Learning with Limited Supervision by Input and Output Coding

Yi Zhang

May 2012
CMU-ML-12-102

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Jeff Schneider, Chair, CMU
Geoffrey Gordon, CMU
Tom Mitchell, CMU
Xiaojin Zhu, University of Wisconsin-Madison

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was funded in part by the National Science Foundation under grant NSF-IIS0911032 and the Department of Energy under grant DESC0002607. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: regularization, error-correcting output codes, supervised learning, semi-supervised learning, multi-task learning, multi-label classification, dimensionality reduction.

Abstract

In many real-world applications of supervised learning, only a limited number of labeled examples are available because the cost of obtaining high-quality examples is high. Even with a relatively large number of labeled examples, the learning problem may still suffer from limited supervision as the complexity of the prediction function increases. Therefore, learning with limited supervision presents a major challenge to machine learning. With the goal of supervision reduction, this thesis studies the representation, discovery and incorporation of extra input and output information in learning.

Information about the input space can be encoded by regularization. We first design a semi-supervised learning method for text classification that encodes the correlation of words inferred from seemingly irrelevant unlabeled text. We then propose a multi-task learning framework with a matrix-normal penalty, which compactly encodes the covariance structure of the joint input space of multiple tasks. To capture structure information that is more general than covariance and correlation, we study a class of regularization penalties on model compressibility. Then we design the projection penalty, which encodes the structure information from a dimension reduction while controlling the risk of information loss.

Information about the output space can be exploited by error correcting output codes. Using the composite likelihood view, we propose an improved pairwise coding for multi-label classification, which encodes pairwise label density (as opposed to label comparisons) and decodes using variational methods. We then investigate problem-dependent codes, where the encoding is learned from data instead of being predefined. We first propose a multi-label output code using canonical correlation analysis, where predictability of the code is optimized. We then argue that both discriminability and predictability are critical for output coding, and propose a max-margin formulation that promotes both discriminative and predictable codes.

We empirically study our methods in a wide spectrum of applications, including document categorization, landmine detection, face recognition, brain signal classification, handwritten digit recognition, house price forecasting, music emotion prediction, medical decision, email analysis, gene function classification, outdoor scene recognition, and so forth. In all these applications, our proposed methods for encoding input and output information lead to significantly improved prediction performance.

Acknowledgments

I am greatly indebted to many people during the course of this degree.

First and foremost, I want to express my sincere gratitude to my advisor Jeff Schneider, for his advice, encouragement, patience and humor. Having Jeff as my advisor is the most amazing experience in my academic life. This thesis would not have been possible without his advice and encouragement, and life during the past several years would not be nearly as enjoyable without his patience and humor.

I deeply appreciate my thesis committee members Tom Mitchell, Geoff Gordon and Jerry Zhu for providing me great insights and feedbacks on my thesis. I also took the course Advanced Statistical NLP from Tom and Optimization from Geoff (and Carlos Guestrin), and worked with them as a teaching assistant in Machine Learning and Optimization, respectively. I have learned so much from them on research and teaching. Jerry discussed with me about my research a lot and has always made great suggestions. I feel very fortunate to have Jerry as my external committee member.

Every course I had at CMU is a wonderful experience and I like to thank all my instructors for their time and effort: Carlos Guestrin, Larry Wasserman, Tom Mitchell, Geoff Gordon, Christos Faloutsos, Anthony Brockwell, Fallaw Sowell, Guy Blelloch, Daniel Golovin, Stephen Fienberg, Matthew Harrison and Surya Tokdar.

I did three great internships at Yahoo!, IBM T. J. Watson and Citadel LLC. I like to thank my mentors and managers for their supports and patience: Ye Chen, Dmitry Pavlov, Rong Yan, Apostol Natsev, Peng Zhao and Jonathan Graham.

I really want to thank my colleagues and friends at Auton Lab, Machine Learning Department and School of Computer Science at CMU, and to name a few, Andrew Arnold, Michael Baysek, Rishi Chandy, Polo Chau, Xi Chen, Carl Doersch, Artur Dubrawski, Bin Fan, Madalina Fiterau, Bin Fu, Roman Garnett, Haijie Gu, Fan Guo, Robert Hall, Jingrui He, Tzu-Kuo Huang, Mladen Kolar, Zhenzhen Kou, Lei Li, Han Liu, Austin McDonald, Barnabas Poczos, Robin Sabhnani, Burr Settles, Yanxin Shi, Diane Stidle, Dougal Sutherland, Rob Tillman, Hanghang Tong, Donghan Wang, Guang Xiang, Liang Xiong, Min Xu, Yang Xu, Liu Yang, Xin Zhang, Bin Zhao, Jun Zhu.

Finally, I want to thank my wife, Xinxin Dong and my mother, Yongping Ren. They are the origin of all my motivation, confidence and strength.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

With several decades of development, machine learning has become a fundamental tool in a wide range of computer-based applications and industrial problems, e.g., natural language processing, computer vision, computational biology, human-computer interaction, clinical and medical decisions. In all these applications, machine learning enables computers to learn from past experience to achieve high-quality decision making. In a supervised learning setting, past experience is represented by a set of labeled training examples, and learning to predict from training examples and generalizing well on unseen samples is the central goal of learning.

In many real-world situations, there are only a limited number of labeled examples because the cost of obtaining high-quality labeled examples is high, or because the prediction task is very specific. Even with a relatively large number of labeled examples, the learning problem may still suffer from limited supervision as the dimensionality of the input space or the complexity of the prediction function increases. This presents a major challenge to modern machine learning systems: *how to learn effectively from limited supervision*. Fortunately, for learning unknown prediction functions, labeled examples are far from the only source of information: extra information may be available from a variety of other sources such as unlabeled examples, other related prediction problems, and domain knowledge. Such extra information can characterize important structural properties of the input and output space, and when used appropriately, can significantly enhance the learning process, control overfitting, and improve the prediction performance.

In order to motivate the representation, discovery and incorporation of extra input and output information in learning, we consider a few examples as follows:

- The Web is an almost unlimited source of unlabeled text. But for a specific text classification problem, most text available from the Web seems irrelevant, i.e., does not belong to any target class or follow the distribution of the labeled examples. In this case, can we still learn some information about the input space from seemingly irrelevant unlabeled text and use it to improve text classification? Humans can definitely learn a lot about a language by reading a large amount of text, but what are people actually learning from reading unlabeled text?

- In many situations we have multiple similar but different tasks to learn, e.g., detecting landmines in different landmine fields or face recognition for different subjects. In this case, tasks usually share the same type of input space due to their commonality, e.g., the basic principle of landmines or the layout of human faces, and as a result, jointly learning multiple tasks may be more effective than learning each task independently. When each task has only a limited number of training examples, can we find a compact way to characterize the joint input space of multiple tasks so that we can share knowledge among tasks to improve multi-task learning?
- Consider building a classifier on images. Image compression has been studied for decades and provides us domain knowledge about images, and for example, the energy of real-world images can be compressed by transforming them to a frequency domain where most of their energy is concentrated in a few frequencies. Naturally, a model that classifies those images may only need to operate on a few relevant frequencies and thus also has compacted energy in the same frequency domain. How can we encode this information into the learning process?
- Dimension reduction techniques are intensively studied and widely used in many applications. A dimension reduction transforms the original input space into a reduced space, and more importantly, reveals structure information about the input space by highlighting certain parts of it. However, directly performing a reduction has the risk of losing relevant information for a prediction task. Indeed, it is usually hard to know in advance whether the information discarded by a reduction is relevant to a prediction task. In this case, can we encode the information conveyed by a dimension reduction but still control the risk of information loss?
- In real-world applications, we often want to predict multiple outputs. For example, an image can be associated with multiple scenes, and as in a multi-label classification setting, we want to predict the appearance of different scenes in an image. A common observation here is that the appearance of different scenes are not independent, e.g., rivers are likely to appear together with forests but highly unlikely to co-occur with deserts. This is recognized as the label dependency in multi-label classification. Facing a multi-label classification problem, will label dependency play an active role in improving classification performance? How can we extract and incorporate the hidden label dependency structure in the learning process?

1.2 Thesis Overview

With the goal of learning with limited supervision, this thesis studies the representation, discovery and incorporation of extra input and output information in learning. The organization of this thesis is summarized in Fig. 1.1 and discussed as follows. In Chapter 1, we present the motivation and overview of this thesis. In Chapter 2, we provide background material for related research areas.

In Part I (Chapter 3 - Chapter 6), we focus on encoding input information into the learning process, mainly by regularization. This part consists of the following four chapters:

- In Chapter 3, we start with encoding the covariance and correlation structure of the input space and propose a semi-supervised learning method for text classification: the semantic correlation of words, as an intrinsic structure of the language, can be inferred from unlabeled

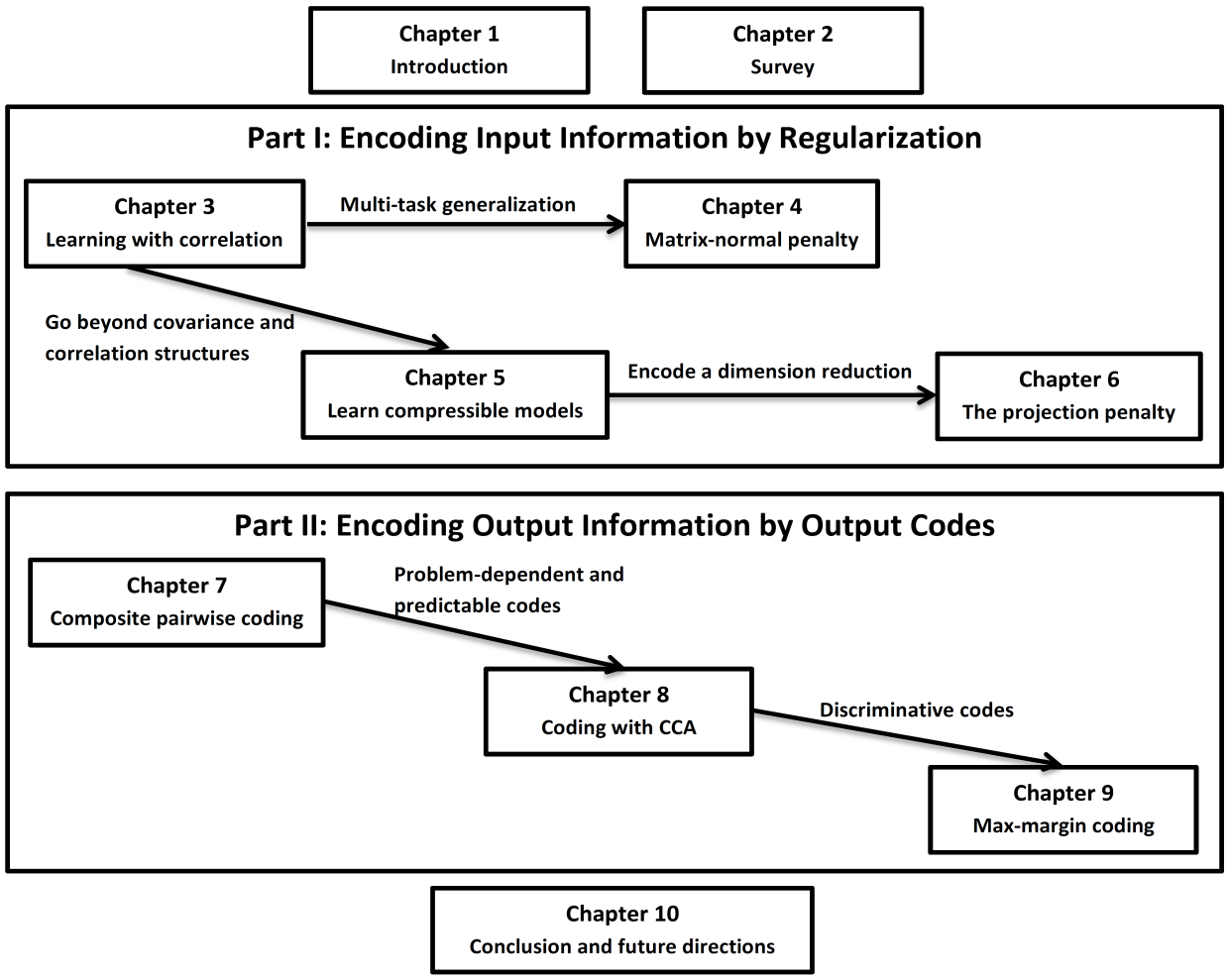


Figure 1.1: Organization of the thesis

text and provides strong structure information that can be incorporated into the regularization of any specific text learning task. As a result, even seemingly irrelevant unlabeled text, such as text downloaded from the Internet, can be used as an important source of extra information and encoded into learning to significantly improve the performance of text classification.

- In Chapter 4, we consider encoding the covariance structure of the *joint* input space when learning multiple related tasks. In multi-task learning, the joint input space (as well as the corresponding model parameters) can be represented as a matrix, where rows correspond to tasks and columns correspond to feature dimensions. Therefore, we design a matrix-normal regularization penalty that decomposes the full covariance of matrix entries into the Kronecker product of row covariance and column covariance, which characterize task relations and feature representations, respectively. Sparse covariance selection is applied on the inverse of both task and feature covariances in order to automatically select task and feature structures.

- In Chapter 5, we study how to encode input information that is more general than just the covariance and correlation structure. We consider learning compressible models, where certain structure information about the input space is available from domain knowledge and can be encoded as a compression operation in the regularization penalty to promote model compressibility. Examples of such model compressibility include local smoothness in brain signal classification and frequency-domain compacted energy in image recognition.
- In Chapter 6, we investigate another common scenario in which the structure information about the input space is highlighted by a dimension reduction, e.g., a subset of features, a clustering of low-level features, or a subspace (or manifold) of the input space. In this case, we want to encode this information (conveyed by the reduction) into learning while controlling the risk of losing relevant information during the reduction. We propose the projection penalty: a reduction of the input space can be viewed as a restriction of the model search to a certain model subspace, and instead of directly imposing this restriction, we can still search in the full model space but penalize the projection distance to the model subspace.

Empirical studies in Part I include a broad array of real-world applications such as document classification, landmine detection, face recognition, brain signal classification, handwritten digit recognition and house price forecasting, where certain structure information about the input space can be gained from unlabeled data, related tasks, and domain knowledge. In all these experiments, our proposed methods for encoding input information significantly outperform a wide range of semi-supervised learning, multi-task learning, regularization and dimension reduction methods.

In Part II (Chapter 7 - Chapter 9), we focus on encoding output information into the learning process, mainly by error-correcting output codes. This part contains the following three chapters:

- In Chapter 7, we start by providing a composite likelihood view for a class of multi-label classification algorithms and then propose an improved pairwise output coding scheme. In existing pairwise decomposition methods, pairwise label comparisons are widely used as the subproblems to learn, and prediction for a testing sample is made by voting from all label comparisons. Inspired by the composite likelihood view, we instead choose pairwise label density estimations as the subproblems in our coding, and for combining subproblem decisions (i.e., decoding), we design a mean-field approximation that minimizes the notion of composite divergence and is potentially more robust to inaccurate estimation in subproblems.
- In Chapter 8, we begin to study problem-dependent output coding for multi-label classification, where the label encoding is learned from data instead of being predefined (as in pairwise decomposition). We propose multi-label output coding via canonical correlation analysis: the label dependency is characterized by the most predictable directions in the label space and extracted by canonical correlation analysis, and the output code is designed to include these predictable label directions as the redundant information to correct prediction errors. Predictions for the output code define a graphical model of labels with both Bernoulli potentials (from classifiers on the original labels) and Gaussian potentials (from regression on the predictable label directions). Decoding is performed by mean-field approximation.
- In Chapter 9, we first argue that, in addition to predictability, discriminability is also an im-

Table 1.1: Common Notation

Symbol	Definition
$\mathbf{D}$	The set of training examples
n	The number of training examples
p	The number of input variables (i.e., features)
q	The number of output variables (i.e., classes or labels)
m	The number of tasks
$\mathbf{x}$	The vector of input variables
$\mathbf{X}$	The $n \times p$ input matrix of training examples
y	The output variable
$\mathbf{y}$	The vector of output variables
$\mathbf{Y}$	The $n \times q$ output matrix of training examples
$\mathbf{w}$	The vector of model parameters
$\mathbf{W}$	The matrix of model parameters (usually defined in multi-task learning)
b	The intercept term for the model

portant property for multi-label output codes: encodings for different label vectors should be significantly different from each other, so that the codeword for the correct output will not be confused with incorrect ones, even under noisy predictions. To this end, we propose a max-margin formulation for output coding, which promotes both discriminability and predictability of the generated codewords. We then convert this formulation to a metric learning problem in the label space, but with an exponentially large number of constraints as encountered in structured prediction problems. Without a label structure for tractable inference, we use overgenerating techniques combined with the cutting plane method for optimization.

Empirical studies in Part II include a variety of multi-label prediction problems for music emotion identification, outdoor scene recognition, gene function classification, medical text categorization and email analysis, where certain hidden label dependency structure can be discovered and exploited. In all these experiments, our proposed multi-label output coding schemes lead to substantial improvements over a large collection of multi-label prediction algorithms.

Finally, in Chapter 10, we conclude the thesis and discuss future directions.

1.3 Common Notation

In Table 1.1 we show a list of common symbols in this thesis. Symbols that are not specifically defined in each chapter follow the definition in this table.

Chapter 2

Background

The work presented in this thesis is based on a broad array of techniques developed in machine learning and statistics. In this chapter, by reviewing related research areas, we provide background to understand key concepts in this thesis. For each research area, we first give an introduction and then present a survey of related work and highlight the connection to this thesis.

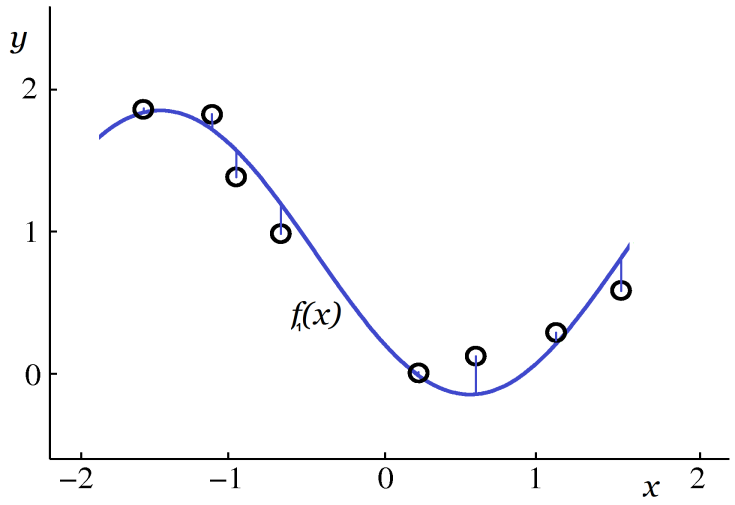
2.1 Regularization

2.1.1 Introduction

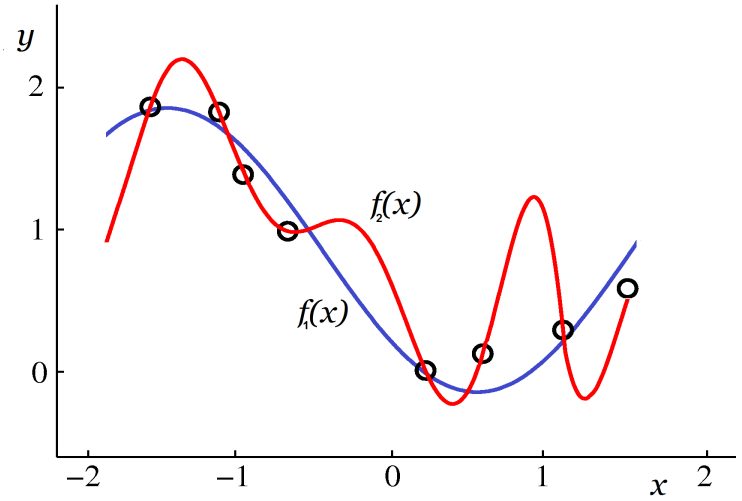
In supervised learning, our goal is to learn a prediction model (function) using a set of training examples. This is usually achieved by minimizing an empirical loss function on training examples, which measures the discrepancy between model predictions and observations in the training set.

Unfortunately, minimizing an empirical loss on training examples may lead to ***overfitting***. Consider the example shown in Fig. 2.1. In Fig. 2.1a, the blue curve represents a function $y = f_1(x)$ and black circles are training examples generated from this function (with random noise added). The vertical blue lines connecting training examples and the blue curve indicate the distance from examples to the curve. Our goal is to estimate this unknown function $y = f_1(x)$ from observations. If we define the empirical loss as the sum of squared (vertical) distance to all training examples, the blue curve $y = f_1(x)$ actually has a low empirical loss on the training set, as it is close to all examples. In this sense, it seems reasonable to minimize the empirical loss to find the true curve.

However, in Fig. 2.1b we show a red curve, representing another function $y = f_2(x)$. Surprisingly, this new function $f_2(x)$ perfectly passes through every training example and fits the training set even better than the true function $y = f_1(x)$. Note that the true function $f_1(x)$ does not fit training examples perfectly due to the existence of random noise. As a result, if we search for a curve by minimizing the sum of squared errors, we will choose $y = f_2(x)$ as represented by the red curve. This function is quite different from the true function $y = f_1(x)$, and thus if we use it to predict y on new samples generated from the true function, we will have high prediction errors.



(a) The true function and generated samples



(b) The true function and a function that overfits

Figure 2.1: Overfitting

As discussed above, *overfitting* describes the situation that minimizing the empirical loss on training samples leads to an estimated function that is significantly different from the true function and thus predicts poorly on new samples. How can we avoid overfitting? Intuitively, in Fig. 2.1b, function $f_2(x)$ is much more complicated than function $f_1(x)$, and according to the principle of *Occam's razor*, one should choose the simplest hypothesis when multiple hypotheses explain the observations similarly well. Formally, both $f_1(x)$ and $f_2(x)$ in Fig. 2.1b can be represented by polynomials, and a function like $f_2(x)$ generally has larger coefficients in its polynomial representation than a function like $f_1(x)$. As a result, controlling the complexity of the function (or more specifically the magnitude of the coefficients) can potentially rule out functions like $f_2(x)$.

Regularization formalizes this idea for preventing overfitting, in which we minimize the empirical loss on the training set plus a regularization penalty that measures the complexity of the model. Consider a set of n training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$, where each $\mathbf{x}_i \in R^p$ is a p-dimensional feature vector and each y_i is a scalar response variable. In regression we have $y_i \in R$ and in binary classification we have $y_i \in \{0, 1\}$. A general formulation for regularization is:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathbf{x}_i, \mathbf{w}) + \lambda J(\mathbf{w}) \tag{2.1}$$

where $L()$ is the empirical loss measuring the discrepancy of model prediction on each training example $(\mathbf{x}_i, y_i)$, and $J()$ is a regularization penalty on the model parameters $\mathbf{w}$. A classical example of regularization is ridge regression [TA77], which fits a linear regression model:

$$\underset{\mathbf{w} \in R^p, b}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i - b)^2 + \lambda ||\mathbf{w}||_2^2 \tag{2.2}$$

where $\mathbf{w}$ is the p-dimensional vector of model coefficients, b is the intercept term, $||\mathbf{w}||_2$ is the ℓ_2-norm penalty on $\mathbf{w}$, and λ is the regularization parameter controlling the strength of regularization. In this equation, we minimize the sum of squared errors on training examples plus the regularization penalty $||\mathbf{w}||_2^2$ that measures the magnitude of coefficients (in terms of ℓ_2 norm). As a result, ridge regression as (2.2) will control the empirical loss as well as shrink the model coefficients.

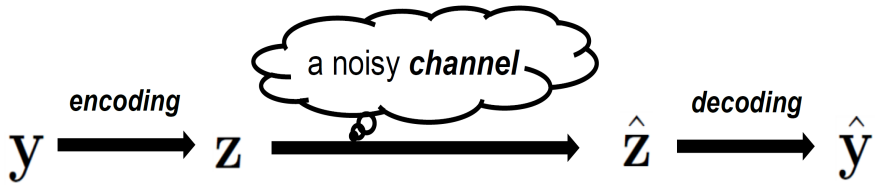
Both the empirical loss and the penalty term in (2.2) can be replaced by other functions. For example, if the ℓ_2 norm penalty is replaced by the ℓ_1 norm, we have the lasso problem [Tib96]:

$$\underset{\mathbf{w} \in R^p, b}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i - b)^2 + \lambda ||\mathbf{w}||_1 \tag{2.3}$$

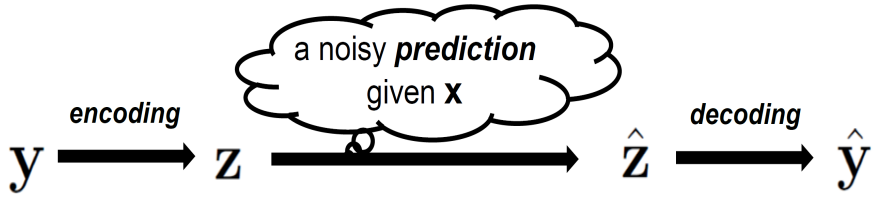
Penalizing the ℓ_1 norm $||\mathbf{w}||_1$ will also shrink the model coefficients as penalizing $||\mathbf{w}||_2^2$ does, but penalizing $||\mathbf{w}||_1$ also leads to sparse estimation: some coefficients of the estimated $\mathbf{w}$ will be shrunk to exactly zeros. Similarly, the sum of squared errors in (2.2) and (2.3) can be replaced by other empirical loss functions such as the hinge loss in SVM [BGV92, CV95] for classification.

2.1.2 Survey

Initially proposed to solve ill-posed problems [TA77], regularization is a principled way to control model complexity and prevent overfitting in learning and has been the focus of statistics and machine learning for decades [HTF01]. Regularization seeks a trade-off between fitting the observations and reducing the model complexity, which is justified by the minimum description length (MDL) principle in information theory [Ris78] and the bias-variance dilemma in statistics [Sul86]. Classical examples of regularization include ridge regression [TA77] in statistics and support vector machines [BGV92, CV95] in machine learning, which correspond to minimizing either squared loss or hinge loss with ℓ_2 regularization. Since the introduction of lasso [Tib96], ℓ_1 regularization has become very popular for learning in high-dimensional spaces. A fundamental assumption of



(a) Channel coding



(b) Error-correcting output codes

Figure 2.2: Channel coding and error-correcting output codes

ℓ_1 regularization is model sparsity. Penalizing the ℓ_1 norm of model coefficients leads to sparse estimation, which corresponds to selecting relevant features and have the advantage of being easy to interpret and good generalization ability in high-dimensional problems.

Recently, designing informative regularization has been one of the main approaches for multi-task learning [AEP07], transfer learning [RNK06] and semi-supervised learning [BNS06]. The key idea is to encode information from related tasks, source domains and unlabeled data into the regularization penalty. Also, additional structure assumptions on model coefficients can be imposed via ℓ_1 regularization. Fused lasso [TSR+05] includes an ℓ_1 penalty on the differences of successive model coefficients and leads to piecewise constant estimations. Elastic net [ZH05] combines ℓ_1 and ℓ_2 norms to address the issue of correlated coefficients, which is also the focus of OSCAR [BR08]. Group lasso [YL06] defines group structures and adds further restrictions on the model sparsity: model coefficients in the same group tend to be set to zero together. Structured sparsity [HZM09] generalizes the group lasso to allow other structured assumptions on model sparsity, i.e., not all sparse patterns are equally likely and we prefer some of them to others.

Regularization is the primary methodology used in Part I (Chapter 3-Chapter 6) of this thesis: we encode information about the input space by designing a variety of regularization penalties.

2.2 Error-Correcting Output Codes

2.2.1 Introduction

The idea of error-correcting output codes comes from ***channel coding*** [CT91, CF07]. In channel coding, our goal is to reliably transmit a message $\mathbf{y}$ through a noisy channel that can alter the message. In order to achieve reliable communication, as shown in Fig. 2.2a, we *encode* the message $\mathbf{y}$ into a redundant codeword $\mathbf{z}$, which is sent through the noisy channel. The received

codeword $\hat{\mathbf{z}}$ may be contaminated during transmission, but it is possible to correctly infer the message $\hat{\mathbf{y}}$ using the redundancy encoded in the codeword. Consider a message $\mathbf{y} = (0, 1, 0)$ and a very simple channel coding scheme, for illustration purpose, to repeat each bit in the message, e.g., $\mathbf{z} = (0, 0, 0, 1, 1, 1, 0, 0, 0)$. Suppose we transmit $\mathbf{z}$ through a noisy channel that will only alter up to one bit, e.g., $\hat{\mathbf{z}} = (0, 0, 0, 1, 1, 1, 0, 0, \mathbf{1})$. Upon receiving $\hat{\mathbf{z}}$, the receiver can correctly infer the message $\hat{\mathbf{y}} = (0, 1, 0)$ given the coding scheme and the fact that the channel will not alter more than one bit. Real-world channel codes are usually much more sophisticated than simply repeating the message, but the idea of using the encoded redundancy for error correction remains.

In ***error-correcting output codes*** (ECOCs), as shown in Fig. 2.2b, predicting an output variable (or a set of output variables) $\mathbf{y}$ is viewed as transmitting a message through a noisy channel, and the prediction error corresponds to the transmission error. As a result, to correct prediction errors, we encode the output variable into a redundant codeword $\mathbf{z}$, learn to predict the codeword, and for a testing example $\mathbf{x}$, we predict its codeword as $\hat{\mathbf{z}}$ and decode the prediction to recover the output variable. ECOCs are mainly designed for multi-class classification problems. For example, consider a 4-class problem and a set of n training example $\{\mathbf{x}_i, y_i\}_{i=1}^n$ where $\mathbf{x}_i \in R^p$ is a feature vector and $y_i \in \{c_1, c_2, c_3, c_4\}$ is the class assignment for example i. For illustration, consider ***one-vs-one decomposition***, which is one of the most popular ECOCs. In a 4-class problem, the encoded codeword $\mathbf{z}$ is defined as the collection of simpler one-vs-one decision subproblems:

$$\mathbf{z} = (c_1 \text{ vs } c_2,\ c_1 \text{ vs } c_3,\ c_1 \text{ vs } c_4,\ c_2 \text{ vs } c_3,\ c_2 \text{ vs } c_4,\ c_3 \text{ vs } c_4) \tag{2.4}$$

For a class $y \in \{c_1, c_2, c_3, c_4\}$, the subproblem c_i vs c_j takes value 1 if $y = c_i$, takes value -1 if $y = c_j$, and takes value 0 otherwise. Therefore, the encodings for the four classes are:

$$\begin{aligned}
\mathbf{z}(c_1) &= (1, 1, 1, 0, 0, 0) \\
\mathbf{z}(c_2) &= (-1, 0, 0, 1, 1, 0) \\
\mathbf{z}(c_3) &= (0, -1, 0, -1, 0, 1) \\
\mathbf{z}(c_4) &= (0, 0, -1, 0, -1, -1)
\end{aligned}$$

For predicting the codeword, a binary classifier is learned for each subproblem c_i vs c_j using the given training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$, but only those belonging to c_i or c_j are used. The classifier can only output 1 (if c_i is predicted) or -1 (if c_j is predicted), but not 0. For a new example $\mathbf{x}$, all classifiers together predict a codeword $\hat{\mathbf{z}}$, which only contains 1 and -1, e.g, $\hat{\mathbf{z}} = (1, 1, 1, 1, -1, 1)$.

For decoding a predicted codeword $\hat{\mathbf{z}}$, one can compare it to the codeword of each class and use certain criterion to decide the class, e.g., choosing the class whose codeword is closest to $\hat{\mathbf{z}}$ in terms of Hamming distance. For example, given $\hat{\mathbf{z}} = (1, 1, 1, 1, -1, 1)$, the closest class codeword is $\mathbf{z}(c_1) = (1, 1, 1, 0, 0, 0)$, with a Hamming distance 3. Therefore, the class assignment is c_1.

2.2.2 Survey

Error-correcting output codes (ECOCs) offer a general framework for decomposing a multiclass classification problem into a number of binary classification problems [DB95]. Via ECOCs, a

multiclass problem can be solved using binary classifiers. More importantly, the binary problems provide a redundant representation of the multiclass problem. As a result, prediction errors can be corrected using such redundancy, as studied in channel coding and error-correcting codes [CT91].

The *encoding* of ECOCs decomposes the multiclass problem into a set of binary problems, and defines the *codeword* as the composition of the outcomes of the binary problems. Popular ECOC decomposition strategies include one-versus-all [DB95], one-versus-one [HT97], random partitions [ASS01], and partitions obtained by problem-dependent heuristic search [CS02, PRV06]. The *decoding* of ECOCs decides the class of an example given the prediction on its codeword. This is achieved by examining all the q candidate classes (for a q-class problem) and choosing the class that minimizes a distance function [DB95], minimizes a loss function [ASS01], maximizes a probability function [HT97, PPF04] or optimizes certain other criteria [EPR10] w.r.t. the predicted codeword.

Error-correcting output codes are the primary methodology used in Part II (Chapter 7-Chapter 9) of this thesis: we design error-correcting output codes to improve the prediction performance of *multi-label* classification, by encoding information about the output space.

2.3 Multi-Label Classification

2.3.1 Introduction

In ***multi-label classification*** problems, an object can be associated with a predefined set of multiple labels, and we want to simultaneously perform classification for all the labels. For example, an image can be associated with multiple scenes (such as rivers, forest, desert, beach, sunset, mountain, and urban), and in scene recognition we want to detect the appearance of different scenes in an image. The key difference between this problem and a multiclass classification problem is that any subset of labels can be true, instead of only one class being true. In this sense, a q-label problem has 2^q possible outputs, as opposed to q possible outcomes in a q-class problem. A key concept in multi-label classification is ***label dependency***: assignments to labels are usually not independent. For example, in scene recognition, rivers are likely to appear together with a forest but highly unlikely to co-occur with a desert. This is one of the main motivations for research in multi-label classification: independently learning and predicting each label may not be optimal.

In the last decade a wide variety of multi-label classification algorithms have been designed. For illustration, we review *pairwise label ranking* [HFCB08] based on one-vs-one decomposition, which is similar to the one-vs-one decomposition for multiclass ECOCs discussed in Section 2.2.1. By introducing this method, we can highlight the difference between multi-label and multiclass problems. As in the multiclass case, we consider a 4-label problem and a set of n training examples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ where $\mathbf{x}_i \in R^p$ is a feature vector and $\mathbf{y}_i = (l_1^i, l_2^i, l_3^i, l_4^i)^T \in \{0,1\}^4$ is the label vector. For a label vector $\mathbf{y} = (l_1, l_2, l_3, l_4)^T$, we consider the following one-vs-one subproblems:

$$\mathbf{z} = (l_1 \text{ vs } l_2,\ l_1 \text{ vs } l_3,\ l_1 \text{ vs } l_4,\ l_2 \text{ vs } l_3,\ l_2 \text{ vs } l_4,\ l_3 \text{ vs } l_4) \tag{2.5}$$

The subproblem l_i vs l_j takes value 1 if $l_i = 1$ and $l_j = 0$, and takes value -1 if $l_i = 0$ and

$l_j = 1$. Examples that l_i and l_j are both 1 or both 0 are excluded from this subproblem, because the subproblem is designed to compare two labels. For prediction, a binary classifier is learned for each subproblem l_i vs l_j using training examples where either $l_i = 1$ or $l_j = 1$ (but not both). For a new example $\mathbf{x}$, the classifier for l_i vs l_j will output a preference order: either $l_i > l_j$ or $l_j > l_i$.

Now we need to combine subproblem decisions to make a multi-label classification. For a q-label problem, it is considered computationally intractable to compare classifier outputs to each of 2^q possible label vectors. As a result, certain heuristics are usually used. In pairwise label ranking, the classifier for each subproblem l_i vs l_j contributes a vote to the wining label in the pair. A total ranking order among labels is obtained by counting how many votes each label obtains.

For a new example, pairwise label ranking is only designed to output a total order over labels. To obtain classification for labels, a simple strategy is to use a threshold, e.g., labels that win more than 50% of its comparisons are assigned 1. There are also adaptive calibration techniques, e.g., [FHMB08], which we will study and empirically compare to in Part II of this thesis.

2.3.2 Survey

Multi-label classification, in which each example is associated with a set of multiple labels, has received considerable attention from the machine learning community [TKV10]. Problem decomposition techniques decompose the multi-label classification problem into a set of simpler and easier-to-learn subproblems, estimate prediction models for the subproblems, and then combine the predictions from subproblems to make the final classification. Popular choices of subproblems include one-vs-all decomposition (i.e., the relevance of each individual label) [RK04], one-vs-one decomposition (i.e., the pairwise comparison between any two labels) [Fur02, HFCB08], a combination of both one-vs-all and one-vs-one decompositions [FHMB08], and conditional relevance of one label given other labels (e.g., classifier chain methods) [RPHF09, DCH10].

Loosely speaking, problem decomposition methods can be viewed as error-correcting output codes for multi-label prediction, although encoding and decoding might not be well defined in these methods. To our knowledge, multi-label compressed sensing [HKLZ09] is perhaps the first output code formally defined for multi-label prediction, where encoding is based on random projections, and decoding relies on sparse reconstruction techniques in compressed sensing [Can06, Don06].

Instead of decomposing and learning subproblems individually, another way to handle multi-label prediction is to jointly model the dependency among all the labels and features using a single graphical model. Research along this direction includes specifying a conditionally trained undirected graphical model (i.e., CRF) [GM05] or learning the structure of Bayesian networks from data [ZZ10, ZSM+11] to capture the joint label relation (conditioned on features). In addition, researchers have also concentrated on adapting specific models (such as decision trees, SVMs, KNNs, neural networks, and boosting) to directly produce multi-label predictions. An overview on a variety of methods for adapting specific models can be found in [TKV10].

A related area to multi-label classification is label ranking [TKV10, GV10, DMS04, HFCB08], where the goal is to predict the total order of labels given an example. Multi-label classification and label ranking are closely related tasks: a ranking order of labels can be turned into a classi-

fication if an additional threshold is obtained [FHMB08], and on the other hand, a classification model can provide a ranking order if it can offer continuous outputs on label relevance. Also, both multi-label classification and label ranking can benefit from capturing the label dependency. As a result, methods and models proposed for label ranking can be adapted to perform multi-label classification, and vice versa.

In Part II (Chapter 7-Chapter 9) of this thesis we design output codes for multi-label classification. Many existing multi-label prediction methods, such as those in [HFCB08, FHMB08, ZZ10, HKLZ09, RPHF09, TV07], are studied as competitors in our experiments.

2.4 Semi-Supervised Learning

Motivated by the limited number of labeled examples and wide availability of unlabeled data in many applications, semi-supervised learning studies the use of unlabeled data in supervised learning problems and has recently become a very active research area in machine learning [Zhu06, CSZ06]. The key question in semi-supervised learning is what can we gain from unlabeled data and under what assumptions and conditions this information can help supervised learning. For example, a very straightforward method for semi-supervised learning based on "change of representation" [Zhu06, CSZ06] is to apply an unsupervised learning algorithm (e.g., principal component analysis) on unlabeled data to learn an alternative representation (e.g., principal components), and then perform supervised learning with labeled examples using the new representation. This method is effective if the representation obtained by unsupervised learning contains information needed for the prediction problem, which, however, is not always guaranteed.

