

CS3244 Machine Learning

Assignment 2

Shawn Tan (U096883L)

1 Introduction

Text classification is a common problem in the field of machine learning and Natural Language Processing (NLP). In this assignment, we were tasked to classify some posts on several newsgroups.

We were given stemmed texts from 5 newsgroups: `comp.graphics`, `comp.os.ms-windows.misc`, `comp.sys.ibm.pc.hardware`, `comp.sys.mac.hardware` and `comp.windows.x`. In our test set, we were given 1425 instances to classify, and 2935 training instances. Several scripts and programs were supplied to perform various tasks:

fs.php and fe.php These two PHP scripts help to extract the features from the texts using TF-IDF and χ^2 feature selection methods.

Formatting.exe This program converts the `.txt` files created by the PHP scripts into `.arff` files which can be read by WEKA.

The end result are two `.arff` files that consist of features that correspond to normalised word frequencies. These are the feature vectors which the various classifiers used will be working with. In this report, we experiment with using 3 different types of classification algorithms: k -Nearest Neighbour, Naive Bayes, and SVMs.

Our approach involves training different classifiers using each of the algorithms using the same dataset. Eventually, we take the best performing classifier from each different algorithm, and use these classifiers together to hopefully reduce any kind of overfitting caused by any of the individual algorithms. After evaluating this classifier, we then use this to classify our test set.

We make use of version 3.7.4 of WEKA for the tasks detailed in this report.

2 Selecting the Features

Setting an overly high value for feature selection may result in feature vectors that are too specific to the training set, and eventually cause overfitting. For our first experiment, we select only the top 50 keywords for each class for our feature vector. This resulted in 203 keywords in total.

Using the selected features, we extract the feature vectors from each of the newsgroup posts. Using this, we train three classifiers (k NN, Naive Bayes, SVM) using the default WEKA settings, and evaluate their performances before proceeding. We do this several times, with

<code>fs_top_num</code>	Keywords/Features	Naive Bayes	SVM	IBk
50	203	0.738	0.77	0.732
100	428	0.74	0.801	0.758
150	641	0.743	0.814	0.767
200	857	0.74	0.822	0.774

Table 1: Experiments with the number of features used.

several different values of `fs_top_num`. Table 2 reports the different values we tried, and the weighted F-measure of the corresponding classifiers.

Increasing `fs_top_num` by 50 at each round of testing, we performed the experiment four times. We decided to use an `fs_top_num` value of 200 for our classification task, as larger feature vectors may cause classification to take long periods of time, making repeated testing difficult.

3 Tuning Performance of Individual Classifiers

In the following section we attempt to tune the performance of individual classifiers by adjusting the parameters for the different algorithms. For each algorithm, we evaluate the classifiers based on their performance on the training set and see how the classifiers can be improved.

3.1 Naive Bayes (NaiveBayes)

The WEKA implementation of the Naive Bayes classifier does not allow for much tweaking of parameters. The default (using WEKA’s settings) classifier has an F-measure of 0.74 on the training set. It is worth noting that the F-measure on class 4 was 0.809 and has a true positive rate of 0.81 for class 3, as this may be useful in our attempt to combine the classifiers later on.

3.2 Support Vector Machines (SMO)

By default, WEKA’s SMO algorithm learns a classifier with a linear decision boundary. By setting the `-K` parameter, we can change this to higher exponents. This is effectively mapping the original representation of data points to a different feature space. This is expected to give us better results, since many real-world problems are unlikely to be linearly separated.

The original weighted F-measure for the SVM is 0.822. Again, we see that the F-measure for class 4 is the highest, but its true positive rate for class 0 documents is at 0.86.

We ran the training set against the SMO algorithm with the `PolyKernel` at different exponents and evaluated the results.

