UNIVERSITY OF CALIFORNIA, SAN DIEGO

Exploiting Label Correlations for Multi-label Classification

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Cheng-Xian Li

Committee in charge:

Professor Charles Elkan, Chair
Professor Lawrence Saul
Professor Nuno Vasconcelos

2011

The thesis of Cheng-Xian Li is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2011

TABLE OF CONTENTS

Signature Page . . . iii

Table of Contents . . . iv

List of Figures . . . vi

List of Tables . . . vii

Abstract of the Thesis . . . viii

Chapter 1 Introduction . . . 1

Chapter 2 Multi-label classification . . . 4
2.1 Definition . . . 4
2.2 Binary relevance method . . . 5

Chapter 3 Correlation analysis . . . 6
3.1 Unconditional label correlation . . . 6
3.2 Conditional label correlation . . . 9
3.2.1 Conditional correlation by feature analysis . . . 10
3.2.2 Conditional correlation by partial correlation . . . 12

Chapter 4 Multi-label learning with conditional random fields . . . 14
4.1 Conditional random fields . . . 14
4.1.1 CML model . . . 16
4.1.2 CMLF model . . . 17
4.2 Supported heuristic for CRFs . . . 19
4.3 Gibbs sampling . . . 20
4.4 Tree structured CRF approximation . . . 23

Chapter 5 More learning with label correlations . . . 26
5.1 Multi-class method . . . 26
5.2 Multi-label k-nearest neighbor . . . 27
5.3 Label space reduction . . . 28
5.3.1 Label PCA . . . 29
5.3.2 Feature/label CCA . . . 31
5.4 Label clustering . . . 31

Chapter 6 Experimental design . . . 35
6.1 CAL500 data set . . . 35
6.1.1 Label cardinality . . . 36
6.2 Evaluation of multi-label classification . . . 37

6.3 Feature preprocessing 40
6.4 Experimental setup 41

Chapter 7 Experimental results 43
7.1 Small label set experiment 43
7.2 Entire CAL500 experiment 44
7.3 Results of multi-label learning methods 45
7.3.1 CRF tree/chain approximations 46
7.3.2 Is correlation-based structure approximation a good idea? 47
7.3.3 Label space reduction 49
7.3.4 Label clustering 50
7.4 Discussion 51

Chapter 8 Conclusion and future work 53

Bibliography 56

LIST OF FIGURES

Figure 4.1: The factor graph of a CML model with 4 labels. 17
Figure 4.2: The factor graph of a CMLF model with 4 labels. 18

LIST OF TABLES

Table 3.1: Two random variables that are unconditionally uncorrelated but conditionally correlated 9

Table 4.1: Time complexity of CML and CMLF with supported heuristic . 20
Table 4.2: Time complexity of CML and CMLF using Gibbs sampling . . . 23

Table 6.1: Label cardinality statistics of CAL500 36
Table 6.2: Label statistics of CAL500 39

Table 7.1: Performance evaluation on a 8-label subset of CAL500 44
Table 7.2: Performance of random prediction with specified label occurrence rate 45
Table 7.3: Performance of random prediction with label occurrence rate consistent with training data 45
Table 7.4: Performance comparison of multi-label learning algorithms on entire CAL500 labels 46
Table 7.5: Normalized inversion count that reflects how correlation-based approximation is related to maximum likelihood 48
Table 7.6: Performance of label PCA with different reduced dimensionalities 50
Table 7.7: Clustering performance with different correlation measures . . . 50

ABSTRACT OF THE THESIS

Exploiting Label Correlations for Multi-label Classification

by

Cheng-Xian Li

Master of Science in Computer Science

University of California, San Diego, 2011

Professor Charles Elkan, Chair

Multi-label classification is widely used for various applications such as automatic music tagging. Often, multi-label learning is done by transforming into multiple independent binary classification problems. In order to produce better classification result, label correlations should be taken into account. This thesis first discusses how to model label correlations in a quantitative way and categorizes the concept into unconditional and conditional correlations. After that, this thesis shows how to exploit both kinds of label correlations for multi-label learning algorithms. The main model this thesis addresses is conditional random fields (CRFs). This thesis shows how to apply CRFs for multi-label classification. Because of the intractable nature of CRF inference, several approximation algorithms to make it applicable for larger label sets are described. Various other learning algorithms

that exploit label correlations are also discussed in this thesis. In the end, all the mentioned multi-label learning algorithms are evaluated with a music data set, CAL500, composed of 502 songs categorized into 174 labels.

Chapter 1

Introduction

Multi-label classification is an extension of the traditional single-label classification problem. Single-label classification is to learn from a set of examples that are associated with a single label from a set of labels, while multi-label classification is instead to learn from examples that are associated with a subset of labels.

Multi-label learning algorithms are widely used for various application including text categorization, automatic music tagging, etc. Take music tagging for example, given a set of tags such as music genre, the goal of multi-label learning is to correctly predict which tags should be associated with a song.

There are several approaches for multi-label classification. The most common method is to regard the problem as a set of independent binary classifications one for each label, that is, to individually predict if each label appears or not. However, this method does not take label correlations into account, while it is often the case that certain labels are correlated. Using our music example, suppose we have learned that the tag *romantic* is always negatively correlated with the tag *fast tempo*, we would never predict something unreasonable such as having both of the tags if this correlation is taken into account.

The main theme of this thesis is to discuss what label correlations are and how to exploit them for multi-label classification. The thesis will break into the following parts. To begin with, we will give a formal definition of multi-label classification problem and briefly describe a commonly used method called binary

relevance. Then we will discuss what label correlation is in a quantitative way. In the discussion, we will categorize label correlation into two kinds: unconditional label correlation and conditional label correlation. Unconditional label correlation is measured only on the labels. Three unconditional label correlation measures are described, which are Pearson's correlation coefficient, mutual information, and χ^2 score. On the other hand, in addition to utilizing information from labels, conditional label correlation takes the associated features into account. Two conditional label correlation measures are illustrated. The first one is to measure the similarity of the relevance between labels and each feature. The second one is formulated by partial correlation.

Once label correlations are formally defined, we will then introduce how to perform multi-label classification by incorporating label correlations into learning algorithms. The first model we discuss is conditional random fields (CRFs) [1]. We give a brief introduction to CRFs, and then show two CRF models that exploit unconditional and conditional correlations respectively. Then we describe several CRF approximation methods because the exact CRF training and inference are intractable. The approximation methods include supported heuristic, Gibbs sampling, and correlation-based CRF tree/chain approximations.

We follow up by describing other multi-label learning algorithms and explain how they utilize label correlations: multi-class method, and multi-label k-nearest neighbor. A different multi-label learning framework called label space reduction will be then introduced, which aims at learning fewer base learners than the size of label set. Two methods will be used: label PCA and feature/label CCA, which utilize unconditional and conditional label correlation respectively.

Since several multi-label learning algorithms we describe are not able to scale up for larger label set, a divide and conquer framework will be introduced which we call label clustering. This breaks the entire label set into several smaller correlated clusters so that those time-consuming learning algorithms are able to handle.

We evaluate all the algorithms with CAL500 [2], a music data set composed of 502 songs categorized into 174 labels. We briefly mention how CAL500 is created

and what the features and labels are, together with some basic statistics. Then we describe 6 metrics to evaluate multi-label classification. Procedures to pre-process raw CAL500 features are introduced next, then our experimental setup.

Finally, we show the experimental results and discuss the performance of each learning algorithm. The thesis ends with listing some potential future directions.

Chapter 2

Multi-label classification

2.1 Definition

The traditional binary classification problem is to learn the mapping from a feature defined in space $\mathcal{X}$ to a binary label outcome in $\mathcal{Y}$ where $|\mathcal{Y}| = 2$, that is, to learn the function $f\colon \mathcal{X} \to \mathcal{Y}$. Multi-label classification takes a step further that generalize binary classification as follows. Given a set of labels $\mathcal{Y}$ which now can be more than two, the goal of multi-label classification is to predict whether each of them appears or not, that is, the outcome is a subset of $\mathcal{Y}$. Therefore, the mapping function to learn could be formulated as $f\colon \mathcal{X} \to 2^{\mathcal{Y}}$ where $2^{\mathcal{Y}}$ denotes the power set of $\mathcal{Y}$. In practice, we usually consider the case that the feature could be represented as a fixed length real vector, therefore multi-label mapping becomes $f\colon \mathbb{R}^D \to 2^{\mathcal{Y}}$ supposing the feature is D-dimensional.

It is a convention to represent any case in the label power set $2^{\mathcal{Y}}$ by a binary vector. Suppose there are L labels ($|\mathcal{Y}| = L$), then any multi-label outcome could be represented by an L-dimensional binary vector $\mathbf{y} = [y^{(1)}, y^{(2)}, \ldots, y^{(L)}]^{\mathsf{T}} \in \{0,1\}^L$ where $y^{(i)} = 1$ if i-th label occurs and $y^{(i)} = 0$ otherwise.[1]

With the binary power set representation, multi-label mapping function $f\colon \mathbb{R}^D \to \{0,1\}^L$ is learned from multi-label training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ composed of N examples where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0,1\}^L$. The goal of multi-label

[1] We stay with the notation that i-th component of vector $\mathbf{y}$ is denoted as $y^{(i)}$; the parenthesis around the index is used to prevent the confusion with exponent.

classification is to learn such mapping that is able to make accurate prediction for an unknown feature vector $\mathbf{x} \in \mathbb{R}^D$ in terms of a specific loss function.

2.2 Binary relevance method

A commonly used approach toward multi-label classification is the binary relevance method. It is based on the assumption that labels are controlled by the feature independently. Therefore, we decompose the original multi-label classification problem into multiple binary classification problems one for each label. Specifically, instead of learning $f\colon \mathbb{R}^D \to \{0,1\}^L$ with training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, binary relevance learns L independent binary classifiers $f_j\colon \mathbb{R}^D \to \{0,1\}$ trained by $\{(\mathbf{x}_i, \mathbf{y}_i^{(j)})\}_{i=1}^N$ for $j = 1, \ldots, L$.

Since binary classification is a well-studied problem, we can apply any binary classification algorithms as the base learner, for instance logistic regression or SVM, to optimize on certain domain specific loss. Binary relevance method runs in time linear to the number of labels. Besides, it is highly parallelizable because the classification problem for each label could be done independently. Therefore, the simplicity of binary relevance makes it one of the standard methods for multi-label classification in practice. We use this method as our baseline to compare with other multi-label learning algorithms we will describe later.

Chapter 3

Correlation analysis

In general, it is claimed that exploiting label correlation could improve multi-label classification result over the commonly used binary relevance method. Few papers give a precise definition of correlation such as [3]. [3] categorizes such concept into conditional and unconditional label dependence, which we think is an insightful direction. In the following section, we use this concept to formulate label correlation in a quantitative manner to describe the degree of correlation.

3.1 Unconditional label correlation

For the multi-label learning problem $\mathcal{X} \rightarrow 2^{\mathcal{Y}}$, unconditional label correlation describes the relationships among labels $\mathcal{Y}$ regardless of the associated feature $\mathcal{X}$. For example, consider the case of two labels $\mathcal{Y} = (Y_1, Y_2)$; if most of the training examples have either $(Y_1, Y_2) = (0, 0)$ or $(Y_1, Y_2) = (1, 1)$ we could claim that Y_1 and Y_2 are positively correlated, and therefore any prediction violating such relationship would be a poor prediction.

For simplicity, we first consider the unconditional correlation based only on the pairwise label statistics of training data other than correlations involving multiple labels at the same time. First of all, we start with the fact that correlation is closely related to probabilistic dependence which has a formal mathematical

definition. Two random variables Y_1 and Y_2 are independent if and only if

$$P(Y_1, Y_2) = P(Y_1)P(Y_2).$$

When two random variables are independent, they are uncorrelated. Unlike dependence has a strict definition, there is no universal definition for correlation, so we stay with this flexible term to describe the relationship among labels to address properties we want to emphasize on. Moreover, when two random variables are shown to be dependent it tells nothing about the degree of dependence so we are not able to base on the degree to neglect certain obscure correlations for efficiency issue.