During the last decade, many semi-supervised learning methods have been proposed in the following directions: 1) co-training [BM98] and related methods try to iteratively label unlabeled samples to augment labeled examples; 2) generative methods [NMTM00] are designed to model the generative distribution of data, for which unlabeled data can provide important information; 3) methods based on low-density separation [Joa99, SK06] assume that the decision boundary should lie close to low-density regions, and unlabeled data can be used to estimate the density of the input space; 4) graph-based methods [BC01, ZGL03] connect labeled examples and unlabeled samples into a single graph and apply regularization on the graph; and 5) change of representation methods [CSZ06] learn a new representation from unlabeled data and use it for supervised learning.

In most of the above methods, we assume that the unlabeled data belong to the same target classes or share the same generative distribution with the labeled examples. As indicated in [RBLN07], however, unlabeled data in real-world applications do not necessarily follow this assumption, and methods based on this assumption may have limited applicability in practice. As a result, some algorithms avoid directly using unlabeled examples in model training and instead focus on finding a useful representation from unlabeled data even when the unlabeled data are not from the target classes or generative distribution of labeled examples [RBLN07, AZ05].

In Chapter 3, we propose a semi-supervised learning method for text classification, where even seemingly irrelevant unlabeled text can be used to improve any specific prediction task.

2.5 Multi-Task Learning

In many situations we need to learn multiple different but similar tasks, e.g., detecting landmines in different landmine fields or recognizing faces for different subjects. Notice that multiple tasks usually share the same input space, e.g., the input for landmine detection has a unified format, and images for face recognition have all been properly sized, scaled and transformed. Also, tasks usually have some underlying commonality, e.g., the basic principle of landmines or the layout of human faces. As a result, learning multiple tasks jointly may be more effective than learning each task independently, especially if each task has only a limited number of training examples.

In ***multi-task learning***, the key question is how to share knowledge among tasks. A brute force approach for knowledge sharing is to pool the training examples from all tasks together and learn them as one single task. Depending on the situation, this method may perform very poorly. In landmine detection, different landmine fields can present different geographical conditions or may even have different type of landmines. In face recognition, pooling training examples together we may learn a model for detecting a general human face but not detectors for individual subjects.

Multi-task learning has been an active research area for more than a decade [Bax95, TO96, Car97]. For joint learning of multiple tasks, connections need to be established to couple related tasks. One approach is to find the feature structure shared by tasks. Along this direction, researchers propose to infer the feature structure by performing covariance estimation [AEP07, AMPY07], principal components [AZ05, CTLY09] and independent components [ZGY06] on the model parameters, to select a common subset of features [BV98, OTJ09], as well as to use shared hidden nodes in neural networks [Bax95, Car97]. On the other hand, assuming all tasks are equally similar is risky. Researchers recently began to directly infer the relatedness of tasks and use this structure to couple the prediction models of different tasks. Efforts towards this direction include using mixtures of Gaussians [BH03] or Dirichlet processes [XLCK07] to model task groups, encouraging clustering of tasks via a convex regularization penalty [JBV08], identifying "outlier" tasks by robust t-processes [YTY07], and inferring a task similarity matrix [BCW08, YCY+07, ZY10].

In Chapter 4 of this thesis, we propose a matrix-normal regularization penalty for multi-task learning to systematically discover and encode both feature structures and task relations.

2.6 Matrix Normal Distributions

The ***matrix-variate normal distribution*** is one of the most widely studied matrix-variate distributions [GN99, Daw81, Dut99]. Consider an $m \times p$ matrix $\mathbf{W}$. Since we can vectorize $\mathbf{W}$ to be a $mp \times 1$ vector, the normal distribution on a matrix $\mathbf{W}$ can be considered as a multivariate normal distribution on a vector of mp dimensions. However, such an ordinary multivariate distribution ignores the special structure of $\mathbf{W}$ as an $m \times p$ matrix, and as a result, the covariance characterizing the elements of $\mathbf{W}$ is of size $mp \times mp$. This size is usually prohibitive for modeling and estimation. To utilize the structure of $\mathbf{W}$, matrix normal distributions assume that the $mp \times mp$

covariance can be decomposed as the Kronecker product $\mathbf{\Sigma} \otimes \mathbf{\Omega}$, and elements of $\mathbf{W}$ follow:

$$Vec(\mathbf{W}) \sim N(Vec(\mathbf{M}), \mathbf{\Sigma} \otimes \mathbf{\Omega}) \tag{2.6}$$

where $\mathbf{\Omega}$ is a $m \times m$ positive definite matrix indicating the covariance between rows of $\mathbf{W}$, $\mathbf{\Sigma}$ is a $p \times p$ positive definite matrix indicating the covariance between columns of $\mathbf{W}$, $\mathbf{\Sigma} \otimes \mathbf{\Omega}$ is the Kronecker product of $\mathbf{\Sigma}$ and $\mathbf{\Omega}$, $\mathbf{M}$ is a $m \times p$ matrix containing the expectation of each element of $\mathbf{W}$, and Vec is the vectorization operation which maps a $m \times p$ matrix into a $mp \times 1$ vector. Due to the decomposition of the full covariance as the Kronecker product $\mathbf{\Sigma} \otimes \mathbf{\Omega}$, the number of entries in the covariance structure is dramatically reduced to $m^2 + p^2$ from $m^2 p^2$. The matrix-variate normal distribution of an $m \times p$ matrix $\mathbf{W}$, parameterized by the mean $\mathbf{M}$, row covariance $\mathbf{\Omega}$ and column covariance $\mathbf{\Sigma}$, has a compact log-density [GN99]:

$$\log P(\mathbf{W}) = -\frac{mp}{2}\log(2\pi) - \frac{p}{2}\log(|\mathbf{\Omega}|) - \frac{m}{2}\log(|\mathbf{\Sigma}|) - \frac{1}{2}tr\{\mathbf{\Omega}^{-1}(\mathbf{W}-\mathbf{M})\mathbf{\Sigma}^{-1}(\mathbf{W}-\mathbf{M})^T\} \tag{2.7}$$

where $|\ |$ is the determinant of a square matrix, and $tr\{\}$ is the trace of a square matrix.

Consider a set of n samples $\{\mathbf{W}_i\}_{i=1}^n$ where each $\mathbf{W}_i$ is a $m \times p$ matrix generated by a matrix-variate normal distribution as eq. (2.7). The ***maximum likelihood estimation (MLE)*** of mean $\mathbf{M}$ is [Dut99]:

$$\hat{\mathbf{M}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{W}_i \tag{2.8}$$

The MLE estimators of $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ are solutions to the following system:

$$\begin{cases} \hat{\mathbf{\Omega}} &= \frac{1}{np}\sum_{i=1}^{n}(\mathbf{W}_i - \hat{\mathbf{M}})\hat{\mathbf{\Sigma}}^{-1}(\mathbf{W}_i - \hat{\mathbf{M}})^T \\ \hat{\mathbf{\Sigma}} &= \frac{1}{nm}\sum_{i=1}^{n}(\mathbf{W}_i - \hat{\mathbf{M}})^T\hat{\mathbf{\Omega}}^{-1}(\mathbf{W}_i - \hat{\mathbf{M}}) \end{cases} \tag{2.9}$$

It is efficient to iteratively solve (2.9) until convergence, known as the "flip-flop" algorithm [Dut99].

Also, $\hat{\mathbf{\Omega}}$ and $\hat{\mathbf{\Sigma}}$ are not identifiable and solutions for maximizing the log density in eq. (2.7) are not unique. If $(\mathbf{\Omega}^*, \mathbf{\Sigma}^*)$ is an MLE estimate for the row and column covariances, for any $\alpha > 0$, $(\alpha\mathbf{\Omega}^*, \frac{1}{\alpha}\mathbf{\Sigma}^*)$ will lead to the same log density and thus is also an MLE estimate. This can be seen from the definition in eq. (2.6), where only the Kronecker product $\mathbf{\Sigma} \otimes \mathbf{\Omega}$ is identifiable. In practice, the flip-flop algorithm (2.9) is usually used to find a stationary solution.

Matrix normal distributions have also been studied in the Bayesian literature. For example, it can serve as a prior for Bayesian variable selection in multivariate regression [BV98], where MCMC is used for sampling from the resulting posterior. Recently, matrix normal distributions have also been used in nonparametric Bayesian approaches, especially in learning Gaussian Processes (GPs) for multi-output prediction [BCW08] and collaborative filtering [YCY+07, YLZG09].

In Chapter 4 of this thesis, we use the matrix-normal distribution as a building block to design our multi-task regularization framework.

2.7 Compressed Sensing

Compressive sampling [Can06] or compressed sensing [Don06] was recently developed for signal acquisition and reconstruction, and has received considerable attention [BCNV08]. According to research in this direction, one can successfully acquire a signal (e.g., an image) from many fewer measurements than required by the Nyquist-Shannon sampling theory. The key assumption is that signals like natural images are compressible, i.e., nearly sparse in a compression domain. Under this assumption, signal acquisition given a few linear measurements on the unknown signal can be achieved by optimizing the sparsity (via a convex relaxation using ℓ_1 norm) of the reconstructed signal in a compression domain while satisfying the measurement constraints:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & ||\mathbf{W}\boldsymbol{\beta}||_1 \\ \text{subject to} \quad & \boldsymbol{\Phi}\boldsymbol{\beta} = \mathbf{y} \end{aligned} \tag{2.10}$$

Here $\boldsymbol{\beta}$ is a $p \times 1$ vector representing the candidate signal. $\mathbf{W}$ is a known compression operation. The $n \times p$ matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n]^T$ is the sensing (projection) matrix [CW08, Don06], whose n rows correspond to n linear measuring or sensing operations conducted on the unknown signal. A common choice of $\boldsymbol{\Phi}$ is random projections. The $n \times 1$ vector $\mathbf{y}$ contains n measurements of the unknown signal. The idea is to find a compressible signal that is consistent with the measurements.

Recently, compressive sampling has attracted increasing interest in the machine learning community. In [JC07], researchers use sparse Bayesian regression and active learning to solve the CS problem in eq. (2.10) and "learn" the optimal projection matrix $\boldsymbol{\Phi}$. Authors in [CJS, ZLW07] consider classification and regression problems, respectively, where data are compressed and not directly observable, e.g., only random projections of the data are available.

Chapter 5 of this thesis is an application of the compressed sensing idea in machine learning: model coefficients can be compressed before being penalized, and model sparsity is only required in a compressed domain rather than the original space. The focus of this chapter is to study the use of different compression operations in ℓ_1 regularized learning, which incorporates domain knowledge about the input space and imposes more appropriate inductive bias in the learning process.

2.8 Canonical Correlation Analysis

Canonical correlation analysis [Hot35, Hot36] is a classical tool for modeling linear associations between two sets of variables. Consider a set of p variables $\mathbf{x} \in \mathcal{X} \subseteq R^p$ and another set of q variables $\mathbf{y} \in \mathcal{Y} \subseteq R^q$. In addition, we have a set of n training samples: $\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$, where $\mathbf{X}$ and $\mathbf{Y}$ are matrices of size $n \times p$ and $n \times q$, respectively. Canonical correlation analysis finds a pair of projection directions $\mathbf{u} \in R^p$ and $\mathbf{v} \in R^q$ such that the *correlation* between the pair of projected variables $\mathbf{u}^T\mathbf{x}$ and $\mathbf{v}^T\mathbf{y}$ (also called *canonical variates*) is maximized. Given the training samples $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$, this maximization problem can

be expressed as the following[1]:

$$\underset{\mathbf{u}\in R^p,\mathbf{v}\in R^q}{\operatorname{argmax}} \quad \frac{\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v}}{\sqrt{(\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u})(\mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v})}} \tag{2.11}$$

Since a rescaling of $\mathbf{u}$ and $\mathbf{v}$ will not change the objective value, two constraints can be added:

$$\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u} = 1 \tag{2.12}$$
$$\mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v} = 1 \tag{2.13}$$

As a result, maximizing the correlation between the canonical variates can be rewritten as the following constrained optimization problem:

$$\begin{aligned} \underset{\mathbf{u}\in R^p,\mathbf{v}\in R^q}{\operatorname{argmax}} \quad & \mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} \\ \text{s.t.} \quad & \mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u} = 1 \\ & \mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v} = 1 \end{aligned} \tag{2.14}$$

By formulating the Lagrangian of the above convex optimization problem, the KKT conditions lead to the following generalized eigenproblems [HSSt04] on $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{X}^T\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{u} = \lambda\mathbf{X}^T\mathbf{X}\mathbf{u} \tag{2.15}$$
$$\mathbf{Y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{v} = \lambda\mathbf{Y}^T\mathbf{Y}\mathbf{v} \tag{2.16}$$

The formulations (2.15) and (2.16) provide more than one pair of projection vectors $(\mathbf{u}, \mathbf{v})$. By solving the first d principal eigenvectors ($d \leq \min(p,q)$), we can obtain d pairs of projection vectors: $\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^d$. These projection vectors successively maximize the correlation between the resulting canonical variates, with generalized orthogonality constraints satisfied:

$$\begin{aligned} \mathbf{u}_k^T\mathbf{X}^T\mathbf{X}\mathbf{u}_j = 0, \quad & \forall k \neq j \\ \mathbf{v}_k^T\mathbf{Y}^T\mathbf{Y}\mathbf{v}_j = 0, \quad & \forall k \neq j \end{aligned}$$

To summarize, given a set of observations $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$, canonical correlation analysis will find pairs of projection directions $\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^d$ to maximize the correlation between projected variables. We denote this process as:

$$\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^d \leftarrow \text{CCA}(\mathbf{X}, \mathbf{Y}) \tag{2.17}$$

A recent overview of CCA with application to learning problems is given in [HSSt04]. Several variants of CCA have been recently proposed: sparse CCA [WTH09, HST09] enforces the sparsity of projection vectors and leads to interpretable models; kernel CCA [FL01, HSSt04] handles nonlinear associations between variables; a nonparametric Bayesian extension of CCA, sparse infinite CCA [RD09], shows good predictive power.

In Chapter 8 of this thesis, CCA is used as a building block to design a multi-label output code, where predictability of the generated codewords is optimized by CCA-based encoding.

[1]For simplicity, one usually assumes that data have been centralized such that columns of $\mathbf{X}$ and $\mathbf{Y}$ have zero means.

Part I

Encoding Input Information

Chapter 3

Learning with Word Correlation from Irrelevant Unlabeled Text

3.1 Overview

In this chapter [ZSD08], we consider encoding the covariance and correlation structure of the input space into learning and propose a semi-supervised learning method for text classification. We show that the semantic correlation of words, as an intrinsic structure of the language, can be inferred from unlabeled text and provides structure information that can be incorporated into the regularization of any specific text learning task. As a result, even seemingly irrelevant unlabeled text, such as text downloaded from the Internet, can be used to improve text classification.

To start, we first identify the semantic correlation of words[1] as a structure of the input space that can be transferred from unlabeled text. Consider a document classification problem, where we have only one positive example containing two words {gasoline, truck} and one negative example containing two words {vote, election}. Most people will agree that a new document with words {gallon, vehicle} should be classified as positive, although *gallon* and *vehicle* have never been observed in the training set. The key reason is that *gallon* is the unit of *gasoline*, and *truck* is a type of *vehicle*. Since the classifier should have positive weights on *gasoline* and *truck* (as they appear in the only positive example), *gallon* and *vehicle* are likely to receive positive weights, too. The *semantic correlation of words* corresponds to a correlation structure of the model coefficients and provides a strong inductive bias in the model space. Also, this is an intrinsic structure of the language and thus will not change dramatically even in irrelevant unlabeled text.

We propose a method to infer the semantic word correlation from seemingly irrelevant unlabeled text and incorporate it into learning of any specific task. We first extract a large number of latent topics from unlabeled text, by repeatedly applying bootstrapping and topic modeling. We then infer the word correlation from the word composition of the extracted topics. The resulting semantic correlation structure is used in ℓ_2-regularization for learning any specific task (as the

[1] We consider the bag-of-word feature space for simplicity. The proposed method can also be applied to n-gram feature spaces, and in this case, the semantic correlation of words is replaced by the semantic correlation of n-grams.

correlation in the Gaussian prior), and helps to control the joint shrinkage of model parameters in a semantically meaningful way. In an empirical study, we construct 190 different text classification tasks from a real-world benchmark, and the unlabeled documents are a mixture from all these tasks. We test the ability of various algorithms to use the mixed unlabeled text to enhance all classification tasks. Empirical results show that the proposed approach is a reliable and scalable method for semi-supervised learning, regardless of the fact that for any specific prediction task, the majority of the unlabeled data is irrelevant.

3.2 Semantic Correlation: Transferable Structure from Unlabeled Text

3.2.1 Latent Topics and Semantic Structure

Latent topics extracted from unlabeled text might be irrelevant to a particular classification task, but the composition of these topics in terms of word distribution reveals information about the semantic structure of the language. Assume a latent topic model [Hof99, BNJ03] of the word space $\mathbf{X}$, or more generally, a latent variable model characterizing the input space $\mathbf{X}$:

$$\mathbf{x} = \mathbf{A}\mathbf{z} \tag{3.1}$$

where $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p]^T$ is the p-dimensional vector of input variables, and $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k]^T$ represents latent variables in the k-dimensional latent space $\mathbf{Z}$. $\mathbf{A}$ is a $p \times k$ matrix, representing a generative process of words from a probabilistic view or a set of linear projections from a deterministic view. For a latent topic model, $\mathbf{x}$ corresponds to the bag-of-words vector of a document divided by the document length, $\mathbf{z}$ is the distribution of k latent topics in the document, and $\mathbf{A}$ is the distribution of p words in k latent topics. Various models can be represented by this formula, including principal component analysis, independent component analysis, sparse coding, and non-negative matrix factorization. Note that eq. (3.1) does not uniquely identify the decomposition $\mathbf{A}$ and $\mathbf{z}$, and additional constraints and assumptions are usually added in a specific model. For example, the orthogonality constraint and unit-length constraint are applied to columns of $\mathbf{A}$ in principal component analysis to enforce that each column of $\mathbf{A}$ is a basis of the transformed space, and the non-negativity constraint and sum-to-one constraint are added to columns of $\mathbf{A}$ in latent topics models so that each column represents the distribution of words in one latent topic.

Different documents have different topic distributions, $\mathbf{z}$, and thus different word distributions, $\mathbf{x}$, but $\mathbf{A}$ can be considered an invariant structure of the language. Each p-dimensional column vector of $\mathbf{A}$ denotes the word distribution in a latent topic and serves as an "observation" in the p dimensional word space, indicating the semantic roles of p words in this topic. Given a large set of k latent topics represented by k p-dimensional vectors $\{\mathbf{a}_{(,1)}, \mathbf{a}_{(,2)}, \ldots, \mathbf{a}_{(,k)}\}$, we can define the semantic covariance of p words as follows. Let $\mathbf{A}$ denote the matrix formed by treating each vector $\mathbf{a}_{(,t)}, t = 1, 2, \ldots, k$ as a column, and let $\mathbf{a}_{(i,)}$ and $\mathbf{a}_{(i,t)}$ denote a row vector and an element of this

matrix, respectively. The *semantic covariance* of word i and word j is defined as:

$$cov_s(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{k}\sum_{t=1}^{k}(\mathbf{a}_{it} - \overline{\mathbf{a}}_{(i,)})(\mathbf{a}_{jt} - \overline{\mathbf{a}}_{(j,)}) = \frac{1}{k}\sum_{t=1}^{k}\mathbf{a}_{it}\mathbf{a}_{jt} - \overline{\mathbf{a}}_{(i,)}\overline{\mathbf{a}}_{(j,)} \tag{3.2}$$

where $\overline{\mathbf{a}}_{(i,)}$ is the mean of the ith row in $\mathbf{A}$. Naturally, the *semantic correlation* is:

$$corr_s(\mathbf{x}_i, \mathbf{x}_j) = \frac{cov_s(\mathbf{x}_i, \mathbf{x}_j)}{\sqrt{cov_s(\mathbf{x}_i, \mathbf{x}_i)cov_s(\mathbf{x}_j, \mathbf{x}_j)}} \tag{3.3}$$

Intuitively, the semantic correlation of two words represents the correlation of their contributions to the composition of different latent topics.

3.2.2 Comparing Semantic Correlation and Data Correlation

Suppose we observe a set of n documents in the word space $\mathbf{X}$, denoted by an $n \times p$ data matrix $\mathbf{D}_X$ where each document corresponds to a p-dimensional vector of counts in the bag-of-words representation. We refer to the correlation between words computed directly from the data matrix $\mathbf{D}_X$ as the *data correlation*. According to eq. (3.1), documents from different sources may have distinct topic distributions $\mathbf{z}$ and thus different word distributions $\mathbf{x}$ in the observation space, and this will lead to different word correlations calculated from observation matrix $\mathbf{D}_X$. As a result, data correlation inferred from unlabeled text may not be useful if the unlabeled text is irrelevant to the prediction task we want to improve.

Here we show why we expect the data correlation to have limited use across distinct sources, while we expect the semantic correlation to be transferable. Consider the latent variable model in eq. (3.1), which relates $\mathbf{A}$ to data space $\mathbf{X}$. We focus on semantic covariance and data covariance, and assume that the bag-of-words vector is divided by the length of the document so that it corresponds to $\mathbf{x}$ in eq. (3.1). From eq. (3.1), an input variable $\mathbf{x}_i$ can be written as $\mathbf{x}_i = \sum_{t=1}^{k}\mathbf{a}_{it}\mathbf{z}_t$, and therefore, the data covariance of word i and word j can be expressed as:

$$\begin{aligned} cov(\mathbf{x}_i, \mathbf{x}_j) &= E[(\mathbf{x}_i - E\mathbf{x}_i)(\mathbf{x}_j - E\mathbf{x}_j)] \\ &= E[\sum_{t=1}^{k}\mathbf{a}_{it}(\mathbf{z}_t - E\mathbf{z}_t)\sum_{t=1}^{k}\mathbf{a}_{jt}(\mathbf{z}_t - E\mathbf{z}_t)] \\ &= \sum_{t=1}^{k}\sum_{t'=1}^{k}\mathbf{a}_{it}\mathbf{a}_{jt'}E[(\mathbf{z}_t - E\mathbf{z}_t)(\mathbf{z}_{t'} - E\mathbf{z}_{t'})] \\ &= \sum_{t=1}^{k}\sum_{t'=1}^{k}\mathbf{a}_{it}\mathbf{a}_{jt'}cov(\mathbf{z}_t, \mathbf{z}_{\mathbf{t}'}) \end{aligned} \tag{3.4}$$

From this expression we can see that data covariance is directly related to the covariance among latent topics. Documents from different sources have different topic distributions and thus different covariance terms $cov(\mathbf{z}_t, \mathbf{z}_{\mathbf{t}'})$ in latent topic space. As a result, the data covariance learned from

one source of documents may *not* be transferable to another class of documents. On the other hand, the semantic covariance in eq. (3.2) is completely determined by the structure of $\mathbf{A}$.

Note that the data covariance among words still depends on $\mathbf{A}$ and thus still contains certain information about the semantic relationship of words. This can also be observed from eq. (3.4). If we ignore the effect of the covariance among topics by assuming that latent topics are independently distributed and have the same variance (denoted as σ^2), eq. (3.4) can be written as:

$$cov(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad \sigma^2 \sum_{t=1}^{k} \mathbf{a}_{it}\mathbf{a}_{jt} \tag{3.5}$$

Comparing this to the last form in eq. (3.2), we see the similarity between data covariance and semantic covariance (and also between data correlation and semantic correlation). In fact, our empirical study shows that data correlation from unlabeled text does contain useful information, but is not as informative as semantic correlation.

3.3 Semantic Structure and Informative Regularization

In this section we introduce a semi-supervised learning framework to transfer knowledge from unlabeled text. Consider a set of n_l labeled documents $D_l = \{(\mathbf{x}_i^l, y_i^l) \in X \times Y_l, i = 1, \cdots n_l\}$, where $X \subseteq R^p$ is the p-dimensional word space, and $Y_l = \{-1, 1\}$ for classification and $Y_l \subseteq R$ for regression. Also assume that a large set of n_u unlabeled documents $D_u = \{\mathbf{x}_i^u \in X, i = 1, \cdots n_u\}$ is available. The goal is to learn a good function $f_l : X \to Y_l$, which is a classifier or a regressor.

Our semi-supervised learning framework contains two steps. In section 3.3.1, we propose an approach to learning the semantic structure of the word space from a set of unlabeled text. In section 3.3.2, we discuss how to effectively and efficiently incorporate this semantic structure to a specific prediction task through regularization.

3.3.1 Learning the Semantic Correlation

The semantic correlation among words can be estimated using eq. (3.3) by observing a large number of different latent topics. However, obtaining a large set of diverse but meaningful topics is hard, since the number of meaningful topics extracted by a latent topic model is usually not very large. To solve this problem, resampling techniques such as bootstrapping [Efr79] can be combined with a chosen latent variable model. The procedure is given in Algorithm 1, which uses all the available data $D = D_u \cup D_l$ and a latent variable model M as the input. The algorithm repeats N iterations. In each iteration it draws an α percentage sample[2] from the data and extracts k latent topics from the sample by applying the model M. After N iterations, the $p \times p$ semantic correlation matrix $\mathbf{\Sigma}_s$ is estimated from the observations of word distribution in kN latent topics.

[2]In this chapter, we use $\alpha = 50\%$ sampling without replacement. Other choices can be made.

Algorithm 1 Estimation of semantic correlation structure

Input: data $D = D_u \cup D_l$, latent variable model M
Output: semantic correlation matrix $\mathbf{\Sigma}_s$
Parameters: α, k, N
Initialize $\mathbf{V} \leftarrow \emptyset$
repeat
  $D_{samp} \leftarrow Sampling(D, \alpha)$
  $\{(\mathbf{z}_1, \mathbf{a}_{(,1)}), (\mathbf{z}_2, \mathbf{a}_{(,2)}), \ldots, (\mathbf{z}_k, \mathbf{a}_{(,k)})\} \leftarrow M(k, D_{samp})$
  $\mathbf{V} \leftarrow \mathbf{V} \cup \{\mathbf{a}_{(,1)}, \mathbf{a}_{(,2)}, \ldots, \mathbf{a}_{(,k)}\}$
until $|\mathbf{V}| \geq kN$
Compute $\mathbf{\Sigma}_s$: $\mathbf{\Sigma}_s(i,j) \leftarrow corr_s(\mathbf{x}_i, \mathbf{x}_j)$

The algorithm requires an appropriate latent variable model M (e.g., latent dirichlet allocation for text data), which will extract k latent topics in each iteration. The number of iterations N is set as large as necessary to obtain a reliable estimation of semantic word correlation.

3.3.2 Knowledge Transfer by Informative Regularization

This section discusses how to use the semantic structure $\mathbf{\Sigma}_s$ to improve the learning of a specific prediction task defined on the input space X. For the prediction model, we mainly consider regularized linear models with an l-2 norm penalty, e.g., support vector machines, ridge regression, logistic regression with a Gaussian prior, etc. The model is represented by a p-dimensional weight vector $\mathbf{w}$ and an intercept b. The prediction is computed as $\mathbf{w}^T\mathbf{x} + b$ for regression or by setting a threshold θ (usually $\theta = 0$) on $\mathbf{w}^T\mathbf{x} + b$ for classification. To learn $\mathbf{w}$ and b, we minimize a loss function L on the training examples plus a regularization term on $\mathbf{w}$:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{n_l} L(y_i^l, \mathbf{w}^T\mathbf{x}_i^l + b) + \lambda \mathbf{w}^T\mathbf{w} \tag{3.6}$$

Different models correspond to different loss functions [HTF01], e.g., SVMs use hinge loss, logistic regression uses log-likelihood loss, and ridge regression uses squared error loss. The regularization term $\lambda \mathbf{w}^T\mathbf{w} = \lambda \mathbf{w}^T\mathbf{I}^{-1}\mathbf{w}$ is well known to be equivalent to the Bayesian approach that imposes a Gaussian prior with zero mean and an identity covariance matrix. The covariance matrix is often set to an identity matrix due to lack of knowledge about the input space. If a covariance or correlation structure is known, e.g., the semantic structure of the word space, the regularization or the Gaussian prior can be much more informative [RNK06]. Incorporating $\mathbf{\Sigma}_s$ into the Gaussian prior leads to a new regularization term and the resulting optimization formulation is:

$$\underset{\mathbf{w},b}{\operatorname{argmin}} \sum_{i=1}^{n_l} L(y_i^l, \mathbf{w}^T\mathbf{x}_i^l + b) + \lambda \mathbf{w}^T\mathbf{\Sigma}_s^{-1}\mathbf{w} \tag{3.7}$$

Extending the discussion on SVMs in [KT07], all regularized linear models in the form of

eq. (3.7) can be easily solved by three steps. First, transform the training examples by

$$\tilde{\mathbf{x}}_i^l = \mathbf{\Sigma}_s^{\frac{1}{2}} \mathbf{x}_i^l \tag{3.8}$$

Second, learn the standard linear model in the transformed space:

$$\underset{\tilde{\mathbf{w}},b}{\operatorname{argmin}} \sum_{i=1}^{n_l} L(y_i^l, \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i^l + b) + \lambda \tilde{\mathbf{w}}^T \tilde{\mathbf{w}} \tag{3.9}$$

Finally, the optimal solution for (3.7) is obtained by:

$$\mathbf{w} = \mathbf{\Sigma}_s^{\frac{1}{2}} \tilde{\mathbf{w}} \tag{3.10}$$

This equivalence is derived from $\mathbf{w}^T \mathbf{x}_i^l = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i^l$ and $\mathbf{w}^T \mathbf{\Sigma}_s^{-1} \mathbf{w} = \tilde{\mathbf{w}}^T \tilde{\mathbf{w}}$.

Given a large collection of unlabeled text, the semantic correlation structure $\mathbf{\Sigma}_s$ can be computed offline. We consider the semantic word correlation $\mathbf{\Sigma}_s$ as an intrinsic structure of the language and thus transferable to any specific prediction task. With the pre-computed $\mathbf{\Sigma}_s$, semi-supervised learning for any prediction task simply requires the linear transformation in eq. (3.8) before training on labeled examples, which is very scalable.

3.4 Experiments

In this section, we present our empirical study on semi-supervised learning using seemingly irrelevant unlabeled text.

Data. We use the by-date version of the 20-NewsGroups data set[3], where 11314 training and 7532 testing documents are divided by date and denoted as D_{tr} and D_{ts} here. Documents are represented by bag-of-words vectors. The vocabulary is built to include the most frequent 200 words in each of the 20 newsgroups, while the 20 most frequent words over all 20 newsgroups are removed. This yields an input space X with $p = 1443$ features (words).

Prediction tasks. Documents come from 20 newsgroups, so we construct 190 binary classification tasks, one for each pair of newsgroups. For each task, a few documents in the two newsgroups are selected from D_{tr} as the labeled examples, denoted as D_l in section 3.3. The rest of the documents in D_{tr} are used as the unlabeled data, denoted by D_u. Note that D_u is a *mixture* from all the 20 newsgroups. In this sense, for any prediction task, the majority of the unlabeled data is irrelevant. As a result, semi-supervised learning algorithms that assume the unlabeled data come from the target task or the same generative distribution are unlikely to work very well. The test data for each binary classification task are all the relevant documents in D_{ts}, i.e., documents in D_{ts} that belong to the two chosen newsgroups. Note that for any task we always have $D_u \cup D_l = D_{tr}$, so Algorithm 1 is run only once on D_{tr} to learn the semantic correlation structure $\mathbf{\Sigma}_s$, which is then used for semi-supervised learning of all 190 tasks.

[3]http://people.csail.mit.edu/jrennie/20Newsgroups/

Experiment settings. There are large numbers of training documents in D_{tr} for each newsgroup. To limit the number of labeled examples for each binary classification task, in each random run we sample 5%, 10%, 20% of the relevant documents from D_{tr} as the labeled examples D_l, and the rest of the relevant documents as well as all irrelevant documents in D_{tr} are used as the unlabeled data D_u. We denote different tests as 5%-Test, 10%-Test, and 20%-Test, indicating the number of labeled examples we sample from the entire training set D_{tr}. The result of each test is averaged over 10 random runs, with D_l randomly selected from D_{tr}. The testing data for each task are fixed to be all relevant documents in D_{ts}, which is invariant among different tests and random runs.

Methods for comparison are as follows.

- SVM/LGR/RR: SVM, L-2 regularized logistic regression or ridge regression directly trained on labeled examples D_l. For ridge regression, examples are labeled as $+1$ and -1, and the classification decision is made by $\mathbf{w}^T\mathbf{x} + b > 0$.
- SVM_{LDA}/LGR_{LDA}/RR_{LDA}: SVM, L-2 regularized logistic regression or ridge regression trained on D_l in the latent topic space extracted by latent dirichlet allocation on $D_l \cup D_u$ [BNJ03].
- SVM_{PCA}/LGR_{PCA}/RR_{PCA}: SVM, L-2 regularized logistic regression or ridge regression trained on D_l in principal component space extracted by PCA on $D_l \cup D_u$.
- SVM_{IR}/LGR_{IR}/RR_{IR}: SVM, L-2 regularized logistic regression or ridge regression trained on D_l using our proposed informative regularization, where the penalty includes the *semantic* correlation structure $\mathbf{\Sigma}_s$ inferred from $D_l \cup D_u$ using Algorithm 1.
- $SVM_{IR(data)}$/$LGR_{IR(data)}$/$RR_{IR(data)}$: SVM, L-2 regularized logistic regression or ridge regression trained on D_l using our proposed informative regularization, where the penalty includes the *data* correlation structure $\mathbf{\Sigma}$ directly estimated from the bag-of-words representation of documents in $D_l \cup D_u$.

Recently a fast semi-supervised SVM using L-2 loss was proposed [SK06], which makes it possible to handle large-scale unlabeled data. We also test the following methods:

- $L2$-SVM: L-2 loss SVM directly trained on labeled examples D_l.
- $L2$-S^3VM: L-2 loss semi-supervised SVM [SK06] trained on $D_l \cup D_u$.
- $L2$-SVM_{IR}: L-2 loss SVM trained on D_l using our informative regularization, where the penalty includes the *semantic* correlation $\mathbf{\Sigma}_s$ inferred from $D_l \cup D_u$ using Algorithm 1.
- $L2$-S^3VM_{oracle}: the semi-supervised SVM ($L2$-S^3VM) may not work very well since the unlabeled data is a mixture from all tasks. Therefore, we also test an "oracle" version of the semi-supervised SVM, where unlabeled examples for each task come from only the two relevant newsgroups. In other words, we handpick unlabeled data for each prediction task.

Here are additional implementation details. The regularization parameter λ for each model is determined by 5-fold cross-validation in the range 10^{-6} to 10^6. LibSVM 2.85 is used for SVM. For PCA, we tried $10, 20, 30, 50, 100, 200, 400$ principal components and report PCA using 200 principal components as the best result. For latent dirichlet allocation, we use the implementation at http://chasen.org/∼daiti-m/dist/lda/. We tried $k = 10, 20, 30, 50, 100, 200$ latent topics with 30

Table 3.1: Classification errors over 190 tasks, based on SVMs

	5%-Test	10%-Test	20%-Test
SVM	14.22%	10.34%	7.88%
$SVM_{LDA(30)}$	9.76% (179/11)	8.01% (171/19)	6.90% (161/29)
$SVM_{PCA(200)}$	13.32% (123/67)	10.31% (104/86)	8.29% (89/101)
SVM_{IR}	**7.58% (190/0)**	**6.11% (190/0)**	**5.13% (183/7)**
$SVM_{IR(data)}$	9.40% (185/5)	7.14% (183/7)	5.70% (180/10)

Table 3.2: Classification errors over 190 tasks, based on regularized logistic regression

	5%-Test	10%-Test	20%-Test
LGR	11.70%	8.43%	6.67%
$LGR_{LDA(30)}$	8.21% (171/19)	7.38% (156/34)	6.79% (134/56)
$LGR_{PCA(200)}$	11.43% (105/85)	8.95% (65/125)	7.28% (64/122)
LGR_{IR}	**6.70% (189/1)**	**5.78% (181/9)**	**5.19% (169/21)**
$LGR_{IR(data)}$	8.46% (172/18)	7.21% (157/33)	6.46% (132/58)

Table 3.3: Classification errors over 190 tasks, based on ridge regression

	5%-Test	10%-Test	20%-Test
RR	14.13%	10.73%	8.90%
$RR_{LDA(30)}$	14.08% (111/101)	11.98% (67/102)	11.34% (42/148)
$RR_{PCA(200)}$	15.50% (56/132)	12.80% (33/157)	11.53% (17/173)
RR_{IR}	**10.55% (182/8)**	**8.88% (161/29)**	**8.01% (134/56)**
$RR_{IR(data)}$	**10.68% (176/14)**	**8.94% (157/33)**	**7.99% (139/51)**

topics performing best. For the proposed method, Algorithm 1 uses latent dirichlet allocation with $k = 30$ topics per sampling, repeats $N = 100$ iterations, and $\mathbf{\Sigma}_s$ is estimated from these 3000 latent topics. $L2\text{-}S^3VM$ (code available as SVMlin [SK06]) has a second parameter λ_u for unlabeled examples, which is set to 1 as in [SK06]. Unlabeled data for $L2\text{-}S^3VM$ is downsampled to 3000 documents for each run to make training (and cross-validation) feasible.

Evaluation measures. For each semi-supervised learning algorithm, we report two performance measures: 1) the average classification error over all 190 tasks, and 2) the number of winning/losing tasks when compared to the corresponding supervised learning method (SVM, LGR, RR, or $L2\text{-}SVM$). Note that the first measure indicates the overall effectiveness of a semi-supervised learning method, while the second measure shows the reliability of the knowledge transfer from unlabeled data (i.e., how often does a semi-supervised learning method help?).

Empirical results are shown in Tables 3.1- 3.4. From Tables 3.1 - 3.3, IR based methods with semantic correlation significantly outperform standard supervised learning, LDA based methods, PCA based methods, and is also generally more effective than IR with data correlation. The LDA based algorithms slightly improve the prediction performance when using SVM or logistic regres-

Table 3.4: Classification errors compared to semi-supervised SVMs over 190 tasks, based on L2-SVM

	5%-Test	10%-Test	20%-Test
$L2\text{-}SVM$	11.18%	8.41%	6.65%
$L2\text{-}S^3VM$	14.14% (14/176)	11.64% (5/185)	10.04% (1/189)
$L2\text{-}S^3VM_{oracle}$	8.22% (189/1)	6.95% (185/5)	6.00% (164/24)
$L2\text{-}SVM_{IR}$	**6.87%** (**188/2**)	**5.73%** (**180/10**)	**4.98%** (**177/13**)

Table 3.5: Top 10 word pairs in terms of the ratio: semantic correlation / data correlation

gaza, lebanes	biker, yamaha	motorcycl, yamaha	batter, clemen	yanke, catcher
0.956/0.007	0.937/−0.004	0.970/0.030	0.932/−0.002	0.934/0.002
palestin, lebanes	cage, ama	toyota, mileag	mileag, mustang	brave, batter
0.946/0.181	0.921/−0.005	0.934/0.009	0.923/−0.002	0.950/0.025

sion as the base classifier, while decreasing the performance when using ridge regression. This is possibly because the loss function of ridge regression is not a good approximation to the 0/1 classification error, and therefore, ridge regression is more sensitive to irrelevant latent features extracted from mixed unlabeled documents. The PCA based methods are generally worse than standard supervised learning, which indicates they are sensitive to the mixed unlabeled data. In Table 3.4, the $L2\text{-}S^3VM$ performs worse than standard $L2\text{-}SVM$, showing that traditional semi-supervised learning cannot handle unlabeled data outside the target task. We can also see that the $L2\text{-}SVM_{IR}$ even outperforms the oracle version of semi-supervised SVM ($L2\text{-}S^3VM_{oracle}$) by achieving similar winning/losing ratio but better average classification error. This is a pleasantly surprising result: the improvement achieved by using the proposed method on a mixture of seemingly irrelevant unlabeled text even exceeds the improvement obtained by running a state-of-the-art semi-supervised learning method on handpicked unlabeled data. In conclusion, the empirical results show that the proposed approach is an effective and reliable method for semi-supervised learning, regardless of the source of unlabeled data.

