Sparse Representations for Text Categorization

Tara N. Sainath[1]*, Sameer Maskey*[1]*, Dimitri Kanevsky*[1]
Bhuvana Ramabhadran[1]*, David Nahamoo*[1]*, Julia Hirschberg*[2]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, U.S.A
[2]Department of Computer Science, Columbia University, New York, NY 10027, U.S.A
{tnsainath, smaskey, kanevsky, bhuvana, nahamoo}@ibm.us.com[1], julia@cs.columbia.edu[2]

Abstract

Sparse representations (SRs) are often used to characterize a test signal using few support training examples, and allow the number of supports to be adapted to the specific signal being categorized. Given the good performance of SRs compared to other classifiers for both image classification and phonetic classification, in this paper, we extended the use of SRs for text classification, a method which has thus far not been explored for this domain. Specifically, we demonstrate how sparse representations can be used for text classification and how their performance varies with the vocabulary size of the documents. In addition, we also show that this method offers promising results over the Naive Bayes (NB) classifier, a standard baseline classifier used for text categorization, thus introducing an alternative class of methods for text categorization.

1. Introduction

Text classification, with a spectrum of applications in Natural Language Processing, ranging from document categorization to information retrieval, is the problem of classifying text documents into topics or pre-defined sets of classes. There has been a tremendous amount of work done on text categorization, including techniques based on decision trees, neural networks, nearest neighbor methods, Rocchios method, support vector machines, linear least squares, naive Bayes, rule-based methods and more. Some of these methods are unsupervised (no labeled documents) [5] while most of the methods assume a set of document or topic labels [1], [4], [7].

In [7], McCallum and Nigam showed that even a simple supervised classifier can produce decent classification accuracy. They showed that text could be categorized by assuming conditional independence between words given the labels and building a Naive Bayes (NB) Classifier. The test document can be classified simply by computing the likelihood of the class label given the words in the document based on Bayes theorem. Although such a trivial method produced promising results, text categorization was further improved by [4], which presented an SVM based classifier to categorize documents. Joachims showed that using a more sophisticated algorithm via SVMs can classify documents better than Naive Bayes.

Besides the improvement in the types of classifiers there has been significant work in feature selection for text categorization. Some of these feature selection methods are based on information gain, odd ratio, F-measure and Chi-Square testing (i.e., [1], [3], [10]). Although the type of feature selection algorithm may vary all the authors [1], [3], [10] agree that feature selection is crucial and improves the performance of text categorizer.

We observe from the aforementioned work that the type of statistical algorithm to model the data and the type of feature selection algorithm used, both play important roles in the performance of text categorizer. However, since the goal of this work is to introduce a novel technique using SRs for text classification, we do not focus on any feature selection techniques but instead use the identity of the word for features. Thus in this paper, we propose a popular technique from signal processing, namely sparse representations (SRs), for text classification. Our motivation for using SRs for classification is twofold. First, SRs have been successfully applied to face recognition [9] and phonetic classification [8], where they have been shown to offer improvements over other well-known classification schemes, including GMMs, SVMs and kNNs. Second, SRs adaptively select the relevant support data points from the training data, allowing us to classify the text document using a few relevant examples from the training document set. To the best of our knowledge, no work on using sparse representation methods, such Approximate Bayesian Compressive Sensing, Elastic Nets or Lasso, have been published in the literature (i.e. [4], [7]). This is the first piece of work that explores the use of these SRs for text classification.

Let us formally describe our SR classifier for text classification. Mathematically speaking, in a typical SR formulation, a dictionary H is constructed using individual examples of training documents, that is $H = [h_1; h_2 \ldots; h_n]$, where each $h_i \in Re^m$ is a feature vector for a specific training document. H is an over-complete dictionary such that the number of examples n is much greater than the dimension of each h_i (i.e. $m << N$). To reconstruct a signal y from H, SR solves the equation $y = H\beta$. A sparseness condition is enforced on β, such that it selects a small number of examples from H to describe y. A classification decision can be made by looking at the values of β coefficients for each training document in H.

This paper makes the following contributions towards exploring SRs for text classification. First, we explore seeding the dictionary H using all training documents. This is in contrast to [8], where only a subset of training examples was selected to seed H. We will demonstrate that using all training documents not only makes the size of H much larger, but also enforces a stronger need for sparseness on β. Our second contribution compares three different frameworks of using the SR solution of β to generate an actual classification decision. Third, we explore the sensitivity of the SR method to the vocabulary size of the training documents, which we denote by m. Since SR methods assume that H is an over-complete dictionary where m ¡¡ N, we investigate how the accuracy of the SR technique changes with varied vocabulary size. All experiments are conducted on the 20 Newsgroup Corpus [7]. We will show that our SR method offers promising results over the NB classifier, thus

introducing an alternative class of methods that work as well as the standard NB approach often used for text categorization.