The most widely used measure to describe the degree of correlation is Pearson's correlation coefficient, which treats the prediction of multi-label as a regression problem. Here we overload the term "correlation coefficient" to describe certain degree. Unlike Pearson's correlation coefficient that describes whether it is positively correlated or negatively by its sign, we want an absolute measure that shows the degree of correlation. One possible design of such correlation coefficient is just taking the square of Pearson's correlation coefficient as the degree:

$$\rho^2(Y_1, Y_2) = \left(\frac{E[(Y_1 - \mu_1)(Y_2 - \mu_2)]}{\sigma_1 \sigma_2} \right)^2 \tag{3.1}$$

where μ_i and σ_i are the mean and standard deviation of Y_i. This can be rewritten into the form of sample correlation coefficient given the labels of training data associated with Y_1 and Y_2 being $\{(p_i, q_i)\}_{i=1}^N$:

$$\rho^2(Y_1, Y_2) = \frac{\left(N \sum_{i=1}^N p_i q_i - \sum_{i=1}^N p_i \sum_{i=1}^N q_i \right)^2}{\left(N \sum_{i=1}^N p_i^2 - \left(\sum_{i=1}^N p_i \right)^2 \right) \left(N \sum_{i=1}^N q_i^2 - \left(\sum_{i=1}^N q_i \right)^2 \right)}. \tag{3.2}$$

Since the random variable for each label is defined on discrete values $Y_i \in \{0, 1\}$, there are some alternative that might be more sensible to describe the correlation degree. One option is mutual information, which is an information theoretic measure of the dependence of two random variables. For case of the label pair $(Y_1, Y_2) \in \{0, 1\}^2$, mutual information is defined as

$$I(Y_1, Y_2) = \sum_{y_1, y_2 \in \{0,1\}} P(y_1, y_2) \log \left(\frac{P(y_1, y_2)}{P(y_1)P(y_2)} \right).$$

The maximum likelihood estimates of $P(y_1, y_2), P(y_1), P(y_2)$ are usually used to compute $I(Y_1, Y_2)$, so it can also be represented equivalently:

$$I(Y_1, Y_2) = \sum_{i,j \in \{0,1\}} \frac{N_{ij}}{N} \log \left(\frac{N N_{ij}}{(N_{i0} + N_{i1})(N_{0j} + N_{1j})} \right) \tag{3.3}$$

where N_{ij} denotes the count of training examples satisfying $(Y_1, Y_2) = (i, j)$ and $N = N_{00} + N_{01} + N_{10} + N_{11}$.

Another widely used option is the χ^2 test, which is used to test the dependence of two random variables in statistics by the following quantity:

$$X^2(Y_1, Y_2) = \sum_{i,j \in \{0,1\}} \frac{(N_{ij} - E_{ij})^2}{E_{ij}}$$

where N_{ij} follows the same definition used previously, and E_{ij} is the expected frequency under the assumption that Y_1 and Y_2 are independent, which can be computed by

$$E_{ij} = N \cdot P(Y_1 = i) \cdot P(Y_2 = j) = N \cdot \frac{N_{i0} + N_{i1}}{N} \cdot \frac{N_{0j} + N_{1j}}{N}$$

so the χ^2 score can be calculated by

$$X^2(Y_1, Y_2) = \frac{N(N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01})(N_{11} + N_{10})(N_{10} + N_{00})(N_{01} + N_{00})}. \tag{3.4}$$

The above criteria provide quantitative measure of the degree of correlation for any two labels. Applying to all pair of labels we could build a correlation matrix $C \in \mathbb{R}^{L \times L}$ where the entry C_{ij} indicates the correlation between label i, j.

To incorporate unconditional correlation into the learning process, it is usual to model the correlation in a more relaxed manner such as the frequency of all four possible combinations $(0, 0), (0, 1), (1, 0), (1, 1)$ for any pair of labels. This statistic could be used for a smoothing post-processing of an initial prediction. One example that exploits unconditional correlation is the classifier stacking method or also known as 2BR [4]. It first makes prediction for each label independently and then use that as an input to train a second-layer classifier to make the final prediction more consistent with the correlation measure.

3.2 Conditional label correlation

Similar to unconditional label correlation mentioned above, conditional correlation is closely related to probabilistic conditional dependence which is defined under a fixed feature $\mathbf{x} \in \mathcal{X}$. Specifically, when two labels Y_1, Y_2 are independent conditioned on $\mathbf{x}$ by definition

$$P(Y_1, Y_2|\mathbf{x}) = P(Y_1|\mathbf{x})P(Y_2|\mathbf{x})$$

and again, this implies Y_1, Y_2 are conditionally uncorrelated given $\mathbf{x}$. Since the classification problems are based on the correlation between feature $\mathcal{X}$ and the corresponding label $\mathcal{Y}$, the conditional correlation may be more useful than unconditional one for the purpose of prediction.

Notice that unconditional correlation does not imply conditional correlation and vice versa. The case that two labels are unconditionally uncorrelated but conditionally correlated can be illustrated from one of the examples borrowed from [3]. Consider the following joint probability table where the feature is composed of a single binary random variable $X \in \{0, 1\}$:

Table 3.1: Two random variables that are unconditionally uncorrelated but conditionally correlated

Y_1	Y_2	$P(X=0, Y_1, Y_2)$	$P(X=1, Y_1, Y_2)$	$P(Y_1, Y_2)$
0	0	0.25	0	0.25
0	1	0	0.25	0.25
1	0	0	0.25	0.25
1	1	0.25	0	0.25

The unconditional probability $P(Y_1, Y_2) = P(X=0, Y_1, Y_2)+P(X=1, Y_1, Y_2)$ is a constant 0.25 for all four possible configurations of Y_1 and Y_2 which shows that Y_1 and Y_2 are uncorrelated. However, it is obvious that Y_1 and Y_2 are actually correlated if the feature X is taken into account, that is, when $X = 0$ we have $Y_1 = Y_2$; when $X = 1$ we have $Y_1 = 1 - Y_2$. This is an example shown that it is possible to be unconditionally uncorrelated while still conditionally correlated.

On the other hand, when two labels Y_1, Y_2 are purely a function of X, Y_1 and Y_2 are conditionally independent given X (in graphical model, Y_1, Y_2 are d-

separated by X) but they are dependent if X is not observed. This shows that even if two labels are conditionally uncorrelated it is still possible that they are unconditionally correlated.

In practice, it is hard to define conditional correlation in the same manner as unconditional correlation for each possible feature configuration, because the feature space $\mathcal{X}$ is too large especially when it's continuous. Moreover, the purpose of correlation analysis is to perform structure learning or label clustering, which we will describe later, so here we try to model the relatedness between two labels by considering the influence of the associated feature.

3.2.1 Conditional correlation by feature analysis

The first method to model conditional correlation could be illustrated from the viewpoint of feature selection. For example, say there are three labels $\mathcal{Y} = \{Y_1, Y_2, Y_3\}$ and each instance has two features $\mathcal{X} = \{X_1, X_2\}$. If feature analysis shows that both Y_1 and Y_2 are only related to X_1 but have nothing to do with X_2, on the other hand Y_3 only depends on X_2 but unrelated to X_1, that is, X_1 controls the outcomes of Y_1 and Y_2 while X_2 controls Y_3, then we could conclude that Y_1 and Y_2 are more correlated than Y_3 and Y_1. The general procedure to apply this concept to measure conditional correlation is to evaluate by how similar a set of features each pair of labels is controlled as shown in Algorithm 1.

There are two parts in Algorithm 1. The first part is to compute the label-feature correlation. Again we could use the square of Pearson's correlation coefficient as in (3.1) to describe the degree of correlation. Another widely used method is the Fisher criterion score [5] that aims at discriminating between two classes. Using the same notation of Algorithm 1, consider f-th feature $\{x_k^{(f)}\}_{k=1}^N$ and ℓ-th label $\{y_k^{(\ell)}\}_{k=1}^N$, the Fisher criterion score is given by

$$r_f^{(\ell)} = \frac{(\mu_f^+ - \mu_f^-)^2}{(\sigma_f^+)^2 + (\sigma_f^-)^2}$$

where μ_f^+ and σ_f^+ are the mean and standard deviation of those $x_k^{(f)}$ for all k satisfying $y_k^{(\ell)} = 1$. Similarly, μ_f^- and σ_f^- are the mean and standard deviation associated with $y_k^{(\ell)} = 0$.

Algorithm 1: Framework to measure conditional correlation coefficient

Input: training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0,1\}^L$

Output: conditional correlation matrix $C \in \mathbb{R}^{L \times L}$ where C_{ij} is the conditional correlation coefficient of label i, j

foreach label ℓ **do**

  foreach feature f **do**

    $r_f^{(\ell)}$ = relevance between feature $\{x_k^{(f)}\}_{k=1}^{N}$ and label $\{y_k^{(\ell)}\}_{k=1}^{N}$

  end

end

foreach label pair $i \neq j$ **do**

  C_{ij} = similarity between $\{r_k^{(i)}\}_{k=1}^{D}$ and $\{r_k^{(j)}\}_{k=1}^{D}$

end

The second part is to compute the correlation coefficient by measuring for each label pair how similar the corresponding two sets of correlation coefficients (the degree of relevance associated with each feature) are. This could be done by several ways: One could use the number of common top-k features as the similarity measure, that is, for each pair of labels Y_1, Y_2, if the corresponding top-k (k is a configurable parameter) relevant features for Y_1, Y_2 are indexed by S_1 and S_2 respectively where S_i is a subset of $\{1, \ldots, D\}$ given the feature space $\mathcal{X} \in \mathbb{R}^D$, then the correlation coefficient could be described as $|S_1 \cap S_2|$. Another approach to measure the similarity between two sequences is to compute their correlation coefficient. Also we can just regard the two sequences as vectors and take the Euclidean distance as their dissimilarity measure.

One drawback of this approach is that it assumes features are independent to each other, that is, no features lie on the space spanned by a set of other features. It is possible that the predictability comes from a subset of features, which does not appear when observing individual features. If such phenomenon is indeed the case, then per-feature relevance analysis would fail to provide useful information.

3.2.2 Conditional correlation by partial correlation

Another approach to model conditional label correlation is partial correlation [6], which measures the degree of correlation between two target random variables with the effect of a set of controlling random variables removed. Traditionally, partial correlation is used under linear regression, computing the correlation between the residuals of the two target random variables. For the classification problem, we can adapt the same concept to evaluate the conditional label correlation as follows.

Let two binary random variables P and Q represent the outcome of two given labels, and the random variable X corresponds to the feature vector associated with labels P and Q. Also let the training data $\{(\mathbf{x}_i, p_i, q_i)\}_{i=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^D$, $p_i, q_i \in \{0, 1\}$ to be sampled from a joint distribution over X, P, and Q. Supposing a linear model is used, we first train two linear classifiers f_P, f_Q to predict P from X and Q from X parameterized by $\mathbf{w}_P$ and $\mathbf{w}_Q$ respectively:

$$p \approx f_P(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}_P^\mathsf{T}\mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases} \qquad q \approx f_Q(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}_Q^\mathsf{T}\mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

where we can assume that a non-zero constant is attached to the original feature vector so a bias term is implicitly integrated into the classifiers, so that the decision threshold is zero.

Having the trained parameters $\mathbf{w}_P$ and $\mathbf{w}_Q$, we can then compute the residuals of each training example by taking the decision score if the example is mispredicted. The residuals of i-th example are

$$r_{P,i} = \begin{cases} \mathbf{w}_P^\mathsf{T}\mathbf{x}_i & \text{if } p_i \neq f_P(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases} \qquad r_{Q,i} = \begin{cases} \mathbf{w}_Q^\mathsf{T}\mathbf{x}_i & \text{if } q_i \neq f_Q(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Since we use the decision score $\mathbf{w}^\mathsf{T}\mathbf{x}$ as the residual, the parameters $\mathbf{w}_P$ and $\mathbf{w}_Q$ should be normalized so that the scores for both labels P and Q are in the same scale. Logistic regression comes with such convenience because it is formulated with a probability measure. Instead of $\mathbf{w}^\mathsf{T}\mathbf{x}$ in the residual formula above, logistic regression uses the normalized score $1/(1+\exp(-\mathbf{w}^\mathsf{T}\mathbf{x})) - 0.5$ which is bounded in $[-0.5, 0.5]$.

With all the residuals of each training example for both labels, we can then compute the square of the sample partial correlation similar to (3.2):

$$\rho^2(P,Q) = \frac{\left(N\sum_{i=1}^{N} r_{P,i}r_{Q,i} - \sum_{i=1}^{N} r_{P,i}\sum_{i=1}^{N} r_{Q,i}\right)^2}{\left(N\sum_{i=1}^{N} r_{P,i}^2 - \left(\sum_{i=1}^{N} r_{P,i}\right)^2\right)\left(N\sum_{i=1}^{N} r_{Q,i}^2 - \left(\sum_{i=1}^{N} r_{Q,i}\right)^2\right)}$$

When $\rho^2(P,Q)$ is undefined due to the fact that any of P and Q has zero variance or two classifiers have no training error, we can just set $\rho^2(P,Q)$ to 0, which regards P, Q as independent.

Applying this to each label pair, we can again build the whole correlation matrix $C \in \mathbb{R}^{L\times L}$ as in the case of unconditional label correlation.

