ICD-9 Coding of Discharge Summaries

Luke Lefebure
Department of Statistics
Stanford University
`luke.lefebure@gmail.com`

Abstract

This paper investigates a neural network model for multi-label classification in the context of assigning ICD-9 codes to Intensive Care Unit (ICU) patient discharge summaries. An early procedure for multi-label classification using neural networks was BP-MLL which uses a novel pairwise ranking loss function for training [5], but recent research suggests a more modern architecture that uses a simpler cross entropy based loss produces better results [3]. The results on the discharge summary classification task using this approach fail to beat a simple baseline.

1 Introduction

Stringent privacy regulations have contributed to a slow adoption of information technology in the healthcare industry. However, with support from the federal government, an increasing number of medical practices are transitioning to electronic health record (EHR) systems for managing patient data. As more data is brought online, large scale analysis becomes possible.

In the United States, ICD-9 is the official system of assigning codes to diagnoses and procedures associated with hospital utilization [1]. Payers rely on these codes for processing claims, and an entire profession (medical coding) exists to manually review patient records and assign codes.

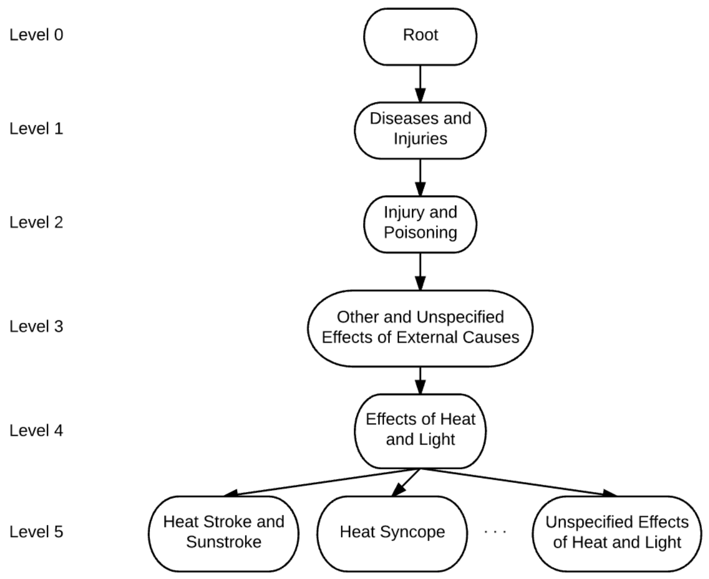
ICD-9 codes are organized in a hierarchical structure. Each code is a leaf node in the ICD-9 tree. For example, "Heat Stroke and Sunstroke" (code 992.0) and "Heat Syncope" (code 992.1) are both children of the "Effects of heat and light" node in the tree. Figure **??** shows the full sample path for these codes.

This project considers approaches for assigning ICD-9 codes to Intensive Care Unit (ICU) patient discharge summaries. Below is an excerpt from the discharge summary of a patient who was assigned codes for Cardiac Arrest, Acute Renal Failure, Congestive Heart Failure and a few others:

> CHIEF COMPLAINT: Admitted from rehabilitation for hypotension (systolic blood pressure to the 70s) and decreased urine output.
> HISTORY OF PRESENT ILLNESS: The patient is a 76-year-old male who had been hospitalized at the [**anonymized**] from [**anonymized**] through [**anonymized**] of 2002 after undergoing a left femoral-AT bypass graft and was subsequently discharged to a rehabilitation facility.
> . . .

As in this example, patients can be (and almost always are) assigned multiple codes. Therefore, I approach this problem as a multi-label classification task. Throughout this paper, I will use the term ICD-9 code to refer to the leaves of the ICD-9 tree and ICD-9 node to refer to a node in the ICD-9

[1]The U.S. healthcare system is in the process of transitioning to ICD-10, an updated and more complex version of ICD-9. However, this paper will exclusively consider ICD-9.



