accurate. In this chapter we will focus on wrapper-based methods since they do not assume features are independent and provide principled and elegant solutions which are often better than those offered by filter-based methods [Guyon and Elisseeff, 2003].

6.2 Outline of Feature Selection for STL

Starting with a training set

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\} \tag{6.2}$$

where $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$, the standard supervised learning problem tries to find an estimate $\hat{f}$ of the function mapping $f : \mathcal{X} \mapsto \mathcal{Y}$. Here our focus is to conduct variable selection in the original feature space, and we limit our discussion to linear prediction functions such that $\mathcal{X} = \mathbb{R}^{F\times 1}$ and $f(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$ where $\langle ., . \rangle$ denotes the *inner product* operation. It is assumed that non-linear feature mapping can be applied in the pre-processing step if the goal is to select features which are known functions of the original set of variables.

Equipped with any regularized linear method [Zhang and Oles, 2001, Zhang and Yang, 2003] as our wrapper, the above estimation problem can be converted into the following optimization problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} L(y_i, \langle \boldsymbol{\theta}, \mathbf{x}_i \rangle) + \lambda \Omega(\boldsymbol{\theta}) \right\} \tag{6.3}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{F\times 1}$ indexes the prediction function $f(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$, $\mathbf{x}_i \in \mathbb{R}^{F\times 1}$ is the i-th input data vector, $L(,.,)$ is some convex loss function for regression or classification, $\Omega(\boldsymbol{\theta})$ here is the penalty function which can be thought as a measure of the model complexity, and $\lambda \in \mathbb{R}^+$ is the regularization coefficient which controls the trade-off between the empirical loss and the model complexity. Finally note that although we mostly use the square loss

$$L(y, \langle \boldsymbol{\theta}, \mathbf{x} \rangle) = (y - \langle \boldsymbol{\theta}, \mathbf{x} \rangle)^2 \tag{6.4}$$

for regression tasks and the logistic loss

$$L(y, \langle \boldsymbol{\theta}, \mathbf{x} \rangle) = \log(1 + \exp(-y \langle \boldsymbol{\theta}, \mathbf{x} \rangle)) \tag{6.5}$$

for classification tasks in the rest of the chapter, in general other choices of convex loss functions can be easily plugged into the framework, such as the absolute error loss for regression, or the hinge loss for classification.

Ideally we would like to perform automatic model selection by selecting variables using the l_0 regularization, e.g. penalize $||\boldsymbol{\theta}||_0 = \sum_f \mathbf{1}(\theta_f \neq 0)$. Due to its intrinsic non-smoothness, the computation of l_0-norm is notorious and known to be NP-hard [Amaldi and Kann, 1998]. As a surrogate, people have used convex approximations to the l_0 regularization [Tibshirani, 1996, Weston et al., 2003].

The most popular choices of the penalty function are l_2 and l_1 regularizations: $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2^2$ and $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1$. However, it is well-known that when

$\Omega(\boldsymbol{\theta})$ takes the form of l_1 regularization, the resulting estimator $\hat{\boldsymbol{\theta}}$ will be sparse and thus achieves an effect of variable selection as a wrapper method [Tibshirani, 1996, Hastie et al., 2001]. On the other hand, l_2 regularization has the property of rotation invariance [Ng, 2004] and thus is often not recommended for the purpose of feature selection.

In particular, equation (6.3) becomes the famous lasso algorithm [Tibshirani, 1996] when $L(,.,)$ takes the form of the square loss and $\Omega(\boldsymbol{\theta})$ is set to the l_1-norm:

$$\hat{\boldsymbol{\theta}}_{lasso}(\lambda) = \arg\min_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} (y_i - \langle \boldsymbol{\theta}, \mathbf{x}_i \rangle)^2 + \lambda \sum_{f=1}^{F} |\theta_f| \right\} \tag{6.6}$$

where we explicitly emphasize that $\hat{\boldsymbol{\theta}}_{lasso}(\lambda)$ as a function of λ. We are able to achieve different degree of sparsity by varying the value of λ. Actually when $\hat{\boldsymbol{\theta}}_{lasso}(0)$ is equivalent to least-square solution, and when $\hat{\boldsymbol{\theta}}_{lasso}(\infty) = \mathbf{0}$. Another way to look at this is to notice that equation (6.6) is mathematically equivalent [Luenberger, 2003, Boyd and Vandenberghe, 2004] to

$$\begin{aligned} & \min_{\boldsymbol{\theta}} \textstyle\sum_{i=1}^{N} (y_i - \langle \boldsymbol{\theta}, \mathbf{x}_i \rangle)^2 \\ \text{subject to:} \quad & \textstyle\sum_{f=1}^{F} |\theta_f| \leq A \end{aligned} \tag{6.7}$$

where each λ corresponds to a positive value A. Geometrically, the diamond-shaped constraint in (6.3) results in the effect that many $\hat{\theta}_f$ elements may be exactly zero. Furthermore, the number of zero elements in $\hat{\boldsymbol{\theta}}_{lasso}$ will go up as we increase λ (decrease A). As a result, feature selection can be automatically conducted while conducting the optimization in equation (6.6) or (6.7).

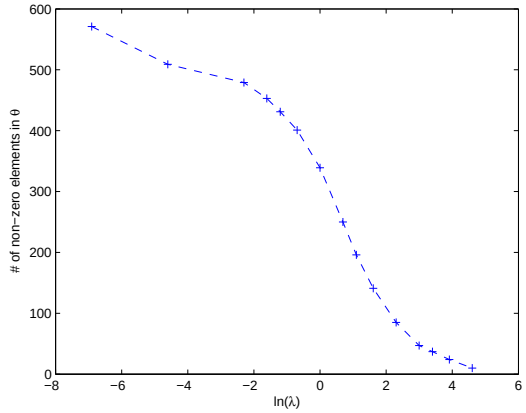

Figure 6.1: An example: # of non-zero elements versus λ

To illustrate how to conduct feature selection using the above wrapper-based method, we give an example in text classification. We use the Reuters-21578 data set [Yang and Liu, 1999] for this purpose, and the experiment is designed to classify whether documents belong to the "earn" category or not. The loss function is taken to be the logistic loss, and we use different λ values for training. Figure 6.1 plots the number of nonzero elements in $\hat{\boldsymbol{\theta}}$ versus λ. We can see that as we increase λ, we are able to achieve more sparse results. Table 6.1 lists the ranked list of remaining features when $\lambda = 100$ (there are only 10 features left in this case), where the ranking is based on the absolute value $|\theta_f|$.

Finally, note that the recently developed least angle regression (LARS) [Efron et al., 2004] is an interesting idea which can find all the solutions $\hat{\boldsymbol{\theta}}_{lasso}(\lambda)$ for all λ values very efficiently, and similar idea has been extended to the SVM method [Hastie et al., 2004].

6.3 Joint Feature Selection for MTL

Our primary interest is how to conduct effective feature selection under the multi-task learning setting, which we will also call *joint feature selection*. The key question is: Given K prediction tasks that are related, can we perform feature selection in a more effective way? The answer, of course, again depends on the underlying assumption about the relatedness of the

rank	feature	$\|\theta\|$	rank	feature	$\|\theta\|$
1	cts	2.49	*6*	dividend	0.78
2	net	1.69	*7*	earnings	0.37
3	shr	1.05	*8*	loss	0.17
4	profit	1.04	*9*	pct	0.12
5	record	0.99	*10*	company	0.07

Table 6.1: An example: selected 10 features when $\lambda = 100.0$ (data set: Reuters-21578, category: *earn*)

tasks. Here we take the most natural one: *tasks share the same subset of relevant features*. We show that when this is the case, suitable models can be designed to take that piece of information (the existence of a subset of joint relevant features) into consideration, and thus have an advantage over traditional methods which select features for each task in a separate manner.

Formally, suppose we have K prediction tasks associated with K datasets respectively:

$$\begin{aligned} \mathcal{D}^{(1)} &= \{(\mathbf{x}_1^{(1)}, y_1^{(1)}) \ldots, (\mathbf{x}_{n_1}^{(1)}, y_{n_1}^{(1)})\} \\ \vdots \quad &\vdots \quad \vdots \\ \mathcal{D}^{(K)} &= \{(\mathbf{x}_1^{(K)}, y_1^{(K)}) \ldots, (\mathbf{x}_{n_K}^{(K)}, y_{n_K}^{(K)})\} \end{aligned} \tag{6.8}$$

where $\mathbf{x}_i^{(k)} = (x_{i,1}^{(k)}, x_{i,2}^{(k)}, \ldots, x_{i,F}^{(k)})^T \in \mathbb{R}^{F \times 1}$. It is assumed that there exists a subset $\mathcal{R} = \{r_1, r_2, \ldots, r_m\} \subset \{1, 2, \ldots, F\}$ such that the functional mappings $f^{(k)}$'s can be written as

$$f^{(k)}(\mathbf{x}) = f^{(k)}(x_1, x_2, \ldots, x_F) = f^{(k)}(x_{r_1}, x_{r_2}, \ldots, x_{r_m}). \tag{6.9}$$

e.g. $p(y|\mathbf{x})$ does not depend on the irrelevant dimensions $\mathcal{I} = \{1, 2, \ldots, F\} \backslash \mathcal{R}$.

Generally speaking, joint feature selection could be useful in the following ways:

1. to more accurately identify relevant features, especially when the number of tasks is large and the number of training instances per task is small;

2. to get a more efficient joint representation across all tasks;

As a result, models for multi-task feature selection should also be evaluated accordingly.

To utilize the shared information among tasks, we can formulate the problem within the regularization learning framework as follows:

$$\hat{\boldsymbol{\theta}}_1, \dots \hat{\boldsymbol{\theta}}_K = \underset{\boldsymbol{\theta}_1,\dots,\boldsymbol{\theta}_K}{\arg\min} \left\{ \sum_{k=1}^{K} \sum_{i=1}^{N_k} L(y_i^{(k)}, \langle \boldsymbol{\theta}_k, \mathbf{x}_i^{(k)} \rangle) + \lambda \Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K) \right\} \quad (6.10)$$

where $\Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$ is some penalty function which measures the model complexity for all K functions simultaneously. More importantly, $\Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$ could impose coupling information between $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ which is essentially used to model the task relatedness.

Similar to the single-task learning case, there are several special cases of Ω. In particular, when

$$\Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K) = \sum_{k=1}^{K} \sum_{f=1}^{F} |\theta_{k,f}| \quad (6.11)$$

or

$$\Omega(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K) = \sum_{k=1}^{K} \sum_{f=1}^{F} \theta_{k,f}^2 \quad (6.12)$$

equation (6.10) decouples among K tasks and thus it is equivalent to learn each task separately with respect to l_1 or l_2 regularization. Furthermore, when l_2 regularization is taken, a more general quadratic form can be obtained by applying [Evgeniou et al., 2005]

$$\Omega(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_K) = \left(\boldsymbol{\theta}_1^T, \boldsymbol{\theta}_2^T, \dots, \boldsymbol{\theta}_K^T\right) \mathbf{D} \begin{pmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_K \end{pmatrix} \quad (6.13)$$

with properly chosen matrix $\mathbf{D} \in \mathbb{R}^{KF \times KF}$ that can be used to specify how those task parameters should be co-regularized (or equivalently, how prior knowledge about those task parameters are correlated in the Bayesian setting). Also notice that setting $\mathbf{D}$ to diagonal matrix $\lambda \mathbf{I}$ recovers the previous special case.

6.3.1 $l_1 \circ l_\infty$ Regularization

In practical applications, often only a small subset of features are relevant/informative to all K prediction tasks. We would like to obtain a sparse solution in terms of $\theta_{k,f}$'s, especially when the cardinality of the set of informative/relevant features $|\mathcal{R}| = |\{r_1, r_2, \ldots, r_m\}| = m$ is much smaller than the total number of features, e.g., $m \ll F$.

Now, for the joint feature selection problem of multi-task learning, we use the following penalty function:

$$\Omega(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \sum_{f=1}^{F} \sup_k |\theta_{k,f}|. \tag{6.14}$$

We name equation (6.14) the "$l_1 \circ l_\infty$ regularization", which comes from the fact that if we let $\boldsymbol{\Theta} \in \mathbb{R}^{F \times K}$

$$\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \ldots, \boldsymbol{\theta}_K) = \begin{pmatrix} \theta_{1,1} & \theta_{2,1} & \ldots & \theta_{K,1} \\ \theta_{1,2} & \theta_{2,2} & \ldots & \theta_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{1,F} & \theta_{2,F} & \ldots & \theta_{K,F} \end{pmatrix} \tag{6.15}$$

to be the parameter matrix, then the penalty function $\Omega(.)$ first does l_∞ regularization for each row, and then it performs l_1 regularization over the resulting elements. In the following we will also use the notation $\sup_k |\theta_{k,f}| = ||\boldsymbol{\theta}_{.,f}||_\infty$ where $\boldsymbol{\theta}_{.,f} = (\theta_{1,f}, \theta_{2,f}, \ldots, \theta_{K,f}) \in \mathbb{R}^{1 \times K}$ to represent the parameter vector of the f-th feature across all K tasks. Just like the reason why l_1 regularization leads to sparse solutions, the above formulation leads to sparse solutions across **all** the tasks.

Intuitively, if some feature is significantly relevant to at least one task, it will be selected; otherwise it is likely to be eliminated by having $\hat{\theta}_{1,f} = \ldots = \hat{\theta}_{K,f} = 0$. Similar intuition of taking the maximum value across different tasks have been used [Yang and Pedersen, 1997] in the setting of filter-based methods such as information gain [Cover and Thomas, 1991], mutual information [Cover and Thomas, 1991] and χ^2-statistics [Yang and Pedersen, 1997, Wasserman, 2005].

6.3.2 Relaxation to $l_1 \circ l_p$ Regularization

It is obvious that the assumption "all tasks share the same subset of relevant features" is restrictive. One way to relax the assumption is to assume that

each relevant feature is shared by many of the tasks, if not all. Notice that in previous model we have $\Omega(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \sum_f \sup_k |\theta_{k,f}|$, the penalization on the f-th feature is decided by $\sup_k |\theta_{k,f}|$, which in turn is contributed to by exactly one of the K tasks by taking the supremum. This could be appropriate if all tasks share the relevant feature, but less so if only some of tasks share it as relevant. On the other hand, we observe that the l_1 regularization will penalize $\sum_k |\theta_{k,f}|$ which is contributed to by every task with the essentially same weight. To remove such a restrictive assumption, we can instead let $\Omega(.)$ to be the more general form

$$\Omega(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \sum_{f=1}^{F} ||\boldsymbol{\theta}_{.,f}||_p \tag{6.16}$$

with $1 \leq p \leq \infty$, where $||\mathbf{x}||_p$ is the l_p-norm and defined as

$$||\mathbf{x}||_p = \left(\sum_i |x_i|^p \right)^{1/p}. \tag{6.17}$$

We name this regularization the $l_1 \circ l_p$ regularization. When p lies in the range $(1, \infty)$, the above formulation also considers the joint selection effect but in a less rigorous way as the $l_1 \circ l_\infty$ regularization. Finally, this formulation can be seen as a generalization of the lasso algorithm to the multi-task learning setting no matter what p's value is. To see this, note that when $K = 1$, based on equation (6.16) we have

$$\Omega(\boldsymbol{\theta}_1) = \sum_{f=1}^{F} ||\boldsymbol{\theta}_{.,f}||_p = \sum_{f=1}^{F} |\theta_{1,f}|. \tag{6.18}$$

6.3.3 Numerical Algorithm

All above methods (including our baseline, the l_1 regularized method for single-task learning) need to solve the following optimization problem

$$\begin{aligned} \hat{\boldsymbol{\theta}}_1, \ldots \hat{\boldsymbol{\theta}}_K &= \underset{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K}{\arg\min}\, \mathcal{O}(\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(K)}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) \\ &= \underset{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K}{\arg\min} \left\{ \sum_{k=1}^{K} \sum_{i=1}^{N_k} L(y_i^{(k)}, \langle \boldsymbol{\theta}_k, \mathbf{x}_i^{(k)} \rangle) + \lambda \sum_{f=1}^{F} ||\boldsymbol{\theta}_{.,f}||_p \right\} \end{aligned} \tag{6.19}$$

with $1 \leq p \leq \infty$[3]. When $p > 1$, $\boldsymbol{\theta}_k$'s $(k = 1, \ldots, K)$ will be coupled together and the optimization problem needs to be solved in a joint manner.

We take the *coordinate descent* approach by modification of the Gauss Seidel algorithm used in [Zhang and Oles, 2001], where in every step we only focus on a single variable $\theta_{k,f}$ and make sure that its change from

$$\theta_{k,f} \leftarrow \theta_{k,f} + \delta \tag{6.20}$$

will monotonically decrease the objective in equation (6.19). Having this monotonicity property is elegant as we would easily stop whenever the accuracy suffices our application (such as when the maximum value of $\delta_{k,f}$ in one iteration is less than $1e-6$, or the relative change of loss $\mathcal{O}$ is small enough). For each coordinate, a quadratic trust-region [Nocedal and Wright, 1999] is formed so that it uniformly upper bounds the Hessian of the objective around the current position. The pseudo code is listed in Algorithm 4 for reference.

6.4 Experiments

In our experiments we first illustrate the effectiveness of the proposed feature selection methods for multi-task learning by using simulated data sets. Results on $l_1 \circ l_\infty$ and $l_1 \circ l_p$ will be compared to lasso under different settings, and then we show empirically the relation between the number of tasks and effective sample size. Finally, we also evaluate its performance in terms of classification performance.

6.4.1 Results on Feature Selection

In order to verify the claimed theoretical properties of our methods, we will conduct some experiments using simulated data. One of the main purposes of our experiments is to verify that when multiple tasks share a small subset of relevant features, whether our method can more accurately select the subset of relevant features (compared to its corresponding single-task feature selection method). To conduct the experiments, we generate a dataset for regression tasks with respect to squared loss $L(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$, as shown in Figure 6.2. Note that we only assume features $1, 2, \ldots, R$ out of

[3]Due to boundary problems, we treat $p = 1$ and $p = \infty$ separately from $1 < p < \infty$ in our implementation. Both special cases can be solved more easily in similar way.

Algorithm 4 Pseudo-code for $l_1 \circ l_p$ regularized algorithm

1. Let $\boldsymbol{\theta}_k = \mathbf{0}$ $(k = 1, 2, \ldots, K)$, and loop steps 2-5 until convergence
2. Pick up a parameter $\theta_{k,f}$ (in certain order or random), and define $\mathcal{O}(\delta) = \sum_{i=1}^{N_k} L(y_i^{(k)}, \langle \boldsymbol{\theta}_k, \mathbf{x}_i^{(k)} \rangle + \delta x_{i,f}^{(k)})$ and $a = \sum_{k' \neq k} |\theta_{k',f}|^p$
3. **If** $a = 0$ (e.g. $\theta_{1,f} = \ldots = \theta_{k-1,f} = \theta_{k+1,f} = \ldots = \theta_{K,f} = 0$):
   (a) **if** $\theta_{k,f} = 0$ and $|\partial\mathcal{O}/\partial\delta| > \lambda$:
   $$\hat{\delta} = \min_{\delta} \{\mathcal{O}(\delta) + \lambda|\delta|\} \tag{6.21}$$
   (b) **if** $\theta_{k,f} \neq 0$:
   $$\hat{\delta} = \min_{\delta \in [-|\theta_{k,f}|, |\theta_{k,f}|]} \{\mathcal{O}(\delta) + \lambda|\theta_{k,f} + \delta|\} \tag{6.22}$$
4. **If** $a > 0$**:**
   $$\hat{\delta} = \min_{\delta} \left\{ \mathcal{O}(\delta) + \lambda(a + |\theta_{k,f} + \delta|^p)^{1/p} \right\} \tag{6.23}$$
5. Update $\theta_{k,f} \leftarrow \theta_{k,f} + \hat{\delta}$

Note: equations 6.21-6.23 are solved using quadratic trust-region method where the actual quadratic form (depends on the form of loss function $L(.,.)$) is taken to be the upper bound of the Hessian.

F total number of features are relevant while the rest features are just random, irrelevant noise. We compare how effective different feature selection methods are in terms of identifying this small subset of relevant features.

We evaluate methods based on the *precision* and *recall* of feature selection, similar to those used in information retrieval:

$$\begin{aligned} \mathsf{precision} &= \frac{\#\text{ of correctly predicted nonzero } \hat{\theta}_j\text{'s}}{\#\text{ of totally predicted nonzero } \hat{\theta}_j\text{'s}} \\ &= \frac{|\hat{R} \cap R|}{|\hat{R}|} \\ \mathsf{recall} &= \frac{\#\text{ of correctly predicted nonzero } \hat{\theta}_j\text{'s}}{\#\text{ of nonzero } \theta_j\text{'s}} \\ &= \frac{|\hat{R} \cap R|}{|R|} \end{aligned} \tag{6.24}$$

where $\hat{R}$ denotes the set of predicted nonzero features. Ideally we have $\mathsf{precision} = 1.0$ and $\mathsf{recall} = |\hat{R}|/|R|$ when $|\hat{R}| \leq |R|$ and $\mathsf{precision} = |R|/|\hat{R}|$ and $\mathsf{recall} = 1.0$ otherwise.

1. Set the number of relevant features be $R = 10$, the total number of features be $F = 100$, and the number of tasks be $K = 20$.

2. Generate $\mathbf{x}_i \in \mathbb{R}^{F \times 1} \sim \mathsf{Normal}(\mathbf{0}, \mathbf{I})$ for $i = 1, 2, \ldots, 100$.

3. Generate K number of tasks, for the k-th task we generate

$$\begin{aligned} \theta_0^{(k)} &\sim \mathsf{Normal}(0, 1) \\ \theta_j^{(k)} &\sim \mathsf{Normal}(0, 1), \quad j = 1, 2, \ldots, R \\ \theta_j^{(k)} &= 0, \quad j = R+1, \ldots, F. \end{aligned} \tag{6.25}$$

   That is, we only assume that $R = 10$ features out of $F = 100$ features are actually relevant.