Chapter 4

Multi-label learning with conditional random fields

4.1 Conditional random fields

Conditional random field (CRF) [1] is a framework that could be used to exploit either unconditional or conditional label correlation for multi-label classification. A CRF is an undirected graphical model in which each vertex represents a random variable and each edge represents a dependency between two random variables associated with the vertices it connects to. In general, CRF is in the form of a probabilistic log-linear model as the following conditional probability parameterized by $\mathbf{w}$:

$$P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_k w_k f_k(\mathbf{x},\mathbf{y})\right) \tag{4.1}$$

where $f_k(\mathbf{x},\mathbf{y})$ is called feature function which could be any real-valued function used to encode any kind of relations between feature $\mathbf{x}$ and label $\mathbf{y}$ in a flexible way. Also, in (4.1), $Z(\mathbf{x})$ is the partition function as follows that normalizes the whole quantity into a probabilistic distribution that satisfies: $\sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\mathbf{w}) = 1$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp\left(\sum_k w_k f_k(\mathbf{x},\mathbf{y}')\right). \tag{4.2}$$

The CRF training step is to estimate parameters $\mathbf{w}$ by the maximum likelihood method on the given training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$. The log likelihood is formulated as

$$\begin{aligned} \ell(\mathbf{w}|\mathcal{D}) &= \log\left(\prod_{i=1}^{N} P(\mathbf{y}_i|\mathbf{x}_i;\mathbf{w})\right) - \sum_k \frac{w_k^2}{2\sigma^2} \\ &= \sum_{i=1}^{N}\left(\sum_k w_k f_k(\mathbf{x}_i,\mathbf{y}_i) - \log Z(\mathbf{x}_i)\right) - \sum_k \frac{w_k^2}{2\sigma^2} \end{aligned} \tag{4.3}$$

where the last term, which is essentially the Gaussian prior, is for L_2 regularization controlled by a pre-defined parameter σ to reduce the risk of over-fitting especially when the number of parameters, $\dim(\mathbf{w})$, is huge. The parameter σ is usually chosen by cross validation.

The parameters $\mathbf{w}$ are estimated by maximizing the regularized log likelihood function $\ell(\mathbf{w}|\mathcal{D})$. Notice that this maximization problem is concave so the global optimum can be obtained by gradient ascent. The gradient of the log likelihood associated with w_k is given by

$$\begin{aligned} \frac{\partial}{\partial w_k}\ell(\mathbf{w}|\mathcal{D}) &= \sum_{i=1}^{N}\left(f_k(\mathbf{x}_i,\mathbf{y}_i) - \frac{\sum_{\mathbf{y}'}\exp\left(\sum_k w_k f_k(\mathbf{x}_i,\mathbf{y}')\right) f_k(\mathbf{x}_i,\mathbf{y}')}{\sum_{\mathbf{y}'}\exp\left(\sum_k w_k f_k(\mathbf{x}_i,\mathbf{y}')\right)}\right) - \frac{w_k}{\sigma^2} \\ &= \sum_{i=1}^{N}\left(f_k(\mathbf{x}_i,\mathbf{y}_i) - \sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x}_i;\mathbf{w}) f_k(\mathbf{x}_i,\mathbf{y}')\right) - \frac{w_k}{\sigma^2} \end{aligned}$$

As for the prediction step, or decoding step, we determine the most likely label combination $\mathbf{y}$ given the feature $\mathbf{x}$ by

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}}\left(\sum_k w_k f_k(\mathbf{x},\mathbf{y})\right) \tag{4.4}$$

The final part holds because the partition function $Z(\mathbf{x})$ is a constant for a fixed feature $\mathbf{x}$.

The work of applying CRF to multi-label classification that takes label correlation into account is described in detail in [7]. Two CRF models are proposed in [7]: one is called *collective multi-label classifier* (CML) which utilizes unconditional label correlation; while the other is called *collective multi-label with features*

classifier (CMLF) which utilizes conditional label correlation. We describe these two models in detail in the following sections.

4.1.1 CML model

The CML model [7] exploits unconditional label correlation by introducing two sets of feature functions. The first set of feature functions models how each feature $x^{(i)}$ influences each single label $y^{(j)}$ and has the following form:

$$f_k(\mathbf{x}, \mathbf{y}) = x^{(i)}\mathbf{I}(y^{(j)} = 1)$$

The index k enumerates through all the following combinations of each feature and label:

$$k \in \{(i, j) : 1 \leq i \leq D, 1 \leq j \leq L\}.$$

The second set of feature functions models the pairwise unconditional label correlation (regardless of the features $\mathbf{x}$) by

$$f_{k'}(\mathbf{y}) = \begin{cases} \mathbf{I}(y^{(i)} = 0)\ \mathbf{I}(y^{(j)} = 0) & \text{if } q = 1 \\ \mathbf{I}(y^{(i)} = 1)\ \mathbf{I}(y^{(j)} = 0) & \text{if } q = 2 \\ \mathbf{I}(y^{(i)} = 0)\ \mathbf{I}(y^{(j)} = 1) & \text{if } q = 3 \\ \mathbf{I}(y^{(i)} = 1)\ \mathbf{I}(y^{(j)} = 1) & \text{if } q = 4 \end{cases}$$

where the index k' enumerates through all label pairs with four possible configurations as $k' \in \{(i, j, q) : q \in \{1, 2, 3, 4\};\ 1 \leq i < j \leq L\}$

Notice that we could drop one feature function out of four because any one of them could be uniquely determined by the other three and the linearity between feature functions makes one of them redundant.

Also we can simplify the model by considering only positive and negative correlations on each label pairs that makes the following set of feature functions:

$$f_{k'}(\mathbf{y}) = \begin{cases} \mathbf{I}(y^{(i)} = y^{(j)}) & \text{if } q = 1 \\ \mathbf{I}(y^{(i)} \neq y^{(j)}) & \text{if } q = 2 \end{cases}$$

where $k' \in \{(i, j, q) : q \in \{1, 2\};\ 1 \leq i < j \leq L\}$, and again we are free to drop one of them to make it as follows for $k' \in \{(i, j) : 1 \leq i < j \leq L\}$

$$f_{k'}(\mathbf{y}) = \mathbf{I}(y^{(i)} = y^{(j)}). \tag{4.5}$$

Figure 4.1 is an example of a CML model with 4 labels represented by a factor graph. Nodes Y_1, Y_2, Y_3, and Y_4 are the 4 labels and the feature random variables are represented as a single node $\mathbf{X}$ for clarity. All pairs of labels are linked together, and the feature node $\mathbf{X}$ is connected to each label node. Each square denotes a factor associated with a feature function when it connects to a pair of labels, or a set of feature functions if it links to the feature node.

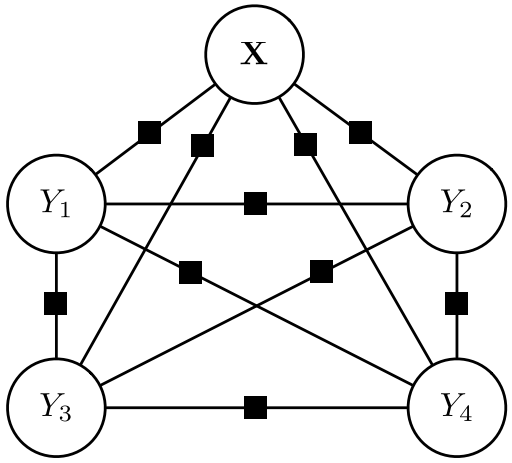

Figure 4.1: The factor graph of a CML model with 4 labels.

4.1.2 CMLF model

Similar to CML model, CMLF model [7] has the following set of feature functions for independent binary classification for each label with the index $k \in \{(i,j) : 1 \leq i \leq D, 1 \leq j \leq L\}$:

$$f_k(\mathbf{x}, \mathbf{y}) = x^{(i)}\mathbf{I}(y^{(j)} = 1)$$

In order to model the conditional label correlation, CMLF introduces another type of feature functions that aims to capture the relationship among all of the combinations of a pair of labels and a feature indexed by

$$k' \in \{(i,j,m,q) : q \in \{1,2,3,4\};\ 1 \leq m \leq D;\ 1 \leq i < j \leq L\}$$

and the feature function is defined as

$$f_{k'}(\mathbf{y}) = \begin{cases} x^{(m)}\ \mathbf{I}(y^{(i)} = 0)\ \mathbf{I}(y^{(j)} = 0) & \text{if } q = 1 \\ x^{(m)}\ \mathbf{I}(y^{(i)} = 1)\ \mathbf{I}(y^{(j)} = 0) & \text{if } q = 2 \\ x^{(m)}\ \mathbf{I}(y^{(i)} = 0)\ \mathbf{I}(y^{(j)} = 1) & \text{if } q = 3 \\ x^{(m)}\ \mathbf{I}(y^{(i)} = 1)\ \mathbf{I}(y^{(j)} = 1) & \text{if } q = 4 \end{cases}$$

Again, we are free to drop one redundant feature function here because of the linearity. Also we could just model the positive and negative correlation by using the following simpler feature function (already in the form of having the redundant one dropped):

$$f_{k'}(\mathbf{y}) = x^{(m)}\mathbf{I}(y^{(i)} = y^{(j)}) \tag{4.6}$$

where $k' \in \{(i, j, m) : 1 \leq i < j \leq L;\ 1 \leq m \leq D\}$.

Figure 4.2 is an example of a CMLF model with 4 labels represented by a factor graph. The feature node connects to all the labels, and each pair of labels and the feature node are linked together.

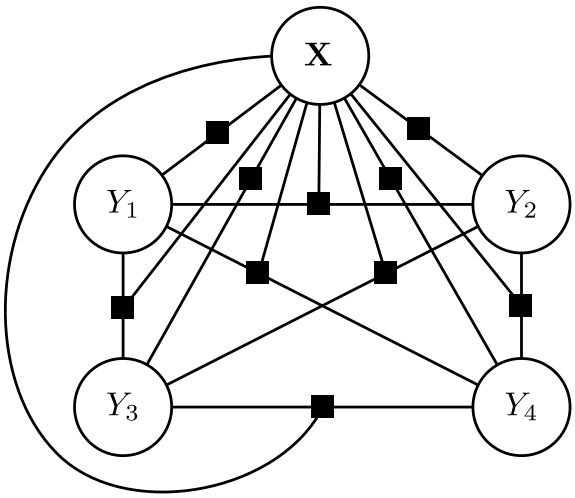

Figure 4.2: The factor graph of a CMLF model with 4 labels.

4.2 Supported heuristic for CRFs

Both training and prediction in CML and CMLF model involve enumerating all possible label combinations for either computing the partition function $Z(\mathbf{x})$ or searching for the label configuration that maximizes $P(\mathbf{y}|\mathbf{x};\mathbf{w})$. In the case of $|\mathcal{Y}| = L$ there are total 2^L label combinations, which makes the training and prediction only workable for small number of L such as $L \leq 10$. Therefore, some approximation methods are proposed in [7]. Here we describe one of the heuristic that assumes all of the label combinations that do not occur in the training data having zero probability, that is, we only consider those combinations (which are called supported combinations) that appear in the training data.

Supposing the supported combination set is $\mathcal{S}$ with $|\mathcal{S}| \ll 2^L$, the partition function $Z(\mathbf{x})$ as in (4.2) is approximated by

$$Z(\mathbf{x}) \approx \sum_{\mathbf{y}\in\mathcal{S}} \exp\left(\sum_k w_k f_k(\mathbf{x},\mathbf{y})\right).$$

Similarly, the prediction is done by

$$\hat{\mathbf{y}} \approx \operatorname*{argmax}_{\mathbf{y}\in\mathcal{S}} P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{S}} \left(\sum_k w_k f_k(\mathbf{x},\mathbf{y})\right)$$

Assume that the number of supported label combinations is S which is essentially $O(N)$. Given the training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \{0,1\}^L$, that is, the number of features is D, the number of labels is L, and the number of training examples is N, the time complexity of training and prediction with the two models CML and CMLF using supported heuristic is analyzed as follows.

For the CML model, there are total $O(DL+L^2)$ feature functions (for both variants described before) as well as the same number of parameters. Supposing a general gradient based optimization is used for parameter estimation, each iteration requires $O(NS(DL+L^2))$ time to compute the gradient of a single parameter. Therefore for all $O(DL+L^2)$ parameters, it takes $O(NS(DL+L^2)^2)$ time for one iteration. Let T denotes the total number of iterations needed to converge, plugging in $S = O(N)$, the overall time complexity for training a CML model is

then $O(N^2L^2(D+L)^2T)$. As for the prediction of CML, it only needs to go through the feature functions without computing the partition function, so the overall time complexity is $O(S(DL + L^2))$ or the upper bound $O(NL(D + L))$.

