Medical Coding Classification by Leveraging Inter-Code Relationships

Yan Yan
ECE Department
Northeastern University
Boston, MA USA
yan.y@neu.edu

Glenn Fung
CAD and Knowledge Sols.
Siemens Healthcare
Malvern, PA USA
glenn.fung@siemens.com

Jennifer G. Dy
ECE Department
Northeastern University
Boston, MA USA
jdy@ece.neu.edu

Romer Rosales
CAD and Knowledge Sols.
Siemens Healthcare
Malvern, PA USA
romer.rosales@siemens.com

ABSTRACT

Medical coding or classification is the process of transforming information contained in patient medical records into standard predefined medical codes. There are several worldwide accepted medical coding conventions associated with diagnoses and medical procedures; however, in the United States the Ninth Revision of ICD (ICD-9) provides the standard for coding clinical records. Accurate medical coding is important since it is used by hospitals for insurance billing purposes. Since after discharge a patient can be assigned or classified to several ICD-9 codes, the coding problem can be seen as a multi-label classification problem. In this paper, we introduce a multi-label large-margin classifier that automatically learns the underlying inter-code structure and allows the controlled incorporation of prior knowledge about medical code relationships. In addition to refining and learning the code relationships, our classifier can also utilize this shared information to improve its performance. Experiments on a publicly available dataset containing clinical free text and their associated medical codes showed that our proposed multi-label classifier outperforms related multi-label models in this problem.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Learning*

General Terms

Algorithms

Keywords

Medical data mining, multi-label classification, medical coding, large margin, classification, L1 regularization

1. INTRODUCTION

1.1 Background on Medical Coding

Hospitals and in general healthcare providers rely on medical coding to record medical services (procedures) and associated causes or conditions (diagnoses). These procedure and diagnosis codes are key elements in the financial transactions processed by insurance companies or other medical reimbursement-processing organizations. Thus, healthcare providers are largely required to use medical codes to classify the present conditions/diseases and procedures performed on basically all patients. In addition to this *de facto* usage, medical codes have many additional uses in automated decision support in medicine, medical quality or guideline adherence, and disease surveillance, among many areas.

The most commonly used diagnosis coding systems are based on the International Statistical Classification of Diseases and Related Health Problems (commonly abbreviated ICD). ICD-9 was created by the World Health Organization (WHO) in 1977 (later extended to ICD-9 CM, where CM stands for Clinical Modification) and is used primarily in the United States. ICD-10 is used in most of the rest of the world, albeit with certain regional modifications. The codes include, in general, classifications for signs, symptoms, abnormal findings, complaints, social circumstances, and causes of injury or disease. In the United States, ICD-9 codes play a key role in reporting medical statistics to government organizations such as the Joint Commission on Accreditation of Healthcare Organizations (JCAHO) and the Center for Medicate and Medicaid Services (CMS). Importantly, ICD-9 codes are used to track certain diseases, such as the flu (ICD-9 code 486), that may have public health implications.

In the considered scenario, when a patient receives a medical service, an ICD-9 code is assigned. The code depends on the reasons for the patient visit. This is normally done manually, by skilled personnel. The codes may be assigned immediately, but in most cases, especially for patients requiring hospitalization, codes are assigned retrospectively after an expert reviews medical documentation (doctor notes, lab reports, etc.) created during the patient visit. That is, an expert coder reads the documentation and, based on medical knowledge, guidelines, regulations, and experience, assigns one or more ICD-9 codes to the patient visit.

Given the large number of codes and their level of specificity for certain (not all) diagnoses, this is a very time consuming process. It is also error prone; it is estimated that only 60% to 80% of assigned codes truly reflect the patient diagnosis [1]. There are a large number of reasons why more efficiency and more accuracy are highly needed in ICD-9 code assignment. These include, the imperative to reduce healthcare costs; the clear need to keep accurate statistics of diseases, specially those important for public health; the need to increase cost transparency in the healthcare system; and the large potential that a correctly maintained and recorded coding system can have in patient treatment and in general decision support.

1.2 Specific Data Mining Problem

In this paper, we address the ICD-9 code assignment problem described above based on the natural language text employed to document the patient hospital visit. This includes text from documents such as doctor notes, lab reports, history and physical, nursing assessments, etc.

ICD-9 coding has several interesting properties from a data mining and statistical modeling point of view. The codes are based on a hierarchy where general categories and several sub-categories are defined. For example, the code 440 is assigned to Atherosclerosis, but more specific codes such as 440.1 and 440.2 are assigned to Atherosclerosis of Renal Artery and Arteries of the Extremities respectively. A key characteristic of this coding system is that assigned codes are often correlated, for example the occurrence of Chest Pain (786.5) is common with the occurrence of Congestive Heart Failure (428.0) for the same patient visit. This co-occurrence of diagnoses is to some degree known by experts and thus this prior domain knowledge should be utilized whenever possible in the design of automated coding algorithms. In addition, the problem is multi-label; one patient visit gets associated with multiple codes. Finally, the data is sparse, some codes are very common while others are very rare. Thus, the data support for the different classes (codes) varies enormously.

Given the specific challenges presented by this problem and application, we developed and analyzed a method for building a multi-label (multi-class) classifier that can estimate the class structure of the data together with the individual classifiers. Given the lack of data for some classes, the method also allows the incorporation of prior knowledge about the structure automatically, adjusting the degree upon which this prior knowledge should be used. We demonstrate experimentally that the approach can be used to solve large problems with a performance similar or superior to related state-of-the-art approaches.

A large number of daily tasks in healthcare are based on manual review of information, specially documents, written in natural language. While approaches on natural language processing have grown rapidly in the past years in many areas (primarily in Web mining and language translation), the area of automated medical coding has been explored in a limited way. Most deployed systems use rule-based approaches [7, 15], while others have used basic learning methods such as k-nearest neighbors or naive Bayes classifiers [9]. Sometimes the approach is semi-automated [11]. The most recent approach we are aware of uses a Gaussian Process (GP) based classifier to learn to assign ICD-9 codes to records [10]. However, related approaches have one or several limitations in that they either do not take advantage of prior domain knowledge efficiently, are not explicitly designed for multi-label data (thus, they usually cannot exploit the class structure), or do not explicitly uncover the underlying class (code) relationships. In this paper we propose a method to address these limitations in a new and simple formulation.