4. Finally, for each task we generate

$$y_i^{(k)} \sim \mathsf{Normal}(\theta_0^{(k)} + \sum_{j=1}^{R} \theta_j^{(k)} x_{i,j}, 1) \tag{6.26}$$

   for $i = 1, 2, \ldots, 100$ as response variables for the regression tasks.

Figure 6.2: Generation process of a synthetic dataset

6.4.1.1 Effectiveness of $l_1 \circ l_\infty$ regularization

In our first experiment, we would like to compare our algorithm $l_1 \circ l_\infty$ to lasso (which is a special case of our algorithm when $K = 1$) which applies to each task individually. Since the numerical value of λ for lasso and the $l_1 \circ l_\infty$ regularization method are not directly comparable, we control the total number of nonzero $\hat{\theta}_j$'s by varying the regularization parameter λ. For lasso it is taken to be the average over $K = 20$ tasks; and for the $l_1 \circ l_\infty$ regularization method it is the number of nonzero $\sup_k |\theta_{k,f}|$'s. The top graph in Figure 6.3 shows the result of a typical run as we vary λ, which clearly illustrates the advantage of the $l_1 \circ l_\infty$ regularization method over lasso when the assumption holds.

As we mentioned earlier, the assumption that all tasks share the same subset of features $X_R = \{X_{r_1}, X_{r_2}, \ldots, X_{r_m}\}$ is restrictive. A more realistic assumption is that relevant features have significant overlaps across tasks. To study

the robustness of the $l_1 \circ l_\infty$ regularization and later the more general $l_1 \circ l_p$ regularization method, we re-generate a simulated data set. The generation is similar to that in Figure 6.2 except that we only assume the first $k \leq R$ features are shared among all tasks, while each task has the remaining $R-k$ relevant features randomly generated from indices $k+1, \ldots, 100$. By choosing different values of $k = 1, \ldots, 10$, we are able to measure how robust the method is with respect to the underlying assumption. The bottom graph in Figure 6.3 shows that the results of the $l_1 \circ l_\infty$ regularization method is not robust (sensitive to the assumption).

6.4.1.2 Effectiveness of $l_1 \circ l_p$ regularization

As pointed out earlier, the assumption that all tasks share the same subset of relevant features is clearly restricting. Often when we find a good application of multi-task learning, the reality is that relevant features are shared by many of the tasks. We would like to investigate how the relaxed model, the $l_1 \circ l_p$ regularization method (for general $1 < p < \infty$) performs when the assumption is violated, and compare it to the methods with $p = 1$ and $p = \infty$.

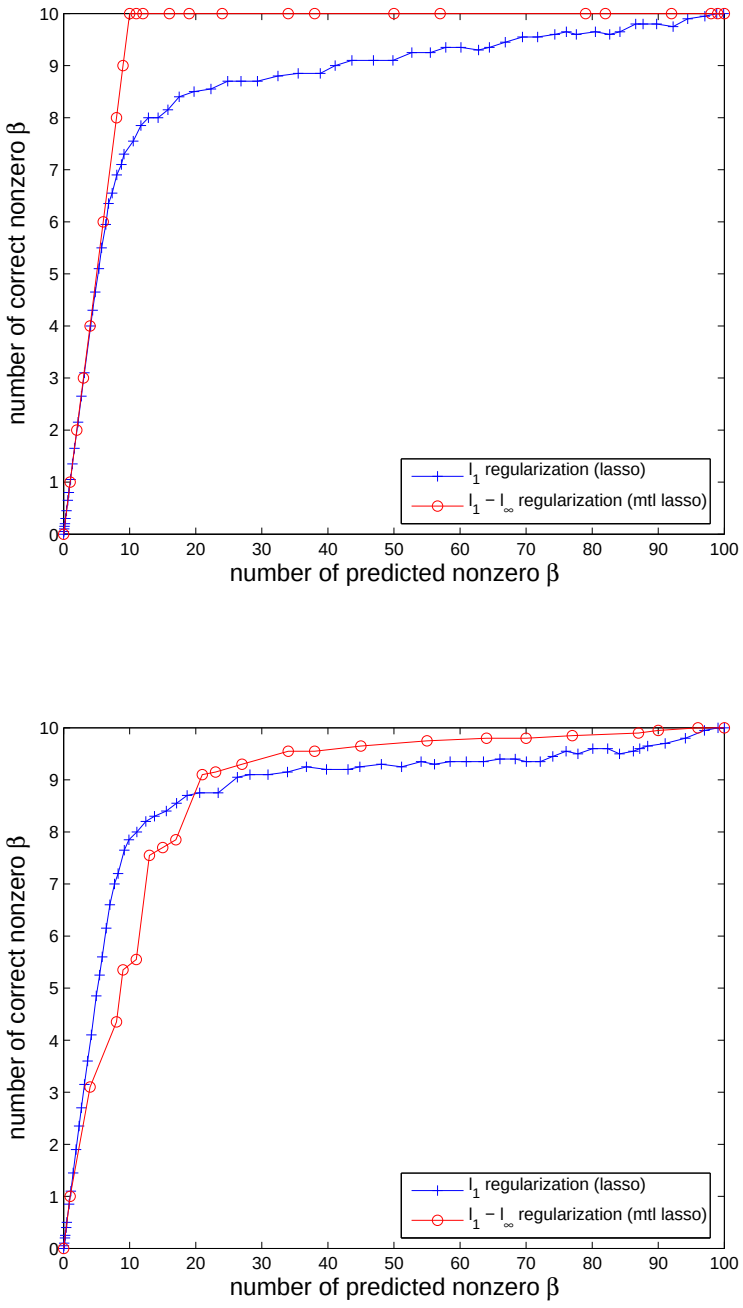

Figure 6.3: **Top:** Effectiveness of Lasso vs. MTL Lasso on Simulated Regression Dataset (This is a typical run. In the setting of $K = 20$ tasks, the $l_1 \circ l_\infty$ regularized method always correctly predicts all nonzero θ's in all 10 runs. E.g., they all have the same piecewise linear curve as shown in the above graph.); **Bottom:** effectiveness when assumption does not strictly hold ($k = 8$)

1. Set the number of relevant features be $R = 10$, the total number of features be $F = 100$, and the number of tasks be $K = 20$.

2. Generate $\mathbf{x}_i \in \mathbb{R}^{F\times 1} \sim \mathsf{Normal}(\mathbf{0}, \mathbf{I})$ for $i = 1, 2, \ldots, 100$.

3. Generate K number of tasks, for the k-th task we generate

$$\begin{aligned} \theta_0^{(k)} &\sim \mathsf{Normal}(0,1) \\ \theta_j^{(k)} &\sim r\mathsf{Normal}(0,1) + (1-r)\delta_0, \;\; j = 1, 2, \ldots, R \\ \theta_j^{(k)} &\sim \frac{R(1-r)}{100-R}\mathsf{Normal}(0,1) + \frac{100-2R+Rr}{100-R}\delta_0, \;\; j = R+1, \ldots, F. \end{aligned} \tag{6.27}$$

   The main motivation of the design is to make sure that on average we have $R = 10$ features out of $F = 100$ features are actually relevant.

4. Finally, for each task we generate

$$y_i^{(k)} \sim \mathsf{Normal}(\theta_0^{(k)} + \sum_{j=1}^{F} \theta_j^{(k)} x_{i,j}, 1) \tag{6.28}$$

   for $i = 1, 2, \ldots, 100$ as response variables for the regression tasks.

Figure 6.4: Generation process of synthetic dataset-2

To conduct such an investigation, in Figure 6.4 we use a modified algorithm of Figure 6.2 to generate the task parameters and data set. For the k-th task, $\beta_j \sim \mathsf{Normal}(0,1)$ with probability r and equals 0 with probability $1-r$, for $j = 1, \ldots, R$; $\beta_j \sim \mathsf{Normal}(0,1)$ with probability $R(1-r)/(100-R)$ and equals 0 otherwise, for $j = R+1, \ldots, F$. Clearly when $r = 1.0$ this repeats the algorithm in Figure 6.2, and when $r < 1.0$ it relaxes the assumption, but the expected number of relevant features per task still remains $R = 10$. In Figure 6.5 we generate simulated data by using $r = 0.8$ and compare l_1, $l_1 \circ l_\infty$ and $l_1 \circ l_p$ for finite p value. Results show that the $l_1 \circ l_\infty$ method suffers a lot from the fact that the assumption does not hold, while the $l_1 \circ l_p$ method is much more robust (after all, it converges to lasso as $p \to 1$).

6.4.1.3 Number of tasks vs. effective sample size

One way to understand why multi-task learning can help feature selection is to view from the aspect of effective sample size. That is, training examples

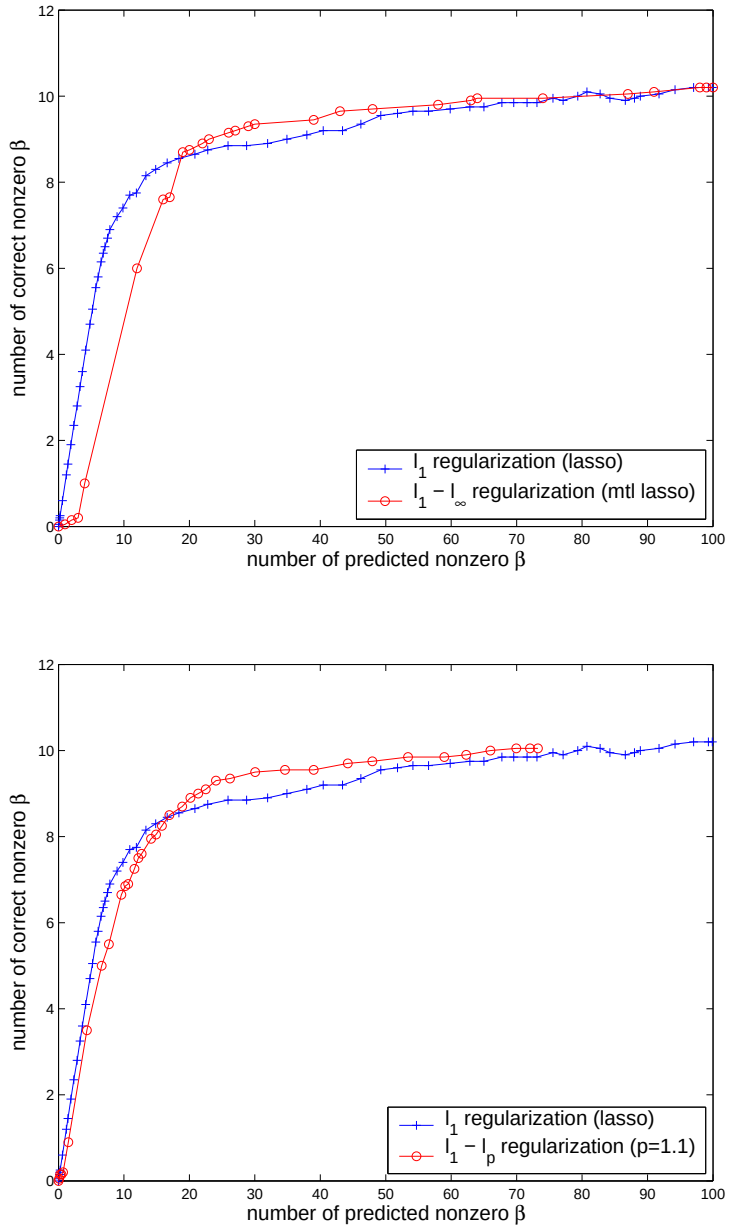

Figure 6.5: $l_1 - l_\infty$ and $l_1 - l_p$ regularization (relaxed assumption)

from other tasks can also contribute to the "effective" number of training examples for one particular task when tasks are related. Here we conduct experiments on simulated data to verify such theoretical claims. The experiment for single-task feature selection is the same as before except that we vary the number of training examples, from 100 to 1000. For multi-task feature selection we vary K - the number of tasks - from 1 to 20 instead, where each task has 100 training examples as before. Again, models are compared by requiring that they achieve the same accuracy of feature selection, and again this is achieved by tuning the regularization parameter λ for each method until they predict exactly 10 non-zero β's (that is, at the recall level 10). To make the problem more difficult, we set $F = 1000$ instead of 100, e.g. only 1% of the total features are actually relevant. Both the dataset and parameters are sampled 20 times according to Figure 6.2 and results are reported in Figure 6.6.

By comparing both graphs in Figure 6.6 we can see that when tasks are related, having additional tasks can significantly contribute to the accuracy of feature selection, in pretty much the same way as we increase the sample size for single-task learning[4]. This result further supports the effectiveness of the proposed joint feature selection method.

6.4.1.4 Summary

We conducted several experiments on several simulated datasets to show the effectiveness of the our approaches. Under the assumption that all tasks share the same set of relevant features, the proposed $l_1 \circ l_\infty$ regularization method works very well, as shown in Figure 6.2. Furthermore, the experiments in Figure 6.6 clearly indicate its strength compared to having a larger sample size in the single-task learning setting. When the assumption is violated, especially when relevant features are shared by many (but not all) tasks, we demonstrate that using the $l_1 \circ l_p$ regularization approach can work better due to its less rigid assumption.

Furthermore, we also conducted simulated experiments for classification tasks with a similar procedure described as above. Specifically, the generation of

[4]Note that the simulation results depend quantitatively on the signal/noise ratio in our experiments.

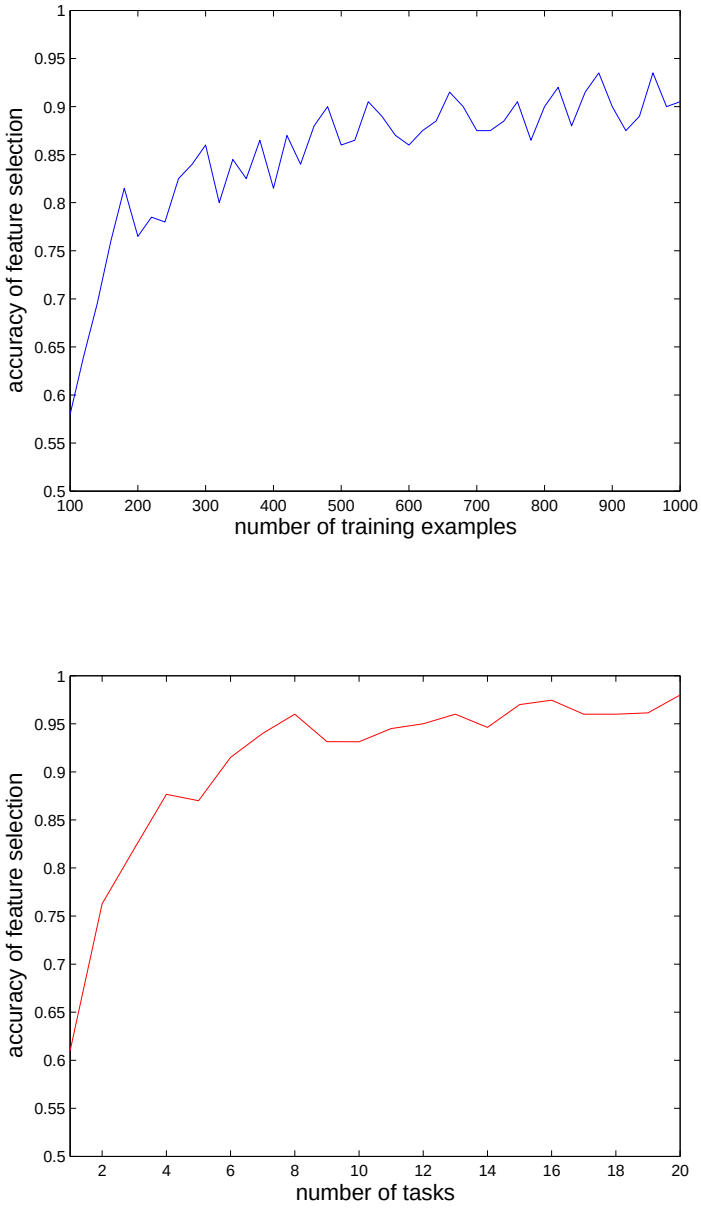

Figure 6.6: **Top**: accuracy of feature selection (at recall level 10) versus number of training examples in l_1 regularized lasso; **Bottom**: accuracy of feature selection (at recall level 10) versus number of tasks in $l_1 \circ l_\infty$ regularized method

response variables in Figure 6.2 is replaced by

$$y_i^{(k)} \sim \text{Bernoulli}\left(\mu(\theta_0^{(k)} + \sum_{j=1}^{10} \theta_j^{(k)} x_{i,j})\right) \tag{6.29}$$

and square loss is replaced by logistic loss during the learning, where $\mu(t) = (1+\exp(-t))^{-1}$ is the sigmoid function. We do not report the results for classification tasks here since they show very similar patterns to the regression case and do not provide more insights.

6.4.2 Results on Handwritten Digits Recognition

In this experiment we investigate our proposed methods for the handwritten digits recognition problem. The dataset we used is a subset of the MNIST which contains 60,000 training images and 10,000 testing images. Each digit (0-9) is represented in a matrix of 28×28 pixels. In our preprocessing we make each pixel a binary value representing white or black, and extract features based on 4×4 shaped square patterns similar to those used in [Ando and Zhang, 2004] . After preprocessing, each digit is represented as a vector with around nine thousand features.

p	*1.0*	*1.001*	*1.01*	*1.1*	*1.5*	*5.0*
error rate	0.1410	0.1238	0.1266	0.1328	0.1286	0.141

Table 6.2: Results on Handwritten Digits Recognition

Since there are 10 digits (0-9) in our experiments, we could treat it as a multi-task learning problem with $K = 10$ where each task is a binary classification problem with respect to a particular digit. In the experiments we examine the effectiveness of our methods by using small number of features (100 features in our case). Given a set of candidate features described as above, we can tune λ to select a subset of features for each learning algorithm. Results are shown in Table 6.2, which show that we can benefit in terms of predictive power by using $p > 1.0$, which in turn implies that there is certain amount of information shared (in terms of relevant features) among the ten prediction tasks we study.

6.5 Summary

In this chapter we propose a new approach for feature selection in the multi-task learning setting, where the goal is to select a joint subset of features that are relevant to multiple prediction tasks. We use a wrapper-based approach by introducing the $l_1 \circ l_\infty$ regularization that penalizes the overall model complexity and naturally imposes parameter sparsity across all tasks, and show that it can be efficiently solved by efficient convex optimization techniques. Furthermore, we also relax the assumption which leads to the discovery of a full spectrum of regularization algorithms based on the $l_1 \circ l_p$ $(1 \leq p \leq \infty)$ regularization. Our model can be thought as a generalization of the lasso algorithm to multi-task learning setting.

We conduct experiments on simulated data sets to verify the theoretical properties and the effectiveness of the proposed models. Furthermore, we demonstrate the contribution of multi-task learning to the effective sample size. The results on handwritten digit recognition problem also show the effectiveness and advantages of the proposed method over conventional single-task learning method.

Chapter 7

Mixture Models

One of the multi-task learning scenarios discussed earlier in chapter 3 is the "clusters of tasks", which is suitable for the situation where task parameters form several clusters. In this chapter we first introduce single-cluster models for multi-task learning [Yu et al., 2005] and then propose to use mixture models. The proposed method obviously generalizes the single-cluster model and has more flexibility, and it can be thought as applying a conventional mixture model to a higher level - the functional space.

7.1 Single-Cluster Models

Here by single-cluster model we mean that a uni-modal distribution (such as multivariate Gaussian) is used as the parametric family. In our multi-task learning setting this means that tasks parameters $\boldsymbol{\theta}_k$'s are fitted using a multivariate Gaussian distribution (e.g., $\boldsymbol{\theta}_k \sim \mathsf{Normal}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$).

7.1.1 Bayesian Linear Model

Given the linear predictive function $f(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle$ (assume $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{F \times 1}$), a Bayesian linear regression model assumes

$$\begin{aligned} \boldsymbol{\theta} &\sim \mathsf{Normal}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ y_i|\mathbf{x}_i &\sim \mathsf{Normal}(\langle \boldsymbol{\theta}, \mathbf{x}_i \rangle, \sigma^2) \end{aligned} \tag{7.1}$$

where the parameter $\boldsymbol{\theta}$ follows a multivariate normal prior. The probability of observing a set of i.i.d. data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ can be written as (with slight abuse of notation)

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|\boldsymbol{\theta}, \mathbf{x}_i). \tag{7.2}$$

Consequently, the posterior distribution of $\boldsymbol{\theta}$ after observing $\mathcal{D}$ can be calculated by applying the Bayes rule:

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) &= \frac{p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(y_i|\boldsymbol{\theta}, \mathbf{x}_i)}{\int p(\boldsymbol{\theta}) \prod_{i=1}^{n} p(y_i|\boldsymbol{\theta}, \mathbf{x}_i) d\boldsymbol{\theta}} \\ &= \frac{\mathsf{Normal}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \prod_{i=1}^{n} \mathsf{Normal}(y_i \mid \langle \boldsymbol{\theta}, \mathbf{x}_i \rangle, \sigma^2)}{\int [\mathsf{Normal}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \prod_{i=1}^{n} \mathsf{Normal}(y_i \mid \langle \boldsymbol{\theta}, \mathbf{x}_i \rangle, \sigma^2)] \, d\boldsymbol{\theta}} \\ &= \frac{\mathsf{Normal}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathsf{Normal}(\mathbf{y} \mid \mathbf{X}\boldsymbol{\theta}, \sigma^2 \mathbf{I})}{\int [\mathsf{Normal}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathsf{Normal}(\mathbf{y} \mid \mathbf{X}\boldsymbol{\theta}, \sigma^2 \mathbf{I})] \, d\boldsymbol{\theta}} \\ &= \mathsf{Normal}(\boldsymbol{\theta} \mid \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \end{aligned} \tag{7.3}$$

where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times F}$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^{n \times 1}$, and the derivation can be easily obtained by using the conjugacy property. In other words, $p(\boldsymbol{\theta}|\mathcal{D})$, the posterior distribution of $\boldsymbol{\theta}$ after observing the dataset $\mathcal{D}$, is still a multivariate normal distribution with mean $\tilde{\boldsymbol{\mu}}$ and covariance $\tilde{\boldsymbol{\Sigma}}$ updated as:

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}} &= \left(\boldsymbol{\Sigma}^{-1} + \frac{1}{\sigma^2} \mathbf{X}^T \mathbf{X} \right)^{-1} \\ \tilde{\boldsymbol{\mu}} &= \tilde{\boldsymbol{\Sigma}} \left(\boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \frac{1}{\sigma^2} \mathbf{X}^T \mathbf{y} \right) \end{aligned} \tag{7.4}$$

To apply this to multi-task learning, we can assume that task parameters

$$\boldsymbol{\theta}_k \sim \mathsf{Normal}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad k = 1, \ldots, K. \tag{7.5}$$

The advantage of modeling multiple tasks using this method is: when tasks form a cluster we can obtain a good estimate of their prior distribution by pulling information together from multiple tasks. Note that although it is possible to learn the prior distribution for single-task learning, it is more difficult especially when the number of parameters is large and the number of training examples is small.