It is also interesting to directly compare the semantic correlation matrix $\mathbf{\Sigma}_s$ and the data correlation matrix $\mathbf{\Sigma}$ learned from unlabeled data. We make three observations: 1) The average value of entries is 0.0147 in the semantic correlation matrix $\mathbf{\Sigma}_s$ and 0.0341 in the data correlation matrix $\mathbf{\Sigma}$. We have 1617834 entries with higher data correlation and 462972 entries with higher semantic correlation. Thus overall speaking, words tend to have higher values in the data correlation. 2) Interestingly, if we list the top 1000 pairs of words with the largest (absolute) difference between semantic correlation and data correlation, they *all* have higher semantic correlation and lower data correlation. In other words, there are many word pairs with significant semantic correlation but no significant data correlation. 3) We list the top 10 such word pairs and their semantic and data correlation in Table 3.5. Each of these word pairs is indeed semantically related, and therefore semantic correlation helps to discover meaningful word pairs that are ignored by data correlation.

3.5 Related Work

In this work, we explore the possibility of extracting *transferable* knowledge about the input space from seemingly irrelevant unlabeled text. This knowledge is represented as the semantic correlation structure of words. The covariance or correlation structure of the input space has been studied in transfer learning [RNK06, KT07] settings. A covariance structure can be transferred from a few related tasks to a target task [RNK06] or inferred from meta-features [KT07]. In fact, one way to view our proposed semi-supervised learning method is: 1) we automatically construct a large number of diverse but meaningful "tasks" from unlabeled text without using external knowledge, where each "task" corresponds to the composition of a latent topic; 2) we learn the semantic correlation structure of the word space from these "tasks" and show that the learned structure is generally transferable regardless of the source of unlabeled data; 3) this semantic correlation structure can be efficiently incorporated into the learning of any prediction task via regularization, which leads to an effective, reliable and scalable way for semi-supervised learning.

3.6 Conclusion and Discussion

In this chapter, we encode the covariance and correlation structure of the input space into learning and propose a semi-supervised learning method for text classification. We first ask the question: what kind of knowledge is *generally transferable* from unlabeled text? We suggest and analyze the semantic correlation of words as a generally transferable structure of the language and propose a new method to learn this structure from unlabeled text using latent topic models. This semantic correlation contains structural information of the language and can be used to control the joint shrinkage of model parameters for any text classification task in the same language space. In our empirical study, we construct 190 different text classification tasks from a real-world benchmark, and the unlabeled documents are a mixture from all these tasks. We test the ability of various algorithms to use the mixed unlabeled text to enhance all classification tasks. Empirical results show that the proposed approach is a reliable and scalable method for semi-supervised learning, regardless of fact that for any specific prediction task, the majority of the unlabeled data is irrelevant.

In semi-supervised learning, unlabeled data can contribute to learning either $P(X)$ (usually in a generative model) or $P(Y|X)$ (usually in a discriminative model). In Section 3.4, we empirically compare the semantic correlation matrix $\mathbf{\Sigma}_s$ and the data correlation matrix $\mathbf{\Sigma}$ learned from unlabeled data, where using semantic correlation matrix $\mathbf{\Sigma}_s$ in regularization leads to more effective learning. One way to think about this observation is that the data correlation of words $\mathbf{\Sigma}$ is directly estimated from data and thus mainly describes the properties of $P(X)$. On the other hand, the semantic correlation of words $\mathbf{\Sigma}_s$ is estimated using latent topic models and represents the correlation of words in contributing to composition of latent topics, and more generally, in forming other discriminative functions. Therefore, $\mathbf{\Sigma}_s$ can convey information about $P(Y|X)$ and is more effective when used in regularization of model coefficients.

Although in this chapter we focus on inferring the semantic correlation of "words" and use it to improve text learning on bag-of-words representations, the proposed method can be extended

in a variety of ways. First, it is straightforward to apply the proposed method to text learning problems using n-gram representations [CT94], and in this case, the dimensionality of the input space will dramatically increase and therefore estimation of the semantic correlation matrix is more challenging. More interestingly, the proposed method can potentially be applied to image learning problems, and the key issue is to define the bag-of-words representation of images. Luckily, the bag-of-visual-word representation [YJHN07] has been intensively studied and considered very effective for recognition and classification tasks on images. In this sense, the proposed method in this chapter can also be extended to learning problems on images.

Chapter 4

A Sparse Matrix-Normal Penalty for Learning Multiple Tasks

4.1 Overview

In this chapter [ZS10b], we consider encoding the covariance structure of the *joint* input space when learning multiple related tasks. In multi-task learning, the joint input space can be viewed as a matrix, where rows correspond to tasks and columns correspond to feature dimensions[1]. As a result, learning multiple (parametric) models can be viewed as estimating a parameter matrix.

Matrix-variate normal distributions are powerful tools to characterize the structure of a matrix. Following the matrix normal density, we design a matrix-normal regularization penalty that decomposes the full covariance of matrix entries into the Kronecker product of row covariance and column covariance, which characterize task relations and feature representations, respectively. To prevent overfitting and automatically select meaningful task and feature structures, we perform sparse covariance selection via ℓ_1 penalties on the inverse of task and feature covariances.

Recent research on multi-task learning has been focused on two major directions: learning a common feature representation across tasks, or directly inferring the relatedness of tasks. If we view learning multiple tasks as estimating a matrix of model parameters, learning the feature structure corresponds to discovering the structure of columns, and modeling the task relations corresponds to finding the relations among rows. As a result, several recently proposed methods for multi-task learning can be viewed as variants of special cases of our formulation.

In our empirical study, we compare the proposed method to several related multi-task learning methods in two real-world problems: detecting landmines in multiple fields and recognizing faces between different subjects. Experimental results show that the proposed method provides an effective and flexible way to model various different structures of multiple tasks.

[1] As most work in multi-task learning, we assume that tasks share the same input feature space.

4.2 A Sparse Matrix-Normal Penalty for Multi-Task Learning

In this section, we propose a matrix-normal penalty with sparse inverse covariances for learning multiple related tasks. In Section 4.2.1 we start with learning multiple tasks with a matrix-normal penalty. In Section 4.2.2 we study how to incorporate sparse covariance selection into our framework by further imposing ℓ_1 penalties on task and feature inverse covariances. In Section 4.2.3 we outline the algorithm, and in Section 4.2.4 we discuss other useful constraints in our framework.

4.2.1 Learning with a Matrix Normal Penalty

Consider a multi-task learning problem with m tasks in a p-dimensional feature space. The training sets are $\{\mathbf{D}_t\}_{t=1}^m$, where each set $\mathbf{D}_t$ contains n_t examples $\{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{n_t}$. We want to learn m models for the m tasks but appropriately share knowledge among tasks. Model parameters are represented by an $m \times p$ matrix $\mathbf{W}$, where parameters for a task correspond to a row.

In Section 2.6 we have presented a review on matrix-variate normal distribution. The last term in the matrix-variate normal density (2.7) provides a structure to couple the parameters of multiple tasks as a matrix $\mathbf{W}$: 1) we set $\mathbf{M} = \mathbf{0}$, indicating a preference for simple models; 2) the $m \times m$ row covariance $\mathbf{\Omega}$ describes the similarity among tasks; 3) the $p \times p$ column covariance matrix $\mathbf{\Sigma}$ represents a shared feature structure. This yields the following total loss $\mathcal{L}$:

$$\mathcal{L} = \sum_{t=1}^{m} \sum_{i=1}^{n_t} L(y_i^{(t)}, \mathbf{x}_i^{(t)}, \mathbf{W}(t,:)) + \lambda \, tr\{\mathbf{\Omega}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^T\} \tag{4.1}$$

where λ controls the strength of the regularization, $(y_i^{(t)}, \mathbf{x}_i^{(t)})$ is the ith example in the training set of the tth task, $\mathbf{W}(t,:)$ is the parameter vector of the tth task, and $L()$ is a convex empirical loss function depending on the specific model we use, e.g., squared loss for linear regression, log-likelihood loss for logistic regression, hinge loss for SVMs, and so forth. When $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ are known and positive definite, eq. (4.1) is convex w.r.t. $\mathbf{W}$ and can be optimized efficiently [NW00].

Now we discuss a few special cases of (4.1) and how is previous work related to them.

- When we fix $\mathbf{\Omega} = \mathbf{I}_m$ and $\mathbf{\Sigma} = \mathbf{I}_p$, the penalty term can be decomposed into standard ℓ_2-norm penalties on the m rows of $\mathbf{W}$. In this case, the m tasks in (4.1) can be learned almost independently using single-task ℓ_2 regularization (but they are still tied by the shared λ).
- When we fix $\mathbf{\Omega} = \mathbf{I}_m$, tasks are linked only by a shared feature covariance $\mathbf{\Sigma}$. This corresponds to multi-task feature learning [AEP06, AMPY07] which optimizes eq. (4.1) w.r.t. $\mathbf{W}$ and $\mathbf{\Sigma}$, with an additional constraint $tr\{\mathbf{\Sigma}\} \leq 1$ on the trace of $\mathbf{\Sigma}$ to avoid setting $\mathbf{\Sigma}$ to infinity.
- When we fix $\mathbf{\Sigma} = \mathbf{I}_p$, tasks are coupled only by a task similarity matrix $\mathbf{\Omega}$. This is used in a recent clustered multi-task learning formulation [JBV08], which optimizes eq. (4.1) w.r.t. $\mathbf{W}$ and $\mathbf{\Omega}$, with additional constraints on the singular values of $\mathbf{\Omega}$ that are motivated and derived from task clustering. A more recent multi-label classification model [HVV10] essentially optimizes $\mathbf{W}$ in eq. (4.1) with a label correlation $\mathbf{\Omega}$ given as prior knowledge and empirical loss L as the max-margin hinge loss.

We usually do not know task and feature structures in advance. Therefore, we would like to infer $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ in eq. (4.1). Note that if we jointly optimize $\mathbf{W}$, $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ in eq. (4.1), we will always set $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ to be infinity matrices. We can impose constraints on $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ to avoid this, but a more natural way is to further expand eq. (4.1) to include all relevant terms w.r.t. $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ from the matrix normal log-density (2.7). As a result, the total loss $\mathcal{L}$ is:

$$\mathcal{L} = \sum_{t=1}^{m} \sum_{i=1}^{n_t} L(y_i^{(t)}, \mathbf{x}_i^{(t)}, \mathbf{W}(t,:)) + \lambda \left[p \log |\mathbf{\Omega}| + m \log |\mathbf{\Sigma}| + tr\{\mathbf{\Omega}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^T\} \right] \tag{4.2}$$

Based on this formula, we can infer task structure $\mathbf{\Omega}$ and feature structure $\mathbf{\Sigma}$ given the model parameters $\mathbf{W}$, as the following problem:

$$\min_{\mathbf{\Omega},\mathbf{\Sigma}} \; p \log |\mathbf{\Omega}| + m \log |\mathbf{\Sigma}| + tr\{\mathbf{\Omega}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^T\} \tag{4.3}$$

This problem is equivalent to maximizing the log-likelihood of a matrix normal distribution as in eq. (2.7), given $\mathbf{W}$ as observations and expectation $\mathbf{M}$ fixed at $\mathbf{0}$. Following Section 2.6, the MLE of $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ can be obtained by the "flip-flop" algorithm:

$$\begin{cases} \hat{\mathbf{\Omega}} &= \frac{1}{p}\mathbf{W}\hat{\mathbf{\Sigma}}^{-1}\mathbf{W}^T + \epsilon \mathbf{I}_m \\ \hat{\mathbf{\Sigma}} &= \frac{1}{m}\mathbf{W}^T\hat{\mathbf{\Omega}}^{-1}\mathbf{W} + \epsilon \mathbf{I}_p \end{cases} \tag{4.4}$$

where ϵ is a small positive constant to improve numerical stability. As discussed in Section 2.6, only the optimal $\mathbf{\Sigma} \otimes \mathbf{\Omega}$ is uniquely defined, and $\hat{\mathbf{\Omega}}$ and $\hat{\mathbf{\Sigma}}$ are only identifiable up to an multiplicative constant. However, this will not affect the optimization of $\mathbf{W}$ using eq. (4.1), since only $\mathbf{\Sigma} \otimes \mathbf{\Omega}$ matters for finding the optimal $\mathbf{W}$ in eq. (4.1).

4.2.2 Sparse Covariance Selection in the Matrix-Normal Penalty

Consider the sparsity of $\mathbf{\Omega}^{-1}$ and $\mathbf{\Sigma}^{-1}$. When $\mathbf{\Omega}$ has a sparse inverse, task pairs corresponding to zero entries in $\mathbf{\Omega}^{-1}$ will not be explicitly coupled in the penalty of (4.2). Similarly, a zero entry in $\mathbf{\Sigma}^{-1}$ indicates no direct interaction between the two corresponding feature dimensions in the penalty (4.2). Also note that a clustering of tasks can be expressed by block-wise sparsity of $\mathbf{\Omega}^{-1}$.

Covariance selection aims to select nonzero entries in the Gaussian inverse covariance and discover conditional independence between variables (indicated by zero entries in the inverse covariance) [Dem72, BGd08, FHT07, DGK08]. The matrix-normal density in eq. (4.2) enables us to perform sparse covariance selection to regularize and select task and feature structures.

Formally, we rewrite (4.2) to include two additional ℓ_1 penalty terms on inverse covariances:

$$\begin{aligned} \mathcal{L} = \sum_{t=1}^{m} \sum_{i=1}^{n_t} L(y_i^{(t)}, \mathbf{x}_i^{(t)}, \mathbf{W}(t,:)) + \lambda [p \log |\mathbf{\Omega}| + m \log |\mathbf{\Sigma}| + tr\{\mathbf{\Omega}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^T\}] \\ + \lambda_\Omega ||\mathbf{\Omega}^{-1}||_{\ell_1} + \lambda_\Sigma ||\mathbf{\Sigma}^{-1}||_{\ell_1} \end{aligned} \tag{4.5}$$

where $||\ ||_{\ell_1}$ is the ℓ_1-norm of a matrix, and λ_Ω and λ_Σ control the strength of ℓ_1 penalties and therefore the sparsity of task and feature structures.

Based on the new regularization formula (4.5), estimating $\mathbf{W}$ given $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ as in (4.1) is not affected, while inferring $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ given $\mathbf{W}$, previously shown as (4.3), becomes a new problem:

$$\min_{\mathbf{\Omega},\mathbf{\Sigma}} \; p \log |\mathbf{\Omega}| + m \log |\mathbf{\Sigma}| + tr\{\mathbf{\Omega}^{-1}\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^T\} + \frac{\lambda_\Omega}{\lambda}||\mathbf{\Omega}^{-1}||_{\ell_1} + \frac{\lambda_\Sigma}{\lambda}||\mathbf{\Sigma}^{-1}||_{\ell_1} \tag{4.6}$$

As in (4.4), we can iteratively optimize $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ until convergence, as follows:

$$\begin{cases} \hat{\mathbf{\Omega}} &= \text{argmin}_{\mathbf{\Omega}} \; p \log |\mathbf{\Omega}| + tr\{\mathbf{\Omega}^{-1}(\mathbf{W}\mathbf{\Sigma}^{-1}\mathbf{W}^T)\} + \frac{\lambda_\Omega}{\lambda}||\mathbf{\Omega}^{-1}||_{\ell_1} \\ \hat{\mathbf{\Sigma}} &= \text{argmin}_{\mathbf{\Sigma}} \; m \log |\mathbf{\Sigma}| + tr\{\mathbf{\Sigma}^{-1}(\mathbf{W}^T\hat{\mathbf{\Omega}}^{-1}\mathbf{W})\} + \frac{\lambda_\Sigma}{\lambda}||\mathbf{\Sigma}^{-1}||_{\ell_1} \end{cases} \tag{4.7}$$

Note that both equations in (4.7) are ℓ_1 regularized covariance selection problems, for which efficient optimization has been intensively studied [BGd08, FHT07, DGK08]. For example, we can use graphical lasso [FHT07] as a basic solver and consider (4.7) as an ℓ_1 regularized "flip-flop" algorithm:

$$\begin{cases} \hat{\mathbf{\Omega}} &= glasso(\frac{1}{p}\mathbf{W}\hat{\mathbf{\Sigma}}^{-1}\mathbf{W}^T, \frac{\lambda_\Omega}{\lambda}) \\ \hat{\mathbf{\Sigma}} &= glasso(\frac{1}{m}\mathbf{W}^T\hat{\mathbf{\Omega}}^{-1}\mathbf{W}, \frac{\lambda_\Sigma}{\lambda}) \end{cases}$$

Finally, an annoying part of eq. (4.5) is the presence of two additional regularization parameters λ_Ω and λ_Σ. Due to the property of matrix normal distributions that only $\mathbf{\Sigma} \otimes \mathbf{\Omega}$ is identifiable, we can reduce the complexity of choosing regularization parameters by considering the restriction:

$$\lambda_\Omega = \lambda_\Sigma \tag{4.8}$$

The following lemma proves that restricting λ_Ω and λ_Σ to be equal in eq. (4.5) will not reduce the space of optimal models $\mathbf{W}$ we can obtain. As a result, we eliminate one regularization parameter.

Lemma 1. Suppose $\mathbf{W}^*$ belongs to a minimizer $(\mathbf{W}^*, \mathbf{\Omega}^*, \mathbf{\Sigma}^*)$ for eq. (4.5) with some arbitrary choice of λ, λ_Ω and $\lambda_\Sigma > 0$. Then, $\mathbf{W}^*$ must also belong to a minimizer for eq. (4.5) with certain choice of λ', λ'_Ω and λ'_Σ such that $\lambda'_\Omega = \lambda'_\Sigma$. Proof of lemma 1 is provided in Appendix A.

4.2.3 The Algorithm

Based on the regularization formula (4.5), we study the following algorithm for multi-task learning:

1) Estimate $\mathbf{W}$ by solving (4.1), using $\mathbf{\Omega} = \mathbf{I}_m$ and $\mathbf{\Sigma} = \mathbf{I}_p$;

2) Given the estimated $\mathbf{W}$ from 1), infer $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ in (4.5) by solving (4.7) until convergence;

3) Estimate $\mathbf{W}$ by solving (4.1), using the inferred $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ from step 2).

One can iterate over steps 2) and 3) and converge to a local minimum of eq. (4.5). However, we observed that a single pass yields good results: further iterations over step 2) and 3) will not dramatically change model estimation, and also, early stopping as regularization might lead to better generalizability. Note that steps 1) and 3) are linear in the number of data points and step 2) is independent of it, so the method scales well with the number of training samples. Step 2) needs to solve ℓ_1 regularized covariance selection problems as (4.7). We use the state of the art technique [FHT07], but more efficient optimization for large covariances is still desirable.

4.2.4 Additional Constraints

We can have additional structure assumptions in the matrix-normal penalty. For example, consider:

$$\Omega_{ii} = 1 \quad i = 1, 2, \ldots, m \tag{4.9}$$
$$\Sigma_{jj} = 1 \quad j = 1, 2, \ldots, p \tag{4.10}$$

In this case, we ignore variances and restrict our attention to *correlation* structures. For example, off-diagonal entries of task covariance $\mathbf{\Omega}$ characterize the task similarity; diagonal entries indicate different amounts of regularization on tasks, which may be fixed as a constant if we prefer tasks to be equally regularized. Similar arguments apply to feature covariance $\mathbf{\Sigma}$. We include these restrictions by converting inferred covariance(s) into correlation(s) in step 2) of the algorithm in Section 4.2.3. In other words, the restrictions are enforced by a projection step.

If one wants to iterative over steps 2) and 3) of the algorithm in Section 4.2.3 until convergence, we may consider the constraints

$$\Omega_{ii} = c_1 \quad i = 1, 2, \ldots, m \tag{4.11}$$
$$\Sigma_{jj} = c_2 \quad j = 1, 2, \ldots, p \tag{4.12}$$

with unknown quantities c_1 and c_2, and consider eq. (4.5) in step 2) as a *constrained optimization* problem w.r.t. $\mathbf{W}$, $\mathbf{\Omega}$, $\mathbf{\Sigma}$, c_1 and c_2, instead of using a projection step. As a result, the "flip-flop" algorithm in (4.7) needs to solve ℓ_1 penalized covariance selection with equality constraints (4.11) or (4.12), where the dual block coordinate descent [BGd08] and graphical lasso [FHT07] are no longer directly applicable. In this case, one can solve the two steps of (4.7) as determinant maximization problems with linear constraints [VBW96], but this is inefficient.

4.3 Empirical Studies

In this section, we present our empirical studies on a landmine detection problem and a face recognition problem, where multiple tasks correspond to detecting landmines at different landmine fields and classifying faces between different subjects, respectively.

4.3.1 Data Sets and Experimental Settings

The landmine detection data set [XLCK07] contains examples collected from different landmine fields. Each example is represented by a 9-dimensional feature vector extracted from radar imaging, which includes moment-based features, correlation-based features, an energy ratio feature and a spatial variance feature. As a binary classification problem, the goal is to predict landmines (positive class) or clutter (negative class). Following [XLCK07], we jointly learn 19 tasks from landmine fields $1-10$ and $19-24$ in the data set. As a result, model parameters can be represented by a 19×10 matrix $\mathbf{W}$, corresponding to 19 tasks and 10 dimensions (including the intercept).

The distribution of examples is imbalanced in each task, with a few dozen positive examples and several hundred negative examples. Therefore, we use the average AUC (Area Under the ROC Curve) over 19 tasks as the performance measure. We vary the size of the training set for each task as $30, 40, 80$ and 160. Note that we intentionally keep the training sets small because the need for cross-task learning diminishes as the training set becomes large relative to the number of parameters being learned. For each training size, we randomly select training examples for each task and the rest is used as the testing set. This is repeated 30 times. Task-average AUC scores are collected over 30 runs, and mean and standard errors are reported. Note that for small training sizes (e.g., 30 per task) we often have some task(s) that do *not* have any positive training sample. It is interesting to see how well multi-task learning handles this case.

The Yale face recognition data set contains 165 images of 15 subjects. The 11 images per subject correspond to different configurations in terms of expression, emotion, illumination, and wearing glasses (or not), etc. Each image is scaled to 32×32 pixels. We use the first 8 subjects to construct $\frac{8\times 7}{2} = 28$ binary classification tasks, each to classify two subjects. We vary the size of the training set as $3, 5$ and 7 images per subject. We have 30 random runs for each training size. In each run, we randomly select the training set and use the rest as the testing set. We collect task-average classification errors over 30 runs, and report mean and standard errors.

Choice of features is important for face recognition problems. In our experiments, we use orthogonal Laplacianfaces [CHHZ06], which have been shown to provide better discriminative power than Eigenfaces (PCA), fisherfaces (LDA) and Laplacianfaces on several benchmarks. In each random run, we extract 30 orthogonal Laplacianfaces using the selected training set of all 8 subjects[2], and conduct experiments of all 28 classification tasks in the extracted feature space.

4.3.2 Models and Implementation Details

We use the logistic regression loss as the empirical loss L in (4.5) and compare following methods.

- **STL**: learn ℓ_2 regularized logistic regression for each task separately.
- **MTL-C**: clustered multi-task learning [JBV08], which encourages task clustering in regularization. As discussed in Section 4.2.1, this formulation is related to the special case of eq. (4.1) with only a task structure $\mathbf{\Omega}$.
- **MTL-F**: multi-task feature learning [AEP06], which corresponds to fixing the task covariance $\mathbf{\Omega}$ as $\mathbf{I}_m$ and optimizing (4.2) with only the feature covariance $\mathbf{\Sigma}$.
- **MTL($\mathbf{I}_m$&$\mathbf{I}_p$)**: learn $\mathbf{W}$ using (4.5) with $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ fixed as identity matrices $\mathbf{I}_m$ and $\mathbf{I}_p$.
- **MTL($\mathbf{\Omega}$&$\mathbf{I}_p$)**: learn $\mathbf{W}$ and task covariance $\mathbf{\Omega}$ using (4.5), with feature covariance $\mathbf{\Sigma}$ as $\mathbf{I}_p$.
- **MTL($\mathbf{I}_m$&$\mathbf{\Sigma}$)**: learn $\mathbf{W}$ and feature covariance $\mathbf{\Sigma}$ using (4.5), with task covariance $\mathbf{\Omega}$ as $\mathbf{I}_m$.
- **MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)**: learn $\mathbf{W}$, $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ using (4.5), inferring both task and feature structures.
- **MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$**: learn $\mathbf{W}$, $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ using (4.5), with additional restrictions on $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ as (4.9) and (4.10).

[2]For experiments with 3 images per subject, we can only extract 23 Laplacianfaces, which is limited by the size of training examples ($3 \times 8 = 24$) [CHHZ06].

Avg AUC Score	30 samples	40 samples	80 samples	160 samples
STL	64.85(0.52)	67.62(0.64)	71.86(0.38)	76.22(0.25)
MTL-C [JBV08]	67.09(0.44)	68.95(0.40)	72.89(0.31)	76.64(0.17)
MTL-F [AEP06]	72.39(0.79)	74.75(0.63)	77.12(0.18)	**78.13(0.12)**
MTL($\mathbf{I}_m$&$\mathbf{I}_p$)	66.10(0.65)	69.91(0.40)	73.34(0.28)	76.17(0.22)
MTL($\mathbf{\Omega}$&$\mathbf{I}_p$)	**74.88(0.29)**	75.83(0.28)	76.93(0.15)	77.95(0.17)
MTL($\mathbf{I}_m$&$\mathbf{\Sigma}$)	72.71(0.65)	74.98(0.32)	**77.35(0.14)**	**78.13(0.14)**
MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)	**75.10(0.27)**	76.16(0.15)	**77.32(0.24)**	**78.21(0.17)***
MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$	**75.31(0.26)***	**76.64(0.13)***	**77.56(0.16)***	**78.01(0.12)**
MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$	**75.19(0.22)**	76.25(0.14)	**77.22(0.15)**	**78.03(0.15)**

Table 4.1: Average AUC scores (%) on landmine detection: means (and standard errors) over 30 random runs. For each column, the best model is marked with * and competitive models (by paired t-tests) are shown in **bold**.

- **MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$**: learn $\mathbf{W}$, $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ using (4.5), with restricted $\mathbf{\Omega}$ as (4.9) but no restriction on $\mathbf{\Sigma}$. Intuitively, free diagonal entries in $\mathbf{\Sigma}$ are useful when features are of different importance, e.g, a sequence of components extracted by orthogonal Laplacianfaces usually capture decreasing amounts of discriminative information [CHHZ06].

We use the conjugate gradient method [NW00] to optimize $\mathbf{W}$ in (4.1), and infer $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ in (4.7) using graphical lasso [FHT07] as the basic solver. Regularization parameters λ and $\lambda_\Omega = \lambda_\Sigma$ are chosen by 3-fold cross validation within the range $[10^{-7}, 10^3]$. The model in [JBV08] has four regularization parameters, and we consider three values for each parameter, leading to $3^4 = 64$ combinations chosen by cross validation.

4.3.3 Results on Landmine Detection

The results on landmine detection are shown in Table 4.1. Each row of the table corresponds to a model in our experiments. Each column is a training sample size. We have 30 random runs for each sample size. We use task-average AUC score as the performance measure and report the mean and standard error of this measure over 30 random runs. The best model is marked with *, and models displayed in bold fonts are statistically competitive models (i.e. not significantly inferior to the best model in a one-sided paired t-test with $\alpha = 0.05$).

Overall speaking, MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$) and MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$ lead to the best prediction performance. For small training sizes, restricted $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ ($\Omega_{ii} = \Sigma_{jj} = 1$) offer better prediction; for large training size (160 per task), free $\mathbf{\Omega}$ and $\mathbf{\Sigma}$ give the best performance. The best model performs better than MTL-F [AEP06] and much better than MTL-C [JBV08] with small training sets.

MTL($\mathbf{I}_m$&$\mathbf{I}_p$) performs better than STL, i.e., even the simplest coupling among tasks (by sharing λ) can be helpful when the size of training data is small. Consider the performance of MTL($\mathbf{\Omega}$&$\mathbf{I}_p$) and MTL($\mathbf{I}_m$&$\mathbf{\Sigma}$), which learn either a task structure or a feature structure. When the size of training samples is small (i.e., 30 or 40), coupling by task similarity is more effective, and as

Avg Classification Errors	3 samples per class	5 samples per class	7 samples per class
STL	10.97(0.46)	7.62(0.30)	4.75(0.35)
MTL-C [JBV08]	11.09(0.49)	7.87(0.34)	5.33(0.34)
MTL-F [AEP06]	10.78(0.60)	6.86(0.27)	4.20(0.31)
MTL($\mathbf{I}_m$&$\mathbf{I}_p$)	10.88(0.48)	7.51(0.28)	5.00(0.35)
MTL($\mathbf{\Omega}$&$\mathbf{I}_p$)	9.98(0.55)	6.68(0.30)	4.12(0.38)
MTL($\mathbf{I}_m$&$\mathbf{\Sigma}$)	**9.87(0.59)**	**6.25(0.27)**	4.06(0.34)
MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)	**9.81(0.49)**	**6.23(0.29)**	4.11(0.36)
MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$	**9.67(0.57)***	**6.21(0.28)**	4.02(0.32)
MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$	**9.67(0.51)***	**5.98(0.29)***	**3.53(0.34)***

Table 4.2: Average classification errors (%) on face recognition: means (and standard errors) over 30 random runs. For each column, the best model is marked with * and competitive models (by paired t-tests) are shown in **bold**.

the training size increases, learning a common feature representation is more helpful. Finally, consider MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$), MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$ and MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$. MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$ imposes a strong restriction and leads to better performance when the training size is small. MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$) is more flexible and performs well given large numbers of training samples. MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$ performs similarly to MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$, indicating no significant variation of feature importance and thus no evident effect of the restriction $\Sigma_{jj} = 1$ on the feature covariance.

4.3.4 Results on Face Recognition

Empirical results on face recognition are shown in Table 4.2, with the best model in each column marked with * and competitive models displayed in bold. MTL-C [JBV08] performs even worse than STL. One possible explanation is that, each prediction task is to classify faces between a pair of two subjects, and there may not be a clear clustered structure among those pairwise classification tasks, and therefore, a cluster norm will be ineffective. In this case, using a task similarity matrix may be more appropriate than clustering over tasks. It is also interesting to notice that MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$ achieves the best performance and shows advantages over other models, especially if given relatively sufficient training data (5 or 7 per subject). Compared to MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$), MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$ imposes restrictions on diagonal entries of task covariance $\mathbf{\Omega}$: all tasks seem to be similarly difficult and should be equally regularized. Compared to MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=\Sigma_{jj}=1}$, on the other hand, MTL($\mathbf{\Omega}$&$\mathbf{\Sigma}$)$_{\Omega_{ii}=1}$ frees the diagonal entries of feature covariance $\mathbf{\Sigma}$ and therefore is able to capture varying degrees of importance of multiple Laplacianfaces.

4.4 Conclusion

In this chapter, we propose a matrix-variate normal penalty with sparse inverse covariances for learning multiple tasks. Learning multiple (parametric) models can be viewed as estimating a pa-

rameter matrix, whose rows and columns correspond to tasks and features. Matrix-variate normal distributions are powerful tools for characterizing the structure of a matrix. We follow the matrix normal density and design a penalty that decomposes the full covariance of matrix elements into the Kronecker product of row covariance and column covariance, which characterize task relations and feature representations, respectively. We then perform sparse covariance selection (via ℓ_1 penalties) on the inverse of task and feature covariances in order to automatically select meaningful task and feature structures. Recent methods on discovering common feature structures among tasks and directly inferring task similarity are variants of the special cases of our formulation.

4.5 Appendix A

Proof of Lemma 1.

We prove lemma 1 by construction. Given an arbitrary choice of λ, λ_Ω and $\lambda_\Sigma > 0$ in eq. (4.5) and an optimal solution $(\mathbf{W}^*, \mathbf{\Omega}^*, \mathbf{\Sigma}^*)$, we want to prove that $\mathbf{W}^*$ also belongs to an optimal solution for eq. (4.5) with certain λ', λ'_Ω and λ'_Σ s.t. $\lambda'_\Omega = \lambda'_\Sigma$. Let's construct λ', λ'_Ω and λ'_Σ as follows:

$$(\lambda', \lambda'_\Omega, \lambda'_\Sigma) = (\lambda, \sqrt{\lambda_\Omega \lambda_\Sigma}, \sqrt{\lambda_\Omega \lambda_\Sigma}) \tag{4.13}$$

We denote the objective function in eq. (4.5) with λ, λ_Ω and λ_Σ as $Obj^{\lambda,\lambda_\Omega,\lambda_\Sigma}(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Sigma})$. Also, we denote the objective function with our constructed parameters λ', λ'_Ω and λ'_Σ as $Obj^{\lambda',\lambda'_\Omega,\lambda'_\Sigma}(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Sigma})$.

For any $(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Sigma})$, we further construct an invertible (i.e., one-to-one) transform as follows:

$$(\mathbf{W}', \mathbf{\Omega}', \mathbf{\Sigma}') = (\mathbf{W}, \sqrt{\frac{\lambda_\Sigma}{\lambda_\Omega}}\mathbf{\Omega}, \sqrt{\frac{\lambda_\Omega}{\lambda_\Sigma}}\mathbf{\Sigma}) \tag{4.14}$$

The key step in our proof is that, by construction, the following equality always holds:

$$Obj^{\lambda,\lambda_\Omega,\lambda_\Sigma}(\mathbf{W}, \mathbf{\Omega}, \mathbf{\Sigma}) = Obj^{\lambda',\lambda'_\Omega,\lambda'_\Sigma}(\mathbf{W}', \mathbf{\Omega}', \mathbf{\Sigma}') \tag{4.15}$$

To see this, notice that eq. (4.5) consists of three parts. The first part is the empirical loss on training examples, depending only on $\mathbf{W}$ (and training data). The second part is the log-density of matrix normal distributions, which depends on $\mathbf{W}$ and $\mathbf{\Sigma} \otimes \mathbf{\Omega}$. The third part is the sum of two ℓ_1 penalties. The equality in eq. (4.15) stems from the fact that all three parts of eq. (4.5) are not changed: 1) $\mathbf{W}' = \mathbf{W}$ so the first part remains unchanged; 2) $\mathbf{\Sigma}' \otimes \mathbf{\Omega}' = \mathbf{\Sigma} \otimes \mathbf{\Omega}$ so the second part of the matrix normal log-density is the same; 3) by our construction, the third part is not changed.

Based on this equality, if $(\mathbf{W}^*, \mathbf{\Omega}^*, \mathbf{\Sigma}^*)$ minimizes $Obj^{\lambda,\lambda_\Omega,\lambda_\Sigma}()$, we have that $(\mathbf{W}^*, \sqrt{\frac{\lambda_\Sigma}{\lambda_\Omega}}\mathbf{\Omega}^*, \sqrt{\frac{\lambda_\Omega}{\lambda_\Sigma}}\mathbf{\Sigma}^*)$ minimizes $Obj^{\lambda',\lambda'_\Omega,\lambda'_\Sigma}()$, where $\lambda' = \lambda$ and $\lambda'_\Omega = \lambda'_\Sigma = \sqrt{\lambda_\Omega \lambda_\Sigma}$.

Chapter 5

Learning Compressible Models

5.1 Overview

In this chapter [ZS10a], we study how to encode input information that is more general than just the covariance and correlation structure. We propose and study learning compressible models, which is based on the combination of compression and ℓ_1-norm regularization.

Since the introduction of lasso [Tib96], ℓ_1-regularization has become very popular for learning in high-dimensional spaces. A fundamental assumption of ℓ_1-regularization is model sparsity, i.e., a large fraction of model coefficients are zeros. Sparse models have the advantage of being easy to interpret and having good generalization ability in high-dimensional problems. However, the sparsity assumption might be too restrictive and not necessarily appropriate in many application domains. Indeed, many signals in the real world (e.g., images, audio, videos, time series) are found to be compressible (i.e., sparse in certain compressed domain) but not directly sparse in the observed space. Naturally, the sparsity assumption can be relaxed to compressibility.

Inspired by the recent development of compressive sampling (or compressed sensing) [Can06, Don06], we propose and study *learning compressible models*: certain structure information about the input space is available from domain knowledge and can be encoded as a compression operation, and by including this compression operation into the ℓ_1 regularization, model coefficients are compressed before being penalized and model sparsity is achieved in a compressed domain rather than in the original space. In this sense, the assumption on model sparsity is relaxed to model compressibility. We investigate the design of different compression operations, by which we can encode various compressibility assumptions and inductive biases, e.g., piecewise local smoothness, and compacted energy in the frequency domain. We conduct experiments on brain-computer interfacing and handwritten character recognition. Empirical results show clear improvements in prediction performance by including compression in ℓ_1 regularization. We also analyze the learned model coefficients under appropriate compressibility assumptions, which further demonstrate the advantages of learning compressible models instead of sparse models.

5.2 Learning Compressible Models

5.2.1 Lasso and Learning Sparse Models

Lasso [Tib96] is a specific example of ℓ_1-norm regularization, which is formulated as:

$$\min_{b,\mathbf{w}} ||\mathbf{y} - \mathbf{1}b - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1 \tag{5.1}$$

where the $n \times p$ matrix $\mathbf{X}$ contains n examples and p explanatory variables (i.e., features), and the $n \times 1$ vector $\mathbf{y}$ is the response variable (or 0/1 labels) of training examples. The sum of squares error $||\ ||_2^2$ is an instantiation of the empirical loss function on observations. Also, $\mathbf{1}$ is a column of 1s, and the intercept b and $p \times 1$ vector $\mathbf{w}$ are model parameters. The intercept b is usually separated from $\mathbf{w}$ and not penalized in regularization. The parameter λ is a balance between minimizing the empirical loss and controlling the model complexity, and is usually chosen by cross-validation.

A notable part of lasso is the use of ℓ_1 norm $||\mathbf{w}||_1$ as the regularization term. As the closest convex relaxation of ℓ_0-norm, ℓ_1-norm in regularization not only controls model complexity but also leads to sparse estimation [Tib96]. This provides both generalization ability and interpretable model coefficients. Analytical results also show that ℓ_1 regularization can consistently recover the true signal from noisy measurements [Tro06, ZY06], given that the true signal is sufficiently sparse.

5.2.2 Learning Compressible Models

To relax the sparsity assumption used by lasso to a more appropriate assumption, we consider the problem of learning compressible models as follows:

$$\min_{b,\mathbf{w}} L(\mathbf{y}, \mathbf{1}b + \mathbf{X}\mathbf{w}) + \lambda||\mathcal{P}(\mathbf{w})||_1 \tag{5.2}$$

The loss function L depends on the prediction model, e.g., sum of squares loss for linear regression, log-likelihood loss for logistic regression, hinge loss for SVMs, and so forth. The compression operation $\mathcal{P}()$ encodes our assumption on compressibility: model coefficients are compressed by $\mathcal{P}()$ before being penalized, and thus tend to follow the compression pattern (i.e., sparse in the compressed domain) rather than simply shrink to zero.