On the other hand, for the CMLF model there are total $O(DL + DL^2) = O(DL^2)$ feature functions. Therefore the overall time complexity of training a CMLF model using gradient-based methods that takes T iterations to converge is $O(NSD^2L^4T)$ which is upper bounded by $O(N^2D^2L^4T)$. As for the prediction of CMLF, it takes $O(SDL^2) = O(NDL^2)$ time to figure out which supported combination is the most likely one.

Table 4.1 compares the time complexity applying supported method with CML and CMLF mentioned above.

Table 4.1: Time complexity of CML and CMLF with supported heuristic

	training	prediction
CML	$O(N^2L^2(D + L)^2T)$	$O(NL(D + L))$
CMLF	$O(N^2D^2L^4T)$	$O(NDL^2)$

4.3 Gibbs sampling

Besides supported approximation, another well-known approach to attack the intractable nature of complex CRF models such as CML or CMLF is through sampling. Usually the training step that uses sampling technique is done with stochastic gradient descent. Unlike conventional gradient descent that updates in the batch manner, in each iteration stochastic gradient descent updates only according to a small portion of training examples (usually one example in practice). In particular, considering the numerator of the CRF probability $\exp \sum_k w_k f_k(\mathbf{x}, \mathbf{y})$ (which is sometimes called product of potential functions), the parameter updating rule using stochastic gradient descent performs the following according to one training example, say $(\mathbf{x}, \mathbf{y})$, at a time:

$$w_k \leftarrow w_k + \eta \left(f_k(\mathbf{x}_i, \mathbf{y}_i) - \sum_{\mathbf{y}'} P(\mathbf{y}'|\mathbf{x}_i; \mathbf{w}) f_k(\mathbf{x}_i, \mathbf{y}') - \frac{w_k}{N\sigma^2} \right)$$

or written equivalently in the expectation form

$$w_k \leftarrow w_k + \eta \left(f_k(\mathbf{x}, \mathbf{y}) - \mathbb{E}_{\mathbf{y}' \sim P(\mathbf{y}|\mathbf{x};\mathbf{w})} \left[f_k(\mathbf{x}, \mathbf{y}') \right] - \frac{w_k}{N\sigma^2} \right)$$

where η is a tunable learning rate which is usually chosen empirically. An alternative approach similar to perceptron algorithm mentioned in [8] updates parameters in the following way:

$$w_k \leftarrow w_k + \eta \left(f_k(\mathbf{x}, \mathbf{y}) - f_k(\mathbf{x}, \hat{\mathbf{y}}) - \frac{w_k}{N\sigma^2} \right)$$

where $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\mathbf{w})$ takes the labeling that maximizes the conditional probability, which is also the objective of the inference step.

The challenging part of this procedure is to compute either the expectation $\mathbb{E}_{\mathbf{y}' \sim P(\mathbf{y}|\mathbf{x};\mathbf{w})} \left[f_k(\mathbf{x}, \mathbf{y}') \right]$ or the most likely labeling $\hat{\mathbf{y}}$, which is where Gibbs sampling helps. The idea of Gibbs sampling is to efficiently generate a sequence of samples to approximate the joint distribution. The general procedure to sample from the conditional distribution $P(\mathbf{y}|\mathbf{x})$ is to first start from a random guess, say $\mathbf{y}$, and then iteratively sample one component say y_i according to the marginal distribution $P(y^{(i)}|y^{(1)}, y^{(2)}, \ldots, y^{(i-1)}, y^{(i+1)}, \ldots, y^{(N)}, \mathbf{x}))$ which will be shorthanded as $P(y^{(i)}|\mathbf{y}^{(-i)}, \mathbf{x})$ in the following discussion. Usually, a sufficient number of samples at the beginning are ignored (the so-called burn-in period) for the Markov chain to approach its stationary distribution. In the CRF case, we could choose the initial value from the training example. The sampling procedure for a binary component $y^{(i)}$ is to draw $y^{(i)}$ from the distribution:

$$P(y^{(i)} = 1|\mathbf{x}, \mathbf{y}^{(-i)}; \mathbf{w}) = \frac{\exp \sum_k w_k f_k(\mathbf{x}, \mathbf{y}^{(-i)}, y^{(i)} = 1)}{\sum_{j \in \{0,1\}} \exp \sum_k w_k f_k(\mathbf{x}, \mathbf{y}^{(-i)}, y^{(i)} = j)}$$

Notice that the sampling process requires no computation of the exact conditional probability $P(\mathbf{y}|\mathbf{x};\mathbf{w})$ involving the partition function, which is computational intractable, because it is a constant when the parameter $\mathbf{w}$ is fixed. Repeating this procedure for all components over and over again, we will obtain a sequence of labelings which could be used to approximate the $\mathbb{E}_{\mathbf{y}' \sim P(\mathbf{y}|\mathbf{x};\mathbf{w})} \left[f_k(\mathbf{x}, \mathbf{y}') \right]$ by taking the unweighted average of $f_k(\mathbf{x}, \mathbf{y}')$ for all $\mathbf{y}'$ being the elements in the sequence. For the perceptron method or the inference step, we can approximate

the most likely labeling by outputting the very $\hat{\mathbf{y}}$ from the sampled sequence that maximizes the conditional probability $P(\mathbf{y}|\mathbf{x};\mathbf{w})$.

An issue of the implementation of stochastic gradient descent is to choose a good learning rate η during updating to obtain more stable convergence. A common heuristic[1] to adjust the learning rate is to decrease it smoothly as time goes by:

$$\eta = \frac{1}{\lambda(t + t_0)}$$

where t is the number of epochs (updating all parameters once) so far, λ and t_0 are two adjustable parameters which could be determined by cross validating on a small portion of training data sampled uniformly at random.

Follow the same notation convention we used before, the number of features is D, the number of labels is L, and the number of training examples is N. For the CML model, each sampling step takes $O(D+L)$ time because it involves computing $O(D + L)$ feature functions associated with a particular label $y^{(i)}$. Supposing K samples are generated to make the prediction by outputting the label combination from the sampled sequences that maximizes the probability $P(\mathbf{y}|\mathbf{x};\mathbf{w})$, the overall time complexity is therefore $O(K(D + L))$ for prediction. As for training, the stochastic updating with each example involves two steps: generating a sequence of samples to compute $\mathbb{E}_{\mathbf{y}' \sim P(\mathbf{y}|\mathbf{x};\mathbf{w})}[f_k(\mathbf{x}, \mathbf{y}')]$ or $f_k(\mathbf{x}, \hat{\mathbf{y}})$ that takes $O(K(D + L))$ time as in prediction, and updating all $O(DL+L^2)$ parameters. Usually, $K = O(L)$ samples are used for a single update, which leads to $O(DL + L^2)$ update time for one training example. A complete update on all N training examples is usually called an epoch. Assuming the whole training process takes T epochs in total, the overall time complexity for training the CML model is therefore $O(NL(D+L)T)$. Similarly, for CMLF model, since there are $O(DL^2)$ feature functions and each sampling involves computing $O(D + DL) = O(DL)$ of them, the overall time complexity is $O(NDL^2T)$ for training and $O(KDL)$ for prediction. Table 4.2 compares the time complexity of CML and CMLF using Gibbs sampling mentioned above.

A simple implementation of stochastic gradient descent is to iterate a fixed

[1] `http://leon.bottou.org/projects/sgd`

Table 4.2: Time complexity of CML and CMLF using Gibbs sampling

	training	prediction
CML	$O(NL(D+L)T)$	$O(K(D+L))$
CMLF	$O(NDL^2T)$	$O(KDL)$

number of epochs and then output the last model as the result. It is possible to use a particular stopping criterion to either save time or produce better trained model. A straightforward stopping criterion is to directly check the log likelihood $\ell(\mathbf{w}|\mathcal{D})$; however, it involves computing the partition function which is intractable and thus this method is not suitable for larger numbers of labels, or some approximation approach such as supported method should be used. An alternative is to evaluate on hold-out training data according to a metric such as F_1 score and stop when there is no obvious improvement with the metric, although it might not be consistent with the maximum likelihood estimation.

4.4 Tree structured CRF approximation

In the above CRF models, we consider the entire pairwise label correlations and learn a parameter associated with each of them. In reality, however, this might not reflect the actual structure that represents the label correlation. Some of the label pairs might be completely independent. In that case, the corresponding feature functions should be removed from the model to avoid the noise they introduce. While learning the structure itself is a challenging task, even if the exact structure is given the intractable inferencing issue might still exist.

Here we take a step further and trade structure consistency for efficiency. When the label dependencies are tree-structured, there are algorithms such as belief propagation [9] that utilize dynamic programming to perform exact CRF inference in polynomial time. Therefore another heuristic is to learn the tree structure that best approximates the originally fully correlated model, that is, we aim at selecting only a subset of feature functions associated with pairwise label correlations to form a tree structure (or simply a linear chain).

The ideal way to construct the tree is to find the specific tree structure that maximizes the likelihood of training data, which we call maximum likelihood spanning tree problem. However, the additional tree-structure constraint makes the optimization problem far from convex (the objective being the negative log likelihood), so gradient based methods are no longer applicable. In fact, this structure learning problem turns out to be a hard combinatorial optimization problem, so it is necessary to resort to approximation algorithms.

It is sensible to assume that the desired tree structure might take advantage of label correlations to produce a more likely generative model. Therefore we could construct the tree based on the label correlation analysis mentioned earlier. Suppose the correlation matrix C (that comes from any of the criteria such as χ^2 score) is given, where each entry is regarded as the weight of the edge associated with the pairwise label correlation. We can use maximum spanning tree algorithm (it is equivalent to the well-known minimum spanning tree algorithm such as Prim or Kruskal algorithm by negating all the weights of the edges) to build the optimal tree by incrementally including the edge with the largest weight without causing a loop.

One advantage of this graph representation is that we can derive any desired structure according to the graph, such as building a linear chain which could be trained/predicted much faster than a tree or smaller cliques that span the whole graph just as what we will describe later in the clustering framework.

Constructing the optimal linear chain is equivalent to the travelling salesman problem which is known to be NP-hard. In practice, we can use the greedy nearest neighbor heuristic (which should be called farthest neighbor when the edge specifies correlation instead of distance) to approximate the optimal chain, which always chooses the next node having the largest weight on the edge connected to the current node without causing a loop.

When the number of labels is small, more aggressive approaches are possible. For example, we can greedily add edges that maximize the overall likelihood for the partially built tree structure. However, that requires training $O(N^2)$ classifiers which might take too much time for larger number of labels.

Or, we can use a randomized algorithm to sample random linear chain CRFs and choose the one that maximizes the likelihood of training data. The reason for staying with linear chained structure instead of a generalized tree structure is that it is easier to uniformly sample a random chain and also it is faster to perform inference in a linear chain CRF. Other than that, we can use ensemble learning that first samples a bunch of random chains and uses all of them to make predictions and take the majority voting result on each label individually to form the final prediction. This is often considered a robust approach against over-fitting.

Chapter 5

More learning with label correlations

5.1 Multi-class method

Another commonly used framework for multi-label learning is the multi-class method, also known as label powerset method. The multi-class method treats each possible combination (a subset of labels) as a single class and applies the traditional multi-class classification algorithm to predict the most likely label subset. One issue of this method is that the number of classes is exponential in the number of labels. Specifically, if there are L labels, the multi-class method has to deal with 2^L classes. Therefore, such property makes this method intractable and cannot scale-up for larger label set.

From the label correlation point of view, multi-class method utilizes conditional label correlation. Each class that corresponds to a label combination implicitly models label correlations in a more general manner (treating the entire correlated label combination as a whole) than the pairwise way described previously. The multi-class classifier relates the feature to each correlated label set and therefore is a framework that exploit conditional label correlation.

In practice, most discriminative multi-class classification algorithms such as multi-class SVM only consider classes having non-zero number of training in-

stances, that is, in the generative point of view it assumes that all of the label combinations that do not appear in the training data have zero probability. In this case, the number of classes involved is down to the total number of label subsets that occur in the training data, which is no more than the size of the training data itself. This is the same idea as the supported heuristic for CRF approximation. It is usually the case that multi-label training data is not very large, and the size of the supported combinations is almost the size of the training data. Under this situation, the degenerated multi-class method will become nearest neighbor method. Also, one of the disadvantages of this method is that since each class might contain only a few number of training instances (sometimes there is only one) which makes the resulting classifier less generalizable. Therefore, multi-class method usually works better when the size of label set is not too large.

5.2 Multi-label k-nearest neighbor

k-nearest neighbor (kNN) is an instance-based learning method that classifies objects based on k closest training examples in the feature space. When the training data is not too large, kNN is a very efficient method because kNN does nothing in the training stage (it is a lazy learning method) and runs in time linear to the number of training data in the prediction stage.