1.3 Technical Overview

This paper presents a solution to the code assignment problem above based on a multi-label, large-margin classification model. In multi-class classification, each point (patient visit) $\mathbf{x}$ may be labeled with a number of distinct classes (codes). A key property of multi-label problems is that class labels are often not mutually independent given a data point; the existence of this class/label structure is a fundamental factor in this paper.

It is possible to approach this multi-label problem using a combination of standard binary (or even multi-class) classifiers; however, this would not exploit the underlying class structure. If the labels are correlated, we could try to represent combinations of labels as new, compound labels. This is often inappropriate due to the combinatorial number of compound classes required and the lack of sufficient data to create meaningful estimates [2]).

We can focus on building a function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$ that can be used to produce properly ranked labels [13]. Various other strategies are possible. If a Hamming loss is used to build the function f basically to reduce the number of label corrections needed to classify the given points, a standard binary classification approach would be sufficient. If the fraction of unordered labels is used as a loss function, this is equivalent to the *rank*-SVM approach [5], one of the state-of-the-art methods. A related non-parametric approach to achieve this is to learn prototype vectors for each class and rank the labels of the given test point according to its inner product with the prototype vectors [3]. A related method, named the multi-label KNN algorithm (ML-KNN) [18], consists on finding the k nearest examples to the data point to be classified and formulating a prediction based on the number of elements of the given class falling in the neighborhood.

The presented approach can represent class relationships where, for example, data points with a particular medical code (class) may be very likely to have a related code but very unlikely to have an unrelated code. These type of relationships very often exist in multi-label problems. A standard classifier will ignore or (at best) not explicitly model this potentially rich structure, even though it is present in the data. In this paper, we focus on explicitly uncovering this structure for ICD classifications since this would be highly valuable for medical interpretability.

In this problem, there exists prior medical knowledge that can help with incorporating this structure into the classifier. This may resemble the notion of a *prior* $p(y)$ in probabilistic modeling approaches; however, for the purposes of this paper we may not use the term *prior knowledge* in an equivalent manner [1]. The availability of prior domain knowledge is clearly another motivation for explicitly representing the class structure. However, in practice it is difficult to determine the degree of accuracy of this prior knowledge. In this paper we do not make a strong assumption about how reliable this knowledge is, instead we automatically determine the level on which it should be taken into account in order to maximize our classification criterion.

2. FORMULATION

Before introducing formally the proposed method, we first define our notation. Suppose we have D data samples (medical document), and each of them is in dimension K, where each data sample component (feature) is a binary value that represents whether or not each one of the K considered words is present in the document. We represent them as D row vectors: $\mathbf{x}_i \in \mathcal{X}$; here $\mathcal{X}$ denotes the domain of instances. Each data point shares an associate L dimensional row label vector $\mathbf{y}_i$, which is formulated from

[1]A prior reflects the knowledge about class labels before observing the data in question. In this paper we do not make this restriction about *prior knowledge*.

$\mathcal{Y}_{\mathbf{x}_i} \subseteq \mathcal{Y}$; $\mathcal{Y} = \{1, 2, \cdots, L\}$. Each $\mathbf{y}_i(l)$ shall take either $+1$ if $l \in \mathcal{Y}_{\mathbf{x}_i}$ (the document belongs to the code i), or -1 if $l \notin \mathcal{Y}_{\mathbf{x}_i}$ (the document does not belong to the code i). In order to simplify our notation, we will denote all the data samples as a $D \times K$ matrix: $X = [\mathbf{x}_1'\mathbf{x}_2' \cdots \mathbf{x}_D']'$, and similarly, all the labels as a $D \times L$ matrix: $Y = [\mathbf{y}_1'\mathbf{y}_2' \cdots \mathbf{y}_D']'$, so individually, each row of X corresponds to one data sample, and its label vector is the corresponding row in Y. Vector $\mathbf{e}$ represents a column vector of ones of proper dimensionality, and $\operatorname{diag}(\mathbf{v})$ represents a diagonal matrix whose diagonal components are the elements of the vector $\mathbf{v}$.

2.1 Proposed Mathematical Programming Formulation

Consider the following optimization problem:

$$\min_{W,M} \mu \sum_{i=1}^{D} \|\mathbf{e} - \operatorname{diag}(\mathbf{y}_i)\hat{\mathbf{y}}_i'\|^2 + \|W\|_1 + \nu \|M - \tilde{\mathcal{M}}_{sim}\|^2_{\text{frob}}$$
$$\texttt{s.t.} \quad -1 \le M(i,j) \le 1, \quad i,j = \{1,\ 2, \ldots,\ D\} \tag{1}$$

where

$$\hat{\mathbf{y}}_i \triangleq (\mathbf{x}_i W M + \gamma)$$

W stands for a $K \times L$ matrix, and each column $\mathbf{w}_i$ of W is a hyperplane classifier predicting the i^{th} label. Therefore, since each data point has L labels, we have in total L columns. M can be defined as a between-labels relation matrix, of size $L \times L$. γ describes the appropriate hyperplane shift from the origin (or threshold) for prediction $\hat{\mathbf{y}}_i$ given by $\mathbf{x}_i WM$. The parameter μ controls the trade-off between classification accuracy and regularization. $\|\cdot\|_{\text{frob}}$ denotes the matrix Frobenius norm, and $\|\cdot\|_1$ denotes a matrix `L1` norm defined as:

$$\|W\|_1 = \sum_i \|\mathbf{w}_i\|_1 = \sum_{ij} |w_{ij}|; \tag{2}$$

this is, the summation of the absolute values of all the elements of the matrix. In supervised learning, it is often the case that even though the total number of input features is large, only a small fraction of these features suffices to build a model with (at least) comparable performance. Furthermore, it is a well-known fact that feature selection can help prevent overfitting in problems with many input features relative to the amount/variability of the data (see [6, 16]). This is generally the case with problems related to text processing where the number of available features corresponds to a large number of available words that are present in the set of documents.

As mentioned before, M is a matrix that captures the relations between the labels according to prior knowledge and available data. $\tilde{\mathcal{M}}_{sim}$ is an initial prior knowledge matrix that contains information about the labels. This prior knowledge can be provided by the user based on experts' domain knowledge or it can be simply empirically estimated from available data. The parameter ν controls how much the matrix M will stay close to the prior knowledge $\tilde{\mathcal{M}}_{sim}$ vs. the optimization of M to minimize empirical error based on the training data.