For simplicity, we restrict our attention to linear compression. Given that the compression operation is a linear and invertible[1] transform, learning compressible models is represented by:

$$\min_{b,\mathbf{w}} L(\mathbf{y}, \mathbf{1}b + \mathbf{X}\mathbf{w}) + \lambda||\mathbf{P}\mathbf{w}||_1 \tag{5.3}$$

The $p \times p$ matrix $\mathbf{P}$ denotes the linear and invertible compression transform, where p is the dimensionality of model coefficients as in (5.1). The optimization of eq. (5.3) can be achieved by applying the inverse compression transform (i.e., the decompression operation) to the input space

[1] A compression transform needs to be invertible, so that the compressed signal can be decompressed.

and then solving a standard ℓ_1 regularization [KKL+08]. First, transform the training examples by

$$\tilde{\mathbf{X}} = \mathbf{X}\mathbf{P}^{-1} \tag{5.4}$$

Second, solve a standard ℓ_1-regularized problem (e.g., lasso or sparse logistic regression):

$$\min_{b,\tilde{\mathbf{w}}} L(\mathbf{y}, \mathbf{1}b + \tilde{\mathbf{X}}\tilde{\mathbf{w}}) + \lambda||\tilde{\mathbf{w}}||_1 \tag{5.5}$$

Finally, the solution for eq. (5.3) is obtained by:

$$\mathbf{w} = \mathbf{P}^{-1}\tilde{\mathbf{w}} \tag{5.6}$$

$$b = b \tag{5.7}$$

This equivalence is derived from $\mathbf{X}\mathbf{w} = \mathbf{X}\mathbf{P}^{-1}\tilde{\mathbf{w}} = \tilde{\mathbf{X}}\tilde{\mathbf{w}}$ and $||\mathbf{P}\mathbf{w}||_1 = ||\mathbf{P}\mathbf{P}^{-1}\tilde{\mathbf{w}}||_1 = ||\tilde{\mathbf{w}}||_1$.

Why do we want to learn compressible models, which are not necessarily sparse in the original space? Compressible models are useful in several aspects. The first is model fitting and prediction accuracy. The inductive bias of model compressibility might be more appropriate than model sparsity, especially if an informative compression operation is specified based on additional information from domain knowledge, unlabeled data or related problems. The second reason, as claimed for standard sparse models, is interpretability. Model coefficients that are sparse in a compressed domain can still be insightful in the original space in many problems, as later shown by our empirical studies on brain-computer interface (in Section 5.5) and handwritten digit recognition (in Section 5.6). The third reason is that, when the compression operation is known in advance, compressible models are very efficient for storage and transmission in the compressed domain. This advantage has been widely recognized in compressive sensing [Can06, Don06] for general signals.

5.3 Model Compression: Local Smoothness

5.3.1 Order-1 Smoothness

Suppose we have a natural order over model coefficients $\{w_j\}_{j=1}^p$, e.g., in temporal domains where each dimension corresponds to a time point, or spectral domains where each dimension corresponds to a frequency. *Order-1 smoothness* assumes the coefficients "do not change very often" along the natural order. Such an assumption characterizes the first-order derivatives. It has been studied in fused lasso [TSR+05] where absolute values of the difference of successive coefficients, i.e., $\sum_{j=2}^p |w_j - w_{j-1}|$, are penalized[2]. This idea was also explored in total variation minimization for noise removal and image enhancement [ROF92]. As a motivating example, we show that the fused lasso penalty can be approximated by a linear and invertible compression in the ℓ_1 penalty.

[2]In fused lasso, standard ℓ_1-norm $\sum_{j=1}^p |w_j|$ and $\sum_{j=2}^p |w_j - w_{j-1}|$ are penalized together to pursue both sparsity and smoothness. We focus on smoothness part ($\sum_{j=2}^p |w_j - w_{j-1}|$) as a specific case of compressibility.

The $p \times p$ matrix $\mathbf{P}$ for model compression based on order-1 smoothness can be defined as:

$$\mathbf{P} = \mathbf{S}_p^1 = \begin{bmatrix} \frac{1}{p} & \frac{1}{p} & \dots & \dots & \dots & \frac{1}{p} \\ 1 & -1 & 0 & \dots & \dots & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \dots & \vdots \\ 0 & 0 & \dots & \dots & 1 & -1 \end{bmatrix} \tag{5.8}$$

Model coefficients in the compressed domain $\mathbf{Pw} = [\overline{w}, w_1 - w_2, \dots, w_{p-1} - w_p]$ tend to be sparse due to ℓ_1 regularization, which achieves the order-1 smoothness. The averaging operation in the first row of $\mathbf{P}$ makes the transform invertible. Note that if the first row of $\mathbf{P}$ is multiplied by a small constant (e.g., 0.00001), $||\mathbf{Pw}||_1$ approximates the fused lasso penalty. In our study, we use the compression in eq. (5.8) without scaling the averaging operation. We want the compression operation to be invertible so that the optimization is efficient as discussed in eq. (5.4) - eq. (5.6).

5.3.2 Order-2 Smoothness and Higher-Order Smoothness

Smoothness of higher orders is also common. For example, a piecewise linear function has piecewise constant first-order derivatives, indicating zero second-order derivatives at most locations. This is defined as *order-2 smoothness*. In this case, the $p \times p$ compression transform $\mathbf{P}$ can be:

$$\mathbf{P} = \mathbf{S}_p^2 = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{S}_{p-1}^1 \end{bmatrix} \cdot \mathbf{S}_p^1 \tag{5.9}$$

where $\mathbf{0}$ is a $(p-1) \times 1$ column vector. By this definition, model coefficients in the compressed domain are $\mathbf{Pw} = [\overline{w}, \overline{\Delta w}, \Delta w_{1,2} - \Delta w_{2,3}, \Delta w_{2,3} - \Delta w_{3,4}, \dots, \Delta w_{p-2,p-1} - \Delta w_{p-1,p}]$, where $\Delta w_{i,i+1} = w_i - w_{i+1}$. In this sense, sparsity of $\mathbf{Pw}$ corresponds to order-2 smoothness assumption in the original space. Note that $\mathbf{S}_p^2$ is invertible since both $\mathbf{S}_{p-1}^1$ and $\mathbf{S}_p^1$ are invertible.

Also, model compression for higher-order smoothness can be defined recursively.

5.3.3 Hybrid Smoothness

Sometimes features under consideration do not follow an universal order, but can be divided into groups, where each group of features has an order or at least some groups of features are ordered. The compression operation can be defined as a block matrix to handle the use of different groups of features. For example, suppose features can be divided into three groups. We assume p_1 model coefficients on the first group of features satisfy order-1 smoothness, p_2 coefficients on the second group of features satisfy order-2 smoothness, and we have no knowledge about the third group of p_3 features. In this case, model compression is defined as:

$$\mathbf{P} = \begin{bmatrix} \mathbf{S}_{p_1}^1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{p_2}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{p_3} \end{bmatrix} \tag{5.10}$$

5.4 Model Compression: Energy Compaction

Another compressibility assumption is energy compaction in the frequency domain. The energy of many real-world signals can be compacted by transforming signals to a frequency domain where most of their energy is concentrated in a few frequencies, e.g., image compression [Wal92, CSE00]. If the target model is applied to classify objects (e.g., images) with compacted energy in a frequency domain, it is reasonable to assume that the model only needs to operate on a few relevant frequencies and thus also has compacted energy in the same frequency domain. Otherwise, most energy of the model is wasted. Naturally, we can include an appropriate compression operation on w within the ℓ_1 penalty to emphasize energy compaction in a frequency domain.

The discrete cosine transform (DCT) is used in the JPEG standard [Wal92], which compresses an image by representing it as a sum of cosine functions at various frequencies, and as a result, small coefficients after compression can be discarded. The 2D DCT for an $m \times n$ object is:

$$
\begin{aligned}
\mathbf{G}'(u,v) \quad &= \quad \frac{2}{\sqrt{mn}}\Lambda(u)\Lambda(v)\sum_{y=0}^{m-1}\sum_{x=0}^{n-1} \\
&\quad \mathbf{G}(x,y)\cos\frac{(2x+1)u\pi}{2n}\cos\frac{(2y+1)v\pi}{2m} \\
\textit{where} \quad &\quad u = 0, 1, \ldots, n-1 \\
&\quad v = 0, 1, \ldots, m-1 \\
&\quad \Lambda(t) = \begin{cases} 2^{-\frac{1}{2}} & if\ t = 0 \\ 1 & otherwise \end{cases}
\end{aligned}
\tag{5.11}
$$

The above formula is a linear operation on $m \times n$ matrices, and can be rewritten as a linear operation on $p \times 1$ vectors, where $p = mn$ is the dimension of linear models on images. This gives a $p \times p$ matrix $\mathbf{P}$. Combining such a compression operation with ℓ_1-norm regularization leads to sparse models in an appropriate frequency domain, representing the compacted energy assumption on model coefficients. Note that real-world image compression operations are more sophisticated [Wal92, CSE00], but studying sophisticated image codings is not the focus of our work.

5.5 Empirical Study: Brain-Computer Interface

In this section, we report our empirical study on brain-computer interface data [B+04]: classifying single-trial Electroencephalography (EEG) signals. The EEG signals contain important information from human brains, and being able to read and understand those signals is a critical step for human-computer interaction. An EEG signal contains multiple channels (i.e., multiple scalp positions), and each channel is sampled over time to produce sequential measurements. As a result, an EEG signal is a multivariate time series. If we assume local smoothness over time on a univariate time series, the hybrid smoothness assumption in Section 5.3.3 is useful for EEG signals.

Table 5.1: Classification errors on EEG brain signals: means (standard errors) over 50 random runs

	means (standard errors) over 50 runs
Lasso	30.22%(0.34%)
LassoCP	25.98%(0.29%)
SLgr	30.00%(0%)
SLgrCP	20.92%(0.16%)

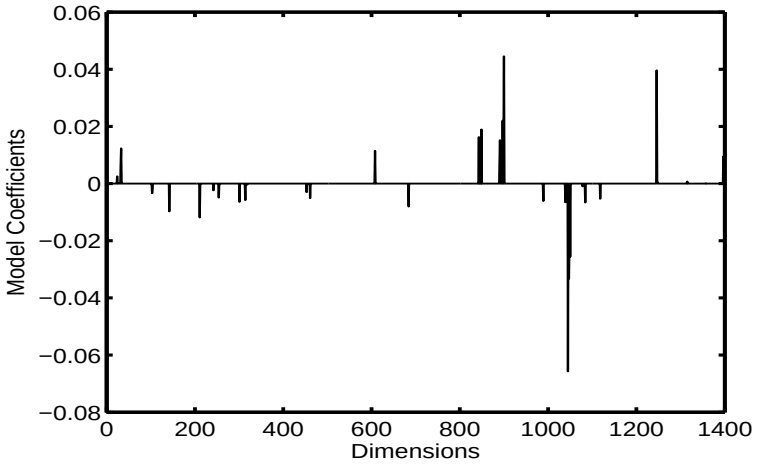
Descriptions of the Data Set. We use data set IV, self-paced tapping, of BCI Competition 2003 [B+04], which is a binary classification task. The task contains a training set of 316 examples and a testing set of 100 examples. Each example has 1400 features, corresponding to 28 channels and 50 measurements from each channel. The number of features is much larger than the number of training examples, indicating the importance of regularization. Each example is measured when a healthy subject, sitting in a chair with fingers in the standard typing position, tries to press the keys using either the left hand or right hand. The objective is to classify an EEG signal to either a left-hand movement or right-hand movement.

Models for Comparison and Implementation Details. We consider lasso with labels as $\{+1, -1\}$ (denoted as Lasso in our discussions), sparse logistic regression (denoted as SLgr), compressible lasso (denoted as lassoCP), and compressible logistic regression (denoted as SLgrCP). Following eq. (5.10), the compression operation $\mathbf{P}$ for learning two compressible models is a 1400×1400 block matrix, with 28 blocks and each block is an order-1 smoothness matrix $\mathbf{S}_{50}^{1}$ defined as in eq. (5.8). Lasso is implemented using the spgl1 Matlab solver[3], and ℓ_1-regularized logistic regression is implemented as [LLAN06] using lasso. The regularization parameter is chosen from 10^{-7} to 10^{7} (step $10^{0.5}$) by 5-fold cross-validation.

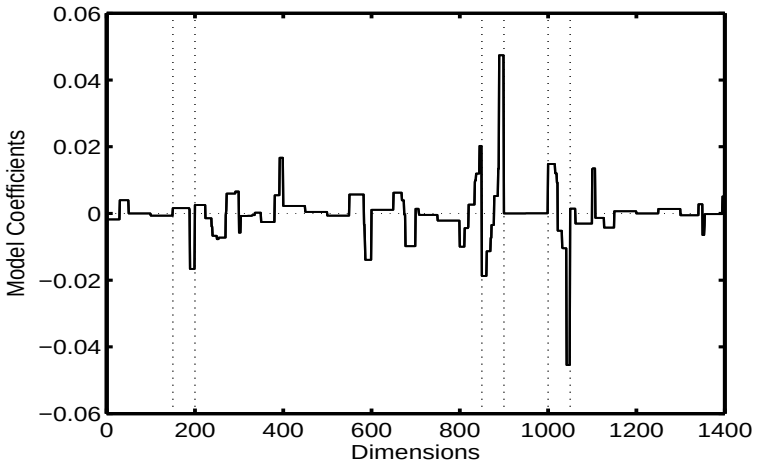
Experimental Procedures. The data set contains a fixed training and testing set for the competition. For standard ℓ_1 regularized models, we train lasso and sparse logistic regression. For compressible models, we train compressible lasso and compressible logistic regression. We learn the four models from the training set and measure their classification errors on the testing set. Note that there is still randomness in the procedure: cross-validation is used to determine the optimal regularization parameter λ. Therefore, we have 50 random runs on each model.

Experimental Results and Analysis. Classification errors are shown in Table 5.1, with both means and standard errors (of means) over 50 random runs. By learning compressible models instead of learning sparse models, the testing error is reduced from 30.22% to 25.98% for lasso (Lasso vs. LassoCP), and from 30% to 20.92% for logistic regression (SLgr vs. SLgrCP). During the competition, 15 submissions were received [B+04]. The best submission achieves 16% error, using features "based on Bereitschaftspotential and event-related desynchronization" [W+04]. The 2nd best submission achieved 19% using 188 time-based, frequency-based, and correlational features "compiled by hand" [B+04]. For the other 13 submissions, six attained errors between 23% and 29%, and the other seven were worse. In our study, compressible logistic regression using 1400 raw features are comparable to the two best submissions with domain-specific features.

[3] http://www.cs.ubc.ca/labs/scl/spgl1/



(a) Sparse LGR coefficients



(b) Compressible LGR coefficients

Figure 5.1: Model coefficients of sparse and compressible (i.e., piecewise smooth) logistic regression on brain-computer interfacing (EEG signal classification)

We also plot the model coefficients learned by a sparse logistic regression and a compressible logistic regression in Figure 5.1. From the plot we have several interesting observations. 1) Sparse logistic regression learns sparse coefficients, and compressible LGR leads to (piecewise) smooth coefficients. These two different patterns represent the inductive biases we incorporate into the learning process (via different regularization penalties). 2) Although in the compressible logistic regression we mainly penalize the difference of successive coefficients, most learned coefficients are actually close to zero. The proposed regularization (piecewise local smoothness) effectively controls the model complexity not only in terms of smoothness but also in terms of the norm of coefficients. 3) In the compressible logistic regression, there still exist a few large coefficient jumps over successive dimensions (within the same channel): we plot in Fig. 5.1b the boundaries (vertical dashed lines) of three selected channels that contain large coefficient jumps. These jumps correspond to large coefficients in the compressed domain (recall that the compressed domain

Table 5.2: Classification errors over 45 tasks on MNIST, 10/20/50 training examples per class: means (and standard errors) over tasks

	10 examples per class	20 examples per class	50 examples per class
Lasso	9.96%	6.94%	4.91%
LassoCP	7.80%	5.30%	3.45%
SLgr	9.79%	6.24%	3.91%
SLgrCP	7.46%	4.99%	3.26%
(Lasso - LassoCP)	2.16%(0.23%)	1.64%(0.18%)	1.46%(0.12%)
(SLgr - SLgrCP)	2.33%(0.21%)	1.25%(0.16%)	0.65%(0.09%)

Table 5.3: Performance comparison on individual tasks between compressible and sparse models on MNIST, 10/20/50 training examples per class: #win/#loss over 45 tasks

	10 examples per class	20 examples per class	50 examples per class
LassoCP vs. Lasso	41/4	42/3	44/1
SLgrCP vs. SLgr	43/2	42/3	40/5

defined by our compression operation is composed of the difference of successive coefficients within the same channel in the original space). The existence of a few large coefficients in the compressed domain is consistent with the notation of compressibility: most information of the original signal is concentrated on a few components after being compressed. This is achieved by performing ℓ_1 regularization[4] in the compressed domain rather than the original domain.

5.6 Empirical Study: Handwritten Character Recognition

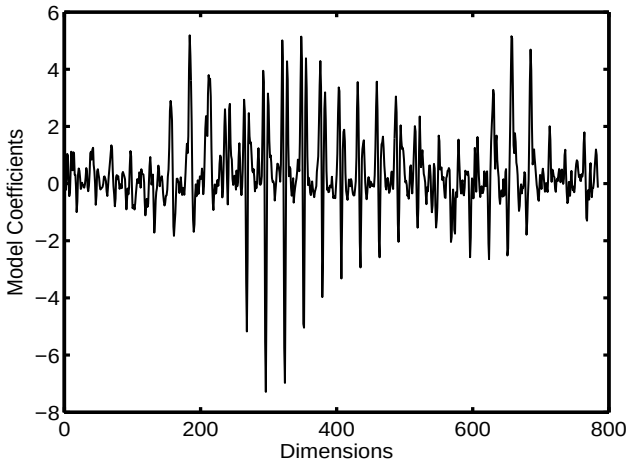
In this section, we study handwritten character recognition on images. As discussed in Section 5.4, compacted energy in the frequency domain can be used as the inductive bias for regularization, assuming that the model only needs to operate on a few frequencies to classify images.

Descriptions of the Data Set. We use the MNIST handwritten digits data set[5], which has 70000 images for 10 digits (from 0 to 9). Images are represented by pixels (in grayscale). The number of features is $p = 784$, corresponding to 28×28 pixels of an image.

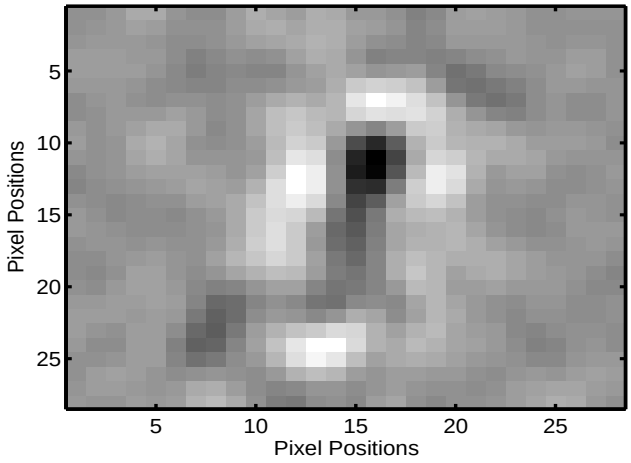
Experimental Procedures. We construct 45 binary classification tasks, each to classify two digits. For each task, a few labeled examples of the two digits are selected from the training set (e.g., 10, 20, or 50 images *per class* in our experiments). As a result, we aim to learn classifiers in a high-dimensional space (784 dimensions) using only a few training examples. Performance for each task is averaged from 20 random runs, with training data randomly selected. For each task, the testing data are fixed as all the images of the two target digits in the testing set.

Model and Implementation Details. The standard ℓ_1 regularized models: lasso and sparse

[4]Ideally, ℓ_0 norm is the best for allowing a few large coefficients, while ℓ_1 norm is the closest convex relaxation.
[5]http://yann.lecun.com/exdb/mnist/



(a) Compressible model coefficients as a 784 dimensional vector (intercept b omitted)



(b) Compressible model coefficients as a 28×28 image (the intercept b omitted)

Figure 5.2: Model coefficients of a compressible logistic regression on MNIST. Task: classifying "1" vs. "8".

logistic regression are the same as in Section 5.5. The model compression operation used for learning compressible lasso and compressible logistic regression is the DCT operation in eq. (5.11). By using a DCT operation in the ℓ_1 penalty, we impose the assumption that model coefficients should be sparse in the DCT frequency domain, implying that the model only needs to operate in a few frequencies. Others implementation details are the same as in Section 5.5.

Experimental Results and Analysis. Results are shown in Table 5.2–Table 5.3 and Figure 5.2. Table 5.2 has two parts. The first part shows average classification errors over 45 tasks of lasso (Lasso), compressible Lasso (LassoCP), sparse logistic regression (SLgr) and compressible logistic regression (SLgrCP), where models are learned using 10, 20 and 50 training examples per class in each task. We omit standard errors (of classification errors) over 45 tasks since they correspond

to the variation of the difficulty of different tasks, which is not of interest. The second part (the last two rows) of Table 5.2 are means (and also standard errors) over 45 tasks for the *difference* of classification errors between a sparse and a compressible model. From Table 5.2 we can see that compressible lasso and compressible logistic regression generally outperform their sparse counterparts. The less the training examples available for learning, the more obvious the advantage of compressible models over sparse models. We also compare the classification performance of compressible and sparse models on each individual task in Table 5.3. The results shows that the majority of tasks benefit from learning compressible models instead of learning sparse models.

In addition, we plot in Figure 5.2 the model coefficients of a compressible logistic regression from a random run, where the task is to classify "1" (negative class) and "8" (positive class). As shown in Fig. 5.2a, model coefficients in the original space (which correspond to image pixels) are not sparse, but they are mainly changing in a few frequencies, indicating sparsity in the compressed (DCT) space, i.e., has compacted energy in the frequency domain. Interestingly, when we plot the model coefficients as a 28×28 image in Fig. 5.2b, the difference between the negative class "1" and the positive class "8" is well emphasized. The model coefficients, although only sparse in the DCT frequency domain, can actually represent a meaningful pattern in the original input space. This should not be a surprise: DCT is the compression operation in the JPEG standard for compressing images, and most useful information in an image tends to be sparse in the DCT domain.

5.7 Conclusion

In this chapter, we study how to encode input information that is more general than just the covariance and correlation structure. We propose learning compressible models: certain structure information about the input space can be encoded as a compression operation, and by including a compression operation into ℓ_1 regularized learning, model coefficients are compressed before being penalized and sparsity is achieved in a compressed domain. This relaxes the assumption on model sparsity to model compressibility, and provides an opportunity encode more appropriate inductive biases, e.g., piecewise local smoothness, and compacted energy in the frequency domain. We conduct experiments on brain-computer interfacing and handwritten character recognition. Empirical results show significant improvements in prediction performance by including compression in the ℓ_1-norm penalty. We also analyze the learned model coefficients under different compressibility assumptions, which further demonstrate the advantages of learning compressible models instead of sparse models.

Chapter 6

Projection Penalties: Learning Safely with Dimension Reduction

6.1 Overview

In this chapter [ZS10c], we study the scenario that certain structure information about the input space is highlighted by a dimension reduction, e.g., a subset of features, a clustering of low-level features, or a subspace (or manifold) of the input space. In this case, we want to encode this dimension reduction into learning while controlling the risk of information loss during the reduction.

Learning in high-dimensional spaces with limited training examples is difficult. As a result, researchers have invented various dimension reduction techniques to reduce the feature space. A dimension reduction method may focus on feature selection or feature extraction, can be linear or nonlinear, and may utilize the target variable (supervised) or not (unsupervised). Despite the wide variety of approaches, dimension reduction is useful for learning predictive models because it can discover and highlight the relevant part of the feature space in fewer dimensions. However, performing dimension reduction and then learning in the reduced space can also degrade the prediction performance, because any reduction discards information, and it is hard to tell whether the information lost by a dimension reduction procedure is relevant to a prediction task or not.

We propose the *projection penalty* framework to encode the input information from a dimension reduction while controlling the risk of losing relevant information during the reduction. The idea is shown in Fig 6.1. Reducing the feature space from R^p to R^d ($d < p$) using a dimension reduction P before learning predictive models can be viewed as restricting the model search in full model space $\mathcal{M}$ to certain model subspace $\mathcal{M}_P$. The idea of projection penalties is that instead of completely restricting model search to the model subspace $\mathcal{M}_P$, we can still search the model $\mathbf{w}$ in the full model space $\mathcal{M}$ but penalize the projection distance to $\mathcal{M}_P$. In this sense, the dimension reduction P is used to guide the model search, rather than to restrict the model search.

We first introduce projection penalties for linear dimension reduction, and then generalize to kernel-based reduction and other nonlinear methods. We test projection penalties with various dimension reduction techniques in different prediction tasks, including principal component regres-

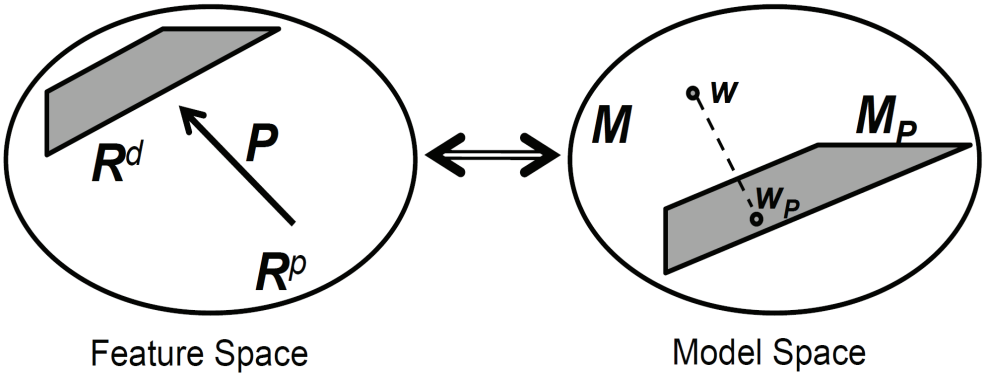


Figure 6.1: The idea of projection penalties

sion and partial least squares in regression tasks, kernel dimension reduction in face recognition, and latent topic modeling in text classification. Experimental results show that projection penalties offer a more effective and reliable way to make use of dimension reduction techniques.

6.2 Projection Penalties

6.2.1 Dimension reduction and parameter subspace

Consider learning a linear prediction model $(\mathbf{w}, b)$ in a p-dimensional space by minimizing an empirical loss L given a set of n training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$:

$$\underset{\mathbf{w}\in R^p, b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) \tag{6.1}$$

where the parameter vector $\mathbf{w} \in R^p$ and the intercept b represent the prediction model, and $\mathbf{x}_i$ and y_i are the p-dimensional feature vector and the response variable of the ith example, respectively. The empirical loss L depends on the choice of prediction models, e.g., squared error loss, logistic log-likelihood loss or hinge loss. If the dimension of the feature space is high, a penalty term $J(\mathbf{w})$ may be added to control the model complexity, such as an ℓ_2-norm penalty in ridge regression [TA77] and support vector machines [Vap95] and an ℓ_1-norm penalty in lasso [Tib96].

A linear reduction of a p-dimensional feature space can be represented as a $d \times p$ matrix P where $d < p$ is the dimension of the reduced feature space. For an example $\mathbf{x}$ in the original feature space, $P\mathbf{x}$ is the representation in the reduced space. In this sense, performing a linear dimension reduction and then learning predictive models in the reduced space can be written as:

$$\underset{\mathbf{v}\in R^d, b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathbf{v}^T(P\mathbf{x}_i) + b) \tag{6.2}$$

where $\mathbf{v} \in R^d$ is the parameter vector learned in the reduced feature space. Eq. (6.2) can be

rewritten as:

$$\underset{\mathbf{v}\in R^d,b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, (P^T\mathbf{v})^T\mathbf{x}_i + b) \tag{6.3}$$

Comparing eq. (6.3) with eq. (6.1), we see the connection between a linear reduction of the feature space and the restriction on the model parameter space. Specifically, performing a linear reduction P in the feature space simply corresponds to confining the p-dimensional parameter vector $\mathbf{w}$ to the following subspace:

$$\mathcal{M}_P = \{\mathbf{w} \in R^p \mid \mathbf{w} = P^T\mathbf{v},\ \exists \mathbf{v} \in R^d\} \tag{6.4}$$

In this sense, eq. (6.2) is equivalent to:

$$\underset{\mathbf{w}\in\mathcal{M}_P,b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) \tag{6.5}$$

6.2.2 Projection penalties for linear reduction

In eq. (6.5) we see that performing a linear dimension reduction P in the feature space is equivalent to restricting the model search to a parameter subspace $\mathcal{M}_P$ as defined in eq. (6.4). From the perspective of model search, we eliminate all candidates that are not in $\mathcal{M}_P$. However, there is no guarantee that, for a prediction task, the optimal model in R^p must belong to the subspace $\mathcal{M}_P$.

We propose to search models in the full parameter space and penalize the projection distance to $\mathcal{M}_P$. This leads to the formulation of a projection penalty for a linear dimension reduction P:

$$\underset{\mathbf{w}\in R^p,b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) + \min_{\mathbf{w}^*\in\mathcal{M}_P} \lambda J(\mathbf{w} - \mathbf{w}^*)$$

or using the definition of $\mathcal{M}_P$ in (6.4), we have:

$$\underset{\mathbf{w}\in R^p,b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\mathbf{x}_i + b) + \min_{\mathbf{v}\in R^d} \lambda J(\mathbf{w} - P^T\mathbf{v}) \tag{6.6}$$

where λ is a regularization parameter usually chosen by cross validation, $J()$ is a penalty function such as $\|\|_2^2$ or $\|\|_1$, $\mathbf{v} \in R^d$ is a parameter vector on the reduced feature space and $\mathbf{w}^* = P^T\mathbf{v}$ defines a member in the parameter subspace $\mathcal{M}_P$. Eq. (6.6) allows us to learn the parameter vector $\mathbf{w}$ in the full parameter space R^p and penalize the part of $\mathbf{w}$ that can *not* be interpreted by models in $\mathcal{M}_P$. This is different from directly performing a dimension reduction P, as shown in eq. (6.5), where model search is completely restricted to the parameter subspace $\mathcal{M}_P$.

It is straightforward to solve eq. (6.6) by a change of notation $\tilde{\mathbf{w}} = \mathbf{w} - P^T\mathbf{v}$:

$$\underset{\tilde{\mathbf{w}}\in R^p,\mathbf{v}\in R^d,b}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \tilde{\mathbf{w}}^T\mathbf{x}_i + \mathbf{v}^T(P\mathbf{x}_i) + b) + \lambda J(\tilde{\mathbf{w}}) \tag{6.7}$$

This can viewed as redefining the representation for each training example $\mathbf{x}_i$ as a $(p+d)$-dimensional vector $[\mathbf{x}_i; (P\mathbf{x}_i)]$, and solving the regularized linear model $([\tilde{\mathbf{w}}; \mathbf{v}], b)$ on the new representation. The specific algorithm for solving eq. (6.7) depends on the choice of empirical loss L and penalty function J, e.g., conjugate gradient methods, subgradient methods, or methods for constrained optimization [BV04]. The parameter vector $\mathbf{w}$ can be computed as:

$$\mathbf{w} = \tilde{\mathbf{w}} + P^T\mathbf{v}$$

6.2.3 Projection penalties for kernel-based reduction

Section 6.2.2 shows that a linear dimension reduction can be used to penalize a linear model via a projection penalty, as in eq. (6.6). A limitation of eq. (6.6) is that the resulting model is still a linear model. Recently, SVMs and other kernel-based learning algorithms have drawn considerable interest in machine learning [MMR+01]. In this section, we generalize projection penalties to dimension reduction and predictive modeling in kernel feature spaces.

Consider learning a prediction model $\mathbf{w}$ given a set of n training examples $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and a dimension reduction operator P, where both the prediction model and the reduction operator are designed to operate on a kernel feature space $\mathcal{F}$. Each training example is represented as $\Phi(\mathbf{x}_i) \in \mathcal{F} \subseteq R^p$ using the kernel feature mapping Φ. The feature mapping Φ is characterized by its inner product via a Mercer kernel $k(\cdot,\cdot)$:

$$(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j)$$

Depending on the choice of kernel functions, the kernel feature space is usually a very high (or infinite) dimensional space and thus p is large or infinite.

Analogously to eq. (6.6), the projection penalty is:

$$\operatorname*{argmin}_{\mathbf{w}\in R^p, b} \sum_{i=1}^{n} L(y_i, \mathbf{w}^T\Phi(\mathbf{x}_i) + b) + \min_{\mathbf{v}\in R^d} \lambda J(\mathbf{w} - P^T\mathbf{v}) \tag{6.8}$$

where $\mathbf{w}$, $\mathbf{v}$ and P are of size $p \times 1$, $d \times 1$ and $d \times p$, respectively. The reduction operator P is provided by a kernel-based dimension reduction such as kernel PCA [SSM98] or generalized discriminant analysis [BA00]. Thus, each $p \times 1$ column basis of P^T is represented by a weighted combination of examples in kernel feature space $\mathcal{F}$.

To solve $\mathbf{w}$ in eq. (6.8), we consider classification problems and use the hinge loss of SVMs as the empirical loss $L()$ and the ℓ_2-norm penalty as $J()$. By introducing slack variables $\{\xi_i\}_{i=1}^n$ for hinge loss [Bur98], we have the following quadratic program:

$$\begin{aligned} \operatorname*{argmin}_{\mathbf{w}\in R^p, \mathbf{v}\in R^d, b, \{\xi_i\}_{i=1}^n} \quad & \frac{1}{2} \parallel \mathbf{w} - P^T\mathbf{v} \parallel_2^2 + C\sum_{i=1}^{n} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned} \tag{6.9}$$

where we use $C \propto \frac{1}{\lambda}$ to replace λ in eq. (6.8). Also by changing to $\tilde{\mathbf{w}} = \mathbf{w} - P^T\mathbf{v}$, we have:

$$\begin{aligned} \underset{\tilde{\mathbf{w}}\in R^p, \mathbf{v}\in R^d, b, \{\xi_i\}_{i=1}^n}{\operatorname{argmin}} \quad & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + C\sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \forall i \\ & \xi_i \geq 0, \ \ \forall i \end{aligned} \tag{6.10}$$

Note that the reduction of examples from $\mathcal{F}$ into d-dimensional space, i.e., $P\Phi(\mathbf{x}_i)$, is actually performed via kernel tricks [MMR+01].

Since the primal problem (6.10) is convex, the KKT conditions are sufficient and necessary for optimal primal and dual solutions [BV04]. To derive KKT conditions and the dual, we add Lagrangian multipliers $\{\alpha_i\}_{i=1}^n$ and $\{\mu_i\}_{i=1}^n$ for the constraints in the primal (6.10), which gives the Lagrangian:

$$\begin{aligned} \mathcal{L} \ = \ & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + C\sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i y_i(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b) \\ & - \sum_{i=1}^n \alpha_i(\xi_i - 1) - \sum_{i=1}^n \mu_i\xi_i \\ & \text{s.t. } \alpha_i \geq 0, \ \mu_i \geq 0, \ \forall i \end{aligned}$$

The KKT optimality conditions include:

$$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{w}}} = \tilde{\mathbf{w}} - \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i) = 0 \tag{6.11}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = -\sum_{i=1}^n \alpha_i y_i P\Phi(\mathbf{x}_i) = 0 \tag{6.12}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^n \alpha_i y_i &= 0 \\ \frac{\partial \mathcal{L}}{\partial \xi_i} = C - \alpha_i - \mu_i &= 0 \\ y_i(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b) - 1 + \xi_i &\geq 0 \\ \xi_i &\geq 0 \\ \alpha_i &\geq 0 \\ \mu_i &\geq 0 \\ \alpha_i[y_i(\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b) - 1 + \xi_i] &= 0 \\ \mu_i\xi_i &= 0 \end{aligned}$$

Using the above conditions to eliminate primal variables in the Lagrangian, we have the following dual problem:

$$\underset{\{\alpha_i\}_{i=1}^n}{\text{argmax}} \quad \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{6.13}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \ \ \forall i$$

$$\sum_{i=1}^n \alpha_i y_i = 0 \tag{6.14}$$

$$\sum_{i=1}^n \alpha_i y_i P\Phi(\mathbf{x}_i) = 0 \tag{6.15}$$

where the kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ computes $\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_j)$. This dual can be solved as a quadratic programming problem, and $\tilde{\mathbf{w}}$, according to condition (6.11), is:

$$\tilde{\mathbf{w}} = \sum_{i=1}^n \alpha_i y_i \Phi(\mathbf{x}_i) \tag{6.16}$$

Note that the dual form (6.13) is similar to the dual form of SVMs [Bur98], with one key difference: the SVM dual form has only one equality constraint (6.14), while the above dual form has additional d equality constraints (6.15). These d additional constraints basically say that dual solutions $\{\alpha_i\}_{i=1}^n$ and the resulting $\tilde{\mathbf{w}}$ do *not* operate on the reduced feature space (w.r.t. the training examples). This is very intuitive: $\tilde{\mathbf{w}}$ in the primal form (6.10) is defined by a change of variable: $\tilde{\mathbf{w}} = \mathbf{w} - P^T\mathbf{v}$. Thus, $\tilde{\mathbf{w}}$ refers to the part of $\mathbf{w}$ that can not be replaced by operating on the reduced feature space. This also coincides with the fact that the constraints (6.15) result from the KKT condition (6.12), which says the derivative of the Lagrangian w.r.t. $\mathbf{v} \in R^d$ is zero. Indeed, if $\tilde{\mathbf{w}}$ is still operating on the reduced space, then $\mathbf{v} \in R^d$ should be changed to take the responsibility of $\tilde{\mathbf{w}}$ since $\mathbf{v}$ is not penalized in the primal form (6.10).

To solve for $\mathbf{v}$ and b, we plug the solution to $\tilde{\mathbf{w}}$ (solved as (6.16)) into the primal form (6.10). The quadratic term becomes a constant, and therefore, $\mathbf{v}$ and b (and slack variables $\{\xi_i\}_{i=1}^n$) can be solved efficiently in (6.10) as a linear programming problem. Note that both $\tilde{\mathbf{w}}^T\Phi(\mathbf{x}_i)$ and $P\Phi(\mathbf{x}_i)$ in the constraints are computed via kernel tricks and treated as known. Finally, given $\tilde{\mathbf{w}}$, $\mathbf{v}$ and b, the prediction on a new example $\mathbf{x}$ is:

$$\tilde{\mathbf{w}}^T\Phi(\mathbf{x}) + \mathbf{v}^T P\Phi(\mathbf{x}_i) + b$$

6.2.4 Regularization in the parameter subspace

Section 6.2.2 and 6.2.3 propose projection penalties for linear dimension reduction and kernel-based dimension reduction. In both cases, there is no penalty on the the parameter vector $\mathbf{v} \in R^d$ that operates on the reduced feature space. In some applications where the number of training examples is very small or the dimensionality of the reduced space is relatively high, a small penalty

on $\mathbf{v} \in R^d$ will be helpful for both the generalization of the model and the numerical stability for optimization. Including a small penalty on $\mathbf{v} \in R^d$ into eq. (6.6) or eq. (6.7) is straightforward and will not affect the optimization algorithm. But for projection penalties in kernel feature space, such as in eq. (6.10), including another penalty will change the dual form in a non-trivial way, and thus is the focus of this section.