One of the famous property of kNN is its strong theoretical consistency guarantee. As the amount of data approaches infinity, nearest neighbor algorithm (1NN) is guaranteed to yield an error rate no worse than twice the Bayes error rate, the minimum achievable error rate given the distribution of the data [10].

The traditional single label kNN could be easily extended to the multi-label scenario by performing individual voting on each label. Usually we use Euclidean distance as the distance metric for simplicity; therefore feature normalization is crucial to obtain good classification result. Our experiment shows that normalizing the feature vector according to its L_2 norm makes the best classification results. In practice, we always choose k to be an odd number for the voting convenience (to avoid a tie) which should be determined by cross validation.

It is worth noting that this multi-label kNN algorithm could be seen as exploiting conditional label correlation because it directly use the labeling outcomes of nearby training examples which already reflect label correlations on their own as in the case of multi-class method. Besides, the distance is measured with the feature vectors so we say kNN makes use of conditional label correlations.

A more sophisticated kNN-based multi-label learning algorithm was proposed in [11] which performs a local probabilistic reasoning with nearby instances to make multi-label prediction.

5.3 Label space reduction

When the number of labels is huge, even the binary relevance method is not applicable due to the large amount of binary classifiers to train. Therefore we have to seek for solutions with running time less than the number of labels.

A label space transformation framework [12] could be used to deal with this situation. Given the training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ where $\mathbf{y}_i \in \{0,1\}^L$ with L labels, first the labels are transformed from the original label space $\{0,1\}^L$ into another space $\mathbb{R}^{L'}$ with $L' \ll L$ with certain efficient mapping function p. That is, the training data is converted into $\{(\mathbf{x}_i, \mathbf{r}_i)\}_{i=1}^N$ where $\mathbf{r}_i = p(\mathbf{y}_i) \in \mathbb{R}^{L'}$. Usually, the coordinates of the transformed space are independent to one another. Therefore L' regression functions f_j could then be trained with the transformed data $\{(\mathbf{x}_i, r_i^{(j)})\}_{i=1}^N$ for $j = 1, \ldots, L'$ independently. For simplicity, linear regression is used here although other regression methods could also be considered. In the prediction step, given a feature vector $\mathbf{x}$, all L' predicted values $f_1(\mathbf{x}), \ldots, f_{L'}(\mathbf{x})$ are computed and then use these L' values to transformed back to the original binary label space with another efficient mapping function q.

Two methods are described below, which both use linear transformation functions as p to convert the original label space to the lower dimensional real space and thresholding on the results of p^{-1} in the real space back to the label space. The first method, label PCA, makes use of unconditional label correlation, while the other method, feature/label CCA, exploits conditional label correlation

that also takes features into account.

5.3.1 Label PCA

Principal component analysis (PCA) is a widely used dimensionality reduction technique that linearly transforms a number of possibly correlated variables into a set of uncorrelated variables called principal components. Formally speaking, PCA transforms the data to a new orthogonal coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (first principal component), the second greatest variance on the second coordinate, and so on. Equivalently, given the input data $\{\mathbf{x}_i\}_{i=1}^{N}$ and L' orthogonal principal component basis vectors $\{\mathbf{p}_j\}_{j=1}^{L'}$, PCA minimizes the reconstruction error

$$\mathcal{E} = \sum_{i=1}^{N} \left\| \mathbf{x}_i - \sum_{j=1}^{L'} (\mathbf{p}_j^{\mathsf{T}} \mathbf{x}_i) \mathbf{p}_j \right\|^2 = \sum_{i=1}^{N} \left(\mathbf{x}_i^2 - \sum_{j=1}^{L'} (\mathbf{p}_j^{\mathsf{T}} \mathbf{x}_i)^2 \right)$$

In the multi-label classification problem, we can regard the label vector $\mathbf{y} \in \{0, 1\}^L$ as the coordinates in the real space. Then PCA is applied on the label vector so that principal components capture the unconditional label correlations and project the original label vector into a lower dimensional real vector. The training step of label PCA is described as Algorithm 2.

Notice that this method is similar to [12] although that does not center the label vectors and directly decomposes the stacked training label vectors by SVD to construct projecting bases. Centering is necessary to find the principal components that maximize the variance (or minimize the reconstruction error) in the shifted coordinate system.

The prediction step is just applying the reverse procedure that projects the regression results back to the original coordinate and rounds into binary label values as Algorithm 3.

A related work introduced in [13] also exploits unconditional label correlation except it performs random projection and applies compressed sensing technique to reconstruct the predicted labels from the regression results with k-sparse constraint taken into account.

Algorithm 2: Training of label PCA

Input: training data $\mathcal{D} = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{N}$ where $\mathbf{x}_i \in \mathbb{R}^D$, $\mathbf{y}_i \in \{0,1\}^L$, the reduced dimension L'

Output: label mean $\boldsymbol{\mu}$, a set of projecting bases $\{\mathbf{p}_j\}_{j=1}^{L'}$, a set of regression functions $\{f_j\}_{j=1}^{L'}$

compute sample mean of label vectors $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i$.

compute sample covariance matrix $\Sigma = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \boldsymbol{\mu})(\mathbf{y}_i - \boldsymbol{\mu})^{\mathsf{T}}$.

compute L' normalized right eigenvectors of Σ associated with L' largest eigenvalues as the projecting bases $\mathbf{p}_1, \ldots, \mathbf{p}_{L'}$ where $\mathbf{p}_j \in \mathbb{R}^L$.

for $j = 1$ **to** L' **do**

train linear regression function $f_j : \mathbb{R}^D \to \mathbb{R}$ with training data $\{(\mathbf{x}_i, \mathbf{p}_j^{\mathsf{T}}(\mathbf{y}_i - \boldsymbol{\mu}))\}_{i=1}^{N}$.

end

Algorithm 3: Prediction of label PCA

Input: label mean $\boldsymbol{\mu}$, a set of projecting bases $\{\mathbf{p}_j\}_{j=1}^{L'}$, a set of regression functions $\{f_j\}_{j=1}^{L'}$

Output: multi-label classification result $\hat{\mathbf{y}} \in \{0,1\}^L$

$\mathbf{r} = \sum_{j=1}^{L'} f_j(\mathbf{x})\mathbf{p}_j + \boldsymbol{\mu}$, note that $\mathbf{r} \in \mathbb{R}^L$.

round $\mathbf{r}$ to binary vector $\hat{\mathbf{y}}$ by element-wisely assigning $\hat{y}^{(j)} = 1$ if $\mathbf{r}^{(j)} \geq 0.5$, $\hat{y}^{(j)} = 0$ otherwise for $j = 1$ to L.

5.3.2 Feature/label CCA

Canonical correlation analysis (CCA) [14] is a common technique for measuring the linear relationship between two multi-dimensional variables. Specifically, consider two (centered) variables $\mathbf{x}$ and $\mathbf{y}$, CCA gives two sets of basis vectors such that the correlation between the projections $\mathbf{x}$ and $\mathbf{y}$ onto these bases are mutually maximized. Supposing one set of basis is used, say $\mathbf{w}_x$ and $\mathbf{w}_y$, the projection of $\mathbf{x}, \mathbf{y}$ are therefore $x = \mathbf{w}_x^\mathsf{T}\mathbf{x}$ and $y = \mathbf{w}_y^\mathsf{T}\mathbf{y}$ respectively. So CCA finds the optimal basis $\mathbf{w}_x, \mathbf{w}_y$ that maximizes the correlation coefficient of x, y:

$$\rho = \frac{\mathbb{E}[xy]}{\sqrt{\mathbb{E}[x^2]\mathbb{E}[y^2]}} = \frac{\mathbb{E}[\mathbf{w}_x^\mathsf{T}\mathbf{x}\mathbf{y}^\mathsf{T}\mathbf{w}_y]}{\sqrt{\mathbb{E}[\mathbf{w}_x^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{w}_x]\mathbb{E}[\mathbf{w}_y^\mathsf{T}\mathbf{y}\mathbf{y}^\mathsf{T}\mathbf{w}_y]}}$$

Since CCA makes use of two views of the same set of objects and projects them onto a lower dimensional space in which the projections are maximally correlation, it can be used to perform supervised dimensionality reduction. In the multi-label setup, each instance has two views: one is the feature vector, and the other is the multi-label outcome represented as the binary label vector. Using CCA, we can project the label vector onto a subspace that preserves most predictability from the (projected) feature. Different from label PCA, feature/label CCA exploits conditional label correlation that takes feature vectors into account. As a by-product, this method could also be used to achieve dimensionality reduction on the feature vector.

Algorithm 5 illustrates the detailed training step of feature/label CCA. The normalized canonical correlation bases are computed through solving an eigen problem that involves both within-sets covariance matrices Σ_{XX}, Σ_{YY}, which are assumed to be non-singular, and between-sets covariance matrices Σ_{XY}, Σ_{YX}.

Similar to label PCA, Algorithm 5 describes the prediction step of feature/label CCA.

5.4 Label clustering

We have described several multi-label learning algorithms that exploit label correlations. Some of them are intractable such as CRFs and therefore they are

Algorithm 4: Training of feature/label CCA

Input: training data $\mathcal{D} = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^N$ where $\mathbf{x}_i \in \mathbb{R}^D$, $\mathbf{y}_i \in \{0,1\}^L$, the reduced dimension L'

Output: label mean $\boldsymbol{\mu}_Y$, a set of projecting bases $\{\mathbf{p}_j\}_{j=1}^{L'}$, a set of regression functions $\{f_j\}_{j=1}^{L'}$

compute feature mean $\boldsymbol{\mu}_X = \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i$.

compute label mean $\boldsymbol{\mu}_Y = \frac{1}{N}\sum_{i=1}^N \mathbf{y}_i$.

compute covariance matrices:

$$\begin{aligned}
\Sigma_{XX} &= \tfrac{1}{N}\textstyle\sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu}_X)(\mathbf{x}_i - \boldsymbol{\mu}_X)^\mathsf{T}, \\
\Sigma_{YY} &= \tfrac{1}{N}\textstyle\sum_{i=1}^N (\mathbf{y}_i - \boldsymbol{\mu}_Y)(\mathbf{y}_i - \boldsymbol{\mu}_Y)^\mathsf{T}, \\
\Sigma_{XY} &= \tfrac{1}{N}\textstyle\sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu}_X)(\mathbf{y}_i - \boldsymbol{\mu}_Y)^\mathsf{T}, \\
\Sigma_{YX} &= \tfrac{1}{N}\textstyle\sum_{i=1}^N (\mathbf{y}_i - \boldsymbol{\mu}_Y)(\mathbf{x}_i - \boldsymbol{\mu}_X)^\mathsf{T}.
\end{aligned}$$

compute L' normalized right eigenvectors of $(\Sigma_{YY}^{-1}\Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY})$ associated with L' largest eigenvalues as the projecting bases $\mathbf{p}_1, \ldots, \mathbf{p}_{L'}$.

for $j = 1$ **to** L' **do**

train linear regression function $f_j : \mathbb{R}^D \to \mathbb{R}$ with training data $\{(\mathbf{x}_i, \mathbf{p}_j^\mathsf{T}(\mathbf{y}_i - \boldsymbol{\mu}_Y))\}_{i=1}^N$.

end

Algorithm 5: Prediction of feature/label CCA

Input: label mean $\boldsymbol{\mu}_Y$, a set of projecting bases $\{\mathbf{p}_j\}_{j=1}^{L'}$, a set of regression functions $\{f_j\}_{j=1}^{L'}$

Output: multi-label classification result $\hat{\mathbf{y}} \in \{0,1\}^L$

$\mathbf{r} = \sum_{j=1}^{L'} f_j(\mathbf{x})\mathbf{p}_j + \boldsymbol{\mu}_Y$, note that $\mathbf{r} \in \mathbb{R}^L$.

round $\mathbf{r}$ to binary vector $\hat{\mathbf{y}}$ by element-wisely assigning $\hat{y}^{(j)} = 1$ if $\mathbf{r}^{(j)} \geq 0.5$, $\hat{y}^{(j)} = 0$ otherwise for $j = 1$ to L.

not applicable when the number of labels is large. Meanwhile, when the number of labels is large, some of them might not be correlated at all. Take CRFs for example, modeling those independent label pairs would results in a poorer model when strong regularization is applied to smooth the model. Therefore, properly decomposing the original pairwise label relationship into several less correlated label clusters and then applying multi-label learning algorithms such as CRFs independently on each cluster might not only simplify the overall model complexity so that the risk of over-fitting is reduced but also greatly speed up the whole process.

As we mentioned earlier, label clustering is a heuristic to approximate the much harder structure learning problem. We can also take the advantage of the information from correlation matrix to achieve our goal. Basically speaking, we regard the correlation matrix as a proximity measure and apply clustering algorithms to decompose label set into clusters.

