CSE 573 Mini-Project: Co-Training and Naive Bayes

Anthony Fader

December 11, 2008

1 Introduction

Co-training [1] is a popular semi-supervised machine learning algorithm that leverages multiple "views" of a dataset to reduce the classification error. In order to apply the co-training algorithm, the views must satisfy these conditions:

C1 Each view alone must be sufficient to learn the class.

C2 The views of an instance must be conditionally independent given the class label of the instance.

For example, in [1] Blum and Mitchell apply co-training to the task of web page classification. Each web page P has one view composed of words on P and another view composed of words in hyperlinks to P. Despite the fact that these two views are probably not conditionally independent, co-training achieves half the error rate of a standard supervised learning algorithm.

Unfortunately, the possible applications of co-training are limited to cases where there are multiple views of a dataset, and even in those cases, the dataset still must satisfy conditions **C1** and **C2**. In this report, I investigate whether co-training can be extended to cases where the data does not have a natural feature division and when condition **C2** is not met.

Specifically, consider the the Naive Bayes assumption, where instances X are represented by n features $x = (x_1, \dots, x_n)$ and each feature is conditionally independent given the class of x. This assumption is much stronger than condition **C2** above in that any two sets of features V and $\bar{V}$ are conditionally independent "views" of the data. So if the Naive Bayes assumption is made, and assuming that there is some amount of redundancy in the features (so **C1** is satisfied), a simple co-training algorithm is:

1. Randomly split the feature set into two roughly equal groups, V and $\bar{V}$.
2. Apply the co-training algorithm on the views V and $\bar{V}$ using Naive Bayes classifiers.

Why would this work? One argument might be that although the Naive Bayes assumption is unrealistic, Naive Bayes classifiers tend to work well in practice. Since the Naive Bayes assumption is stronger than the co-training condtions, co-training might also be successful.

The rest of this report roughly recreates the results from [2].

2 Co-training and Self-training Algorithms

There are many different co-training algorithms to choose from, each with slight variations. In the experiments, I used the following co-training algorithm:

- Input: labeled data L, unlabeled data U, parameters k and m
- Iterate k times:
    1. Train a Naive Bayes classifier C_1 on the first view of L
    2. Label U using C_1
    3. For both positive and negative labels, choose m instances from U that were most confidently labeled by C_1 and move them from U to L
    4. Train a Naive Bayes classifier C_2 on the second view of L
    5. Label U using C_2
    6. For both positive and negative labels, choose m instances from U that were most confidently labeled by C_2 and move them from U to L
- Combine C_1 and C_2 into a classifier C
- Return C

The measure of confidence used was the log likelihood ratio between the two classes. The two classifiers C_1 and C_2 are combined into C by returning the label that maximizes the product of the likelihoods given by C_1 and C_2.

The self-training algorithm I used is very similar to the co-training algorithm above, only it uses a single view and a single classifier.

3 Experimental Setup

As a test, I compared an implementation of the above algorithm to a supervised Naive Bayes classifier. Also, since the random feature split algorithm might be more accurate than a supervised Naive Bayes classifier simply because it is taking unlabeled into account, I compared it to a self-trained Naive Bayes classifier.

The data I used were two categories from the 20 Newsgroup dataset (`comp.graphics` and `alt.atheism`). There were a total of 1,772 posts, with roughly equal proportions from each category. I split the data into 25% training data and 75% test data. Within the training data, I split it into 50 posts to act as labeled data and 393 posts to act as unlabeled data. The documents were pre-processed by removing stop words, punctuation, and words that occurred in less than 5 documents.

The supervised Naive Bayes classifier was trained on the 50 labeled training posts and then tested on the test data.

Algorithm	**# labeled**	**# unlabeled**	**accuracy**
Naive Bayes	50	-	76%
Self-training	50	393	51%
Co-training (random split)	50	393	80%
Naive Bayes (all)	443	-	96%

Table 1: The results of the experiments. The accuracy is the average accuracy for 5 random splits of the data.

The semi-supervised algorithms were given the 50 labeled examples as seeds (the L argument to the algorithm) as well as the remaining 393 unlabeled examples (the U argument). Each algorithm was allowed to label $m = 8$ unlabeled data points at each iteration for $k = 10$ iterations. The supervised Naive Bayes classifier was also tested on the full 25% of the training data for comparison. Each algorithm was tested 5 times for random splits of the data.

4 Results

Table 1 shows the results of the experiments. Surprisingly, the self-training algorithm performed much worse than the Naive Bayes algorithm. I suspect that the self-training algorithm suffered from overfitting the initial seed labeled data. (On many iterations, it labeled almost all data as a single class, which is why it is performing at near-baseline accuracy). For a fixed amount of labeled data, the co-training algorithm outperforms both the Naive Bayes algorithm and self-training algorithm.

5 Discussion

This experiment gives preliminary evidence that when the Naive Bayes assumption is used, accuracy can be improved by applying co-training with a random split of the data. Some next steps might be to investigate ways to identify "latent" splits in the data by looking for conditionally independent features. At the end of [2], the authors outline an algorithm for finding a good split in the data that attempts to minimize the conditional mutual information between the two feature sets. I implemented a version this algorithm in my code that uses a spectral graph cut technique, but I did not have time to run it on the data.

On a more technical side, my implementations of these algorithms could be improved. (The poor performance of the self-training algorithm is suspicious, there might be a bug in my code, but I couldn't find it in the limited amount of time for this project.)

6 Implementation Notes

I implemented all of the algorithms described in this paper. I used some basic text processing software from Python's NLTK library for pre-processing the documents.

References

[1] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *In Proceedings of COLT*, pages 92–100, 1998.

[2] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, pages 86–93, 2000.