Level 0
Level 1
Level 2
Level 3
Level 4
Level 5
Root
Diseases and Injuries
Injury and Poisoning
Other and Unspecified Effects of External Causes
Effects of Heat and Light
Heat Stroke and Sunstroke
Heat Syncope
· · ·
Unspecified Effects of Heat and Light

Figure 1: Sample Path in the ICD-9 Hierarchy

tree (including the leaves). Additionally, we say that a non-leaf ICD-9 node is assigned to a training example if and only if at least one of its children is assigned to that training example.

2 Related Work

The particular problem of assigning ICD-9 codes to discharge summaries has previously been studied with Support Vector Machines [4]. In that paper, the authors consider two different approaches for predicting the entire ICD-9 hierarchy (i.e. each node in the ICD-9 hierarchy is considered a class).

The first, called the flat SVM, involves training a binary classifier on each node in the ICD-9 hierarchy (excluding the root) independently using all of the data. Then at test time, the final multi-label prediction is constructed by taking the set of positive predictions from all of these classifiers and augmenting it with their ancestors in the tree that don't already belong to the set.

The second approach, called the hierarchy-based SVM, differs from the flat SVM in two ways. First, during training, the binary SVM for a particular node is trained on only the examples for which that node's parent is positive (the root is always considered positive). Second, during testing, the trained SVM for a node only makes a prediction if the SVM for the parent node predicted positive. Otherwise that node is predicted negative.

The authors use a bag of words model to represent documents. They first filter out the 10,000 tokens with the highest tf-idf scores across the training set. Then they represent each document as a 10,000 dimensional vector where the i'th element of that vector is the count of occurrences of token i in the document.

3 Approach

3.1 Document Representation

There are many ways to represent the patient discharge summaries as input vectors. Previous research provides code for representing these documents as bag-of-words vectors [4]. In particular, it takes the 10,000 tokens with the largest tf-idf scores from the training set and uses those as the vocabulary for bag-of-words. I will experiment with using this representation as well as a lower dimensional SVD representation on the Document-Term matrix.

3.2 Model

In multi-label classification, the label vector is no longer a one-hot vector. Let $Y = \{1, 2, 3, \dots\}$ denote the set of all possible class labels. Then, for the i'th training example $\boldsymbol{x}_i$, the associated label vector $\boldsymbol{y}_i$ is of dimension $|Y|$ and:

$$y_i^\ell = \begin{cases} 1 & \text{class } \ell \text{ assigned to } i'\text{th training example} \\ 0 & \text{class } \ell \text{ not assigned to } i'\text{th training example} \end{cases}$$

Previous research suggests the following architecture [3]. The model consists of one hidden layer, with relu activation on the hidden layer and sigmoid activation on the output layer:

$$\boldsymbol{h} = \text{relu}(\boldsymbol{x}_i \boldsymbol{W} + \boldsymbol{b}_1)$$

$$\boldsymbol{o} = \text{sigmoid}(\boldsymbol{h}\boldsymbol{U} + \boldsymbol{b}_2)$$

The dimensions of the input and model parameters are given by

$$\boldsymbol{x}_i \in \mathbb{R}^{1\times d} \qquad \boldsymbol{W} \in \mathbb{R}^{d\times k} \qquad \boldsymbol{b}_1 \in \mathbb{R}^{1\times k} \qquad \boldsymbol{U} \in \mathbb{R}^{k\times c} \qquad \boldsymbol{b}_2 \in \mathbb{R}^{1\times c}$$

where d is the dimension of the input vectors and k is the dimension of the hidden layer.