Agglomerative hierarchical clustering method is suitable to separate the label set into disjoint clusters. For example, we can use single linkage clustering algorithm with the following linkage function $\text{corr}(P, Q)$ to define the correlation between two clusters $P, Q \subseteq \mathcal{Y}$ given the correlation matrix C:

$$\text{corr}(P, Q) = \max_{p \in P, q \in Q} C_{pq}.$$

Notice that since correlation matrix describes the similarity between any two labels, it is opposite to a distance measure that has smaller value when two labels are similar to (or close to) each other. So the linkage function defined on correlation is different from the function defined on distance that uses the min operator instead.

Having this linkage definition, a greedy method similar to Kruskal's minimum spanning tree algorithm could be used to merge labels into groups incrementally. The merging process continues until the merged cluster exceeds a size that is computable for the multi-label learning algorithms.

One disadvantage of this method is that it might end up with several small clusters (with only one label or two). A workaround is to continue the merging process, leaving those clusters which are large enough untouched, until the cor-

relation score is smaller than certain threshold. This is a reasonable adjustment that takes more advantage of label correlations while still keeping the labels within a manageable scale. Another workaround is to use a normalized graph cut algorithm to iteratively separate labels into two balanced sets, until all clusters are within a manageable size. However, although the decomposed clusters are more balanced, the balancing constraint might result in correlated label pairs being cut off or including some less correlated pairs.

Another approach is to generate overlapping clusters and take the voting result for each label as the final prediction. We can also use the single linkage rule while adapting Prim's minimum spanning tree algorithm to grow a single cluster starting from each label until the cluster reaches the desired size or the correlation score is under certain threshold. If there are identical clusters, only one copy is kept. This method might end up having exactly L clusters where L is the number of labels because cluster growing procedure is performed on each label. One advantage is that each label could take advantage of most of its highly correlated other labels and the voting step could provide a more fair judge.

To clarify, although we use the term clustering and also algorithms for the traditional unsupervised clustering problem, the goal of clustering here is actually not quite the same. In our case, clustering is for decomposing the original large scale problem into several subproblems. It is possible that two different clusters might be strongly correlated and they are separated only because the union of them is too large to compute. We do require the members within a cluster to be highly correlated so that label correlations could be fully utilized.

The label clustering framework is similar to the idea of random k-labelset (RAkEL) [15] which randomly samples a bunch of overlapping clusters with the same size. Although our approach clusters labels in a more controlled way instead of a randomized manner, it is based on pairwise correlation measures, which might not reflect actual correlations that involve more than two labels, or are not linear. RAkEL might be a simpler (and more generalizable) alternative for label set decomposition if that is the case.

Chapter 6

Experimental design

6.1 CAL500 data set

Computer Audition Lab 500 (CAL500) [2] is a multi-label data set created by the UCSD Computer Audition Lab[1]. CAL500 is a corpus of a 502 tracks of western popular songs, each of which is described in two kinds of representation: semantic annotation and musical features.

The annotation part involves 135 semantic concepts including instruments, vocal characteristics, genres, emotions, acoustic qualities, and usage terms. These concepts are then expanded into 237 words by splitting all bipolar concepts into two individual words (or labels in our naming convention). Each song is manually annotated by at least three human labelers. Once a word has at least 80% agreement among all labelers, it is assigned to that song. In the end, all concepts that are associated with fewer than eight songs are pruned and therefore the total word count is reduced from 237 to 174. In our naming conventions these 174 words are the label set $\mathcal{Y}$.

As for the music features, each song is made from a sequence of Mel-frequency cepstral coefficient (MFCC) vectors of a sliding overlapping window over the song. A delta cepstrum vector is computed by appending the first and second derivatives of each MFCC to the original MFCCs. Since only the first 13

[1]Available at `http://mkl.ucsd.edu/dataset/cal500`

MFCCs are used, and 10000 delta cepstrum vectors are randomly sampled from each song, it ends up being 10000 39-dimensional feature vectors as the musical representation.

As a summary, CAL500 is a set of 502 songs and each of which is represented as a set of 10000 39-dimensional feature vectors together with 174-dimensional binary label vector. Our goal is to use the musical feature of a given song to predict the binary label vector, that is, to decide which conceptual words should be associated with that song.

6.1.1 Label cardinality

CAL500 has strong consistency in terms of label cardinality [16], the average number of labels associated with a training example. Given the training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ where $\mathbf{y}_i \in \{0,1\}^L$, label cardinality is defined as

$$\text{label cardinality} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} y_i^{(j)}$$

Table 6.1 shows that the label cardinality of CAL500 is roughly a constant because of the small standard deviation. It also shows the label vector is sparse: only 15% of the labels are presented on average.

Table 6.1: Label cardinality statistics of CAL500

max	min	mean	standard deviation	total # labels
48	13	26.0438	5.7484	174

This property is closely related to the concept of k-sparse (for CAL500 it is 48-sparse) in the coding theory, which is a constraint for signal recovery in compressive sensing, the technique that [13] used for multi-label classification. This property could also be addressed in another ways in multi-label learning such as serving as a constraint so that the number of positive labels is close to the average training label cardinality on the argmax operation for inference as in (4.4) to filter out certain unreasonable label combinations.

6.2 Evaluation of multi-label classification

Most evaluation metrics for multi-label classification are extended from traditional single label ones. Here we describe three widely used metrics that are derived from contingency tables, one for each label, and three other metrics that are not so popular.

Given the training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ and the corresponding predictions $\{\hat{\mathbf{y}}_i\}_{i=1}^N$ where $\mathbf{y}_i, \hat{\mathbf{y}}_i \in \{0,1\}^L$ meaning that there are total L labels, six measures are used to evaluate the performance of multi-label classification in our experiment: accuracy, micro-averaged F_1 score, macro-averaged F_1 score with respect to labels, macro-averaged F_1 score with respect to instances, micro-averaged Matthews correlation coefficient, and macro-averaged Matthews correlation coefficient.

Subset accuracy is the exact match ratio which is a direct extension of the accuracy measure of standard classification evaluation:

$$\text{subset accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{I}(\mathbf{y}_i = \hat{\mathbf{y}}_i)$$

where $\mathbf{I}(\cdot)$ is the indicator function having the value 1 if the argument is true and the value 0 otherwise.

A disadvantage of subset accuracy measure is that it does not take partial matches into account, so it might be possible that predictions are only wrong on a single component but ends up with poor subset accuracy. Therefore, a relaxed version called average accuracy (or accuracy) is used which is closely related to Hamming loss:

$$\text{average accuracy} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\mathbf{I}(y_i^{(j)} = \hat{y}_i^{(j)})$$

where we use the notation $y^{(j)}$ to denote the j-th component of vector $\mathbf{y}$. We will keep using this indexing convention in the following text.

Another commonly used performance measure is F_1 score, which is the harmonic mean of precision and recall. Mathematically, harmonic mean is always smaller than geometric mean or arithmetic mean, which could be regarded as a smoothed minimum function. Therefore, optimizing on the F_1 score to some extent

implies maximizing the worse of precision and recall. The F-score can be generalized into a weighted averaged version between precision and recall parameterized by non-negative β to emphasize one over another:

$$F_\beta = (1+\beta^2)\frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

but in the experiments, we stay with the balanced F_1 score with $\beta = 1$.

For a single label indexed by j, the precision and recall are defined as:

$$\text{precision} = \frac{\sum_{i=1}^{N} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{i=1}^{N} \hat{y}_i^{(j)}} \qquad \text{recall} = \frac{\sum_{i=1}^{N} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{i=1}^{N} y_i^{(j)}} \tag{6.1}$$

From the definition, we can derive the formula of F_1 score for the single label j:

$$\frac{2\ \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\sum_{i=1}^{N} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{i=1}^{N} \hat{y}_i^{(j)} + \sum_{i=1}^{N} y_i^{(j)}}$$

To extend the single label F_1 score to multi-label classification, one measure is to treat every entry of the label vector as an individual instance regardless of label distinction, which is called micro-averaged F_1 score:

$$\text{micro } F_1 \text{ score} = \frac{2\sum_{j=1}^{L}\sum_{i=1}^{N} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{j=1}^{L}\sum_{i=1}^{N} \hat{y}_i^{(j)} + \sum_{j=1}^{L}\sum_{i=1}^{N} y_i^{(j)}} \tag{6.2}$$

The other way to apply F_1 score for multi-label case is to compute the average of F_1 scores of each label, which is usually called macro-averaged F_1 score in the literature:

$$\text{macro } F_1 \text{ score} = \frac{1}{L}\sum_{j=1}^{L} \frac{2\sum_{i=1}^{N} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{i=1}^{N} \hat{y}_i^{(j)} + \sum_{i=1}^{N} y_i^{(j)}} \tag{6.3}$$

It is possible that for certain labels, none of the instances have a positive label value or positive label prediction, especially when we do cross validation in a less controlled way splitting test data. That is, we might have $\sum_{i=1}^{N} \hat{y}_i^{(j)} + \sum_{i=1}^{N} y_i^{(j)} = 0$ for some label index j. In this case, we just ignore those labels when computing macro-averaged F_1 score

$$\text{macro } F_1 \text{ score} = \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} \frac{2\sum_{i=1}^{N} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{i=1}^{N} \hat{y}_i^{(j)} + \sum_{i=1}^{N} y_i^{(j)}}$$

where $\mathcal{S}$ is the set of label indices such that $\sum_{i=1}^{N} \hat{y}_i^{(j)} + \sum_{i=1}^{N} y_i^{(j)} \neq 0$ for all $j \in \mathcal{S}$.

Notice that since micro-averaged F_1 score averages over all predictions regardless label distinction, from the definition in (6.2) we can see that micro-averaged F_1 score tends to over-emphasize performance on the labels that are mostly positive, because they dominate the summed denominator and numerator in (6.2). On the other hand, macro-averaged F_1 score tends to over-emphasize performance on the labels that have the fewest positive outcomes. Table 6.2 illustrates the statistics of $\{\sum_{i=1}^{N} y_i^{(j)}\}_{j=1}^{L}$ in the same notation used before and shows that this unbalanced label occurrence rate is indeed the case of CAL500. Therefore we should examine both metrics when evaluating the performance of multi-label classification.

Table 6.2: Label statistics of CAL500

max	min	mean	standard deviation	total # instances
444	5	75.1379	81.1606	502

As mentioned earlier, CAL500 has strong constant label cardinality property, so instead of calculating the macro-averaged F score with respect to each label, we could compute the macro-average with respect to each example to get a more stable measurement:

$$\text{macro } F_1 \text{ wrt instance} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \sum_{j=1}^{L} \hat{y}_i^{(j)} y_i^{(j)}}{\sum_{j=1}^{L} \hat{y}_i^{(j)} + \sum_{j=1}^{L} y_i^{(j)}} \tag{6.4}$$

So far, all the metrics described are formulated with part of the information from the confusion matrix which summarizes the relationship between actual value and the corresponding prediction with four numbers: true positive (TP), false positive (FP), false negative (FN), and true negative (TN). From (6.1) we can see that precision is actually $TP/(TP + FP)$ and recall is $TP/(TP + FN)$. Both precision and recall do not take all four statistics into account which might not be the most fair way to evaluate the classification performance. Correlation is also a standard way to measure the classification performance [17]. Instead of the most commonly used Pearson correlation coefficient, we choose to use Matthews correlation coefficient (MCC) [18] which is generally regarded as one of the best

summarizations of a confusion matrix that utilizes all four statistics in the confusion matrix:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

If any of the four sums in the denominator is zero, Matthews correlation coefficient is set to zero to be consistent with the limiting value.

Similar to F score, Matthews correlation coefficient could be extended to the multi-label case by taking the micro average and macro average. Micro-averaged MCC calculates the confusion matrix over all the components of the label vector regardless of label distinction. On the other hand, macro-averaged MCC computes the MCC one for each instance (we only consider macro-averaged MCC with respect to instance which takes the label cardinality property into account) and then averages them.

6.3 Feature preprocessing

Since the original musical feature of CAL500 is represented as a set of 10000 39-dimensional real vectors that is too large to compute with efficiently, we first reduce the dimensionality by running k-means algorithm to find 100 cluster centroids from all 39-dimensional feature vectors in the entire corpus. Then for each song we quantize all 10000 feature vectors into these 100 centroids and generate corresponding bag-of-words representation. Having this 100-dimensional bag-of-words vector, we need to further normalize it to achieve the best performance for each learning algorithm.

Given the bag-of-words feature vector $\mathbf{x}$, four normalization methods are considered:

$\mathbf{L_2}$ normalization Scale the feature vector $\mathbf{x}$ to unit length by dividing $\mathbf{x}$ by its L_2 norm to get $\mathbf{x}' = \mathbf{x}/\|\mathbf{x}\|_2$.

