Multi-Label Output Codes using Canonical Correlation Analysis

Yi Zhang **Jeff Schneider**
School of Computer Science, Carnegie Mellon University

Abstract

Traditional error-correcting output codes (ECOCs) decompose a multi-class classification problem into many binary problems. Although it seems natural to use ECOCs for multi-label problems as well, doing so naively creates issues related to: the validity of the encoding, the efficiency of the decoding, the predictability of the generated codeword, and the exploitation of the label dependency.

Using canonical correlation analysis, we propose an error-correcting code for multi-label classification. Label dependency is characterized as the most predictable directions in the label space, which are extracted as canonical output variates and encoded into the codeword. Predictions for the codeword define a graphical model of labels with both Bernoulli potentials (from classifiers on the labels) and Gaussian potentials (from regression on the canonical output variates). Decoding is performed by mean-field approximation.

We establish connections between the proposed code and research areas such as compressed sensing and ensemble learning. Some of these connections contribute to better understanding of the new code, and others lead to practical improvements in code design.

In our empirical study, the proposed code leads to substantial improvements compared to various competitors in music emotion classification and outdoor scene recognition.

1 Introduction

Error-correcting output codes (ECOCs) are traditionally designed to decompose a multiclass classification problem into many binary problems [9]. As a result, the multiclass problem can be solved using only binary classifiers, and the binary problems also provide a redundant representation to correct prediction errors. Two key components of an output coding scheme are *encoding* and *decoding*. The encoding maps each class to a *codeword*, which contains the outcomes of all decomposed binary problems on that class. Models are learned from training examples to predict the codewords of new examples, and the class of a new example is obtained by decoding the predicted codeword.

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

Unlike classes, labels in multi-label classification are not mutually exclusive. In a q-label problem, the cardinality of the output space $\mathcal{Y} = \{0,1\}^q$ is 2^q instead of q in a q-class problem. This change of output space presents new challenges to output coding:

- **Validity of the encoding**. An ECOC decomposes the target problem into a number of binary decision problems, each differentiating two subsets of classes. In multi-label problems, however, two subsets of labels can be simultaneously satisfied, which makes the corresponding binary decision ill-defined on certain examples. Ideally, the encoding should be well-defined for all possible label vectors $\mathbf{y} \in \{0,1\}^q$.
- **Efficiency of the decoding**. An ECOC usually decodes a predicted codeword by searching over all classes to optimize certain criteria, e.g., the Hamming loss to the predicted codeword. For an q-label problem, searching over all 2^q label vectors in $\{0,1\}^q$ is inefficient. An observation is that if encoding is nonlinear (e.g., defined upon a number of binary decisions), decoding has to solve a nonlinear system.
- **Predictability of the codeword**. Given an ECOC, classification is performed by predicting the codeword and then decoding. In this sense, predictability of the codeword is an important concern for code design. In multi-label classification, encoding via decomposing into binary problems is not always well-defined, and a new encoder that produces both valid and predictable codewords is needed.

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

- **Dependency among labels**. A key difference between multi-label and multi-class problems is the existence of label dependency. In multiclass problems, mutual exclusion of classes eliminates most dependency structures. In multi-label problems, however, rich dependency may exist among labels, which provides critical information for obtaining predictable codewords and strong error-correcting output codes.

1.2 Overview: Coding with Canonical Correlation Analysis

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

We establish connections between our work and other research areas such as compressed sensing [10, 20] and ensemble learning [2, 12, 4]. Our coding scheme can be viewed as an advanced sensing protocol, where random projection directions in compressed sensing are replaced by the most predictable directions in the label space, and noninformative sparsity priors are replaced by informative Bernoulli priors from trained classifiers. Also, we view ensemble learning in a prediction system as simulating the behavior of repetition codes in channel coding [7]. As a result, the successful use of repetition codes in concatenated coding [6] suggests the concatenation of ensemble learning with our coding scheme to produce more powerful output codes.