3.3 Loss

Many different loss functions have been proposed for multi-label classification, including pairwise ranking loss and cross entropy [5][3]. Pairwise ranking loss was used in early research[5], but recent research suggests that cross entropy produces better results and is far more computationally efficient than the original pairwise ranking loss [3]. Therefore, I will use cross entropy loss, which is given by the following for a single training example:

$$J(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{y}_i) = -\sum_\ell y_i^\ell \log(o_\ell) + (1 - y_i^\ell)\log(1 - o_\ell)$$

To compute the loss for the training set, I sum this metric over batches of training examples. Finally, I also include an $L2$ penalty in my implementation.

3.4 Response

There are over 5,000 ICD-9 codes which makes predicting them directly very challenging. However, most are very infrequent. To restrict the scope of this project, I will consider two approaches. First, I will predict only the 20 and 50 most frequent codes in the training set. Second, I will predict nodes in the second level of the ICD-9 hierarchy.

3.5 Evaluation

I use the rank loss metric to evaluate performance. Rank loss operates on the ordering of the predicted probabilities. For the i'th training example, it is defined as follows. Let $Z_i = \{\ell : y_i^\ell = 1\}$ be the set of labels assigned to the i'th training example, and let $\bar{Z}_i = Y/Z_i$. Then the rank loss is

$$\frac{|D_i|}{|Z_i||\bar{Z}_i|}$$

where $D_i = \{(x, y) : \boldsymbol{o}_x < \boldsymbol{o}_y, x \in Z_i, y \in \bar{Z}_i\}$.

4 Experiments

4.1 Data

I will use data from the MIMIC-II clinical database for this project [2]. This database consists of 22,815 discharge summaries from ICU patients labeled with ICD-9 codes. I split the data into training, test, and dev sets consisting of 80%, 10%, and 10% of the full data set respectively.

4.2 Input

As mentioned above, I consider two different approaches for input features. The first is a 10,000 dimensional bag-of-words vector where the i'th feature is an indicator of whether token i is present in the document ($d10,000$ input). The second is a 100 dimensional vector resulting from SVD on the Document-Term matrix from the first representation ($d100$ input).

4.3 Response

There are 5,030 ICD-9 codes and 7,042 ICD-9 nodes represented in the entire dataset. As mentioned above, I explore two types of responses in the analysis that follows. First, I predict only the X most frequent ICD-9 codes in the training set (top X output for X = 20, 50). Second, I attempt to predict nodes in the second level of the ICD-9 hierarchy of which there are 49 (level two output).

Level	Count
0	1
1	3
2	49
3	309
4	1345
5	2821
6	2109
7	405

Table 1: Counts of ICD-9 Nodes by Level in the ICD-9 Tree

Min	1
Max	39
Mean	9.46
Median	9

(a) ICD-9 Codes per Training Example

Min	1
Max	20
Mean	5.43
Median	5

(b) Level 2 ICD-9 Nodes per Training Example

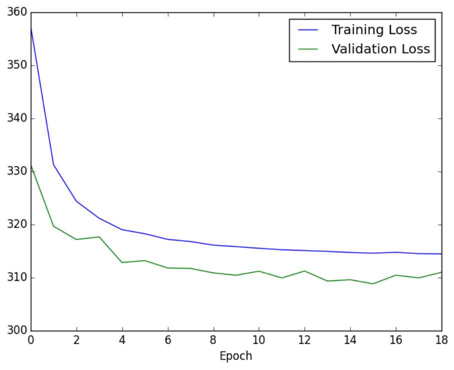
Figure 2: Label Summary Statistics

4.4 Baseline

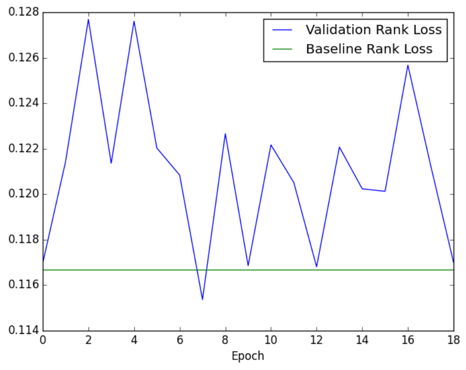
I evaluate results using the rank loss metric described above. As a baseline, I consider a model that simply predicts labels in order of their frequency in the training set for all documents.

