Large Scale Semi-Supervised Document Classification

Christopher Clark
Department of Computer Science
University of Washington
Seattle, WA, 98195
chrisc36@uw.edu

Coauthor Kevin Clark
Department of Computer Science
University of Washington
Seattle, WA, 98195
kevinc22@uw.edu

Abstract

We demonstrate how two popular text classification algorithms can leverage a pool of unlabeled documents to enhance their performance. Our first technique uses Expectation-Maximization (EM) alongside a Multinomial Naive Bayes (MNB) model. The classifier is trained on the labeled data and used to generated probabilistic labels for the unlabeled data. The classifier is then trained using all the documents and regenerates label estimates for the unlabeled data. We also show how the algorithm can be run efficiently in a mapreduce framework. Additional we integrate unlabeled data into the k-nearest neighbor (KNN) algorithm through the use of the semi-supervised condensed KNN algorithm. We demonstrate how kernel hashing and ball trees can be used to efficiently run this algorithm over large datasets.

1 Introduction

We focus on text classification, but the techniques discussed here could generalize to other domains. Textual data is often abundant in places like digital records, electronic archives, and company databases but labeling that data often must be done by hand. This can make training many conventional algorithms overly expansive, as labeling sufficient data for these classifiers to achieve high accuracy is costly. In this paper, we follow the approaches used by Nigam at al. [1] and Su et al. [2] to show how unlabeled data could be used to improve generative classifier parameter estimation when there is little labeled data, reducing the need to label large numbers of examples. We also give an example of how non-parametric algorithms can be augmented through the incorporation of unlabeled data by evaluating a semi-supervised extension of the k-nearest-neighbor algorithm described in [3].

The amount of available data in the world is growing at a rapid rate, with sources ranging from social media to scientific studies. Because of this, it is becoming increasingly important to develop algorithms that can efficiently handle large data sets in a distributed way. To address this, we scale up the approaches through using the mapreduce framework and kernel hashing, a dimensionality reduction technique.

2 Multinomial Naive Bayes and EM

2.1 Multinomial Naive Bayes

One common text classification model is a Multinomial Naive Bayes classifier (MNB). Consider the task of classifying a list of documents as belonging to a set of discrete classes. We represent each document as a bag-of-words taken from a vocabulary V consisting of words $\langle w_1, w_2...w_{|V|}\rangle$. For a list of documents $D = \langle d_1, d_2, ...d_{|D|}\rangle$, we represent a document d_i as a a vector of words counts $d_i = \langle w_{i,1}, w_{i,2}...w_{i,|V|}\rangle$ where $w_{i,n}$ denotes the number of times w_n appears in d_i. We also have a list of potential classes for each document $Y = \{y_1, y_2, ...y_{|Y|}\}$. To facilitate our EM integration, we generalize the standard algorithm by allowing the documents to have mixed membership. Thus for a every document d_i we assume we have a list $\langle c_{i,1}, c_{i,2}, ..c_{i,|Y|}\rangle$ where $c_{i,n}$ denotes the probability d_i belongs to class y_n . When considering labeled data, for any document d_i labeled y_j we would set $c_{i,k} = 1$ for $k = j$ and $c_{i,k} = 0$ if $k \neq j$.

A MND is determined by a set of parameters we denote as Θ. Let $\Theta_{w_t|y_j} = P(w_t|y_j;\Theta)$, in other words $\Theta_{w_t|c_j}$ represents the probability w_t appears in a document of class c_j. A MNB model also includes a set of priors for each class written as Θ_{y_j}. representing our prior belief the probability a randomly selected document will belong to class y_j Training a MNB consists of estimating the parameters to Θ given a list of documents and list of labels.