1.3 Organization

The rest of this paper is structured as follows. In Section 2 we review related work. In Section 3 we propose the new output code. In Section 4 we establish connections between the new code and other areas. In Section 5 we present our empirical study. In Section 6 we conclude the paper and discuss the future work.

2 Related Work

Error-correcting output codes have been studied for more than a decade [9]. The *encoding* of an ECOC decomposes the multi-class problem into a set of binary problems, e.g., one-versus-all [9], one-versus-one [17], random partitions [1], and partitions from problem-dependent search [8, 26]. The *decoding* of an ECOC recovers the class of an example by searching over all classes to optimize a distance function [9], a margin-based loss [1], a probability function [17, 25] or certain other criteria [11] w.r.t. the predicted codeword.

Multi-label compressed sensing [20] is perhaps the first output code defined for multi-label prediction. Our work is fundamentally different from this prior work. First, the goal of [20] is to reduce the number of predictions by applying *source coding* (i.e., compression) to labels, while our work aims to introduce additional redundancy and improve prediction by applying *channel coding* (error-correcting codes). Second, the encoding in [20] uses random projections as in compressed sensing [10, 5], while our encoding includes the most predictable directions in the label space for error correction. Third, the decoding in [20] recovers real-valued signals as in compressed sensing, while our decoding addresses the difficulty of inferring label assignments. We will study this prior work in our empirical study.

A building block of our coding scheme is canonical correlation analysis [19]. In a multi-output problem, CCA finds the projection directions for both input and output variables so that correlations between inputs and outputs are maximized along the projection directions. Canonical (projected) output variates are also known as *most predictable* variates [18, 19]. A recent overview of CCA is given in [16]. Sparse CCA [32, 15], kernel-based CCA [14], and nonparametric Bayesian CCA [27] have been studied. A least-square formulation of CCA is proposed in [29]. We use standard CCA in this paper and will consider new variants in future work.

Some methods for mining multi-label data [31] transform the problem into a set of subproblems and essentially adapt multi-class ECOCs to multi-label classification. Calibrated pairwise label ranking [13] is based on one-vs-one decomposition plus an adaptive calibration technique. For the validity of the encoding, a pairwise comparison may not be well-defined on certain examples (where the pair of labels are both true or both false), and the solution is to ignore all such training examples in learning, and apply the learned pairwise model to every testing example regardless of the issue. This may lead to ineffective use of training examples and unavoidable errors on certain testing examples. For decoding, voting from all pairwise models is used. We will study this method in our experiments.

Ensemble learning methods such as bagging [2], boosting [12] and random forests [4] are also related to output coding in the sense that they simulate the encoding and decoding process of repetition codes [7, 6].

3 Multi-Label Output Codes using Canonical Correlation Analysis

In this section, we propose our error-correcting output codes for multi-label classification. In Section 3.1 we review canonical correlation analysis (CCA) in multi-label settings. In Section 3.2 we introduce the CCA-based encoding and training algorithm. In Section 3.3 we present the decoding and prediction algorithm.

3.1 Canonical Correlation Analysis

Canonical correlation analysis [18, 19] is a classical tool for modeling linear associations between two sets of variables. Consider a set of p variables $\mathbf{x} \in \mathcal{X} \subseteq R^p$ and another set of q variables $\mathbf{y} \in \mathcal{Y} \subseteq R^q$. For a multi-label classification problem, $\mathbf{x}$ will denote the feature vector and $\mathbf{y}$ will denote the label vector. In this case, we will have $\mathbf{y} \in \mathcal{Y} = \{0,1\}^q$. In addition, we have a set of n training examples: $\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$, where $\mathbf{X}$ and $\mathbf{Y}$ are matrices of size $n \times p$ and $n \times q$, respectively. Canonical correlation analysis finds a pair of projection directions $\mathbf{u} \in R^p$ and $\mathbf{v} \in R^q$ such that the *correlation* between the pair of projected variables $\mathbf{u}^T\mathbf{x}$ and $\mathbf{v}^T\mathbf{y}$ (also called *canonical variates*) is maximized. Given the training examples $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^n$, this maximization problem can be expressed as the following[1]:

$$\underset{\mathbf{u}\in R^p, \mathbf{v}\in R^q}{\operatorname{argmax}} \quad \frac{\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v}}{\sqrt{(\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u})(\mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v})}} \tag{1}$$