We start from eq. (6.10) and add a new penalty on $\mathbf{v}$:

$$\begin{aligned} \underset{\tilde{\mathbf{w}} \in R^p, \mathbf{v} \in R^d, b, \{\xi_i\}_{i=1}^n}{\operatorname{argmin}} \quad & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + \frac{\gamma}{2} \parallel \mathbf{v} \parallel_2^2 + C \sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & y_i(\tilde{\mathbf{w}}^T \Phi(\mathbf{x}_i) + \mathbf{v}^T P \Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned} \tag{6.17}$$

where $0 \leq \gamma \leq 1$ is a new regularization parameter. Adding Lagrangian multipliers $\{\alpha_i\}_{i=1}^n$ and $\{\mu_i\}_{i=1}^n$ for the two sets of constraints, the Lagrangian is:

$$\begin{aligned} \mathcal{L} \quad = \quad & \frac{1}{2} \parallel \tilde{\mathbf{w}} \parallel_2^2 + \frac{\gamma}{2} \parallel \mathbf{v} \parallel_2^2 + C \sum_{i=1}^n \xi_i \\ & - \sum_{i=1}^n \alpha_i y_i (\tilde{\mathbf{w}}^T \Phi(\mathbf{x}_i) + \mathbf{v}^T P \Phi(\mathbf{x}_i) + b) \\ & - \sum_{i=1}^n \alpha_i (\xi_i - 1) - \sum_{i=1}^n \mu_i \xi_i \\ & \text{s.t. } \alpha_i \geq 0, \ \mu_i \geq 0, \ \forall i \end{aligned}$$

Most KKT conditions remain the same as in Section 6.2.3, except condition (6.12), which is now:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \gamma \mathbf{v} - \sum_{i=1}^n \alpha_i y_i P \Phi(\mathbf{x}_i) = 0 \tag{6.18}$$

Given the new Lagrangian and the new set of KKT conditions, the dual becomes:

$$\begin{aligned} \underset{\{\alpha_i\}_{i=1}^n}{\operatorname{argmax}} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ & - \frac{1}{2\gamma} \sum_{i,j} \alpha_i \alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T (P\Phi(\mathbf{x}_j)) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \quad \forall i \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned} \tag{6.19}$$

Comparing this dual form to the dual (6.13), the constraints (6.15) are removed, and instead, the quadratic term in the objective now includes a new part: $-\frac{1}{2\gamma} \sum_{i,j} \alpha_i \alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T (P\Phi(\mathbf{x}_j))$.

We can see the connection between the dual problems (6.19) and (6.13) by setting $\gamma \to 0$. In this case, the new dual problem has an infinite weight on $-\sum_{i,j} \alpha_i \alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T (P\Phi(\mathbf{x}_j))$. The matrix of inner products is positive semidefinite. Therefore, to maximize objective (6.19), the solution $\{\alpha_i\}_{i=1}^n$ must satisfy the constraints (6.15) to keep the infinitely weighted non-positive term $-\sum_{i,j} \alpha_i \alpha_j y_i y_j (P\Phi(\mathbf{x}_i))^T (P\Phi(\mathbf{x}_j))$ as zero. In this sense, the dual problem (6.19) is equivalent to the duel problem (6.13) when $\gamma \to 0$.

Given the solution $\{\alpha_i\}_{i=1}^n$ to (6.19), $\tilde{\mathbf{w}}$ and $\mathbf{v}$ are obtained via KKT conditions (6.11) and (6.18):

$$\begin{aligned} \tilde{\mathbf{w}} &= \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i) \\ \mathbf{v} &= \frac{1}{\gamma} \sum_{i=1}^{n} \alpha_i y_i P\Phi(\mathbf{x}_i) \end{aligned}$$

Finally, b can be obtained similarly as in SVMs [Bur98] via examples that lie on the margin (i.e., examples with $0 < \alpha_i < C$).

6.2.5 Adaption to other nonlinear reduction

In this section, we extend projection penalties to work with an arbitrary reduction operation. Recall that projection penalties proposed in Section 6.2.2 and 6.2.3 requires the reduction operator P to be either linear in the input space or at least "linear" in the kernel feature space. This requirement is the basis for defining the parameter subspace $\mathcal{M}_P$ in eq. (6.4), which is then used to formulate eq. (6.6) and eq. (6.8). However, when we proceed to eq. (6.7) and eq. (6.10), the parts that involve the reduction operator P are just $P\mathbf{x}_i$ in (6.7) and $P\Phi(\mathbf{x}_i)$ in (6.10). Therefore, a simple trick is to replace these two terms by an arbitrary reduction function $\Psi(\mathbf{x}_i)$. Note that eq. (6.7) and (6.10) with this trick are no longer connected to eq. (6.6) and (6.8), but they can still be solved as discussed in Section 6.2.2 and 6.2.3. In prediction time, we can handle any new example $\mathbf{x}$ as long as we can calculate the reduced feature representation $\Psi(\mathbf{x})$ using the reduction function Ψ.

This trick may be very useful for some application domains. Consider a fully probabilistic topic model such as latent Dirichlet allocation [BNJ03]. The reduction operation is neither linear in the input space nor linear in any kernel space. In fact, extracting a topic distribution (e.g., performing the dimension reduction) for a given document via latent Dirichlet allocation involves intractable inference of posterior distribution. In this sense, the trick discussed above allows us to encode information from a probabilistic topic model when learning predictive models on text.

For learning a prediction model in a kernel space, this trick also enables us to use a different kernel or the same kernel with different kernel parameters for the dimension reduction operation. This is not allowed in Section 6.2.3, since the kernel-based reduction P in eq. (6.8) was assumed to operate on exactly the same kernel feature space as the prediction model $\mathbf{w}$.

Table 6.1: R^2 for predicting Boston housing prices: means and standard errors over 500 random runs.

	R^2: Mean	R^2: Standard Error
Ridge	49.53%	0.98%
PCR($d = 11$)	52.93%	0.96%
PLS($d = 11$)	52.58%	0.97%
Proj-PCR($d = 4$)	53.55%	0.68%
Proj-PLS($d = 1$)	53.63%	0.72%

6.3 Empirical Studies

In this section, we present our empirical studies on applying projection penalties to different dimension reduction techniques for housing price prediction, text classification, and face recognition.

6.3.1 Linear Reduction in Regression

Experiment Settings. We use the Boston Housing data set from the UCI Machine Learning Repository[1] as an example regression task. The data set contains 506 examples, each corresponding to a U.S. census tract in the Boston area. The target variable is the median value of owner-occupied homes and the 13 input variables include crime rate, student-teacher ratio, tax rate, and so forth. We conducted 500 random runs in our experiments. In each run, we randomly sample 50 training examples and the remaining 456 records are used as testing examples. We use R^2, one minus the proportion of unexplained variance, as the performance measure for this regression task.

Competing Methods. We study the following methods. 1) Ridge regression (*Ridge*) trained on the original 13 variables. 2) Principal component regression (*PCR*), which first performs dimension reduction using PCA and then fits a least squares model on the principal components. 3) Partial least squares (*PLS*), which can be viewed as sequentially extracting a number of components using both input variables and the response variable and fitting a least squares model on the extracted components. 4) Projection penalty (*Proj-PCR*) in eq. (6.6) with PCA as the dimension reduction P and least squares as the loss function L. 5) Projection penalty (*Proj-PLS*) in eq. (6.6) using a trained partial least squares model to provide dimension reduction and least squares as the loss function. For ridge regression and the projection penalty models, the regularization parameter λ is chosen by 5-fold cross-validation from the range $[10^{-8}, 10^{-6}, \ldots, 10^{10}]$. Also, for the projection penalty models, $\frac{\lambda}{1000}||\beta||_2^2$ is added to penalize the model in the reduced space, as discussed in Section 6.2.4. The dimension of the reduced space d is chosen for each model to optimize the performance.

Results. The experimental results are shown in Table 6.1. For each model, we report the average R^2 over 500 random runs as well as the standard errors. The optimal dimension of the reduced space is also displayed with each model (if applicable). From the results we can see

[1] http://www.ics.uci.edu/∼mlearn/MLRepository.html

Table 6.2: Text classification error averaged over tasks (2% train data): mean and standard error over 20 random runs

	Mean	Standard Error
L2-LGR	17.78%	0.082%
LGR-Topic($d = 30$)	12.51%	0.059%
Proj-Topic($d = 30$)	10.36%	0.067%

Table 6.3: Text classification error averaged over tasks (5% train data): mean and standard error over 20 random runs

	Mean	Standard Error
L2-LGR	11.08%	0.039%
LGR-Topic($d = 30$)	10.45%	0.049%
Proj-Topic($d = 30$)	7.87%	0.042%

Table 6.4: Text classification error averaged over tasks (10% train data): mean and standard error over 20 random runs

	Mean	Standard Error
L2-LGR	8.30%	0.029%
LGR-Topic($d = 30$)	9.39%	0.027%
Proj-Topic($d = 30$)	6.77%	0.029%

that dimension reduction techniques are helpful for predicting the house prices. Both principal component regression (PCR) and partial least squares (PLS) perform better than ridge regression. Projection penalty models (Proj-PCR and Proj-PLS) further improve the predictions.

6.3.2 Topic Modeling in Text Classification

Experiment Settings. We use the 20-Newsgroups data set[2], which contains 11314 training and 7532 testing documents collected from 20 newsgroups. We denote training and testing sets as D_{tr} and D_{ts}, respectively. Documents are represented as bags of words. We select the most frequent 200 words in each newsgroup except the 20 most frequent common words across all newsgroups. This leads to $p = 1443$ words (i.e., features) in the vocabulary. Several latent Dirichlet allocation [BNJ03] models are estimated from D_{tr} and used as dimension reduction models, with the number of topics d specified as $10, 20, 30, 50$, and 100. We construct 190 binary classification tasks using all pairs of newsgroups. For each task, we randomly sample 2%, 5% and 10% of the relevant documents in D_{tr} as training examples for classifiers. We use the average classification error over 190 tasks as the performance measure. This procedure is repeated for 20 random runs. The testing documents for each task are fixed as all relevant documents in D_{ts}.

[2]http://people.csail.mit.edu/jrennie/20newsgroups

Competing Methods. We consider three methods. 1) ℓ_2-regularized logistic regression (L2-LGR) in the original feature space: we directly train a logistic regression classifier for each task, and use ℓ_2 regularization to prevent overfitting in the high-dimensional feature (word) space. 2) Logistic regression in the topic space (LGR-Topic): we use latent Dirichlet allocation to reduce the feature space (into topic space), and then train a logistic regression for each task in the topic space. 3) Projection penalty (Proj-Topic) as in eq. (6.7) with latent Dirichlet allocation as the dimension reduction operator. Note that the trick discussed in Section 6.2.5 is needed since extracting the topic distribution is not a linear reduction of the word space. The parameter λ in the ℓ_2 regularization is chosen via 5-fold stratified cross-validation from the range $[10^{-8}, 10^{-6}, \ldots, 10^{10}]$. For methods using dimension reduction, $d = 30$ topics gives the best performance.

Results. Empirical results on average classification errors over tasks are shown in Tables 6.2, 6.3 and 6.4, with 2%, 5% and 10% training examples used. Means and standard errors over 20 random runs are reported. Using a projection penalty with a topic model (*Proj-Topic*) achieves the best performance in all sizes of training sets. The performance is better than both directly fitting regularized models in the word space (*L2-LGR*) and fitting models in the reduced topic space (*LGR-Topic*). It is interesting to compare the performance of *L2-LGR* and *LGR-Topic* when the training size varies. Using 2% of the training examples, learning classifiers on topics performs significantly better than fitting models in the original word space, as the topic space highlights important information in a lower dimensionality. However, when the training size is increased to 10%, directly fitting a model on the word distribution is superior to learning models in the reduced topic space. This confirms that the dimension reduction also loses valuable information. Finally, the projection penalty with a topic model dominates the other two methods for all training sizes, which indicates the projection penalty effectively encodes the information from the topic model but also controls the loss of information from the reduction.

6.3.3 Kernel Methods in Face Recognition

Experiment Settings. We use the Yale face data set[3], which contains face images of 15 subjects. There are 11 images per subject, corresponding to different configurations in terms of expression, emotion, illumination, and wearing glasses (or not), etc. Each image is scaled to 32×32 pixels. We vary the size of the training set as $3, 5$ and 7 images per subject. We have 50 random runs for each size of training sets. In each run, we randomly select training examples and the rest is used as the testing set. Training examples of all subjects are used to learn dimension reduction models (discussed later). We consider all 105 binary classification tasks, each classifying between two subjects. The average classification error over tasks is the performance measure, and aggregated results over 50 runs are reported.

In this experiment, we study dimension reduction and predictive modeling in a kernel feature space. We use the degree-2 polynomial kernel to define our kernel feature space. We consider three dimension reduction methods: kernel PCA [SSM98], generalized discriminant analysis (GDA) [BA00] and orthogonal Laplacian faces (OLap) [CHHZ06].

[3]http://cvc.yale.edu/projects/yalefaces/yalefaces.html

Table 6.5: Face classification error averaged over tasks (3 training images per subject): mean and standard error over 50 random runs

	Mean	Standard Error
SVM-Kernel	16.17%	0.26%
SVM-KPCA($d = 45$)	16.22%	0.25%
Proj-KPCA($d = 45$)	15.33%	0.26%
SVM-GDA($d = 14$)	11.21%	0.25%
Proj-GDA($d = 14$)	10.38%	0.24%
SVM-OLap($d = 14$)	8.58%	0.26%
Proj-OLap($d = 14$)	7.93%	0.22%

Table 6.6: Face classification error averaged over tasks (5 training images per subject): mean and standard error over 50 random runs

	Mean	Standard Error
SVM-Kernel	12.68%	0.30%
SVM-KPCA($d = 75$)	12.72%	0.30%
Proj-KPCA($d = 75$)	12.47%	0.29%
SVM-GDA($d = 14$)	9.32%	0.23%
Proj-GDA($d = 14$)	7.13%	0.20%
SVM-OLap($d = 14$)	4.74%	0.21%
Proj-OLap($d = 14$)	4.57%	0.20%

Table 6.7: Face classification error averaged over tasks (7 training images per subject): mean and standard error over 50 random runs

	Mean	Standard Error
SVM-Kernel	11.43%	0.28%
SVM-KPCA($d = 105$)	11.37%	0.29%
Proj-KPCA($d = 105$)	11.19%	0.28%
SVM-GDA($d = 14$)	7.59%	0.23%
Proj-GDA($d = 14$)	6.37%	0.23%
SVM-OLap($d = 14$)	3.43%	0.17%
Proj-OLap($d = 14$)	3.40%	0.17%

Competing Methods. We compare the following methods. 1) SVM in the kernel feature space (*SVM-Kernel*): train an SVM classifier for each task with a polynomial kernel. This can be viewed as fitting a linear model directly in the high-dimensional kernel feature space. 2) SVM in the reduced feature space (*SVM-KPCA, SVM-GDA, SVM-OLap*): perform KPCA, GDA or OLap, and then fit a linear SVM in the reduced space. KPCA and GDA are used with polynomial kernels. Thus, SVM-KPCA and SVM-GDA can be viewed as a reduction of the high-dimensional ker-

nel feature space followed by a linear model fit in the reduced space. 3) Projection penalty as in eq. (6.17) with KPCA, GDA and OLap (*Proj-KPCA, Proj-GDA, Proj-OLap*). Note that OLap is not a kernel-based reduction technique, so the trick in Section 6.2.5 is used. The regularization parameter C for SVMs is chosen by stratified cross-validation from the range $[10^{-8}, 10^{-6}, \ldots, 10^{10}]$. For the projection penalty in eq. (6.17), we set $\gamma = 10^{-5}$.

Results. Experimental results are shown in Table 6.5, Table 6.6 and Table 6.7. These three tables present results for 3, 5 and 7 training images per subject, respectively. The optimal dimension d of the reduced feature space is reported with each method. In each table, *SVM-Kernel* corresponds to learning a linear SVM in the kernel feature space and serves as the baseline. We observe the following results. 1) *SVM-KPCA* vs *Proj-KPCA*: KPCA is not a very effective dimension reduction tool for classification, mainly because it does not use the information from labels. *SVM-KPCA* performs similarly or even worse than the baseline model. *Proj-KPCA* performs better than *SVM-KPCA* and the baseline *SVM-Kernel*, showing that the projection penalty is safe and reliable when used with an ineffective reduction. 2) *SVM-GDA* vs *Proj-GDA*: GDA is significantly more effective than KPCA for dimension reduction. *SVM-GDA*'s performance is superior to the baseline, and *Proj-GDA* further improves the performance. 3) *SVM-OLap* vs *Proj-OLap*: orthogonal Laplacian faces is a powerful reduction technique for supervised learning. All OLap-based methods predict better than KPCA- and GDA-based methods. Performance of our *Proj-OLap* is slightly better than *SVM-OLap*: OLap provides very high-quality dimension reduction for classification, and thus there may not be a significant loss of predictive information. As a result, our proposed *Proj-OLap* only slightly outperforms *SVM-OLap* (i.e., directly performing the reduction).

6.4 Conclusion

In this chapter, we investigate a common scenario in which the structure information about the input space is highlighted by a dimension reduction. We propose the projection penalty framework to encode a dimension reduction while controlling the risk of information loss during the reduction: reducing the input space and learning in the reduced space can be viewed as restricting the model search to some parameter subspace, and instead of restricting the search to a parameter subspace, we can still search in the full space but penalize the projection distance to the parameter subspace. We design projection penalties for linear dimension reduction, then generalize to kernel-based reduction and other nonlinear reduction methods. We empirically study projection penalties with various dimension reduction techniques in regression, text classification, and face recognition. Experimental results show that the projection penalty framework is an effective and reliable way to gain from dimension reduction techniques without losing important predictive information.

Part II

Encoding Output Information

Chapter 7

A Composite Likelihood Method for Multi-Label Classification

7.1 Overview

In this chapter [ZS12a], we provide a composite likelihood view for many recent multi-label classification algorithms, and then we show a simple modification of the existing pairwise decomposition methods using the composite likelihood view. The composite likelihood view also provides the framework for the more advanced output coding schemes presented in Chapter 8 and 9.

Multi-label classification has recently received considerable attention from the machine learning community [TKV10]. As reviewed in Section 2.3, in multi-label classification we aim to simultaneously classify a set of multiple labels for each example, and a key assumption is the existence of certain label dependency structures, i.e., label assignments are usually not independent. A very popular class of multi-label classification algorithms are based on problem decomposition, in which the multi-label classification problem is decomposed into a set of subproblems, prediction models are learned for the subproblems, and then the final classification is made by combining subproblem predictions. Subproblems are usually chosen to be simple decision problems that can capture certain dependencies among labels. Popular choices of subproblems include one-vs-all decomposition (i.e., the relevance of each individual label) [RK04], one-vs-one decomposition (i.e., the pairwise comparison between any two labels) [Fur02, HFCB08], a combination of both one-vs-all and one-vs-one decompositions [FHMB08], and conditional relevance of one label given other labels (e.g., classifier chain methods) [RPHF09, DCH10].

Despite having some empirical success, the choice of subproblems in many decomposition methods seems arbitrary, and the combination of subproblem predictions in the final decision making is usually based on heuristics (e.g., voting). To address these issues, we first show the connection between multi-label decomposition algorithms and composite likelihood [VRF11, LYS11], a technique based on partial specification of the likelihood as the product of simple component likelihoods to efficiently model complex dependencies. We believe that this connection holds great promise for improving the design of multi-label decomposition methods, especially in the choice

of subproblems and the combination of subproblem predictions in decision making.

As an attempt to exploit this connection, we design a composite marginal method that improves the popular pairwise decomposition approaches. In existing pairwise decomposition methods, pairwise label comparisons are widely used as the subproblems to learn, and prediction for a testing sample is made by voting from all label comparisons. Inspired by the composite likelihood view, we instead choose pairwise label density estimations as the subproblems to learn, and for combining subproblem decisions (i.e., decoding), we design a mean-field approximation procedure based on the composite likelihood view, which minimizes the notion of composite divergence and is potentially more robust to inaccurate estimation in subproblems.

On five real-world data sets, we compare our method with several multi-label decomposition methods and a joint learning approach that captures label dependency using a graphical model. Empirical results show that, given limited training samples, the proposed method outperforms alternatives and provides superior predictions under a variety of evaluation criteria.

7.2 Composite Likelihood Methods

In this section we first give an overview of commonly used forms of composite likelihoods, and then we briefly motivate and discuss parameter estimation via composite likelihoods.

7.2.1 Overview

Composite likelihood is a partial specification of the full likelihood function by multiplying a set of simple component likelihoods, where each component likelihood is in the form of either a marginal or a conditional density. Composite likelihood can be viewed as an oversimplified form of the full likelihood, but such an approximation can provide certain benefits in parameter estimation, notably in computation, statistical efficiency (with limited samples), and robustness (to model misspecification). Research on composite likelihood methods can be dated back to Besag's pseudolikelihood [Bes74] and Cox's partial likelihood [CR75]. Lindsay [Lin88] formalizes the term composite likelihood for "product of likelihoods". Some excellent overviews and discussion on this subject have been recently published [VRF11, LYS11].

Formally, consider q random variables $\mathbf{Y} = (Y_1, \ldots, Y_q)$, the parameter vector $\theta \in \Theta$ and the full likelihood function $L(\theta; \mathbf{y})$ with one observation $\mathbf{y}$. Following the notation in [Lin88, VRF11], we denote by $\{\mathcal{A}_1, \ldots, \mathcal{A}_K\}$ a collection of K marginal or conditional events, with corresponding component likelihoods $\{L_k(\theta; \mathbf{y}) \propto f(\mathbf{y} \in \mathcal{A}_k; \theta)\}_{k=1}^K$. A composite likelihood approximates the likelihood function $L(\theta; \mathbf{y})$ by:

$$L_C(\theta; \mathbf{y}) = \prod_{k=1}^{K} L_k(\theta; \mathbf{y})^{w_k} \tag{7.1}$$

where $\{L_k(\theta; \mathbf{y})\}_{k=1}^K$ are marginal or conditional densities, and $\{w_k\}_{k=1}^K$ are (optional) nonnegative weights on components. In this chapter we mainly consider $w_k = 1, k = 1, 2, \ldots, K$ for simplicity.

Composite marginal likelihoods make up a large class of composite likelihoods that use low-dimensional marginal densities as component likelihoods. The simplest example of composite marginal likelihood is the so-called *independence likelihood*. For q variables $\mathbf{y} = (y_1, \ldots, y_q)$, we have:

$$L_{ind}(\theta; \mathbf{y}) = \prod_{i=1}^{q} f(y_i; \theta) \tag{7.2}$$

Naturally this specification does not capture interactions between different variables. Another well studied class of likelihoods, which considers small blocks of variables, is the *pairwise likelihood* [Kuk00, Cox04, VRF11, LYS11]:

$$L_{pair}(\theta; \mathbf{y}) = \prod_{i=1}^{q-1} \prod_{j=i+1}^{q} f(y_i, y_j; \theta) \tag{7.3}$$

which includes interactions between pairs of variables into the likelihood function.

Composite conditional likelihoods is another popular class of composite likelihoods based on conditional densities, which date back to pseudolikelihood [Bes74]. Its general form is the following:

$$L_{cond}(\theta; \mathbf{y}) = \prod_{k=1}^{K} f(y_{B_k} | y_{\mathcal{N}(B_k)}; \theta) \tag{7.4}$$

where each B_k indexes a block of variables, and $\mathcal{N}(B_k)$ indexes neighbors of the variables in B_k. Two widely used composite conditional likelihoods in both longitudinal studies [MV05] and bioinformatics [MHTS08] are the *pairwise conditional likelihood*:

$$L_{pcl}(\theta; \mathbf{y}) = \prod_{i=1}^{q} \prod_{j=1}^{q} f(y_i | y_j; \theta) \tag{7.5}$$

and the *full conditional likelihood*:

$$L_{fcl}(\theta; \mathbf{y}) = \prod_{i=1}^{q} f(y_i | y_{(-i)}; \theta) \tag{7.6}$$

where $y_{(-i)}$ denotes all variables but y_i.

7.2.2 Computation, robustness, and statistical efficiency of estimation

Composite likelihood methods provide an approximation to the full likelihood function that has been widely used in parameter estimation. Research on composite likelihood estimation has been focused on computational efficiency, robustness to model misspecification, and statistical efficiency. Parameter estimation via composite likelihoods is computationally more efficient than optimizing the joint likelihood function, especially when parameters in component likelihoods are decoupled and estimated separately, e.g., as in [FV06, MV05]. Also, it has been argued by

many researchers that composite likelihood estimation is more robust to model misspecification since only model assumptions on low-dimensional conditional or marginal densities, instead of the detailed specification of the full joint density, are required [VRF11, LYS11]. The statistical efficiency of the estimation has also been intensively studied [MKHT09, HV08, VRF11, LYS11], e.g., in [MKHT09], composite conditional estimators are shown to be fully efficient (and identical to maximum likelihood estimators in terms of efficiency) in the multivariate normal distribution.

7.3 A Composite Likelihood View of Multi-Label Problem Decomposition

In this section, we show the connection between composite likelihoods and many recent multi-label decomposition methods.

7.3.1 The Full Multi-Label Model

One way to capture the full dependency of q labels is to treat each of the 2^q label combinations as a separate class. This is usually called the *label powerset method* (LP) [TKV10]. In the statistics literature, this method is linked to directly modeling the joint probability of all 2^q entries in a q-dimensional contingency table [Cox72]. The drawback of this approach is obvious. Large numbers of observations are needed to estimate this full model well, and the computational complexity of the training algorithm usually scales exponentially in q. This is what motivates composite likelihoods: the full likelihood is too difficult to estimate and perform inference on.

7.3.2 Multi-Label Problem Decomposition and Composite Likelihoods

Multi-label decomposition methods transform the original problem into a set of subproblems, learn each subproblem, and combine subproblem predictions to make the final classification. In general, the choice of subproblems in a decomposition method corresponds to a certain instantiation of the composite likelihood:

$$L_C(\theta; \mathbf{y}|\mathbf{x}) = \prod_{k=1}^{K} L_k(\theta; \mathbf{y}|\mathbf{x}) \tag{7.7}$$

where $\mathbf{y} = (y_1, \ldots, y_q) \in \{0, 1\}^q$ denotes the label vector and $\mathbf{x}$ denotes the feature vector. Learning the subproblems corresponds to assuming the parameters inside different $L_k(\theta; \mathbf{y}|\mathbf{x})$ are independent and estimating them separately. Combining subproblem predictions is usually based on heuristics or exhaustive search, without leveraging properties of the composite likelihood.

One-vs-all decomposition. The simplest way to decompose multi-label classification is *one-vs-all decomposition*, or the *binary relevance method* (BR), which has been empirically justified in the context of multi-class classification [RK04]. For q labels, we consider q subproblems, each to

model the relevance of one label independently. This method corresponds to a composite likelihood of the form:

$$L_{BR}(\theta; \mathbf{y}|\mathbf{x}) = \prod_{i=1}^{q} f(y_i|\mathbf{x}; \theta_i) \tag{7.8}$$

where θ_i denotes the parameters used to model label i. For prediction, each subproblem provides the conditional probability of one label, and prediction can be done by thresholding (e.g., at 0.5) and there is no need to combine subproblem decisions. Since this method ignores the dependency among labels, suboptimal performance will be obtained if the evaluation criterion calls for capturing the label dependency [DCH10]. For example, this method might predict each label reasonably well, but rarely get all the labels classified correctly.

Pairwise label ranking. Another popular strategy is *one-vs-one decomposition* [Fur02], which is used in *pairwise label ranking* (PLR) [HFCB08]. For q labels, this method captures label dependency by formulating $\frac{q(q-1)}{2}$ subproblems, each learning a classifier to compare two labels. In this sense, the choice of subproblems in pairwise label ranking indicates the following composite likelihood:

$$L_{PLR}(\theta; \mathbf{y}|\mathbf{x}) = \prod_{i=1}^{q-1} \prod_{j=i+1}^{q} f(y_i \geq y_j|\mathbf{x}; \theta_{ij}) \tag{7.9}$$

where θ_{ij} is the vector of parameters for label pair (i, j), and the component form $f(y_i \geq y_j|\mathbf{x}; \theta_i)$ comes from the fact that in one-vs-one decomposition, subproblems are *pairwise comparisons*. For prediction, votes are collected from all pairwise classifiers, and labels are ranked by the number of winning votes they receive, e.g., the label receiving $q - 1$ winning votes, if there is one, will be ranked first as it wins all the comparisons against other $q - 1$ labels. Note that this method is only a label ranking method: a rank order does not give a classification, and one has to find a threshold for classification.

Calibrated Label Ranking. *Calibrated label ranking* (CLR) is a strategy that combines both one-vs-one and one-vs-all decompositions for multi-label classification [FHMB08]. The key idea is to introduce a virtual label that is always ranked between the relevant set and the irrelevant set of labels. As a result, all we need for classifying q labels is to obtain a rank order of $q + 1$ labels (q labels plus one virtual label) by pairwise label ranking. The difficulty, however, is how to learn the pairwise classifier between each actual label and the virtual label. In [FHMB08], this is solved by noticing that the pairwise comparison between label i and the virtual label is semantically equivalent to the one-vs-all decision for label i. As CLR uses both one-vs-one and one-vs-all subproblems, it corresponds to the composite likelihood:

$$L_{CLR}(\theta; \mathbf{y}|\mathbf{x}) = [\prod_{i=1}^{q} f(y_i|\mathbf{x}; \theta_i)] \cdot [\prod_{i=1}^{q-1} \prod_{j=i+1}^{q} f(y_i \geq y_j|\mathbf{x}; \theta_{ij})] \tag{7.10}$$

Classifier chains. Classifier chains (CC) [RPHF09] and probabilistic classifier chains (PCC) [DCH10] use subproblems that describe the conditional relation of labels. Both methods start with a randomly chosen label order. CC learns a function to predict each label, conditional on the input features and antecedent labels along the chosen order. In PCC, each prediction function is probabilistic, and all functions together define the joint label probability as in a Bayesian network. In this sense, given a label order $((1), (2), \ldots, (q))$, CC and PCC are linked to:

$$L_{CC}(\theta; \mathbf{y}|\mathbf{x}) = \prod_{i=1}^{q} f(y_{(i)}|\mathbf{x}, y_{(1)}, \ldots, y_{(i-1)}; \theta_{(i)}) \tag{7.11}$$

For classification using CC, labels are predicted in the chosen order, based on the input features and predicted antecedent labels. For classification using PCC, however, efficient inference is not provided and exhaustive search is used to find label assignments [DCH10]. Indeed, if we consider (7.11) as a Bayesian network, the largest conditional table contains all the labels.

Labelset methods. The label powerset method considers all the 2^q label combinations as 2^q classes. This corresponds to the full likelihood approach with 2^q configurations. Along this direction, the pruned labelset method considers only label combinations that are sufficiently frequent in the data [RPH08], and the random k-labelsets focuses on label combinations that only involve k labels [TV07]. These methods generate multi-class classification problems with less than 2^q classes, and it corresponds to specifying a single component likelihood, but with less entries than in the full likelihood.

7.3.3 Other Multi-Label Classification Methods

All the decomposition methods discussed in Section 7.3.2 can be enhanced using randomization and ensemble learning. A committee of experts can be constructed by launching a given method multiple times with certain randomization, such as sampling training examples, label orders or label subsets. Ensemble-based extensions that have been specifically studied include ensemble classifier chains [RPHF09], ensemble probabilistic classifier chains [DCH10], ensemble pruned labelsets [RPH08] and random K-labelsets [TV07]. It is generally recognized that the ensemble version of an algorithm tends to outperform the original non-ensemble algorithm, although this benefit usually comes at the cost of much more intensive computation.

Instead of decomposing and learning subproblems individually, another direction is to jointly model the dependency among all the labels and features via a single graphical model. Research along this direction includes specifying a conditionally trained undirected graphical model (i.e., CRF) [GM05] or learning the structure of Bayesian networks from data [ZZ10, ZSM+11] to capture the joint label relation (conditioned on features). Joint modeling via a single graphical model provides an elegant method of multi-label learning, but potential challenges include: 1) learning the graphical model (such as training an undirected graphical model or learning the structure of a Bayesian network) is usually intractable, unless certain approximation is made (e.g., greedy search

for learning the structure of a Bayesian network); 2) joint modeling is likely to demand more training samples than learning simpler subproblems.

As an alternative to model-independent problem decomposition methods, researchers have also concentrated on adapting specific models (such as decision trees, SVMs, KNNs, neural networks, boosting) to directly produce multi-label predictions.

An recent overview on a variety of multi-label learning methods can be found in [TKV10].

7.4 A Composite Marginal Model for Multi-Label Classification

Exploiting the composite likelihood view of problem decomposition, we propose a composite marginal method for multi-label classification. We consider the subproblems of estimating the univariate and bivariate label densities as components in a composite likelihood, and based on this composite likelihood, we develop a new mean-field approximation procedure that minimizes the notion of composite divergence to perform efficient inference for combining subproblem decisions. This procedure is potentially more robust to certain common estimation errors in subproblems.

7.4.1 Implications of Composite Likelihood to Multi-Label Problem Decomposition

Subproblem design. A primary implication of the composite likelihood perspective to multi-label classification is on the design of subproblems. For example, consider the composite marginal likelihood of pairwise label ranking in (7.9) and that of calibrated label ranking in (7.10). Both composite likelihoods contain *pairwise comparison* events $\{f(y_i \geq y_j|\mathbf{x};\theta_{ij})\}$ as component likelihoods. However, is the component likelihood in the form of $f(y_i \geq y_j|\mathbf{x};\theta_{ij})$ the most informative description of the pairwise label relationship? This issue will be addressed in Section 7.4.2.

Combination of subproblem decisions. Combining subproblem predictions to make a final classification is critical to multi-label decomposition methods. However, most existing decomposition methods combine subproblem decisions using heuristics such as voting. The composite likelihood function approximates the joint label probability in a composite form of subproblem predictions, and this form enables the design of efficient inference procedures that are particularly suitable for multi-label classification. In Section 7.4.3, we develop a new robust mean-field approximation by exploiting the composite form of the likelihood.

7.4.2 Composite Marginal Modeling: Specification and Estimation

As raised in Section 7.4.1, one issue in (7.10) is the use of likelihood components of the form $\{f(y_i \geq y_j|\mathbf{x};\theta_{ij})\}$. Although pairwise comparison between two labels is a very natural subproblem to consider from the algorithm perspective, it is not the most informative component in

a likelihood function. Knowing "$y_i \geq y_j$" only rules out one scenario $(y_i = 0, y_j = 1)$ and still leaves three possibilities: $(y_i = 0, y_j = 0)$, $(y_i = 1, y_j = 0)$, or $(y_i = 1, y_j = 1)$. In this sense, the bivariate marginal density fully describes the pairwise relation of two labels and thus provides more information to the composite likelihood. Also, bivariate densities of the form $f(y_i, y_j|\mathbf{x}; \theta_{ij})$ are more natural than $\{f(y_i \geq y_j|\mathbf{x}; \theta_{ij})\}$ when considered as part of a likelihood function. Therefore, we consider univariate and bivariate density estimation as subproblems, which give the following composite likelihood:

$$L_{CMM}(\theta; \mathbf{y}|\mathbf{x}) = [\prod_{i=1}^{q} f(y_i|\mathbf{x}; \theta_i)] \cdot [\prod_{i=1}^{q-1} \prod_{j=i+1}^{q} f(y_i, y_j|\mathbf{x}; \theta_{ij})]^{\lambda} \tag{7.12}$$

where $\{f(y_i|\mathbf{x}; \theta_i)\}$ and $\{f(y_i, y_j|\mathbf{x}; \theta_{ij})\}$ are univariate and bivariate marginal densities of labels conditional on the feature vector $\mathbf{x}$, $\{\theta_i\}$ and $\{\theta_{ij}\}$ are two sets of parameter vectors for univariate and bivariate densities, and λ is the nonnegative weight of bivariate densities, which we set to 1.

Given a set of N training examples $\mathbf{D} = \{\mathbf{y}^n, \mathbf{x}^n\}_{n=1}^N$, the overall composite likelihood function is:

$$L_{CMM}(\theta; \mathbf{D}) = \prod_{n=1}^{N} L_{CMM}(\theta; \mathbf{y}^n|\mathbf{x}^n) \tag{7.13}$$

Since we assume parameters in different components are independent, maximum composite likelihood estimation (MCLE) can be performed separately:

$$\hat{\theta}_i = \underset{\theta_i}{\operatorname{argmax}} \prod_{n=1}^{N} f(y_i^n|\mathbf{x}^n; \theta_i), \quad i = 1, 2, \ldots, q \tag{7.14}$$

$$\hat{\theta}_{ij} = \underset{\theta_{ij}}{\operatorname{argmax}} \prod_{n=1}^{N} f(y_i^n, y_j^n|\mathbf{x}^n; \theta_{ij}), \quad i = 1, 2, \ldots, q-1 \text{ and } j = i+1, \ldots, q \tag{7.15}$$

Note that problem (7.14) is essentially to estimate a probabilistic binary classifier for each label i, and problem (7.15) is essentially to estimate a probabilistic 4-class classifier (for the four possible configurations 00, 01, 10 and 11 of two labels i and j).

7.4.3 Composite Marginal Modeling: Inference via Robust Mean-Field Approximation

Given $(\{\hat{\theta}_i\}, \{\hat{\theta}_{ij}\})$, the composite likelihood provides a joint conditional probability of labels:

$$\hat{P}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} \cdot [\prod_{i=1}^{q} f(y_i|\mathbf{x}; \hat{\theta}_i)] \cdot [\prod_{i=1}^{q-1} \prod_{j=i+1}^{q} f(y_i, y_j|\mathbf{x}; \hat{\theta}_{ij})]^{\lambda} \tag{7.16}$$

where $f(y_i|\mathbf{x}; \hat{\theta}_i)$ and $f(y_i, y_j|\mathbf{x}; \hat{\theta}_{ij})$ are discrete potentials, and Z is the partition function.

For a testing example $\mathbf{x}$, exact inference on $\mathbf{y} \in \{0,1\}^q$ using $\hat{P}(\mathbf{y}|\mathbf{x})$ has a time complexity exponential in q. For efficient classification, we may consider the classical mean-field approximation [JGJS99, KF09]:

$$Q(\mathbf{y}) = \prod_{i=1}^{q} Q_i(y_i) \tag{7.17}$$

which is the fully factorized distribution on $\mathbf{y}$, with each $Q_i(y_i)$ a Bernoulli distribution on label y_i. Traditionally, we would use fixed point equations to minimize the following KL divergence [KF09]:

$$\hat{Q} = \operatorname*{argmin}_{Q} KL(Q||\hat{P}) = \operatorname*{argmin}_{Q} \sum_{\mathbf{y}\in\{0,1\}^q} Q(\mathbf{y}) \log \frac{Q(\mathbf{y})}{\hat{P}(\mathbf{y}|\mathbf{x})} \tag{7.18}$$

However, $KL(Q||\hat{P})$ is **highly sensitive** to a specific type of estimation error that happens **frequently** in multi-label classification: underestimating the probability of rare label combinations. When a label combination $\mathbf{y}$ is rare, maximum likelihood estimation as in (7.14) and (7.15) will produce model parameters that further underestimate $\hat{P}(\mathbf{y}|\mathbf{x})$, thus pushing $\hat{P}(\mathbf{y}|\mathbf{x})$ to zero. This is known as the class imbalance problem [CJK04]. Unfortunately, since Q is the base distribution, $KL(Q||\hat{P})$ is sensitive to regions where Q is non-zero but $\hat{P}$ is close to zero. This is also evident in (7.18) as $\hat{P}$ appears in the denominator. Thus, minimizing $KL(Q||\hat{P})$ reacts dramatically to underestimation errors in $\hat{P}$, i.e., Q is forced to be zero wherever $\hat{P}$ approaches zero.

