Assignment 2

Machine Learning

Yuyutsu Saini

4^{th} October, 2022

COL 774: Machine Learning

Contents

1 Text Classification

We will be using Naïve Bayes approach to classify the text in one of the two categories i.e. review being negative and positive. Here, Our assumption is occurrence of a word in a review is conditionally independent of occurrence of another word. All precision, recall and f1 scores are on test data for different features taken for training the model.

- Libraries Used
  - ➢ os
  - ➢ sys
  - ➢ time
  - ➢ numpy
  - ➢ pickle
  - ➢ nltk for stopwords and PorterStemmer
  - ➢ wordcloud
  - ➢ random

1.1 Naïve Bayes

1.1.1 Training and Test Accuracies

We used a slight variation of multinomial classifier model i.e. the bag of words model. This part is just simply using the words in reviews without stemming in unigram form to train the model. I have used logarithms to avoid underflow issues. The observations made are:

1. The value of smoothing parameter $\alpha = 1$.

2. The accuracy over training data is 94.912%

3. The accuracy over test data is 82.91333333333333%

1.1.2 Word Clouds for Positive and Negative Reviews without stemming

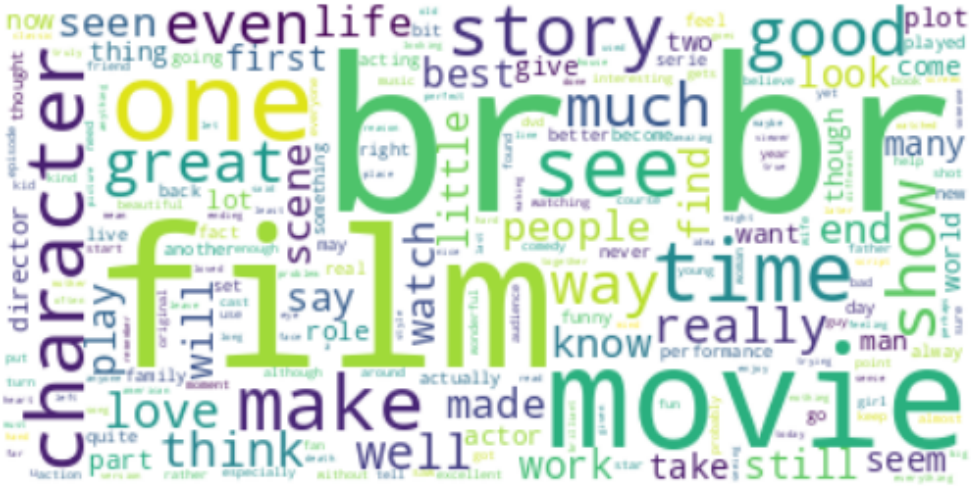


Figure 1: Positive Reviews Word Cloud

1.2 Random and Positive Guessing Models

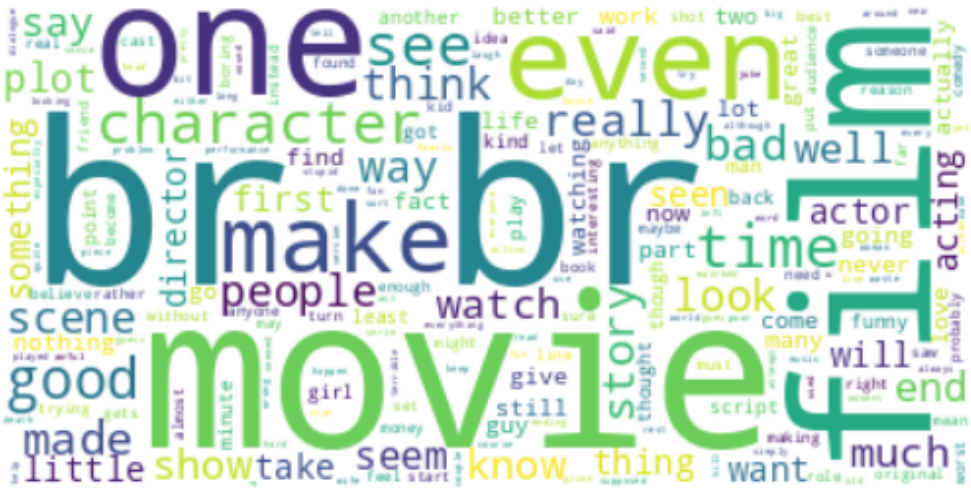


Figure 2: Negative Reviews Word Cloud

1.3 Random and Positive Guessing Models

1.3.1 Random Prediction Model

The probability of predicting a new review to be positive or negative is $\frac{1}{2}$ as it equally probable to guess a particular review to be of correct type. Here, in the given test samples we get the accuracy as 50.1933333333333% which is quite close to 50% implying both reviews are nearly equally likely in this data sample.

