Machine Learning CS 726, Fall 2011 *Christoph Schulze, Michael Whidby, and Kotaro Hara*

Project 1: Basic classification

1 WU1

2 WU2

3 WU3

4 WU4

5 WU5

6 WU6

7 WU7

8 WU8

9 WU9

10 WU10b

We decided to work on collective classification. We use a CiteSeer dataset shared on the Statistical Relational Learning Group's webpage. (http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html). The following description of the dataset is excerpted from the webpage. *The CiteSeer dataset consists of 3312 scientific publications classified into one of six classes. The citation network consists of 4732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words.*

We implement a stacking algorithm that is described at the end of the chapter 5 of the textbook (Algorithms 20 and 21.) We implement two flavors of the stacking algorithm (to see which one performs better):

- Simple. This implementation will only use predictions on neighbors from a previous layer. (Neighbors are examples cites/are cited by the example)
- Cumulative. This implementation will take all the predictions on neighbors until the Kth layer $(y_1, ..., y_{k-1})$ to train the Kth layer's classifier.

On the first layer, we simply use Megam in our script to train a multi-class classifier. Output (predictions) from Megam is parsed and stored inside prediction data. On each of succeeding layers, we first generate training data by combining an original training data (training data from the first layer) and predictions from a previous layer. I.e. we append predicted labels of neighbors into the training example as features. Then we train a classifier for the layer using Megam.

We split the dataset into five parts for cross-validation and calculate a training error and a test error for each layer. For each validation, we train classifiers with 2650 examples and test with 662 examples.

Figures show training errors and testing errors against layer indexes. On the first layer of the simple implementation, training error was about 11%. The classifier is trained only with word features. On the second layer, the error went up to 13% as we append predictions from the first layer into the training dataset. For the rest of the layers, training error did not change from 13%.

The testing error was 30% on the first layer. Testing error did not change more than 0.1% for any of following layers.

The training error on the cumulative implementation was 11% on the first layer. On the second layer, it went up to 13%. After the third layer, training error started to decrease, and on the fourth layer, the training error became almost 0. Then the training error stayed close to 0.

The testing error was 30% at the beginning, but it showed a slight improvement around layer 5 and 6. Errors went down to 29%.

Both simple and cumulative implementations of the stacking algorithm did not show a significant improvement in test accuracy. We believe there are several reasons:

- Dataset's network is sparse. There are more word features compared to neighbor features. Therefore neighbors' labels are less influential for classification.
- Neighbors' labels do not contain much information. Collective classification assumes neighbors' labels are informative because ML papers are more likely cited by other ML papers. However, in this dataset, AI paper can certainly cite ML papers, and so for others.

Another problem. Scalability (related to overfitting from the cumulative version)

Conclusion. Stacking did not really help.