As a result, we need a new divergence measure $D(Q||\hat{P})$ which, as $KL(Q||\hat{P})$, can be optimized efficiently by fixed point equations (to reach a stationary point), and more importantly, $D(Q||\hat{P})$ needs to be robust to underestimation errors in $\hat{P}$. A convenient way to define a divergence measure for a composite likelihood is to use *composite divergence* [VRF11], which is the linear weighted combination of divergences to all the component distributions. Since $\hat{P}$ is the normalized L_{CMM} in (7.12), we propose the following quadratic divergence:

$$D(Q||\hat{P}) = \sum_{i=1}^{q} D(Q||f(\cdot|\mathbf{x};\hat{\theta}_i)) + \lambda \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} D(Q||f(\cdot,\cdot|\mathbf{x};\hat{\theta}_{ij})) \tag{7.19}$$

in which the divergence between Q and each univariate discrete distribution $f(\cdot|\mathbf{x};\hat{\theta}_i)$ is:

$$D(Q||f(\cdot|\mathbf{x};\hat{\theta}_i)) = \sum_{y_i=0}^{1} (Q_i(y_i) - f(y_i|\mathbf{x};\hat{\theta}_i))^2 \tag{7.20}$$

and the divergence between Q and each bivariate discrete distribution $f(\cdot,\cdot|\mathbf{x};\hat{\theta}_{ij})$ is:

$$D(Q||f(\cdot,\cdot|\mathbf{x};\hat{\theta}_{ij})) = \sum_{y_i=0}^{1} \sum_{y_j=0}^{1} (Q_i(y_i)Q_j(y_j) - f(y_i,y_j|\mathbf{x};\hat{\theta}_{ij}))^2 \tag{7.21}$$

Table 7.1: Results on Medical data: means (and standard errors) over 30 random runs. The best model is marked with *, and all competitive models (by paired t-tests with the best model, $\alpha = 0.05$) are shown in **bold**.

Methods	Hamming Loss	Subset 0/1 Loss	Ranking Loss	One Error
BR	0.0325 (0.0007)	0.2608 (0.0048)	0.3627 (0.0075)	0.0787 (0.0033)
PLR	0.3721 (0.0011)	1.0000 (0.0000)	0.3678 (0.0050)	0.0978 (0.0029)
CLR	0.0338 (0.0006)	0.2708 (0.0043)	0.3687 (0.0054)	0.0888 (0.0025)
CC	0.0358 (0.0007)	0.2707 (0.0061)	0.3888 (0.0018)	0.1606 (0.0047)
RK	0.0329 (0.0007)	0.2610 (0.0055)	0.3863 (0.0016)	0.1109 (0.0030)
LEAD	0.0331 (0.0006)	**0.2563** (0.0046)	**0.3400** (0.0113)	0.0770 (0.0029)
CMM	**0.0313***(0.0005)	**0.2524***(0.0043)	**0.3276***(0.0082)	**0.0706***(0.0027)

which are the sum of squared differences of the probability mass over possible events in $f(\cdot|\mathbf{x};\hat{\theta}_i)$ and $f(\cdot,\cdot|\mathbf{x};\hat{\theta}_{ij})$, respectively. Note that Q reduces to Q_i and $Q_i \cdot Q_j$ in the two equations as Q is fully factorized. Compared to the KL divergence in (7.18), the new divergences in (7.20) and (7.21) are more robust to the underestimation cases $f(y_i|\mathbf{x};\hat{\theta}_i) \to 0$ and $f(y_i, y_j|\mathbf{x};\hat{\theta}_{ij}) \to 0$, and thus so is the linear combination $D(Q||\hat{P})$ in (7.19).

Minimizing $D(Q||\hat{P})$ in (7.19) w.r.t. a single Bernoulli Q_i with all other $\{Q_j\}_{j\neq i}$ fixed can be solved in closed form. Thus, we can iteratively apply fixed point equations and converge quickly to a stationary point. Since the solution $Q(\mathbf{y}) = \prod_{i=1}^{q} Q_i(y_i)$ is a fully factorized distribution over labels, inference on $Q(\mathbf{y})$ (such as assigning 0/1 value to each label) can be performed efficiently.

7.5 Empirical Studies

Data. We perform our experiments on five real-world multi-label data sets: Enron, Medical, Yeast, Scene, and Emotion[1]. Enron and Medical are text data with labels related to email analysis and medical decision. Yeast is a biological data set where genes are labeled by functions. Scene is an image data set where labels denote different scenes. Emotion is a collections of songs labeled by emotions. We select the top ten labels if the data set has more than ten labels.

Methods. We compare our proposed method to several popular multi-label decomposition methods as well as a joint modeling approach that captures the label dependency using graphical models [ZZ10]. See Section 7.3 for a review of decomposition methods.

- *Binary relevance (BR)*: classify each label independently (i.e., one-vs-all decomposition).
- *Pairwise label ranking (PLR)*: perform pairwise label comparison to rank all the labels.
- *Calibrated label ranking (CLR)*: combine both pairwise comparison and one-vs-all classifiers.
- *Classifier chain (CC)*: a classifier chain to model conditional label relation given a label order.
- *Random k-labelsets (RK)*: an ensemble algorithm combines classifiers for label sets of size K.

[1] http://mulan.sourceforge.net/

Table 7.2: Results on Yeast data: means (and standard errors) over 30 random runs. The best model is marked with *, and all competitive models (by paired t-tests with the best model, $\alpha = 0.05$) are shown in **bold**.

Methods	Hamming Loss	Subset 0/1 Loss	Ranking Loss	One Error
BR	0.3137 (0.0020)	0.9140 (0.0033)	0.4531 (0.0017)	0.4823 (0.0058)
PLR	0.3050 (0.0011)	0.9588 (0.0026)	**0.3833***(0.0009)	**0.2401***(0.0008)
CLR	0.2963 (0.0014)	0.9147 (0.0042)	**0.3839** (0.0008)	**0.2412** (0.0008)
CC	0.3225 (0.0022)	**0.8598** (0.0034)	0.4290 (0.0013)	0.2452 (0.0013)
RK	0.2664 (0.0008)	0.8631 (0.0053)	0.4097 (0.0031)	0.2711 (0.0075)
LEAD	0.2684 (0.0010)	0.8811 (0.0051)	0.3886 (0.0007)	0.2413 (0.0010)
CMM	**0.2651***(0.0009)	**0.8594***(0.0043)	0.3916 (0.0008)	**0.2418** (0.0012)

Table 7.3: Results on Emotion data: means (and standard errors) over 30 random runs. The best model is marked with *, and all competitive models (by paired t-tests with the best model, $\alpha = 0.05$) are shown in **bold**.

Methods	Hamming Loss	Subset 0/1 Loss	Ranking Loss	One Error
BR	0.2354 (0.0027)	0.7863 (0.0045)	0.4937 (0.0030)	0.3394 (0.0074)
PLR	0.2681 (0.0018)	0.8645 (0.0021)	0.4859 (0.0028)	0.2954 (0.0042)
CLR	0.2294 (0.0024)	0.7810 (0.0057)	0.4877 (0.0019)	**0.2914** (0.0049)
CC	0.2431 (0.0033)	0.7818 (0.0067)	0.5134 (0.0026)	0.4285 (0.0069)
RK	**0.2155***(0.0019)	**0.7465***(0.0051)	0.5238 (0.0029)	0.3109 (0.0066)
LEAD	0.2600 (0.0025)	0.8713 (0.0052)	0.4845 (0.0034)	0.3281 (0.0060)
CMM	**0.2163** (0.0018)	**0.7599** (0.0044)	**0.4787***(0.0018)	**0.2884***(0.0044)

- *Multi-label learning by exploiting label dependency (LEAD)*: a joint modeling approach which learns and incorporates a Bayesian network to capture the label dependency.
- *Composite marginal model (CMM)*: the proposed method described in Section 7.4, which exploits the composite likelihood view to improve subproblem choice and prediction combination.

All the methods need base learners to solve the subproblems. We use linear SVMs for binary decisions, multi-class SVMs (based on one-vs-one decomposition) for multi-class cases. If probabilistic estimates are required for binary problems, logistic regression is used instead of SVMs, and for multi-class probabilistic estimates, we use the method in [WLW04]. All these learners are available in the LIBSVM[2] and LIBLINEAR packages[3]. Structure learning of Bayesian networks is performed using the Bayesian Net Toolbox (BNT)[4] and its structure learning extension (BNT-SLP)[5].

All problem decomposition methods can be extended to randomized ensemble versions by

[2]http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[3]http://www.csie.ntu.edu.tw/ cjlin/liblinear/
[4]http://code.google.com/p/bnt/
[5]http://ofrancois.tuxfamily.org/slp.html

Table 7.4: Results on Scene data: means (and standard errors) over 30 random runs. The best model is marked with *, and all competitive models (by paired t-tests with the best model, $\alpha = 0.05$) are shown in **bold**.

Methods	Hamming Loss	Subset 0/1 Loss	Ranking Loss	One Error
BR	0.1439 (0.0015)	0.6118 (0.0039)	0.5072 (0.0058)	0.3669 (0.0040)
PLR	0.3356 (0.0009)	0.9968 (0.0005)	**0.4753** (0.0053)	0.3265 (0.0062)
CLR	0.1397 (0.0012)	0.6181 (0.0046)	**0.4733***(0.0057)	0.3237 (0.0052)
CC	0.1491 (0.0015)	0.5410 (0.0053)	0.5030 (0.0030)	0.4162 (0.0039)
RK	0.1223 (0.0010)	0.5724 (0.0063)	0.5057 (0.0038)	0.3339 (0.0039)
LEAD	0.1291 (0.0011)	0.5739 (0.0048)	**0.4749** (0.0064)	0.3494 (0.0040)
CMM	**0.1138***(0.0008)	**0.5250***(0.0037)	0.4862 (0.0057)	**0.2872***(0.0027)

Table 7.5: Results on Enron data: means (and standard errors) over 30 random runs. The best model is marked with *, and all competitive models (by paired t-tests with the best model, $\alpha = 0.05$) are shown in **bold**.

Methods	Hamming Loss	Subset 0/1 Loss	Ranking Loss	One Error
BR	0.1958 (0.0016)	0.8653 (0.0067)	0.2911 (0.0025)	0.2753 (0.0043)
PLR	0.3121 (0.0016)	0.9921 (0.0005)	0.3058 (0.0039)	**0.2356** (0.0024)
CLR	0.1897 (0.0015)	0.8505 (0.0053)	0.3026 (0.0033)	**0.2341***(0.0030)
CC	0.1960 (0.0019)	0.8417 (0.0046)	0.3730 (0.0038)	0.2622 (0.0042)
RK	**0.1811** (0.0007)	**0.8269***(0.0022)	0.3554 (0.0031)	0.2945 (0.0076)
LEAD	0.1932 (0.0007)	0.8614 (0.0036)	**0.2887** (0.0028)	0.2898 (0.0038)
CMM	**0.1796***(0.0008)	**0.8269***(0.0026)	**0.2854***(0.0020)	0.2450 (0.0036)

sampling examples, label orders or label sets. However, in this chapter we limit our attention to the methods themselves (except RK, which is already the ensemble version). Random k-labelsets is tested in its ensemble form (of size 30) since the raw version is not a complete multi-label classifier. All methods except CLR need a threshold to assign 0/1 values to labels. We simply use 0.5. One can also search for an optimal threshold in $[0, 1]$. The regularization parameters of the base learners are chosen by cross validation. We use K=2 for RK since most rivals focus on pairwise label relations.

Evaluation. We report on four evaluation measures. 1) Hamming loss: the percentage of misclassified labels; 2) subset 0-1 loss: the percentage of examples that at least one label is misclassified, which tests the ability to capture label dependency; 3) ranking loss: the probability that an irrelevant label is ranked higher than a relevant label, which measures the ability to capture the relative order between labels. 4) One error: the percentage of examples for which the top ranked label turns out to be irrelevant.

Experimental settings. We perform 30 random runs and report means and standard errors of each evaluation measure. Data sets come with more than enough training examples (several of them contain thousands of training samples). As we are interested in performance with limited

training data, in each random run we sample 200 training examples.

Results are shown in Table 7.1 to Table 7.5. We summarize the results as follows:

- PLR and CLR perform well on ranking loss and one error, because their decisions are mainly based on pairwise comparison of labels and focus more on obtaining the correct rank order of labels. PLR has terrible performance on hamming loss and subset 0/1 loss because PLR is designed as a ranking algorithm and is incapable of deciding the relevance/irrelevance of labels. CLR is designed to improve PLR by introducing a virtual label (which serves as an adaptive threshold between relevant and irrelevant labels) and thus CLR obtains average performance on hamming loss and subset 0/1 loss.
- CC attains average performance on hamming loss and subset 0/1 loss, but falls behind on ranking loss and one error. The conditional relation of labels captures certain label dependencies, but is not well suited for label ranking. It has been reported that ensemble CC has better performance, and interesting future work is to compare the ensemble versions of different methods.
- RK: as the only method coupled with a randomized ensemble technique, RK achieves decent performance on hamming and subset 0/1 loss but below-average scores on ranking loss and one error. One potential reason for the unsatisfactory ranking performance is the use of simple voting to combine predictions from subproblems, which may lose information that is critical for accurate ranking.
- LEAD offers above-average performance on Medical and Yeast data sets but is not very competitive on other data sets. Jointly modeling the label dependency using a single graphical model is promising, but it also tends to require more training samples than learning simple subproblems in decomposition methods. Also, structure learning of Bayesian nets (or parameter estimation in undirected graphical models) are still computationally intractable.
- CMM delivers the top performance on all four evaluation criteria (in terms of the number of wining data sets). This is the only method competitive for both classification and ranking. Subproblems of estimating univariate and bivariate label densities convey more information than pairwise label comparisons, and the robust mean-field procedure provides a decent approximation to the composite likelihood without losing critical information for classification and ranking.

7.6 Conclusion

In this chapter we show the connection between multi-label problem decomposition methods and composite likelihoods. We believe this connection holds great promise for improving the design of multi-label problem decomposition in both the choice of subproblems and the combination of subproblem decisions. As an attempt to exploit this connection, we design a composite marginal method that improves existing pairwise decomposition algorithms. Pairwise label comparisons are replaced by pairwise label density estimations, which offer more informative and natural components in the composite likelihood. For combining subproblem decisions, we propose a new mean-field approximation procedure that minimizes the notion of composite divergence and is potentially more robust to inaccurate estimation in subproblems. Empirical studies show that the

proposed method outperforms many alternatives under a variety of evaluation criteria.

Chapter 8

Multi-Label Output Codes using Canonical Correlation Analysis

8.1 Introduction

In this chapter [ZS11], we study problem-dependent output coding for multi-label classification, where the label encoding (i.e., the choice of subproblems) is adaptively learned from data instead of being predefined (as in pairwise decomposition). We propose an output coding scheme using label projections, and more specifically, the most predictable directions in the label space found by canonical correlation analysis. These predictable label projections, as part of the new output code, will be predicted by regression models. From the composite likelihood view, we no longer use pre-defined pairwise label densities as in Chapter 7, but instead we define Gaussian densities on the adaptively learned label projections. This new coding scheme emphasizes the predictability of the generated output codes and thus can effectively improve the performance of multi-label classification. In the rest of this section, we first review error-correcting output codes (ECOCs) designed for multi-class classification problems and a few important issues when applying ECOCs to multi-label prediction, and then we give an overview of our new multi-label output codes.

8.1.1 From Multi-class Output Codes to Multi-label Output Codes

Error-correcting output codes (ECOCs) are traditionally designed to decompose a multiclass classification problem into many binary subproblems [DB95]. Using ECOCs, the multiclass problem can be solved using only binary classifiers for subproblems, and more importantly, since the binary subproblems also provide a redundant representation for the original multiclass classification problem, certain prediction errors can be corrected during the decoding process. Two key components of an output coding scheme are *encoding* and *decoding*. The encoding maps each class to a *codeword*, which contains the outcomes of all decomposed binary subproblems on that class. Models are learned from training examples to predict the codewords of new examples, and the class of a new example is obtained by decoding the predicted codeword.

Unlike classes, labels in multi-label classification are not mutually exclusive. In a q-label problem, the cardinality of the output space $\mathcal{Y} = \{0, 1\}^q$ is 2^q instead of q in a q-class problem. This change of output space brings up a few issues when applying ECOCs to multi-label problems:

- **Validity of the encoding**. Many multi-class ECOCs encode the output into a number of binary decision subproblems, each differentiating two subsets of classes. In multi-label cases, two subsets of labels can be simultaneously satisfied, which makes the binary outcome ill-defined. For example, in pairwise decomposition, the comparison between label i and label j is not well-defined on examples where both labels are true. We may ignore all such training examples when training the binary classifier, but in the testing phase we will not know if a test example should be ignored by this classifier. Ideally, we want the encoding to be well-defined for all examples.
- **Efficiency of the decoding**. An ECOC usually decodes a predicted codeword either by simple heuristics such as voting, or by exhaustively searching over all classes to optimize certain criteria (e.g., the Hamming loss to the predicted codeword). For an q-label problem, searching over all 2^q label vectors in $\{0, 1\}^q$ is inefficient, and effective yet efficient decoding is desirable. One observation we notice is that if the encoding is nonlinear (e.g., decomposition into pairwise comparisons of labels), the decoding has to solve a nonlinear system.
- **Predictability of the codeword**. Given an ECOC, classification is performed by predicting the codeword and then decoding. In this sense, predictability of the codeword is an important concern for code design, i.e., the subproblems should be easy to learn. In multi-label classification, encoding via decomposing into binary subproblems is not always easy to solve, and a new encoding protocol that produces predictable codewords is needed. Note that the predictability issue also exists in multi-class ECOCs, but in multi-label problems, we have new opportunities to address this issue by exploiting the label dependency.
- **Dependency among labels**. A key difference between multi-label classification and multi-class classification is the existence of rich label dependency structures. In multiclass classification, mutual exclusion of classes (i.e., one class is true and all other classes are false) eliminates most dependency structures. In multi-label classification problems, however, a variety of co-occurring patterns can exist among labels, which provides critical information and new opportunities for finding predictable codewords and thus effective error-correcting output codes.

8.1.2 Overview: Coding with Canonical Correlation Analysis

In this chapter, we propose an error-correcting output code for multi-label classification, which offers valid encoding, efficient decoding, and predictable codewords which exploit label dependency. In our coding scheme, label dependency is characterized as the most predictable directions in the label space and extracted as the canonical output variates in canonical correlation analysis. The codeword encodes these most predictable variates as well as the original labels. Predictions for the codeword define a graphical model of labels with both Bernoulli potentials (from binary classifiers on the labels) and Gaussian potentials (from regression on the canonical variates). The decoding is performed by mean-field approximation, which offers a tractable predictive distribution on labels and improves the prediction performance in multi-label classification.

We establish connections between our work and compressed sensing [Don06, HKLZ09] and ensemble learning [Bre96, FS96, BS01]. Our coding scheme can be viewed as an advanced sensing protocol, where random projection directions in compressed sensing are replaced by the most predictable directions in the label space, and noninformative sparsity priors are replaced by informative Bernoulli priors from trained classifiers. Also, we view ensemble learning in a prediction system as simulating the behavior of repetition codes in channel coding [CT91]. As a result, the successful use of repetition codes in concatenated coding [CF07] suggests the concatenation of ensemble learning with our coding scheme to produce more powerful output codes.

In our empirical study, the proposed output coding scheme leads to substantial improvements on multi-label prediction accuracy compared to a variety of state-of-the-art competitors in music emotion classification and outdoor scene recognition.

8.2 Related Work

Multi-label compressed sensing [HKLZ09] is perhaps the first output code defined for multi-label prediction. Our work is fundamentally different from this prior work. First, the goal of [HKLZ09] is to reduce the number of predictions by applying *source coding* (i.e., compression) to labels, while our work aims to introduce additional redundancy and improve prediction by applying *channel coding* (error-correcting codes). Second, the encoding in [HKLZ09] uses random projections as in compressed sensing [Don06, Can06], while our encoding includes the most predictable directions in the label space for error correction. Third, the decoding in [HKLZ09] recovers real-valued signals as in compressed sensing, while our decoding addresses the difficulty of inferring label assignments. We will study this prior work in our empirical study.

"Problem transformation" methods for mining multi-label data [TKV10] transform the multi-label classification problem into a set of subproblems and essentially adapt multi-class ECOCs to multi-label classification. Calibrated pairwise label ranking [FHMB08] is based on one-vs-one decomposition plus an adaptive calibration technique. For the validity of the encoding, a pairwise comparison may not be well-defined on certain examples (where the pair of labels are both true or both false), and the solution is to ignore all such training examples in learning, and apply the learned pairwise model to every testing example regardless of the issue. This may lead to ineffective use of training examples and unavoidable errors on certain testing examples. For decoding, voting from all pairwise models is used. We will study this method in our experiments.

Ensemble methods such as bagging [Bre96], boosting [FS96] and random forests [BS01] are also related to output coding in the sense that they simulate repetition codes [CT91, CF07].

8.3 Multi-Label Output Codes using Canonical Correlation Analysis

Algorithm 2 The encoding and training algorithm

Input: training data $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$
Parameters: d, the number of canonical variates
Output: q classifiers and d regression models

$\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^{d} \leftarrow \text{CCA}(\mathbf{X}, \mathbf{Y})$
$\mathbf{z}^{(i)} = (y_1^{(i)}, \ldots, y_q^{(i)}, \mathbf{v}_1^T \mathbf{y}^{(i)}, \ldots, \mathbf{v}_d^T \mathbf{y}^{(i)})^T, \ \forall i$
for $j = 1$ to q **do**
  $\hat{p}_j \leftarrow$ learn_classifier$(\{(\mathbf{x}^{(i)}, z_j^{(i)})\}_{i=1}^{n})$
end for
for $k = 1$ to d **do**
  $\hat{m}_k \leftarrow$ learn_regression$(\{(\mathbf{x}^{(i)}, z_{q+k}^{(i)})\}_{i=1}^{n})$
end for

8.3.1 Encoding

Channel coding [CT91, CF07] studies reliable communication through noisy channels. To reliably transmit a message $\mathbf{y}$ through a noisy channel, we *encode* it into a redundant codeword $\mathbf{z}$. When the corrupted $\hat{\mathbf{z}}$ is received at the other end of the channel, the encoded redundancy can be used to correct the transmission error and recover the true message. In multi-label prediction, the label vector $\mathbf{y} \in \{0, 1\}^q$ of an example $\mathbf{x}$ is the message. The learning-and-prediction process is viewed as transmission through a noisy channel, and prediction errors correspond to transmission errors from the channel. As in channel coding, the goal of encoding the label vector $\mathbf{y}$ into a redundant codeword $\mathbf{z}$ is to recover $\mathbf{y}$ accurately given our noisy predictions from $\mathbf{x}$.

It is critical to realize a key difference between a noisy channel and a prediction system: transmission errors of a noisy channel are usually *independent* of the information being transmitted, while prediction errors of a trained system often *depend* on the signal being predicted. In other words, some signals are easier to predict than others. As a result, it is highly desirable that the codeword $\mathbf{z}$ generated by an output coding scheme is ***predictable*** from the input.

Consider a set of p variables $\mathbf{x} \in \mathcal{X} \subseteq R^p$ and another set of q variables $\mathbf{y} \in \mathcal{Y} = \{0, 1\}^q$. For a multi-label classification problem, $\mathbf{x}$ will denote the feature vector and $\mathbf{y}$ will denote the label vector. In addition, we have a set of n training examples: $\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{n}$, where $\mathbf{X}$ and $\mathbf{Y}$ are matrices of size $n \times p$ and $n \times q$, respectively. As introduced in Section 2.8, canonical correlation analysis finds pairs of projection directions $\{(\mathbf{u}_k \in R^p, \mathbf{v}_k \in R^q)\}_{k=1}^{d}$ to successively maximize the correlation between the pair of projected variables $\mathbf{u}_k^T \mathbf{x}$ and $\mathbf{v}_k^T \mathbf{y}$ for $k = 1, 2, \ldots, d$. The canonical output variates $\{\mathbf{v}_k^T \mathbf{y}\}_{k=1}^{d}$ found by CCA as in (2.17) are known as the *most predictable* variates [Hot35]. To see this, we can rewrite the original CCA formulation (2.14)

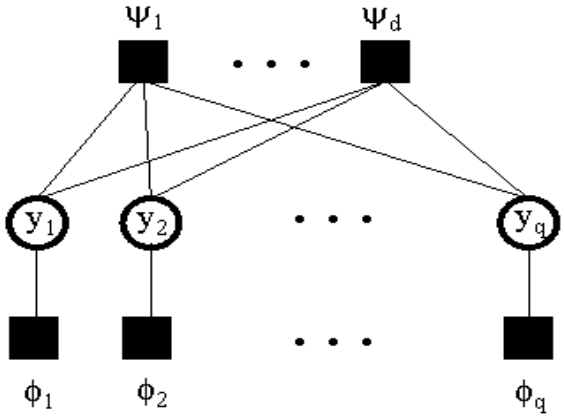


Figure 8.1: Factor graph representation of the undirected graphical model for decoding.

into the following equivalent problem:

$$\begin{aligned} \underset{\mathbf{u}\in R^p,\mathbf{v}\in R^q}{\operatorname{argmin}} \quad & ||\mathbf{X}\mathbf{u}-\mathbf{Y}\mathbf{v}||^2 \\ \text{s.t.} \quad & \mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u}=1 \\ & \mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v}=1 \end{aligned} \tag{8.1}$$

which simply minimizes the sum of squared errors over examples $(\mathbf{X},\mathbf{Y})=\{(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\}_{i=1}^n$ when using $\mathbf{u}$ as a linear model to predict the variate $\mathbf{v}^T\mathbf{y}$ from $\mathbf{x}$.

Thus, canonical output variates are ideal to be included in the codeword $\mathbf{z}$ as they will invoke minimal prediction errors. For a label vector $\mathbf{y}=(y_1,\ldots,y_q)^T$ and its canonical output variates $\{\mathbf{v}_k^T\mathbf{y}\}_{k=1}^d$, the codeword $\mathbf{z}\in R^{q+d}$ is defined by the following encoding:

$$\mathbf{z}=(y_1,\ldots,y_q,\mathbf{v}_1^T\mathbf{y},\ldots,\mathbf{v}_d^T\mathbf{y})^T \tag{8.2}$$

Note that this defines a systematic code [CT91], where the codeword contains the original message (the label vector $\mathbf{y}=(y_1,\ldots,y_q)^T$ in this case) as a subcomponent. The intuition is that the message always contains high information about itself. Given a set of training examples $(\mathbf{X},\mathbf{Y})=\{(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\}_{i=1}^n$, we will learn q classifiers $\{\hat{p}_1,\ldots,\hat{p}_q\}$ to predict the q labels $\{y_1,\ldots,y_q\}$, and d regression models $\{\hat{m}_1,\ldots,\hat{m}_d\}$ to predict the d canonical variates $\{\mathbf{v}_1^T\mathbf{y},\ldots,\mathbf{v}_d^T\mathbf{y}\}$. The encoding and training procedure is summarized in **Algorithm 2**.

Note that the challenges discussed in Section 8.1.1 are being addressed: 1) the encoding in (8.2) is well-defined for any label vector $\mathbf{y}\in\{0,1\}^q$; 2) the encoding is *linear*, which enables us to develop efficient decoding (in the next section); 3) $\{\mathbf{v}_k^T\mathbf{y}\}_{k=1}^d$ included in the codeword $\mathbf{z}$ are the most (linearly) predictable variates; 4) the label dependency in $\mathbf{y}$ is exploited (via projection vectors $\{\mathbf{v}_k\}_{k=1}^d$) to produce predictable codewords[1].

[1]Note that $\mathbf{v}^T\mathbf{y}$ represents the label combination that is most predictable from $\mathbf{x}$, so the projection vector $\mathbf{v}$ mainly captures the *conditional* label dependency given $\mathbf{x}$.

Algorithm 3 The decoding and prediction algorithm

Input: a testing point $\mathbf{x}$, q classifiers $\{\hat{p}_j\}_{j=1}^q$, d regression models $\{(\hat{m}_k, \hat{\sigma}_k^2)\}_{k=1}^d$
Parameters: λ
Output: a fully factorized predictive distribution $Q(\mathbf{y}) = \prod_{j=1}^q Q_j(y_j)$ for $\mathbf{y}$

$\{Q_j(y_j)\}_{j=1}^q \leftarrow$ mean-field-approximation($\mathbf{x}$, $\{\hat{p}_j\}_{j=1}^q$, $\{(\hat{m}_k, \hat{\sigma}_k^2)\}_{k=1}^d$, λ)
Use $Q_j(y_j)$ as a predictive distribution on label y_j, $j = 1, 2, \ldots, q$

8.3.2 Decoding

For a testing example $\mathbf{x}$, classification and regression models learned in Algorithm 2 provide a predictive distribution on the codeword $\mathbf{z}$. Each classifier $\hat{p}_j$ predicts a Bernoulli distribution $\phi_j(y_j)$ for a label y_j:

$$\phi_j(y_j) = \hat{p}_j(\mathbf{x})^{y_j}(1 - \hat{p}_j(\mathbf{x}))^{(1-y_j)}, \quad j = 1, 2, \ldots, q \tag{8.3}$$

and each regression model $\hat{m}_k$ predicts a Gaussian distribution $\psi_k(\mathbf{y})$ for a canonical output variate $\mathbf{v}_k^T\mathbf{y}$:

$$\psi_k(\mathbf{y}) \propto \exp -\frac{(\mathbf{v}_k^T\mathbf{y} - \hat{m}_k(\mathbf{x}))^2}{2\hat{\sigma}_k^2}, \quad k = 1, 2, \ldots, d \tag{8.4}$$

where the variance $\hat{\sigma}_k^2$ can be estimated by cross validation on the training examples in Algorithm 2.

For decoding, predictive distributions (8.3) and (8.4) can be viewed as the factor graph in Figure 8.1, which defines the following joint probability for the label vector $\mathbf{y}$ given $\mathbf{x}$:

$$\log P(\mathbf{y}) = -\log \mathcal{Z} + \sum_{k=1}^d \log \psi_k(\mathbf{y}) + \lambda \sum_{j=1}^q \log \phi_j(y_j) \tag{8.5}$$

where $\mathcal{Z}$ is the partition function, $\log$ is natural logarithm, and λ balances two types of potentials.

Unfortunately, exact inference over $\mathbf{y} \in \{0, 1\}^q$ in (8.5) has a time complexity *exponential* in q, due to the fact that each Gaussian potential ψ_k in (8.4) usually involves all the q labels. We consider a mean-field approximation to $P(\mathbf{y})$ in (8.5), as the following:

$$Q(\mathbf{y}) = \prod_{j=1}^q Q_j(y_j) \tag{8.6}$$

which is the class of fully factorized distributions of $\mathbf{y}$ that allows tractable inference over labels. Note that each $Q_j(y_j)$ is a Bernoulli distribution on the label y_j.

To find the best approximation $Q(\mathbf{y})$ in the fully factorized class, we need to minimize the KL divergence $KL(Q||P)$, or equivalently, to maximize the energy functional [JGJS99]. Based on the definition of $P(\mathbf{y})$ in (8.5), the fixed-point equation for updating each Bernoulli factor $Q_j(y_j)$ in Q can be written as [KF09]:

$$Q_j(y_j) \leftarrow \frac{1}{\mathcal{Z}_j} \exp\{\, \lambda \log \phi_j(y_j) + \sum_{k=1}^d \mathbf{E}_{\mathbf{y}\sim Q}[\log \psi_k(\mathbf{y})|y_j] \,\} \tag{8.7}$$

where $\mathcal{Z}_j$ is the normalization constant that makes Q_j a valid Bernoulli distribution. Note that each term $\mathbf{E}_{\mathbf{y}\sim Q}[\log \psi_k(\mathbf{y})|y_j]$ can be computed with a time complexity *polynomial* in q (or more specifically, in $O(q^2)$), due to the fact that Q is fully factorized over labels and the Gaussian log density $\log \psi_k$ involves only second order interactions between labels. Thus, the mean-field approximation $Q(\mathbf{y})$ can be obtained by iterating the fixed-point update over j until convergence.

For each testing example $\mathbf{x}$, mean-field approximation provides a factorized distribution $Q(\mathbf{y}) = \prod_{j=1}^{q} Q_j(y_j)$ on labels, where factors $\{Q_j(y_j)\}_{j=1}^{q}$ can be directly used for various prediction tasks, e.g., classification, ranking, probabilistic inference. The decoding and prediction procedure is summarized in **Algorithm 3**.

8.4 Connections to Other Research Areas

In this section, we establish and analyze connections between the proposed code and other research areas such as compressed sensing and ensemble learning.

8.4.1 Multi-label Compressed Sensing

Encoding. Multi-label compressed sensing [HKLZ09] is perhaps the earliest work that formally defines an output code for multi-label prediction. Given a label vector $\mathbf{y} \in \{0,1\}^q$, the *encoding* is defined as the following:

$$\mathbf{z} = (\mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T \tag{8.8}$$

where each $\mathbf{v}_k \in R^q, k = 1, 2, \ldots, d$ is a *random* projection vector, e.g., with i.i.d. Gaussian or Bernoulli entries. The codeword is shorter than the original label vector, i.e., $d < q$, since the design goal of multi-label compressed sensing is to *reduce* the number of prediction models. However, this encoding can produce highly redundant codewords by using a large number of projections ($d >> q$), which we will study as an error-correcting output code in our experiments.

Decoding. For a test example $\mathbf{x}$, its codeword is predicted as $\hat{\mathbf{z}} = (\hat{m}_1(\mathbf{x}), \ldots, \hat{m}_d(\mathbf{x}))^T$ by trained regression models $\{\hat{m}_k\}_{k=1}^{d}$. The *decoding* assumes the *sparsity* of the true label vector $\mathbf{y}$ and reconstructs $\mathbf{y}$ using *sparse approximation* techniques. Popular choices include ℓ_1 penalized convex optimization [Tro06] and iterative methods such as CoSaMP [NT08]. To analyze the link between compressed sensing and our coding scheme, we focus on ℓ_1 penalized convex relaxation [Tro06]:

$$\underset{\mathbf{y}\in R^q}{\operatorname{argmin}} \ \frac{1}{2}\sum_{k=1}^{d}(\mathbf{v}_k^T\mathbf{y} - \hat{m}_k(\mathbf{x}))^2 + \lambda \sum_{j=1}^{q}|y_j| \tag{8.9}$$

which minimizes approximation errors on the d projection (i.e., "sensing") measurements plus a sparsity-inducing ℓ_1 penalty $\sum_{j=1}^{q}|y_j| = ||\mathbf{y}||_1$ on labels. Note that the optimization is performed over the continuous domain $\mathbf{y} \in R^q$ for computational tractability.

Connection. To see the connection between our proposed code and the compressed sensing above, we consider optimization of the joint probability in (8.5) over labels $\mathbf{y} \in \{0,1\}^q$. Plugging

Bernoulli potentials (8.3) and Gaussian potentials (8.4) into the joint probability and ignoring the log partition term, maximizing (8.5) is equivalent to the following minimization problem:

$$\underset{\mathbf{y}\in\{0,1\}^q}{\operatorname{argmin}} \frac{1}{2}\sum_{k=1}^{d}\frac{(\mathbf{v}_k^T\mathbf{y}-\hat{m}_k(\mathbf{x}))^2}{\hat{\sigma}_k^2}+\lambda\sum_{j=1}^{q}y_j\log(\frac{1-\hat{p}_j(\mathbf{x})}{\hat{p}_j(\mathbf{x})}) \tag{8.10}$$

where the objective contains d squared error terms and q linear terms, from the d Gaussian potentials and q Bernoulli potentials in (8.5), respectively. We can see that optimizing the joint label probability as in (8.10) is similar to reconstructing signals in compressed sensing as in (8.9), but with the following key differences:

- Projection vectors $\{\mathbf{v}_k\}_{k=1}^d$ are canonical output directions from CCA instead of random projections as in compressed sensing.
- Each error term $(\mathbf{v}_k^T\mathbf{y}-\hat{m}_k(\mathbf{x}))^2$ is weighted by $1/\hat{\sigma}_k^2$.
- An informative penalty $\sum_{j=1}^q y_j\log(\frac{1-\hat{p}_j(\mathbf{x})}{\hat{p}_j(\mathbf{x})})$ (derived from the trained classifiers $\{\hat{p}_j\}_{j=1}^q$ on q labels) replaces the noninformative ℓ_1 penalty $\sum_{j=1}^q|y_j|$.
- Optimization is performed in the label space $\mathbf{y}\in\{0,1\}^q$ instead of a relaxed domain R^q.

Note that our decoding uses mean-field approximation to solve the intractable optimization problem.

8.4.2 Ensemble Learning as Repetition Codes

Repetition codes are a class of error-correcting codes where the message is simply encoded by repetition, e.g., $\mathbf{y}=(y_1,y_2,\ldots,y_q)^T\in\{0,1\}^q$ will be encoded as:

$$\mathbf{z}=(y_1^1\ldots y_1^r,y_2^1\ldots y_2^r,\ldots,y_q^1\ldots y_q^r)^T\in\{0,1\}^{rq} \tag{8.11}$$

where r is the number of repetitions. If the channel randomly flips each bit in $\mathbf{z}$ with a probability $p<0.5$, decoding for y_j can take a majority voting from received bits $\{y_j^1\ldots y_j^r\}$. If the channel is making *independent* flipping errors (with $p<0.5$), we can achieve error-free communication using a sufficiently large r.

Ensemble learning imitates repetition codes in a prediction system by learning multiple models for the same label. However, if we simply repeat a label, predictions from repeatedly trained models will be identical. If the "channel" makes *identical* (instead of independent) errors, majority voting will not correct any of them. Therefore, ensemble learning aims to create a *diverse* set of models whose prediction errors are as independent as possible [KHDM98] via perturbation on training samples [Bre96], sample weights [FS96], features [BS01], etc[2].

Concatenated coding. Repetition codes, although not efficient when used alone due to their high redundancy, play an active role in *concatenated coding*. A concatenated code involves the cascade of an *outer* code and an *inner* code. The message is first encoded by the outer encoder and the resulting codeword is further encoded by the inner encoder. An efficient decoding can be

[2]Bagging and random forests fit this view better than boosting, as boosting is not a simple random repetition.

Table 8.1: Subset accuracy on Emotions data: means (standard errors) over 30 runs.