1.3.2 Positive Guessing Model

Here, the probability depends on the test samples i.e. which are types of reviews are more likely and hence changing the probability. in the given test samples we get the accuracy as 66.66666666666666% signifying that it is more likely to get a positive review compared to a negative review.

1.3.3 Analysing Accuracy of different Models

It is quite evident that both of the above model are very bad for predicting that a review is positive or negative. Our model gives the following improvement in accuracy over the above given models:

1. Improvement in accuracy over the random prediction model: $82.91333333333333\% - 50.1933333333333\% = 32.720\%$.
2. Improvement in accuracy over the random prediction model: $82.91333333333333\% - 66.66666666666666\% = 16.24666666666667\%$

1.3.4 Confusion Matrix

Confusion Matrix for different Models are as follows:
The confusion matrix for our trained model is:

$$\begin{bmatrix} 11753 & 525 \\ 747 & 11975 \end{bmatrix}$$

The confusion matrix for random prediction model is:

$$\begin{bmatrix} 5023 & 2494 \\ 4977 & 2506 \end{bmatrix}$$

The confusion matrix for Positive guessing model is:

$$\begin{bmatrix} 10000 & 5000 \\ 0 & 0 \end{bmatrix}$$

The confusion matrix given by our model as highest diagonal entry if we see per 100 examples (i.e percentage) and its denotes the accuracy of our model which is number of samples cases correctly guessed by our model is the accuracy for that model.

We can observe that the True Positive value of all the matrices is the maximum among all the entries implying that all the models are majorly predicting the positive to be correct. Also in our model the false negative error is more than the false positive in the confusion matrix, thus the model is predicting more positive reviews to be negative than the negative reviews to be positive.

In the positive guessing model, no positive review is wrong predicted because it is simply saying all reviews to be positive, therefore no negative, hence no positive being wrong predicted.

1.4 Stemming and excluding Stopwords

The results of stemming and excluding Stopwords the `test data` are:

1. The training data accuracy is 95.56666666666666%
2. The test data accuracy is 83.56666666666666%

Here, we observe that both the training accuracy and the testing accuracy increases because of removing usual words and adjusting similar words.

Figure 3: Positive Reviews Word Cloud with Stemming

1.5 Random and Positive Guessing Models

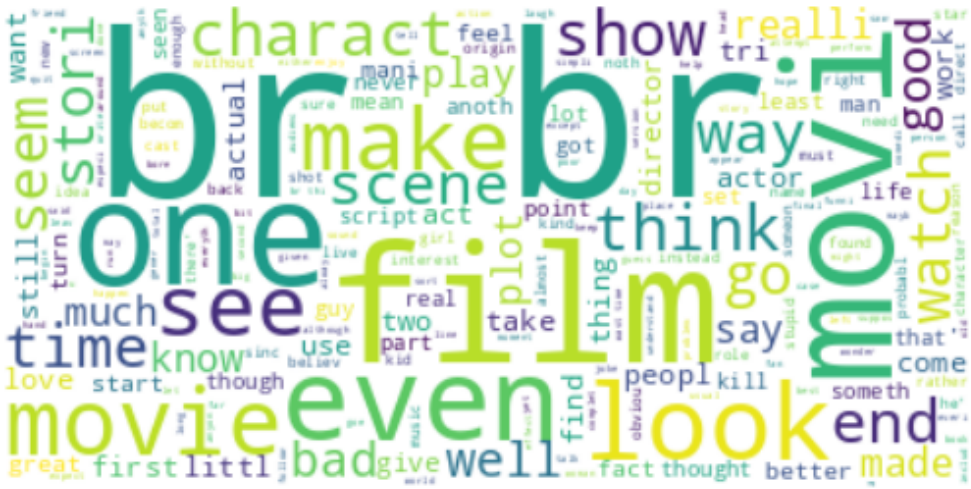


Figure 4: Negative Reviews Word Cloud with Stemming

1.6 Feature Engineering

1.6.1 Bigram over unigrams without stemming

1. The training data accuracy is 99.724%
2. The test accuracy is 86.313%

The training accuracy improves significantly because while using bigram over unigrams, we are basically over fitting the data. Contrary, there is a very slight increase in the accuracy of test data set as model learns quite low in general.