Note that from the definition of $\hat{\mathbf{y}}_i$ in formulation (1), prediction of label l depends not only on the corresponding classifier $\mathbf{w}_l$ (l column of W) but also on the remaining $L-1$ classifiers corresponding to the other $L-1$ labels. This dependence is controlled by the matrix M, so defining $\tilde{\mathcal{M}}_{sim}$ that encodes prior knowledge about the relation among the expected output of the classifiers, intuitively, is a good choice of an initial point for our algorithm.

For empirical estimation of the similarity matrix $\tilde{\mathcal{M}}_{sim}$, one very general choice that is also employed in our experiments is as follows:

$$\begin{aligned} \mathcal{M}_{sim} &= [\omega_{ij}]_{L\times L} & D_{\mathcal{M}} &= [d_{ij}]_{L\times L} \\ \tilde{\mathcal{M}}_{sim} &= \mathcal{M}_{sim} D_{\mathcal{M}}{}^{-1} & \omega_{ij} &= \textstyle\sum_{q=1}^{D} \mathbf{y}_q(i)\mathbf{y}_q(j) \\ d_{ii} &= \sum_j |\omega_{ij}| & d_{ij} &= 0 \ \ (i \neq j) \end{aligned} \tag{3}$$

In order to interpret equation (3) it is useful to note the following:

- For each data sample, if the i^{th} and the j^{th} labels agree with each other (both positive or both negative), ω_{ij} increases by a unit, otherwise it decreases by a unit.
- Possible $D+1$ values of ω_{ij} range from $-D$ to D.
- After obtaining $\mathcal{M}_{sim} = [\omega_{ij}]_{L\times L}$, for all the available data points being considered, we normalize each column of $\mathcal{M}_{sim}$ by dividing by the 1-norm of the column.
- After normalization, $\tilde{\mathcal{M}}_{sim}$ represents an empirical prior similarity matrix where the i^{th} column contains proportional (normalized) information about each class $\{1 \ldots L\}$ similarity to the i^{th} class.

Different representations are possible depending on the particular problem at hand.

It is important to note that even when formulation (1) is not convex, it is a box-constrained bi-convex optimization problem for which optimization algorithms with fast convergence rates can be applied as explained below.

Another interesting characteristic of formulation (1) is that strictly speaking, $\|W\|_1$ is not differentiable (around the origin). However, a simple and effective way to overcome this difficulty is the use of differentiable close approximations to the `L1` norm.

We will use the smooth approximation to the L1 penalty proposed in [14] that based on the following:

(i) $|x| = (x)_+ + (-x)_+$, where the *plus* function is defined as $(x)_+ = max\{x, 0\}$

(ii) The plus function can be approximated (smoothly), by the integral to a smooth approximation of the sigmoid function:

$$(x)_+ \approx p(x,\alpha) = x + \frac{1}{\alpha}\log(1 + \exp(-\alpha x)) \tag{4}$$

Combining these facts, we arrive to the following smooth approximation for the absolute value function consisting of the sum of the integral of two sigmoid functions (Fig. 1 plots this approximation near 0 for different values of α):

$$\begin{aligned} |x| &= (x)_+ + (-x)_+ \approx p(x,\alpha) + p(-x,\alpha) \\ &= x + \tfrac{1}{\alpha}\log(1+\exp(-\alpha x)) \\ &\qquad - x + \tfrac{1}{\alpha}\log(1+\exp(\alpha x)) \\ &= \tfrac{1}{\alpha}\left[\log(1+\exp(-\alpha x)) + \log(1+\exp(\alpha x))\right] \\ &= |x|_\alpha \end{aligned}$$

In practice, $\alpha = 10^5$ yields results that are within some small tolerance of the results produced by (optimal) constrained optimization methods. As opposed to the L1-penalty, this approximation is amenable to standard unconstrained optimization methods since it is twice-differentiable:

$$\nabla(|x|) \approx (1+\exp(-\alpha x))^{-1} - (1+\exp(\alpha x))^{-1}$$

$$\nabla^2(|x|) \approx 2\alpha\exp(\alpha x)/(1+\exp(\alpha x))^2$$

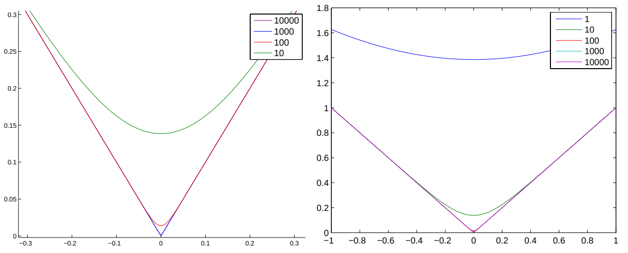


Figure 1: Approximation for $|x|$ near $x = 0$ using the function $|x|_\alpha$ for different settings of the parameter α

This approximation can be used in conjunction with any general likelihood or loss functions. It has been shown in [14] that for optimization problems derived from learning methods with L1 regularization, the solutions of the smooth approximated problems approach the solution to the original problems when α approaches infinity.

When dealing with large data sets, it is not computationally feasible to update the complete W and M at one single time, both of the two live in a large-dimensional space. Instead, we implemented a computationally inexpensive algorithm that updates W and M one column at a time respectively.

In order to understand our large-scale algorithm, it is useful to rewrite W in the form of combinations of column vectors, such that $W = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_L]$. Similarly M is written as combinations of row vectors $M = [\mathbf{m_1^r}' \; \mathbf{m_2^r}' \cdots \mathbf{m_L^r}']'$. Using these definitions, we could simplify formulation (1) as:

$$\begin{aligned} \min_{W,M} \mu \sum_{i=1}^{D} \|\mathbf{y}_i - \mathbf{x}_i \sum_{j=1}^{L} \mathbf{w}_j \mathbf{m_j^r}'\|^2 \\ + \sum_{j=1}^{L} \|\mathbf{w}_j\|_1 + \nu \|M - \tilde{\mathcal{M}}_{sim}\|^2_{\text{frob}} \\ \texttt{s.t.} \; : \; -1 \le M(i,j) \le 1 \quad i,j = \{1,\, 2, \ldots,\, D\}. \end{aligned} \tag{5}$$