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

$$\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u} = 1 \tag{2}$$
$$\mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v} = 1 \tag{3}$$

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

$$\begin{aligned} \underset{\mathbf{u}\in R^p, \mathbf{v}\in R^q}{\operatorname{argmax}} \quad & \mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} \\ \text{s.t.} \quad & \mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u} = 1 \\ & \mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v} = 1 \end{aligned} \tag{4}$$

By formulating the Lagrangian of the above convex optimization problem, the KKT conditions lead to the following generalized eigenproblems [16] on $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{X}^T\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{u} = \lambda\mathbf{X}^T\mathbf{X}\mathbf{u} \tag{5}$$
$$\mathbf{Y}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{v} = \lambda\mathbf{Y}^T\mathbf{Y}\mathbf{v} \tag{6}$$

The formulations (5) and (6) provide more than one pair of projection vectors $(\mathbf{u}, \mathbf{v})$. By solving the first d principal eigenvectors, we can obtain d pairs of projection vectors: $\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^d$. These projection vectors successively maximize the correlation between the resulting canonical input and output variates, with generalized orthogonality constraints satisfied:

$$\mathbf{u}_k^T\mathbf{X}^T\mathbf{X}\mathbf{u}_j = 0, \qquad \forall k \neq j$$
$$\mathbf{v}_k^T\mathbf{Y}^T\mathbf{Y}\mathbf{v}_j = 0, \qquad \forall k \neq j$$

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

$$\{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=1}^d \leftarrow \mathrm{CCA}(\mathbf{X}, \mathbf{Y}) \tag{7}$$

Algorithm 1 The encoding and training algorithm

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

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

3.2 Encoding

Channel coding [7, 6] studies reliable communication through noisy channels. To reliably transmit a message $\mathbf{y}$, we *encode* it into a redundant codeword $\mathbf{z}$, which is sent through the channel. A randomly corrupted $\hat{\mathbf{z}}$ will be received, and the encoded redundancy will be used to correct the transmission error and recover the message. In multi-label prediction, the label vector $\mathbf{y} \in \{0,1\}^q$ of a new example $\mathbf{x}$ is the message we want to communicate. In this sense, *transmission* through a *noisy channel* corresponds to *prediction* using trained but *imperfect models*. Therefore, the goal of encoding in a multi-label output code is to find a redundant codeword $\mathbf{z}$ for the label vector $\mathbf{y}$, such that we can train models to predict the codeword given $\mathbf{x}$ and recover the label vector $\mathbf{y}$ from the prediction.

It is critical to realize a key difference between a noisy channel and a prediction system: transmission errors of a noisy channel are usually *independent* of the information being transmitted, while prediction errors of a trained system often *depend* on the signal being predicted. In other words, some signals are easier to predict than others. Canonical correlation analysis plays

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

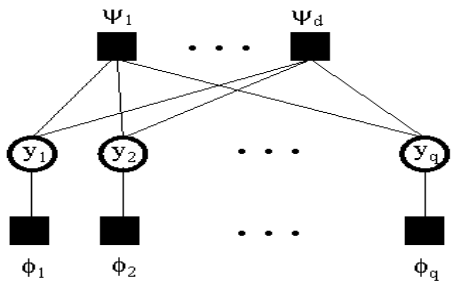

Figure 1: Factor graph representation of the undirected graphical model for decoding.

an important part at this point: canonical output variates $\{\mathbf{v}_k^T\mathbf{y}\}_{k=1}^d$ found by CCA as in (7) are known as the *most predictable* variates [18]. To see this, we can rewrite (4) into the following equivalent problem:

$$\begin{aligned} \underset{\mathbf{u}\in R^p,\mathbf{v}\in R^q}{\text{argmin}} \quad & ||\mathbf{X}\mathbf{u}-\mathbf{Y}\mathbf{v}||^2 \\ \text{s.t.} \quad & \mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u}=1 \\ & \mathbf{v}^T\mathbf{Y}^T\mathbf{Y}\mathbf{v}=1 \end{aligned} \tag{8}$$

