Machine Learning Project
Classification of news articles

Jakob Sahlström
jasa5691@student.uu.se

Anders Lindström
anli6945@student.uu.se

Jesper Dürebrandt
jedu6357@student.uu.se

Uppsala University

Thursday 22nd May, 2014

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1 Introduction

2 Theory

2.1 Naive Bayes classifier

Let c be the class and $A = a_1, \ldots, a_n$ be the attributes of a document. Then with Bayes Theorem

$$p(c|A) = \frac{p(A|c)p(c)}{p(A)}, \tag{1}$$

the attributes A is classified as class C if and only if

$$f_b(A) = \frac{p(c|A)}{p(\neg c|A)} \geq 1, \tag{2}$$

where $f_b(A)$ is called a *Bayesian* classifier. Assuming all attributes are independent given the class,

$$p(A|c) = p(a_1, \ldots, a_n|c) = \prod_{i=1}^{n} p(a_i|c)$$

the final classifier can be written as

$$f_{nb}(A) = \frac{p(c)}{p(\neg c)} \prod_{i=1}^{n} \frac{p(a_i|c)}{p(a_i|\neg c)} \tag{3}$$

where f_{nb} is called the *Naive Bayesian* (NB) classifier.

Two models that uses the Naive Bayes assumption are the *multi-variate Bernoulli* model and the *multinomial* model. The main difference is that in the Bernoulli model the attributes are binary, indicating if a word from a vocabulary has occurred at least once or not. In the multinomial model the frequency of words are taken into account.

2.2 Random Forest

Random Forest (RF) is based on building several considerably small decision trees. Consider having a feature vector that is of length N, then randomly select $n << N$ of those features. Build the trees with some kind of algorithm (e.g. C4.5), where information gain is taken into consideration when splitting, and no pruning is done (i.e. expand the tree fully). Then repeat selecting n new variables from N until the wanted number of trees are built.

After all the trees are built, they can be used to let a new vector of data pass through all the trees and then letting each tree *vote* on what class the vector most probably should be a part

of.

Algorithm 1 Random Forest

Let $x = (x_1, x_2, ..., x_N)$ be a set of features;
while *not enough trees* **do**
Randomly pick with replacement a subset containing $n << N$ features;
Use training set to build a decision tree using a classification algorithm, e.g. C4.5, except no pruning is done.
end while

2.3 Support Vector Machine Classifier

Support vector machine classifiers (SVM's) classifies data belonging to two classes by finding the hyperplane with the widest margin that separates the classes. The data vectors that restrict the margin of the hyperplane are referred to as suport vectors. This results in a maximization problem, where the objective function describes the width of the margin. This is solved using quadratic programming. An advantage with this approach is that the maximization problem is convex, meaning that the maximum found is guaranteed to be the global maximum. This requires, however, that the classes are linearly separable.

3 Method

3.1 News Crawler

A crawler was written and used that extracted articles from BBC[1]. The crawler extracts which topic the article belongs to, which is written in the HTML-code of the page on BBC's articles. In total, about `--NUMBER--` articles where extracted under `--NUMBER--` different topics. This will be used as training and test data for the classifiers.

3.2 Coding

Pythons has been chosen as programming language to extend our knowledge and experience of implementing different types of classifiers with different classify-packages and further understand how the data has to be prepared and presented to the classifiers.

[1] `http://www.bbc.com/`

3.3 Preparation of Data

To classify the articles as accurately and fast as possible, the data has to be prepared in such way that it contains as much information as possible in a format as dense as possible. To achieve this, the articles were parsed into words with white-space as separator, then removed of all characters not found in the English alphabet, though keeping the '-' character. After that, the words were stemmed so that similar words would be on the same form, and not be seen as two different words. (E.g. "argued", "argues", "arguing" reduce to "argu".) When they had been stemmed, all so called stop words were removed (such as "a", "the", etc.)

When the crawler was extracting articles, the crawler sometimes for various reasons failed to obtain the body text for some of the articles. The result was an empty field where all the body text would go. This was of course removed from the set of articles.

3.4 Datatypes

The different datatypes were

Binary:
Does article contain word or not.

Count:
How many times does article contain word.

Normalized Count by article length:
The Count array divided by the number of words in article (sum < 1).

Normalized Count by sum of Count:
The Count array divided by the sum of the Count array (sum $= 1$).

3.5 Classifiers

In the Scikit-learn package[2] there are plenty of different classifiers, amongst them the four we want to investigate. The four classifiers we will be investigating are

- Naïve Bayes: Bernoulli
- Naïve Bayes: Multinomial
- Support Vector machines (SVM)
- Random forest

[2] `http://scikit-learn.org/`

4 Results

4.1 Naive Bayes Classifier

4.2 Support Vector Machine Classifier

5 Related work

5.1 Naive Bayes Classifier

Naive Bayes models are widely used because of it's simplicity and efficiency. A. McCallum and K. Nigam compares the two most common models, the multi-variate Bernoulli and the multinomial model, in the realm of document classification. They are explained in detailed both theoretically and empirically and in general the multinomial model outperforms the Bernoulli model [7].

5.2 Random Forest Classifier

Random Forest is a decision tree based classification model. It has been used to classify web documents by keywords, where it for five and seven topics performed better than e.g. Naive Bayes and MLP [2]. In I. Kopriska, J. Poon, J. Clark, J. Chan's paper, they use Random Forest for classifying e-mails. Random Forest was able to out perform other methods, such as DT, SVM and NB[3].

5.3 Support Vector Machine Classifier

Support vector machine classifiers performs well on data that is linearly separable and is guaranteed to find the optimal hyperplane that separates the data. They can however only separate data into two classes, but if combined they are able to perform multi-class classification. A simple approach is to use k SVMs to solve a k-class classification problem. The SVMs may also be combined in a more sophisticated fashion, so that less than k SVMs can be used. Both methods are investigated in [5].

6 Conclusions & Future work

References

[1] R. Berwick. An Idiots guide to Support vector machines, 2003.

[2] Myungsook Klassen and Nikhila Paturi. Web document classification by keywords using random forests, 2010.

[3] Irena Koprinska, Josiah Poon, James Clark, and Jason Chan. Learning to classify e-mail, 2006.

[4] V. Tampakas M. Ikonomakis, S. Kotsiantis. Text Classification Using Machine Learning Techniques, 2005.

[5] Eddy Mayoraz and Ethem Alpaydin. Support Vector Machines for Multi-class Classification, 1999.

[6] Andrew McCallum and Kamal Nigam. A comparison of event models for Naive Bayes text classification, 1998.

[7] Harry Zhang. The Optimality of Naive Bayes, 2004.