The gradient and Hessian of the objective function of (5) with respect to each column vector $\mathbf{w}_j$ can be written as follows:

$$\nabla_{\mathbf{w}_j} f_2 = 2\mu(G_{XX}WM - G_{XY})\mathbf{m_j^r} + \phi_j \tag{6}$$

$$\phi_j(i) = (1 + \mathbf{e}^{-\alpha \mathbf{w}_j(i)})^{-1} - (1 + \mathbf{e}^{\alpha \mathbf{w}_j(i)})^{-1} \tag{7}$$

$$\Delta_{\mathbf{w}_j} f_2 \;=\; 2\mu(\mathbf{m_j^r}'\mathbf{m_j^r})G_{XX} + \Lambda \tag{8}$$

$$\Lambda(i,k) \;=\; \delta(i-k)\frac{\alpha \mathbf{e}^{\alpha \mathbf{w}_j(i)}}{(1 + \mathbf{e}^{\alpha \mathbf{w}_j(i)})^2} \tag{9}$$

where Λ is a $K \times K$ matrix and $\delta(\cdot)$ is the characteristic function. Similarly, using the same idea on M,we can rewrite it as the combination of column vectors: $M = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_L]$. Then, the gradient and the Hessian of the objective function of formulation (5) with respect to each column vector $\mathbf{m}_j$ is:

$$\begin{aligned} \nabla_{\mathbf{m}_j} f_2 &= 2(\mu W' G_{XX} W \mathbf{m}_j - W' G_{XY}^{(j)} \\ &+ \nu(\mathbf{m}_j - \tilde{\mathcal{M}}_{sim}^{(j)}) \end{aligned} \tag{10}$$

$$\Delta_{\mathbf{m}_j} f_2 \;=\; 2(\mu W' G_{XX} W + \nu I) \tag{11}$$

$G_{XY}^{(j)}$ and $\tilde{\mathcal{M}}_{sim}^{(j)}$ indicate the $j^{\texttt{th}}$ column of the corresponding matrices. By using equations (6-11) with box constraints on M, we can derive an effective implementation that is ideal for large scale problems. We used Quasi-Newton techniques in our implementation, helping our algorithm converge fast to a local minimum, in practice.

Without loss of generality, we set $\mathbf{m_x}$ and $\mathbf{m_y}$ to be 0, which could be achieved by preprocessing the data, so the optimal γ is zero given by $\gamma_{\texttt{opt}} = \mathbf{m_y} - \mathbf{m_x} WM$. Algorithm 1 box shows the large-scale inexpensive *SVM-sim* algorithm. In this algorithm, we set the initial W randomly, but one may also choose W from the result of one-vs-rest SVMs in order to expect faster convergence, and threshold ϵ usually depends on the scale of matrix W and thus depends on the scale of the multi-label problems themselves. A typical value for ϵ is 10^{-4}. This setting provided satisfactory results in most situations.

Algorithm 1 Large Scale SVM-sim Algorithm

```
input: $\nu$, $\mu$, $\epsilon$, $\tilde{\mathcal{M}}_{sim}$, $W_{\text{initial}}$
$W^{\text{new}} \leftarrow W_{\text{initial}}$, $W \leftarrow \mathbf{O}$
while $\|W^{\text{new}} - W\|_{\text{frob}} \ge \epsilon$ do
    $W \leftarrow W^{\text{new}}$
    $M \leftarrow M^{\text{new}}$
    for $i$ = 1 to $L$ do
        updating $\mathbf{w}_i$ by equation (6-8) using BFGS optimization techniques
    end for
    $W^{\text{new}} \leftarrow [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_L]$
    for $i$ = 1 to $L$ do
        updating $\mathbf{m}_i$ by equation (10-11) with box constraint $|M(i,j)| \le 1$ using trust-region-reflective algorithm
    end for
    $M^{\text{new}} \leftarrow [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_L]$
end while
return $W$ and $M$
```

Note that we used two publicly available optimization methods as part of our proposed algorithm:

1. **For updating** W, we used a solver based on the standard BFGS algorithm [12] with a few minor modifications as implemented in the optimization toolbox[2]. The BGFS algorithm is probably one of the most popular Quasi-Newton methods where a superlinear rate of convergence is achieved without the explicit expensive calculation of complete second order information (Hessian matrix).

2. **For updating** M, we used the standard 'trust-region-reflective' algorithm [8] as implemented in the *Matlab* optimization toolbox.

3. EXPERIMENTS AND DISCUSSION OF RESULTS

In this section, we describe our data in more detail, provide the pre-processing steps we performed to prepare our data, present our experimental set-up, report our results, and discuss these results.

3.1 The ICD-9 Data and Pre-Processing

Each document sample in the ICD-9 Data represents a recording of the events that occurs in a patient's hospital visit. The text in these documents are free-form notes regarding examinations, treatments, procedures, evaluations (examples include radiology notes,

[2] *http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html*

personal physician notes, lab tests, etc.). These documents are authored by different people with different qualifications (e.g., physician, nurse, radiologist, etc.). There are a total of 978 samples in this database. We apply a unigram feature representation, with each feature indicating the presence (represented as $+1$) or absence (0) of a word. As commonly done in text processing, we first removed stop words and applied stemming, resulting in a 1931-word dictionary. We then retained only words that occurred in at least 5% of the data samples, which leaves us with 1155 words. We centered our data to have zero-mean and normalized each feature to have variance of one. There are 140 different symptom or disease codes (classes) in this database. Due that the majority of the classes have very few training examples, in this study we only worked with the codes (classes) that occurred in at least 5% of the samples, corresponding to a total of 20 codes.

3.2 Experimental Set-Up

In our experiments we investigate whether or not taking advantage of the relationships among classes can improve the classification of ICD-9 codes over the standard one-vs-rest SVM, which learns the classifiers for each binary label separately. We also compare our technique (*SVM-sim*) with three other different types of multi-label algorithms: an SVM-based algorithm (*rank*-SVM [5]), a k-nearest-neighbor-based method (ML-KNN [18]), and a neural-network-based method (BP-MLL [17]). Besides checking for performance based on accuracy, another advantage of our approach is that it can learn the relationship among the classes. We also present the similarity matrices that *SVM-sim* discovers. We report our accuracies based on a five-fold cross-validation. Within the training set of each fold, we separate 10% for tuning. We tuned our parameters (μ and ν) for all the methods over 11 values ranging from 10^{-5} to 10^{5}. For the nearest neighbor algorithm, we tried $k = 1, 2, 4, 8, 16$ and reported the results from the best k together with the k value used. For *rank*-SVM, we tuned the cost parameter over eight values from 10^{-5} to 10^{2}. For BP-MLL, we set the number of neurons and number of hidden layers, learning rate and number of training epochs as suggested by the authors [17]. All results from the SVM-based classifiers here are with a linear kernel.