7.1.2 Gaussian Process

The Bayesian linear regresssion can also be viewed in the function space, and in that case it becomes a special case of Gaussian process. The Gaussian process viewpoint does not only enhance our understand, but also have practical advantages (as discussed below). We follow the steps in [Williams, 1998]. Note that based on equation (7.1), for any $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ we have

$$\begin{aligned} \mathbb{E}(f(\mathbf{x}_i)) &= \langle \boldsymbol{\mu}, \mathbf{x}_i \rangle \\ \mathbb{C}(f(\mathbf{x}_i), f(\mathbf{x}_j)) &= \langle \mathbf{x}_i, \boldsymbol{\Sigma} \mathbf{x}_j \rangle \end{aligned} \tag{7.6}$$

So from the functional viewpoint, this is a Gaussian process over the regression functions $f \sim \mathsf{GP}(m(.), K(.,.))$ with the mean function defined as $m(\mathbf{x}) = \langle \boldsymbol{\mu}, \mathbf{x} \rangle$ and the covariance function defined as $K(\mathbf{x}_i, \mathbf{x}_j) \stackrel{\triangle}{=} \mathbf{K}_{i,j} = \langle \mathbf{x}_i, \boldsymbol{\Sigma} \mathbf{x}_j \rangle$.

Often one work with Gaussian process using the kernel representer theorem [Kimeldorf and Wahba, 1971]. Given a finite set of training instances, the estimation of the mean of the prediction function can be represented as

$$\mathbb{E}(f(\mathbf{x})) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}). \tag{7.7}$$

It can be shown that [Yu et al., 2005] the model in equation (7.1) corresponds to the following model

$$\begin{aligned} \boldsymbol{\alpha} &\sim \mathsf{Normal}(\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha) \\ \mathbf{y} &\sim \mathsf{Normal}(\mathbf{X}\mathbf{X}^T \boldsymbol{\alpha}, \sigma^2 \mathbf{I}) \end{aligned} \tag{7.8}$$

with properly chosen $\boldsymbol{\mu}_\alpha$ and $\boldsymbol{\Sigma}_\alpha$ such that $\mathbf{X}^T \boldsymbol{\mu}_\alpha = \boldsymbol{\mu}$ and $\mathbf{X}^T \boldsymbol{\Sigma}_\alpha \mathbf{X} = \boldsymbol{\Sigma}$.

There are several advantages of viewing Bayesian linear model in terms of Gaussian processes. First of all, we can apply more flexible mean function and covariance function in Gaussian processes and thus easily extend to nonlinear functions. Second, it is computationally pleasant to work with GP when the number of features is greater than the number of instances (which often happens in practice). For example, the input vector $\mathbf{x}$ in all previous derivations can be replaced by a non-linear, high-dimensional mapping $\phi(\mathbf{x})$. Even if $\phi(\mathbf{x})$ is infinite dimensional, we can still work with Gaussian process in the finite sample space as suggested by equation (7.8). In particular, equation (7.8) becomes

$$\begin{aligned} \boldsymbol{\alpha} &\sim \mathsf{Normal}(\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha) \\ \mathbf{y} &\sim \mathsf{Normal}(\mathbf{K}\boldsymbol{\alpha}, \sigma^2 \mathbf{I}) \end{aligned} \tag{7.9}$$

where $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T$ is the kernel matrix and $\mathbf{\Phi}$ is defined as $(\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n))^T$.

7.2 Mixture Models

It is well-known that a multivariate Gaussian/normal distribution can only model single-cluster distributions well. For exactly the same reason, when used for multi-task learning problems, both the Bayesian linear model and Gaussian process have the limitation that task predictive functions $f^{(k)}$'s are assumed to form a **single** cluster. In this section we propose to use mixture models for the more general scenario called "cluster of tasks" introduced in Chapter 3. Clearly, mixture models are generalizations of the single-cluster models and can handle more complicated multi-task learning cases, and the existence of many tasks makes the usage of mixture model justified and estimatable.

7.2.1 Mixture of Bayesian Linear Models

To extend the Bayesian linear model in equation (7.1), we assume that

$$\begin{aligned} \boldsymbol{\theta}^{(k)} &\sim \pi_1 \mathsf{Normal}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1) + \ldots + \pi_H \mathsf{Normal}(\boldsymbol{\mu}_H, \mathbf{\Sigma}_H) \\ \mathbf{y}^{(k)} &\sim \mathsf{Normal}(\mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \sigma^2\mathbf{I}) \end{aligned} \tag{7.10}$$

where $\pi_h \geq 0 \, (h = 1, \ldots, H)$ and $\sum_{h=1}^{H} \pi_h = 1$. $\mathbf{X}^{(k)} = (\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_{n_k}^{(k)})^T \in \mathbb{R}^{n_k \times F}$ and $\mathbf{y}^{(k)} = (y_1^{(k)}, \ldots, y_{n_k}^{(k)})^T \in \mathbb{R}^{n_k \times 1}$ are again the simplified representation of the input and output data instances for the k-th task. In other words, tasks are assumed to be generated from one of the clusters. To simplify notations, we will use the notation

$$\begin{aligned} & \mathsf{MoNormal}\left((\pi_h, \boldsymbol{\mu}_h, \mathbf{\Sigma}_h)_{h=1}^{H}\right) \\ \triangleq \; & \pi_1 \mathsf{Normal}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1) + \ldots + \pi_H \mathsf{Normal}(\boldsymbol{\mu}_H, \mathbf{\Sigma}_H) \end{aligned} \tag{7.11}$$

to represent the mixture of normal distributions.

If we know the parameters $\Omega = \{(\pi_h, \boldsymbol{\mu}_h, \mathbf{\Sigma}_h)_{h=1}^{H}, \sigma^2\}$ we can obtain the posterior distribution of $p(\boldsymbol{\theta}^{(k)} \mid \mathcal{D}^{(k)}, \Omega)$ similar to the case of Bayesian linear model (and this is essentially the E-step in the EM algorithm introduced later):

$$p(\boldsymbol{\theta}^{(k)} \mid \mathcal{D}^{(k)})$$

$$
\begin{aligned}
&= \frac{p(\boldsymbol{\theta}^{(k)})p(\mathcal{D}^{(k)} \mid \boldsymbol{\theta}^{(k)})}{\int \left[p(\boldsymbol{\theta}^{(k)})p(\mathcal{D}^{(k)} \mid \boldsymbol{\theta}^{(k)})\right] d\boldsymbol{\theta}^{(k)}} \\
&= \frac{\mathsf{MoNormal}\left(\boldsymbol{\theta}^{(k)} \mid (\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)_{h=1}^H\right) \mathsf{Normal}(\mathbf{y}^{(k)} \mid \mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \sigma^2\mathbf{I})}{\int \left[\mathsf{MoNormal}\left(\boldsymbol{\theta}^{(k)} \mid (\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)_{h=1}^H\right) \mathsf{Normal}(\mathbf{y}^{(k)} \mid \mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \sigma^2\mathbf{I})\right] d\boldsymbol{\theta}^{(k)}} \\
&= \mathsf{MoNormal}\left(\boldsymbol{\theta}^{(k)} \mid (\tilde{\pi}_h, \tilde{\boldsymbol{\mu}}_h, \tilde{\boldsymbol{\Sigma}}_h)_{h=1}^H\right) \qquad (7.12)
\end{aligned}
$$

After some math manipulation, it is not surprisingly to see that the posterior distribution of $p(\boldsymbol{\theta}^{(k)} \mid \mathcal{D}^{(k)})$ is also a mixture of normal distribution, with parameters updated as

$$
\begin{aligned}
\tilde{\boldsymbol{\Sigma}}_h^{(k)} &= \left(\boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2}\langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle\right)^{-1} \\
\tilde{\boldsymbol{\mu}}_h^{(k)} &= \tilde{\boldsymbol{\Sigma}}_h \left(\boldsymbol{\Sigma}_h^{-1}\boldsymbol{\mu}_h + \frac{1}{\sigma^2}\langle \mathbf{X}^{(k)}, \mathbf{y}^{(k)} \rangle\right) \qquad (7.13) \\
\tilde{\pi}_h^{(k)} &= \frac{c_h^{(k)}\pi_h}{\sum_{h'=1}^H c_{h'}^{(k)}\pi_{h'}}
\end{aligned}
$$

where c_h is a normalization factor defined as

$$
\begin{aligned}
& c_h^{(k)} \\
&= \int \mathsf{Normal}(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)\mathsf{Normal}(\mathbf{y}^{(k)} \mid \mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \sigma^2\mathbf{I}) d\boldsymbol{\theta}^{(k)} \\
&= \frac{|\tilde{\boldsymbol{\Sigma}}_h|^{1/2}}{(2\pi\sigma^2)^{n_k/2}|\boldsymbol{\Sigma}_h|^{1/2}} \exp\left(-\frac{\boldsymbol{\mu}_h^T\boldsymbol{\Sigma}_h^{-1}\boldsymbol{\mu}_h + \frac{1}{\sigma^2}\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)} \rangle - \tilde{\boldsymbol{\mu}}_h^T\tilde{\boldsymbol{\Sigma}}_h^{-1}\tilde{\boldsymbol{\mu}}_h}{2}\right) \qquad (7.14)
\end{aligned}
$$

The last step is obtained after some tedious calculation (see Appendix A). The above updating should be carried out for all tasks $k = 1, \ldots, K$.

We use the empirical Bayes method to learn the parameters $(\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)_{h=1}^H$ and σ^2. This will be conducted by an EM algorithm that is summarized in Algorithm 5. The details of the derivations can be found in Appendix B at the end of this chapter.

7.2.1.1 Hyper-prior smoothing

If we have a relatively small K (number of tasks) compared to H, the number of mixture components, we may overfit as the number of parameters to

Algorithm 5 EM Algorithm for Mixture of Bayesian Linear Regression

1. **Initialization**
   Given H, the number of clusters, initialize $(\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)_{h=1}^H$ and σ^2

2. **Loop until convergence**

   (a) **E-step**: for each task and each component cluster ($k = 1, \ldots, K$; $h = 1, \ldots, H$) update $\tilde{\pi}_h^{(k)}$, $\tilde{\boldsymbol{\mu}}_h^{(k)}$ and $\tilde{\boldsymbol{\Sigma}}_h^{(k)}$ as follows (appendix B for details):

$$
\begin{aligned}
\tilde{\pi}_h^{(k)} &\propto \pi_h \frac{|\tilde{\boldsymbol{\Sigma}}_h^{(k)}|^{1/2}}{|\boldsymbol{\Sigma}_h|^{1/2}} \exp\left[-\frac{\boldsymbol{\mu}_h^T \boldsymbol{\Sigma}_h^{-1} \boldsymbol{\mu}_h - \langle \tilde{\boldsymbol{\mu}}_h^{(k)}, (\tilde{\boldsymbol{\Sigma}}_h^{(k)})^{-1} \tilde{\boldsymbol{\mu}}_h^{(k)} \rangle}{2} \right] \\
\tilde{\boldsymbol{\Sigma}}_h^{(k)} &= \left(\boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right)^{-1} \\
\tilde{\boldsymbol{\mu}}_h^{(k)} &= \tilde{\boldsymbol{\Sigma}}_h^{(k)} \left(\boldsymbol{\Sigma}_h^{-1} \boldsymbol{\mu}_h + \frac{1}{\sigma^2} \langle \mathbf{X}^{(k)}, \mathbf{y}^{(k)} \rangle \right)
\end{aligned}
$$

   (b) **M-step**: for each component cluster, update π_h, $\boldsymbol{\mu}_h$, $\boldsymbol{\Sigma}_h$

$$
\begin{aligned}
\boldsymbol{\mu}_h &= \frac{\sum_{k=1}^K \tilde{\pi}_h^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}} \\
\boldsymbol{\Sigma}_h &= \frac{1}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}} \sum_{k=1}^K \tilde{\pi}_h^{(k)} \left(\tilde{\boldsymbol{\Sigma}}_h^{(k)} + (\tilde{\boldsymbol{\mu}}_h^{(k)} - \boldsymbol{\mu}_h)(\tilde{\boldsymbol{\mu}}_h^{(k)} - \boldsymbol{\mu}_h)^T \right) \\
\pi_h &= \frac{1}{K} \sum_{k=1}^K \tilde{\pi}_h^{(k)}
\end{aligned}
$$

   and

$$
\begin{aligned}
\sigma^2 &= \frac{1}{\sum_{k=1}^K n_k} \sum_{k=1}^K \Bigg[\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)} \rangle - 2 \langle \mathbf{y}^{(k)}, \mathbf{X}^{(k)} \rangle (\sum_{h=1}^H \tilde{\pi}_h^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}) \\
&+ \sum_{h=1}^H \tilde{\pi}_h^{(k)} \langle \mathbf{X}^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}, \mathbf{X}^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)} \rangle + \sum_{h=1}^H \tilde{\pi}_h^{(k)} \mathsf{Tr} \left[\langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \tilde{\boldsymbol{\Sigma}}_h^{(k)} \rangle \right] \Bigg]
\end{aligned}
$$

be estimated is large ($\boldsymbol{\mu}_h$'s and $\boldsymbol{\Sigma}_h$'s). We would associate a hyper-prior distribution $H(\boldsymbol{\mu}, \boldsymbol{\Sigma}_h)$ over $\{\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h\}_{h=1}^H$ (especially $\boldsymbol{\Sigma}_h$'s) to avoid overfitting. The main difference is in the M-step, where the maximization w.r.t. $\boldsymbol{\mu}_h$ and $\boldsymbol{\Sigma}_h$ is penalized by $\log H(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$. An even simpler method is to smooth the estimate in the M-step such as

$$\boldsymbol{\Sigma}_h = \frac{1}{\lambda + \sum_{k=1}^K \tilde{\pi}_h^{(k)}} \sum_{k=1}^K \tilde{\pi}_h^{(k)} \left(\lambda \mathbf{I} + \tilde{\boldsymbol{\Sigma}}_h^{(k)} + (\tilde{\boldsymbol{\mu}}_h^{(k)} - \boldsymbol{\mu}_h)(\tilde{\boldsymbol{\mu}}_h^{(k)} - \boldsymbol{\mu}_h)^T \right) \tag{7.15}$$

so that estimates of $\boldsymbol{\Sigma}_h$ will not be ill-behaved.

7.2.1.2 Connection to the MTL Framework

The MTL framework proposed in Chapter 3 can be easily adapted to support the mixture of Bayesian linear models. Mixture of Bayesian linear models can be achieved by assuming that each column of the matrix Λ follow a multivariate normal distribution

$$\begin{aligned} \boldsymbol{\theta}^{(k)} &= \Lambda \mathbf{s}_k \\ \mathbf{s}_k &\sim \mathsf{Multinomial}(\pi_1, \pi_2, \ldots, \pi_H) \\ \Lambda_{\cdot,h} &\sim \mathsf{Normal}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) \end{aligned} \tag{7.16}$$

together with the observation model

$$y_i^{(k)} \sim \mathsf{Normal}(\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_i^{(k)} \rangle, \sigma^2) \quad k = 1, \ldots, K; i = 1, \ldots, n_k. \tag{7.17}$$

The prior probability of $\boldsymbol{\theta}^{(k)}$ can be easily seen to be a mixture of normal distributions by summing over $\mathbf{s}_k$:

$$p(\boldsymbol{\theta}^{(k)}) = \sum_{\mathbf{s}_k} p(\mathbf{s}_k) p(\boldsymbol{\theta}^{(k)} | \mathbf{s}_k) = \sum_{h=1}^H \pi_h \mathsf{Normal}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) \tag{7.18}$$

Thus it follows that equation (7.16) and (7.17) is equivalent to the model in equation (7.10).

7.2.1.3 Mixtures with Common Covariance

Sometimes we would like to obtain simpler mixture model than the one used in equation (7.16), where each mixture component has its own mean but share a common covariance. One of the benefits of doing so is to reduce the number of parameters and prevent overfitting. The model in equation (7.16) can be simply modified to the following:

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \mathbf{e}_k \\ \mathbf{s}_k &\sim \mathsf{Multinomial}(\pi_1, \pi_2, \ldots, \pi_H) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \boldsymbol{\Sigma}) \end{aligned} \tag{7.19}$$

Geometrically, all task functions form H clusters, where they share the same covariance $\boldsymbol{\Sigma}$. Details of algorithm can be found in Appendix B.

7.2.2 Mixture of Gaussian Processes

Mixture of Gaussian processes can be seen as a particular variant of mixture of experts in [Jacobs et al., 1991]. Tresp [Tresp, 2001] introduced the mixture of Gaussian process model with different scale parameters and discussed its connection to related methods; Rasmussen and Ghahramani [Rasmussen and Ghahramani, 2002] proposed the infinite mixture of Gaussian processes whose covariance functions are learned from the data.

In the multi-task learning setting we will show that mixture of Gaussian processes can naturally handle the "clusters of tasks" scenario, where each cluster can be modeled by a Gaussian process expert with different mean and covariance function.

Under the assumption of mixture of Gaussian processes, the hierarchical model can be written as

$$\begin{aligned} f^{(k)} | \mathbf{s}_k = h &\sim \mathsf{GP}(m_h(.), K_h(.,.)) \\ \mathbf{s}_k &\sim \mathsf{Multinomial}(\pi_1, \pi_2, \ldots, \pi_H) \end{aligned} \tag{7.20}$$

which has parameters $\pi_1, \ldots, \pi_H$ as well as $m_h(.)$'s and $K_h(.,.)$'s.

Note that we are free to choose any valid covariance function, possibly different ones for different component. One flexible choice that is often used in the literature [Williams, 1998, Rasmussen and Ghahramani, 2002] takes the

following form:

$$K(\mathbf{x}_i, \mathbf{x}_j) = v_0 \exp\left(-\frac{1}{2}\sum_{f=1}^{F}\frac{(x_{i,f} - x_{j,f})^2}{w_f^2}\right) + v_1\delta(i,j) \tag{7.21}$$

with hyper-parameters v_0 controlling the signal variance, v_1 controlling the noise variance, and w_f controlling the relevance of the f-th feature in predicting the response variable.

By using the representation in equation (7.8), we can formulate the multi-task learning algorithm by learning $\boldsymbol{\mu}_{\alpha,h}$ and $\boldsymbol{\Sigma}_{\alpha,h}$ as well as π_h's. Since the derivation of the EM algorithm is similar to that of mixture of Bayesian linear models, we only give the results in Algorithm 6 .

7.3 Experiments

7.3.1 Synthetic Dataset

The purpose of this experiment is to show that if tasks are formed into clusters, our model can correctly identify those clusters as well as give an accurate estimation of the scales of the clusters.

We generate K samples $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ from a mixture of three 2-dimensional normal distributions with the following parameters: $\pi_1 = 0.5$, $\boldsymbol{\mu}_1 = (2,2)^T$, $\boldsymbol{\Sigma}_1 = ((0.5, 0.4)^T, (0.4, 0.5)^T)$; $\pi_2 = 0.3$, $\boldsymbol{\mu}_2 = (2,-2)^T$, $\boldsymbol{\Sigma}_2 = 0.5\mathbf{I}$; $\pi_3 = 0.2$, $\boldsymbol{\mu}_2 = (-2,2)^T$, $\boldsymbol{\Sigma}_3 = 0.5\mathbf{I}$. Figure 7.1 shows the probability density of this mixture distribution. For each generated parameter $\boldsymbol{\theta}_k$, we further generate an associated dataset $\{(\mathbf{x}_i^{(k)}, y_i^{(k)})_{i=1}^n\}$ such that

$$\begin{aligned} \mathbf{x}_i^{(k)} &\sim \mathsf{Normal}(\mathbf{0}, 2\mathbf{I}) \\ y_i^{(k)} &\sim \mathsf{Normal}(\langle \boldsymbol{\theta}_k, \mathbf{x}_i^{(k)} \rangle, 1). \end{aligned} \tag{7.22}$$

In our simulation, we generate $K = 100$ tasks and for each task we generate $n = 10$ pairs of data instances.