1.6.2 Bigrams over unigrams + Stemming

1. The training data accuracy increases to 99.956%
2. The test data accuracy is 85.713%

With stemming the training data accuracy further increases possibly due the frequency of similar occurring words in training data but the test data accuracy decreases as it overfits to the data taking into account the very details in training data.

1.6.3 TriGrams over Bigrams over unigrams + Stemming

1. The training data accuracy is 100%
2. The test accuracy is 86.0%

The training accuracy improves to 100 percent because we learning more and more on training data and trying to match the test data more and more with training data. Here, the test accuracy also increases with small amount as the test may be in resemblence with the training data. Further moving on to QuadGrams, it will lead to more and more overfitting leading to decrease in test accuracy and learning more and more on training data.

1.6.4 Analysis of different features

1. The stemming increases the accuracy because the model is randomly distrubed by the different forms of different and hence was more dependent on the training and test data. After stemming our model becomes less sensitive to given data.
2. Performing bi-grams takes into the account the probability of occurring of a word after a particular word (conditional probability) which increases the accuracy on training data and here, in our case test accuracy also increases which implies that the words are dependent on the previous words and there is some probability related with words occurring in specific sequence.
3. TriGrams further improves accuracy on training data set by overfitting on training data and taking into account conditional probabilities of three consecutive words.

1.6.5 Comparison with models of previous parts

1. The initial model is working quite fine on the training and test data both.
2. The stemming model increases the accuracy by 2 percent because stemming decreases the different forms of same word to a same word.
3. BiGrams over stemming further improves the test accuracy to about 86 percent.
4. BiGrams without stemming improves the test accuracies to shoot above 86 percent. It may be happening because of the occurrence of a specific form of a word after a particular word in a specific class (positive review or negative review class).

1.7 Precision, Recall and F1 Score

1.7.1 First Model without stemming

The Precision, Recall and F1 Score for the first model is as follows:

1. Precision is 0.9152428810720268%

2. Recall is 0.8196%
3. F1 Score is is 0.8647850171458717%

1.7.2 Model with stemming

The Precision, Recall and F1 Score for the first model is as follows:

1. Precision is 0.9196836359585607%
2. Recall is 0.8256%
3. F1 Score is is 0.8701059176898351%

1.7.3 Model with BiGrams + Stemming

The Precision, Recall and F1 Score for the first model is as follows:

1. Precision is 0.9334657398212513%
2. Recall is 0.846%
3. F1 Score is is 0.8875832765042228%

1.7.4 Analysing the above metrics

F1 Scores performs the best because it takes into account contribution from both precision and recall. Hence F1 score is more general and best among all the metrics.

2 Binary Image Classification

- Libraries Used
  - ➢ os
  - ➢ sys
  - ➢ time
  - ➢ numpy
  - ➢ pickle
  - ➢ sklearn.svm
  - ➢ cvxopt
  - ➢ cvxopt.solvers

2.1 Binary Classification

Here in this problem, I'm using the RBG image of size $32 \times 32 \times 3$ to form a vector of length 3072 and then classifying the images into two classes using SVMs. My entry number ends with 1 and therefore the classes considered are 1 and 2 in the Support Vector Machine.

2.1.1 Using Linear Kernel

We are using the CVXOPT package, we need to first transform the dual problem into the CVXOPT API acceptable form

$$\begin{aligned} &\alpha^T P \alpha + q^T \alpha + d \\ &G\alpha \preceq H \\ &A\alpha = b \end{aligned} \tag{1}$$

The dual in summation format is given as:

$$\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^i y^j (x^i)^T x^j - \sum_{i=1}^{m}\alpha_i \tag{2}$$

It is easy to see that P_{ij} is the coefficient of $\alpha_i\alpha_j$. Therefore, P_{ij} is given as:

$$\begin{aligned} P_{ij} &= y^i y^j (x^i)^T x^j \\ \implies P &= X_y \times X_y^T \\ \text{where } X_y &= \text{each row of } X \text{ multiplied by } Y \end{aligned} \tag{3}$$