4.5 Results

Figure 3 shows the result of training the model with $d100$ input and level two output. The corresponding plots produced by the other input/output combinations all produced very similar patterns. Table 2 shows the numeric results from all of the models. These results are presented using the best training parameters found according to validation loss. In particular, for $d100$ input, 100 hidden units and .02 L2 penalty were used, while for $d10000$ input, 500 hidden units and .04 L2 penalty were used. Learning rate of .01 was used.



(a) Loss During Training



(b) Validation Rank Loss during Training

Figure 3: Training on d100 input and level 2 output

Input	**Output**	**Validation Rank Loss**	**Validation Baseline Rank Loss**	**Change**	**Test Rank Loss**	**Test Baseline Rank Loss**	**Change**
$d100$	level 2	.117	.117	0	.151	.139	+.012
	top 20	.302	.331	-.029	.338	.315	+.023
	top 50	.293	.317	-.024	.297	.252	+.045
$d10000$	level 2	.123	.117	+.006	.164	.139	+.025
	top 20	.317	.331	-.014	.391	.315	+.086
	top 50	.358	.317	+.041	.282	.252	+.030

Table 2: Results Table

5 Conclusion

My model achieves roughly the same performance according to rank loss as one that predicts label frequencies from the training set. Decreases in validation loss fail to correspond to decreases in validation rank loss which is inconsistent with previous findings [1]. I think that my model suffered because of a large number of classes, an extremely uneven label distribution, and the fact that documents are assigned many different labels. These facts combined with the limited training set mean that the model has trouble learning features of the rare labels (which make up a majority).

I could restrict the number of output classes further but that would make the task less and less valuable. Moving forward, I think training on more data would help performance the most. An updated version of the dataset that I used for this analysis that is double the size is now available (MIMIC-III [2]). With more data, there are many parts of the model that I could change.

I could experiment with different document representations for input. For example, instead of using a binary representation in the bag-of-words model, I could fill a document vector with counts of tokens or tf-idf scores of tokens. An entirely different approach would be to represent documents as the sum or average of the word vectors of the five or ten tokens in the document with the highest tf-idf scores. However, because there are a lot of complex medical terms in these documents that occur very infrequently, training word vectors could be a challenge.

I could also experiment with different model architectures. For example, the discharge summaries have some structure (e.g. sections for "Chief Complaint", "History of Present Illness", and more). I could represent each of these sections as a vector using one of the approaches described above and then use these vectors as sequential inputs to a Recurrent Neural Network.

Through this project, I learned a lot about how to formulate a task as a problem that can be tackled with neural networks. I reviewed literature and explored novel architectures to find a model that fit my problem. Despite the poor performance of the final model, I also learned to appreciate the importance of tuning parameters. I found that carefully tuning parameters led to big improvements in the final convergence of my validation loss curve during training. While the results of the final model were disappointing, I gained many practical skills through this project for approaching learning tasks with neural networks.

References

[1] K. Dembczynski, W. Kotlowski, and E. Huellermeier. Consistent Multilabel Ranking through Univariate Losses. *ArXiv e-prints*, June 2012.

[2] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3, 2016.

[3] J. Nam, J. Kim, E. Loza Mencía, I. Gurevych, and J. Fürnkranz. Large-scale Multi-label Text Classification - Revisiting Neural Networks. *ArXiv e-prints*, December 2013.

[4] A. Perotte, R. Pivovarov, K. Natarajan, N. Weiskopf, F. Wood, and N. Elhadad. Diagnosis code assignment: models and evaluation metrics. *Journal of the American Medical Informatics Association*, 21(2):231–237, 2014.

[5] M.L. Zhang and Z.H. Zhou. Multi-label neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18:1338–1351, 2006.