We estimate Θ by setting:

$$\Theta_{y_j} = \frac{\sum_{i=1}^{|D|} c_{i,j}}{|D|}, \qquad \Theta_{w_j|y_j} = \frac{k + \sum_{i=1}^{|D|} c_{i,j} w_{i,j}}{|V| * k + \sum_{n=1}^{|D|} \sum_{v=1}^{|v|} c_{i,j} w_{i,v}}$$

where k is a smoothing parameter. Now for an unlabeled document d_k using a given Θ we can calculate:

$$P(y_i|d_k,\Theta) = \Theta_{y_j} \prod_{i=1}^{|V|} (\Theta_{w_i|y_j})^{w_{k,i}}$$

For classification purposes we can predict a class $\hat{c_k}$ for d_k by setting:

$$\hat{c_k} = \arg\max_{y_j} P(y_i|d_k,\Theta) = \arg\max_{y_j} (\log(\Theta_{y_j}) + \sum_{i=1}^{|V|} \log(\Theta_{w_i|y_j}) * w_{k,i})$$

To select the smoothing parameter and the vocabulary, a grid search was performed on a validation set. Our vocabulary was selected as the set of words in the training data. We found we got the best results when we converted all words to lowercase, removed stop words, removed words that occurred only once, and removed words that occurred in over 50% of the documents. We found $k = 0.40$ was the optimum smoothing parameter.

2.2 Using EM to Incorporate Unlabeled Data

The first step of our algorithm is to build a standard MNB using the labeled data only to get an initial estimate of Θ, $\hat{\Theta}$. Next we use Expectation Maximization in an attempt to leverage the unlabeled data to improve our model's parameters. We do this following standard EM procedure except we do not alter the label of the labeled data in the E-step. Particularly:

- E-step. We calculate a mixed membership label prediction for the unlabeled data. For a document d_k:
$$\forall j \in \{1,2,3...|Y|\}\ c_{k,j} = \frac{P(y_j|d_k,\hat{\Theta})}{\sum_{n=1}^{|Y|} P(y_n|d_k,\hat{\Theta})}$$
- M-step. We reestimate $\hat{\Theta}$ using both the labeled and unlabeled examples.

We repeat the E-step and M-step until $\hat{\Theta}$ converges.

Since there are usually many more labeled than unlabeled examples, the unlabeled documents dominate the EM and skew the parameter estimates. To fix this, we introduce a new parameter λ determining the weight unlabeled documents. We set $\lambda = 0.5$, which we found to provide good results in the experimental evaluation.

2.3 Semi-supervised Frequency Estimation

Semi-supervised Frequency Estimate (SFE) is a different extension to the MNB algorithm proposed by Su et al [2]. It runs like the EM algorithm described above except instead of weighting word counts by class membership probabilities for each document, it uses the class membership probabilities for each word obtained from the supervised learning. That is,

$\Theta_{w_t|y_j}$ is set to

$$\Theta_{w_t|y_j} = \frac{k + \sum_{i=1}^{|D|} P(c|w_i)_l w_{i,j}}{|V| * k + \sum_{n=1}^{|D|} \sum_{v=1}^{|v|} P(c|w_i)_l w_{i,v}}$$

.

Although this method performed worse than EM in our evaluation, there are other advantages to using this approach. Semi-supervised learning methods require large amounts of unlabeled data and thus need to scale well with the number of unlabeled documents. EM+MNB is a factor of $|C|$ slower than SF because EM requires a probability estimate for each unlabeled document belonging to each class while SF does not. Su et al. also found it outperformed EM on some data sets.

2.4 Scaling MNB and EM

To increase the algorithms ability to handle large datasets, we developed a parallelized version that can run on the mapreduce framework. Our implementation requires one mapreduce job per an iteration of EM. The full algorithm is as follows:

Preprocess: We precompute the words counts per a class and number of documents per class of the labeled data. We use the labeled data to compute our current parameter estimate for the MNB, $\hat{\Theta}$.

