Machine Learning **Shreshth Tuli**
COL774 - Lab Assignment 12 2016CS10680
Supervisor - Prof. Parag Singla Date: 13/03/2019

Introduction

In this assignment, we implemented Naive Bayes and Support Vector Machine algorithms in Python. The algorithms and models were tested using Yelp and MNIST datasets for accuracy and efficacy. We present different models, design decisions, parameters, hyper parameter tuning strategies used.

Naive Bayes

Implemented Naive Bayes algorithm for review classification on the Yelp dataset.

Simple tokenization of text - 1(a)

Using basic tokenization of the text, the training and test accuracy were:

1. Training Accuracy = 69.607 %, Macro-F1 score = 0.6179
2. Test Accuracy = 61.871 %, Macro-F1 score = 0.51

Random/Majority guessing - 1(b)

If we would have randomly guessed out of one the 5 classes then the probability of being correct would be 0.2, so we would get an expected accuracy of 20 %. Based on 5 experiments accuracies were : 20.8%, 21.7%, 19.7%, 20.01%, 19.9% all of which are close to 20%. The Naive Bayes model increases the accuracy by 41.8% compared to random guessing.

If we would have predicted the majority class i.e. class with stars = 5.0, which has 234,731 examples, then the probability of being correct out of total 534,872 examples would be 0.438, so the expected accuracy would be 43.9895 %. Experimentally too we get the same accuracy. The Naive Bayes model increases the accuracy by 17.8% compared to majority guessing.

Confusion Matrix - 1(c)

The confusion matrix for the test data is shown in Fig 1.
We observe the following:

1. Among the diagonal of the confusion matrix, the value corresponding to 5 is highest. This means that the prediction corresponding to the reviews with 5 stars is highest. This may be because of large number of class 5 examples in both training and test sets, and hence class 5 has high prior.
2. Other values like 1 and 4 also have relatively high prediction accuracy compared to reviews with stars = 2 or 3. This is because of the classifier easily learning how to classify low and high star reviews.
3. The predicted label is higher for the cell corresponding to the true label and less for others. In case of true label = 1, this distribution is not so apparent in other labels/classes.
4. The classifier typically classifies 4 or 5 in either of the two classes. In other words very seldom does the classifier misclassifies reviews with 4/5 stars with low stars.

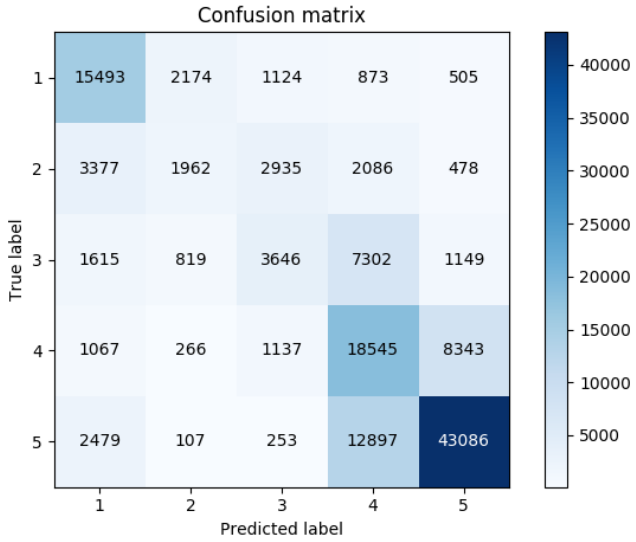


Figure 1: Confusion Matrix for 1(c)

Stemming and Stopword removal - 1(d)

As we had considered earlier the simple case of a feature vector containing all the words in the review, we can also consider a case where the stop words have been removed and all words are stemmed to their 'root'. This feature generation gives test accuracy = 60.686%, Macro-F1 score = 0.5197. We observe reduction in performance when removing stop words and stemming the text because the sentiment related information is lost. Consider 'can excel' and 'excellent' give the same result after stemming and stop-word removal but have very different meanings.

Feature Engineering - 1(e)

The following new features were tried over and above stemming and stopword removal:

1. Bigrams: Test accuracy = 62.918%, Macro-F1 score = 0.561
   Using bigrams improved the test accuracy slightly because the model then contains slight information of context which is important in sentiment analysis. For example: 'very' might be used with either 'good' or 'bad', so as such tells very limited information about the sentiment but a bigram 'verygood' is significantly different from 'verybad'. Similarly, 'good' can give very less information because it can be prefixed with 'very' or 'not' but as bigrams they provide significantly different information which is utilized to give better classification accuracy.

2. Increasing weight of first 10 words: Test accuracy = 61.901%, Macro-F1 score = 0.5281
   The initial few (say 10) words contain most of the information about the star rating as most of the words that follow are just describing it so we give a higher weight to the first 10 words in the review. This also increases performance.