We apply our algorithm where we use $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ as the base kernel, and the number of clusters is chosen by 5-fold likelihood-based cross validation (see Chapter 8), and in this case H^* is found to be 3 which is the correct number of clusters. Figure 7.1 shows the contours of densities for

Algorithm 6 EM Algorithm for Mixture of Gaussian Processes

1. **Initialization**
   Given H, the number of clusters, initialize $(\pi_h, \boldsymbol{\mu}_{\alpha,h}, \boldsymbol{\Sigma}_{\alpha,h})_{h=1}^H$ and σ^2; specify K_h's parametric form for $h = 1, \ldots, H$

2. **Loop until convergence**

   (a) **E-step**: for each task and each component cluster ($k = 1, \ldots, K$; $h = 1, \ldots, H$) update $\tilde{\pi}_h^{(k)}$, $\tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)}$ and $\tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)}$:

$$
\begin{aligned}
\tilde{\pi}_h^{(k)} &\propto \pi_h \frac{|\tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)}|^{1/2}}{|\boldsymbol{\Sigma}_{\alpha,h}|^{1/2}} \exp\left[-\frac{\boldsymbol{\mu}_{\alpha h}^T \boldsymbol{\Sigma}_{\alpha,h}^{-1} \boldsymbol{\mu}_{\alpha,h} - \langle \tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)}, (\tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)})^{-1} \tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)} \rangle}{2} \right] \\
\tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)} &= \left(\boldsymbol{\Sigma}_{\alpha,h}^{-1} + \frac{1}{\sigma^2} \langle \mathbf{K}_h^{(k)}, \mathbf{K}_h^{(k)} \rangle \right)^{-1} \\
\tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)} &= \tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)} \left(\boldsymbol{\Sigma}_{\alpha,h}^{-1} \boldsymbol{\mu}_{\alpha,h} + \frac{1}{\sigma^2} \langle \mathbf{K}_h^{(k)}, \mathbf{y}^{(k)} \rangle \right)
\end{aligned}
$$

   where $\mathbf{K}_h^{(K)}$ is the kernel matrix between $\mathbf{X}^{(k)}$ and $\mathbf{X} = \cup_k \mathbf{X}^{(k)}$, implemented with K_h.

   (b) **M-step**: for each component cluster, update π_h, $\boldsymbol{\mu}_{\alpha,h}$, $\boldsymbol{\Sigma}_{\alpha,h}$

$$
\begin{aligned}
\boldsymbol{\mu}_{\alpha,h} &= \frac{\sum_{k=1}^K \tilde{\pi}_h^{(k)} \tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)}}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}} \\
\boldsymbol{\Sigma}_{\alpha,h} &= \frac{1}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}} \sum_{k=1}^K \tilde{\pi}_h^{(k)} \left(\tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)} + (\tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)} - \boldsymbol{\mu}_{\alpha,h})(\tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)} - \boldsymbol{\mu}_{\alpha,h})^T \right) \\
\pi_h &= \frac{1}{K} \sum_{k=1}^K \tilde{\pi}_h^{(k)}
\end{aligned}
$$

   and

$$
\begin{aligned}
\sigma^2 &= \frac{1}{\sum_{k=1}^K n_k} \sum_{k=1}^K \Bigg[\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)} \rangle - 2 \langle \mathbf{y}^{(k)}, (\sum_{h=1}^H \tilde{\pi}_h^{(k)} \mathbf{K}_h^{(k)} \tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)}) \rangle \\
&+ \sum_{h=1}^H \tilde{\pi}_h^{(k)} \langle \mathbf{K}_h^{(k)} \tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)}, \mathbf{K}_h^{(k)} \tilde{\boldsymbol{\mu}}_{\alpha,h}^{(k)} \rangle + \sum_{h=1}^H \tilde{\pi}_h^{(k)} \mathsf{Tr} \left[\langle \mathbf{K}_h^{(k)}, \mathbf{K}_h^{(k)} \tilde{\boldsymbol{\Sigma}}_{\alpha,h}^{(k)} \rangle \right] \Bigg]
\end{aligned}
$$

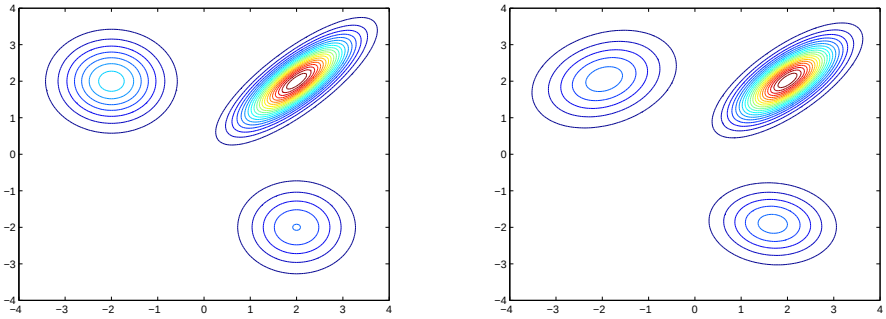

Figure 7.1: Left: mixture model ($\boldsymbol{\theta}^{(k)} \sim \mathsf{MoNormal}(\{\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h\}_{h=1}^3)$); Right: estimated density

model	SSE		model	SSE
Mixture	$\mathbf{2.53} \pm 0.19$		Ridge $\lambda = 0.1$	2.84 ± 0.18
MLE	2.89 ± 0.18		Ridge $\lambda = 0.2$	2.82 ± 0.19
Ridge $\lambda = 0.01$	2.88 ± 0.18		Ridge $\lambda = 0.5$	2.83 ± 0.21
Ridge $\lambda = 0.02$	2.88 ± 0.18		Ridge $\lambda = 1.0$	3.07 ± 0.25
Ridge $\lambda = 0.05$	2.86 ± 0.18		Ridge $\lambda = 2.0$	3.86 ± 0.28

Table 7.1: Sum of Squared Error (results are summarized over 10 random trials)

both the true density and the estimated mixture model. From the graph we can see that the estimation nicely resembles the true underlying density.

Furthermore, we also evaluate the model in terms of the Sum Squared Errors (SSE):

$$\mathsf{SSE} = \sum_{k=1}^{K} (\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k)^T (\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k). \tag{7.23}$$

For the mixture model we use the posterior mean $\boldsymbol{\theta}_{MAE}$ as the estimator, and compare the results with ridge regression estimators with parameter λ (when $\lambda = 0$ it becomes the Maximum Likelihood Estimation). Results are shown in table 7.1, which clearly shows that the SSE of mixture model is significantly better than using single-task learning algorithms.

7.3.2 Preference Prediction

Here we apply our model to the task of predicting user preferences in collaborative filtering, which has been used as a test bed for multi-task learning [Yu et al., 2006]. The major difference is that here we would like to investigate how well does mixture model perform. The dataset we use here is the MovieLens dataset[1], which contains 100,000 ratings collected from 943 users over 1,682 movies in total. Furthermore, the minimum number of rated movies by any user is 20 in this data set. Each rating is an integer score ranging from 1 to 5, with 1 meaning least favorable and 5 meaning most favorable. Furthermore, each movie in this dataset is assigned a set of genre labels. There are 19 different genres in total: *Unknown, Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western.* In the dataset each genre is given as a binary feature and they will be used to predict movie ratings for each user.

In the multi-task learning setting we treat each user as a task and each movie as a data point. Thus we have each movie $\mathbf{x} \in \mathbb{R}^{20}$ representing 19 binary features plus one bias term. Since the matrix is sparse we do not expect users to rate the same set of movies, e.g. the data instances are not shared across tasks in this case. We use the Mean Absolute Error (MAE) as our evaluation measure, as people usually do in this type of experiments. It is defined as

$$\mathsf{MAE} = \frac{1}{n}\sum_{i=1}^{n} |r(i) - \hat{r}(i)| \tag{7.24}$$

where i is the index of test pairs of (movie, user) for which we have true relevance judgment, $r(i)$ is the user's true rating, and $\hat{r}(i)$ is the predicted rating by our algorithms.

We conduct experiments by varying the number of movies known to the system (e.g. number of training examples) from 5, 10 to 20, where in each run we randomly sample 100 users (e.g., 100 tasks). First of all, we run single-task learning algorithms and report the results in Table 7.2. The algorithm we use is ridge regression (see Chapter 2), with the regularization coefficient λ chosen by cross-validation for each condition.

[1] It is available at http://www.grouplens.org/.

	optimal λ (by cross-validation)	MAE
$n_k = 5$	$\lambda^* = 0.1$	1.144 ± 0.0314
$n_k = 10$	$\lambda^* = 1.0$	1.024 ± 0.0189
$n_k = 20$	$\lambda^* = 1.0$	0.9313 ± 0.0119

Table 7.2: Results on Movie Rating (single-task learning)

We then run the mixture of Gaussian processes with base kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ and results are shown in Table 7.3. From the results we can see that using mixture models for the preference prediction problem does not provide better performance. This is a little bit surprising as before we expected to see clusters of users' predictive functions based on the genres of different movies. In other words, the results suggest that a single multivariate normal distribution does a very good job in terms of modeling the $\boldsymbol{\theta}_k$'s.

# of clusters	$n_k = 5$	$n_k = 10$	$n_k = 20$
$H = 1$	$\mathbf{0.8759} \pm 0.0129$	$\mathbf{0.8554} \pm 0.0128$	$\mathbf{0.8253} \pm 0.0060$
$H = 2$	0.8764 ± 0.0135	0.8607 ± 0.0163	0.8317 ± 0.0096
$H = 3$	0.8761 ± 0.0134	0.8638 ± 0.0179	0.8321 ± 0.0078
$H = 4$	0.8769 ± 0.0139	0.8672 ± 0.0203	0.8361 ± 0.0077
$H = 6$	0.8775 ± 0.0145	0.8682 ± 0.0205	0.8399 ± 0.0089
$H = 8$	0.8772 ± 0.0143	0.8699 ± 0.0206	0.8419 ± 0.0094
$H = 10$	0.8784 ± 0.0153	0.8732 ± 0.0228	0.8441 ± 0.0091
$H = 14$	0.8783 ± 0.0152	0.8720 ± 0.0224	0.8469 ± 0.0122
$H = 18$	0.8796 ± 0.0160	0.8739 ± 0.0225	0.8477 ± 0.0116

Table 7.3: Results on Movie Rating (base kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$)

We would like to verify our conjecture about the unimodality and normality of the task parameters $\boldsymbol{\theta}_k$'s. Procedurely we want to first obtain a set of estimated $\hat{\boldsymbol{\theta}}_k$'s, and then test their normality. Among all users who rated more than 50 movies, we randomly select 500 users. And for the k-th user we randomly select 50 movies as training set for the k-th prediction task. Now we are able to do maximum likelihood estimation on $K = 500$ tasks to obtain $\hat{\boldsymbol{\theta}}_k$'s. We would like to conduct hypothesis testing about whether $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K$ are from a multivariate normal distribution or not.

If $\hat{\boldsymbol{\theta}}_k$'s are one-dimensional we can easy conduct the test or even visualize the results. Since $\hat{\boldsymbol{\theta}}_k \in \mathbb{R}^{20}$, we instead consider its projection into

one-dimensional real line. Based on the CRAMER-WOLD theorem, a random vector $\mathbf{x}$ follows multivariate normal if and only if its every projection (into 1-dimension) follows a univariate normal distribution. Furthermore, instead of testing for any projection $\mathbf{p}$ whether $\mathbf{p}^T\hat{\boldsymbol{\theta}}_1, \ldots, \mathbf{p}^T\hat{\boldsymbol{\theta}}_K$ follows a one-dimensional normal distribution, we would first identify the most non-Gaussian direction and then perform the test (e.g. the worst case). In our experiments we use the fastICA algorithm [Hyvarinen et al., 2001] to find the most non-Gaussian projection from samples $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K$, and then verify how far it is deviate from normal distribution.

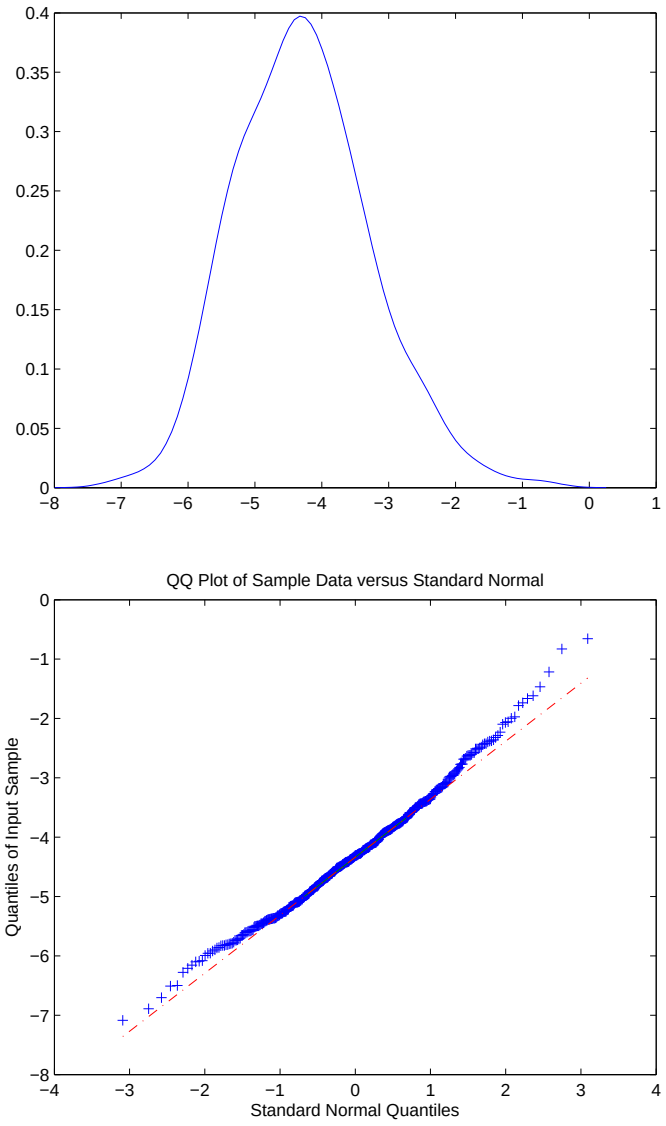

Figure 7.2: non-Gaussian projection and qq-plot of $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_K$

The top graph in Figure 7.2 shows the projected distribution of $\hat{\boldsymbol{\theta}}_k$'s, where the projection is the most non-Gaussian like projection found by the fastICA algorithm, and the density is smoothed using kernel density estimator. We can see that even the most non-Gaussian projection is not far from Gaussian.

The bottom graph shows the qq-plot[2] of the projected samples, which further verifies our previous claim.

Because of the fact that the task parameters do roughly follow a multivariate normal distribution, we try to break the distribution by using different base kernel K_h's. In this new set of experiments, we choose $K_{h=1}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. For $h > 1$ and each data pair $\mathbf{x}_i$ and $\mathbf{x}_j$, we randomly select a subset of features $I_h = \{i_1, \ldots, i_{q_h}\} \subset \{1, \ldots, F\}$ and define

$$K_h(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad \langle \tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j \rangle \tag{7.25}$$

where $\tilde{\mathbf{x}}_{i,f} = \mathbf{x}_{i,f}$ if $f \in I_h$ and otherwise 0, and the same thing is done to get $\tilde{\mathbf{x}}_{j,f}$. That is, $\tilde{\mathbf{x}}_i$ only keeps features whose index is in the random set by setting the rest of the features to zero. The way that we choose I_h is very simple: each feature is randomly chosen with probability 0.5. By using this newly defined base kernel (as we can see, for each cluster component we use a different kernel due to the random feature selection) we re-run the experiments and the results are shown in Table 7.4.

system	$n_k = 5$	$n_k = 10$	$n_k = 20$
baseline: $H = 1$	0.8821 ± 0.0149	0.8527 ± 0.0096	0.8331 ± 0.0080
mixture:H^* by CV	$\mathbf{0.8384} \pm 0.0050$	$\mathbf{0.8365} \pm 0.0057$	$\mathbf{0.8230} \pm 0.0067$

Table 7.4: Results on Movie Rating (with random kernel)

Note that H^* denotes the optimal number of clusters, which is obtained by a 5-fold likelihood-based cross-validation (see Chapter 8 for details). From the results we can see that by using randomly selected base kernels, our mixture model improved the performance of the single-cluster model in cases when the number of rated movies is small. Our way of selecting random base kernel is motivated by the idea of Random Forest by Breiman. This results illustrate that when single-cluster models are not sufficient for the task scenario, mixture models can provide more powerful representation and fit the task scenario better.

[2]Briefly speaking, qq-plot (quantile-quantile plot) is a graphical technique for determining if two datasets have the same distribution. It shows the quantile of one dataset w.r.t. the quantile of another dataset. In our case, it is the quantile of normal vs. the quantile of our data (which is the projection of $\hat{\boldsymbol{\theta}}_k$'s). If the plot does not deviate much from a straight line, then it is reasonable to accept that the data follows a normal distribution.

Appendix A: Normalization Const

Below we ignore scripts k and h:

$$\begin{aligned} & \int \mathsf{Normal}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathsf{Normal}(\mathbf{y} \mid \mathbf{X}\boldsymbol{\theta}, \sigma^2 \mathbf{I}) d\boldsymbol{\theta} \\ = & \int \frac{|2\pi\boldsymbol{\Sigma}|^{-1/2}}{|2\pi\sigma^2\mathbf{I}|^{1/2}} \exp\left(-\frac{(\boldsymbol{\theta}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu}) + \frac{1}{\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})}{2}\right) d\boldsymbol{\theta} \end{aligned}$$

Let us define

$$\begin{aligned} A &\stackrel{\triangle}{=} (\boldsymbol{\theta}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta}-\boldsymbol{\mu}) + \frac{1}{\sigma^2}(\mathbf{y}-\mathbf{X}\boldsymbol{\theta})^T(\mathbf{y}-\mathbf{X}\boldsymbol{\theta}) \\ &= \boldsymbol{\theta}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta} - 2\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\theta} + \boldsymbol{\mu}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{\sigma^2}\mathbf{y}^T\mathbf{y} - \frac{2}{\sigma^2}\mathbf{y}^T\mathbf{X}\boldsymbol{\theta} + \frac{1}{\sigma^2}\boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta} \\ &= \boldsymbol{\theta}^T(\boldsymbol{\Sigma}^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X})\boldsymbol{\theta} - 2(\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1} + \frac{1}{\sigma^2}\mathbf{y}^T\mathbf{X})\boldsymbol{\theta} + \boldsymbol{\mu}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{\sigma^2}\mathbf{y}^T\mathbf{y} \\ &= (\boldsymbol{\theta}-\tilde{\boldsymbol{\mu}})^T\tilde{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\theta}-\tilde{\boldsymbol{\mu}}) + \boldsymbol{\mu}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{\sigma^2}\mathbf{y}^T\mathbf{y} - \tilde{\boldsymbol{\mu}}^T\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}} \end{aligned}$$

where

$$\begin{aligned} \tilde{\boldsymbol{\Sigma}} &= (\boldsymbol{\Sigma}^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X})^{-1} \\ \tilde{\boldsymbol{\mu}} &= \tilde{\boldsymbol{\Sigma}}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{y}) \end{aligned}$$

As a result, we have

$$\begin{aligned} & \frac{|2\pi\boldsymbol{\Sigma}|^{-1/2}}{|2\pi\sigma^2\mathbf{I}|^{1/2}} \int \exp(-\frac{A}{2}) d\boldsymbol{\theta} \\ = & \frac{|2\pi\boldsymbol{\Sigma}|^{-1/2}}{|2\pi\sigma^2\mathbf{I}|^{1/2}} |2\pi\tilde{\boldsymbol{\Sigma}}|^{1/2} \exp\left(-\frac{\boldsymbol{\mu}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{\sigma^2}\mathbf{y}^T\mathbf{y} - \tilde{\boldsymbol{\mu}}^T\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}}{2}\right) \\ = & \frac{|\tilde{\boldsymbol{\Sigma}}|^{1/2}}{(2\pi\sigma^2)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{\boldsymbol{\mu}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \frac{1}{\sigma^2}\mathbf{y}^T\mathbf{y} - \tilde{\boldsymbol{\mu}}^T\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\mu}}}{2}\right) \end{aligned}$$

■

Appendix B: EM for Mixture of Bayesian Linear Regression

The parameters we would like to estimate are $\Omega = \{(\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)_{h=1}^H, \sigma^2\}$, and the basic idea is that we compute them by maximizing the log-likelihood $\sum_{k=1}^K \log p(\mathcal{D}^{(k)} \mid \Omega)$. Since we have two set of variables to be integrated out ($\boldsymbol{\theta}^{(k)}$'s and $Z^{(k)}$'s, where $Z^{(k)} \in \{1, \ldots, H\}$ is the indicator variable of the mixture component for the k-th task), we derive the E-step and M-step formulas from the beginning by using Jensen's inequalities.