EM: while the algorithm has not converged, start a mapreduce job with as follows:

- Mapper: Accepts lists $\langle w_1, w_2 ... w_{|V|}\rangle$ of word occurrences for an unlabeled document. The mapper uses $\hat{\Theta}$ the current parameter estimate to calculate the predicted class probability distribution of current document. It then adds the word count weighted by class probability to a running total or weighted word counts per class. Likewise it adds the class probability distribution to a running total of weighed document counts. When all input is processed emit $\{y, \{wc, d\}\}$ key-value pairs where y is a class, wc is an array of length $|V|$ of weighted words counts for the class, and c is the weighted document count of that class.
- Reducer: Accepts $y, \langle\{wc, c\}\rangle$ key-value pairs. It sums the words counts and document counts and emits the resulting $\{wc, d\}$ tuple

Retrain: We add the words counts per class and documents counts per class emitted from the reducers with our words counts and documents per class we precomputed for the labeled data in our preprocessing step multiplied by however much we wish to weight the labeled examples. Together this composes our new $\hat{\Theta}$ estimate.

3 Semi-Supervised Condensed k-Nearest-Neighbor

3.1 Condensed k-Nearest-Neighbor

The k-nearest neighbor (KNN) algorithm is another popular approach to document classification. KNN classifies an object based on the closest training examples in the feature space. Given a set of labeled points $L = \{(x_1, y_1, ..., (x_N, y_N)\}$ and new data point x, the algorithm finest finds L_k, the k nearest points in L to x according to the Euclidean distance metric. The set L_k provides a confidence score c_{L_k} that example x_i has label y, which is given by a weighted vote of the nearest points.

$$c_{L_k}(x, y) = \frac{1}{k} \sum_{(x', y') \in L_k, y' = y} 1/||x' - x||_2^2$$

The predicted label $\hat{y}$ of x is the label with the highest confidence:

$$\hat{y} = \arg\max_y c_{L_k}(x, y)$$

The condensed k-nearest neighbor (CKNN) algorithm, does this process on a subset S of L, which can lead to faster and more accurate classification. However, choosing the best such subset is intractable. CKNN is a simple technique for picking an approximation of the best subset. It does this by only including data points that are not predicted well by the current set of points. The CKNN algorithm is defined below.

Algorithm 1 Condensed k-Nearest-Neighbor

```
Let $L = \{(x_1, y_1, ..., (x_N, y_N)\}$ be a set of labeled examples.
Let $S = \emptyset$
**for** $(x_i, y_i) \in L$ **do**
    Let $S_k$ be the set of at most $k$ points $(x, y) \in S$ with the smallest values of $||x - x_i||_2^2$.
    **if** $\arg\max_y c_{S_k}(x, y) \neq y_i$ **then**
        $S = S \cup \{(x_i, y_i\}$
    **end if**
**end for**
**return** $S$
```

3.2 Incorporating Unlabeled Data

Intuitively, CNN tries to reduce the set of points to those that best represent each cluster, generally points near cluster centers. However, if there are not very many labeled data points, points near cluster centers may not be included. For example, in the figure below it takes several labeled points to represent a cluster for the data. However, if we augment the labeled data with a large number of unlabeled points, there will be a higher chance of finding points near cluster centers. In the case of the figure below, an unlabeled point provides a better representation of the cluster.

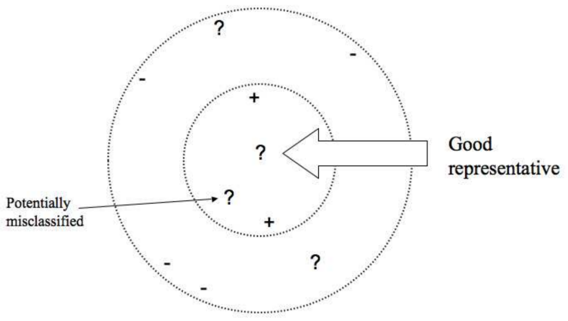

Figure 1: Unlabeled data can help provide a better representation in condensed training sets. In the figure using the center unlabeled point as a positive point in a kNN model will help us classify points in the center region as positive.