3.3 Results and Discussion

Table 1 reports the five-fold cross-validated accuracies for all five methods. In the table, we highlighted the best performing method for each code class in bold and in red. The higher the accuracies, the better. In Figure 2, we also present the accuracies (as shown on the (*left*) subfigure) for each class in ascending order for all the methods, and their corresponding standard deviations (as shown on the (*right*) subfigure). Note that our method, *SVM-sim*, performed the best in almost all of the classes and came in close second for the other cases. In all cases, our approach had the lowest variance in accuracies among the different folds. Note, too, how taking advantage of the relationship among classes drastically improved the results over one-vs-rest. This data set is a difficult classification task with only about $60\% - 80\%$ accuracy when manually labeled [1]; however, by taking advantage of relationships and data from the other classes helped improve our performance, where we reached accuracies between 95% to 99%.

In Figure 3, we also display the receiver operating characteristic (ROC) curves for all the methods and their corresponding area under the curve (AUC). We cannot show all of the 20 ROC plots; instead, we provide the average of the ROC plots for the 20 classes. Observe that our approach, *SVM-sim* still performed the best. In Table 2, we report the five-fold cross-validated area under the curve (AUC) results for all five methods on the ICD-9 data for all 20 classes (codes), with the best results highlighted in bold and in red. Note that our approach is the best in all cases except for class 18. We take a closer look at this worst case by plotting the ROC curves on class 18 as shown in Figure 5. Even though our approach is the worst in terms of AUC on class 18, it performed the best in terms of true positive detection at low false positive ratios (below 20%).

The results reveal that the multi-label classification of ICD-9 codes benefits from the sharing of information among the classes. The codes represent different symptoms/diseases, many of which are highly correlated to each other. For example, patients that suffer from a particular pain belongs to a subset of symptoms/diseases. The elements in that subset share many common words and are sometimes even just different descriptions of the same disease. Let us say, a patient has vesicoureteral problems ($C5$: vesicoureteral reflux). It is highly probable that this patient is suffering disorders of the urinary system type ($C2$). Inversely, symptoms that are located at the urinary system would have no correlation with asthma.

In contrast to all the other multi-label methods, we also learn the relationships among classes. $\tilde{\mathcal{M}}_{sim}$ and M in equation (3) are the prior and similarity matrices learned by *SVM-sim*. To highlight the relationship among the classes, we permute the rows and columns of this similarity matrix using co-clustering [4]. The co-clustering algorithm we implemented basically alternates k-means clustering of the rows and k-means clustering of the columns until convergence. After permutation, we obtain the matrix M. Note that the relationship matrix does not have to be symmetric. We can interpret the relationship from this matrix as a directed graph (showing only the strongest 10% relations). Figures 7 and Figure 8 display directed graphs showing the prior code relations $\tilde{\mathcal{M}}_{sim}$ and the learnt relations M respectively. These figures show that our approach besides improving our classification performance also provides us with information regarding the relationship among the classes (codes). For instance, Figure 7 (the graph for prior $\tilde{\mathcal{M}}_{sim}$) reveals that $C13$ (which stands for "Fever and other physiologic disturbances of temperature regulation") and $C18$ ("Lung field: Coin lesion lung, shadow lung") can help predict patients as having $C15$ ("Symptoms involving respiratory system and other chest symptoms"), which makes sense.

Comparing Figures 7 and 8, we find that the topology of relations changed during the learning process. The graph in Figure 8 (the graph for final M) shows that fever ($C13$), chest pain ($C16$) and respiratory abnormalities ($C14$) can cause respiratory symptoms ($C15$). The graph also shows that urinary tract infection ($C9$) and chest pain ($C16$) are associated with fever ($C13$). Note that the graph from the final M matrix makes more sense than the prior graph. In the prior relation graph, it indicates that bladder disorders ($C8$) is predictive of asthma ($C2$), which does not seem right. However, after the learning process, the graph from the final M shows that chest pain ($C16$), lung field ($C18$) and pulmonary collapse ($C3$) can be predictive of asthma ($C2$).

Besides learning the class code relations, our approach also learns the features that are important for each code (class) through the `L1` norm regularization in our formulation. We consider each column of W as a baseline classifier (the classifier before structure sharing by matrix M). Because of the `L1` norm regularization on W, the coefficients for each classifier will be sparse. In Figure 4, we report the number of features selected for each class. We considered a feature as being selected for class j if the absolute value of its weight is greater than 0.05 ($|W(i,j)| \geq 0.05$). Note that different classes selected different numbers of features and that we achieved substantial amount of reduction from the original 1155 features. In Table 3, we also list the selected words (features) for each class. We limit the list here to the top 5 words (in terms of absolute weight)

Table 1: Accuracies Obtained by the Different Methods on ICD-9 Data

Class / Methods	class 1 Pneu-monia	class 2 Asthma	class 3 Pulm. collapse	class 4 Rheu. fever	class 5 Vesic. reflux	class 6 Kidney adhesion	class 7 Neuro. bladder	class 8 Bladder disorder	class 9 Urinary infect.(uns)	class 10 Hema-turia
simSVM	**95.6%**	**98.6%**	**97.8%**	**97.4%**	**97.1%**	**96.6%**	**99.8%**	**98.3%**	**98.9%**	**99.8%**
1vsR SVM	91.3%	96.2%	96.8%	90.4%	88.0%	93.4%	95.0%	97.8%	85.1%	96.0%
rankSVM	95.2%	96.9%	97.1%	96.9%	95.8%	95.2%	99.0%	97.2%	97.7%	98.4%
BPMLL	92.5%	95.6%	96.6%	97.3%	95.0%	95.0%	97.2%	97.4%	98.5%	96.7%
MLKNN	92.9%	95.9%	96.3%	94.5%	89.3%	93.2%	95.0%	97.4%	93.1%	96.0%
Class / Methods	class 11 Hydroc-ephalus	class 12 Kidney anomal.	class 13 Fever	class 14 Resp. abnorm.	class 15 Cough	class 16 Chest pain	class 17 Urinary incont.	class 18 Lung lesion	class 19 Urinary infect.	class 20 Urinary disorder
simSVM	98.8%	**97.9%**	**95.3%**	**99.7%**	**95.9%**	99.1%	**99.3%**	**95.3%**	**96.3%**	**95.9%**
1vsR SVM	97.4%	97.0%	83.8%	95.0%	70.2%	96.0%	96.2%	94.6%	92.0%	95.3%
rankSVM	**98.9%**	97.4%	**95.3%**	99.1%	94.2%	**99.3%**	97.8%	94.2%	95.5%	93.8%
BPMLL	98.3%	91.5%	93.3%	96.3%	85.9%	98.1%	97.0%	94.8%	92.4%	92.2%
MLKNN	97.3%	97.0%	90.7%	95.4%	87.7%	98.1%	96.2%	94.3%	93.1%	94.8%