$\mathbf{L_1}$ normalization Scale the feature vector $\mathbf{x}$ so that the absolute value of all components are summed to one (in the bag-of-words case, the feature vector

is non-negative). This can be done by dividing $\mathbf{x}$ by its L_1 norm to get $\mathbf{x}' = \mathbf{x}/\|\mathbf{x}\|_1$.

Rescaling Given the training data, we linearly rescale the components to $[-1, +1]$. Specifically, given the feature data $\{\mathbf{x}_i\}_{i=1}^N$

$$x^{(j)\prime} = \frac{2\left(x^{(j)} - \min_i x_i^{(j)}\right)}{\max_i x_i^{(j)} - \min_i x_i^{(j)}} - 1, \quad \text{for all } j = 1, \ldots, 100$$

Z-score Normalize the feature vector into zero mean and unit variance. Specifically, given the feature data $\{\mathbf{x}_i\}_{i=1}^N$ where the j-th entry $\{x_i^{(j)}\}_{i=1}^N$ has mean $\mu^{(j)}$ and standard deviation $\sigma^{(j)}$

$$x^{(j)\prime} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}, \quad \text{for all } j = 1, \ldots, 100$$

The normalization method is chosen by 5-fold cross validation on micro-averaged F_1 score. Notice that micro-averaged F_1 score is also the metric for choosing parameters in some algorithms such as selecting k for kNN.

6.4 Experimental setup

The CAL500 corpus is divided into a training set of 452 instances and a test set of 50 instances. The training set is manually sampled so that there are at least one positive and one negative example for each label. All the multi-label algorithms are trained on the training set and evaluated on the test set.

Six metrics we mentioned earlier are used to evaluate multi-label classification performance: average (abbreviated "Acc"), micro-averaged F_1 score (abbreviated "μF_1"), macro-averaged F_1 score with respect to label (abbreviated "$\mathrm{M}_\ell F_1$"), macro-averaged F_1 score with respect to instance (abbreviated "$\mathrm{M}_i F_1$"), micro-averaged Matthews correlation coefficient (abbreviated "μMC"), and macro-averaged MCC with respect to instance (abbreviated "M_iMC").

Our experiment breaks into two parts. The first part aims at demonstrating all the multi-label classification methods we mentioned earlier including those intractable ones on a small subset of CAL500 labels. The second part works on the

entire 174 labels, so instead of running the exact CRF method we only evaluate approximation methods. Also the clustering framework is examined.

Both of these two parts are compared with the following baseline methods: binary relevance, k-nearest neighbors method (abbreviated "kNN"), and multi-class method. We use logistic regression (abbreviated "logisReg") and SVM with linear kernel (abbreviated "linear SVM") and RBF kernel (abbreviated "RBF SVM") as our base binary classifiers for binary relevance framework.

In our experiments, all the parameters involved such as the regularization weight are selected by grid search according to the cross validation result on micro-averaged F_1 score. Also notice that multi-class method uses one-versus-one linear SVMs.

As for the CRF related methods including CML and CMLF, instead of modeling all four possible combinations of a label pair, we only use the feature functions that capture positive and negative correlations as in (4.5) and (4.6) for simplicity.

All the learning algorithms we use are implemented in MATLAB. We use LIBSVM [19] as the SVM library. All the optimizations are done by `minFunc`[2] which is an unconstrained optimization solver in MATLAB implemented by Mark Schmidt. We use L-BFGS as the optimization option to perform gradient descent. Tractable exact inference in CRF tree/chain is done by UGM[3] which is a MATLAB toolbox for probabilistic undirected graphical models implemented also by Mark Schmidt.

[2] `minFunc` is available at `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`
[3] UGM is available at `http://people.cs.ubc.ca/~schmidtm/Software/UGM.html`

Chapter 7

Experimental results

7.1 Small label set experiment

Since some of the models described earlier such as CRFs are intractable and not suitable for large label set learning, in order to examine how these methods perform, we manually select 8 labels within the emotion category from the total 174 labels of CAL500 to test on. These labels are the 2nd, 6th, 12th, 22nd, 24th, 32nd, 34th, 36th label of CAL500, that are chosen to make the resulting label data not too sparse as a multi-class data set. The label cardinality for this setup is 3.7012 ± 1.6091.

Other than the baseline methods mentioned earlier, including binary relevance, kNN, and multi-class method, we examine the multi-class method and two CRF models, CML and CMLF, with exact training and inference. The feature normalization result shows that multi-class and kNN are best to have the feature vector normalized with the L_2 normalization method, while all the others are best normalizing the feature with rescaling method.

From Table 7.1 we can see that CML outperforms all the other methods and the more complicated CMLF also does a good job. However, both kNN and multi-class perform poorly comparing to all the binary relevance methods on every metric we use. Although kNN and multi-class method do not perform as good result as CRF models, both of them are much faster for both training and prediction. In our experiment involving 8 labels, it takes more than an hour to train

and predict with CML model, and CMLF doubles the time, while kNN takes couple of seconds and multi-class takes less than a second (because multi-class solver is implemented efficiently in compiled executable instead of MATLAB). Moreover, the non-linear RBF kernel does not improve performance, probably because the feature dimensionality is high enough for linear classifiers to perform well.

Table 7.1: Performance evaluation on a 8-label subset of CAL500

	Acc	μF_1	$\mathrm{M}_\ell F_1$	$\mathrm{M}_i F_1$	μMC	M_iMC
linear SVM	0.6925	0.6516	0.6227	0.6351	0.3764	0.3658
RBF SVM	0.6875	0.6439	0.5948	0.6464	0.3655	0.3576
logisReg	0.6925	0.6535	0.6120	0.6422	0.3772	0.3631
kNN	0.6500	0.6000	0.5534	0.5969	0.2889	0.3141
multi-class	0.6700	0.6229	0.5977	0.5910	0.3296	0.3352
CML	**0.7200**	**0.6744**	**0.6597**	**0.6578**	**0.4294**	**0.4228**
CMLF	0.7000	0.6648	0.6526	0.6332	0.3936	0.3751

7.2 Entire CAL500 experiment

In this section, we examine the performance of multi-label learning algorithms on entire 174 labels of CAL500 data set. To begin with, we first show how the metrics we use for evaluation behave with some trivial predictions on the entire data as another baseline. Table 7.2 shows the result of random predictions that all components of the label vector are sampled with certain label occurrence rate, $P(y^{(j)} = 1)$ for all $j = 1, \ldots, L$, identically applying to all the labels. The result shows that it is not proper to evaluate only on certain metric. Since positive labels are sparse in general, the higher the label occurrence rate, the lower the accuracy it ends up with. On the other hand, all the F_1 scores shows the opposite: the scores are higher as the occurrence rate increases. As for Matthews correlation, all of them are close to zero which reflects the fact that the predictions are completely random.

Another trivial baseline we consider is a modified version of the random predictor we just mentioned except that each label is randomly predicted with the occurrence rate consistent with the training data, that is, to follow the distribution

Table 7.2: Performance of random prediction with specified label occurrence rate

$P(y=1)$	Acc	μF_1	$\mathrm{M}_\ell F_1$	$\mathrm{M}_i F_1$	μMC	M_iMC
0	0.8511	0.0000	0.0000	0.0000	0.0000	0.0000
0.25	0.6768	0.1873	0.1438	0.1865	0.0013	0.0042
0.5	0.5037	0.2347	0.1964	0.2325	0.0097	0.0102
0.75	0.3308	0.2522	0.2176	0.2507	0.0117	0.0117
1	0.1489	0.2591	0.2300	0.2576	0.0000	0.0000

$P(y^{(j)}=1) = \sum_{i=1}^{N} y_i^{(j)}/N$. The result is shown in Table 7.3. As we can see, this controlled random prediction significantly improves the performance comparing to the previous random version. In fact, this naive method employs the same concept as other more complicated generative models except our random prediction takes only individual label frequency into account without making use of feature information or even label correlations.

Table 7.3: Performance of random prediction with label occurrence rate consistent with training data

Acc	μF_1	$\mathrm{M}_\ell F_1$	$\mathrm{M}_i F_1$	μMC	M_iMC
0.7907	0.3115	0.0742	0.3103	0.1882	0.1905

7.3 Results of multi-label learning methods

In the experiment that involves the entire label set of CAL500, we evaluate two kinds of CRF approximation algorithms: supported heuristic, and several CRF tree/chain approximations. Moreover, two label space reduction methods, label PCA and feature/label CCA, are evaluated. In the end, we examine the clustering framework that decomposes the entire CAL500 label set into several subsets with nearly equal size.

Before showing the result, we want to first point out that the number of supported label combinations of the entire CAL500 data set is exactly 502, the same size as that of the data. That is to say, there is no redundancy at all to

provide statistically enough of evidence to distinguish the difference among label combinations. As a result, the behavior of multi-class method will be identical to nearest neighbor algorithm (1NN).

Because of the intractable nature of CRF, we only run the CML model with the supported heuristic on the entire CAL500 label set (abbreviated "supported CML").

Table 7.4: Performance comparison of multi-label learning algorithms on entire CAL500 labels

	Acc	μF_1	$\mathrm{M}_\ell F_1$	$\mathrm{M}_i F_1$	μMC	M_iMC
linear SVMs	0.8652	0.4417	0.1793	0.4342	0.3832	0.3962
RBF SVM	0.8702	0.4289	0.1516	0.4272	0.3874	0.3913
logisReg	**0.8703**	0.4508	0.1667	0.4447	**0.4006**	**0.4082**
kNN	0.8620	0.4284	0.1701	0.4255	0.3676	0.3714
multi-class	0.8229	0.4084	0.2288	0.4013	0.3043	0.3062
supported CML	0.8245	**0.4678**	**0.2684**	**0.4611**	0.3663	0.3664
χ^2 chain	0.8228	0.4078	0.2230	0.4042	0.3036	0.3160
χ^2 tree	0.8286	0.4104	0.2146	0.4062	0.3104	0.3227
sampled χ^2 chain	0.8226	0.3994	0.2139	0.3954	0.2954	0.3062
random chain	0.8222	0.4425	0.2239	0.4412	0.3384	0.3492
sampled ML chain	0.8252	0.4427	0.2217	0.4418	0.3399	0.3506
chain ensembles	0.8243	0.4490	0.2300	0.4462	0.3461	0.3551
χ^2 clustering	0.8567	0.4438	0.2276	0.4310	0.3687	0.3630
random clustering	0.8541	0.4442	0.2192	0.4374	0.3663	0.3669
label PCA	0.8615	0.4388	0.1866	0.4345	0.3739	0.3874
feature/label CCA	0.8636	0.3103	0.0481	0.3173	0.3047	0.3116

7.3.1 CRF tree/chain approximations

Several CRF tree approximation methods are used in the experiments. First, we construct both tree and linear chain CRFs based on χ^2 score as in (3.4). In Table 7.4, the optimal tree is denoted as "χ^2 tree", and the greedily constructed linear chain CRF is referred to as "χ^2 chain".

We compare the approximated optimal chain with the chain that maximize the total edge weights (according to χ^2 score) out of 1001 randomly sampled chains, which is referred to as "sampled χ^2 chain".

Since the correlation based heuristic aims to approximate the maximum likelihood structure, we also compare previous methods with the chain that maximize the likelihood out of 1001 randomly sampled chains, which is referred to as "sampled ML chain".

Finally, we also compare with the ensemble method that learns 1001 randomly sampled chains and use the individual voting result for each label as the final prediction, which is referred to as "chain ensembles".

From Table 7.4, we can see that χ^2 tree performs slightly better than χ^2 chain, probably because a tree is a more general structure than a chain and is able to model label correlations more correctly. The performance of sampled χ^2 chain is not as good as χ^2 chain constructed by greedy algorithm. As we actually check on the resulting total weights of both chains, we found that χ^2 chain leads to larger value which implies that greedy heuristic does produce a good chain. As for sampled ML chain, it has the best result over all the correlation-based methods including χ^2 chain, χ^2 tree, and sampled χ^2 chain. Surprisingly, the random chains produce almost as good results as sampled ML chains. We discuss this in the next section. Finally, chain ensembles perform the best on most of the metrics.

7.3.2 Is correlation-based structure approximation a good idea?

As we mentioned earlier, the goal of correlation-based structure approximation is to obtain desired structure that yields maximum likelihood. However, Table 7.4 shows that a random chain outperforms any correlation-based chains. In order to examine whether a chain with larger total weights implies larger likelihood, we measure the *sortedness* of the total weights of a set of randomly sampled chains according to the order of their corresponding likelihoods. A common way to examine sortedness is to calculate the number of inversions, the number of pairs that are out of order. Specifically, given a bunch of chains, assume their likelihoods specify the correct ordering, we first sort these chains by their likelihoods in descending order and then count how many pairs of chains do not follow the same order (that is, the one in the front has smaller total weights than the other). There should

be zero inversions for a perfectly sorted list, and the better the correlation-based heuristic is the lower inversion count is.