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

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

$$\mathbf{z}=(y_1,\ldots,y_q,\mathbf{v}_1^T\mathbf{y},\ldots,\mathbf{v}_d^T\mathbf{y})^T \tag{9}$$

Note that this defines a systematic code [7], where the codeword contains the original message (the label vector $\mathbf{y}=(y_1,\ldots,y_q)^T$ in this case) as a subcomponent. The intuition is that the message always contains high information about itself. Given a set of training examples $(\mathbf{X},\mathbf{Y})=\{(\mathbf{x}^{(i)},\mathbf{y}^{(i)})\}_{i=1}^n$, we will learn q classifiers $\{\hat{p}_1,\ldots,\hat{p}_q\}$ to predict the q labels $\{y_1,\ldots,y_q\}$, and d regression models $\{\hat{m}_1,\ldots,\hat{m}_d\}$ to predict the d canonical variates $\{\mathbf{v}_1^T\mathbf{y},\ldots,\mathbf{v}_d^T\mathbf{y}\}$. The encoding and training procedure is summarized in **Algorithm 1**.

Note that the challenges discussed in Section 1.1 are being addressed: 1) the encoding in (9) is well-defined for any label vector $\mathbf{y}\in\{0,1\}^q$; 2) the encoding is *linear*, which enables us to develop efficient decoding (in the next section); 3) $\{\mathbf{v}_k^T\mathbf{y}\}_{k=1}^d$ included in the codeword $\mathbf{z}$ are the most (linearly) predictable variates; 4) the label dependency in $\mathbf{y}$ is exploited (via projection vectors $\{\mathbf{v}_k\}_{k=1}^d$) to produce predictable codewords[2].

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

3.3 Decoding

For a testing example $\mathbf{x}$, classification and regression models learned in Algorithm 1 provide a predictive distribution on the codeword $\mathbf{z}$. Each classifier $\hat{p}_j$ predicts a Bernoulli distribution $\phi_j(y_j)$ for a label y_j:

$$\phi_j(y_j)=\hat{p}_j(\mathbf{x})^{y_j}(1-\hat{p}_j(\mathbf{x}))^{(1-y_j)}, \quad j=1,2,\ldots,q \tag{10}$$

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

$$\psi_k(\mathbf{y}) \;\propto\; \exp-\frac{(\mathbf{v}_k^T\mathbf{y}-\hat{m}_k(\mathbf{x}))^2}{2\hat{\sigma}_k^2}, \quad k=1,2,\ldots,d \tag{11}$$

where the variance $\hat{\sigma}_k^2$ can be estimated by cross validation on the training examples in Algorithm 1.

For decoding, predictive distributions (10) and (11) can be viewed as the factor graph in Figure 1, which defines the following joint probability for the label vector $\mathbf{y}$ given the testing example $\mathbf{x}$:

$$\log P(\mathbf{y})=-\log\mathcal{Z}+\sum_{k=1}^d\log\psi_k(\mathbf{y})+\lambda\sum_{j=1}^q\log\phi_j(y_j) \tag{12}$$

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

Unfortunately, exact inference over $\mathbf{y}\in\{0,1\}^q$ in (12) has a time complexity *exponential* in q, due to the fact that each Gaussian potential ψ_k in (11) usually involves all the q labels. We consider a mean-field approximation to $P(\mathbf{y})$ in (12), as the following:

$$Q(\mathbf{y})=\prod_{j=1}^q Q_j(y_j) \tag{13}$$

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

To find the best approximation $Q(\mathbf{y})$ in the fully factorized class, we need to minimize the KL divergence $KL(Q||P)$, or equivalently, to maximize the energy functional [21]. Based on the definition of $P(\mathbf{y})$ in (12), the fixed-point equation for updating each Bernoulli factor $Q_j(y_j)$ in Q can be written as [23]:

$$Q_j(y_j)\leftarrow\frac{1}{\mathcal{Z}_j}\exp\{\,\lambda\log\phi_j(y_j)+\sum_{k=1}^d\mathbf{E}_{\mathbf{y}\sim Q}[\log\psi_k(\mathbf{y})|y_j]\,\} \tag{14}$$

where $\mathcal{Z}_j$ is the normalization constant that makes Q_j a valid Bernoulli distribution. Note that each term $\mathbf{E}_{\mathbf{y}\sim Q}[\log\psi_k(\mathbf{y})|y_j]$ can be computed with a time complexity *polynomial* in q (or more specifically, in $O(q^2)$), due to the fact that Q is fully factorized over labels and

Algorithm 2 The decoding and prediction algorithm

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

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

the Gaussian log density $\log \psi_k$ involves only second order interactions between labels. Thus, the mean-field approximation $Q(\mathbf{y})$ can be obtained by iterating the fixed-point update over j until convergence.

For each testing example $\mathbf{x}$, mean-field approximation provides a factorized distribution $Q(\mathbf{y}) = \prod_{j=1}^q Q_j(y_j)$ on labels, where factors $\{Q_j(y_j)\}_{j=1}^q$ can be directly used for various prediction tasks, e.g., classification, ranking, probabilistic inference. The decoding and prediction procedure is summarized in **Algorithm 2**.

4 Connections to Other Research

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

4.1 Multi-label Compressed Sensing

Encoding. Multi-label compressed sensing [20] is perhaps the earliest work that formally defines an output code for multi-label prediction. Given a label vector $\mathbf{y} \in \{0,1\}^q$, the *encoding* is defined as the following:

$$\mathbf{z} = (\mathbf{v}_1^T\mathbf{y}, \ldots, \mathbf{v}_d^T\mathbf{y})^T \tag{15}$$

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

Decoding. For a test example $\mathbf{x}$, its codeword is predicted as $\hat{\mathbf{z}} = (\hat{m}_1(\mathbf{x}), \ldots, \hat{m}_d(\mathbf{x}))^T$ by trained regression models $\{\hat{m}_k\}_{k=1}^d$. The *decoding* assumes the *sparsity* of the true label vector $\mathbf{y}$ and reconstructs $\mathbf{y}$ using *sparse approximation* techniques. Popular choices include ℓ-1 penalized convex optimization [30] and iterative methods such as CoSaMP [24]. To analyze the link between compressed sensing and our coding scheme, we focus on ℓ-1 penalized convex relaxation [30]:

$$\underset{\mathbf{y}\in R^q}{\operatorname{argmin}} \ \frac{1}{2}\sum_{k=1}^d (\mathbf{v}_k^T\mathbf{y} - \hat{m}_k(\mathbf{x}))^2 + \lambda \sum_{j=1}^q |y_j| \tag{16}$$

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

Connection. To see the connection between our proposed code and the compressed sensing above, we consider optimization of the joint probability in (12) over labels $\mathbf{y} \in \{0,1\}^q$. Plugging Bernoulli potentials (10) and Gaussian potentials (11) into the joint probability and ignoring the log partition term, maximizing (12) is equivalent to the following minimization problem:

$$\underset{\mathbf{y}\in\{0,1\}^q}{\operatorname{argmin}} \ \frac{1}{2}\sum_{k=1}^d \frac{(\mathbf{v}_k^T\mathbf{y} - \hat{m}_k(\mathbf{x}))^2}{\hat{\sigma}_k^2} + \lambda \sum_{j=1}^q y_j \log(\frac{1-\hat{p}_j(\mathbf{x})}{\hat{p}_j(\mathbf{x})}) \tag{17}$$

where the objective contains d squared error terms and q linear terms, from the d Gaussian potentials and q Bernoulli potentials in (12), respectively. We can see that optimizing the joint label probability as in (17) is similar to reconstructing signals in compressed sensing as in (16), but with the following key differences:

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

Recall that our decoding uses mean-field approximation to solve this intractable optimization problem.