Table 2: The Resulting Area Under the Curve Obtained by the Different Methods on the ICD-9 Data

Class / Methods	class 1 Pneu-monia	class 2 Asthma	class 3 Pulm. collapse	class 4 Rheu. fever	class 5 Vesic. reflux	class 6 Kidney adhesion	class 7 Neuro. bladder	class 8 Bladder disorder	class 9 Urinary infect.(uns)	class 10 Hema-turia
simSVM	**0.915**	**0.974**	**0.984**	**0.991**	**0.977**	**0.942**	**0.995**	**0.929**	**0.983**	**0.994**
rankSVM	0.909	0.840	0.939	0.971	0.932	0.914	0.989	0.775	0.980	0.909
BPMLL	0.891	0.844	0.929	0.977	0.937	0.924	0.974	0.850	0.970	0.878
MLKNN	0.867	0.737	0.950	0.961	0.851	0.818	0.908	0.687	0.946	0.842
Class / Methods	class 11 Hydroc-ephalus	class 12 Kidney anomal.	class 13 Fever	class 14 Resp. abnorm.	class 15 Cough	class 16 Chest pain	class 17 Urinary incont.	class 18 Lung lesion	class 19 Urinary infect.	class 20 Urinary disorder
simSVM	**0.990**	**0.945**	**0.969**	**0.994**	**0.984**	**0.963**	**0.960**	0.828	**0.960**	**0.874**
rankSVM	0.981	0.802	0.940	0.963	0.963	0.955	0.853	0.833	0.931	0.828
BPMLL	0.984	0.822	0.911	0.891	0.844	0.955	0.829	0.853	0.916	0.865
MLKNN	0.833	0.751	0.905	0.887	0.929	0.952	0.771	**0.866**	0.927	0.791

per class. From the table, we observe that different classifiers do find different key words for the different classes and that the key words selected made sense and are associated to the class. We also noticed that class 13 & 15 use the largest amount of words (135 words for the 13th class, and 172 words for the 15th class), corresponding to fever and cough classes respectively. These two classes are symptoms that commonly occur in many diseases and are more general than other symptom/disease codes; thus, they need more words compared to other more specific symptom/disease codes.

Finally, we investigate how our prior class relationship structure, $\tilde{\mathcal{M}}_{sim}$, affects the performance of our classifier. The first $\tilde{\mathcal{M}}_{sim}$ is as what we suggested in Equation 3. Let us call this the similarity prior. We compare this against two other priors: (1) identity and (2) random. Identity assumes no correlation among the classes and serves as our baseline. The other prior assumes a random $L \times L$ matrix, where each element is sampled from a uniform distribution in the set of $[-1, +1]$. This prior assumes an arbitrary set of relations between the different classes. The average ROC results for the 20 classes are shown in Figure 6. We observe that the similarity prior outperforms the other two priors. This shows that knowing a good prior improves the performance of our classification. The figure also shows that even when we start with a bad prior, like random, we still obtain ROC results that are comparable with the other competing multi-label classifiers (see results on Figure 3).

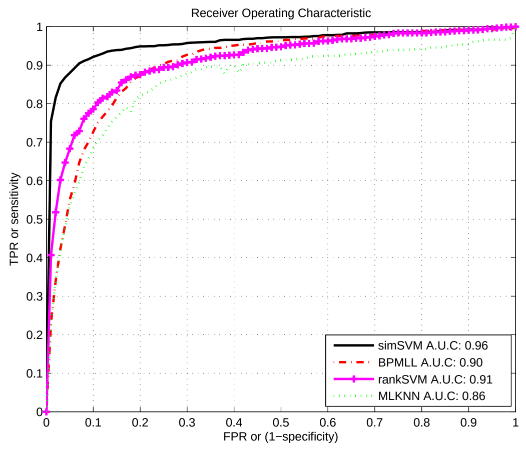


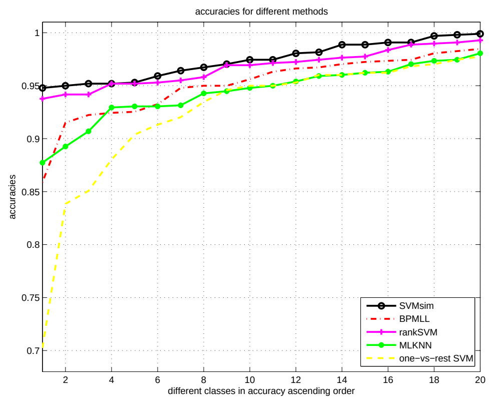
Figure 3: Results for ICD9 Text Data: Average ROC curves for the different methods from 20 codes.

4. CONCLUSIONS

Hospitals and healthcare providers rely on medical coding to record medical services and associated causes and conditions during a patient's visit. This coding is normally done manually. Given the large number of codes and their level of specificity for certain diagnosis, this is a very time consuming process and is error prone

Table 3: A list of words whose absolute weight value is larger than 0.05 for each medical code (class). Note that we only limit the list to up to five words per class.

class 1 Pneu-monia	class 2 Asthma	class 3 Pulm. collapse	class 4 Rheu. fever	class 5 Vesic. reflux	class 6 Kidney adhesion	class 7 Neuro. bladder	class 8 Bladder disorder	class 9 Urinary infect.(uns)	class 10 Hema-turia
postvoid brochial ultrasound pneumonia cystitis	lobe pneumonia wheezing asthma opacity	atelectasis collapse	hydron-ephrosis cough atelectasis pyelectasis	reflux hydron-ephrosis decrease time	pyelectasis reflux myelomen-ingocele cough	neurogenic pyelectasis	neurogenic hematuria pain enuresis	UTI tract urinary hematuria cough	hematuria tract UTI urinary cough
class 11 Hydroc-ephalus	**class 12 Kidney anomal.**	**class 13 Fever**	**class 14 Resp. abnorm.**	**class 15 Cough**	**class 16 Chest pain**	**class 17 Urinary incont.**	**class 18 Lung lesion**	**class 19 Urinary infect.**	**class 20 Urinary disorder**
hematuria infection	enuresis atelectasis hematuria lobe wheezing	fever cough atelectasis hematuria febrile	wheezing fever myelomen-ingocele ring	cough ring appearing fever effusion	wheezing bilateral bronchitis pain raise	pain enuresis duplication	pain hematuria neurogenic	pain infection tract urinary UTI	urinary urothelial