In the following derivations we will use the notation $\mathring{\Omega}$ to denote the parameter Ω obtained in previous M-step. For the k-th task, the log-likelihood can be lower bounded as follows:

$$
\begin{aligned}
& \log p(\mathcal{D}^{(k)} \mid \Omega) \\
= \quad & \log \left(\sum_{h=1}^H p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \frac{p(Z^{(k)} = h, \mathcal{D}^{(k)} \mid \Omega)}{p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)})} \right) \\
\geq \quad & \sum_{h=1}^H p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \log \frac{p(Z^{(k)} = h, \mathcal{D}^{(k)} \mid \Omega)}{p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)})} \\
\overset{\arg\max}{=} \quad & \sum_{h=1}^H p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \log p(Z^{(k)} = h, \mathcal{D}^{(k)} \mid \Omega) \\
= \quad & \sum_{h=1}^H p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \\
& \times \quad \log \left(\int p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h) \frac{p(\boldsymbol{\theta}^{(k)}, Z^{(k)} = h, \mathcal{D}^{(k)} \mid \Omega)}{p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h)} d\boldsymbol{\theta}^{(k)} \right) \\
\geq \quad & \sum_{h=1}^H p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \\
& \times \quad \left(\int p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h) \log \frac{p(\boldsymbol{\theta}^{(k)}, Z^{(k)} = h, \mathcal{D}^{(k)} \mid \Omega)}{p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h)} d\boldsymbol{\theta}^{(k)} \right) \\
\overset{\arg\max}{=} \quad & \sum_{h=1}^H p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \\
& \times \quad \left(\int p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h) \log p(\boldsymbol{\theta}^{(k)}, Z^{(k)} = h, \mathcal{D}^{(k)} \mid \Omega) d\boldsymbol{\theta}^{(k)} \right)
\end{aligned}
$$

where the inequalities are due to Jensen inequality, and the operator $\overset{\arg\max}{=}$ means "*equivalent w.r.t. the* arg max *operation over parameters* Ω". So in the E-step we should calculate

$$\begin{aligned}
\tilde{\pi}_h^{(k)} &\triangleq p(Z^{(k)} = h \mid \mathring{\Omega}, \mathcal{D}^{(k)}) \\
&\propto \mathring{\pi}_h p(\mathcal{D}^{(k)} \mid Z^{(k)} = h, \mathring{\Omega}) \\
&= \mathring{\pi}_h p(\mathcal{D}^{(k)} \mid \mathring{\boldsymbol{\mu}}_h, \mathring{\boldsymbol{\Sigma}}_h, \mathring{\sigma}^2) \\
&= \mathring{\pi}_h \int \mathsf{Normal}(\boldsymbol{\theta}^{(k)} \mid \mathring{\boldsymbol{\mu}}_h, \mathring{\boldsymbol{\Sigma}}_h) \mathsf{Normal}(\mathbf{y}^{(k)} \mid \mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \mathring{\sigma}^2\mathbf{I}) d\boldsymbol{\theta}^{(k)} \\
&= \frac{\mathring{\pi}_h |\tilde{\boldsymbol{\Sigma}}_h^{(k)}|^{1/2}}{(2\pi\mathring{\sigma}^2)^{n_k/2} |\mathring{\boldsymbol{\Sigma}}_h|^{1/2}} \\
&\times \exp\left[-\frac{\mathring{\boldsymbol{\mu}}_h^T \mathring{\boldsymbol{\Sigma}}_h^{-1} \mathring{\boldsymbol{\mu}}_h + \frac{1}{\mathring{\sigma}^2}\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)} \rangle - \langle \tilde{\boldsymbol{\mu}}_h^{(k)}, (\tilde{\boldsymbol{\Sigma}}_h^{(k)})^{-1} \tilde{\boldsymbol{\mu}}_h^{(k)} \rangle}{2} \right] \\
&\propto \mathring{\pi}_h \frac{|\tilde{\boldsymbol{\Sigma}}_h^{(k)}|^{1/2}}{|\mathring{\boldsymbol{\Sigma}}_h|^{1/2}} \exp\left[-\frac{\mathring{\boldsymbol{\mu}}_h^T \mathring{\boldsymbol{\Sigma}}_h^{-1} \mathring{\boldsymbol{\mu}}_h - \langle \tilde{\boldsymbol{\mu}}_h^{(k)}, (\tilde{\boldsymbol{\Sigma}}_h^{(k)})^{-1} \tilde{\boldsymbol{\mu}}_h^{(k)} \rangle}{2} \right]
\end{aligned}$$

and

$$\begin{aligned}
& p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h) \\
= & \frac{p(\boldsymbol{\theta}^{(k)}, \mathcal{D}^{(k)}, Z^{(k)} = h \mid \mathring{\Omega})}{\int p(\boldsymbol{\theta}^{(k)}, \mathcal{D}^{(k)}, Z^{(k)} = h \mid \mathring{\Omega}) d\boldsymbol{\theta}} \\
= & \frac{p(Z^{(k)} = h \mid \mathring{\Omega}) p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, Z^{(k)} = h) p(\mathcal{D}^{(k)} \mid \boldsymbol{\theta}^{(k)}, \mathring{\Omega}, Z^{(k)} = h)}{\int \left(p(Z^{(k)} = h \mid \mathring{\Omega}) p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, Z^{(k)} = h) p(\mathcal{D}^{(k)} \mid \boldsymbol{\theta}^{(k)}, \mathring{\Omega}, Z^{(k)} = h) \right) d\boldsymbol{\theta}^{(k)}} \\
= & \frac{\pi_h \mathsf{Normal}(\boldsymbol{\theta}^{(k)} \mid \mathring{\boldsymbol{\mu}}_h, \mathring{\boldsymbol{\Sigma}}_h) \mathsf{Normal}(\mathbf{y}^{(k)} \mid \mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \mathring{\sigma}^2\mathbf{I})}{\int \left(\pi_h \mathsf{Normal}(\boldsymbol{\theta}^{(k)} \mid \mathring{\boldsymbol{\mu}}_h, \mathring{\boldsymbol{\Sigma}}_h) \mathsf{Normal}(\mathbf{y}^{(k)} \mid \mathbf{X}^{(k)}\boldsymbol{\theta}^{(k)}, \mathring{\sigma}^2\mathbf{I}) \right) d\boldsymbol{\theta}^{(k)}} \\
= & \mathsf{Normal}(\boldsymbol{\theta}^{(k)} \mid \tilde{\boldsymbol{\mu}}_h^{(k)}, \tilde{\boldsymbol{\Sigma}}_h^{(k)})
\end{aligned}$$

where

$$\begin{aligned}
\tilde{\boldsymbol{\Sigma}}_h^{(k)} &= \left(\mathring{\boldsymbol{\Sigma}}_h^{-1} + \frac{1}{\mathring{\sigma}^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right)^{-1} \\
\tilde{\boldsymbol{\mu}}_h^{(k)} &= \tilde{\boldsymbol{\Sigma}}_h^{(k)} \left(\mathring{\boldsymbol{\Sigma}}_h^{-1} \mathring{\boldsymbol{\mu}}_h + \frac{1}{\mathring{\sigma}^2} \langle \mathbf{X}^{(k)}, \mathbf{y}^{(k)} \rangle \right)
\end{aligned}$$

This finishes the E-step. We plug in the E-step results and define $Q^{(k)}(\Omega) \stackrel{\triangle}{=} \log p(\mathcal{D}^{(k)} \mid \Omega)$ to be

$$\begin{aligned}
Q^{(k)}(\Omega) = & \sum_{h=1}^{H} \tilde{\pi}_h^{(k)} \left(\int \mathsf{Normal}(\boldsymbol{\theta}^{(k)} \mid \tilde{\boldsymbol{\mu}}_h^{(k)}, \tilde{\boldsymbol{\Sigma}}_h^{(k)}) \log p(\boldsymbol{\theta}^{(k)}, Z^{(k)}, \mathcal{D}^{(k)} \mid \Omega) d\boldsymbol{\theta}^{(k)} \right) \\
= & \sum_{h=1}^{H} \tilde{\pi}_h^{(k)} \left\{ \log \pi_h - \frac{1}{2} \log |2\pi \boldsymbol{\Sigma}_h| - \frac{1}{2} \mathbb{E} \left[(\boldsymbol{\theta}^{(k)} - \boldsymbol{\mu}_h)^T \boldsymbol{\Sigma}_h^{-1} (\boldsymbol{\theta}^{(k)} - \boldsymbol{\mu}_h) \right] \right. \\
& \left. - \frac{1}{2} \log |2\pi\sigma^2 \mathbf{I}| - \frac{1}{2\sigma^2} \mathbb{E} \left[(\mathbf{y}^{(k)} - \mathbf{X}^{(k)} \boldsymbol{\theta}^{(k)})^T (\mathbf{y}^{(k)} - \mathbf{X}^{(k)} \boldsymbol{\theta}^{(k)}) \right] \right\} \\
= & \mathsf{C}^{(k)} + \sum_{h=1}^{H} \tilde{\pi}_h^{(k)} \log \pi_h - \frac{n_k \log \sigma^2}{2} - \frac{\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)} \rangle}{2\sigma^2} \\
& - \sum_{h=1}^{H} \frac{\tilde{\pi}_h^{(k)}}{2} \log |\boldsymbol{\Sigma}_h| - \sum_{h=1}^{H} \frac{\tilde{\pi}_h^{(k)}}{2} \boldsymbol{\mu}_h^T \boldsymbol{\Sigma}_h^{-1} \boldsymbol{\mu}_h \\
& + \sum_{h=1}^{H} \tilde{\pi}_h^{(k)} \mathbb{E} \left[\left(\boldsymbol{\mu}_h^T \boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{y}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \boldsymbol{\theta}^{(k)} \right] \\
& - \sum_{h=1}^{H} \frac{\tilde{\pi}_h^{(k)}}{2} \mathbb{E} \left[\langle \boldsymbol{\theta}^{(k)}, \left(\boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \boldsymbol{\theta}^{(k)} \rangle \right]
\end{aligned}$$

where $\mathsf{C}^{(k)} = -\frac{F+n_k}{2} \log 2\pi$ is a constant that does not depend on any of the parameters, and

$$\begin{aligned}
\mathbb{E} \left[\langle \boldsymbol{\theta}^{(k)}, \left(\boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \boldsymbol{\theta}^{(k)} \rangle \right] &= \mathsf{Tr} \left[\left(\boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \tilde{\boldsymbol{\Sigma}}_h^{(k)} \right] \\
&+ \left\langle \tilde{\boldsymbol{\mu}}_h^{(k)}, \left(\boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \tilde{\boldsymbol{\mu}}_h^{(k)} \right\rangle \\
\mathbb{E} \left[\left(\boldsymbol{\mu}_h^T \boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{y}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \boldsymbol{\theta}^{(k)} \right] &= \left(\boldsymbol{\mu}_h^T \boldsymbol{\Sigma}_h^{-1} + \frac{1}{\sigma^2} \langle \mathbf{y}^{(k)}, \mathbf{X}^{(k)} \rangle \right) \tilde{\boldsymbol{\mu}}_h^{(k)}
\end{aligned}$$

Finally, we have the log-likelihood

$$Q(\Omega) = \sum_{k=1}^{K} Q^{(k)}(\Omega).$$

To obtain the M-step, we set the partial derivatives of Q w.r.t. task-dependent parameters $(\pi_h, \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)_{h=1}^H$ to zeros (note that for $\partial Q / \partial \pi_h$ we

should add the constraint $\sum_h \pi_h = 1$ using Lagrange multiplier method):

$$\begin{aligned}
\boldsymbol{\mu}_h &= \frac{\sum_{k=1}^K \tilde{\pi}_h^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}} \\
\boldsymbol{\Sigma}_h &= \frac{1}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}} \sum_{k=1}^K \tilde{\pi}_h^{(k)} \left(\tilde{\boldsymbol{\Sigma}}_h^{(k)} + (\tilde{\boldsymbol{\mu}}_h^{(k)} - \boldsymbol{\mu}_h)(\tilde{\boldsymbol{\mu}}_h^{(k)} - \boldsymbol{\mu}_h)^T \right) \\
\pi_h &= \frac{1}{K} \sum_{k=1}^K \tilde{\pi}_h^{(k)}
\end{aligned}$$

and set the partial derivative of Q w.r.t. task-independent parameter σ^2 to zero:

$$\begin{aligned}
\sigma^2 &= \frac{1}{\sum_{k=1}^K n_k} \sum_{k=1}^K \left[\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)} \rangle - 2 \langle \mathbf{y}^{(k)}, \mathbf{X}^{(k)} \rangle (\sum_{h=1}^H \tilde{\pi}_h^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}) \right. \\
&+ \left. \sum_{h=1}^H \tilde{\pi}_h^{(k)} \langle \mathbf{X}^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}, \mathbf{X}^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)} \rangle + \sum_{h=1}^H \tilde{\pi}_h^{(k)} \mathsf{Tr} \left[\langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \tilde{\boldsymbol{\Sigma}}_h^{(k)} \rangle \right] \right]
\end{aligned}$$

EM for Mixture Models with Common Covariance

Most of the derivations are the same except that the subscripts h on $\tilde{\boldsymbol{\Sigma}}^{(k)}$, $\boldsymbol{\Sigma}$ and $\mathring{\boldsymbol{\Sigma}}$ will disappear, and the parameters to be estimated are $\Omega = \{(\pi_h, \boldsymbol{\mu}_h)_{h=1}^H, \boldsymbol{\Sigma}, \sigma^2\}$. In the E-step we compute

$$\begin{aligned}
\tilde{\pi}_h^{(k)} &\propto \mathring{\pi}_h \exp \left[-\frac{\mathring{\boldsymbol{\mu}}_h^T \mathring{\boldsymbol{\Sigma}}^{-1} \mathring{\boldsymbol{\mu}}_h - \langle \tilde{\boldsymbol{\mu}}_h^{(k)}, (\tilde{\boldsymbol{\Sigma}}^{(k)})^{-1} \tilde{\boldsymbol{\mu}}_h^{(k)} \rangle}{2} \right] \\
\tilde{\boldsymbol{\Sigma}}^{(k)} &= \left(\mathring{\boldsymbol{\Sigma}}^{-1} + \frac{1}{\mathring{\sigma}^2} \langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)} \rangle \right)^{-1} \\
\tilde{\boldsymbol{\mu}}_h^{(k)} &= \tilde{\boldsymbol{\Sigma}}^{(k)} \left(\mathring{\boldsymbol{\Sigma}}^{-1} \mathring{\boldsymbol{\mu}}_h + \frac{1}{\mathring{\sigma}^2} \langle \mathbf{X}^{(k)}, \mathbf{y}^{(k)} \rangle \right)
\end{aligned}$$

where $\tilde{\boldsymbol{\mu}}_h^{(k)}$ and $\tilde{\boldsymbol{\Sigma}}^{(k)}$ are updated mean and covariance of $p(\boldsymbol{\theta}^{(k)} \mid \mathring{\Omega}, \mathcal{D}^{(k)}, Z^{(k)} = h)$; in the M-step we compute

$$\boldsymbol{\mu}_h = \frac{\sum_{k=1}^K \tilde{\pi}_h^{(k)} \tilde{\boldsymbol{\mu}}_h^{(k)}}{\sum_{k=1}^K \tilde{\pi}_h^{(k)}}$$

$$
\begin{aligned}
\boldsymbol{\Sigma} &= \frac{1}{K}\sum_{k=1}^{K}\left(\tilde{\boldsymbol{\Sigma}}^{(k)} + \sum_{h=1}^{H}\tilde{\pi}_h^{(k)}(\boldsymbol{\mu}_h - \tilde{\boldsymbol{\mu}}_h^{(k)})(\boldsymbol{\mu}_h - \tilde{\boldsymbol{\mu}}_h^{(k)})^T\right) \\
\pi_h &= \frac{1}{K}\sum_{k=1}^{K}\tilde{\pi}_h^{(k)}
\end{aligned}
$$

and

$$
\begin{aligned}
\sigma^2 &= \frac{1}{\sum_{k=1}^{K} n_k}\sum_{k=1}^{K}\left[\langle \mathbf{y}^{(k)}, \mathbf{y}^{(k)}\rangle - 2\langle \mathbf{y}^{(k)}, \mathbf{X}^{(k)}\rangle(\sum_{h=1}^{H}\tilde{\pi}_h^{(k)}\tilde{\boldsymbol{\mu}}_h^{(k)})\right. \\
&+ \left.\sum_{h=1}^{H}\tilde{\pi}_h^{(k)}\langle \mathbf{X}^{(k)}\tilde{\boldsymbol{\mu}}_h^{(k)}, \mathbf{X}^{(k)}\tilde{\boldsymbol{\mu}}_h^{(k)}\rangle + \sum_{h=1}^{H}\tilde{\pi}_h^{(k)}\mathsf{Tr}\left[\langle \mathbf{X}^{(k)}, \mathbf{X}^{(k)}\tilde{\boldsymbol{\Sigma}}^{(k)}\rangle\right]\right]
\end{aligned}
$$

Chapter 8

Model Selection in MTL

Model selection is an important step in standard supervised and unsupervised learning in order to control model complexity and to achieve good generalization performance on future test data. For multi-task learning it also plays an important role, since we not only want to generalize well on future data of a particular task, but also want to achieve good performance on future tasks. There are two types of model complexities in multi-task learning: the model complexity of each predictive function $f^{(k)}$ and the model complexity of the joint modeling of all $f^{(k)}$'s. Since the former type of model complexity has been extensively studied in the literature [Hastie et al., 2001, Wasserman, 2005], in this chapter we focus on the investigation of the latter.

8.1 Introduction

Model selection is a common topic that exists in almost every application of machine learning. Basically speaking, it aims to find the model that has the "best" trade-off between good fit (explains the data well) and complexity (so that reliable estimation can be obtained). As stated in [Hastie et al., 2001], the goal of model selection is to estimate the performance of different models in order to choose the (approximate) best one.

A well-known statistical concept is that as the model complexity increases, the prediction error of the model typically decreases first and then increases. Figure 8.1 shows a toy example where we try to learn a one-dimensional Gaussian process model for regression with kernel $K(x, y) = \exp(-||x -$

$y||^2/2s^2$). For this model, we have high model complexity when s is small and vice versa. The data is generated according to

$$\begin{aligned} x &\sim \mathsf{Uniform}(0, 10) \\ y &\sim \mathsf{Normal}(f(x), 0.5^2) \end{aligned} \tag{8.1}$$

where the true function is set to

$$f(x) = \log(x)\sin(x^{1.2}). \tag{8.2}$$

From the top graph in Figure 8.1 we can see that as the model complexity increases, the prediction error first goes down and then goes up. "Overfitting" often refers to the situation where model complexity is more than needed (supported by the amount of availability data), and in this case the model has low bias but high variance. On the other hand, "underfitting" refers to the situation where model is not complex enough to explain the data well, and in this case the model has low variance but high bias. The bottom graph shows the true regression function and the fitted function with appropriately chosen scale/bandwidth parameter s.

Generally speaking there are many ways to do model selection [Hastie et al., 2001, Wasserman, 2005], such as AIC, BIC, cross-validation, etc. Here we focus on the cross-validation approach because it is simple to implement, easy to use and very powerful.

8.2 Cross-Validation

8.2.1 Cross-validation for STL

K_{cv}-fold[1] cross-validation is a procedure defined as the following. Given a training set $\mathcal{D}$, first split it into K_{cv} equal-sized subsets $\mathcal{D}_1, \ldots, \mathcal{D}_{K_{cv}}$, then estimate the extra sample loss [Hastie et al., 2001] on each subset using the remaining $K_{cv} - 1$ subsets as the training data, and average over all the subsets to obtain the CV score. The final CV score can be written as

$$\mathsf{CV}(\theta) = \frac{1}{n}\sum_{i=1}^{N} L(y_i, f^{\setminus \mathcal{D}(i)}(\mathbf{x}_i|\theta)) \tag{8.3}$$

[1] Because K is reserved for the number of tasks, we use K_{cv} to denote the number of folds in cross-validation.

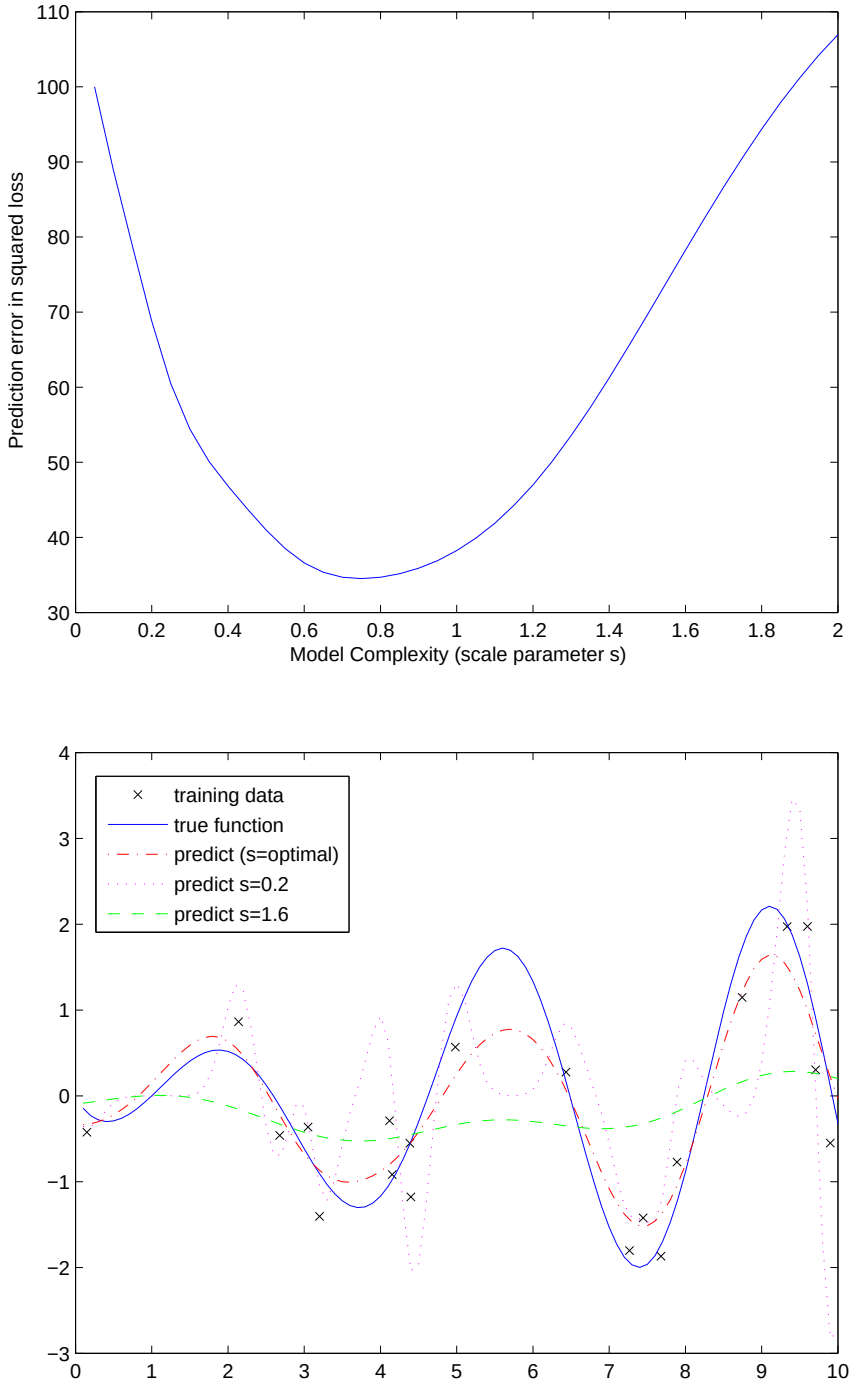

Figure 8.1: Top: model complexity vs. prediction error (the smaller the scale parameter s, the more complex the model); Bottom: true function vs. fitted function (with appropriately chosen bandwidth s)

where $f^{\setminus\mathcal{D}(i)}(.)$ denotes the function that is estimated using all data in $\mathcal{D}$ except the subset that contains the i-th data point. Note that if the prediction function $f(.)$ is indexed by its parameter $\theta \in \Theta$, then the model selection problem is to find the best parameter that gives the lowest score $\hat{\theta} = \arg\min_\theta \mathsf{CV}(\theta)$.

