MOVIE REVIEW SENTIMENT ANALYSIS

ANQI CHENG, WANYAN XIE

Abstract. In this work we performed sentiment classification of online movie reviews using different machine learning techniques. In section I we analyzed 20,000 supervised examples using Naive Bayes, linear support vector machine (SVM), linear classifiers with stochastic gradient decent (SGD) learning and Ridge algorithms; In section II we analyzed 50,000 unsupervised examples using k-means algorithm, with different initializations and latent semantic analysis.

1. Text feature extraction

1.1. **Bag of words representation.** The raw data of text is a sequence of symbols, and cannot be directly fed to most algorithms that expect numerical feature vector of fixed size. To address this problem we converted the text files into the 'bag of words representation' using tools provided by Python scikit-learn package. The process is as follows [1]:

- tokenizing strings and giving an integer id for each possible token.
- counting the occurrences of tokens in each document.
- normalizing and weighting with diminishing importance tokens.

where features and samples are defined as follows:

- each individual token occurrence frequency is treated as a feature.
- the vector of all the token frequencies for a given document is considered a multivariate sample.

1.2. **Re-weighting features.** Using term-frequency to represent textual information in the vector space has a problem: it scales up frequent terms and scales down rare terms which however may be more informative than the high frequency terms. One example is that some of the most frequent words (e.g., "the", "a", "is") might carry very little information about the actual contents of the documents. Therefore we re-weighted the features by term frequency inverse document frequency normalization ($tf - idf$). idf is defined as:

$$idf(t) = log\frac{|D|}{1 + |\{d : t \in d\}|}$$

where $D = \{d_1, d_2, ...d_n\}$ is the documents space, $|\{d : t \in d\}|$ is the number of documents where the term t appears. Therefore that a high weight of the $tf - idf$ calculation is reached only when a high term frequency (tf) and a low document frequency(df) are achieved. This weighting strategy help us to obtain a fair representation of the words.

2. Training supervised data

In this part we trained supervised data using Naive Bayes, linear support vector machine (SVM), linear classifiers with stochastic gradient decent (SGD) learning and Ridge algorithms. We randomly choose 20,000 movie reviews as the training data set, and the rest 5000 movie reviews as the validation data.

2.1. **Naive Bayes.** We used multinomial Naive Bayes as the classifier and performed 5-fold cross validation to find the best value for the additive smoothing parameter α. The scores of cross-validation for different α's are in table 1:

Table 1. 5-fold cross-validataion of Naive Bayes

α	0.1	1.0	2.0
Accuracy(%)	85.95(0.46)	86.34(0.80)	86.05(0.72)

We chose the parameters based on the best score, and tested the model on validation set. The confusion matrix is shown in Fig. 1. Naive Bayes classifier did not perform very well for our data, since apparently the 'naive' assumption of independence between every pair of features in movie reviews is not true.

2.2. **Linear SVM.** Next we implemented linear SVM classifier and did 5-fold cross validation to find the best parameters (penalty parameter C of the error term, and loss function: 'l1 is the hinge loss while 'l2 is the squared hinge loss.). The scores of cross-validation for different combination of parameters are shown in table 2:

Table 2. 5-fold cross-validataion by linear SVM

(C, loss)	Accuracy(%)
0.5, l1	89.22(0.48)
1.0, l1	89.30(0.53)
2.0, l1	88.78(0.25)
0.5, l2	89.43(0.34)
1.0, l2	89.34(0.29)
2.0, l2	89.19(0.48)

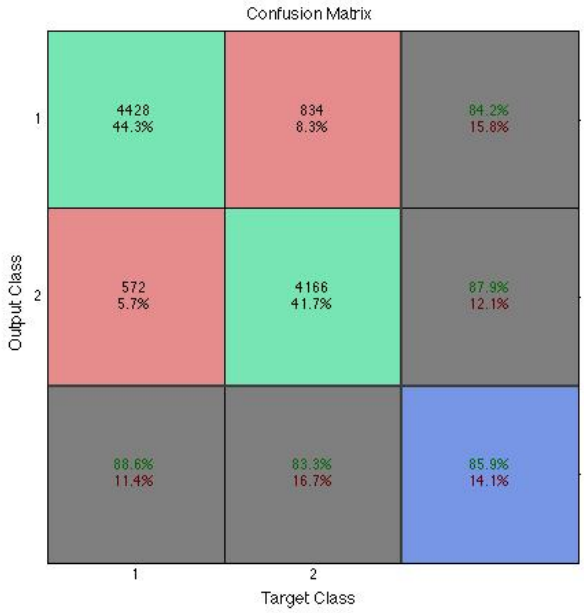

FIGURE 1. Confusion matrix on validation set by Naive Bayes.