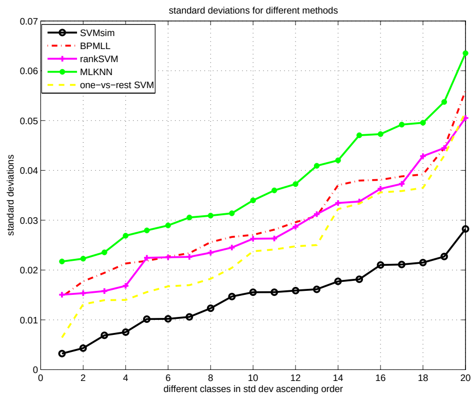


Figure 2: Results for ICD9 Text Data: Five-fold cross-validated accuracies (left) and standard deviations (right) for the different methods for each of the 20 codes.

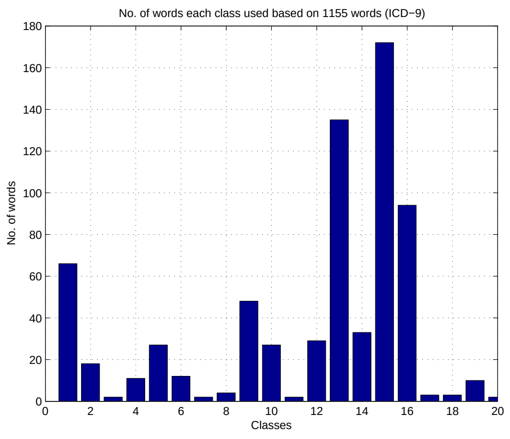


Figure 4: The number of features (words) selected by *SVM-sim* for each class out of the original 1155 words.

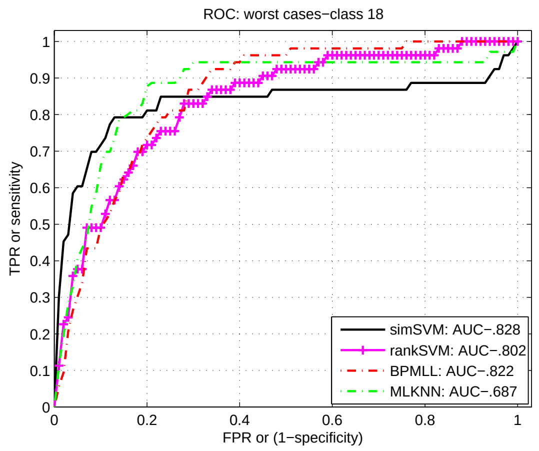


Figure 5: The ROC curves for Class 18, the worst AUC result for *SVM-sim*.

with an estimated accuracies of around only 60% to 80%. In this study, we have developed a multi-label classifier to automate the process of classifying ICD-9 codes from text recordings. We cast the medical coding problem as a multi-label classification problem because each text document can be classified to one or more codes (classes). The number of codes/classes is high and the occurrence of these codes may be few in a training set. Moreover, codes (which are associated with symptoms and/or diseases) are correlated. This

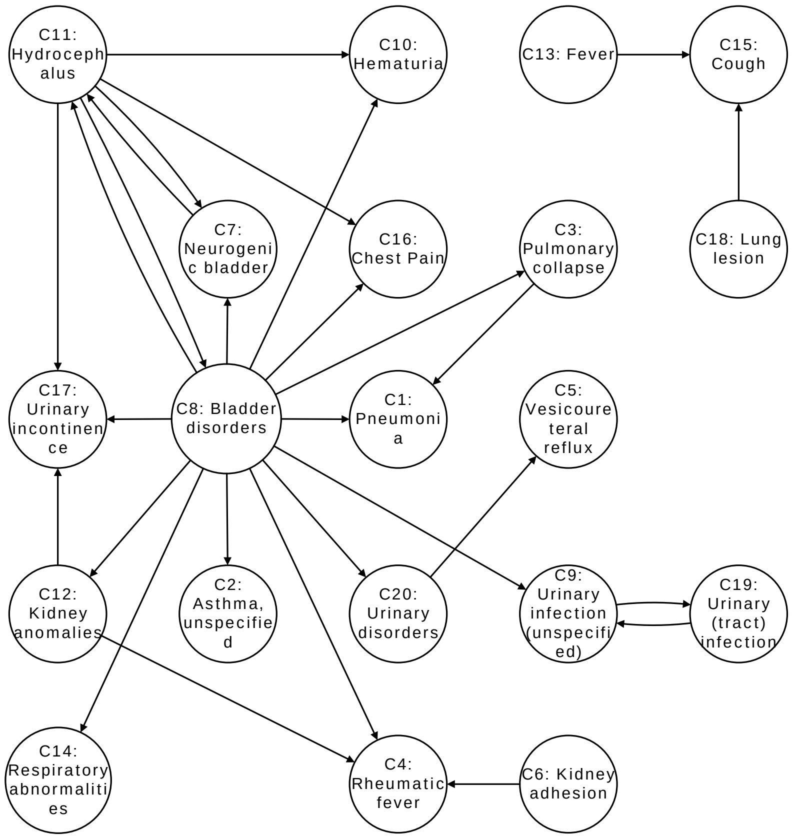


Figure 7: Class relationship graph for ICD9 Data, showing only the strongest 10% relations: Based on the prior $\tilde{\mathcal{M}}_{sim}$.

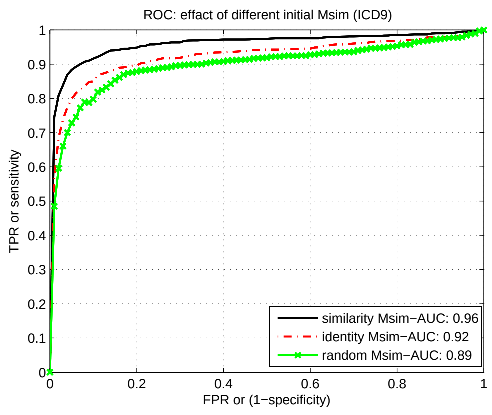


Figure 6: ROC results for three different initial prior class relationship matrix, $\tilde{\mathcal{M}}_{sim}$: similarity, identity, and random.

motivated us to take advantage of code/class relations to help build better classifiers for classes with few training samples. In this paper, we have introduced a multi-label large-margin formulation that explicitly represents the medical code/class structure and simultaneously learns the classifier and the code/class structure from the data. Moreover, our formulation enables the incorporation of prior knowledge about the structure automatically into the classifier. We found that sharing the code/class structure among the classifiers help improve the performance of each binary classification. Our experiments demonstrated the ability of our approach in discovering medical code relations. Furthermore, our results reveal that incorporation of prior domain knowledge helped *SVM-sim* to obtain classification performance superior to other multi-label approaches for classifying ICD-9 medical codes, where we reached accuracies between 95% to 99%.