Also, d is 0 and q is a vector with all -1:

$$\begin{bmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix} \tag{4}$$

The condition on α_i is $0 \leq \alpha_i \leq c$. Therefore G and H are given as:

$$\begin{aligned} G &= \begin{bmatrix} -1 & 0 & \dots & 0 \\ 0 & -1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -1 \\ 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \\ H &= \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \\ c \\ \vdots \\ c \end{bmatrix} \end{aligned} \tag{5}$$

The equality condition is $\sum_{i=1}^{m}\alpha_i y^i = 0$. Therefore A and b are given as:

$$\begin{aligned} A &= \begin{bmatrix} y_1 & y_2 & \dots & y_m \end{bmatrix} \\ b &= 0 \end{aligned} \tag{6}$$

In this format, we pass them to the CVXOPT quadratic program solver and we got these results:

1. Training Time: 16.70463514328003 seconds
2. Number of support vectors in Class 1 are 826.0
3. Number of support vectors in Class 1 are 826.0
4. Number of support vectors in Class 2 are 721.0
5. Total Number of support vectors in Class 2 are 1547.0
6. Support Vector percentage in training examples is 38.675 %
7. w for linear kernel is: $\begin{bmatrix} -0.74565394 \\ 0.03278262 \\ -0.48368186 \\ \vdots \\ -0.09051983 \\ 0.09775812 \\ -1.04531297 \end{bmatrix}$
8. b for linear kernel is : -0.01833718
9. Testing Time over training data set:0.7171199321746826 seconds
10. Testing Time over testing data set: 0.39896464347839355 seconds
11. The accuracy over training data is: 94.875%
12. The accuracy over test data is: 78.10000000000001%

The support vector images corresponding to the top-5 coefficients are as follows:

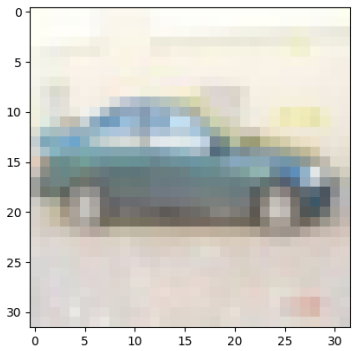


Figure 5: Support Vector A1

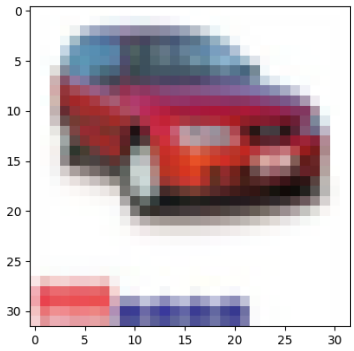


Figure 6: Support Vector A2

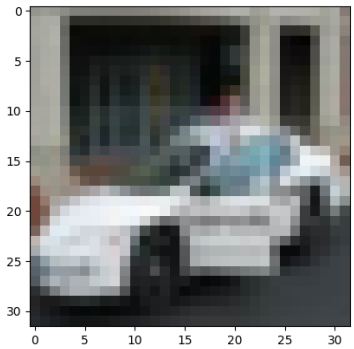


Figure 7: Support Vector A3

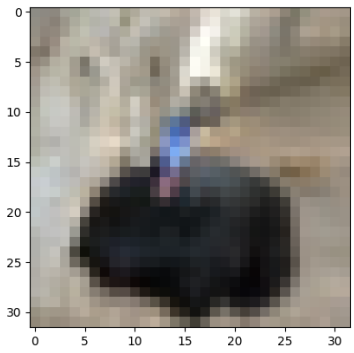


Figure 8: Support Vector A4

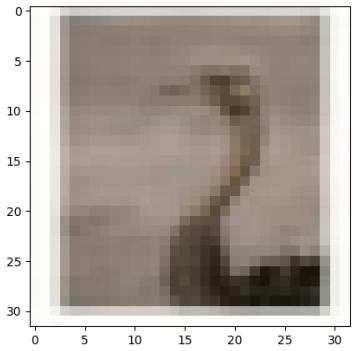


Figure 9: Support Vector A5

The Plot w (weight) vector is as follows:

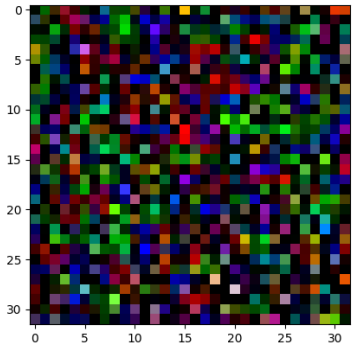


Figure 10: w (weight) Vector

2.2 Using Gaussian Kernel

The dual SVM problem transforms as following if use the guassian kernel:

$$\min(-\sum_{i=1}^{m}\alpha_i + \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^i y^j \phi(x^i)^T\phi(x^j)) \tag{7}$$

The only difference will be in the value of P, and P is given as:

$$\begin{aligned} P_{ij} &= y^i y^j \phi(x^i)^T \phi(x)^j \\ \implies P_{ij} &= y^i y^j \exp(-\gamma||x^i - x^j||^2) \\ \implies P_{ij} &= y^i y^j \exp(-\gamma(||x^i||^2 + ||x^j||^2 - 2x^i x^j)) \end{aligned} \tag{8}$$

We generalise this equation for computing the *product* of any two vectors X, Z of sizes n, m respectively as:

$$\mathcal{P}(X,Z) = \begin{bmatrix} ||X_1||^2 & ||X_1||^2 & (m \text{ times}) \dots & ||X_1||^2 \\ ||X_2||^2 & ||X_2||^2 & (m \text{ times}) \dots & ||X_2||^2 \\ \vdots & \vdots & \ddots & \vdots \\ ||X_n||^2 & ||X_n||^2 & (m \text{ times}) \dots & ||X_n||^2 \end{bmatrix} + \begin{bmatrix} ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \\ ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \\ \vdots & \vdots & & \vdots \\ n \text{ times} & n \text{ times} & \ddots & n \text{ times} \\ \vdots & \vdots & & \vdots \\ ||Z_1||^2 & ||Z_2||^2 & \dots & ||Z_m||^2 \end{bmatrix} - 2(X \otimes Z) \tag{9}$$

This type of formulation is made to solve the Gaussian kernel using the numpy methods. The values are then again passed to the CVXOPT quadratic problem solver. We make predictions as:

$$(\alpha \times Y) \times \exp(-\gamma \mathcal{P}(X_{SV}, X_{data})) + b \tag{10}$$

The following are the results obtained:

1. Training Time: 15.430006265640259 seconds
2. Number of support vectors in Class 1 are 945.0
3. Number of support vectors in Class 2 are 936.0
4. Total Number of support vectors in Class 2 are 1881.0
5. Support Vector percentage in training examples is 47.025 %
6. The accuracy over test data is: 82.1%

The accuracy obtained here is approximately 4 percent more than the model trained using cvxopt package implying scikit trains better than cvxopt package.

The support vector images corresponding to the top-5 coefficients are as follows:

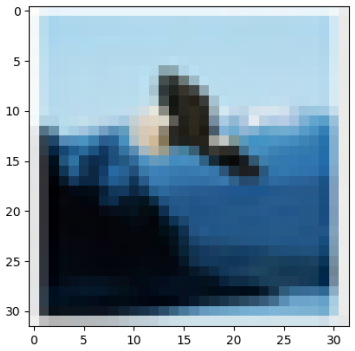


Figure 11: Support Vector B1

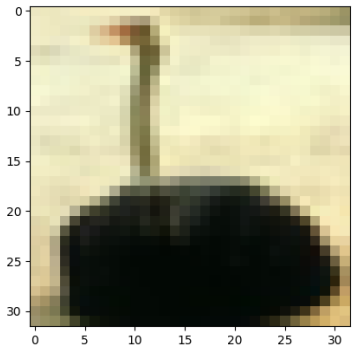


Figure 12: Support Vector B2

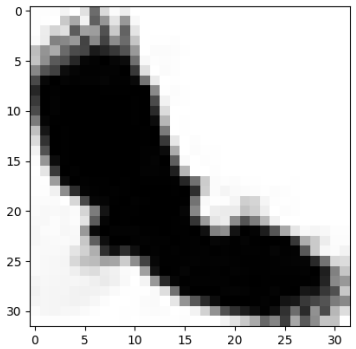


Figure 13: Support Vector B3

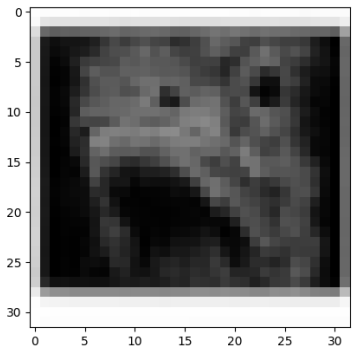


Figure 14: Support Vector B4

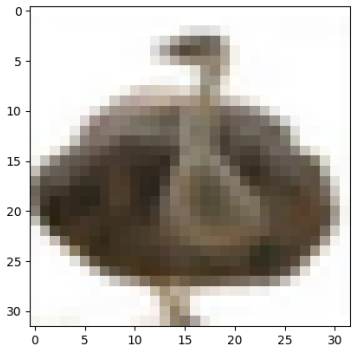


Figure 15: Support Vector B5

2.3 Using scikit learn SVM function, based on LIBSVM package

Linear Kernel The results are as follows:

1. Time taken to train with Linear kernel by scikit: 17.533120155334473 seconds
2. No. of Support Vectors in Class 1 are 822
3. No. of Support Vectors in Class 2 are 719
4. Total Number of support vectors in Class 2 are 1541
5. Support Vector percentage in training examples is 38.525 %
6. w for linear kernel is: $\begin{bmatrix} -0.74586634 \\ 0.03277407 \\ -0.48379131 \\ \vdots \\ -0.09057429 \\ 0.09792686 \\ -1.04533378 \end{bmatrix}$
7. b for linear kernel is : 0.01147231
8. The accuracy over test data is: 78.05%

Gaussian Kernel The results are as follows:

1. Time taken to train with Gaussian kernel by scikit: 11.59799337387085seconds
2. Number of support vectors in Class 1 are 939
3. Number of support vectors in Class 2 are 926
4. Total Number of support vectors (nSV) are 1865
5. Support Vector percentage in training examples is 46.625 %
6. The accuracy over test data is: 87.4%

Training Time Comparison

- The scikit uses approximately 1 second more than cvxopt to train the model with linear kernel, scikit takes slightly more and trains upto less error.
- The scikit uses approximately 4 second more than cvxopt to train the model with gaussian kernel, scikit is relatively slow than cvxopt and requires more time when more computation is there.

3 Multi-Class Image Classification

- Libraries Used
  - ➢ os
  - ➢ sys
  - ➢ time
  - ➢ numpy
  - ➢ pickle
  - ➢ cvxopt
  - ➢ cvxopt.solvers
  - ➢ sklearn.svm
  - ➢ random

3.1 One vs One Classifier using CVXOPT package (Gaussian kernel is Used)

The Class with maximum number of votes is considered as predicted class. In case of ties, the label with highest score is chosen.
The results are as follows:

1. Training Time over all classes by cvxopt: 168.1200668811798 sec
2. The accuracy over test data is: 57.02%

3.2 One vs One Classifier using Scikit Learn SVM LIBSVM package (Gaussian kernel is Used)

The results are as follows:

1. Training Time over classes by scikit: 130.7175416946411 sec
2. The accuracy over test data is: 59.3%

The training time is quite less while scikit which trains in approximately 38 seconds before the cvxopt. Here, it is quite clear that scikit faster than cvxopt in multi class classification.

3.3 Confusion Matrix

The confusion matrix using CVXOPT is:

$$\begin{bmatrix} 724 & 121 & 135 & 79 & 75 \\ 46 & 562 & 23 & 32 & 18 \\ 101 & 87 & 457 & 128 & 283 \\ 75 & 193 & 214 & 684 & 200 \\ 54 & 37 & 171 & 77 & 424 \end{bmatrix} \tag{11}$$

Observation

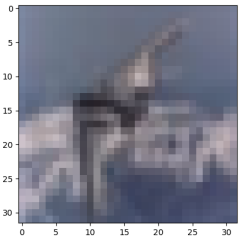
- The trace of the confusion matrix corresponds to correct prediction and rest all are the wrong predictions.
- Many Birds are wrongly classified as Cat or Deer. Also more of deer are wrongly classified as cat or bird.