Of course, adding an unlabeled point to the cluster requires giving that point a label, which may be incorrect. To minimize the chance of this occurring, only points where label prediction was given with high confidence (greater than some $\alpha \in [0, 1]$ are included. The algorithm is given below.

Algorithm 2 Semi-Supervised Condensed k-Nearest-Neighbor

```
Let $L = \{(x_1, y_1, ..., (x_N, y_N)\}$ be a set of labeled examples. and $U = \{x_1, ..., x_M\}$ be a set of unlabeled examples.
Let $A$ be a copy of $L$.
for $x_i \in U$ do
    Let $A_k$ be the set of at most $k$ points $(x, y) \in A$ with the smallest values of $||x - x_i||_2^2$
    if $\max_y c_{A_k}(x, y) > \alpha$ then
        $A = A \cup \{(x_i, \arg\max_y c_{A_k}(x, y)\}$
    end if
end for
Let $S = \emptyset$.
for $(x_i, y_i) \in A$ do
    Let $S_k$ be the set of at most $k$ points $(x, y) \in S$ with the smallest values of $||x - x_i||_2^2$.
    if $\arg\max_y c_{S_k}(x, y) \neq y_i$ then
        $S = S \cup \{(x_i, y_i\}$
    end if
end for
return $S$
```

Note that unlike in the work by Sgaards, we do not condense until after all unlabeled points have been added, as this improved performance. We believe this is because a larger initial set of labeled points provides more accurate confidence scores for unlabeled points.

3.3 Scaling SS-CKNN

Even the relatively small 20 Newsgroup dataset produced over 40k features in our bag of word representation, making computing nearest neighbors extremely expensive. To allow this algorithm to scale up to the task at hand, we employed feature hashing and the ball tree data structure.

Feature hashing is a technique that can be used to project data onto a lower dimension space. We used hash kernels, an approach described in [4], to reduce the dimensionality of our data. A hash function is applied to the individual words that appear in a document and result is used as an index to update a fixed length array. A second hash function that returns either 1 or -1 is factored into the update to keep the values in the array unbiased. With a sufficiently large array (although still potentially much smaller than the number of features) the reduction will largely preserve the Euclidean distance between any two points allowing us to run KNN efficiently without a steep loss of accuracy.

To allow efficient classification of new points, we stored our condensed data points in a ball tree [5]. Ball trees consist of a tree where each node is associated with a ball in the feature space. Parent nodes are associated with balls that encompass its childrens balls. The tree can be searched with a depth-first branch-and-bound method to rapidly find neighbors of new points.

4 Results

4.1 The 20 Newsgroups Dataset

We evaluated our classifier on the 20 Newsgroups data set, which consists of about 20,000 newsgroup documents divided almost evenly into 20 different UseNet discussion newsgroups. The classification task is to determine which newsgroup a given article is posted in. Some of the newsgroups are closely related to each other (there are five about computers, for example).

We partitioned the data into four disjoint subsets: labeled training data (20%), unlabeled training data (40%), validation data (20%), and test data (20%). The test and validation sets were taken from articles that were written later than the train ones. This simulates the practical use for a document classifier, which would be trained on a set of articles and then asked to classify ones that were written later. The sets were created so each newsgroup contained the same number of documents.

4.2 EM and MNB Results

We varied the amount of labeled examples available to the classifier and recorded the classification accuracies of the MNB, SFE, and EM classifiers. In each case, the classifiers had access to all (roughly 7500) unlabeled examples. Both EM and SFE were run for 10 iterations, which we found to be sufficient for convergence. For smaller numbers of labeled training examples, we ran over multiple sets and averaged the results.