Evaluating kernels of exponent 0.5, 1, 1.5, 2, and 3, we saw that the best performing SVM was when `-E` was set at 1.5, which performed slightly better than the default settings. This setting brings up the F-measure over all classes to over 0.8, which hopefully, gives us better performance. The best reported F-measure was for class 3 at 0.869. Higher orders of exponents seem to result in poorer results, suggesting that the data does not fit well to functions of orders 2 and above.

We attempt the same classification task with the RBF kernel, but obtained results poorer than the default settings for the SVM. Again, this suggests that the RBF kernel is not a good fit for the data.

Label	Exponent	TP Rate	FP Rate	Precision	Recall	F-Measure
SVM	1	0.821	0.045	0.823	0.821	0.822
SVM0.5	0.5	0.748	0.063	0.75	0.748	0.748
SVM1.5	1.5	0.844	0.039	0.845	0.844	0.844
SVM2.0	2.0	0.838	0.041	0.839	0.838	0.838
SVM3.0	3.0	0.831	0.042	0.833	0.831	0.832
SVMRBF	—	0.751	0.063	0.787	0.751	0.749

Table 2: Experiments with the number of features used.

Eventually, we decided to use **SVM1.5** for our SVM classifier. Our experimental results are shown in Table 2. These are listed as the weighted averages of the different shown values.

3.3 k -Nearest Neighbours (IBk)

The default setting for IBk on WEKA has the $k = 1$. This results in every new instance being classified the same as the first nearest neighbour it sees. We experiment with different values of k to find a good classifier.

The classifier trained with the default settings gave an F-measure of 0.774, and an F-measure of 0.805 on class 0 and 0.807 on class 4. From the performance of the other classifiers on this classification, this suggests that instances in classes 0 and 4 are more easily distinguishable from the rest of the dataset.

We experiment with different values of k , evaluating the classifier with different k values.

Label	k	TP Rate	FP Rate	Precision	Recall	F-Measure
IB1	1	0.774	0.056	0.775	0.774	0.774
IB5	5	0.762	0.06	0.77	0.762	0.763
IB10	10	0.768	0.058	0.775	0.768	0.768
IB20	20	0.769	0.058	0.776	0.769	0.77
IB30	30	0.76	0.06	0.767	0.76	0.761

Table 3: Experiments with the number of features used.

Using the `-X` function to choose k did not seem to contribute much to the results of the classifier. One noticeable characteristic of this classifier was that it was much faster than the training times of the SVM. However, the results of the classifiers tend to be poorer than that of the SVM as well. We also tried several iterations with the `-I` and `-F` parameters, which alter the weighting function so that the classifier behaves slightly differently, but these did not improve the performance of the overall classifier much.

It is worth noting that the IBk classifiers generally have nearly 0% errors when run on their own training set. This is because the points to be classified fall directly on themselves, giving the same results. As such, testing the IBk classifier on the same set of data as the training set is unproductive.

4 Combining the Classifiers

From each type of classifier, we pick the two best. We then combine the 6 resulting classifiers into one using the `weka.classifiers.meta.Vote` classifier. We choose to combine them by

majority vote, which means that the classification will be determined by the most common classification among the 6 classifiers.

Our chosen classifiers are:

1. NB
2. SVM1.0
3. SVM4.0-L
4. IB10-I
5. IB20-I

In our selection of the classifiers we have to take into consideration the performance of each of the included classifier. Having multiple classifiers prone to errors would result in a correspondingly error prone combined classifier. The benefit of combining them, however, is due to the fact that the different algorithms have different strengths and weaknesses, and giving only the output which majority of the classifiers agree on is likely to improve the overall performance. The error made by one classifier could then be corrected by the other classifiers.

Combining the classifiers, we run the training set against the newly formed classifier. The obtained results are in Table 4.

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
0	0.998	0	0.998	0.998	0.998
1	0.998	0	0.998	0.998	0.998
2	1	0	1	1	1
3	1	0	1	1	1
4	1	0	1	1	1
Weighted Avg.	0.999	0	0.999	0.999	0.999

Table 4: Class breakdown of the performance of the combined classifier.