Here we consider all 5 correlation measures: square of Pearson's correlation as in (3.2), mutual information as in (3.3), χ^2 score as in (3.4), correlation based on feature analysis (referred to as feature correlation), and partial correlation. In addition, we also examine an extra correlation measure that calculates the amount of log-likelihood gain of a label pair trained with CRF (CML model) to two independently trained logistic regression. Specifically, for any label pair y_i, y_j

$$\text{log-likelihood gain} = \ell_{\text{CRF}}(y_i, y_j) - (\ell_{\text{logisReg}}(y_i) + \ell_{\text{logisReg}}(y_j))$$

where ℓ denotes the log-likelihood. The reason to consider this measure is that both CRF and logistic regression are log-linear models while CRF takes label correlations into account but independent logistic regression on each label ignores them completely. Therefore this measure shows how much gain could be obtained by exploiting label correlations, and we assume that the maximum likelihood structure would be the one achieving the largest total gain.

As a matter of representation, for a list of n elements, the maximum number of inversions is $n(n-1)/2$ (i.e., total number of pairs) which could be used to normalize the inversion count. Table 7.5 shows the normalized inversion count of 1001 random sampled CRF chains with all 6 criteria mentioned above.

Table 7.5: Normalized inversion count that reflects how correlation-based approximation is related to maximum likelihood

correlation measure	normalized # inversions
Pearson's correlation	0.4873
mutual information	0.4775
χ^2 score	0.4873
feature correlation	0.4912
partial correlation	0.4839
log-likelihood gain	0.4792

All these criteria are no better than random samples with expected normalized inversion count of 0.5, which shows the correlation-based approximation is not

necessarily related to the optimal parameters learned from the model. Therefore, a randomly sampled structure could simply be used to approximate the maximum likelihood structure. This explains why random chain performs no worse than any of the correlation-based methods in Table 7.4.

7.3.3 Label space reduction

In this part, we evaluate the performance of two label space reduction methods: label PCA and feature/label CCA. We use linear regression with an extra bias term as our regression model on the transformed labels. Preliminary experiments show that label PCA is best to normalize the feature vector with rescaling method; while feature/label CCA is best with L_2 normalization on the feature vector.

The last part of Table 7.4 are the results of these two methods that reduce the original 174-dimensional label space into a 50-dimensional space. Table 7.4 shows that label PCA performs quite well in all the metrics we use. On the other hand, feature/label CCA performs poorly. Although feature/label CCA makes use of conditional label correlation, its main objective is to project into a space that maximizes the correlation between feature and label but has nothing to do with the recovery quality. On the other hand, label PCA aims at minimizing the reconstruction error on the label space, therefore it produces such accurate result which is comparable to that of other methods.

Since label PCA works quite well, we also examine how different reduced dimensionalities influence the performance of multi-label learning. It is quite surprising that even if we use only the first principal component, it produces acceptable results on most metrics. Except that $\mathrm{M}_\ell F_1$ increases as the dimensionality increases, label PCA gets even better result on most of other metrics when in a low dimensional label space. We can think of this as label PCA providing a smoothing observation that effectively reduces the noise of data as well as the potential of over-fitting.

Table 7.6: Performance of label PCA with different reduced dimensionalities

dim	Acc	μF_1	$M_\ell F_1$	$M_i F_1$	μMC	M_iMC
1	0.8716	0.4143	0.1167	0.4102	0.3838	0.3850
5	0.8693	0.4440	0.1590	0.4369	0.3939	0.4033
10	0.8689	0.4377	0.1619	0.4298	0.3889	0.3980
20	0.8684	0.4444	0.1800	0.4386	0.3920	0.4021
50	0.8615	0.4388	0.1866	0.4345	0.3739	0.3874
100	0.8599	0.4395	0.1947	0.4340	0.3716	0.3843
150	0.8591	0.4381	0.1910	0.4329	0.3692	0.3835

7.3.4 Label clustering

For the label clustering method, we use the modified single linkage clustering algorithm that keeps merging when the resulting cluster does not exceed a certain size (10 in our experiment) to split the original label set into disjoint groups of nearly the same size based on the correlation matrix.

We use multi-class method as the base multi-label classifier for each cluster. All 5 correlation measures we described in correlation analysis are used to evaluate the performance of label clustering framework. Table 7.7 shows that among the 5 correlation-based clustering methods, both Pearson's correlation and χ^2 score produce better multi-label classification results. However, random clustering achieve comparable result and in some metrics it even performs the best. This reflects the fact we discussed earlier that correlation-based approximations might not be the optimal configuration in terms of maximum likelihood.

Table 7.7: Clustering performance with different correlation measures

	Acc	μF_1	$M_\ell F_1$	$M_i F_1$	μMC	M_iMC
Pearson's correlation	0.8563	0.4430	**0.2283**	0.4306	0.3685	**0.3630**
mutual information	0.8501	0.4225	0.2041	0.4128	0.3434	0.3398
χ^2 score	**0.8567**	**0.4438**	0.2276	**0.4310**	**0.3687**	**0.3630**
feature correlation	0.8544	0.4316	0.2044	0.4202	0.3567	0.3559
partial correlation	0.8548	0.4123	0.1812	0.4038	0.3428	0.3465
random clustering	0.8541	0.4442	0.2192	0.4374	0.3663	0.3669

7.4 Discussion

In the small label set experiment, binary relevance with logistic regression gives the best result over all binary relevance methods with other base classifiers. As a generalized logistic regression (log-linear model), CRF exploits label correlations and is shown to outperform all the other methods. This confirms that label correlations provides useful information to improve multi-label learning.

However the exact CRF training and inference are not able to scale up for the entire label set because of its intractable nature. Only CRF approximation methods are applicable for large label set learning. Among all the multi-label learning algorithms we examined, supported CRF performs well especially in terms of F_1 scores.

Comparing binary relevance methods to multi-label algorithms that exploit label correlations, binary relevance methods in general do better on most of the metrics especially on the accuracy. It is reasonable to see that binary relevance is good at accuracy because accuracy is usually directly related to the base binary classifiers. Besides, making use of label correlations means to impose more constraints on the original binary relevance framework which might drive the optimization goal to predict something reasonable (without unlikely label combinations) instead of making it as accurate as possible on each label individually.

The other reason why multi-label algorithms that take label correlation into account do not perform as well as binary relevance methods is that the number of labeled songs in CAL500 is too small to be statistically representative for the problem space. When the training data is not large enough, strong regularization might be required to avoid over-fitting. This essentially takes less correlation information into account and falls back to a prior according to how we regularize the objective function (in our case, Gaussian prior for L_2 regularization). Moreover, from the result we can see that conditional label correlation is not as useful as unconditional label correlation for CAL500. This is again because there are even more parameters to learn, which increases the potential to over-fit such small training data. In particular, considering the entire CAL500 label set, there are 32,452 parameters (including one for regularization) for CML model, and 1,522,501

parameters for CMLF model, which is far more than 502, the total number of examples in CAL500. This explains why CAL500 favors simple models.

As for the clustering method, we always use multi-class algorithm as the base learners on each cluster for efficiency. Given that CML performs quite well on 8-label setup, we expect switching the base classifiers to CML might lead to better results. Nevertheless, comparing to multi-class method, clustering methods with multi-class base learners improves the classification result. The clustering framework can effectively reduce the total number of parameters, making parameter estimation more robust.

As for label space reduction, label PCA works surprising well even when only a couple of base learners are used. This suggests that it might be a useful framework to handle even larger label size.

Another lesson learned from our experiments is that feature engineering plays an important role in multi-label classification. Although we only consider four feature normalization methods, choosing the right one makes a significant difference. The results suggest that only L_2 normalization and rescaling should be considered, and rescaling usually produces acceptable results in general.

It is worth noting that randomized algorithms work well in our experiment. A randomly generated structure can perform better than the one based on correlation analysis. Also, chain ensembles outperform any of the single tractable structure methods that make CRFs applicable for larger label set. Together with the fact that random clustering with multi-class base learners produces much better result than multi-class methods on entire label set, it is not surprising that RAkEL [15] is considered one of the best multi-label framework in practice, which is essentially a more general form of our clustering method.

Chapter 8

Conclusion and future work

In this thesis, we have discussed how to model label correlations of a multi-label data set. We categorized label correlation into unconditional correlation and conditional correlation and described how to measure correlation in a quantitative manner. Unconditional correlation is calculated from the label combinations of the training data, while conditional correlation considers how labels are correlated with the associated features taken into account.

Various multi-label learning algorithms were then discussed. We have shown how to exploit both kinds of label correlations using conditional random fields (CRFs) for multi-label classification, and the results show that CRFs indeed work successfully. Because of the intractable nature of CRFs, we described several approximation heuristics to make them applicable for larger label sets, including the supported method, Gibbs sampling, and approximating with tractable structures such as tree or linear chain. Besides CRFs, we also showed several other multi-label learning algorithms including kNN, multi-class method, and clustering based on correlation analysis. To further reduce the computational complexity, we described two label space reduction methods: label PCA and feature/label CCA. The relationship between these multi-label learning algorithms and label correlations was also illustrated. In the end, all the learning algorithms were evaluated on the CAL500 music data set.

There are several directions to extend our work. First, we should evaluate the described learning algorithms on more multi-label data sets to see how they

work in general. We expect learning with label correlations would give better result if the number of training examples are large enough and statistically representative. If labeling a multi-label instance is expensive, manually created high-quality multi-label data sets inevitably have a relatively small number of instances, as in the case of CAL500. Developing learning algorithms to handle this specific situation would be useful in practice.

As for the general multi-label classification problem, there is still space for further study. First of all, we only model label correlations in a linear way, while in fact there are cases where labels are correlated non-linearly. Therefore being able to model non-linear correlation might be a useful extension. Besides, we only consider pairwise label correlations in the multi-label CRF models.

Second, we could also try to use models with latent variables, for example, a model similar to a two-layer neural network that introduces latent variables to model the actual related factors that generate the labels we want. Some generative models such as [20] employ this concept, which would be interesting to compare with algorithms we use.

Third, in our experiment, we only use a few metrics for evaluation while there are other widely used metrics such as the area under the ROC curve (AUC), mean average precision, etc. It would be better to compare with these alternative metrics. Moreover, since most of the evaluation metrics are calculated with the entries from confusion matrix, there are algorithms to directly optimize on a specific metric in this type, such as [21] which is based on the optimization technique in structured SVM [22].

Finally, an alternative to CRFs that also has the similar concept of incorporating various feature functions is structured SVMs [22], which extend traditional SVMs for structured output. It is possible to rewrite the feature functions we use in CRF to be loss functions that capture the distance between any two label combinations as the margin. Since CRF performs pretty well for multi-label learning except it is not able to scale up for larger label set, it would be interesting to see how structured SVM scales with the size of the label set using the cutting plane optimization [22], and also compare the results using structured SVM with that of

CRF on multi-label classification.

Bibliography

[1] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[2] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 439–446. ACM, 2007.

[3] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-label Data*, pages 5–12, Haifa, Israel, June 2010.

[4] G. Tsoumakas, A. Dimou, E. Spyromitros, V. Mezaris, I. Kompatsiaris, and I. Vlahavas. Correlation-based pruning of stacked binary relevance models for multi-label learning. *Learning from Multi-label Data at ECML PKDD 2009*, page 101, 2009.

[5] R.O. Duda, P.E. Hart, and D.G. Stork. Pattern classification. *Published by Wiely-Interscience*, 2000.

[6] M. Kendall and A. Stuart. The advanced theory of statistics. Vol. 2: Inference and relationship. 1979.

[7] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, page 200. ACM, 2005.

[8] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, page 8. Association for Computational Linguistics, 2002.

[9] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[10] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 2002.

[11] M.L. Zhang and Z.H. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[12] F. Tai and H.T. Lin. Multi-label classification with principle label space transformation. In *Workshop Proceedings of Learning from Multi-label Data*, pages 45–52, Haifa, Israel, June 2010.

[13] D. Hsu, S.M. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. *Advances in Neural Information Processing Systems*, 22:772–780, 2009.

[14] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[15] G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. *Lecture Notes in Computer Science*, 4701:406, 2007.

[16] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[17] P. Baldi, S. Brunak, Y. Chauvin, C.A.F. Andersen, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412, 2000.

[18] BW Matthews. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[19] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[20] M. Hoffman, D. Blei, and P. Cook. Easy as CBA: A simple probabilistic model for tagging music. In *Proceedings of the 10th International Conference on Music Information Retrieval*. Citeseer, 2009.

[21] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

[22] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.