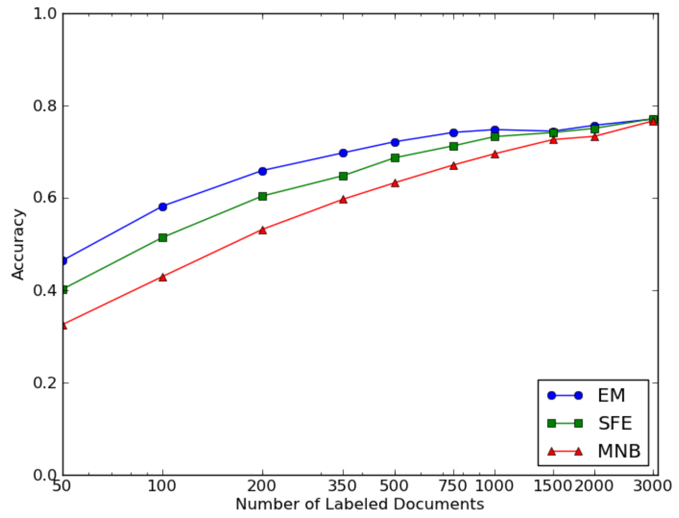

Figure 2: Classification accuracy on the 20 Newsgroups dataset

Using unlabeled data significantly boosted model performance, especially when the amount of training data was small. As the number of labeled examples increases, the gain from the additional data decreased, as the MNB could learn close to optimal parameters directly from the labeled data.

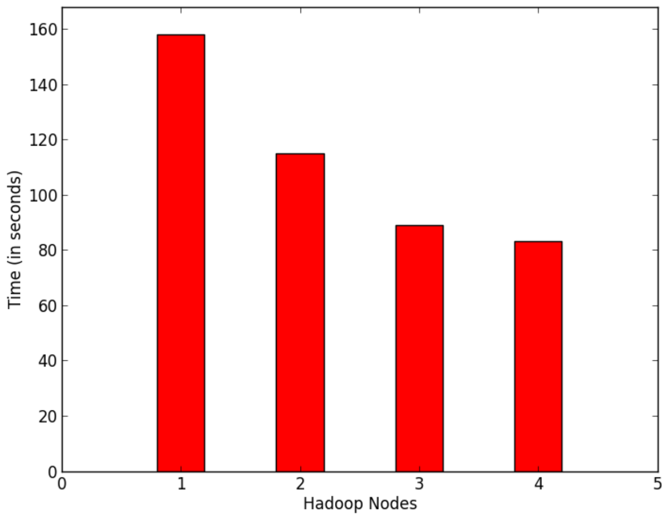

Figure 3: Runtime of a mapreduce iteration of the EM algorithm

Figure 4 shows how the runtime of completing an EM iteration using the mapreduce algorithm decreases as we add additional nodes to our cluster demonstrating the benefits of parallelization. We were able to significantly reduce runtime as we spread the computation over more nodes. We ran the the mapreduce job with over 200,000 unlabeled data points (we duplicated many of our data points to gain a more substantial dataset to test with). Three nodes could complete the task in under a minute and a half.

4.3 SS-CKNN Results

As in the EM and MNB experiments, we varied the amount of labeled data and recorded the classification accuracies of SS-CKNN and standard KNN algorithms. We empirically chose $k = 5$ and $\alpha = 0.9$ based on performance on the validation set. We also employed kernel hashing with 10000 buckets to obtain a more compact feature representation.