Acknowledgments

This work is supported by NSF IIS-0915910.

5. REFERENCES

[1] C. Benesch, D. W. Jr, A. Wilder, P. Duncan, G. Samsa, and D. Matchar. Inaccuracy of the international classification of diseases ICD-9-cm in identifying the diagnosis of ischemic cerebrovascular disease. *Neurology*, 1997.

[2] M. Boutell, J. Luo, X. Shen, and C. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37:9:1757–71, 2004.

[3] K. Crammer and Y. Singer. A new family of online algorithms for category ranking. In *ACM SIGIR*, pages 151–158, 2002.

[4] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA, 2001. ACM.

[5] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *In Advances in Neural*

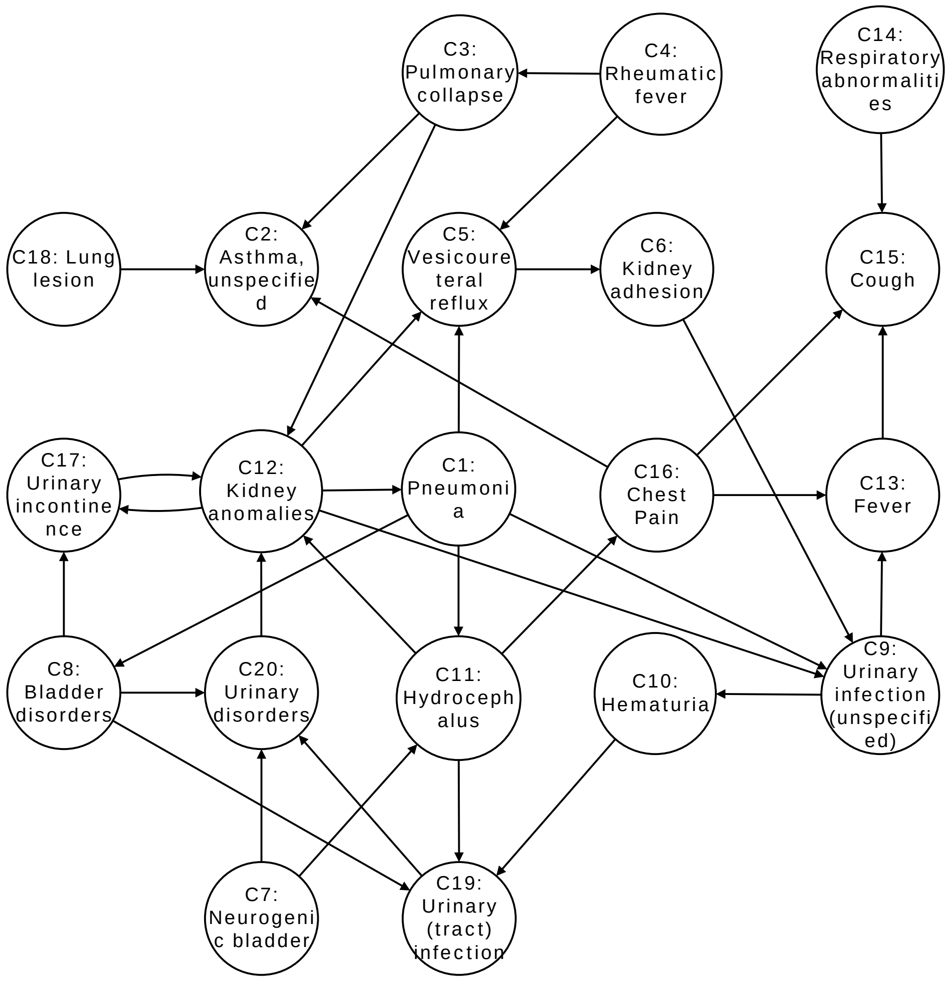


Figure 8: Class relationship graph for ICD9 Data, showing only the strongest 10% relations: Based on the final learned M.

Information Processing Systems 14, pages 681–687. MIT Press, 2001.

[6] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, 2003.

[7] http://www.icd9coding.com/.

[8] R. B. Jean, J. Charles, and G. J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 1996.

[9] L. Larkey and W. B. Croft. Automatic assignment of ICD9 codes to discharge summaries. IR IR-64, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, 1995.

[10] L. Lita, S. Yu, S. Niculescu, and J. Bi. Large scale diagnostic code classification for medical patient records. In *AIME*, pages 331–339, 1995.

[11] C. Lovis, P. Michel, R. Baud, and J. Scherrer. Use of a conceptual semi-automatic ICD-9 encoding system in a hospital environment. In *AIME*, pages 331–339, 1995.

[12] J. Nocedal and S. Wright. *Numerical Optimization (2nd ed.)*. Springer-Verlag, Berlin, New York, 2003.

[13] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. In *Machine Learning*, pages 135–168, 2000.

[14] M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pages 286–297, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] A. Sonel, C. Good, H. Rao, A. Macioce, L. Wall, R. Niculescu, S. Sandilya, P. Giang, S. Krishnan, P. Aloni, and R. Rao. Use of REMIND artificial intelligence software for rapid assessment of adherence to disease specific management guidelines in acute coronary syndromes. *AHRQ*, 2006.

[16] J. Weston, A. Elisseeff, B. Scholkopf, and M. Tipping. Use of the zero norm with linear models and kernel methods. *JMLR*, 3:1439–1461, 2003.

[17] M.-L. Zhang. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Trans. on Knowl. and Data Eng.*, 18(10):1338–1351, 2006.

[18] M.-L. Zhang and Z.-H. Zhou. A k-nearest neighbor based algorithm for multi-label classification. In *IEEE International Conference on Granular Computing*, volume 2, pages 718–721 Vol. 2. The IEEE Computational Intelligence Society, 2005.