Typical choices of K_{cv} include $5, 10$ and N. It is well-known that when K_{cv} is large, the cross-validation score can be a low bias but high variance estimator of the true prediction loss; while on the other hand when K_{cv} is small, we get a high bias but low variance estimation. When K_{cv} equals N it is also known as Leave One Out Cross-Validation (LOOCV). Although cross-validation methods are extremely simple, they are theoretically justified. For example, it can be shown that the LOOCV score is almost an unbiased estimation for the true prediction error.

Based on the the choice of the loss $L(.,.)$ we can have several variants of cross-validation methods, and common choices include negative log-likelihood and prediction error:

- *cross-validation by likelihood*: when $L(y, f(\mathbf{x}))$ takes the form of negative log-likelihood, e.g.

$$L(y, f(\mathbf{x})) \quad = \quad -\log p(y|\mathbf{x}, \boldsymbol{\theta}). \tag{8.4}$$

  In order to use this method we need to have a probabilistic model for the response variable y conditioned on $\mathbf{x}$, and thus it may be sensitive to the model assumption.

- *cross-validation by prediction error*: typically we use

$$\begin{aligned} \mathsf{regression}: \; L(y, f(\mathbf{x})) &= (y - f(\mathbf{x}))^2 \\ \mathsf{classification}: \; L(y, f(\mathbf{x})) &= I(yf(\mathbf{x}) < 0) \end{aligned} \tag{8.5}$$

  where $I(.)$ equals 1 if the argument is true and 0 otherwise. The benefit of this method is that the error measures do not need to be dependent on the model assumptions. For example, even if the response variable is assumed to be corrupted with Gaussian noise, we can still use the absolute error $|y - f(\mathbf{x})|$ as the choice of the loss function if it makes sense for the application.

8.2.2 Cross-validation for MTL

Applying cross-validation to multi-task learning is straightforward. The only difference from its conventional usage is that we apply it to the task-level instead of data-level. Given K tasks with their associated training datasets, we split the tasks into K_{cv} folds randomly such that: $T_1 \cup T_2 \cup \ldots \cup T_{K_{cv}} = \{1, 2, \ldots, K\}$. Again we can have two choices for the CV loss function:

- *cross-validation by likelihood*: The c-th iteration of the cross-validation involves three steps: (1) a generative model $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is fitted using the (K_{cv}-1) folds' tasks $T_1, \ldots, T_{c-1}, T_{c+1}, \ldots, T_{K_{cv}}$ by the MTL algorithm; (2) for each task in the validation fold T_c, a single-task learning algorithm is used to obtain point estimations $\hat{\boldsymbol{\theta}}_k$'s; (3) compute the negative log-likelihood $-\log \hat{p}^{\setminus c}(\hat{\boldsymbol{\theta}}_k)$ for $k \in T_c$. The final score is computed as:

$$\mathsf{CV} \quad = \quad \sum_{c=1}^{K_{cv}} \sum_{k \in T_c} -\log \hat{p}^{\setminus c}(\hat{\boldsymbol{\theta}}_k). \tag{8.6}$$

- *cross-validation by prediction error*: For the c-th iteration of the cross-validation: (1) a generative model $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is fitted using the (K_{cv}-1) folds' tasks ($T_1, \ldots, T_{c-1}, T_{c+1}, \ldots, T_{K_{cv}}$); (2) for all tasks in the rest fold (validation fold), prior $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is evaluated in each task, where the evaluation is conducted with another error-based cross-validation at the data-level. The final objective can be summarized as:

$$\mathsf{CV} \quad = \quad \sum_{c=1}^{K_{cv}} \sum_{k \in T_c} \mathsf{CV}_k(\hat{p}^{\setminus c}(\boldsymbol{\theta})) \tag{8.7}$$

  where $\mathsf{CV}_k(\hat{p}^{\setminus c}(\boldsymbol{\theta}))$ is the error-based cross-validation score (like the one defined in equation (8.3)) obtained by using $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ as the prior of the $\boldsymbol{\theta}$ for the k-th task, and the cross-validation is done by splitting the training set $\mathcal{D}^{(k)}$ for the k-th task.

We can see that in order to conduct cross-validation at the task level, we need a model[2] to measure the closeness of the tasks (often in terms of their parameters $\boldsymbol{\theta}_k$'s). Also the latter one is computationally more expensive since another inner-loop cross-validation needs to be done to obtain the score.

[2] Although the model need not be probabilistic, having probabilistic model over $\boldsymbol{\theta}$ is a natural choice.

8.3 Experiments

We conduct simulations to illustrate the use of the previously described cross-validation methods. For simplicity we focus on the mixture model presented in Chapter 7. We use mixture models to generate the parameters $\boldsymbol{\theta}$'s of prediction functions, in which the true number of clusters varies from 1 to 8. For each mixture model we generate 100 tasks $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{100}$ from a prior distribution

$$\boldsymbol{\theta}_k \sim \mathsf{MoNormal}(\{\pi_h, \mathbf{m}_h, \mathbf{V}_h\}_{h=1}^{H}). \tag{8.8}$$

The parameters π_h, $\mathbf{m}_h$ and $\mathbf{V}_h$ of the mixture model are randomly generated as follows:

$$\begin{aligned} \pi_h &\propto 0.3 + \mathsf{Uniform}(0,1) \\ \mathbf{m}_h &\sim \mathsf{Uniform}\left(\begin{bmatrix} -6 \\ -6 \end{bmatrix}, \begin{bmatrix} 6 \\ 6 \end{bmatrix}\right) \\ \mathbf{V}_h &\sim \frac{1}{19}\mathsf{Wishart}(\mathbf{I}, 20). \end{aligned} \tag{8.9}$$

Finally, for each task we generate 10 training examples and 100 test examples using

$$\begin{aligned} \mathbf{x}_i^{(k)} &\sim \mathsf{Normal}(\mathbf{0}, \mathbf{I}) \\ y_i^{(k)} &\sim \mathsf{Normal}(\langle \boldsymbol{\theta}_k, \mathbf{x}_i^{(k)} \rangle, \sigma^2) \end{aligned} \tag{8.10}$$

where we simply use $\sigma^2 = 1.0$.

In our experiments we create 6 generative models for $\boldsymbol{\theta}_k$'s with the number of clusters taken to be $H = 1, 2, 3, 4, 6, 8$ respectively. Figure 8.2 shows one sample of the generative models we used for the 6 cases. We repeat the simulation process 20 times, which results in $20*6 = 120$ runs of our mixture model algorithm.

We evaluate the results using the Mean Squared Error (MSE) measure with the following notation:

- $\mathsf{MSE}(\hat{f}_{\hat{H}})$: MSE for the mixture model where the number of clusters $\hat{H}$ is chosen by cross-validation;
- $\mathsf{MSE}(\hat{f}_H)$: MSE for the mixture model where the true number of clusters H is given;

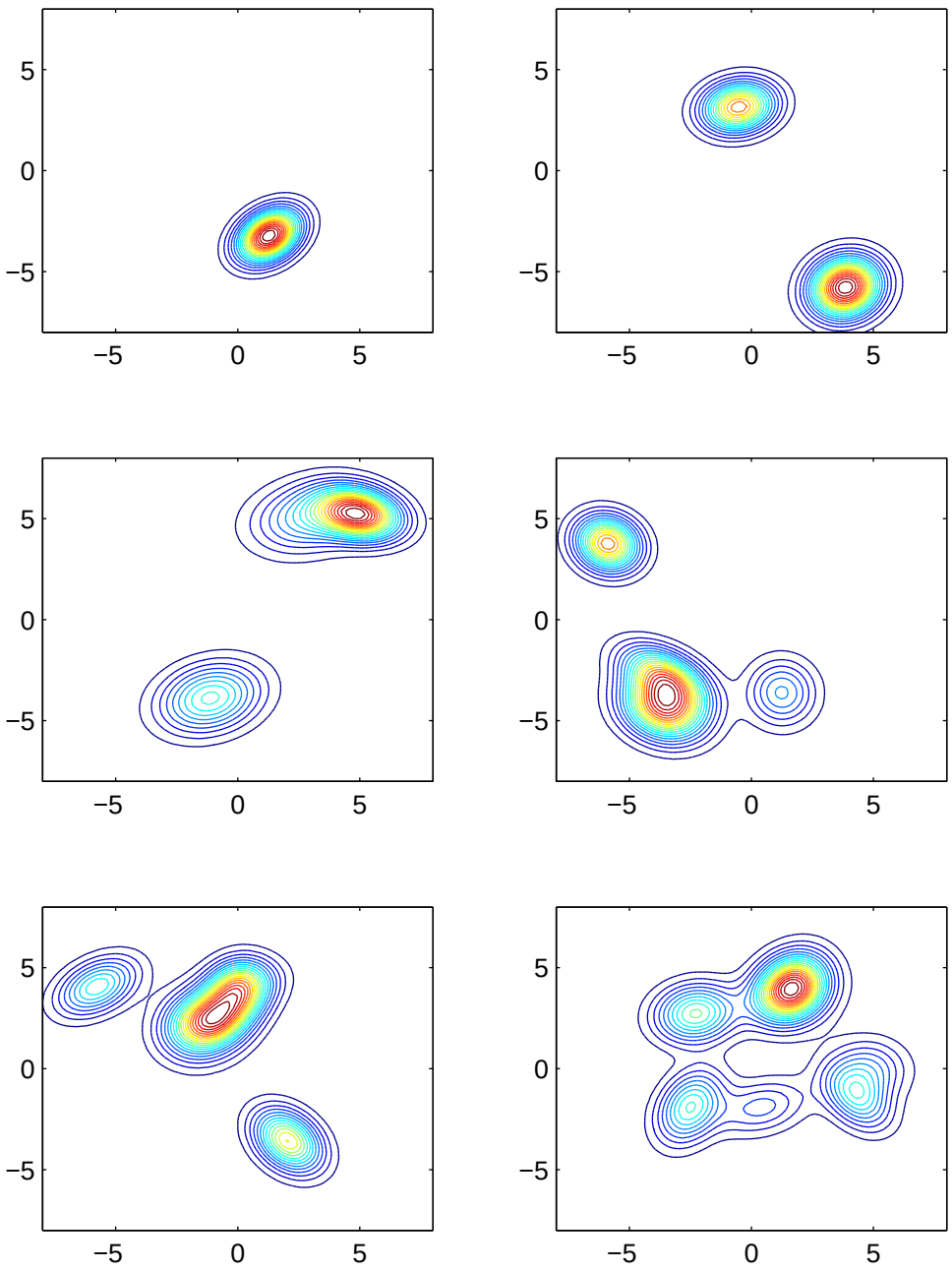

Figure 8.2: Example densities of generative models for $\boldsymbol{\theta}_k$'s; H equals 1,2,3,4,6,8 from top to bottom, left to right

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.0011 ± 0.0033	1.0028 ± 0.0029	1.0400 ± 0.0253
2	1.0000 ± 0.0064	1.0099 ± 0.0105	1.0357 ± 0.0215
3	0.9984 ± 0.0105	1.0084 ± 0.0079	1.0327 ± 0.0133
4	1.0025 ± 0.0080	1.0120 ± 0.0095	1.0321 ± 0.0188
6	1.0007 ± 0.0054	1.0186 ± 0.0155	1.0347 ± 0.0132
8	0.9984 ± 0.0067	1.0128 ± 0.0136	1.0255 ± 0.0191

Table 8.1: Results for cross-validation by likelihood

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.0000 ± 0.0019	1.0016 ± 0.0055	1.0474 ± 0.0275
2	0.9993 ± 0.0081	1.0041 ± 0.0091	1.0408 ± 0.0236
3	0.9993 ± 0.0086	1.0091 ± 0.0088	1.0394 ± 0.0203
4	0.9985 ± 0.0101	1.0102 ± 0.0146	1.0359 ± 0.0169
6	1.0005 ± 0.0054	1.0113 ± 0.0100	1.0284 ± 0.0158
8	1.0050 ± 0.0144	1.0190 ± 0.0209	1.0256 ± 0.0259

Table 8.2: Results for cross-validation by prediction error

- $\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$: MSE for the mixture model where the true prior $p(\boldsymbol{\theta})$ (which is a mixture of normal) is given[3];
- $\mathsf{MSE}(\hat{f}_{STL})$: MSE obtained by using single-task learning algorithms;

We are interested in several comparisons from the experiments. First of all, we would like to know how good is our fitted model compared to the one obtained by knowing H, the true number of clusters. Second, we want to measure the relative goodness of the fitted model with respect to the "golden model" where we are given the true prior distribution of $\boldsymbol{\theta}_k$'s. Finally, we want to see how good is the model obtained by using single-task learning algorithm which does not consider the relations among tasks.

Table 8.1 and 8.2 show the results for the likelihood-based and error-based cross-validation, respectively. There are several observations. First, the model $\hat{f}_{\hat{H}}$ (with the number of clusters identified by cross-validation) is almost identical to the one fitted by given the true number of clusters. Furthermore, it is slightly inferior to the "golden model" which is given the

[3]This is the upper bound of the performance we can possibly achieve.

true prior $p(\boldsymbol{\theta}_k)$. Second, the performance obtained by single-task learning (e.g. without learning a joint prior over $\boldsymbol{\theta}_k$'s) can be significantly worse, as shown in the last column of both tables. Finally, we observed that both the likelihood-based CV and error-based CV methods work well and they perform very similarly.

Also note that if those clusters are well-separated then they can be easily identified by our algorithm; otherwise (e.g. when clusters are overlapping with each other) it is very difficult to identify the correct number of clusters. In either case, however, the identified model works well in terms of predictive power.

8.4 Summary

In our experiments we show the application of cross-validation techniques to multi-task learning, and results for two methods are comparable. Although we only illustrated this ability using the learning of number of components in mixture models, this should not be interpreted as the only application of the idea. We could, as another example, select the appropriate multi-task learning scenario.

The application of cross-validation techniques to the model selection problem in multi-task learning is procedurely straight-forward yet conceptually stimulating. In order to conduct cross-validation for multi-task learning (e.g. across tasks), it is essentially for the MTL method to have the capability of "passing" or "transferring" knowledge from old tasks to new tasks. All MTL methods proposed in this thesis can pass knowledge as a prior and pass it into new tasks. The error-based cross-validation is computationally more expensive since an inner loop CV is needed to evaluate the prior learned from other tasks.

Chapter 9

Unsupervised Multi-Task Learning

In previous chapters we have focused on the probabilistic framework in equation (3.2) for multi-task learning in a supervised learning setting, e.g. with classification and regression tasks. In this chapter we extend the framework to enable its use in unsupervised learning, and apply it to novelty detection [Allan et al., 2000] as a significant and concrete example. We also show the theoretical connections between this new framework and other unsupervised learning methods, including Latent Dirichlet Allocation (LDA) [Blei et al., 2003b] and Correlated Topic Models (CTM) [Blei and Lafferty, 2005] proposed in information retrieval, by recasting the latter models from a multi-task learning point of view.

9.1 Extending the Framework from Supervised to Unsupervised

Recall that our probabilistic framework has been defined with K supervised learning tasks, each of which corresponds to a function $f(\mathbf{x}|\boldsymbol{\theta}_k) = \boldsymbol{\theta}_k^T \mathbf{x}$, which the prior over $\boldsymbol{\theta}_k$ defined as

$$\begin{aligned} \boldsymbol{\theta}_k &= \Lambda \mathbf{s}_k + \mathbf{e}_k \\ \mathbf{s}_k &\sim p(.|\Phi) \\ \mathbf{e}_k &\sim \mathsf{Normal}(\mathbf{0}, \Psi) \end{aligned} \tag{9.1}$$

where $\boldsymbol{\theta}_k$ is the vector of task-specific coefficients, the columns of matrix $\Lambda = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_H) \in \mathbb{R}^{F \times H}$ are the shared components (e.g., the hidden "topics" or "factors") among tasks; vector $\mathbf{s}_k$ consists of the mixture weights of the components in the k-th task, generated at random from some distribution $p(\mathbf{s}_k|\Psi)$ (open to further specification as a prior); and $\mathbf{e}_k$ is the non-shared component (random noise) in the k-th task. Given K training sets

$$\mathcal{D}^{(k)} = \{(\mathbf{x}_1^{(k)}, y_1^{(k)}), \ldots, (\mathbf{x}_{n_k}^{(k)}, y_{n_k}^{(k)})\}, \quad k = 1, \ldots, K, \tag{9.2}$$

the learning problem is to fit the model parameters that best explain the data under certain Bayesian priors over the parameters. An alternative way to look at the problem is that we have an infinite space of prediction functions, and we want to find the K optimal functions simultaneously, one per task. By introducing the shared components as parameters of the models (i.e., the functions), we aim to learn more effectively from limited training examples when the tasks are not totally independent from each other.

To make this framework suitable for unsupervised learning, we introduce some different settings. First, the training data are unlabeled, that is,

$$\mathcal{D}^{(k)} = \{\mathbf{x}_1^{(k)}, \ldots, , \mathbf{x}_{n_k}^{(k)}\}. \tag{9.3}$$

Second, our objective here is to optimize the generative model $g(\mathcal{D}^{(k)}|\boldsymbol{\theta}_k)$ for each dataset instead of the prediction function $f(\mathbf{x}^{(k)}|\boldsymbol{\theta}_k)$, although both can be done using the likelihood principle.

Having the tasks re-defined, the remaining equation (3.2) in our framework are the same for both supervised learning and unsupervised learning. In the latter, $\boldsymbol{\theta}_k$ is a random variable (vector), inheriting the randomness from $\mathbf{s}_k$ and $\mathbf{e}_k$, respectively, and specifies the probability distribution for generating $\mathcal{D}^{(k)}$. The two-step process, i.e. first generating the $\boldsymbol{\theta}_k$ (with some Bayesian priors over $\mathbf{s}_k$ and $\mathbf{e}_k$) and second generating the data $\mathcal{D}^{(k)}$ using parameter $\boldsymbol{\theta}_k$, can be viewed as a hierarchical generative model for the data sets $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(K)}$. By modeling all the tasks together, more reliable estimation of the model parameters using limited training data is possible if the tasks are closely related to each other, or, when the estimated density functions $g(\mathcal{D}^{(k)}|\boldsymbol{\theta}_k)$ for $k = 1, \ldots, K$ have certain dependencies among each other.

Despite the different settings, it should be point out that both supervised and unsupervised models can be put together into a comprehensive framework for multi-task learning. That is, the learning problem is to search through

a function space for the optimal one per task. The function space consists of prediction functions (predicting output variables given input variables) in the supervised settings, and of density functions (generating input variables) in the unsupervised settings.

In the remaining sections we show how to apply the comprehensive framework to concrete problems in unsupervised learning, and how to establish theoretical connections from existing unsupervised methods to multi-task learning. Because all the examples are used in modeling documents, we will change our notation for convenience in the rest of this chapter:

1. We use $\mathbf{x} = (n_1^{(\mathbf{x})}, n_2^{(\mathbf{x})}, \ldots, n_V^{(\mathbf{x})})$ to represent a document vector [1] where each element $n_v^{(\mathbf{x})}$ is the within-document term frequency of the v-th word, and V is the total size of the vocabulary.

2. Every single document $\mathbf{x}_i$ is a task[2] (i.e., $\mathbf{x}_i$ is the training dataset for the i-th task, $\mathcal{D}^{(i)}$), and thus we have $i = 1, \ldots, N$ tasks instead of K tasks before.

3. For each document, we try to estimate the density $g(\mathbf{x}_i|\boldsymbol{\theta}_i)$, and we use T instead of H to denote the number of hidden components (topics).

It is interesting to point out that under this setting, single-task learning does not make much sense: it just memorizes the bag-of-words representation in every document!

9.2 Multi-Task Learning and Unsupervised Clustering

Latent Dirichlet Allocation (LDA) [Blei et al., 2003b] and Correlated Topic Models (CTM) [Blei and Lafferty, 2005] are two well-known approaches to unsupervised clustering of documents. By projecting a document onto a set of "topics", documents can be better represented, interpreted and visualized

[1] This is the so-called "bag-of-words" representation which simply ignores the word occurring order in the document.

[2] This may seem a little bit weird. However, recall that if we want to model a document $\mathbf{x}$ using Multinomial distribution, then each word x_t can be thought as a "data point" for the task. Thus we are still using a lot of data points to estimate each task here.

using the estimated topics rather than the original bag-of-words representation. Next we show that the above two topic models can be seen as special cases of equation (9.1) for multi-task learning[3].

Recall that in LDA, a document $\mathbf{x}$ is generated in the following steps:

1. A topic distribution variable $\mathbf{s} \in \mathbb{R}^{T\times 1}$ is first generated as $\mathbf{s} \sim \mathsf{Dirichlet}(\boldsymbol{\alpha})$ (that is, $\mathbf{s}$ belongs to the $(T-1)$ dimensional simplex);
2. For each word w in the document[4]:
   (a) Choose a topic $z \sim \mathsf{Multinomial}(\mathbf{s})$;
   (b) Choose a word $w \sim \mathsf{Multinomial}(\boldsymbol{\beta}_z)$, where $\boldsymbol{\beta}_z \in \mathbb{R}^{F\times 1}$ is the multinomial parameter vector for topic z.