4.2 Ensemble Learning as Repetition Codes

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

$$\mathbf{z} = (y_1^1 \ldots y_1^r, y_2^1 \ldots y_2^r, \ldots, y_q^1 \ldots y_q^r)^T \in \{0,1\}^{rq} \tag{18}$$

where r is the number of repetitions. If the channel randomly flips each bit in $\mathbf{z}$ with a probability $p < 0.5$, decoding for y_j can take a majority voting from received bits $\{y_j^1 \ldots y_j^r\}$. If the channel is making *inde-*

pendent flipping errors (with $p < 0.5$), we can achieve error-free communication using a sufficiently large r.

Ensemble learning imitates repetition codes in a prediction system by learning multiple models for the same label. However, if we simply repeat a label, predictions from repeatedly trained models will be identical. If the "channel" makes *identical* (instead of independent) errors, majority voting will not correct any of them. Therefore, ensemble learning aims to create a *diverse* set of models whose prediction errors are as independent as possible [22] via perturbation on training samples [2], sample weights [12], features [4], etc[3].

Concatenated coding. Repetition codes, although not efficient when used alone due to their high redundancy, play an active role in *concatenated coding*. A concatenated code involves the cascade of an *outer* code and an *inner* code. The message is first encoded by the outer encoder and the resulting codeword is further encoded by the inner encoder. An efficient (although suboptimal) decoding can be performed by sequentially applying the inner and outer decoder. Using two simpler codes, concatenated coding provides a longer, more powerful code with efficient encoding and decoding and has produced high performance code in practice, e.g., the code in NASA standards in 1970s [6].

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

5 Experiments

Data. We conduct our experiments on two multi-label data sets: *Emotions* and *Scene*[4]. Emotions is a music classification data set containing 391 songs in the training set and 202 songs in the testing set. Songs are labeled with six emotions: amazed, happy, relaxed, quiet, sad and angry. Each song is represented by 72 rhythmic and timbre features. Scene is an image collection for outdoor scene recognition, with 1211 training images and 1196 testing images tagged with six scenes: beach, sunset, fall foliage, field, mountain and urban. Features are 294 dimensional color statistics.

Methods. We study the following methods:

- *One-vs-All(LGR/SVM)*: one-vs-all decomposition with ℓ-2 penalized logistic regression or SVMs.
- *Pairwise(LGR/SVM)*: calibrated pairwise label ranking [13] with ℓ-2 logistic regression or SVMs.
- *CCA-Reduce(LGR/SVM)*: supervised dimension reduction by CCA and one-vs-all in reduced feature space. We report results with optimal dimensions.
- *CompressSensing*: multi-label compressed sensing (CS) [20], with 300 projections to produce highly redundant codewords, ridge regression to predict, and CoSaMP [24] to decode. CS recovers real values, and we use 0.5 as the threshold. We report results with the optimal CoSaMP parameter (sparsity level).
- *One-vs-All(Tree/Forest)*: one-vs-all decomposition with classification trees [3] or random forests [4] (50 trees per label). Comparing these two models shows the power of ensemble learning as a repetition code.
- *CCA-OC*: our proposed coding with ℓ-2 logistic regression and ridge regression as classifiers and regression models. We simply use the maximal d from CCA in Algorithm 1, and the parameter λ in Algorithm 2 is chosen from $\{\frac{1}{4}, 1, 4\}$.
- *CCA-OC-Forest*: our proposed coding with random forests, where random forests provide both accurate classification and regression. This is a concatenated coding scheme with our code as the outer code and a repetition code as the inner code.

The regularization parameter of ℓ-2 penalized logistic regression, SVMs and ridge regression is set by cross validation within each individual learning task.

Evaluation measures. We report on three measures:

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

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

Experimental results on three measures are shown in Table 1 - Table 3 for Emotions data and Table 4 - Table 6 for Scene data. The proposed methods are **in bold**. We briefly highlight the results as follows:

- CCA-OC significantly outperforms other methods and CCA-OC-Forests further enhances CCA-OC with concatenated coding. Improvements are most evident on exact matching rates (Table 1 and 4),

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

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

Table 1: Exact matching rates on Emotions data: means (and standard errors) over 30 random runs.

