EE 559 Project

Classify Forest Cover

05/13/2014

Shankhoneer Chakrovarty

chakrova@usc.edu

Abstract

In this project we have tried to create a classifier to identify different types of forest covers. Different types of classifiers, qdc, ldc, knnc, perlc and svm, were explored and compared in order to solve the problem. 5-fold cross validation was implemented after normalization of data to get the optimal dimension and parameter values. In the end, **svm with rbf kernel** classifier emerged victorious. On the intermediate test set it gave an accuracy of **80.82%** and on final testing data, the accuracy was **81.2857%.**

Baseline Performance Measure

To get the baseline performance measure, *qdc* classifier was used which is provided by prtools. 5 times

5-fold cross validation was performed to estimate the performance on the training set.

Baseline performance data:

Classifier	**Average Error Rate**	**Standard Deviation**
qdc	84.51	5.9

So we see that the baseline performance **error rate** is **84.51±5.9%**. So the **accuracy** is **15.49±5.9%**

Confusion Matrix is presented below:

True Labels	**Estimated Labels**							**Totals**
	1	**2**	**3**	**4**	**5**	**6**	**7**	
1	0	0	204	324	0	0	12	540
2	0	0	113	421	0	0	6	540
3	0	0	3	537	0	0	0	540

4	0	0	1	539	0	0	0	540
5	0	0	27	513	0	0	0	540
6	0	0	0	540	0	0	0	540
7	0	0	410	89	0	0	41	540
Totals	**0**	**0**	**758**	**2963**	**0**	**0**	**59**	**3780**

Comments:

`qdc’ classifier gives really poor result on the given dataset. This could be due to following reasons:

1. Data is not normalized.
2. 54 dimension may be too `complex’ for the classifier to classify fairly accurately, we need to reduce the dimension to see which dimension gives the best performance.
3. Since the distribution of the data is unknown, it might be that quadratic classifier is not powerful enough to do the job, instead, we might need some other statistical classifier.

Appropriate Features

a) **Feature Extraction/Feature Selection:**

No remapping of the given feature set was done. Instead, I relied on normalization and dimension reduction procedures to find the optimum performance.

b) **Normalization/Scaling:**

`normm’ method provided by prtools was used for normalizing the given data. `normm’ method normalizes the distances of all features in the dataset such that their Minkowski-P distances to the origin equal one. Minkowski-P distance between two points

P = (x_1,x_2,…,x_n) and Q = (y_1,y_2,…,y_n) ∈ $\mathbb{R}^{\boldsymbol{n}}$

is defined as:

$$\left(\sum_{i=1}^{n} |x^i - y^i|^p\right)^{1/p}$$

Classification performance of the normalized data using 5-fold cross validation on training data:

Classifier	**Average Error Rate**	**Standard Deviation**
qdc	63.74	17

So we see that the baseline performance error rate is **63.74±17%**. So the accuracy is **36.26±17%**

Confusion Matrix is presented below:

True Labels	**Estimated Labels**							**Totals**
	1	**2**	**3**	**4**	**5**	**6**	**7**	
1	47	0	161	0	90	3	239	540
2	14	0	164	6	196	4	156	540
3	0	0	253	282	5	0	0	540
4	0	0	4	536	0	0	0	540
5	4	0	241	0	277	2	16	540
6	0	0	231	275	7	27	0	540
7	0	0	320	0	9	0	211	540

Totals	**65**	**0**	**1374**	**1099**	**584**	**36**	**622**	**3780**

From the observed classifier performance data, it can be concluded that the accuracy of qdc classifier improves if the data is normalized.

