Text Classification

Using SVM

Project Report

Machine Learning

To the

The University Of Texas at Dallas

Submitted by:

Akshay Hacholli

axh141530@utdallas.edu

The University of Texas at Dallas

Dallas – 75252

May, 2015

Abstract

This project uses Support Vector Machines (SVMs) for learning text classifiers from given corpus. It analyzes the particular properties of learning with text data and extracts features. SVMs achieve substantial improvements over the currently best performing methods and behave robustly over a variety of different learning tasks. Furthermore, they are fully automatic, eliminating the need for manual parameter tuning. Here we are interested in examining whether automatic classification of corpus can be improved by a pre-filtering the vocabulary to reduce the feature set used in the computations. First we compare artificial neural network and support vector machine algorithms for use as text classifiers of news items. Secondly, we identify a reduction in feature set that provides improved results.

Introduction

Corpus-based supervised learning is now a standard approach to achieve high-performance in natural language processing. However, the weakness of supervised learning approach is to need an annotated corpus, the size of which is reasonably large. Even if we have a good supervised-learning method, we cannot get high-performance without an annotated corpus. The problem is that corpus annotation is labor intensive and very expensive. In order to overcome this, several methods are proposed, including minimally-supervised learning methods (e.g., (Yarowsky, 1995; Blum and Mitchell, 1998)), and active learning methods (e.g., (Thompson et al., 1999; Sassano, 2002)). The spirit behind these methods is to utilize precious labeled examples maximally.

Background

a) **Data Set** - The data is part of the 20 newsgroup dataset – there are 20 classes, and the documents are from discussion forums. You can find the data here: http://www.hlt.utdallas.edu/~yangl/cs6375/project/.

The files are:

I. train - this directory contains all the training files. There are 20 directory corresponding to the 20 categories.
II. dev - You can use files in this directory to optimize/tune your system. You need to have a separate testing module that takes this directory and predicts the class label for each file in this dir.
III. class_name.txt - This is the mapping file, from the 20 category names to a class index of 0 to 19.
IV. dev_label.txt - This is the reference label for the files in dev dir. You can compare your prediction with this to get the performance of your system. Note here the reference classes use the IDs above.

b) **Support Vector Machines (SVM)** - SVM classification algorithms to solve two-class problems, are based on finding a separation between hyper planes defined by classes of data, shown in Figure 1. This means that the SVM algorithm can operate even in fairly large feature sets as the goal is to measure the margin of separation of the data rather than matches on features. The SVM is trained using pre-classified documents. Research has shown that SVM scales well and has good performance on large data sets.

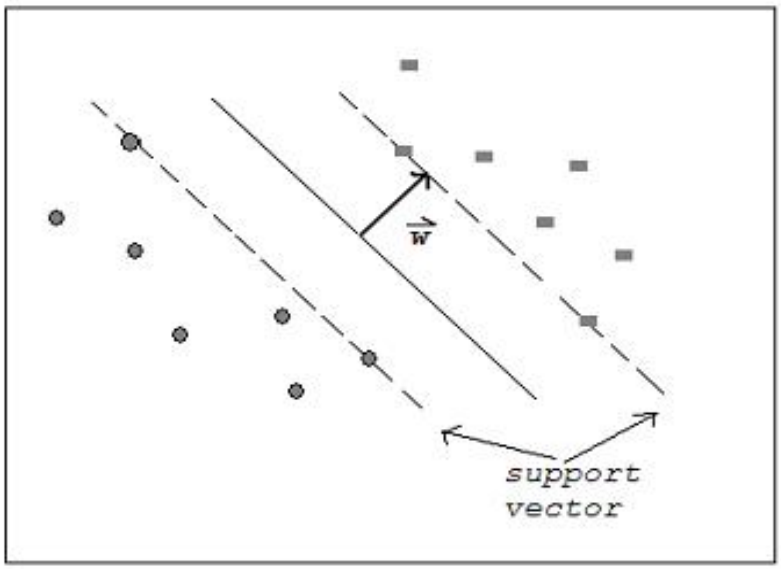


Figure 1. Example of SVM hyperplane pattern

Using the entire vocabulary as the feature set, we found that the SVM algorithm outperformed the Naïve Bayes algorithm used on these data sets; text related documents in 20 categories. Naïve Bayes classification algorithms are based on an assumption that the terms used in documents are independent. Both Bayes and SVM algorithms are linear, efficient, and scalable to large document sets. We used a reduced vocabulary as the feature set by first word stemming and then using a stop list of very frequent words and elimination of very infrequent words from the feature set. Using 11,314 documents from the corpus, we compared the performance of several algorithms including SVM, and Naïve Bayes. For both document sets, this test indicated that the SVM performed better and we also believe that the SVM algorithm performs most accurately in a test that compared the Naïve Bayes, Decision Trees, and SVM.

c) **Stemming** - Removing suffixes by automatic means is an operation which is especially useful in the field of information retrieval. In a typical IR environment, one has a collection of documents, each described by the words in the document title and possibly by words in the document abstract. Ignoring the issue of precisely where the words originate, we can say that a document is represented by a vector of words, or terms. Terms with a common stem will usually have similar meanings, for example:

CONNECT
CONNECTED
CONNECTING
CONNECTION
CONNECTIONS

Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT. In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

Why Should SVMs Work Well for Text Categorization?