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

Table 2: *Micro*-averaged F-measures on Emotions data: means (and standard errors) over 30 random runs.

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

Table 3: *Macro*-averaged F-measures on Emotions data: means (and standard errors) over 30 random runs.

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

which require capturing the label dependency. We also note that two proposed methods with 200 training samples on Emotions and 400 on Scene achieve similar or better performance than baselines trained with 391 samples on Emotions and 1211 on Scene.

- One-vs-All decomposition is a strong baseline. This is consistent with other known results [28].
- Calibrated pairwise label ranking (Pairwise) shows slight improvements over one-vs-all decomposition.
- Dimension reduction via CCA (CCA-Reduce) is not effective: the canonical input directions may not be predictive for the original output variables (labels).
- Multi-label compressed sensing is not a capable error-correcting code even with highly redundant codewords, mainly due to its non-adaptive encoding with random projections and real-valued decoding.

Table 4: Exact matching rates on Scene data: means (and standard errors) over 30 random runs.

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

Table 5: *Micro*-averaged F-measures on Scene data: means (and standard errors) over 30 random runs.

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

Table 6: *Macro*-averaged F-measures on Scene data: means (and standard errors) over 30 random runs.

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

6 Conclusion and Future Work

Using CCA, we propose error-correcting codes for multi-label classification. Label dependency is characterized as the most predictable directions in the label space, extracted as canonical output variates and encoded into the codeword. Predictions for the codeword define a graphical model of labels with both Bernoulli potentials (from classifiers on the labels) and Gaussian potentials (from regression on the canonical variates). Decoding is performed by mean-field approximation. In the empirical study on music and image classification, our proposed codes offer substantial improvements over every competitor in every test.

In the future work we will consider recent variants of sparse and nonlinear CCA for encoding, and state-of-the-art variational inference techniques for decoding.

References

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

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1 edition, 1984.

[4] L. Breiman and E. Schapire. Random forests. *Machine Learning*, pages 5–32, 2001.

[5] E. J. Candes. Compressive Sampling. In *Proceedings of International Congress of Mathematicians*, 2006.

[6] D. J. Costello and G. D. Forney. Channel coding: The road to channel capacity. *Proceedings of the IEEE*, 95(6):1150–1177, 2007.

[7] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[8] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Mach. Learn.*, 47(2-3):201–233, 2002.

[9] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[10] D. L. Donoho. Compressed Sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006.

[11] S. Escalera, O. Pujol, and P. Radeva. On the decoding process in ternary error-correcting output codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):120–134, 2010.

[12] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[13] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2):133–153, 2008.

[14] C. Fyfe and P. L. Lai. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10:365–374, 2001.

[15] D. R. Hardoon and J. Shawe-Taylor. Sparse canonical correlation analysis. http://arxiv.org/abs/0908.2724, 2009.

[16] D. R. Hardoon, S. R. Szedmak, and J. R. Shawe-taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.*, 16(12):2639–2664, 2004.

[17] T. Hastie and R. Tibshirani. Classification by pairwise coupling. In *NIPS '97*, 1997.

[18] H. Hotelling. The most predictable criterion. *Journal of Educational Psychology*, 26:139–142, 1935.

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

[20] D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *NIPS*, pages 772–780. 2009.

[21] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[22] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.

[23] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[24] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3), 2008.

[25] A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE Transactions on Neural Networks*, pages 45–54, 2004.

[26] O. Pujol, P. Radeva, and J. Vitri. Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1007–1012, 2006.

[27] P. Rai and H. Daume. Multi-label prediction via sparse infinite cca. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1518–1526. 2009.

[28] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, 2004.

[29] L. Sun, S. Ji, and J. Ye. Canonical correlation analysis for multilabel classification: A least-squares formulation, extensions, and analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33:194–200, 2011.

[30] J. A. Tropp. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006.

[31] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*. Springer, 2 edition, 2010.

[32] D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, July 2009.