3. Adding hash for 'good'/'bad': Test accuracy = 62.748%, Macro-F1 score = 0.564
   To give slight human based knowledge to the classifier we can also add some hash values that correspond to pleasant words like 'amazing', 'good', 'best', 'love', etc. and another hash for unpleasant words like 'bad', 'dissapoint', 'poor', etc. This way the algorithm can have a posterior probability highly dependent of which hash is present in the feature vector. This with bigrams gives the best accuracy = 63.298%, Macro-F1 score = 0.572.

F1 score - 1(f)

The F1 scores are as follows: 1 - 0.82, 2 - 0.19, 3 - 0.11, 4 - 0.48, 5 - 0.79

The macro-F1 score for the best model is 0.572. For such classification tasks F1 score is a better metric because it also considers precision and recall of the classifier.

Training on full Yelp dataset - 1(g)

Test accuracy = 73.642%
Macro-F1 score = 0.643

Support Vector Machines

Binary Classification

In this section we classified 0 vs 1 from the MNIST dataset. The expressions for y, w and b for linear kernel are:

- **w*** = $\sum_{i=1}^{m} \alpha_i y^{(i)} x(i)$
- **b*** = $-\frac{max_{i:y^{(i)}=-1} w^{*T} x^{(i)} + min_{i:y^{(i)}=1} w^{*T} x^{(i)}}{2}$
- **y** = $w^T x + b$

Genera case:

- **b*** = $\frac{1}{N_S} \sum_{i \in S} (y^{(i)} - \sum_{j \in S} \alpha_j y^{(j)} k(x^{(i)}, x^{(j)})$ where S is set of support vectors and N_S is it's cardinality
- **y** = $\sum_{i=1}^{m} \alpha_i y^{(i)} k(x, x_i) \ + \ b$ where k(x,z) = $x^T z$ for linear kernel, and k(x,z) = $e^{-\gamma ||x-z||^2}$ for gaussian kernel

CVXOPT results Linear kernel

- Test accuracy = 99.9054373522
- Bias = 0.68507907023
- Number of support vectors = 6
- Training time = 43.75 seconds

CVXOPT results Gaussian kernel

- Test accuracy = 99.7635933806
- Bias = -0.672641168
- Number of support vectors = 241
- Training time = 236.88 seconds

Linear Kernel performs slightly better than Gaussian one, but for other values of D this may not be the case.

LIBSVM results Linear kernel

- Test accuracy = 99.9054373522
- Bias = 0.787314
- Number of support vectors = 53
- Training time = 0.56 seconds

LIBSVM results Linear kernel

- Test accuracy = 99.8108747045
- Bias = -0.752569
- Number of support vectors = 745
- Training time = 2.22 seconds

Here too Linear Kernel performs slightly better than Gaussian one, but for other values of D this may not be the case. LIBSVM is much faster compared to cvxopt. Also bias values are similar for cvxopt and libsvm.

The examples that are wrongly classified by the Linear kernel are truly difficult to classify!

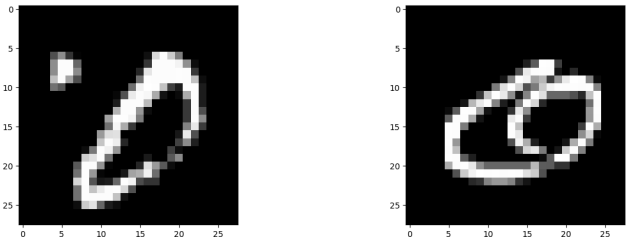


Figure 2: Wrongly classified to 1 by Linear kernel SVM

Multiclass classification

CVXOPT results

- Multiclass Training Accuracy (Linear Kernel) = 96.89%
- Multiclass Test Accuracy (Linear Kernel) = 90.85%
- Multiclass Training Accuracy (Gaussian Kernel) = 97.195%
- Multiclass Test Accuracy (Gaussian Kernel) = 96.29%

Training time for Gaussian kernel = 47 minutes 28.34 seconds

LIBSVM results

- Multiclass Training Accuracy (Linear Kernel) = 98.705%
- Multiclass Test Accuracy (Linear Kernel) = 92.73%
- Multiclass Training Accuracy (Gaussian Kernel) = 99.92%
- Multiclass Test Accuracy (Gaussian Kernel) = 97.24%

Training time for Gaussian kernel = 6 minutes 17.298 seconds. LIBSVM is much faster.