To find out what methods are promising for learning text classifiers, we should find out more about the properties of text.

High dimensional input space: When learning text classifiers, one has to deal with very many (more than 10000) features. Since SVMs use over fitting protection, which does not necessarily depend on the number of features, they have the potential to handle these large feature spaces.

Few irrelevant features: One way to avoid these high dimensional input spaces is to assume that most of the features are irrelevant. Feature selection tries to determine these irrelevant features. Unfortunately, in text categorization there are only very few irrelevant features. All features are ranked according to their (binary) information gain. Then a naive Bayes classifier is trained using only those features ranked. The results show that even features ranked lowest still contain considerable information and are somewhat relevant. A classifier using only those "worst" features has a performance much better than random. Since it seems unlikely that all those features are completely redundant, this leads to the conjecture that a good classifier should combine many features (learn a "dense" concept) and that aggressive feature selection may result in a loss of information.

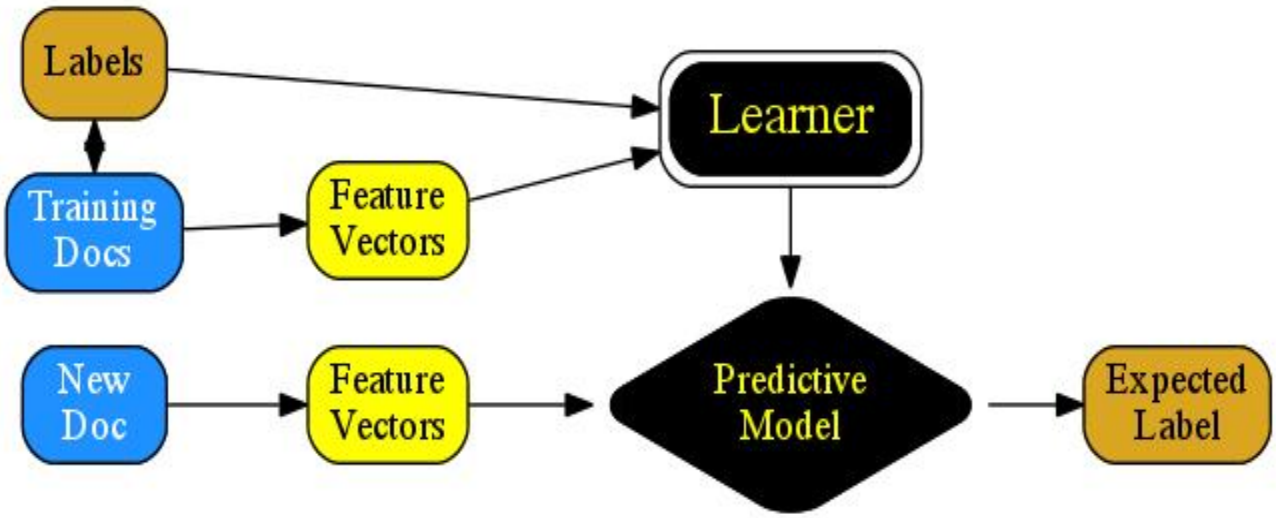
Document vectors are sparse: For each document, the corresponding document vector contains only few entries which are not zero. There are both theoretical and empirical evidence for the mistake bound model that "additive" algorithms, which have a similar inductive bias like SVMs, are well suited for problems with dense concepts and sparse instances.

Most text categorization problems are linearly separable: All assumed categories are linearly separable and so are many from the given corpus. The idea of SVMs is to find such linear (or polynomial, RBF, etc.) separators. These arguments give theoretical evidence that SVMs should perform well for text categorization.

These arguments give theoretical evidence that SVMs should perform well for text categorization.