It is important to realize that, by combining steps 2(a) and 2(b), we can integrate out the latent variable z which represents the topic:

$$\begin{aligned} p(w) &= \sum_{z=1}^{T} p(z)p(w|z) \\ &= \sum_{z=1}^{T} \mathsf{Multinomial}(z|\mathbf{s})\mathsf{Multinomial}(w|\boldsymbol{\beta}_z) \\ &= \sum_{z=1}^{T} s_z \boldsymbol{\beta}_z(w) \\ &= \boldsymbol{\theta}(w) \end{aligned} \tag{9.4}$$

where T is the total number of topics, $\boldsymbol{\beta}_z(w)$ is the element of $\boldsymbol{\beta}_z$ that corresponds to word w, and $\boldsymbol{\theta} = \sum_{z=1}^{T} s_z \boldsymbol{\beta}_z \in \mathbb{R}^{F\times 1}$. That is, by integrating out z, we have $w \sim \mathsf{Multinomial}(\boldsymbol{\theta})$[5]. Thus, the overall generation process can be summarized using the following succinct form

$$\begin{aligned} \boldsymbol{\theta}_i &= \Lambda \mathbf{s}_i \\ \mathbf{s}_i &\sim \mathsf{Dirichlet}(\boldsymbol{\alpha}) \end{aligned} \tag{9.5}$$

[3]Note that it has been previously pointed out in [Buntine, 2002] that LDA can also be seen as multinomial PCA.

[4]We ignore the document length here as it does not affect the LDA model.

[5]Basically it results from the fact that a mixture of multinomial is still a multinomial, if we limit our discussion to the special case of multinomial distributions $\mathsf{Multinomial}(N = 1, p_1, \ldots, p_J)$ where only one ball is selected out of a bag of J colored balls with proportional probabilities $p_1, \ldots, p_J$.

where $\boldsymbol{\theta}_i$ is the multinomial parameter for the i-th document, $\mathbf{s}_i$ lies in the $(T-1)$ dimensional simplex and denotes the topic proportional distribution, and columns of $\Lambda = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_T) \in \mathbb{R}^{F \times T}$ are the multinomial parameters of those topics.

Comparing these formulae to equation (9.1), it is easy to see that our multi-task learning framework resembles the LDA formula except that now each column of Λ is restricted to a high-dimensional simplex in order to be a valid model. That is, it has to satisfy $\Lambda_{t,f} \geq 0$ and $\sum_{f=1}^{F} \Lambda_{t,f} = 1$ for $\forall t = 1, \ldots, T$. In other words, LDA can be thought as a special application of the multi-task learning framework to unsupervised clustering.

Correlated Topic Models (CTM) is an alternative approach to unsupervised clustering. It can be viewed as a modification of LDA so that the correlations among topics can be explicitly modeled. The model can be written in a succinct form as

$$\begin{aligned} \boldsymbol{\theta}_i &= \Lambda \tilde{\mathbf{s}}_i \\ \mathbf{s}_i &\sim \mathsf{LogNormal}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \end{aligned} \tag{9.6}$$

where the vector $\tilde{\mathbf{s}}_i$ is a re-scaled version of $\mathbf{s}_i$ such that $\sum_{h=1}^{H} \tilde{s}_{i,h} = 1$, a necessary condition for ensuring $\boldsymbol{\theta}_k$ to be a valid multinomial parameter (belongs to a $(F-1)$ dimensional simplex).

Comparing the $\mathbf{s}_i$ formula in CTM with the one in LDA: By using the LogNormal distribution [Gelman et al., 2003] [6] instead of the Dirichlet distribution, correlations among topic occurrence (e.g., elements of $\mathbf{s}_i$) can be encoded in the covariance parameter $\boldsymbol{\Sigma}$ and thus the topic models estimated are generally correlated. As a result, the CTM model is more flexible than the LDA model in the sense it users $T + T(T+1)/2$ number of parameters to model $\mathbf{s}_i$ while in LDA only T parameters are used to model $\mathbf{s}_i$. In other words, CTM can model both the first-order (mean) and second-order statistics (covariance) of $\mathbf{s}_i$ while LDA is only capable of modeling the first-order statistics. Nevertheless, comparing the CTM formula to equation (9.1), again our multi-task learning framework resembles the CTM model, except the assumed prior distribution $\mathbf{s}_i$ is different from the prior used in LDA.

The above connection between those topic models and models for multi-task learning is simple, yet interesting. Ideas and insights from one field can motivate research problems in the other. For example, it is known in

[6] Briefly speaking, if a random variable X has a Normal distribution, then $\exp(X)$ has a LogNormal distribution.

multi-task learning that task-specific components $\mathbf{e}_k$'s are important in order to have good performance as the number of training examples grows. In document modeling although each document is a task (and thus has fixed number of words), adding a task-specific component might be helpful in modeling topics as well. Since in the context of topic modeling each task corresponds to a single document, task-specific component could be learned, say, by limiting its deviation from a general English multinomial distribution [Miller et al., 1999, Zhang et al., 2004].

As a summarization statement, the connections between LDA, CTM and models for multi-task learning have not been analyzed so far, to our knowledge. Nevertheless, these connections are not surprising to see, but rather conceptually natural. From a higher-level point of view, the common goal of LDA, CTM and multi-task learning is to model functions $f^{(k)}$ in some generic metric space $\mathcal{H}$, either in a supervised or unsupervised way, and our framework supports the search for solutions under different scenarios.

9.3 Unsupervised Learning of Novelty Detection[7]

In this section we illustrate how to use a probabilistic model for novelty detection. The task of *online document clustering* is to group documents into clusters as long as they arrive in a temporal sequence. Generally speaking, it is difficult for several reasons: First, it is unsupervised learning and the learning has to be done in an online fashion, which imposes constraints on both strategy and efficiency. Second, similar to other learning problems in text, we have to deal with a high-dimensional space with tens of thousands of features. And finally, the number of clusters can be as large as thousands in newswire data. The objective of novelty detection is to identify the novel objects from a sequence of data, where "novel" is usually defined as dissimilar to previous seen instances. Here we are interested in novelty detection in the text domain, where we want to identify the earliest report of every new event in a sequence of news stories. The most obvious application of novelty detection is that, by detecting novel events, systems can automatically alert people when new events happen, for example. Applying online document clustering to the novelty detection task is straightforward by assigning the first seed of every cluster as novel and all its remaining ones as non-novel.

Our probabilistic model can also be seen as a very special case of equation (9.1). To be more specific, we use non-parametric Dirichlet process

[7] This part is primarily based on our previous paper [Zhang et al., 2004].

prior to model the growing number of clusters (which is modeled by multinomial distribution with parameter $\boldsymbol{\theta}$), and use a prior of general English language model as the base distribution of DP to handle the generation of novel clusters. Furthermore, cluster uncertainty is modeled with a Bayesian Dirichlet-multinomial distribution. The Bayesian inference can be easily carried out due to conjugacy, and model hyper-parameters are estimated using a historical dataset by the empirical Bayes method. The probabilistic model is applied to the novelty detection task in Topic Detection and Tracking (TDT), which has been regarded as the hardest task in TDT [Allan et al., 2000], and compared with existing approaches in the literature.

9.3.1 A Probabilistic Model for Online Document Clustering

Below we describe the generative probabilistic model for online document clustering.

Dirichlet-Multinomial Model

The multinomial distribution has been the most frequently used language model for probabilistic representation of documents in information retrieval. Let $\mathbf{x} = (n_1^{(\mathbf{x})}, \ldots n_V^{(\mathbf{x})})$ be the vector representation of a document and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_V)$ be the model parameter of a document cluster, a document $\mathbf{x}$ is generated with the following probability:

$$p(\mathbf{x}|\theta) = \frac{\left(\sum_{v=1}^{V} n_v^{(\mathbf{x})}\right)!}{\prod_{v=1}^{V} n_v^{(\mathbf{x})}!} \prod_{v=1}^{V} \theta_v^{n_v^{(\mathbf{x})}}. \tag{9.7}$$

From the formula we can see the so-called naive assumption: words in a document are assumed to be independent of each other[8]. Given a collection of documents generated from the same model, the parameter $\boldsymbol{\theta}$ can be estimated with Maximum Likelihood Estimation. In a Bayesian approach we would like to put a Dirichlet prior over the parameter ($\boldsymbol{\theta} \sim \mathsf{Dirichlet}(\boldsymbol{\alpha})$) such that the probability of generating a document is obtained by integrating over the parameter space:

$$p(\mathbf{x}) = \int p(\boldsymbol{\theta}|\boldsymbol{\alpha}) p(\mathbf{x}|\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{9.8}$$

This integration can be easily written down due to the conjugacy between Dirichlet and multinomial distributions. The key difference between the

[8]Strictly speaking, words are weakly dependent given N, the document length, where the weak dependency comes from the fact that $N = \sum_v n_v(\mathbf{x})$.

Bayesian approach and the MLE is that the former uses a distribution to model the uncertainty of the parameter θ, while the latter gives only a point estimation.

Online Document Clustering with Dirichlet Process Mixture Model

In our system documents are grouped into clusters in an online fashion. Each cluster is modeled with a multinomial distribution whose parameter $\boldsymbol{\theta}$ follows a Dirichlet prior. First, a cluster is chosen based on a Dirichlet process prior (can be either a new or existing cluster), and then a document is drawn from that cluster. We use Dirichlet Process (DP) to model the prior distribution of $\boldsymbol{\theta}$'s, and our hierarchical model is as follows:

$$\begin{aligned} \boldsymbol{\theta}_i &\sim G \\ G &\sim \mathsf{DP}(\lambda, G_0) \\ \mathbf{x}_i | c_i &\sim \mathsf{Multinomial}(.|\boldsymbol{\theta}^{c_i}) \end{aligned} \tag{9.9}$$

where c_i is the cluster indicator variable, $\boldsymbol{\theta}_i$ is the multinomial parameter for each document, and $\boldsymbol{\theta}^{(c_i)}$ is the unique $\boldsymbol{\theta}$ for the cluster c_i. G is a random distribution generated from the Dirichlet process $\mathsf{DP}(\lambda, G_0)$ [Ferguson, 1973], which has a precision parameter λ and a base distribution G_0. Here our base distribution G_0 is a Dirichlet distribution $\mathsf{Dirichlet}(\gamma\pi_1, \gamma\pi_2, \ldots, \gamma\pi_V)$ with $\sum_{t=1}^{V} \pi_t = 1$, which reflects our expected knowledge about G. Intuitively, our G_0 distribution can be treated as the prior over general English word frequencies, which has been used in information retrieval literature [Zaragoza et al., 2003] to model general English documents.

The exact cluster-document generation process can be described as follows:

1. Let $\mathbf{x}_i$ be the current document under processing (the i-th document in the input sequence), and $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m$ are already generated clusters.

2. Draw a cluster c_i based on the following Dirichlet process prior [Ferguson, 1973]:

$$\begin{aligned} p(c_i = \mathbf{C}_j) &= \frac{|\mathbf{C}_j|}{\lambda + \sum_{j=1}^{m} |\mathbf{C}_j|} \quad (j = 1, 2, \ldots, m) \\ p(c_i = \mathbf{C}_{m+1}) &= \frac{\lambda}{\lambda + \sum_{j=1}^{m} |\mathbf{C}_j|} \end{aligned} \tag{9.10}$$

   where $|\mathbf{C}_j|$ stands for the cardinality of cluster j with $\sum_{j=1}^{m} |\mathbf{C}_j| = i-1$, and with certain probability a new cluster $\mathbf{C}_{m+1}$ will be generated.

3. Draw the document $\mathbf{x}_i$ from the cluster c_i.

Model Updating

Our models for each cluster need to be updated based on incoming documents. We can write down the probability that the current document $\mathbf{x}_i$ is generated by any cluster as

$$p(\mathbf{x}_i|\mathbf{C}_j) = \int p(\boldsymbol{\theta}^{(\mathbf{C}_j)}|\mathbf{C}_j)p(\mathbf{x}_i|\boldsymbol{\theta}^{(\mathbf{C}_j)})d\boldsymbol{\theta}^{(\mathbf{C}_j)} \;\; (j = 1, 2, \ldots, m, m+1) \quad (9.11)$$

where $p(\boldsymbol{\theta}^{(\mathbf{C}_j)}|\mathbf{C}_j)$ is the posterior distribution of parameters of the j-th cluster whose update is based on equation (9.14) and we use $p(\boldsymbol{\theta}^{(\mathbf{C}_{m+1})}|\mathbf{C}_{m+1}) = p(\boldsymbol{\theta}^{(\mathbf{C}_{m+1})})$ to represent the prior distribution of the parameters of the new cluster for convenience. Although the dimensionality of $\boldsymbol{\theta}$ is high ($V \approx 10^5$ in our case), closed-form solution can be obtained under our Dirichlet-multinomial assumption. Once the conditional probabilities $p(\mathbf{x}_i|\mathbf{C}_j)$ are computed, the probabilities $p(\mathbf{C}_j|\mathbf{x}_i)$ can be easily calculated using Bayes rule:

$$p(\mathbf{C}_j|\mathbf{x}_i) = \frac{p(\mathbf{C}_j)p(\mathbf{x}_i|\mathbf{C}_j)}{\sum_{j'=1}^{m+1} p(\mathbf{C}_{j'})p(\mathbf{x}_i|\mathbf{C}_{j'})} \quad (9.12)$$

where the prior probability of each cluster is calculated using equation (9.10). Now there are several choices we can consider on how to update the cluster models. The first choice, which is correct but obviously intractable, is to fork $m+1$ children of the current system where the j-th child is updated with document $\mathbf{x}_i$ assigned to cluster j, while the final system is a probabilistic combination of those children with the corresponding probabilities $p(\mathbf{C}_j|\mathbf{x}_i)$. The second choice is to make a hard decision by assigning the current document $\mathbf{x}_i$ to the cluster with the maximum probability:

$$\mathbf{c}_i = \arg\max_{\mathbf{C}_j} p(\mathbf{C}_j|\mathbf{x}_i) = \frac{p(\mathbf{C}_j)p(\mathbf{x}_i|\mathbf{C}_j)}{\sum_{j'=1}^{m+1} p(\mathbf{C}_{j'})p(\mathbf{x}_i|\mathbf{C}_{j'})}. \quad (9.13)$$

The third choice is to use a soft probabilistic updating, which is similar in spirit to the Assumed Density Filtering (ADF) [Minka, 2001] in the literature. That is, each cluster is updated by exponentiating the likelihood function with probabilities:

$$p(\boldsymbol{\theta}^{(\mathbf{C}_j)}|\mathbf{x}_i, \mathbf{C}_j) \;\; \propto \;\; \left(p(\mathbf{x}_i|\boldsymbol{\theta}^{(\mathbf{C}_j)})\right)^{p(\mathbf{C}_j|\mathbf{x}_i)} p(\boldsymbol{\theta}^{(\mathbf{C}_j)}|\mathbf{C}_j) \quad (9.14)$$

However, we have to specially deal with the new cluster since we cannot afford both time-wise and space-wise to generate a new cluster for each incoming document. Instead, we will update all existing clusters as above, and new cluster will be generated only if $c_i = \mathbf{C}_{m+1}$. We will use HD and PD (hard decision and probabilistic decision) to denote the last two candidates in our experiments.

9.3.2 Learning Model Parameters

In the above probabilistic model there are still several hyper-parameters not specified, namely the π and γ in the base distribution

$$G_0 = \mathsf{Dirichlet}(\gamma\pi_1, \gamma\pi_2, \ldots, \gamma\pi_V), \tag{9.15}$$

and the precision parameter λ in the $\mathsf{DP}(\lambda, G_0)$. Since we can obtain a partially labeled historical dataset[9], we now discuss how to estimate those parameters respectively. We will mainly use the empirical Bayes method [Gelman et al., 2003] to estimate those parameters instead of taking a full Bayesian approach, since it is easier to compute and generally reliable when the number of data points is relatively large compared to the number of parameters. Because the $\boldsymbol{\theta}_i$'s are IID. from the random distribution G, by integrating out the G we get

$$\boldsymbol{\theta}_i|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{i-1} \sim \frac{\lambda}{\lambda + i - 1} G_0 + \frac{1}{\lambda + i - 1} \sum_{j<i} \delta_{\boldsymbol{\theta}_j} \tag{9.16}$$

where the distribution is a mixture of continuous and discrete distributions, and the $\delta_{\boldsymbol{\theta}}$ denotes the probability measure giving point mass to $\boldsymbol{\theta}$.

Now suppose we have a historical dataset H which contains K labeled clusters $H_j (j = 1, 2, \ldots, K)$, with the k-th cluster $H_k = \{\mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \ldots, \mathbf{x}_{k,m_k}\}$ having m_k documents. The joint probability of $\boldsymbol{\theta}$'s of all documents can be obtained as

$$p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{|H|}) = \prod_{i=1}^{|H|} \left(\frac{\lambda}{\lambda + i - 1} G_0 + \frac{1}{\lambda + i - 1} \sum_{j<i} \delta_{\boldsymbol{\theta}_j} \right) \tag{9.17}$$

[9] Although documents are grouped into clusters in the historical dataset,we cannot make directly use of those labels due to the fact that clusters in the test dataset are different from those in the historical dataset.

where $|H|$ is the total number of documents. By integrating over the unknown parameter $\boldsymbol{\theta}$'s we can get

$$\begin{aligned} p(H) &= \int \left(\prod_{i=1}^{|H|} p(\mathbf{x}_i|\boldsymbol{\theta}_i) \right) p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{|H|}) d\boldsymbol{\theta}_1 d\boldsymbol{\theta}_2 \ldots d\boldsymbol{\theta}_{|H|} \\ &= \prod_{i=1}^{|H|} \left(\int p(\mathbf{x}_i|\boldsymbol{\theta}_i) \left(\frac{\lambda}{\lambda + i - 1} G_0 + \frac{1}{\lambda + i - 1} \sum_{j<i} \delta_{\boldsymbol{\theta}_j} \right) d\boldsymbol{\theta}_i \right) \end{aligned} \quad (9.18)$$

Empirical Bayes method can be applied to equation (9.18) to estimate the model parameters by maximization[10]. In the following we discuss how to estimate parameters individually in detail.

Estimating π_t's

Our hyper-parameter π vector contains V number of parameters for the base distribution G_0, which can be treated as the expected distribution of G – the prior of the cluster parameter $\boldsymbol{\theta}$'s. Although π contains $V \approx 10^5$ number of actual parameters in our case, we can still use the empirical Bayes to do a reliable point estimation since the amount of data we have to represent general English is large (in our historical dataset there are around 10^6 documents, around 1.8×10^8 English words in total) and highly informative about π. We use the smoothed estimation

$$\pi \propto (1 + n_1^{(H)}, 1 + n_2^{(H)}, \ldots, 1 + n_V^{(H)}) \quad (9.19)$$

where $n_t^{(H)} = \sum_{\mathbf{x} \in H} n_t^{(\mathbf{x})}$ is the total number of times that term t happened in the collection H, and $\sum_{t=1}^{V} \pi_t$ should be normalized to 1. Furthermore, the pseudo-count one is added to alleviate the out-of-vocabulary problem (a more systematic way is to assign a Dirichlet prior).

Estimating γ

Though γ is just a scalar parameter, it has the effect of controlling the uncertainty of the prior knowledge about how clusters are related to the general English model with the parameter π. We can see that γ controls how far each new cluster can deviate from the general English model[11]. It

[10] Since only a subset of documents are labeled in the historical dataset H, the maximization is only taken over the union of the labeled clusters.

[11] Recall that the mean and variance of a Dirichlet distribution $(\theta_1, \theta_2, \ldots, \theta_V) \sim \mathsf{Dirichlet}(\gamma\pi_1, \gamma\pi_2, \ldots, \gamma\pi_V)$ are: $\mathbb{R}[\theta_v] = \pi_v$ and $\mathbb{V}[\theta_v] = \frac{\pi_v(1-\pi_v)}{(\gamma+1)}$.

can be estimated as follows:

$$\begin{aligned}\hat{\gamma} &= \arg\max_{\gamma} \prod_{k=1}^{K} p(H_k|\gamma) \\ &= \arg\max_{\gamma} \prod_{k=1}^{K} \int p(H_k|\boldsymbol{\theta}^{(k)}) p(\boldsymbol{\theta}^{(k)}|\gamma) d\boldsymbol{\theta}^{(k)} \end{aligned} \tag{9.20}$$

By setting the derivative to zero, $\hat{\gamma}$ can be numerically computed by solving the following equation:

$$\begin{aligned} 0 &= K\Psi(\gamma) - K\sum_{v=1}^{V} \Psi(\gamma\pi_v)\pi_v \\ &+ \sum_{k=1}^{K}\sum_{v=1}^{V} \Psi(\gamma\pi_v + n_v^{(H_k)})\pi_v - \sum_{k=1}^{K} \Psi(\gamma + \sum_{v=1}^{V} n_v^{(H_k)}) \end{aligned} \tag{9.21}$$

where the digamma function $\Psi(x)$ is defined as $\Psi(x) \equiv \frac{d}{dx}\ln\Gamma(x)$. Alternatively we can choose γ by evaluating over the historical dataset. This is applicable (though computationally expensive) since it is only a scalar parameter and we can pre-compute its possible range based on the result of equation (9.20).

Estimating λ

The precision parameter λ of the DP is also very important for the model, which controls how far the random distribution G can deviate from the baseline model G_0. In our case, it is also the prior belief about how quickly new clusters will be generated in the sequence. Similarly we can use equation (9.20) to estimate λ, since items related to λ can be factored out as

$$\prod_{i=1}^{|H|} \frac{\lambda^{y_i}}{\lambda + i - 1}. \tag{9.22}$$

Suppose we have a labeled subset $H^L = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$ of training data, where y_i is 1 if $\mathbf{x}_i$ is a novel document or 0 otherwise. Here we describe two possible choices:

1. The simplest way is to assume that λ is a fixed constant during the process, and it can be computed as

$$\hat{\lambda} = \arg\max_{\lambda} \prod_{i \in H^L} \frac{\lambda^{y_i}}{\lambda + i - 1}, \tag{9.23}$$

here H^L denotes the subset of indices of labeled documents in the whole sequence.