The rest of this paper is as follows. Section 2 describes the theory behind SRs for classification and how it can be adapted for text classification. Section 3 presents the implementation of the traditional Naive Bayes classifier which serves as our baseline. Section 4 describes our experiments while Section 5 provides an analysis when using sparse representations. In particular, we address the variations in performance for different vocabulary sizes and the sparsity structure learned. Finally, Section 6 summarizes this work where we show that sparse representation based techniques are very promising for text classification and presents directions for future research.

2. Classification via Sparse Representations

In this section we discus how sparse representations can be used for classification.

2.1. Classification Based on Sparse Representations

The goal of classification is to use training data from k different classes to determine the best class to assign to a test document vector y. First, let us consider taking all training examples n_i from class i and concatenating them into a matrix H_i as columns, in other words $H_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n_i}] \in \Re^{m \times n_i}$, where $x \in \Re^m$ represents the feature vector of a document from the training set belonging to class i with dimension m. For example we can think of each x as as term-frequency feature, where the dimension m corresponds to the size of the vocabulary. Given sufficient training examples from class i, [9] shows that a test sample y from the same class can be represented as a linear combination of the entries in H_i weighted by β, that is:

$$y = \beta_{i,1} x_{i,1} + \beta_{i,2} x_{i,2} + \ldots + \beta_{i,n_i} x_{i,n_i} \tag{1}$$

However, since the class membership of y is unknown, we define a matrix H to include training examples from all k classes in the training set, in other words the columns of H are defined as $H = [H_1, H_2, \ldots, H_k] = [x_{1,1}, x_{1,2}, \ldots, x_{1,n_1}, \ldots x_{k,1}, x_{k,2} \ldots x_{k,n_k}] \in \Re^{m \times N}$ where n_k is the total number of feature vectors in class k. Here m is the dimension of each feature vector x and N is the total number of all training examples from all classes. H can be thought of as an over-complete dictionary where $m << N$. We can then write a test document y as a linear combination of all training examples, in other words $y = H\beta$. Ideally the optimal β should be sparse, and only be non-zero for the elements in H will belong to the same class as y. Thus ideally y will assign itself to lie in the linear span of examples from the training set of the true class it belongs to.

In this work, we solve the problem $y = H\beta$ subject to a sparseness constraint on β. As [9] discusses, the sparseness constraint on β acts as a regularization term to prevent overfitting and reduce sensitivity to outliers, and often allows for better classification performance than without sparseness. This is particularly important when $m << N$, which we will demonstrate in Section 5.1. Various sparse representation methods can be used to solve the above problem. In this paper, we solve for β using the Approximate Bayesian Compressive Sensing (ABCS) method [8], which imposes a combination of an l_1 and l_2 norm on β.

2.2. Classification Rule

Now that we have described our method to solve for β, we now discuss how to assign y as belonging to a specific class. We explore various classification rules, which we highlight below.

2.2.1. Maximum Support

Ideally, all nonzero entries of β should correspond to the entries in H with the same class as y. In this ideal case, y will assign itself to one training example from H, and we can assign y to the class which has the largest support in β.

$$i^* = \max_i(\beta) \tag{2}$$

2.2.2. Maximum l_2 Support

However, due to noise and modeling error, β belonging to other classes could potentially be non-zero. Therefore, we compute the l_2 norm for all β entries within a specific class, and choose the class with the largest l_2 norm support. More specifically, let us define a selector $\delta_i(\beta) \in \Re^N$ as a vector whose entries are non-zero except for entries in β corresponding to class i. We then compute the l_2 norm for β for class i as $\| \delta_i(\beta) \|_2$. The best class for y will be the class in β with the largest l_2 norm. Mathematically, the best class i^* is defined as

$$i^* = \max_i \| \delta_i(\beta) \|_2 \tag{3}$$

2.2.3. Residual Error

As [9] discusses, a classification decision can also be formulated by measuring how well y assigns itself to different classes in H. This can be thought of as looking at the residual error between y and the $H\beta$ entries corresponding to a specific class [9]. Let us define a selector $\delta_i(\beta) \in \Re^N$ as a vector whose entries are non-zero except for entries in β corresponding to class i. We then compute the residual error for class i as $\| y - H\delta_i(\beta) \|_2$. The best class for y will be the class with the smallest residual error. Mathematically, the best class i^* is defined as

$$i^* = \min_i \| y - H\delta_i(\beta) \|_2 \tag{4}$$

3. Naive Bayes

In contrast to the exemplar-based sparse representation method for classification, in this section we review the naive Bayes (NB) method for classification.