The confusion matrix using LIBSVM is:

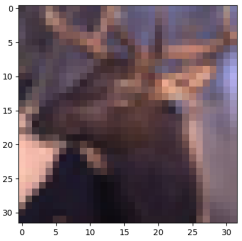
$$\begin{bmatrix} 729 & 100 & 145 & 82 & 98 \\ 83 & 731 & 61 & 97 & 48 \\ 78 & 42 & 409 & 123 & 212 \\ 61 & 92 & 133 & 572 & 118 \\ 49 & 35 & 252 & 126 & 524 \end{bmatrix} \tag{12}$$

- Scikit also gives quite similar results. More of Birds are classified as deer in the case of scikit model.
- Scikit trained model also gives more accuracy over the test data.

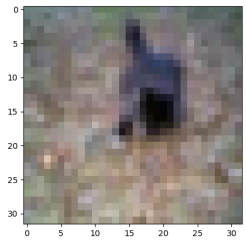
The 10 misclassified examples by cvxopt package are as follows:



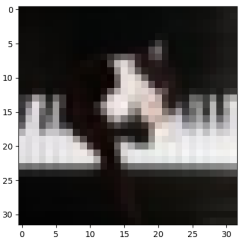
(a) Example 1: Predicted Aeroplane(Bird)



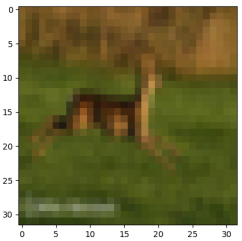
(b) Example 2: Predicted Bird(Deer)



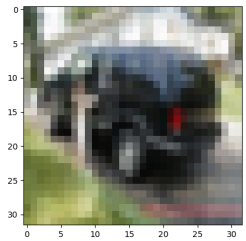
(c) Example 3: Cat(Bird)



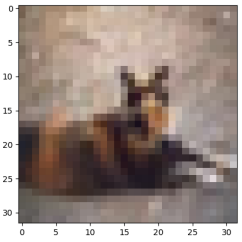
(d) Example 4: Predicted Aeroplane(Bird)



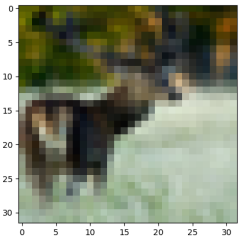
(e) Example 5: Predicted Bird(Deer)



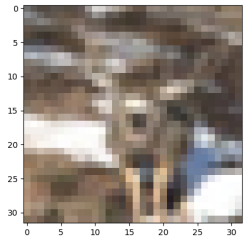
(f) Example 6: Predicted Deer(Car)



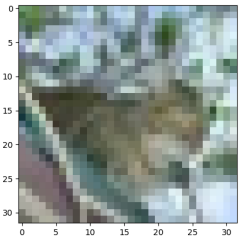
(g) Example 7: Predicted Deer(Cat)



(h) Example 8: Predicted Bird(Deer)

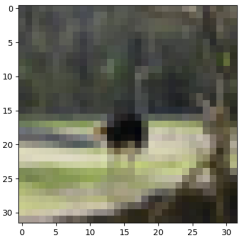


(i) Example 9: Predicted Bird(Deer)

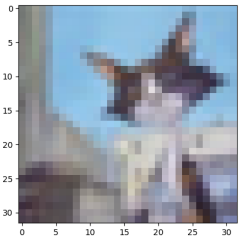


(j) Example 10: Predicted Cat(Deer)

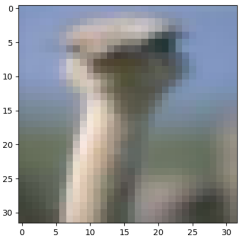
The 10 misclassified examples by scikit package are as follows:



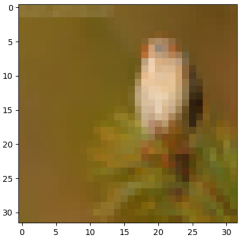
(a) Example 1: Predicted Cat(Bird)



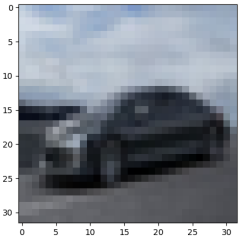
(b) Example 2: Predicted Aeroplane(Cat)



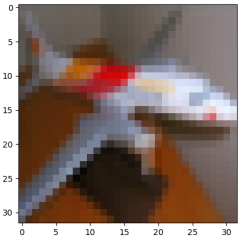
(c) Example 3: Predicted Cat(Bird)



