NLP Assignment 1 Mini-report

SVM-based Sentiment Detection of Reviews

Zébulon Goriely, Queens', zg258

Wednesday 30th October, 2019

Word Count: 557[1]

1 Introduction

In a 2002 paper[2], Pang et al. considered the problem of classifying reviews according to positive or negative sentiment. In this report, we reimplement two of the machine learning systems described in paper; the Naive Bayes (NB) classifier (with smoothing) and the Support Vector Machine (SVM) classifier.

These methods are applied to a dataset of movie reviews which were given to us in the framework of a course in NLP.

2 Background

Both NB and SVM use a bag-of-features framework to represent the documents. In this report we examine three of the seven features used in (Pang et al, 2002):

- unigrams (single words such as "enjoy"),
- bigrams (pairs of words such as "didn't hate"),
- unigrams and bigrams.

Each document d is then represented as a document vector $\vec{d} := (n_1(d), n_2(d), \ldots, n_m(d))$ where $n_i(d)$ is the number of times f_i appears in document d.

2.1 Naive Bayes

The first classifier derived in the paper is referred to as the *Naive Bayes* (NB) classifier. A given document d is assigned to the class $c^* = \arg\max_c P(c|d)$. By decomposing Bayes' rule:

$$P_{\text{NB}}(c|d) := \frac{P(c)(\prod_{i=1}^{m} P(f_i|c)^{n_i(d)})}{P(d)}$$

It is 'Naive' because we assume that the features f_i are conditionally independent in order to calculate $P(d|c)$ needed to apply Bayes' rule.

As in the paper, we use relative-frequency estimation using add-one smoothing for the training of our NB classifier.

2.2 Support Vector Machine

The third classifier derived in the paper is the *Support Vector Machine* classifier. Letting $c_j \in 1, -1$ (corresponding to positive and negative) be the correct class of document d_j, the procedure finds the hyperplane represented by vector $\vec{w}$ which separates document vectors into the two classes as widely as possible.

Once trained, classification proceeds simply by determining which side of the hyperplane the d_i falls on for each test instance. As in the paper, we use Joachim's (1999) SVM^{light} package [3], using default parameters.

[1] *texcount docs/assignment1/report.tex*, using just the text in each section

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of EMNLP

[3] http://svmlight.joachims.org

	Features	frequency or presence?	NB	SVM
(1)	unigrams	freq.	**81.6**	73.1
(2)	unigrams	pres.	82.9	**85.8**
(3)	bigrams	pres.	**83.7**	83.4
(4)	unigrams+bigrams	pres.	84.7	**87.1**

Table 1: Average ten-fold cross-validation accuracies, in percent. Dataset: 2000 stemmed movie reviews. Boldface: best performance for a given setting (row).

System A	System B	p value
NB,freq,unigrams	SVM,freq,unigrams	7.59×10^{-7}
NB,freq,unigrams	NB,pres,unigrams	0.53
NB,freq,unigrams	NB,pres,bigrams	0.32
NB,freq,unigrams	NB,pres,unigrams+bigrams	0.138
SVM,freq,unigrams	SVM,pres,unigrams	2.97×10^{-9}
SVM,freq,unigrams	SVM,pres,bigrams	9.57×10^{-7}
SVM,freq,unigrams	SVM,pres,unigrams+bigrams	6.24×10^{-7}

Table 2: The p-values from two-tailed sign-tests with a significance level of $k = 0.01$.

2.3 Feature frequency against feature presence

During evaluation of these two systems, Pang et al. also experimented with binarising the document vectors, setting $n_i(d)$ to 1 iff feature f_i appears in d and running NB and SVM^{light} on these new vectors[4]. We have also reproduced this as shown in Table 1.

3 Method

We reimplemented NB and SVM using Python. Before running these classifiers, we used an implementation of the Porter Stemming Algorithm[5] to stem all of the movie review files.

When training the models on 90% of the data, we get 33000 unigram features and 383000 bigram features[6]. For the sake of not interfering with the algorithm as much as possible, we have not cut off any of the bigrams.

4 Results

We ran 8 different classifiers as detailed in Table 1, using different combinations of features and using frequency or presence-based document vectors. For each classifier, we calculated the accuracy using ten-fold cross validation[7].

Both of these machine learning algorithms perform significantly[8] better than the random-choice baseline of 50%.

As can be seen by line (1) of Table 1, NB performs significantly better than SVM^{light} when trained on unigrams with frequency-based document vectors with an average accuracy of 81.6% compared to 73.1% using SVM^{light}.

As can be seen by lines (2), (3) and (4) of Table 1, much better performance is achieved for

[4]Note: Pang et al. used arbitrary language here, it may be worth referring to the Multinomial Naive Bayes and the Bernoulli Naive Bayes which are related to this notion of feature frequency and presence.

[5]The Stemmer used was the PorterStemmer from the nltk.tokenize package.

[6]These values are rounded to the nearest 1000 as they vary between folds

[7]The folds were chosen using a deterministic round-robin algorithm.

[8]The test for significance was a two-tailed sign-test with a significance level of $k = 0.01$.

SVMs by accounting only for feature presence, not feature frequency, giving average accuracies of 85.8%, 83.4% and 87.1% respectively compared to the 73.1% (feature frequency).

The best result achieved by all settings was the 87.1% when running SVM^{light} on presence-based document vectors composed of unigrams and bigrams.

Conclusion

Through this report we have confirmed the findings of (Pang et al, 2002) to find that the machine learning methods perform better than random-choice, that Naive Bayes performs better than SVMs when using frequency-based unigram features and that significant gains in performances can be had for SVMs when switching to a presence-based document vector with more features to choose from.