3.1. Formulation

Given a test document vector y, the a-posterior probability for class C_i given y is defined as follows:

$$p(C_i|y) = \frac{p(C_i)p(y|C_i)}{p(y)} \tag{5}$$

Within the NB framework, the best class is defined as the one which maximizes the posterior probability. In other words

$$i^* = \max_i p(C_i|y) \tag{6}$$

Below we describe how the terms $p(C_i)$ and $p(y|C_i)$ are estimated.

3.2. Probability Estimates

$p(C_i)$ is the prior probability of class C_i. This term is computed on the training set by counting the number of occurrences of each class. In other words if N is the total number of documents in training and N_i is the number of documents from class i, then $P(C_i) = \frac{N_i}{N}$.

To compute the term $p(y|C_i)$, we assume that document y is comprised of the following words $y = \{w_1, w_2, \ldots, w_n\}$, where n is the number of words. A "naive" conditional independence assumption is made on the term $p(y|C_i) = p(w_1, \ldots, w_n|C_i)$ and it is expressed as

$$p(w_1, \ldots w_n|C_i) = \prod_{j=1}^{n} P(w_j|C_i) \tag{7}$$

Each term $P(w_j|C_i)$ is computed by counting the number of times word w_j appears in the training documents from class C_i. In order to avoid non-zero probabilities if word w_j is not found in class C_i, we perform add-one smoothing. Thus if we define N_{ij} as the number of times word w_j is found in class C_i, we define $P(w_j|C_i)$ as follows, where m is the size of the vocabulary.

$$P(w_j|C_i) = \frac{N_{ij} + 1}{\sum_i N_{ij} + m} \tag{8}$$

We can see from the above equations that instead of making a classification decision on a test document using information about individual examples in training, the NB method pools all information about training data to estimate probability models for $P(C_i)$ and $P(w_j|C_i)$.

Given a test document y, we weight each probability $P(w_j|C_i)$ by the term-frequency (TF) occurrence of word w_j in this test document.

4. Experiments and Discussion

We performed our experiments on text categorization using 20 Newsgroup corpus [7], which has been widely used for evaluating text categorization algorithms. The corpus consists of approximately 18,000 newsgroup documents that are divided into 20 different classes. 40% of the documents are separated as a held-out test set (roughly 7, 532) while remaining 60% (rough 11,314) are used for training the models. These text documents are quite noisy with headers containing email addresses and the text body containing links, addresses.

Various kinds of features have been explored for text categorization ([10], [3], [1]) with Term Frequency (TF) being one of them. TF feature can provide important information about the word distribution relevant for the class label of the document. For example, documents that are labeled as 'Hockey' will contain words such as 'hockey' and 'puck' in higher frequency than words that are related to other sports. Here TF is defined by Equation 9 where n_i^d is the number of times the i_{th} term occurred in document d, and D is the total number of documents.

$$TF_i = \sum_{d=1}^{D} n_i^d \tag{9}$$

Typically TF features are often weighted by Inverse Document Frequency (IDF) features, which provide even better discriminating properties among different classes of documents. However, [6] shows that for the 20 Newsgroup Corpus the TF.IDF is not necessarily better or worse than TF. Thus, after extracting only TF features for all the words in a document, we represent each document with a TF vector of length $|m|$ that is equal to the vocabulary size of the whole corpus, which is $55,710$.

Since our SR method requires that the number of documents in H (11,314), be less than the dimension of each TF feature vector (i.e. $|V|$), we explore pruning the vocabulary size of TF vectors. This pruning is accomplished by removing words from the vocabulary if the total number of occurrences in training is less than a certain frequency threshold. We analyze the NB and SR classifiers varying the vocabulary size of the TF vectors from 1,000 to 10,000 in increments of 1,000.

5. Results

5.1. Sparsity Analysis

We first explore the behavior of the the β coefficients obtained by solving $y = H\beta$ using ABCS [8]. Figure 1 shows the β coefficients for a randomly sampled test document y. The $1,400$ β coefficients, corresponding to $1,400$ training document in H, were obtained by picking every 8^{th} β coefficient from the full set of $11,314$. The figure illustrates that the β entries are quite sparse, suggesting that the SR technique is using only a few samples in H are used to characterize y. For example, roughly only 1% of the absolute value of the β coefficients are 0.035. As [9] discusses, this sparsity can be thought of as a form of discrimination, as certain documents are selected as "good" in H while jointly assigning zero weights "bad" documents in H.