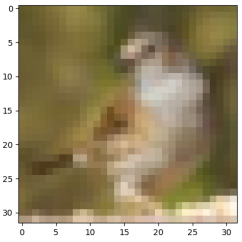
(d) Example 4: Predicted Cat(Bird)



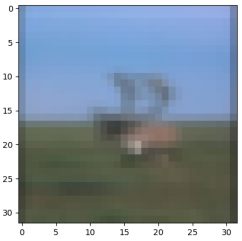
(e) Example 5: Predicted Car(Cat)



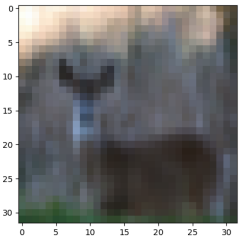
(f) Example 6: Predicted Bird(Aeroplane)



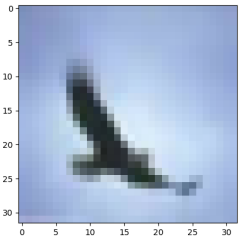
(g) Example 7 :Predicted Bird(Cat)



(h) Example 8: Aeroplane(Deer)



(i) Example 9: Predicted Deer(Cat)



(j) Example 10: Predicted Plane(Bird)

3.4 Cross-Validation and Regularization

The 5-Fold Cross Validation was implemented with Sckit-learn's SVM model and one out of 5 division of training data is taken as Validation set. The Average of Validation set accuracies and test set accuracies are as follows:

C	Validation Accuracy	Test Accuracy
10^{-5}	19.86%	20
10^{-3}	21.1%	27.36%
1	59.74%	60.54%
5	60.39%	61.44
10	61.57%	63.05%

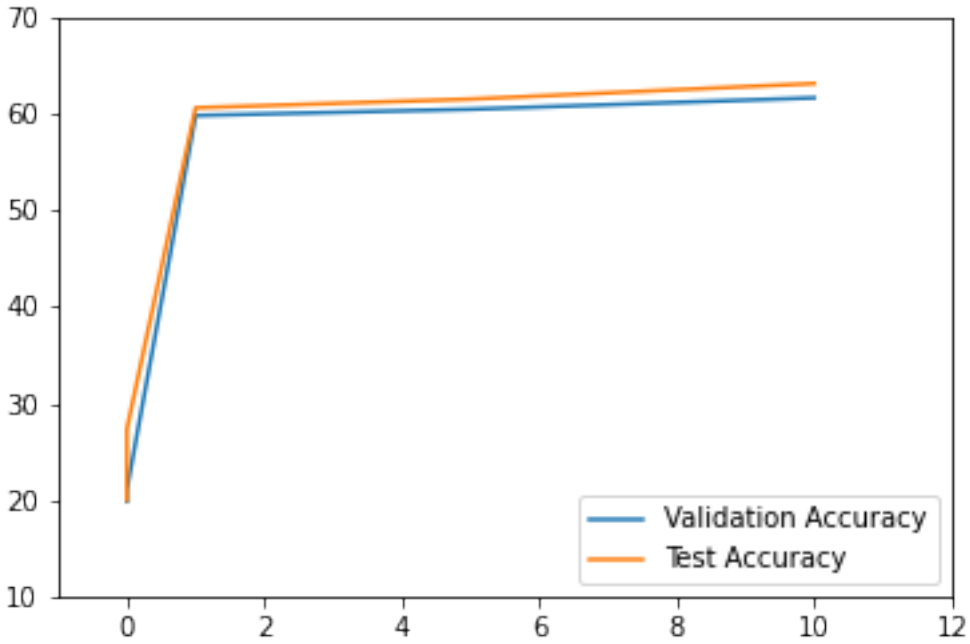


Figure 18: Validation and Test Accuracy Graph

- As seen in the graph, the validation accuracy peaks at $C = 10$, which also gives the highest accuracy in the test data. As a result, we can infer that the cross-validation approach is a reliable estimate for identifying the model's hyperparameters, which may help it perform at its peak during testing.
- With the lower values of C, there is huge relaxation on ξ_i, hence our model underfits on the training data set giving very less accuracy. With the increase in the value of C, there gets tighter on ξ_i learning well on the training data and hence increasing accuarcy. Although, too high value of parameter C leads to overfitting of the model on the training data and hence decreasing the test set accuracy.