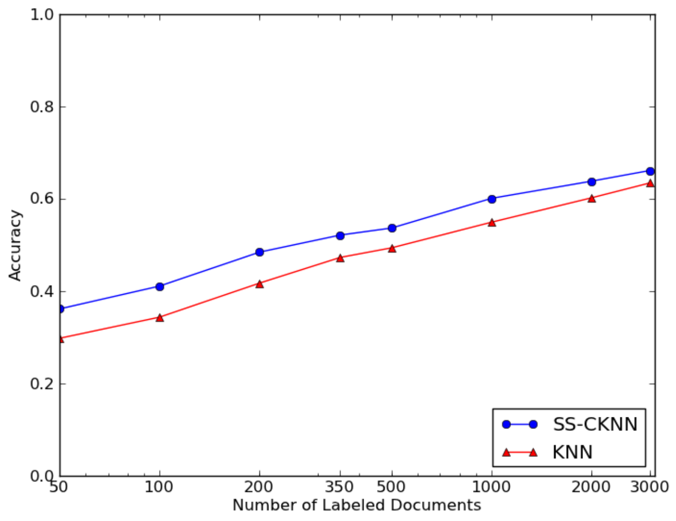

Figure 4: Classification accuracy on the 20 Newsgroups dataset

Making use of unlabeled examples improved classification accuracy on the 20 Newsgroup dataset, although to a smaller degree than with EM and Naive Bayes. We also evaluated kernel hashing by running SS-CKNN with different numbers of hash buckets and recording the accuracy and runtime. In all cases the algorithms used 200 labeled documents and all of the unlabeled ones.

Hash buckets	KNN accuracy	SS-CKNN accuracy	SS-CKNN Runtime
1000	27.18	30.74	0.76
2000	33.26	37.22	1.88
5000	38.73	44.59	5.85
10000	41.67	47.91	12.19
20000	43.93	51.48	27.31

As expected, the runtime is almost exactly linear in the number of dimensions of the data points, since computing Euclidean distances is linear in the number of dimensions. The accuracy decreased as the number of dimensions decreased, as kernel hashing does an imperfect job of preserving distances.

4.4 Comparison

The MNB + EM algorithm substantially outperformed the KNN algorithm

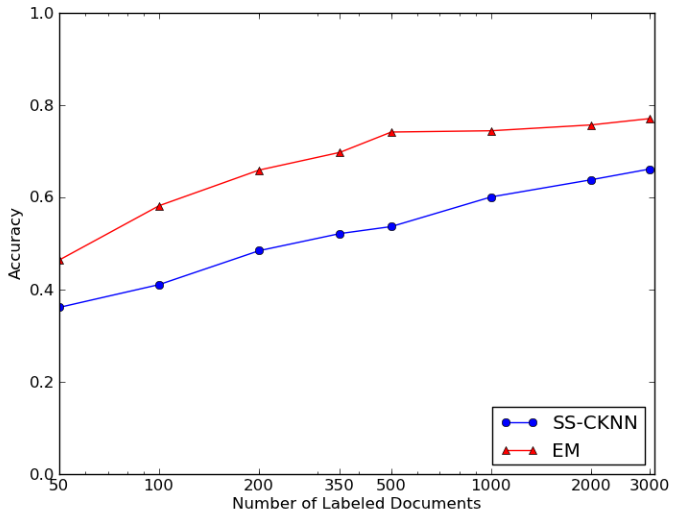

Figure 5: Classification accuracy on the 20 Newsgroups dataset

5 Conclusion

Our results show that using unlabeled data can lead to substantial accuracy increases in two standard text classification algorithms. For MNB, the EM algorithm allowed us to use unlabeled data to better learn the model parameters when we have little labeled data. We also found that integrating select unlabeled data points in a KNN model boosted its accuracy. Unlike with MNB, we found that the benefit of using unsupervised data with KNN decreased slowly as more labeled data was used. However, EM+MNB outperformed SS-CKNN on the 20 Newsgroups dataset.

References

[1] Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998) Learning to classify text from labeled and unlabeled documents. In *Proceedings of the National Conference on Artificial Intelligence*, pp. 792-799. JOHN WILEY & SONS LTD.

[2] Su, J., Sayyad-Shirabad, J., & Matwin, S. (2011) Large Scale Text Classification using Semi-supervised Multinomial Nave Bayes. ICML.

[3] Sgaard, A. (2011, June). Semi-supervised condensed nearest neighbor for part-of-speech tagging. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (pp. 48-52).

[4] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009, June). Feature hashing for large scale multitask learning. In Proceedings of the 26th Annual International Conference on Machine Learning (pp. 1113-1120). ACM.

[5] Omohundro, S. M. (1989). *Five balltree construction algorithms*. International Computer Science Institute.