Figure 1: Plot of β Coefficients for 1,400 Training Documents

We can further analyze the effect of sparseness by looking at the classification accuracy when we enforce a sparseness on β versus no sparseness. Table 1 compares the results for the two approaches for a vocabulary size of $6,215$. Note that we use the classification metric defined by Equation 3. The table illustrates that enforcing sparseness, and thus utilizing only a small fraction of examples, provides a small improvement in accuracy.

Method	Accuracy
No Sparseness Constraint	78.6
Sparseness Constraint	**78.8**

Table 1: Classification Accuracy with and Without Sparseness Constraint on β

5.2. Classification Metrics

Second, we explore the accuracy with different sparse representation classification metrics. Table 2 lists these accuracies for a vocabulary size of $6,215$, and also shows the NB accuracy as a comparison. First, notice that using the Maximum Support as a metric is too hard of a decision, as β from other classes is often non-zero. Therefore, making a softer decision by using the l_2 norm of β offers higher accuracy. In addition, notice that using the residual error offers the lowest accuracy. Because the features are so sparse, the residual value of $\| y - H\delta_i(\beta) \|_2$ when $\delta_i(\beta) \approx 0$ will reduce to $\| y \|_2$ which is a very small number and might not offer good distinguishability from class residuals in which $\delta_i(\beta)$ is high. Thus, in the rest of the experiments we use the l_2 norm of β to make classification decisions.

Classification Decision	Accuracy
NB	77.9
Maximum Support	77.2
Maximum l_2 Support	**78.8**
Minimum Residual Error	55.5

Table 2: Classification Accuracy for Different Sparse Representation Decisions

5.3. Classification Across Varied Vocabulary Size

Finally, we explore the behavior of the sparse representation and NB methods as the vocabulary size is varied from 1,000 to 10,000. Figure 2 shows the accuracies of the two classifiers for varied vocabulary size. We observe that the accuracy of both classifiers increases as the the vocabulary size increases, similar to the results reported in [7]. The larger vocabulary size allow more features to be used in discriminating the documents across different classes. Between a vocabulary size of 1,000 and 8,000, the sparse representation method offers between a $0.3 - 0.8\%$ absolute improvement compared to NB, and a McNemar's significance test [2] also confirms that this difference is statistically significant.

When the vocabulary size increases beyond 8,000, the accuracy of the SR method drops and approaches that of the NB method. This can be attributed to the behavior of the SR technique, which requires that N, the number of training documents, is much less than m, the vocabulary size. As m approaches N, the SR method has less degrees of freedom to choose β, as there are more systems of equations to solve when computing $y = H\beta$. Thus, sparsity becomes less beneficial and the SR accuracy approaches that of a non-sparse solution, which we saw from Table 1 was slightly lower than the SR accuracy when sparsity can be applied.

We did not compare the accuracy of the SR and NB methods above 10,000 since this would make the SR ineffective since $m > N$. However, we should note that [7] observed that the accuracy does not improve significantly when the vocabulary size is greater than 10,000.

Figure 2: Classification Accuracies for Varied Vocabulary Size

6. Conclusions and Future Work

We presented a technique to perform text classification using sparse representations. SRs have mostly been used in signal processing research and we believe we have presented a step towards using SRs in NLP research tasks such as text categorization. The results show that our SR method offers slight improvements over a standard Naive Bayes (NB) classifier across varying vocabulary sizes. Though our current SR method may not perform better than state of the art text classification techniques, we have shown with our preliminary results that we can effectively use SRs for text categorization. In the future, we would like to explore using SRs for text classification with better feature selection techniques, and also compare our results to SVMs, similar to as we did for phonetic classification in [8].

7. References

[1] A. Dasgupta, P. Drineas, B. Harb, V. Josifovski, and M. W. Mahoney, "Feature selection methods for text classification," in *Proceedings of KDD*, 2007, pp. 230–239.

[2] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. John Wiley and Sons, Inc., 2001.

[3] F. George, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, 2003.

[4] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of ECML*, 1998.

[5] Y. Ko and J. Seo, "Automatic text categorization by unsupervised learning," in *Proceedings of COLING*, 2000.

[6] M. Lan, C.-L. Tan, H.-B. Low, and S.-Y. Sung, "A Compreshensive Comparative Study on Term Weighting Scheme for Text Categorization with Support Vector Machines," in *Proceedings of WWW*, 2005.

[7] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Procedings of the AAAI-98 Workshop on Learning for Text Categorization*. AAAI Press, 1998, pp. 41–48.

[8] T. N. Sainath, A. Carmi, D. Kanevsky, and B. Ramabhadran, "Bayesian Compressive Sensing for Phonetic Classification," in *Procedings of ICASSP*, 2010.

[9] J. Wright, A. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 210–227, 2009.

[10] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of ACM SIGIR*, 1997.