Experiments

Following is the flow diagram of how system works



For this experiment we used the corpus as mentioned in the Dataset section, which contains 11,314 documents in plain text format and 20 predefined categories. As for the training purpose, the documents are available in their respective categories. For the SVM to be trained, we are using an online available tool LIBLINEAR, which requires the training file to be in a particular format. The following procedure is followed to convert the document into training file -

- **Tokenization** – Initially every document is read and broken into a stream of words, this is done by line by line reading of file and then removing out all the punctuations, numbers, special symbols etc. This forms the pre-processing step.
- **Stemming** - Just the alphabets were retained and then each word was sent to porter stemmer which returns the stemmed word, then a list of unique words is formed, which forms feature set.
- **Stopword Filtering** – This step is meant to omitting meaningless words (e.g. a, an, this, that), word stemming for reducing the number of distinct words, lowercase conversion etc. (The list of stopwords was taken from Wikipedia)
- **Transforming Text Documents into Vector Space** – A map of category and features is formed where in every feature is nothing but the stemmed word along with its frequency in the document. This is what the training file consists off.

LIBLINEAR Classification

LIBLINEAR supports two popular binary linear classifiers: LR and linear SVM. For multi-class problems, we implement the one-vs-the-rest strategy and a method by Crammer and Singer. In this project we are using L2-regularized L2-loss SVM type to train and classify documents. Also there are two parameters based on which the model classifies documents –

- C - cost of constraints violation
- Eps - stopping criteria

Initial experiments and observation

- **Stopwords** - Initially when stopwords were considered the training took lot of time and also accuracy was less.
- **Stemming** – Also when the words were not stemmed the number of features got increased which resulted in large training file and larger time to process.
- **SVM Parameters (Cross Validation)** – To find values of the parameters we performed cross validation. Cross-Validation set (20% of the original data set): This data set is used to compare the performances of the prediction algorithms that were created based on the training set. We choose the parameters that has the best performance.
- **Lemmatization** – Instead of stemming words, we tried lemmatizing word, but what we found was that processing time increased very high with a small gain in accuracy.

Results/Statistics

Following are the result/observation during the course of experiment. **The system was able to achieve 84.55% accuracy.** There are four stages through which program goes through

I. Build Training file
II. Training SVM
III. Build Test file
IV. Classifying test data

	Experiment	**Time to build training file**	**Time taken for training**	**Time to build testing file**	**Time taken to classify**
1	Just raw data (without stemming and filtering)	836.151 sec	99.542 sec	185.364 sec	8.684 sec
2	With filtering stopwords	704.965 sec	69.698 sec	158.168 sec	7.533 sec
3	Filtering and stemming	567.213 sec	49.327 sec	103.545 sec	5.989 sec
4	Filtering and Lemmatizing	991.639 sec	48.616 sec	225.465 sec	5.755 sec
5	Filtering, stemming and with proper SVM parameters	567.213 sec	35.123 sec	103.545 sec	4.985 sec

Conclusion

This report introduces support vector machines for text categorization. It provides both theoretical and empirical evidence that SVMs are very well suited for text categorization. The theoretical analysis concludes that SVMs acknowledge the particular properties of text: (a) high dimensional feature spaces, (b) few irrelevant features (dense concept vector), and (c) sparse instance vectors.

The experimental results show that SVMs consistently achieve good performance on text categorization tasks. With their ability to generalize well in high dimensional feature spaces, SVMs eliminate the need for feature selection, making the application of text categorization considerably easier. Another advantage of SVMs over the conventional methods is their robustness. SVMs show good performance in all experiments, avoiding catastrophic failure, as observed with the conventional methods on some tasks. Using Support Vector Machines as a machine learning algorithm is nowadays the most popular approach. Some aspects of SVMs are covered that reflect why they are suitable for Text Classification.

Acknowledgements

- I would like to thank LIBSVM and LIBLINEAR for assisting while working on the project. The java libraries produced by this team well document and coded.
- Dr. Yang Liu for giving an opportunity to come out with our own implementation of text classification and also providing training/testing corpus.

Reference

[1] Text Categorization and Support Vector Machines Istvan Pilaszy Department of Measurement and Information Systems Budapest University of Technology and Economics e-mail: pila@mit.bme.hu

[2] Text Categorization with Support Vector Machines: Learning with Many Relevant Features Thorsten Joachims Universit at Dortmund Informatik LS8, Baroper Str. 301 44221 Dortmund, Germany

[3] Support Vector Machines for Text Categorization A. Basu, C. Watters, and M. Shepherd Faculty of Computer Science Dalhousie University Halifax, Nova Scotia, Canada B3H 1W5 {basu | watters | shepherd@cs.dal.ca}

[4] A Practical Guide to Support Vector Classification Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin Department of Computer Science National Taiwan University, Taipei 106, Taiwan http://www.csie.ntu.edu.tw/~cjlin Initial version: 2003 Last updated: April 15, 2010

[5] LIBLINEAR -- A Library for Large Linear Classification - http://www.csie.ntu.edu.tw/~cjlin/liblinear/