We choose the parameters based on the best score, and test the model on validation set. The confusion matrix is shown in Fig.2.

2.3. **Ridge algorithms.** We also implemented Ridge regression to train our data. We used 5-fold cross validation to find the best parameters (alpha, corresponds to $(2 * C)^{-1}$ in other linear models such as linear SVM). The scores of cross-validation on the validation set are in table 3.

TABLE 3. Accuracy of 5-fold cross-validataion, tol=1e-2, solver="lsqr"

alpha	1.0	2.0	3.0
Accuracy(%)	89.20(0.15)	89.33(0.36)	88.91(0.45)

We chose the parameters based on the best score, and tested the model on validation set, The confusion matrix is shown in Fig.3.

2.4. **Linear classifiers with SGD learning.** Lastly we implemented several linear classifiers (linear SVM, logistic regression, etc) with SGD learning. This learning method uses stochastic gradient decent to estimate to update the loss function along

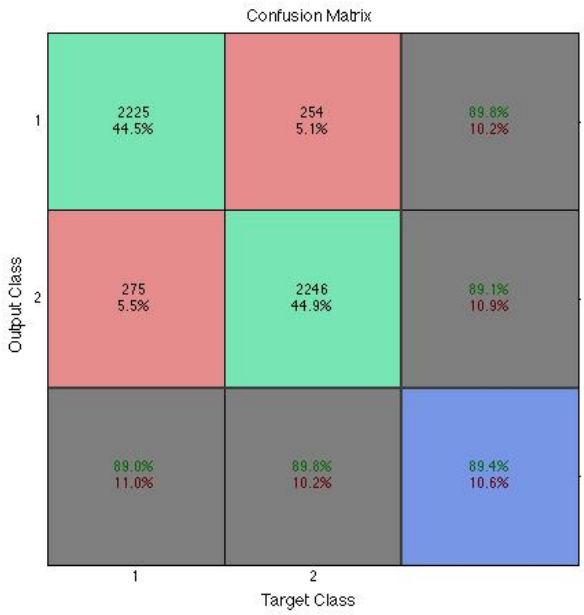

FIGURE 2. Confusion Matrix on validation set by Linear SVM.

with a decreasing strength schedule (aka learning rate).

We used 5-fold cross validation to find the best values of the parameters:

- loss function:
  - 'hinge' is linear SVM.
  - 'log is logistic regression.
  - 'modified-huber' brings tolerance to outliers and probability estimates.
  - 'perceptron' is the linear loss used by the perceptron algorithm.
- penalty: 'l1 is the hinge loss; 'l2 is the squared hinge loss.
- alpha: constant in front of the regularization term.
- niter: the number of passes over the training data.

The results are shown in table 4.

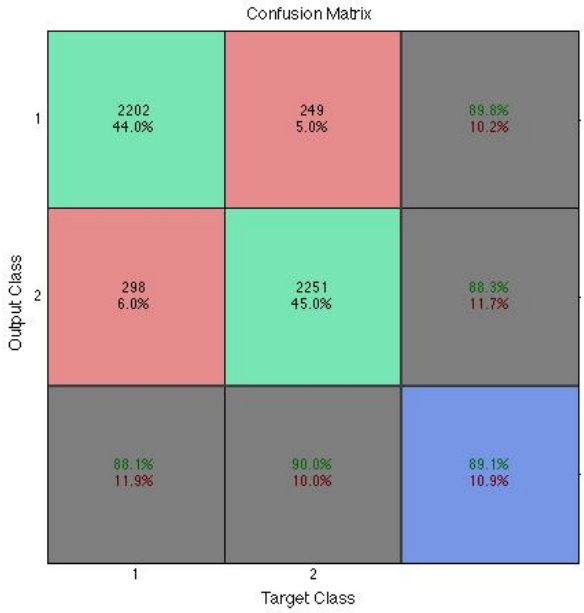

FIGURE 3. Confusion matrix on validation set by linear SVM.

TABLE 4. 5-fold cross-validataion by SGD

(loss, penalty, alpha, niter)	Accuracy(%)
perceptron, l2, 1e-5, 5	86.76(0.57)
log, l2, 1e-5, 5	87.91(0.54)
hinge, l2, 1e-3, 5	88.78(0.25)
hinge, l2, 1e-4, 5	88.81(0.41)
hinge, l2, 1e-5, 5	88.12(0.44)
modified-huber, l1, 1e-4, 5	86.52(0.49)
modified-huber, l2, 1e-5, 5	89.08(0.42)
modified-huber, l2, 1e-4, 15	89.37(0.33)

We choose the parameters based on the best score, and test the model on validation set. The confusion matrix is shown in Fig.4.