2. The assumption that λ is fixed may be too restrictive in reality, especially considering the fact that it reflects the generation rate of new clusters. More generally, we can assume that λ is some function of variable i[12]. In particular, we assume $\lambda = a/i + b + ci$ where a, b and c are non-negative numbers. This formulation is a generalization of the above case, where the i^{-1} term allows a much faster decrease at the beginning, and c is the asymptotic rate of events happening as $i \to \infty$. Again the parameters a, b and c are estimated by MLE over the training dataset:

$$\hat{a}, \hat{b}, \hat{c} = \arg\max_{a,b,c>0} \prod_{i \in H^L} \frac{(a/i + b + ci)^{y_i}}{a/i + b + ci}. \tag{9.24}$$

9.3.3 Experiments

We apply the above online clustering model to the novelty detection task in Topic Detection and Tracking (TDT). TDT has been a research community since its 1997 pilot study, which is a research initiative that aims at techniques to automatically process news documents in terms of events. There are several tasks defined in TDT, and among them Novelty Detection (a.k.a. First Story Detection or New Event Detection) has been regarded as the hardest task in this area [Allan et al., 2000]. The objective of the novelty detection task is to detect the earliest report for each event as soon as that report arrives in the temporal sequence of news stories.

Dataset

We use the TDT2 corpus as our historical dataset for estimating parameters, and use the TDT3 corpus to evaluate our model[13]. Notice that we have a subset of documents in the historical dataset (TDT2) for which events labels are given. The TDT2 corpus used for novelty detection task consists of 62,962 documents, among them 8,401 documents are labeled in 96 clusters. Stopwords are removed and words are stemmed, and after that there are on average 180 words per document. The total number of features (unique words) is around 100,000.

[12] It is not a DP anymore after this adaptation.

[13] Strictly speaking we only used the subsets of TDT2 and TDT3 that is designated for the novelty detection task.

Evaluation Measure

In our experiments we use the standard TDT evaluation measure [Yang et al., 2002] to evaluate our results. The performance is characterized in terms of the probability of two types of errors: **Miss** and **False Alarm** (**FA**) (P_{Miss} and P_{FA}). These two error probabilities are then combined into a single detection cost, C_{det}, by assigning costs to Miss and FA errors:

$$C_{det} = C_{Miss} \cdot P_{Miss} \cdot P_{target} + C_{FA} \cdot P_{FA} \cdot P_{non-target} \tag{9.25}$$

where

1. C_{Miss} and C_{FA} are the costs of a miss and a false alarm, respectively,
2. P_{Miss} and P_{FA} are the conditional probabilities of a miss and a false alarm, respectively and,
3. P_{target} and $P_{non-target}$ is the priori target probabilities such that $P_{target} = 1 - P_{non-target}$.

It is the following normalized cost that is actually used in evaluating various TDT systems:

$$(C_{det})_{norm} = \frac{C_{det}}{\min(C_{Miss} \cdot P_{target}, C_{FA} \cdot P_{non-target})} \tag{9.26}$$

where the denominator is the minimum of two trivial systems. Besides, two types of evaluations are used in TDT, namely macro-averaged (topic-weighted) and micro-averaged (story-weighted) evaluations. In macro-averaged evaluation, the cost is computed for every event, and then the average is taken. In micro-averaged evaluation the cost is averaged over all documents' decisions generated by the system, thus large event will have bigger impact on the overall performance. Note that macro-averaged evaluation is used as the primary evaluation measure in TDT.

In addition to the binary decision "novel" or "non-novel", each system is required to generated a confidence score for each test document. The higher the score is, the more likely the document is novel. Here we mainly use the minimum cost to evaluate systems by varying the threshold, which is independent of the threshold setting.

Methods

One simple but effective method is the "GAC-INCR" clustering method [Yang et al., 1998, 1999] with cosine similarity metric and TFIDF term weighting, which has remained to be the top performing system in TDT 2002 & 2003 official evaluations. For this method the novelty confidence score we used is one minus the similarity score between the current cluster $\mathbf{x}_i$ and its nearest neighbor cluster:

$$s(\mathbf{x}_i) = 1.0 - \max_{j<i} \mathsf{sim}(\mathbf{c}_i, \mathbf{c}_j) \tag{9.27}$$

where $\mathbf{c}_i$ and $\mathbf{c}_j$ are the clusters that $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to, respectively,and the similarity is taken to be the cosine similarity between two cluster vectors, where the ltc TFIDF term weighting scheme [Salton and Buckley, 1988] is used to scale each dimension of the vector. Our second method is to train a logistic regression model which combines multiple features generated by the GAC-INCR method. Those features not only include the similarity score used by the first method, but also include the size of its nearest cluster, the time difference between the current cluster and the nearest cluster, etc. We call this method "Logistic Regression", where we use the posterior probability $p(\mathsf{novelty}|\mathbf{x}_i)$ as the confidence score. Finally, for our online clustering algorithm we choose the quantity $s(\mathbf{x}_i) = \log p(\mathbf{C}_{new}|\mathbf{x}_i)$ as the output confidence score.

Experimental Results

Our results for three methods are listed in Table 9.1, where both macro-averaged and micro-averaged minimum normalized costs are reported. Furthermore, we also report the **Miss** and **FA** results to show the trade-off (recall that they are the two components of the cost in equation 9.25). The GAC-INCR method performs very well, so does the logistic regression method. For our DP results, we observed that using the optimized $\hat{\gamma}$ will get results (not listed in the table) that are around 10% worse than using the γ obtained through validation, which might be due to the flatness of the optimal function value as well as the sample bias of the clusters in the historical dataset[14]. Another observation is that the probabilistic decision does not actually improve the hard decision performance, especially for the λ_{var} option (remember that in the case of λ_{fix} option we learn $\hat{\lambda}$ from the data; in the case of λ_{var}-option we actually assume it to be a function of document index $\lambda = a/i + b + ci$ and learn the function parameters $\hat{a}$, $\hat{b}$ and $\hat{c}$). Generally speaking, our DP methods are comparable to the other two methods, especially in terms of topic-weighted measure.

[14] It is known that the cluster labeling process of LDC is biased toward topics that are

Method	**Topic-weighted Cost (Miss, FA)**	**Story-weighted Cost (Miss, FA)**
GAC-INCR	0.6945 (0.5614, 0.0272)	0.7090 (0.5614, 0.0301)
Logistic Regression	0.7027 (0.5732, 0.0264)	**0.6911** (0.5732, 0.0241)
DP with λ_{fix}, HD	0.7054 (0.4737, 0.0473)	0.7744 (0.5965, 0.0363)
DP with λ_{var}, HD	**0.6901** (0.5789, 0.0227)	0.7541 (0.5789, 0.0358)
DP with λ_{fix}, PD	0.7054 (0.4737, 0.0473)	0.7744 (0.5965, 0.0363)
DP with λ_{var}, PD	0.9025 (0.8772, 0.0052)	0.9034 (0.8772, 0.0053)

Table 9.1: Results for Novelty Detection on TDT3 Corpus

9.4 Summary

In this chapter we presented an extended version of our multi-task learning framework, to include both supervised and unsupervised settings. As a concrete and significant example, we show how to apply the framework to the novelty detection problem, with the evaluation results on a benchmark corpus that are comparable to the results of the best system in novelty detection. We also establish theoretical connections between our framework and other well-known Bayesian approaches to unsupervised learning of topic models such as the Latent Dirichlet Allocation and Correlated Topic Models.

Related work in unsupervised learning are the follows. Zaragoza et al. [Zaragoza et al., 2003] applied a Bayesian Dirichlet-multinomial model to the ad hoc information retrieval task and showed that it is comparable to other smoothed language models. Blei et al. [Blei et al., 2003a] used Chinese Restaurant Processes to model topic hierarchies for a collection of documents.

Another interesting and related research topic is *semi-supervised learning* where some of the response variable y's are given and some of them are missing. Various approaches have been proposed, such as [Zhu et al., 2003, Zhou et al., 2005, Zhang and Ando, 2005]. Viewed from the multi-task learning perspective, it is possible to extend our multi-task learning framework further to model parameters $\boldsymbol{\theta}_{unlabeled}$ and $\boldsymbol{\theta}_{labeled}$ jointly while considering $p(\mathbf{x}|\boldsymbol{\theta}_{unlabeled})$ and $p(y|\mathbf{x}, \boldsymbol{\theta}_{labeled})$. By capturing the dependencies between $\boldsymbol{\theta}_{unlabeled}$ and $\boldsymbol{\theta}_{labeled}$, we may be able to make more effective use of both labeled and unlabeled data.

---

covered in multiple languages instead of one single language.

Chapter 10

Summary and Discussions

In this thesis we have presented a unified probabilistic framework for multi-task learning, together with a series of models suitable for different task scenarios. In our framework task relatedness is explained by sharing a common structure through latent variables, and mathematically a flexible prior distribution is learned for task parameters using all training resources. Experiments show that they are able to take advantage of multiple related tasks to improve performance. Contributions of the thesis include:

- *A Unified Probabilistic Framework for Multi-Task Learning:* We proposed a novel probabilistic framework for multi-task learning. It can be seen as a hierarchical Bayesian model or latent variable model, whose flexibility (i.e., the capability to support a variety of task scenarios) mainly comes from two sources: the statistical assumption about latent variable $\mathbf{s}$ and the form of the shared structure (e.g., the mixing matrix Λ).

- *Systematic Exploration of Multi-Task Learning Scenarios:* We analyzed a series of important multi-task learning scenarios, and presented suitable models within the framework. The scenario analysis also sheds light on *how to properly formulate various applications into multi-task learning problems*.

- *Sparsity Models for Multi-Task Learning:* We proposed sparsity models for multi-task learning, where the sparsity is either in terms of the hidden source $\mathbf{s}_k$ or the linear mixing matrix Λ. In the former each prediction function is a sparse linear combination of basis functions;

while in the later each prediction function is a linear combination of basis functions that are sparse.

- *New Algorithms for Joint Feature Selection for Multi-Task Learning:* We proposed the $l_1 \circ l_p$ regularization algorithm which can be seen as a generalization of lasso for the multi-task learning setting. It couples the the same feature coefficients of all tasks by using a l_p-norm penalty, and thus is capable of utilizing information from all tasks.

- *Mixture Models for "Clusters of Tasks":* We generalize previous work by proposing mixture models for multi-task learning, which are the right choice for the "*clusters of tasks*" scenario. An efficient learning algorithm based on EM is presented and we achieved good results on both simulated data and collaborative filtering tasks.

- *Investigation on Model Selection for Multi-Task Learning:* We adapted the general idea of model selection to the multi-task learning setting, where the best joint model of all task parameters is chosen. This effort covers an unexplored area in multi-task learning, and is indispensable in order to find good models, especially when domain knowledge does not lead to an obvious choice.

There are still many open questions and opportunities in multi-task learning research:

- The performance gain of multi-task learning depends a lot on the number of tasks available. Ando and Zhang [2004] used heuristics to create many auxiliary tasks from unlabeled data and got good performance in several applications. How to design auxiliary tasks such that the multi-task learning can most benefit is an open question and deserves careful investigation.

- Classification with structured outputs has become a very popular research topic and has been applied to many interesting problems in natural language processing, information extraction and bio-informatics, where structured outputs naturally exist. A deep understanding of its connection to multi-task learning will be contribute to both research fields.

- In this thesis we mainly focused on supervised learning problems, and briefly discussed unsupervised multi-task learning. More generally we

can consider semi-supervised learning, where we have both labeled and unlabeled tasks. Can we develop effective approaches to semi-supervised multi-task learning?

- All scenarios discussed in this thesis assume that tasks have the same input space $\mathcal{X}$. However, it is not necessary since we can have multiple tasks with the their input space $\mathcal{X}^{(k)} = \mathcal{X} \oplus \mathcal{Z}^{(k)}$ where the $\mathcal{X}$ part of the input space is shared. In particular, semi-parametric models [Bickel et al., 1998] might be a good candidate which can have a non-parametric part for $\mathcal{Z}^{(k)}$ and a common parametric component for $\mathcal{X}$. It is still unclear how effective multi-task learning methods are for this partially sharing situation.

- In Chapter 8 we have focused on cross-validation techniques for model selection in multi-task learning. It is meaningful to investigate how well other model selection techniques perform, such as Bayesian model selection, AIC, BIC, MDL, GCV, etc. Another important question is, can we design efficient algorithms to obtain cross-validation errors without carrying out the expensive K-fold computation (using approximation or bounds?), especially the leave-one-task-out cross-validation?

We think that successfully addressing the above problems will significantly contribute to multi-task learning and make multi-task learning a more mature field.

Bibliography

J. Allan, V. Lavrenko, and H. Jin. First story detection in tdt is hard. In *Proceeding of CIKM*, 2000.

Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Technical Report RC23462, IBM T.J. Watson Research Center*, 45, 2004.

J. Baxter. Theoretical models of learning to learn, 1997.

J. Baxter. A bayesian/information theoretic model of bias learning. In *Proceedings of the 9th International Conference on Computational Learning Theory (COLT)*, 1996.

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

M.J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. Thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning, 2003.

J. Bernardo and A. Smith. *Bayesian Theory*. John Wiley & Sons, 1993.

P. Bickel, C. Klaassen, Y. Ritov, and J. Wellner. *Efficient and Adaptive Estimation for Semiparametric Models*. Springer, 1998.

D. Blei and J. Lafferty. Correlated topic models. In *Neural Information Processing Systems (NIPS) 18*, 2005.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *Neural Information Processing Systems (NIPS) 15*, 2003a.

D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003b.

Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the vapnikchervonenkis dimension. *Journal of the ACM*, 36:929–965, 1989.

G. Box and G. Tiao. *Bayesian Inference in Statistical Analysis*. Addison-Wesley, 1973.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

L. Breiman and J.H. Friedman. Predicting multivariate responses in multiple linear regression. *J. Royal. Statist. Soc B.*, 59(1):3–54, 1997.

J. Brutlag and C. Meek. Challenges of the email domain for text classification. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 2000.

W. Buntine. Variational extensions to em and multinomial pca. In *Proceedings of European Conference on Machine Learning*, 2002.

R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley, 1991.

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

Annette J. Dobson. *An Introduction to Generalized Linear Models*. Chapman and Hall/CRC, second edition, 2001.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 2004.

A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 110–120. Morgan Kaufmann, 1988.

B. Everitt. *An Introduction to Latent Variable Models*. Chapman and Hall, 1984.

T. Evgeniou and M. Pontil. Regularized multitask learning. In *Proceedings of 17th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2004.

T. Evgeniou, C. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

T. Ferguson. A bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230, 1973.

J. Friedman and J. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, 23(9):881–890, 1974.

A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, second edition, 2003.

Z. Ghahramani and M. Beal. Variational inference for bayesian mixtures of factor analysers. In *Advances in Neural Information Processing Systems*, volume 12, pages 449–455, 2000.

Z. Ghahramani and G. Hinton. Parameter estimation for linear dynamical systems. *University of Toronto, Technical Report CRG-TR-96-2*, 1996.

M.N. Gibbs and D. MacKay. Efficient implementation of gaussian processes. URL `http://www.inference.phy.cam.ac.uk/mng10/GP/`.

M. Girolami, editor. *Advances in Independent Component Analysis*. Springer-Verlag, 2000.

G. Golub and C. Van Loan. *Matrix Computation*. John Hopkins University Press, third edition, 1996.

R. Gorsuch. *Factor Analysis*. Lawrence Erlbaum Associates, second edition, 1983.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, first edition, 2001.

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

Tom Heskes. Empirical bayes for learning to learn. In *Proc. 17th International Conf. on Machine Learning*, pages 367–374. Morgan Kaufmann, San Francisco, CA, 2000.

P. Huber. Project pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

A. Hyvarinen, J. Karhumen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc., 2001.

A.G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(4):364–378, 1971.

T. Jaakkola and M. Jordan. A variational approach to bayesian logistic regression models and their extensions. In *Proceedings of 6th International Worksop on AI and Statistics*, 1997.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

W. James and C. Stein. Estimation with quadratic loss. In *Proceeding of Fourth Berkeley Symp. Math. Statist. Probab.*, pages 361–380, 1961.

M. Jordan. *An Introduction to Graphical Models*. unpublished manuscript, 2002.

R.E. Kass and A.E. Raftery. Bayes factors and model uncertainty. *Technical Report 254, University of Washington*, 1993.

G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95, 1971.

K. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 2000.

D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, 1997.

J.B. Kruskal. *Toward a practical method which helps uncover the structure of a set of observations by finding the line transformation which optimizes a new index of condensation.* Academic Press, New York, 1969.

E. Lehmann and G. Casella. *Theory of Point Estimation.* Springer-Verlag, second edition, 1998.

M. Lewicki and T.J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12:337–365, 2000.

D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Huan Liu and Rudy Setiono. Incremental feature selection. *Applied Intelligence*, 9(3):217–230, 1998.

David Luenberger. *Linear and Nonlinear Programming.* Springer, second edition, 2003.

A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall/CRC, second edition, 1989.

D. Miller, T. Leek, and R. Schwartz. Bbn at trec 7: Using hidden markov models for information retrieval. In *Proceeding of TREC 7*, 1999.

T. Minka. From hidden markov models to linear dynamical systems. *Technical Report, MIT*, 1999. URL `http://vismod.media.mit.edu/tech-reports/TR-531-ABSTRACT.html`.

T. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. Ph.D. Thesis, MIT, 2001.

T. Minka and R. Picard. Learning how to learn is learning with point sets, 1997.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in Artificial Intelligence*, 1999.

R. Neal. Probabilistic inference using markov chain monte carlo methods. *Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto*, 1993.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14. Cambridge, MA*, 2002.

Andrew Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, 2004.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1998.

Carl Rasmussen and Zoubin Ghahramani. Infinite mixtures of gaussian process experts. In *Neural Information Processing Systems (NIPS) 14*, 2002.

S. Roberson and D. Hull. The trec-9 filtering track report. In *Proceedings of TREC-9*, 2001.

C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Text in Statistics, 2005.

S. Roberts and R. Everson, editors. *Independent Component Analysis: Principles and Practice*. Cambridge University Press, 2001.

R. Rosenblatt. *Principles of neuro dynamics*. Spartan books, New York, 1959.

Y. Rubinstein and T. Hastie. Discriminative vs. informative learning. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997.

D. Rumelhart, G. Hinton, and R. williams. *Learning internal representations by error propagation.* MIT Press, Cambridge, MA, 1986.

G. Salton and C. Buckley. Term-weighting aapproaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

F.E. Satterthwaite. An approximation distribution of estimates of variance components. *Biometrics Bulletin*, 2:110–114, 1946.

L. Saul, T. Jaakkola, and M. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

D. Silver and R. Mercer. Selective functional transfer: Inductive bias from related tasks. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing (ASC2001)*, pages 182–189, 2001.

C. Spearman. General intelligence objectively determined and measured. *American Journal of Psychology*, 15:201–293, 1904.

C. Stein. Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceeding of Third Berkeley Symp. Math. Statist. Probab.*, pages 197–206, 1955.

Y. Teh, M. Seeger, and M. Jordan. Semiparametric latent factor models. In *AISTAT*, 2005.

S. Thrun and L. Pratt. *Learning to Learn.* Kluwer Academic Publishers, 1998.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 640–646. The MIT Press, 1996.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, 58:1:267–288, 1996.

A. N. Tikhonov. Solutions of incorrectly formulated problems and the regularization method. *Soviet Math Dokl*, 4:1035–1038, 1963.

M. Tipping and C. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 21(3):611–622, 1999.

Volker Tresp. Mixture of gaussian processes. In *Neural Information Processing Systems (NIPS) 13*, 2001.

van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 2000.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, second edition, 1999.

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2005.

L. Wasserman. *All of Nonparametrics*. Springer, 2006.

Jason Weston, Andre Elisseeff, Bernhard Scholkopf, and Mike Tipping. The use of zero-norm with linear models and kernel methods. *Journal of Machine Learning Research Special Issue on Feature Selection*, 3(Mar):1439–1461, 2003.

C. Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. 1998.

J. Winn. *Variational Message Passing and Its Applications*. Ph.D. Thesis, Department of Physics, University of Cambridge, 2003.

E. Xing, M. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Proceedings of Uncertainty in Artificial Intelligence*, 2003.

Y. Yang, T. Pierce, and J. Carbonell. A study on retrospective and on-line event detection. In *Proceeding of SIGIR*, 1998.

Y. Yang, J. Carbonell, R. Brown, T. Pierce, B. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems: Special Issue on Applications of Intelligent Information Retrieval*, 14:32–43, 1999.

Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proceeding of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

Yiming Yang, Jian Zhang, and Bryan Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference*, 2003.

Yiming Yang, Shinjae Yoo, Jian Zhang, and Bryan Kisiel. Robustness of adaptive filtering methods in a cross-benchmark evaluation. In *Proceedings of the 28th Annual International ACM SIGIR Conference*, 2005.

J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Technical Report TR-2001-22, Mitsubishi Electric Research Laboratories, Inc.*, 2002.

K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of 22nd International Conference on Machine Learning (ICML)*, 2005.

Shipeng Yu, Kai Yu, Volker Tresp, and Hans-Peter Kriegel. Collaborative ordinal regression. In *Proceedings of 23rd International Conference on Machine Learning (ICML)*, 2006.

H. Zaragoza, D. Hiemstra, D. Tipping, and S. Robertson. Bayesian extension to the language model for ad hoc information retrieval. In *Proceeding of SIGIR*, 2003.

J. Zhang. Preliminary results on email prioritization. In *Unpublished Manuscript*, 2002.

J. Zhang and Y. Yang. Robustness of regularized linear methods in text classification. In *SIGIR*, 2003.

J. Zhang, Z. Ghahramani, and Y. Yang. A probabilistic model for online document clustering with application to novelty detection. In *Neural Information Processing Systems (NIPS) 17*, 2004.

J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *Neural Information Processing Systems (NIPS) 18*, 2005.

T. Zhang and R. Ando. Graph-based semi-supervised learning and spectral kernel design. *Technical Report RC23713, IBM T.J. Watson Research Center*, 2005.

Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

Yi Zhang. Using bayesian priors to combine classifiers for adaptive filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference*, 2004.

D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. In *The 22nd International Conference on Machine Learning (ICML)*, 2005.

Z. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *The Twentieth International Conference on Machine Learning (ICML)*, 2003.