COL774: Assignment 2

Nikhil Goyal
2015CS50287

March 12, 2018

Note: In all the confusion matrices row represents truth label and column denotes predicted label

1 Text Classification

1.a

Every punctuation and non-printable character(emojis etc.) replaced by a space character. All words converted to lowercase.

Training Accuracy = 68.400%
Test Accuracy = 38.708%

1.b

The set of classes(S) = {1, 2, 3, 4, 7, 8, 9, 10}. Probability of correctly guessing single label correctly(p)= $\frac{1}{|S|} = \frac{1}{8}$

Let X be random variable denoting number of correct guesses by the random classifier. Then $X \sim B(m, p)$, where m = number of examples. Accuracy of random classifier is given by $\frac{E[X]}{m} = \frac{mp}{m} = p$. So, accuracy of random classifier = 12.50%

Total examples(m) = 25000 and max occurrence class is 1(5100 times). In this case accuracy is $\frac{5100}{25000} = 20.40\%$. Our algorithm achieves about 90% improvement over max occurrence classifier.

1.c

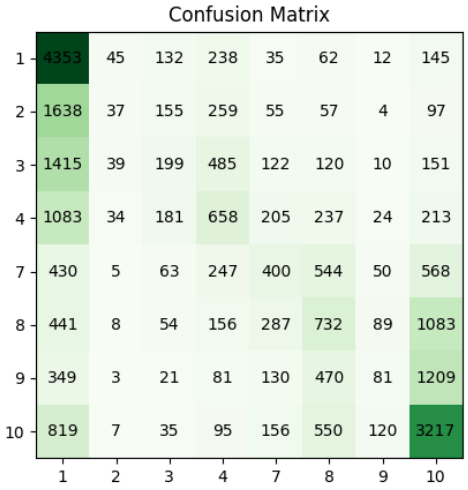

Figure 1: Confusion matrix for the Naive Bayes model described in part 1.a

High values on diagonal entries for label 1 and 10 signifies that Naive Bayes classifier does a good job in classifying these labels, however we can see from the confusion matrix all lot of class 2 and class 9 examples have been misclassified as 1 and 10 respectively suggesting classifier does a very poor job in classifying adjacent classes. This is due to the fact that language used in the adjacent classes is similar leading to ambiguity in training data. Hence values adjacent to diagonal matrix have a high value and form bulk of misclassified examples

Category 1 has the maximum value of diagonal entry. The training data provided is heavily biased as classes 1 and 10 make almost 40% of the data. Its leads to serious errors in the classifier and is supported by the fact that about huge proportion of wrongly classified examples are misclassified as 1 or 10

1.d

Stopword removal and stemming performed.

Training Accuracy = 67.996%
Test Accuracy = 38.684%

No improvement can be seen in the classifier even after using stemming and stop-words. It may be possible that certain stop words are really associated with certain review class (ex. too is related to highly polar reviews). Also generally a high rated review would be of long length but stop-word removal gives away the length of review completely.

1.e

As discussed in previous section length of review can be a good feature. Adding a feature for the number of words in the review doesn't change the accuracy as it is heavily biased with stop-words. Performing stop-word removal and stemming in conjunction with it gives test accuracy = 38.784 %, thus providing no boost to performance. However performing only stop-word removal and keeping review length as feature increases the test accuracy to 39.264%. Training accuracy in this case was 71.86%. This may be due to the fact that particular words are characteristic of some class (ex. Excellent) but stemming leads to removal of such crucial features.

Using bi-grams on top of stop-word removal and stemming didn't yield any better results. Accuracy obtained in this case was 38.58%. Training model by taking equal examples from each class did decrease increase accuracy of non polar reviews but the overall accuracy decreased due to reduction of training data size. Removing features having same count in class 1 and class 10 didn't help either.
I think somehow we need to come up with a parameter that helps us to find correlation between a word and it's presence in the the class. This will help to remove class imbalance and improve accuracy significantly.

2 MNIST Handwritten digit Classification

2.a

Let the batch size = k and consider this approximate formulation $f(w,b) = \min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{k} max(0, 1 - t_i)$, where $t_i = y^{(i)}(w^T x^{(i)} + b)$. Mini-batch stochastic gradient descent update rule for t^{th} iteration is

$$(w,b) \leftarrow (w,b) - \eta_t \nabla_{(w,b)} f(w,b)$$

Solving further, it works out to be the following

$$w \leftarrow (1 - \eta_t) w + \eta_t C \sum_{i=1}^{k} \mathbf{CH}(y^{(i)}(w^T x^{(i)} + b) < 1) y^{(i)} x^{(i)}$$

$$b \leftarrow b - \eta_t C \sum_{i=1}^{k} \mathbf{CH}(y^{(i)}(w^T x^{(i)} + b) < 1) y^{(i)}$$

where $\eta_t = \frac{1}{t}$ and **CH** is characteristic function

Stopping criteria: Fixed number of iterations(2000)

2.b

One vs One classification using Pegasos algorithm

Training Accuracy = 93.965%
Test Accuracy = 92.38%

2.c

Linear Kernel SMO: Test Accuracy = 92.76%
RBF kernel SMO: Test Accuracy = 97.23%

Accuracy obtained in Pegasos and Linear SMO is comparable. However Pegasos converges to a near optimum solution very fast.
RBF Kernel works better then both of these as the classes are easily separable in the infinite length feature space formed by the transformation.