3. Clustering unsupervised data

In this part we classified 50,000 movie reviews into two sentiment classes (positive and negative) using k-means algorithm.

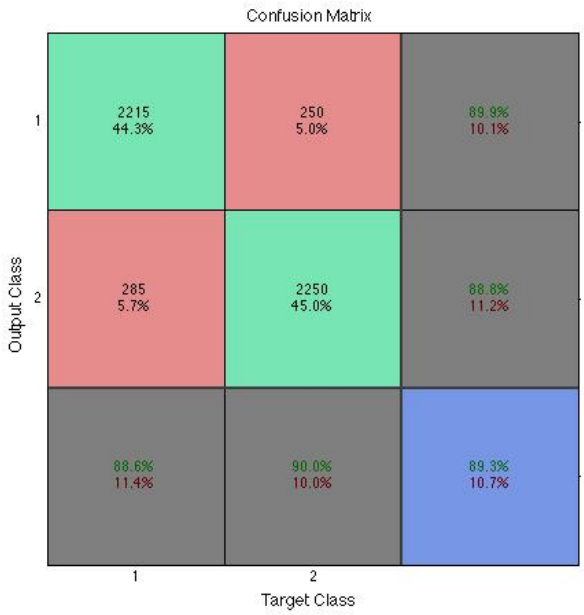

FIGURE 4. Confusion matrix on validation set by SGD.

3.1. **Initialization.** As the objective function of k-means is non-convex, it will likely end up in a local minimum. Therefore the location of initial centroids plays an important role in the performance of clustering. We explored three different kinds of initialization:

- Random initialization.
- From 20,000 supervised examples, randomly picked one example for each class, say (a_0, a_1).
- From 20,000 supervised data, took the mean of all examples for each class, say $(\bar{a}_0, \bar{a}_1)$.

For each initialization method we repeated 100 times and took the average of the outcome. The performances on a validation set of 5,000 examples are:

TABLE 5. Accuracy of k-means clustering with different initialization

Init. method	random	(a_0, a_1)	$(\bar{a}_0, \bar{a}_1)$
Accuracy(%)	46.7(13.0)	56.8(11.8)	66.5(1.0)

From table 3.1 we can see the performance is very sensitive to the initial location of the centroids. Input information from the supervised data greatly improved the performance. This makes sense because besides classifying the reviews by sentiment, there could be other ways of clustering (e.g, by topics of the movies).

3.2. **Latent semantic analysis.** From section 3.1 we chose the initial centroids to be the mean of two classes from the 20,000 supervised examples. To further improve performance of clustering we implemented truncated singular value decomposition (SVD) onto the term-document matrices, which is known as latent semantic analysis (LSA)[2].

Truncated SVD computes the k largest singular values where k is specified by users. Since it can transform overly sparse matrices to a "semantic" space of low dimensionality, LSA is known to combat the effects of synonymy and polysemy, and therefore can potentially improve the clustering performance.

With LSA we transformed 103036 features of the unsupervised data to k ($k = 200, 500, 1000$) features, and validated on 5,000 examples (again, each k repeated 100 times). The results in table 6:

Table 6. Accuracy of k-means clustering with LSA

Truncation	$k = 200$	$k = 500$	$k = 1000$
Accuracy(%)	67.6(2.5)	69.1(2.3)	71.4(1.5)

From table 6 we can see that LSA indeed improved the clustering performance, and the accuracy increases with the truncation size k. However since computing expense grows rapidly with increasing k, we stopped at $k = 1000$.

4. Summary

The best performance on the validation set for all models we explored are summarized in table 7.

Table 7. Comparison of all methods we explored

Model	Navie Bayes	Linear SVM	Ridge	SGD	K-means with LSA
Best Accuracy(%)	85.9	89.4	89.1	89.3	71.4

We can see that in general the supervised training performs much better than the unsupervised clustering, even though the size of unsupervised data is twice of the supervised data. In the end we chose linear SVM classifier to train the full 25,000 supervised reviews and score we got on the test set of 25,000 reviews is 87.56%.

References

[1] http://scikit-learn.org/stable/modules/feature_extraction.html
[2] http://scikit-learn.org/stable/modules/decomposition.html#principal-component-analysis-pca