Method: #models	200 samples	250 samples	300 samples	350 samples	391 samples
One-vs-All(LGR):6	0.2058(0.0037)	0.2185(0.0036)	0.2213(0.0034)	0.2239(0.0027)	0.2206(0.0013)
One-vs-All(SVM):6	0.2167(0.0062)	0.2190(0.0045)	0.2195(0.0065)	0.2381(0.0054)	0.2386(0.0052)
Pairwise(LGR):21	0.2086(0.0036)	0.2208(0.0043)	0.2264(0.0030)	0.2322(0.0035)	0.2331(0.0020)
Pairwise(SVM):21	0.2190(0.0045)	0.2239(0.0040)	0.2328(0.0040)	0.2381(0.0044)	0.2512(0.0039)
CCA-Reduce(LGR):6	0.2079(0.0054)	0.2259(0.0040)	0.2251(0.0041)	0.2451(0.0035)	0.2594(0.0006)
CCA-Reduce(SVM):6	0.2059(0.0060)	0.2190(0.0074)	0.2289(0.0045)	0.2408(0.0082)	0.2551(0.0060)
CompressSensing:300	0.2096(0.0049)	0.2116(0.0026)	0.2195(0.0021)	0.2226(0.0022)	0.2221(0.0007)
One-vs-All(Tree):6	0.1343(0.0043)	0.1376(0.0043)	0.1452(0.0039)	0.1469(0.0043)	0.1436(0)
One-vs-All(Forest):300	0.2426(0.0041)	0.2540(0.0026)	0.2571(0.0032)	0.2649(0.0035)	0.2667(0.0034)
CCA-OC:12	**0.2944(0.0051)**	**0.3153(0.0037)**	**0.3322(0.0033)**	**0.3366(0.0024)**	**0.3472(0.0016)**
CCA-OC-Forest:600	**0.3318(0.0033)**	**0.3411(0.0029)**	**0.3419(0.0024)**	**0.3432(0.0026)**	**0.3480(0.0023)**

performed by sequentially applying the inner and outer decoder. Using two simpler codes, concatenated coding provides a longer, more powerful code with efficient encoding and decoding and has produced high performance code in practice, e.g., the code in NASA standards in 1970s [CF07].

In our experiments, we will also study a concatenated coding scheme, where our proposed code serves as the outer code and ensemble learning is the inner code. The intuition is that ensemble learning, as a repetition code, improves the prediction on each individual output; our code, on the other hand, contributes by exploiting the dependency among different outputs.

8.5 Experiments

Data. We conduct our experiments on two multi-label data sets: *Emotions* and *Scene*[3]. Emotions is a music classification data set containing 391 songs in the training set and 202 songs in the testing set. Songs are labeled with six emotions: amazed, happy, relaxed, quiet, sad and angry. Each song is represented by 72 rhythmic and timbre features. Scene is an image collection for outdoor scene recognition, with 1211 training images and 1196 testing images tagged with six scenes: beach, sunset, fall foliage, field, mountain and urban. Features are 294 dimensional color statistics.

Methods. We study the following methods:

- *One-vs-All(LGR/SVM)*: one-vs-all decomposition with ℓ_2 penalized logistic regression or SVM.
- *Pairwise(LGR/SVM)*: calibrated pairwise label ranking [FHMB08] with ℓ_2 logistic regression or SVMs.
- *CCA-Reduce(LGR/SVM)*: supervised dimension reduction by CCA and one-vs-all in reduced feature space. We report results with optimal dimensions.
- *CompressSensing*: multi-label compressed sensing (CS) [HKLZ09], with 300 projections to pro-

[3]http://mulan.sourceforge.net/datasets.html.

Table 8.2: Micro-averaged F1 score on Emotions data: means (standard errors) over 30 runs.

Method: #models	200 samples	250 samples	300 samples	350 samples	391 samples
One-vs-All(LGR):6	0.6181(0.0034)	0.6301(0.0027)	0.6338(0.0026)	0.6416(0.0019)	0.6433(0.0005)
One-vs-All(SVM):6	0.6258(0.0040)	0.6322(0.0029)	0.6404(0.0040)	0.6505(0.0023)	0.6498(0.0036)
Pairwise(LGR):21	0.6219(0.0029)	0.6315(0.0025)	0.6350(0.0024)	0.6414(0.0020)	0.6421(0.0011)
Pairwise(SVM):21	0.6286(0.0042)	0.6295(0.0040)	0.6373(0.0033)	0.6458(0.0020)	0.6518(0.0022)
CCA-Reduce(LGR):6	0.6045(0.0033)	0.6214(0.0027)	0.6285(0.0032)	0.6338(0.0020)	0.6366(0.0008)
CCA-Reduce(SVM):6	0.6006(0.0040)	0.6200(0.0042)	0.6332(0.0034)	0.6372(0.0023)	0.6445(0.0030)
CompressSensing:300	0.5729(0.0050)	0.5795(0.0033)	0.5820(0.0026)	0.5849(0.0019)	0.5890(0.0008)
One-vs-All(Tree):6	0.5550(0.0043)	0.5603(0.0038)	0.5677(0.0030)	0.5778(0.0038)	0.5540(0)
One-vs-All(Forest):300	0.6247(0.0030)	0.6389(0.0022)	0.6413(0.0024)	0.6499(0.0024)	0.6518(0.0021)
CCA-OC:12	**0.6499(0.0047)**	**0.6646(0.0031)**	**0.6792(0.0032)**	**0.6813(0.0020)**	**0.6828(0.0015)**
CCA-OC-Forest:600	**0.6828(0.0026)**	**0.6887(0.0021)**	**0.6888(0.0023)**	**0.6935(0.0018)**	**0.6959(0.0015)**

Table 8.3: Macro-averaged F1 score on Emotions data: means (standard errors) over 30 runs.

Method: #models	200 samples	250 samples	300 samples	350 samples	391 samples
One-vs-All(LGR):6	0.6081(0.0038)	0.6198(0.0029)	0.6220(0.0029)	0.6304(0.0023)	0.6290(0.0009)
One-vs-All(SVM):6	0.6124(0.0045)	0.6179(0.0035)	0.6297(0.0049)	0.6378(0.0040)	0.6355(0.0055)
Pairwise(LGR):21	0.6102(0.0031)	0.6187(0.0025)	0.6216(0.0026)	0.6284(0.0023)	0.6293(0.0015)
Pairwise(SVM):21	0.6145(0.0043)	0.6159(0.0042)	0.6198(0.0047)	0.6339(0.0032)	0.6349(0.0041)
CCA-Reduce(LGR):6	0.5954(0.0033)	0.6112(0.0031)	0.6178(0.0032)	0.6229(0.0023)	0.6202(0.0004)
CCA-Reduce(SVM):6	0.5894(0.0047)	0.6085(0.0043)	0.6202(0.0036)	0.6228(0.0030)	0.6282(0.0053)
CompressSensing:300	0.5263(0.0046)	0.5343(0.0036)	0.5354(0.0029)	0.5370(0.0022)	0.5417(0.0009)
One-vs-All(Tree):6	0.5467(0.0043)	0.5528(0.0040)	0.5603(0.0032)	0.5585(0.0039)	0.5426(0)
One-vs-All(Forest):300	0.5973(0.0040)	0.6124(0.0030)	0.6172(0.0032)	0.6286(0.0026)	0.6326(0.0025)
CCA-OC:12	**0.6392(0.0047)**	**0.6558(0.0029)**	**0.6715(0.0031)**	**0.6740(0.0021)**	**0.6767(0.0015)**
CCA-OC-Forest:600	**0.6712(0.0026)**	**0.6771(0.0023)**	**0.6782(0.0024)**	**0.6846(0.0019)**	**0.6856(0.0017)**

duce highly redundant codewords, ridge regression to predict, and CoSaMP [NT08] to decode. CS recovers real values, and we use 0.5 as the threshold. We report results with the optimal CoSaMP parameter (sparsity level).

- *One-vs-All(Tree/Forest)*: one-vs-all decomposition with classification trees [BFSO84] or random forests [BS01] (50 trees per label). Comparing these two models shows the power of ensemble learning as a repetition code.
- *CCA-OC*: our proposed coding with ℓ_2 logistic regression and ridge regression as classifiers and regression models. We simply use the maximal d from CCA in Algorithm 2, and the parameter λ in Algorithm 3 is chosen from $\{\frac{1}{4}, 1, 4\}$.
- *CCA-OC-Forest*: our proposed coding with random forests, where random forests provide both accurate classification and regression. This is a concatenated coding scheme with our code as

Table 8.4: Subset accuracy on Scene data: means (standard errors) over 30 runs.

Method: #models	400 samples	600 samples	800 samples	1000 samples	1211 samples
One-vs-All(LGR):6	0.4356(0.0031)	0.4542(0.0035)	0.4710(0.0037)	0.4785(0.0030)	0.4868(0.0011)
One-vs-All(SVM):6	0.4210(0.0041)	0.4439(0.0046)	0.4634(0.0032)	0.4758(0.0026)	0.4861(0.0018)
Pairwise(LGR):21	0.4312(0.0032)	0.4572(0.0035)	0.4653(0.0023)	0.4749(0.0027)	0.4884(0.0012)
Pairwise(SVM):21	0.4183(0.0037)	0.4449(0.0034)	0.4634(0.0028)	0.4687(0.0026)	0.4857(0.0013)
CCA-Reduce(LGR):6	0.1585(0.0032)	0.2940(0.0024)	0.3622(0.0022)	0.4071(0.0020)	0.4438(0.0002)
CCA-Reduce(SVM):6	0.1618(0.0034)	0.2938(0.0022)	0.3594(0.0027)	0.3983(0.0030)	0.4259(0.0045)
CompressSensing:300	0.3917(0.0047)	0.4107(0.0030)	0.4199(0.0024)	0.4246(0.0017)	0.4322(0.0008)
One-vs-All(Tree):6	0.2948(0.0034)	0.3205(0.0039)	0.3373(0.0034)	0.3537(0.0031)	0.3678(0)
One-vs-All(Forest):300	0.4417(0.0020)	0.4813(0.0019)	0.5012(0.0012)	0.5139(0.0016)	0.5276(0.0014)
CCA-OC:12	**0.5908(0.0041)**	**0.6063(0.0050)**	**0.6276(0.0037)**	**0.6326(0.0024)**	**0.6367(0.0011)**
CCA-OC-Forest:600	**0.6483(0.0040)**	**0.6780(0.0020)**	**0.6906(0.0015)**	**0.6977(0.0011)**	**0.7060(0.0010)**

the outer code and a repetition code as the inner code.

The regularization parameter of ℓ_2 penalized logistic regression, SVMs and ridge regression is set by cross validation within each individual learning task.

Evaluation measures. We report on three measures:

- Subset accuracy: rates of correctly classifying *all* the labels of an example. This is a difficult measure, and exploiting the label dependency is helpful.
- Micro-averaged F1 scores: aggregate true positives/negatives and false positives/negatives over labels, and then calculate an F1 score from them.
- Macro-averaged F1 scores: calculate F1 score for each label and then take the average over labels.

Experimental settings. For the Emotions data, we sampled 200, 250, 300, 350 as well as all the 391 examples from the training set. For the Scene data, we sampled 400, 600, 800, 1000 as well as all the 1211 training examples. For each training size, we perform 30 random runs and report means and standard errors of each evaluation measure over 30 runs.

Experimental results on three measures are shown in Table 8.1 - Table 8.3 for Emotions data and Table 8.4 - Table 8.6 for Scene data. The proposed methods are **in bold**. We briefly highlight the results as follows:

- CCA-OC significantly outperforms other methods and CCA-OC-Forests further enhances CCA-OC with concatenated coding. Improvements are most evident on subset accuracy (Table 8.1 and 8.4), which require capturing the label dependency. We also note that two proposed methods with 200 training samples on Emotions and 400 on Scene achieve similar or better performance than baselines trained with 391 samples on Emotions and 1211 on Scene.
- One-vs-All decomposition is a strong baseline. This is consistent with other studies [RK04].
- Calibrated pairwise label ranking (denoted by Pairwise in all tables) shows slight improvements over one-vs-all decomposition.

Table 8.5: Micro-averaged F1 score on Scene data: means (standard errors) over 30 runs.

Method: #models	400 samples	600 samples	800 samples	1000 samples	1211 samples
One-vs-All(LGR):6	0.6203(0.0029)	0.6368(0.0024)	0.6480(0.0024)	0.6542(0.0017)	0.6589(0.0010)
One-vs-All(SVM):6	0.6091(0.0023)	0.6284(0.0027)	0.6395(0.0021)	0.6474(0.0019)	0.6570(0.0010)
Pairwise(LGR):21	0.6239(0.0024)	0.6445(0.0022)	0.6506(0.0011)	0.6599(0.0014)	0.6674(0.0008)
Pairwise(SVM):21	0.6171(0.0027)	0.6367(0.0023)	0.6460(0.0016)	0.6515(0.0014)	0.6622(0.0009)
CCA-Reduce(LGR):6	0.4185(0.0028)	0.5178(0.0022)	0.5677(0.0015)	0.6002(0.0015)	0.6259(0.0001)
CCA-Reduce(SVM):6	0.4156(0.0029)	0.5162(0.0022)	0.5654(0.0016)	0.5919(0.0020)	0.6153(0.0021)
CompressSensing:300	0.5510(0.0042)	0.5689(0.0026)	0.5781(0.0022)	0.5837(0.0017)	0.5911(0.0004)
One-vs-All(Tree):6	0.5177(0.0029)	0.5424(0.0032)	0.5549(0.0025)	0.5690(0.0022)	0.5885(0)
One-vs-All(Forest):300	0.6110(0.0021)	0.6463(0.0017)	0.6637(0.0012)	0.6742(0.0014)	0.6864(0.0011)
CCA-OC:12	**0.6541(0.0028)**	**0.6713(0.0034)**	**0.6872(0.0029)**	**0.6927(0.0023)**	**0.7000(0.0011)**
CCA-OC-Forest:600	**0.7200(0.0021)**	**0.7418(0.0015)**	**0.7518(0.0012)**	**0.7568(0.0010)**	**0.7641(0.0010)**

Table 8.6: Macro-averaged F1 score on Scene data: means (standard errors) over 30 runs.

Method: #models	400 samples	600 samples	800 samples	1000 samples	1211 samples
One-vs-All(LGR):6	0.6303(0.0029)	0.6463(0.0020)	0.6558(0.0020)	0.6608(0.0016)	0.6644(0.0009)
One-vs-All(SVM):6	0.6196(0.0023)	0.6387(0.0024)	0.6480(0.0018)	0.6546(0.0017)	0.6630(0.0009)
Pairwise(LGR):21	0.6318(0.0026)	0.6519(0.0021)	0.6579(0.0010)	0.6667(0.0014)	0.6724(0.0007)
Pairwise(SVM):21	0.6253(0.0028)	0.6449(0.0022)	0.6533(0.0015)	0.6576(0.0014)	0.6672(0.0009)
CCA-Reduce(LGR):6	0.4233(0.0029)	0.5229(0.0023)	0.5728(0.0016)	0.6041(0.0015)	0.6291(0.0001)
CCA-Reduce(SVM):6	0.4199(0.0029)	0.5215(0.0025)	0.5702(0.0016)	0.5951(0.0025)	0.6195(0.0025)
CompressSensing:300	0.5409(0.0052)	0.5616(0.0032)	0.5712(0.0025)	0.5775(0.0019)	0.5851(0.0004)
One-vs-All(Tree):6	0.5263(0.0029)	0.5504(0.0031)	0.5620(0.0026)	0.5767(0.0022)	0.5966(0)
One-vs-All(Forest):300	0.5988(0.0024)	0.6370(0.0019)	0.6562(0.0014)	0.6669(0.0017)	0.6802(0.0012)
CCA-OC:12	**0.6632(0.0028)**	**0.6799(0.0030)**	**0.6952(0.0025)**	**0.7000(0.0021)**	**0.7064(0.0012)**
CCA-OC-Forest:600	**0.7233(0.0022)**	**0.7462(0.0015)**	**0.7563(0.0012)**	**0.7612(0.0009)**	**0.7686(0.0010)**

- Dimension reduction via CCA (CCA-Reduce) is not effective: the canonical input directions may not be predictive for the original output variables (labels).
- Multi-label compressed sensing is not a capable error-correcting code even with highly redundant codewords, mainly due to its non-adaptive encoding with random projections and real-valued decoding.

8.6 Conclusion

Using canonical correlation analysis, we propose a multi-label output coding scheme with predictable codewords: label dependency is characterized as the most predictable directions in the

label space, extracted as canonical output variates and encoded into the codeword as the redundant information for error correction. Predictions for the codeword define a graphical model of labels with both Bernoulli potentials (from classifiers on the labels) and Gaussian potentials (from regression on the canonical variates). Decoding is performed by mean-field approximation. In the empirical study on multi-label classification of music and images, the proposed output code offer substantial improvements over every competitor in every test.

Chapter 9

Maximum Margin Output Coding for Multi-Label Classification

9.1 Overview

In this chapter [ZS12b], we first argue that both discriminability and predictability are important properties for multi-label output codes. Then we propose a max-margin formulation for output coding, which promotes both discriminative and predictable codes (in contrast to the CCA-based coding proposed in Chapter 8, where only the predictability of the codes is addressed).

In traditional channel coding [CT91, CF07], a message is encoded into an alternative (and usually redundant) representation so that it can be recovered accurately after being transmitted through a noisy channel. Error-correcting output coding (ECOC) applies the idea of channel coding to multi-class classification [DB95, ASS01] and more recently to multi-label prediction [HKLZ09, TL10, ZS11]: we encode the output into a codeword, learn models to predict the codeword, and for a testing example, the output is obtained by decoding the predicted codeword.

We study output coding for multi-label prediction and focus on two important issues. First, as studied in the previous chapter, output codes should be ***predictable***. In output coding, codewords need to be predicted from the input (instead of being actually transmitted through a channel), so it is critical that codewords are easy to predict. From the channel coding perspective, having predictable codewords (and thus low prediction errors) corresponds to reducing the channel error.

Second, the coding also needs to be ***discriminative***: encodings for different outputs should be significantly different from each other, so that the codeword for the correct output will not be confused with incorrect ones, even under noisy predictions. This corresponds to the concept of code distance in coding theory and is related to good error-correcting capabilities [CT91]. Note that this is what the CCA-based coding in the previous chapter is missing.

To design output codes that are both discriminative and predictable, we propose a max-margin formulation defined on the encoding transform. For each sample, the prediction from the input should be close to the encoding of the correct output, and at the same time, the prediction should also be far away from the encoding of any incorrect output. This is naturally captured by maximiz-

ing the margin between the prediction distance to correct and incorrect output encodings.

We then convert this formulation to a metric learning problem of finding the optimal distance metric in the label space, but with an exponentially large number of constraints as commonly encountered in structured prediction problems. In multi-label prediction, however, the output space does not provide a structure for tractable inference, and we use overgenerating (i.e., relaxation) techniques [FJ08] combined with the cutting plane method to optimize the metric learning formulation. The encoding and decoding can be derived from the optimal distance metric in the label space.

We conduct experiments on multi-label classification of images, text and music. Empirical results show that the proposed output code outperforms many recent multi-label prediction methods.

9.2 Multi-Label Output Codes: Framework and Existing Methods

In this section, we first introduce the general framework for multi-label output coding. Then we review three recently-proposed output coding schemes [HKLZ09, TL10, ZS11], where the encoding is based on random projections, principal component analysis and canonical correlation analysis, respectively. We also argue that these existing output coding schemes are not designed to optimize both discriminability and predictability of the codewords.

9.2.1 Framework

An output coding scheme usually contains three parts: encoding, prediction and decoding. Consider a set of p input variables $\mathbf{x} \in \mathcal{X} \subseteq R^p$ and a set of q output variables $\mathbf{y} \in \mathcal{Y} \subseteq R^q$. In multi-label classification, $\mathbf{y}$ will denote the label vector, and thus $\mathbf{y} \in \mathcal{Y} = \{0, 1\}^q$. We have a set of n training examples: $\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$, where $\mathbf{X}$ and $\mathbf{Y}$ are matrices of size $n \times p$ and $n \times q$, respectively.

Encoding. Following previous work [HKLZ09, TL10, ZS11], we consider linear encoding. In this case, the encoding transform can be defined by a set of d linear projections

$$\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2 \ldots, \mathbf{v}_d) \tag{9.1}$$

where d is the number of projections, each $\mathbf{v}_k$ ($k = 1, 2, \ldots, d$) is a $q \times 1$ vector representing a projection direction in the q-dimensional label space, and $\mathbf{V}$ is a $q \times d$ matrix.

Given the projection vectors $\mathbf{V}$, the codeword $\mathbf{z}$ for an example $(\mathbf{x}, \mathbf{y})$ is defined:

$$\mathbf{z} = \mathbf{V}^T\mathbf{y} = (\mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T \tag{9.2}$$

where $\mathbf{z}$ is a $d \times 1$ vector. Alternatively, as in Chapter 8, we can also include the q original labels $\mathbf{y} = (y_1, \ldots, y_q)$ into the codeword $\mathbf{z}$, and in this case we have:

$$\mathbf{z} = [\mathbf{I}_q, \mathbf{V}]^T\mathbf{y} = (y_1, \ldots, y_q, \mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T \tag{9.3}$$

where $\mathbf{I}_q$ is a $q \times q$ identity matrix, $\mathbf{V}$ is a $q \times d$ matrix, and $\mathbf{z}$ is a $(q+d) \times 1$ vector.

Prediction. After defining the encoding projections $\{\mathbf{v}_k\}_{k=1}^d$, we then learn prediction models from training samples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$ to predict the codeword. For a label projection $\mathbf{v}_k^T\mathbf{y}$ in the codeword, a regression model is usually considered:

$$\hat{m}_k \leftarrow \text{learn\_regression}(\{(\mathbf{x}^{(i)}, \mathbf{v}_k^T\mathbf{y}^{(i)})\}_{i=1}^n) \tag{9.4}$$

and for an original label y_j ($j = 1, 2, \ldots, q$), a classifier can be learned from training samples:

$$\hat{p}_j \leftarrow \text{learn\_classifier}(\{(\mathbf{x}^{(i)}, y_j^{(i)})\}_{i=1}^n) \tag{9.5}$$

Given a new sample $\mathbf{x}$, a regression model $\hat{m}_k$ predicts:

$$\hat{m}_k(\mathbf{x}) = E(\mathbf{v}_k^T\mathbf{y}|\mathbf{x}) \tag{9.6}$$

and a classifier $\hat{p}_j$ predicts:

$$\hat{p}_j(\mathbf{x}) = (\hat{p}_{j0}(\mathbf{x}),\ \hat{p}_{j1}(\mathbf{x})) \tag{9.7}$$

where

$$\hat{p}_{j0}(\mathbf{x}) = P(y_j = 0|\mathbf{x}) \tag{9.8}$$
$$\hat{p}_{j1}(\mathbf{x}) = P(y_j = 1|\mathbf{x}) \tag{9.9}$$

Decoding. Given a new testing sample $\mathbf{x}$, the decoding procedure recovers the unknown label vector $\mathbf{y}$ from our prediction for the codeword $\mathbf{z}$. The prediction contains $\{\hat{m}_k(\mathbf{x})\}_{k=1}^d$ and optionally $\{\hat{p}_j(\mathbf{x})\}_{j=1}^q$:

$$\hat{\mathbf{y}} \leftarrow \text{decoding}(\mathbf{x}, \{\mathbf{v}_k\}_{k=1}^d, \{\hat{m}_k(\mathbf{x})\}_{k=1}^d, \{\hat{p}_j(\mathbf{x})\}_{j=1}^q) \tag{9.10}$$

The decoding is usually achieved by maximizing a probability function or minimizing a loss function defined on possible label vector $\mathbf{y}$. Since $\mathbf{y} \in \mathcal{Y} = \{0, 1\}^q$, this optimization is usually combinatorial in nature and intractable. As a result, certain approximation is required to obtain the solution $\hat{\mathbf{y}}$, e.g., relaxing $\mathbf{y}$ into a continuous domain and then rounding the relaxed solution [HKLZ09, TL10] or using approximate inference such as the variational approaches in Chapter 7 and Chapter 8.

9.2.2 Coding with compressed sensing

Multi-label compressed sensing [HKLZ09] is one of the earliest works that formally defines a multi-label output code. For encoding, each projection vector $\mathbf{v}_k \in R^q$ ($k = 1, 2, \ldots, d$) is randomly generated as in compressed sensing [Don06, Can06], e.g., a projection vector with i.i.d. Gaussian entries or i.i.d. Bernoulli entries. Thus, the codeword $\mathbf{z} = (\mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T$ contains random projections of the label vector $\mathbf{y}$.

Decoding follows the sparse approximation algorithms in compressed sensing. Two popular classes are convex relaxation such as ℓ_1 penalized least squares [Tro06], and iterative greedy algorithms such as CoSaMP [NT08]. For example, an ℓ_1 penalized convex relaxation solves the following problem:

$$\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y} \in R^q}{\operatorname{argmin}} \frac{1}{2} \sum_{k=1}^{d} (\mathbf{v}_k^T \mathbf{y} - \hat{m}_k(\mathbf{x}))^2 + \lambda \sum_{j=1}^{q} |y_j| \tag{9.11}$$

where $\{\hat{m}_k(\mathbf{x})\}_{k=1}^d$ are predictions for the codeword $\mathbf{z} = (\mathbf{v}_1^T \mathbf{y}, \ldots, \mathbf{v}_d^T \mathbf{y})^T$, and the ℓ_1 penalty $\sum_{j=1}^q |y_j| = ||\mathbf{y}||_1$ promotes the sparsity of the solution $\hat{\mathbf{y}}$. Note that this optimization problem is solved in a relaxed continuous space $\mathbf{y} \in R^q$.

Use of random projections is justified in compressed sensing, e.g., by the restricted isometry property, that if the true signal $\mathbf{y}$ is sufficiently sparse, one can recover $\mathbf{y}$ from only a small number of random projections. However, for output coding, random projections do not specifically promote either discriminative or predictable codewords, and may not be the most effective encoding.

9.2.3 Coding with principal component analysis

Given the n training examples, principal label space transformation [TL10] uses the top d principal components in the label space as the encoding projections:

$$\{\mathbf{v}_k\}_{k=1}^d \leftarrow \text{top\_}d\text{\_principal\_components}(\mathbf{Y}) \tag{9.12}$$

which is solved by performing SVD on the label matrix $\mathbf{Y}$ and taking the top d right singular vectors. Therefore, the corresponding codeword $\mathbf{z} = (\mathbf{v}_1^T \mathbf{y}, \ldots, \mathbf{v}_d^T \mathbf{y})^T$ contains the top d coordinates of $\mathbf{y}$ in the principal component space.

Given predicted codeword $\hat{\mathbf{z}} = (\hat{m}_1(\mathbf{x}), \ldots, \hat{m}_d(\mathbf{x}))^T$ for a test sample $\mathbf{x}$, decoding is performed by projecting $\hat{\mathbf{z}}$ back to coordinates in the original label space and then rounding them element-wise to 0s and 1s:

$$\hat{\mathbf{y}} \leftarrow \text{round}(\mathbf{V}\hat{\mathbf{z}}) \tag{9.13}$$

Note that coding with principal components analysis can potentially produce *discriminative* codewords. The top d principal components provide a coordinate system that keeps as much sample variance as possible by any d-dimensional projections. As a result of variance maximization, the generated codewords for training samples tend to be spread out and far away from each other, although this does not exactly maximize the minimum codeword distance.

However, coding with principal components analysis does not promote the predictability of the codewords. Indeed, finding encoding projections as in eq. (9.12) is solely based on the label matrix $\mathbf{Y}$ and does not involve the input $\mathbf{X}$ at all. As a result, this encoding may generate codewords with large code distance but difficult to predict from the input.

9.2.4 Coding with canonical correlation analysis

Predictability for multi-label output codes is addressed in recent work [ZS11], where output projections are obtained by canonical correlation analysis. CCA tries to find an input projection $\mathbf{u} \in R^p$ in the feature space and an output projection $\mathbf{v} \in R^q$ in the label space such that the projected variables $\mathbf{u}^T\mathbf{x}$ and $\mathbf{v}^T\mathbf{y}$ are maximally correlated:

$$\underset{\mathbf{u}\in R^p, \mathbf{v}\in R^q}{\operatorname{argmax}} \quad \frac{\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v}}{\sqrt{(\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u})(\mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v})}} \tag{9.14}$$

This can be solved as a generalized eigenvalue problem, and the top d pairs of eigenvectors $\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^d$ contain the encoding projections $\{\mathbf{v}_k\}_{k=1}^d$.

The codeword in this method is defined as $\mathbf{z} = (y_1, \ldots, y_q, \mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T$. For a new sample $\mathbf{x}$, regression predictions for label projections are $\{\hat{m}_k(\mathbf{x})\}_{k=1}^d$ and classification predictions for original labels are $\{\hat{p}_{j0}(\mathbf{x}), \hat{p}_{j1}(\mathbf{x})\}_{j=1}^q$. Decoding is performed by maximizing a joint probability function (including d Gaussian potentials from regression and q Bernoulli potentials from classifiers), or equivalently minimizing the function [ZS11]:

$$\hat{\mathbf{y}} \leftarrow \underset{\mathbf{y}\in\{0,1\}^q}{\operatorname{argmin}} \frac{1}{2}\sum_{k=1}^{d} \frac{(\mathbf{v}_k^T\mathbf{y} - \hat{m}_k(\mathbf{x}))^2}{\hat{\sigma}_k^2} + \lambda \sum_{j=1}^{q} y_j \log\left(\frac{\hat{p}_{j0}(\mathbf{x})}{\hat{p}_{j1}(\mathbf{x})}\right) \tag{9.15}$$

where $\hat{\sigma}_k^2$ is the estimated mean squared error for regression model $\hat{m}_k$. Since the problem is defined on the label space $\mathbf{y} \in \{0, 1\}^q$, approximate inference such as mean-field approximation is used for optimization.

Coding with canonical correlation analysis improves the code predictability by choosing the projection directions that are maximally correlated with the input. However, this criterion does not optimize the discriminability of the generated codewords. In other words, codewords of different outputs may be close to each other, leading to inadequate error-correcting capabilities. Consequently, even a small amount of prediction error can significantly affect the decoding result.

9.3 Maximum Margin Output Coding

In this section we propose a max-margin output coding scheme where the encoding transform promotes both discriminability and predictability of the codewords.

9.3.1 A Max-Margin Formulation

As before, codewords are predicted using regression models:

$$\hat{\mathbf{M}}(\mathbf{x}) = (\hat{m}_1(\mathbf{x}), \ldots, \hat{m}_d(\mathbf{x}))^T \tag{9.16}$$

where each $\hat{m}_k()$ ($k = 1, \ldots, d$) is a univariate regression function for predicting $\mathbf{v}_k^T\mathbf{y}$, which is learned as in (9.4), and $\hat{\mathbf{M}}()$ is the corresponding multivariate regression function for the entire codeword $\mathbf{V}^T\mathbf{y}$.

For each sample i, the codeword $\mathbf{V}^T\mathbf{y}^{(i)}$ should be both predictable and discriminative. For predictability, we want $\hat{\mathbf{M}}(\mathbf{x}^{(i)})$ to be close to the correct codeword $\mathbf{V}^T\mathbf{y}^{(i)}$. For discriminability, we want the correct codeword $\mathbf{V}^T\mathbf{y}^{(i)}$ to have a large distance to any incorrect codeword $\mathbf{V}^T\mathbf{y}, \forall \mathbf{y} \neq \mathbf{y}^{(i)}$. In the context of prediction with output coding, it is even more strightforward and effective to require that the prediction $\hat{\mathbf{M}}(\mathbf{x}^{(i)})$ itself has a large distance to any incorrect codeword $\mathbf{V}^T\mathbf{y}, \forall \mathbf{y} \neq \mathbf{y}^{(i)}$.

Based on these goals, we propose a max-margin formulation on output projections $\mathbf{V}$:

$$\underset{\mathbf{V} \in R^{q \times d}, \{\xi_i\}_{i=1}^n}{\operatorname{argmin}} \frac{1}{2}||\mathbf{V}||_F^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{9.17}$$

$$\text{s.t. } ||\hat{\mathbf{M}}(\mathbf{x}^{(i)}) - \mathbf{V}^T\mathbf{y}^{(i)}||_2^2 + \triangle(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i \leq ||\hat{\mathbf{M}}(\mathbf{x}^{(i)}) - \mathbf{V}^T\mathbf{y}||_2^2, \quad \forall \mathbf{y} \in \{0,1\}^q, \forall i \tag{9.18}$$

$$\xi_i \geq 0, \quad \forall i$$

where $||\ ||_F$ is the Frobenius norm, $||\ ||_2$ is the ℓ_2 norm, C is a regularization parameter, $\triangle(\mathbf{y}^{(i)}, \mathbf{y})$ is the hamming distance between binary vectors, and $\{\xi_i\}_{i=1}^n$ are slack variables, each for a training sample. With the help from slack variables, constraint (9.18) requires that for any sample i, the prediction distance to the correct codeword, denoted by $||\hat{\mathbf{M}}(\mathbf{x}^{(i)}) - \mathbf{V}^T\mathbf{y}^{(i)}||_2^2$, must be smaller than the prediction distance to any codeword $||\hat{\mathbf{M}}(\mathbf{x}^{(i)}) - \mathbf{V}^T\mathbf{y}||_2^2$ by a margin of at least $\triangle(\mathbf{y}^{(i)}, \mathbf{y})$. Note that this encourages both small prediction distance to the correct codeword and large prediction distance to incorrect codewords, and hence promotes predictable and discriminative codes.

To simplify this formulation, we assume the regression functions $\hat{\mathbf{M}}(\mathbf{x}) = (\hat{m}_1(\mathbf{x}), \ldots, \hat{m}_d(\mathbf{x}))^T$ are *linear* and estimated by *least squares*. Then given training samples $(\mathbf{X}, \mathbf{Y})$, we define the $p \times q$ projection matrix $\mathbf{P}$:

$$\mathbf{P} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{9.19}$$

A small amount of regularization can be added to the diagonal of $\mathbf{X}^T\mathbf{X}$ for numerical stability. Using $\mathbf{P}$, the regression functions can be written in closed form:

$$\hat{m}_k(\mathbf{x}) = [\mathbf{P}\mathbf{v}_k]^T\mathbf{x}, \quad k = 1, 2, \ldots, d \tag{9.20}$$

and

$$\hat{M}(\mathbf{x}) = [\mathbf{P}\mathbf{V}]^T\mathbf{x} \tag{9.21}$$

Plugging eq. (9.21) into problem (9.17), we have the following max-margin formulation that is completely defined on projections $\mathbf{V}$ and slack variables $\{\xi_i\}_{i=1}^n$:

$$\underset{\mathbf{V} \in R^{q \times d}, \{\xi_i\}_{i=1}^n}{\operatorname{argmin}} \frac{1}{2}||\mathbf{V}||_F^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{9.22}$$

$$\text{s.t. } ||\mathbf{V}^T(\mathbf{P}^T\mathbf{x}^{(i)} - \mathbf{y}^{(i)})||_2^2 + \triangle(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i \leq ||\mathbf{V}^T(\mathbf{P}^T\mathbf{x}^{(i)} - \mathbf{y})||_2^2, \quad \forall \mathbf{y} \in \{0,1\}^q, \forall i$$

$$\xi_i \geq 0, \quad \forall i$$

9.3.2 Metric Learning Formulation

Problem (9.22) is a quadratic program with quadratic constraints, and we first convert it to a metric learning problem. Define $q \times q$ matrix $\mathbf{Q}$:

$$\mathbf{Q} = \mathbf{V}\mathbf{V}^T \tag{9.23}$$

which is the Mahalanobis distance metric induced by $\mathbf{V}$. Also, define a set of new feature vectors:

$$\phi_{i\mathbf{y}} = \mathbf{P}^T\mathbf{x}^{(i)} - \mathbf{y}, \quad \forall \mathbf{y} \in \{0,1\}^q, \forall i \tag{9.24}$$

Now we formulate the metric learning problem as:

$$\begin{aligned} \underset{\mathbf{Q}\in S_q^+, \{\xi_i\}_{i=1}^n}{\text{argmin}} \quad & \frac{1}{2}\text{trace}(\mathbf{Q}) + \frac{C}{n}\sum_{i=1}^n \xi_i \\ \text{s.t.} \quad & \phi_{i\mathbf{y}^{(i)}}^T \mathbf{Q} \phi_{i\mathbf{y}^{(i)}} + \triangle(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i \leq \phi_{i\mathbf{y}}^T \mathbf{Q} \phi_{i\mathbf{y}}, \quad \forall \mathbf{y} \in \{0,1\}^q, \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned} \tag{9.25}$$

where $\mathbf{Q} \in S_q^+$ is positive semidefinite. Notice that both the objective function and the constraints are now linear in $\mathbf{Q}$ and $\{\xi_i\}_{i=1}^n$.

We briefly show the equivalence between problem (9.22) and (9.25) as follows. For any feasible solution $\mathbf{V}$ to (9.22), we can define $\mathbf{Q} = \mathbf{V}\mathbf{V}^T \in S_q^+$. Also, for any feasible solution $\mathbf{Q}$ to (9.25), since $\mathbf{Q}$ is positive semidefinite and thus has no negative eigenvalue, we can define $\mathbf{V}$ as:

$$\mathbf{V} = \mathbf{Q}^{\frac{1}{2}} = \mathbf{U}\mathbf{D}^{\frac{1}{2}} \tag{9.26}$$

where the $q \times q$ matrix $\mathbf{U}$ contains (as columns) the q eigenvectors of $\mathbf{Q}$, and $\mathbf{D}$ is the diagonal matrix of eigenvalues. Given this one-to-one mapping between $\mathbf{V}$ and $\mathbf{Q}$, we have $\text{trace}(\mathbf{Q}) = ||\mathbf{V}||_F^2$ and $\phi_{i\mathbf{y}}^T \mathbf{Q} \phi_{i\mathbf{y}} = ||\mathbf{V}^T(\mathbf{P}^T\mathbf{x}^{(i)} - \mathbf{y})||_2^2$. Therefore, any feasible (or optimal) solution to (9.25) gives a feasible (or optimal) solution to (9.22), and vice versa.

9.3.3 Incorporating Original Labels and Their Classifiers

As shown in eq. (9.3), the codeword can also include q original labels, i.e., $\mathbf{z} = (y_1, \ldots, y_q, \mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T$. Classifiers $\{\hat{p}_j\}_{j=1}^q$ can be learned to predict original labels as in (9.5), and the decoding algorithm can make use of both regression and classifier outputs, e.g., as in eq. (9.15). In this case, the encoding projection should also be aware of the original labels $(y_1, \ldots, y_q)$ in the codeword, so that the projection part $(\mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})$ can provide *complementary* information.

To adapt our max-margin formulation (9.25) to this new information, we assume that classifiers $\{\hat{p}_j\}_{j=1}^q$ have already been learned, and thus for each sample $\mathbf{x}$ we know the classifier output

$\hat{p}_{j0}(\mathbf{x}) = P(y_j = 0|\mathbf{x})$ and $\hat{p}_{j1}(\mathbf{x}) = P(y_j = 1|\mathbf{x})$. We have the new formulation:

$$\begin{aligned} &\underset{\mathbf{Q}\in S_q^+, \{\xi_i\}_{i=1}^n}{\text{argmin}} \frac{1}{2}\text{trace}(\mathbf{Q}) + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } \phi_{i\mathbf{y}^{(i)}}^T \mathbf{Q}\phi_{i\mathbf{y}^{(i)}} - \log P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) + \triangle(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i \leq \phi_{i\mathbf{y}}^T \mathbf{Q}\phi_{i\mathbf{y}} - \log P(\mathbf{y}|\mathbf{x}^{(i)}), \quad \forall \mathbf{y} \in \{0,1\}^q, \forall i \\ &\quad \xi_i \geq 0, \quad \forall i \end{aligned} \tag{9.27}$$

where

$$P(\mathbf{y}|\mathbf{x}^{(i)}) = \prod_{j=1}^{q} P(y_j|\mathbf{x}^{(i)}) = \prod_{j=1}^{q} \hat{p}_{jy_j}(\mathbf{x}^{(i)}) \tag{9.28}$$

is the joint probability of label vector $\mathbf{y} = (y_1, \ldots, y_q)$ on sample $\mathbf{x}^{(i)}$ from classifiers $\{\hat{p}_j\}_{j=1}^q$.