2.d

C	Validation accuracy	Test Accuracy
10^{-5}	71.610	72.100
10^{-3}	71.675	72.100
1	97.435	97.230
5	97.525	97.290
10	97.525	97.290

Table 1: Average validation set accuracy and test set accuracy

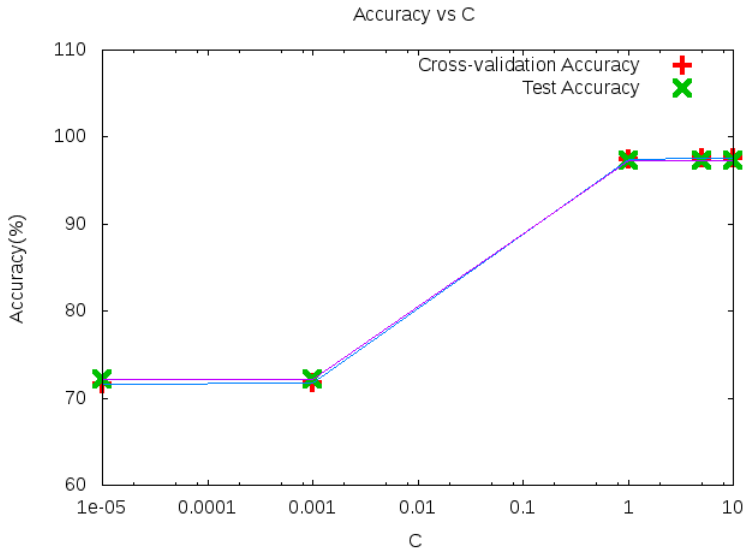

Figure 2: Average validation set accuracy and test set accuracy vs C(log-scale)

$C = 5$ gives the best average cross-validation accuracy and test set accuracy when $\gamma = 0.05$.

Usually, cross-validation is used to estimate the performance obtained using a method for building a model, rather than for estimating the performance of a model.
If cross-validation is used to estimate the hyper-parameters of a model, then cross-validation estimate of performance is likely to be optimistically biased. This is because part of the model (the hyper-parameters) have been selected to minimize the cross-validation performance.

2.e

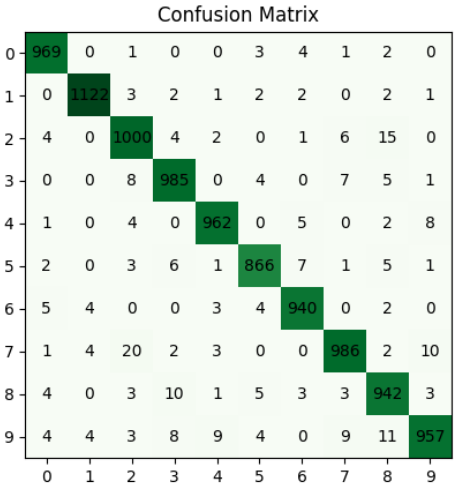

Figure 3: Confusion matrix for model trained with $C = 5$ and $\gamma = 0.05$

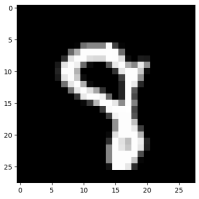

Figure 4: Truth label = 8 Predicted label = 9

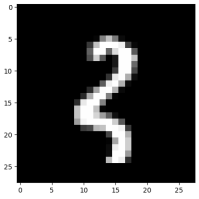

Figure 5: Truth label = 3 Predicted label = 2

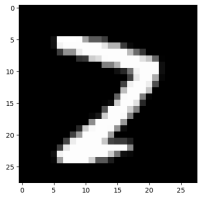

Figure 6: Truth label = 7 Predicted label = 2

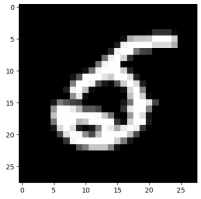

Figure 7: Truth label = 5 Predicted label = 6

As we can see most of the misclassified examples are ambiguous for even humans to read.

Accuracy of digit i = $\frac{CM[i][i]}{\sum_{j=0}^{9} CM[i][j]}$
Digit 9 has the worst accuracy of 95.22%