Confusion Matrix

We can observe the following:

1. We see that the diagonal values of the confusion matrix are the highest in each row and column. This measn that our classifier is performing well

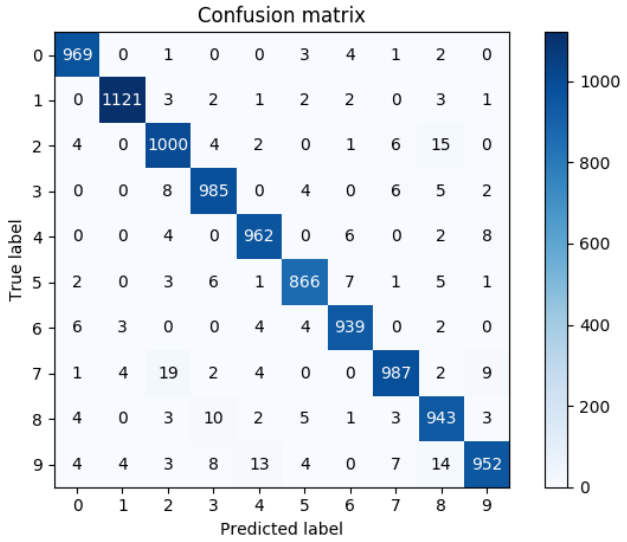


Figure 3: Confusion Matrix for Multiclass Gaussian kernel SVM classifier

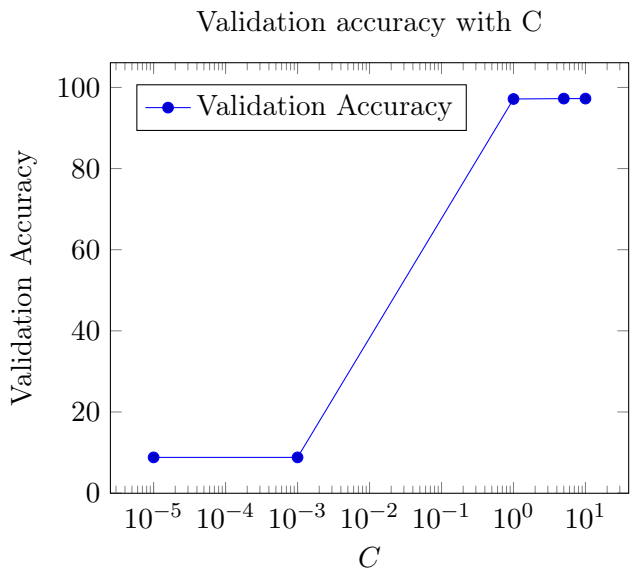
2. Due to closeness in geometric resemblance of the symbols the following are most of mis-classified = (0,6), (1,2), (2,8), (3,2), (4,9), (5,6), (6,0), (7,2), (8,3), (9,4) where the left number is the true value and right one is predicted value.

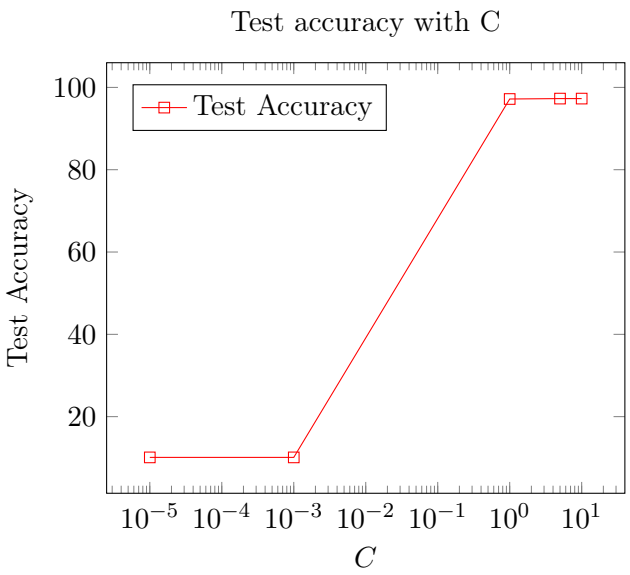
3. These deviations from true value make sense due to their similarity

Validation

The validation and test accuracies for C = (10^{-5}, 0.001, 1, 5, 10) are given below:

1. Validation Accuracy with C = 1e-05 is : 8.8%
2. Test Accuracy with C = 1e-05 is : 10.1%
3. Validation Accuracy with C = 0.001 is : 8.8%
4. Test Accuracy with C = 0.001 is : 10.1%
5. Validation Accuracy with C = 1 is : 97.15%
6. Test Accuracy with C = 1 is : 97.18%
7. Validation Accuracy with C = 5 is : 97.25%
8. Test Accuracy with C = 5 is : 97.28%
9. Validation Accuracy with C = 10 is : 97.25%
10. Test Accuracy with C = 10 is : 97.28%





We see that as the validation accuracy increases (with increasing C), the test accuracy also increases. The best set of test/validation accuracies are for C = 5 or C = 10.