In this new formation (9.27), $\phi_{i\mathbf{y}}^T \mathbf{Q}\phi_{i\mathbf{y}}$ is extended into $\phi_{i\mathbf{y}}^T \mathbf{Q}\phi_{i\mathbf{y}} - \log P(\mathbf{y}|\mathbf{x}^{(i)})$. Recall that $\phi_{i\mathbf{y}}^T \mathbf{Q}\phi_{i\mathbf{y}}$ is equivalent to $||\mathbf{V}^T(\mathbf{P}^T\mathbf{x}^{(i)} - \mathbf{y})||_2^2$ in (9.22), which is the distance between the regression prediction on the ith sample and the encoding of the label vector $\mathbf{y}$. We expect that the correct label vector $\mathbf{y}^{(i)}$ should lead to lower values on this term than other $\mathbf{y}$. Similarly, $\log P(\mathbf{y}|\mathbf{x}^{(i)})$ is the log-probability of $\mathbf{y}$ predicted by classifiers on sample i, and we expect that $\mathbf{y}^{(i)}$ should give higher values on this term than other label vectors $\mathbf{y}$. As a result, we now use the combined term $\phi_{i\mathbf{y}}^T \mathbf{Q}\phi_{i\mathbf{y}} - \log P(\mathbf{y}|\mathbf{x}^{(i)})$ to measure the margin between correct and incorrect outputs. The main outcome of this new formulation is that distance metric $\mathbf{Q}$ will focus on the constraints where $-\log P(\mathbf{y}|\mathbf{x}^{(i)})$ alone is not strong enough to ensure the margin. In other words, the output coding concentrates on the cases where classifiers $\{\hat{p}_j\}_{j=1}^q$ alone tend to misclassify.

9.3.4 Cutting Plane Method with Overgenerating

In this section we consider how to solve problem (9.27). This problem involves an exponentially large number of constraints due to the combinatorial nature of the label space $\{0,1\}^q$. As studied in structured prediction [THJA04, TGK03], problem (9.27) could be solved efficiently, e.g., by the cutting-plane method, if a computationally tractable separation oracle exists to determine which of the exponentially many constraints is most violated [THJA04]. However, without a specific structure (e.g., a chain or a tree) in the label space to enable efficient inference, the separation oracle for problem (9.27) is computationally intractable.

To address this issue, we use overgenerating (i.e., relaxation) [FJ08] with the cutting plane method. To use the overgenerating technique, we need to relax $\forall \mathbf{y} \in \{0,1\}^q$ in the constraint of (9.27) to a continuous domain, e.g., $\forall \mathbf{y} \in [0,1]^q$. However, $\triangle(\mathbf{y}^{(i)}, \mathbf{y})$ and $\log P(\mathbf{y}|\mathbf{x}^{(i)})$ in (9.27) are only defined on $\mathbf{y} \in \{0,1\}^q$. To handle this, we redefine $\triangle(\mathbf{y}^{(i)}, \mathbf{y})$ as

$$\widetilde{\triangle}(\mathbf{y}^{(i)}, \mathbf{y}) = ||\mathbf{y}^{(i)} - \mathbf{y}||_1 = \sum_{j=1}^{q} |y_j^{(i)} - y_j| \tag{9.29}$$

Then noticing $\log P(\mathbf{y}|\mathbf{x}^{(i)}) = \sum_{j=1}^{q} \log P(y_j|\mathbf{x}^{(i)})$, we redefine:

$$\log \widetilde{P}(\mathbf{y}|\mathbf{x}^{(i)}) = \sum_{j=1}^{q} \log \widetilde{P}(y_j|\mathbf{x}^{(i)}) \tag{9.30}$$

where each $\log \widetilde{P}(y_j|\mathbf{x}^{(i)})$ is the linear interpolation of $\log P(y_j = 0|\mathbf{x}^{(i)})$ and $\log P(y_j = 1|\mathbf{x}^{(i)})$.

Using (9.29) and (9.30), the new relaxed problem is:

$$\begin{aligned} &\underset{\mathbf{Q} \in S_q^+, \{\xi_i\}_{i=1}^n}{\operatorname{argmin}} \frac{1}{2}\operatorname{trace}(\mathbf{Q}) + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ &\text{s.t. } \phi_{i\mathbf{y}^{(i)}}^T \mathbf{Q} \phi_{i\mathbf{y}^{(i)}} - \log \widetilde{P}(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) + \widetilde{\triangle}(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i \leq \phi_{i\mathbf{y}}^T \mathbf{Q} \phi_{i\mathbf{y}} - \log \widetilde{P}(\mathbf{y}|\mathbf{x}^{(i)}), \quad \forall \mathbf{y} \in [0,1]^q, \forall i \\ &\quad \xi_i \geq 0, \quad \forall i \end{aligned} \tag{9.31}$$

where $\forall \mathbf{y} \in \{0,1\}^q$ in (9.27) is relaxed to $\forall \mathbf{y} \in [0,1]^q$.

This new problem can be solved by the cutting plane method, because the separation oracle (i.e., finding the most violated constraint for each sample i) is:

$$\underset{\mathbf{y} \in [0,1]^q}{\operatorname{argmin}} \ \phi_{i\mathbf{y}}^T \mathbf{Q} \phi_{i\mathbf{y}} - \log \widetilde{P}(\mathbf{y}|\mathbf{x}^{(i)}) - \widetilde{\triangle}(\mathbf{y}^{(i)}, \mathbf{y}) \tag{9.32}$$

where $\log \widetilde{P}(\mathbf{y}|\mathbf{x}^{(i)})$ and $\widetilde{\triangle}(\mathbf{y}^{(i)}, \mathbf{y})$ are linear in $\mathbf{y}$, and $\phi_{i\mathbf{y}}^T \mathbf{Q} \phi_{i\mathbf{y}}$ is quadratic in $\mathbf{y}$ given $\phi_{i\mathbf{y}}$ defined as (9.24). So (9.32) is a simple box-constrained quadratic program.

9.3.5 Encoding and Decoding

After solving $\mathbf{Q}$ in (9.31), encoding projections are obtained as (9.26), and one can choose d, the number of projections, by keeping only the first d columns of $\mathbf{V}$ in (9.26) for any $d \leq q$. The codeword as in (9.3) includes original labels, and decoding is performed as (9.15).

9.4 Empirical Study

Data. We perform experiments on three real-world data sets[1]: an image data set (*Scene*), a text data set (*Medical*) and a music data set (*Emotions*). *Scene* is an image collection for outdoor scene recognition. Each image is represented by 294 dimensional color features and labeled as: beach, sunset, fall foliage, field, mountain and urban. *Emotions* is a music classification problem. Each song is represented by 72 rhythmic and timbre features, and tagged with six emotions: amazed, happy, relaxed, quiet, sad and angry. *Medical* is a clinical text collection, where each document is represented by 1449 words and labeled with ICD-9-CM codes. Many labels in *Medical* are rare, so we select the 10 most common labels to study.

[1] http://mulan.sourceforge.net/

Methods. We compare the proposed max-margin output coding to several recently proposed multi-label output codes as well as a number of other multi-label classification methods:

- *Binary relevance (BR)*. This baseline method learns to classify each label independently. It is also called one-vs-all decomposition.
- *Coding with compressed sensing (CodingCS)* [HKLZ09]. As reviewed in Section 9.2.2, this method uses random projections for encoding and sparse approximation for decoding. Specifically, we use CoSaMP [NT08] for decoding.
- *Coding with PCA (CodingPCA)* [TL10]. As reviewed in Section 9.2.3, this method uses principal components for encoding, and PCA reconstruction and rounding for decoding.
- *Coding with PCA-Redundant (CodingPCA-R)*. CodingPCA does not include original labels into the codeword. We also try this option to produce more redundancy as in eq. (9.3). Decoding follows eq. (9.15).
- *Coding with CCA (CodingCCA)* [ZS11]. As reviewed in Section 9.2.4, this method uses CCA for encoding. Decoding follows eq. (9.15).
- *Calibrated label ranking (CLR)* [FHMB08]. This method combines one-vs-one and one-vs-all classifiers for multi-label classification. It can also be viewed as an output coding method.
- *Multi-label learning by exploiting label dependency (LEAD)* [ZZ10]. This method learns a Bayes network on labels and use it to capture label dependency in multi-label classification.
- *Max-Margin coding (MaxMargin)*. Our max-margin coding formulation where encoding is obtained by solving (9.31) and (9.26). Decoding follows eq. (9.15).

Evaluation measures. We consider three evaluation measures for multi-label classification:

- Subset accuracy: rates of correctly classifying *all* the labels. It is obvious that achieving high subset accuracy is difficult.
- Macro-averaged F1 score: calculate the F1 score for each label and take the average over labels. F1 score is popular since the distribution of positive examples and negative examples for a label is usually very imbalanced.
- Micro-averaged F1 score: aggregate true positives, true negatives, false positives and false negatives over labels, and then calculate an overall F1 score.

Experimental settings. On each data set, we perform 30 random runs and report means and standard errors of each evaluation measure. The number of training samples in each run is 300.

For *CodingCS*, the number of projections d is set to 100 to provide highly redundant codewords. For *CodingPCA*, *CodingPCA-R*, *CodingCCA* and *MaxMargin*, the number of output projections is set to the maximum possible number: the number of original labels.

For all methods, base regression models are ridge regression and base classifiers are ℓ_2-penalized logistic regression, and their regularization parameters are chosen by cross validation. For LEAD, the Bayes net is learned using the score-based searching algorithm in the Bayesian Net Toolbox[2]. For decoding that follows (9.15), λ is set to 1, i.e., classifiers and regression models are equally

[2]http://code.google.com/p/bnt/

Table 9.1: Subset accuracy on Scene data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(6)	0.4238	0.0040
CodingCS(100)	0.3821	0.0047
CodingPCA(6)	0.3691	0.0075
CodingPCA-R(12)	0.4305	0.0046
CodingCCA(12)	0.4928	0.0090
CLR(21)	0.4218	0.0034
LEAD(6)	0.4547	0.0048
MaxMargin(12)	**0.5448**	0.0073

Table 9.2: Macro-F1 score on Scene data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(6)	0.6209	0.0029
CodingCS(100)	0.5234	0.0050
CodingPCA(6)	0.5279	0.0096
CodingPCA-R(12)	0.6049	0.0039
CodingCCA(12)	0.6312	0.0038
CLR(21)	0.6238	0.0026
LEAD(6)	0.5958	0.0042
MaxMargin(12)	**0.6462**	0.0046

Table 9.3: Micro-F1 score on Scene data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(6)	0.6117	0.0030
CodingCS(100)	0.5345	0.0041
CodingPCA(6)	0.5404	0.0076
CodingPCA-R(12)	0.6014	0.0032
CodingCCA(12)	0.6251	0.0035
CLR(21)	0.6163	0.0024
LEAD(6)	0.6002	0.0036
MaxMargin(12)	**0.6382**	0.0047

weighted in decoding. The parameter C in (9.31) is set to 10^6. Most methods need to round their final predictions into 0/1 assignments (e.g., from a probability forecast or a relaxed solution to the label assignment), and in these cases we use 0.5 as the threshold without further optimization.

Empirical Results. Results for the Scene data set are shown in Table 9.1 - Table 9.3; results for Medical are shown in Table 9.4 - Table 9.6; results for Emotions are shown in Table 9.7 - Table 9.9.

Table 9.4: Subset accuracy on Medical data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(10)	0.7673	0.0039
CodingCS(100)	0.7071	0.0019
CodingPCA(10)	0.7541	0.0041
CodingPCA-R(20)	0.7803	0.0031
CodingCCA(20)	0.7824	0.0029
CLR(55)	0.7632	0.0037
LEAD(10)	0.7718	0.0038
MaxMargin(20)	**0.7930**	0.0042

Table 9.5: Macro-F1 score on Medical data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(10)	0.8626	0.0029
CodingCS(100)	0.7987	0.0026
CodingPCA(10)	0.8523	0.0032
CodingPCA-R(20)	0.8697	0.0021
CodingCCA(20)	0.8703	0.0020
CLR(55)	0.8556	0.0029
LEAD(10)	0.8550	0.0035
MaxMargin(20)	**0.8710**	0.0039

Table 9.6: Micro-F1 score on Medical data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(10)	0.8785	0.0022
CodingCS(100)	0.8333	0.0013
CodingPCA(10)	0.8780	0.0020
CodingPCA-R(20)	0.8853	0.0018
CodingCCA(20)	0.8867	0.0018
CLR(55)	0.8757	0.0022
LEAD(10)	0.8736	0.0023
MaxMargin(20)	**0.8919**	0.0025

Each table contains one evaluation measure. From the results we can see:

- BR provides a solid baseline with good performance.
- CodingCS generally underperforms, and encoding with random projections is not effective.
- CodingPCA-R outperforms CodingPCA as CodingPCA-R uses more redundant codewords.
- LEAD's performance is not stable across data sets. Structure learning for Bayesian Networks is

Table 9.7: Subset accuracy on Emotions data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(6)	0.2264	0.0031
CodingCS(100)	0.1665	0.0035
CodingPCA(6)	0.2198	0.0027
CodingPCA-R(12)	0.2601	0.0024
CodingCCA(12)	0.3005	0.0040
CLR(21)	0.2266	0.0037
LEAD(6)	0.1559	0.0035
MaxMargin(12)	**0.3114**	0.0042

Table 9.8: Macro-F1 score on Emotions data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(6)	0.6248	0.0027
CodingCS(100)	0.4742	0.0043
CodingPCA(6)	0.5592	0.0041
CodingPCA-R(12)	0.6367	0.0028
CodingCCA(12)	0.6539	0.0027
CLR(21)	0.6197	0.0029
LEAD(6)	0.4512	0.0046
MaxMargin(12)	**0.6609**	0.0029

Table 9.9: Micro-F1 score on Emotions data set: mean and standard error over 30 random runs

Method (#Base Models)	Mean	Standard Error
BR(6)	0.6363	0.0024
CodingCS(100)	0.5260	0.0029
CodingPCA(6)	0.5957	0.0032
CodingPCA-R(12)	0.6482	0.0026
CodingCCA(12)	0.6633	0.0025
CLR(21)	0.6331	0.0025
LEAD(6)	0.5142	0.0038
MaxMargin(12)	**0.6688**	0.0027

still a very challenging problem.

- CLR performs comparably to BR, despite the fact that it is one of the most redundant methods in terms of the number of base models used.
- CodingPCA-R, CodingCCA and MaxMargin are most successful. Their codewords include both label projections and original labels, and their decodings combine both regression and classification outputs.

- MaxMargin outperforms CodingCCA and CodingPCA, because max-margin encoding promotes both code discriminability and code predictability.
- CodingCCA outperforms CodingPCA-R, showing the importance of predictable codewords.

9.5 Related Work

The proposed method in this chapter follows the direction of multi-label output coding [HKLZ09, TL10, ZS11] and is motivated by the recent success of coding with PCA [TL10] and CCA [ZS11] and their connections to code distance and code predictability. Our max-margin formulation is converted into a metric learning problem, as in [WBS06], but with a metric defined for the label space and an exponential number of constraints caused by label combinations. The optimization technique developed for structured prediction [THJA04, TGK03], more specically the cutting plane method with overgenerating [FJ08], is used to solve our metric learning problem.

9.6 Conclusion

Discriminability and predictability are both important for output codes. In this chapter we propose a max-margin formulation for multi-label output coding, which promotes both discriminative and predictable codes. We convert this formulation into a metric learning problem in the label space, and combine overgenerating with the cutting plane method for optimization. Our method outperforms many existing methods on multi-label image, text and music data sets.

Chapter 10

Conclusion, Discussion and Future Directions

10.1 Conclusion

With the goal of learning with limited supervision, this thesis studies the representation, discovery and incorporation of extra input and output information in learning.

Information about the input space can be gained from unlabeled data, related tasks and domain knowledge, and then can be encoded into the learning process by regularization.

- In this direction we first design a semi-supervised learning method for text classification, where the covariance and correlation structure of the input space can be inferred from even seemingly irrelevant unlabeled text and used for regularization in any text classification task.
- To generalize this idea to multi-task learning, we propose a matrix-normal penalty to compactly encode the covariance structure of the joint input space of multiple tasks, which enables us to simultaneously infer and encode both task relations and feature structures.
- To capture input information that is more general than the covariance and correlation structure, we propose learning compressible models and study a class of regularization penalties on model compressibility, e.g., local smoothness and frequency-domain compacted energy.
- To encode input information conveyed by a dimension reduction while controlling the risk of losing predictive information, we propose projection penalties: we still search the model in the full model space but penalize the projection distance to the model subspace indicated by the reduction.

Information about the output space can be incorporated into learning by error-correcting output coding, and we design several coding schemes to improve multi-label classification.

- We first propose an improved pairwise coding for multi-label classification, which, inspired by composite likelihoods, encodes pairwise label density estimations (as opposed to pairwise label comparisons) and decodes using approximate inference on the composite likelihood.

- We then study problem-dependent output coding, where the encoding is learned from data. We propose multi-label output coding via canonical correlation analysis: the most predictable directions in the label space are extracted by canonical correlation analysis, and the code is designed to include these predictable label directions as redundant information to correct prediction errors.
- We argue that both discriminability and predictability are critical properties for multi-label output codes, and propose a max-margin formulation that simultaneously promotes discriminative and predictable codewords. This formulation is converted into a metric learning problem in the label space and solved by the cutting plane method combined with overgenerating techniques.

10.2 Discussion

In this section we discuss a few issues regarding the input and output coding methods designed in this thesis. We organize this section by first asking the questions and then providing discussions and answers.

Q: What are the connections between different input coding methods?

In Part I of this thesis we have presented four methods for encoding input information. All these methods use regularization as the basic tool to incorporate extra input information into the learning process. Chapter 3 introduces the use of a covariance operator in regularization, where the covariance structure is inferred from unlabeled data. Chapter 4 extends the idea of using a covariance operator in regularization to multi-task learning settings. A benefit of this extension is that we no longer need unlabeled data to infer the covariance structure, since multiple tasks serve as information sources for each other. But a challenge we need to face is the increased dimensionality of the joint model space of multiple tasks (and thus the difficulty of estimating the covariance structure), for which we propose a matrix-normal penalty as a compact representation of the joint covariance operator for multiple tasks. Chapter 5 shows that a compression operator can be used in regularization to encode input information, which generalizes the covariance operator in Chapter 3 and Chapter 4 to a much richer class of regularization operators. However, fully specifying a compression operator for model coefficients requires strong domain knowledge about the input space, which might not be available in many real-world problems. On the other hand, dimension reduction techniques are widely used in many learning problems. A dimension reduction can be viewed as a compression of the input space, so can we gain some insight about how to regularize the model coefficients from a dimension reduction? Chapter 6 answer this question by presenting the projection penalty, a new regularization operator that only requires a dimension reduction of the input space instead of a fully specified compression operator to control model complexity wisely.

Q: Which input coding method should I use and when is the method applicable?

The four methods presented in Part I of this thesis differ in that they discover the input information from different sources: unlabeled data, related tasks, domain knowledge, and dimension

reduction. In this sense, the choice of methods depends on the external source of information we have. Specifically, learning with the semantic correlation of words (Chapter 3) is applicable to learning problems where the bag-of-words representation is appropriate and unlabeled data is available; multi-task matrix-normal regularization (Chapter 4) is applicable to learning multiple similar or related tasks; learning compressible models (Chapter 5) is designed for learning problems where strong domain knowledge about the input space is available to specify a compression operator; the projection penalty (Chapter 6) is applicable when we have a certain dimension reduction that conveys useful information about the input space.

Q: What are the connections between different output coding methods?

In Part II of this thesis we have designed three output coding schemes to encode label dependency in multi-label classification. Chapter 7 introduces a modification of the popular pairwise coding approach based on a composite likelihood view of multi-label classification. Noticing that the new coding scheme in Chapter 7 is still problem-independent (i.e., always based on pairwise decomposition regardless of the problem), Chapter 8 and Chapter 9 present novel problem-dependent coding, where codes are adaptively learned from data in the form of label projections. All three output codes can be unified using the composite likelihood view of multi-label classification: encoding corresponds to decomposing the joint likelihood function (and hence the joint probability of labels given the input) into the product of simple component likelihoods, and decoding corresponds to approximate inference given the estimation of the composite likelihood. The difference between the three methods is the choice of component likelihoods, which are pre-defined pairwise label densities in Chapter 7, and Gaussian densities on adaptively learned label projections in Chapter 8 and Chapter 9. The difference between Chapter 8 and Chapter 9 is the way we find the label projections that can provide good error-correcting capability in output codes.

Q: Which output coding method should I use and when is the method applicable?

Output coding in multi-label classification is usually designed to capture certain label dependencies, so the choice of output code depends on the type of label dependency we want to capture. For example, pairwise coding only captures pairwise label dependency, using either label comparisons (as in pairwise label ranking and calibrated label ranking) or pairwise label densities (as in the composite pairwise code in Chapter 7). The CCA-based codes (Chapter 8) and max-margin codes (Chapter 9), on the other hand, capture dependencies among all labels using linear projections of labels. It is also possible to capture nonlinear relations among labels by extending CCA-based coding to include nonlinear projections of labels, e.g., coding based on kernel CCA, and this will be the subject of future work.

In our empirical studies, we find that CCA-based coding and max-margin coding achieve the best prediction performance. Their codes contain label projections that are adaptively learned from data and therefore may be able to represent certain label dependencies that are not well captured by predefined codes (such as pairwise coding). Max-margin coding usually achieves better prediction performance than CCA-based coding, but it is also computationally more expensive to learn than CCA-based coding: max-margin coding needs to solve a metric learning formulation with an

exponential number of constraints, as opposed to a single generalized eigenvalue problem in CCA.

Q: Will the relative performance of different multi-label output codes change when we alter the number of training examples?

Different output coding methods have different numbers of parameters to learn and hence different sample complexities. The 1-vs-all method (i.e., binary relevance) only includes original labels into the codeword and each label is classified independently, i.e., the learning is very simple as we only need one classifier learned independently for each label. The pairwise coding approaches (pairwise label ranking, calibrated label ranking and our proposed composite pairwise coding in Chapter 7) all have more parameters to estimate, as the length of the output code (and hence the number of subproblems to learn) in these approaches is quadratic in the number of original labels. For CCA-based coding and max-margin coding in Chapter 8 and 9, the number of subproblems to learn is at most twice the number of original labels (i.e., q original labels plus at most q label projections), although the label projections are not predefined and thus also need to be learned.

In practice one expects that simple methods perform well when the amount of training data is small. The 1-vs-all method does not capture label dependency, but when the number of training samples is small, it might perform relatively well compared to some of the more sophisticated coding methods. In our empirical studies, the 1-vs-all method has outperformed two pairwise coding methods (pairwise label ranking and calibrated label ranking) on some data sets (see, e.g., Chapter 7). Max-margin coding and CCA-based coding always deliver the best performance in our experiments, but it is possible that, given an *extremely* small amount of training data, simpler methods (such as the 1-vs-all method) may be a serious contender. When the number of training examples becomes large, on the other hand, sample complexity may become less critical and the performance of different methods may depend more on whether they can represent the true label dependency structure well. In this case, problem-dependent coding schemes such as CCA-based coding and max-margin coding may show more advantages.

Q: How many label projections should I use in CCA-based and max-margin output coding?

For CCA-based and max-margin coding, we use the maximum possible number of label projections (which is limited by the number of labels and features and in many cases equals the number of labels) to avoid parameter tuning. Our observation is that using the maximum number of projections generally leads to very good performance, although it may not give the best performance on certain data sets. In this sense, one can use cross-validation to choose the best number of projections to use. Using the maximum number of label projections maximizes the redundancy of the output code. On the other hand, as we use more label projections, the redundancy of the code increases but the quality of the code might decrease. For example, the first label projection found by CCA is the most predictable direction, and successive projections found by CCA are usually less predictable than the first one.

10.3 Future Work

Future work on input and output coding can extend this thesis in the following directions.

- Multiple sources of input information: in Part I we study encoding input information from different sources such as unlabeled data, related tasks, domain knowledge and dimension reduction. In certain situations we may have input information available from multiple sources, e.g., learning multiple tasks where for each task we have a dimension reduction. In this case, it is possible to exploit input information from both dimension reduction and related tasks.
- More redundant output coding: in Part II we design several output codes for multi-label classification. All these codes, however, can provide only a limited amount of redundancy: either the code is predefined with a fixed length, or the code length is limited by the number of original labels. In the future we will study output coding with high redundancy, which can be potentially implemented by removing the orthogonality constraints on label projections and using a randomized procedure to generate a large number of label projections with certain variation.
- Combining input and output coding: Part I and Part II of this thesis study input and output coding separately. Methods proposed in these two parts, however, are compatible with each other, and thus we can perform both input and output coding on certain problems. For example, in multi-label text classification, label dependency can be utilized by one of our output codes, and the semantic correlation structure of the input space can also be incorporated by regularization.

Bibliography

[AEP06] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS*, 2006.

[AEP07] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *NIPS 19*. MIT Press, 2007.

[AMPY07] Andreas Argyriou, Charles A. Micchelli, Massimiliano Pontil, and Yiming Ying. A spectral regularization framework for multi-task structure learning. In *NIPS*, 2007.

[ASS01] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, 2001.

[AZ05] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

[B+04] B. Blankertz et al. The BCI Competition 2003: Progress and Perspectives in Detection and Discrimination of EEG Single Trails. *IEEE Trans. Biomedical Engineering*, 51(6):1044–1051, 2004.

[BA00] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Comput.*, 12(10):2385–2404, 2000.

[Bax95] J. Baxter. Learning Internal Representations. In *COLT*, pages 311–320, 1995.

[BC01] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26, 2001.

[BCNV08] R. G. Baraniuk, E. J. Candes, R. Nowak, and M. Vetterli. Compressive Sampling (Special Issue). *IEEE Signal Processing Magazine*, 25:12–101, 2008.

[BCW08] E. Bonilla, K. M. Chai, and C. Williams. Multi-task gaussian process prediction. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS*, pages 153–160. 2008.

[Bes74] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *J. Roy. Statist. Soc. Ser. B*, 36:192–236, 1974.

[BFSO84] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1 edition, 1984.

[BGd08] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach.*

Learn. Res., 9:485–516, 2008.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[BH03] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[BM98] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, 1998.

[BNJ03] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *JMLR*, 3:993–1022, 2003.

[BNS06] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.

[BR08] H. D. Bondell and B. J. Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64:115–123, 2008.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[BS01] Leo Breiman and E. Schapire. Random forests. *Machine Learning*, pages 5–32, 2001.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[BV98] P. J. Brown and M. Vannucci. Multivariate Bayesian Variable Selection and Prediction. *Journal of the Royal Statistical Soceity, Series B*, 60(3):627–641, 1998.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[Can06] E. J. Candes. Compressive Sampling. In *Proceedings of International Congress of Mathematicians*, 2006.

[Car97] R. Caruana. Multitask Learning. *Machine Learning*, 28:41–75, 1997.

[CF07] Daniel J. Costello and G. David Forney. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150–1177, 2007.

[CHHZ06] D. Cai, X. He, J. Han, and H. Zhang. Orthogonal laplacianfaces for face recognition. *IEEE Transactions on Image Processing*, 15(11):3608–3614, 2006.

[CJK04] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6:1–6, 2004.

[CJS] Robert Calderbank, Sina Jafarpour, and Robert Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. Preprint, 2009.

[Cox72] D. R. Cox. The analysis of multivariate binary data. *J. Roy. Statist. Soc. Ser. C*, 21:113–120, 1972.

[Cox04] D. R. Cox. A note on pseudolikelihood constructed from marginal densities. *Biometrika*, 92:729–737, 2004.

[CR75] D. Cox and N. Reid. Partial likelihood. *Biometrika*, 62:269–276, 1975.

[CS02] Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. *Mach. Learn.*, 47(2-3):201–233, 2002.

[CSE00] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 Still Image Coding System: An Overview. *IEEE Trans. Consumer Electronics*, 46(4):1103–1127, 2000.

[CSZ06] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-supervised Learning*. The MIT Press, 2006.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[CT94] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[CTLY09] J. Chen, L. Tang, J. Liu, and J. Ye. A Convex Formulation for Learning Shared Structures from Multiple Tasks. In *ICML*, 2009.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.

[CW08] E. J. Candes and M. B. Wakin. An Introduction to Compressive Sampling. *IEEE Signal Processing Magazine*, 25:21–30, 2008.

[Daw81] A. P. Dawid. Some matrix-variate distribution theory: Notational considerations and a bayesian application. *Biometrika*, 68(1):265–274, 1981.

[DB95] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[DCH10] K. Dembczyński, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 279–286. Omnipress, June 2010.

[Dem72] A. P. Dempster. Covariance selection. *Biometrics*, 1972.

[DGK08] J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse gaussians. In *Proceedings of the Twenty-fourth Conference on Uncertainty in AI (UAI)*, 2008.

[DMS04] Ofer Dekel, Christopher Manning, and Yoram Singer. Log-linear models for label ranking. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. 2004.

[Don06] D. L. Donoho. Compressed Sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006.

[Dut99] P. Dutilleul. The MLE Algorithm for the Matrix Normal Distribution. *J. Statist.*

Comput. Simul., 64:105–123, 1999.

[Efr79] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7, 1979.

[EPR10] Sergio Escalera, Oriol Pujol, and Petia Radeva. On the decoding process in ternary error-correcting output codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):120–134, 2010.

[FHMB08] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, June 2008.

[FHT07] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007.

[FJ08] Thomas Finley and Thorsten Joachims. Training structural svms when exact inference is intractable. In *ICML*, pages 304–311, 2008.

[FL01] C. Fyfe and P. L. Lai. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10:365–374, 2001.

[FS96] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[Fur02] J. Furnkranz. Round robin classification. *J. Mach. Learn. Res.*, 2:721–747, 2002.

[FV06] S. Fieuws and G. Verbeke. Pairwise fitting of mixed models for the joint modeling of multivariate longitudinal profiles. *Biometrics*, 62(2):424–31, 2006.

[GM05] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 195–200, New York, NY, USA, 2005. ACM.

[GN99] A. K. Gupta and D. K. Nagar. *Matrix Variate Distributions*. Chapman Hall, 1999.

[GV10] T. Gärtner and Shankar Vembu. Label ranking algorithms: A survey. In Eyke Hüllermeier Johannes Fürnkranz, editor, *Preference Learning*. Springer–Verlag, 2010.

[HFCB08] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916, 2008.

[HKLZ09] Daniel Hsu, Sham Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *NIPS*, pages 772–780. 2009.

[Hof99] T. Hofmann. Probabilistic latent semantic analysis. In *UAI*, 1999.

[Hot35] H. Hotelling. The most predictable criterion. *Journal of Educational Psychology*, 26:139–142, 1935.

[Hot36] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[HSSt04] David R. Hardoon, Sandor R. Szedmak, and John R. Shawe-taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.*,

16(12):2639–2664, 2004.

[HST09] D. R. Hardoon and J. Shawe-Taylor. Sparse canonical correlation analysis. http://arxiv.org/abs/0908.2724, 2009.

[HT97] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *NIPS '97*, 1997.

[HTF01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[HV08] N. Hjort and C. Varin. Ml, pl, ql in markov chain models. *Scandinavian Journal of Statistics*, 35:64–82, 2008.

[HVV10] B. Hariharan, S.V.N. Vishwanathan, and M. Varma. Large scale max-margin multi-label classification with priors. In *ICML*, 2010.

[HZM09] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. In *ICML '09*, 2009.

[JBV08] L. Jacob, F. Bach, and J. P. Vert. Clustered multi-task learning: A convex formulation. In *NIPS*, pages 745–752, 2008.

[JC07] S. Ji and L. Carin. Bayesian Compressive Sensing and Projection Optimization. In *ICML*, 2007.

[JGJS99] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[Joa99] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[KHDM98] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.

[KKL+08] S. J. Kim, K. Ko, M. Lustig, S. Boyd, and D. Gorinevsky. An Interior-Point Method for Large-Scale L1-Regularized Least Squares. *IEEE Journal of Selected Topics in Signal Processing*, 1:606–617, 2008.

[KT07] E. Krupka and Naftali Tishby. Incorporating Prior Knowledge on Features into Learning. In *AISTATS*, 2007.

[Kuk00] A. Kuk. A pairwise likelihood approach to analyzing correlated binary data. *Statistics and Probability Letters*, 47:329–335, 2000.

[Lin88] B. G. Lindsay. Composite likelihood methods. *Contemporary Mathematics*, 80:221–239, 1988.

[LLAN06] S. I. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient L-1 Regularized Logistic Regression. In *AAAI*, 2006.

[LYS11] B. G. Lindsa, G. Y. Yi, and J. Sun. Issues and strategies in the selection of composite

likelihoods. *Statistica Sinica*, 21:71–105, 2011.

[MHTS08] K. V. Mardia, G. Hughes, C. C. Taylor, and H. Singh. A multivariate von mises distribution with applications to bioinformatics. *Canadian Journal of Statistics-revue Canadienne De Statistique*, 36:99–109, 2008.

[MKHT09] K. V. Mardia, J. T. Kent, G. Hughes, and C. C. Taylor. Maximum likelihood estimation using composite likelihoods for closed exponential families. *Biometrika*, 96(4):975–982, 2009.

[MMR+01] K-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Trans. Neural Networks*, 12(2):181–201, 2001.

[MV05] G Molenberghs and G Verbeke. *Models for Discrete Longitudinal Data*. Springer, New York, 2005.

[NMTM00] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39:103–134, 2000.

[NT08] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3), 2008.

[NW00] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000.

[OTJ09] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009.

[PPF04] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, pages 45–54, 2004.

[PRV06] Oriol Pujol, Petia Radeva, and Jordi Vitri. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1007–1012, 2006.

[RBLN07] R. Raina, A. Battle, H. Lee, and B. Packerand A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, pages 759–766, 2007.

[RD09] P. Rai and H. Daume. Multi-label prediction via sparse infinite cca. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1518–1526. 2009.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[RK04] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, December 2004.

[RNK06] R. Raina, A. Y. Ng, and D. Koller. Constructing Informative Priors using Transfer Learning. In *ICML*, pages 713–720, 2006.

[ROF92] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[RPH08] J. Read, B. Pfahringer, and G. Holmes. Multi-label classification using ensembles of pruned sets. In *IEEE International Conference on Data Mining*, pages 995–1000,

2008.

[RPHF09] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In *Principles of Data Mining and Knowledge Discovery*, pages 254–269, 2009.

[SK06] V. Sindhwani and SS. Keerthi. Large Scale Semi-Supervised Linear SVMs. In *SIGIR*, 2006.

[SSM98] B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998.

[Sul86] F. O. Sullivan. A statistical perspective on ill-posed inverse problems. *Statist. Sci.*, 1:502–518, 1986.

[TA77] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. Winston and Sons, 1977.

[TGK03] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

[THJA04] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[Tib96] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Soceity, Series B*, 58(1):267–288, 1996.

[TKV10] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*. Springer, 2 edition, 2010.

[TL10] F. Tai and H.-T. Lin. Multi-label classification with principal label space transformation. In *Second International Workshop on learning from Multi-Label Data*. 2010.

[TO96] S. Thrun and J. O'Sullivan. Discovering Structure in Multiple Learning Tasks: The TC Algorithm. In *ICML*, pages 489–497, 1996.

[Tro06] J. A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Trans. on Information Theory*, 52(3):1030–1051, 2006.

[TSR+05] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. Sparsity and smoothness via the fused lasso. *Journal Of The Royal Statistical Society Series B*, 67(1):91–108, 2005.

[TV07] G. Tsoumakas and I. Vlahavas. Random k-labelsets: an ensemble method for multi-label classification. In *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pages 406–417, Berlin, Heidelberg, 2007.

[Vap95] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[VBW96] L. Vandenberghe, S. Boyd, and S. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19:499–533, 1996.

[VRF11] C. Varin, N. Reid, and D. Firth. An overview of composite likelihood methods. *Sta-*

tistica Sinica, 21:5–42, 2011.

[W+04] Y. Wang et al. The BCI Competition 2003 - Data Set IV: An Algorithm Based on CSSD and FDA for Classifying Single-Trail EEG. . *IEEE Trans. Biomedical Engineering*, 51(6):1081–1086, 2004.

[Wal92] G. K. Wallace. The JPEG Still Picture Compression Standard. *IEEE Trans. Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[WBS06] K. Q. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*. 2006.

[WLW04] T. Wu, C. Lin, and R. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, pages 975–1005, 2004.

[WTH09] D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[XLCK07] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

[YCY+07] K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu. Stochastic relational models for discriminative link prediction. In *NIPS*, pages 1553–1560, 2007.

[YJHN07] Jun Yang, Yu-Gang Jiang, Alexander G. Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *The international workshop on multimedia information retrieval*, MIR '07, 2007.

[YL06] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[YLZG09] K. Yu, J. Lafferty, S. Zhu, and Y. Gong. Large-scale collaborative prediction using a nonparametric random effects model. In *ICML*, pages 1185–1192, 2009.

[YTY07] S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *ICML*, page 1103, 2007.

[ZGL03] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[ZGY06] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *NIPS*, pages 1585–1592, 2006.

[ZH05] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67:301–320, 2005.

[Zhu06] X. Zhu. Semi-supervised learning literature survey, 2006.

[ZLW07] S. Zhou, J. Lafferty, and L. Wasserman. Compressed Regression. In *NIPS*, 2007.

[ZS10a] Y. Zhang and J. Schneider. Learning Compressible Models. In *SDM*, 2010.

[ZS10b] Y. Zhang and J. Schneider. Learning Multiple Tasks with a Sparse Matrix-Normal Penalty. In *NIPS*, 2010.

[ZS10c] Y. Zhang and J. Schneider. Projection Penalty: Dimension Reduction without Loss. The 27th International Conference on Machine Learning. In *ICML*, 2010.

[ZS11] Y. Zhang and J. Schneider. Multi-label Output Codes using Canonical Correlation Analysis. In *AISTATS*, 2011.

[ZS12a] Y. Zhang and J. Schneider. A Composite Likelihood View for Multi-Label Classification. In *AISTATS*, 2012.

[ZS12b] Y. Zhang and J. Schneider. Maximum Margin Output Coding. In *Submission to ICML*, 2012.

[ZSD08] Y. Zhang, J. Schneider, and A. Dubrawski. Learning the Semantic Correlation: An Alternative Way to Gain from Unlabeled Text. In *NIPS*, 2008.

[ZSM+11] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga. Bayesian chain classifiers for multidimensional classification. In *IJCAI*, pages 2192–2197, 2011.

[ZY06] P. Zhao and B. Yu. On model selection consistency of lasso. *J. Mach. Learn. Res.*, 7:2541–2563, 2006.

[ZY10] Y. Zhang and D. Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the Twenty-fourth Conference on Uncertainty in AI (UAI)*, 2010.

[ZZ10] M. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *KDD*, pages 999–1008, 2010.