c) Dimensionality Reduction

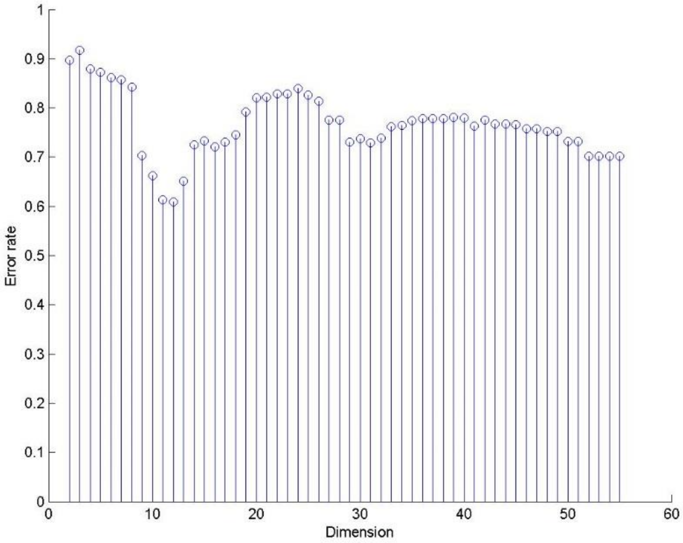
There are more than one option provided by prtools in order to reduce the dimension of the data namely klm and fisherm. The former uses PCA and Fisher's LDA is used by the latter method. For analysis of our data, klm was used.

Qdc classifier was trained using normalized data which was reduced to 11 dimension to get optimal accuracy. Following are the statistics of the qdc classifier on ` testing_set_int_labeled' which was also reduced to 11D

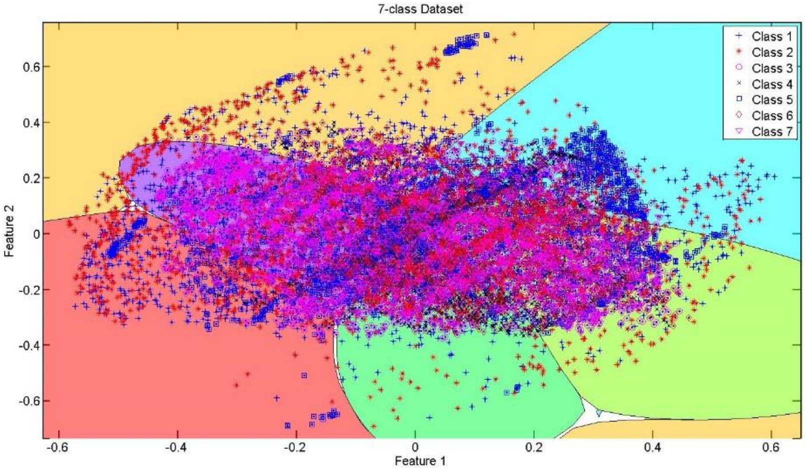
Classifier	**Average Error Rate**
qdc	60.98

So we see that the baseline performance **error rate** is **60.98%**. So the **accuracy** is **39.02%**

Classification performance (accuracy percentage) vs. dimensionality of feature on training data



Scatter plot in feature space for the 2-dimensional case, showing decision boundaries, training data, and testing data



Comparison with baseline performance:

Clearly, reducing the dimension improved our result but only for some specific dimensions. From the statistics above, for the qdc classifier, if the 54D data is reduce to 11D after normalization maximum efficiency could be achieved. The error rate vs the dimension follows kind of a parabolic graph with mouth opening up and minimum at dimension 11.

Comparison of classifiers

Five different classifiers were compared. Each classifier has been described below. All the classifiers were evaluated based on the **error rate** of each classifier on the data.

a) KNN Classifier (k-Nearest Neighbor classifier)

knnc is the method which implements k-NN classifier in prtools.

Setting of the experiment:

i. After normalization of the data (norm method was used with p=1), PCA was used as feature reduction tool and the optimal feature and optimal `k' was selected using 5-fold cross validation (`prcrossval' method) repeated 5 times on each dimension from 1 to 54 and on each `k' from 1 to 15.

ii. After getting the cross validation accuracy, data from testing_set_int_labeled was used to get error rate

iii. The error rate from above step and cross validation accuracy was taken into consideration while choosing the optimal dimension and optimal K.

Although knn-classifier was tested on features from 1 to 54 and k from 1 to 15, it was observed that as feature dimension increased, values of k greater than three gave sub-optimal values, so only a subset of the results are presented here.

The table below shows how the cross validation error rate of knnc on training data for dimension equal to 1 to 54 and k equal to 1 to 3.

Dimension → **K ↓**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	79.46	67.16	51.86	35.94	32.91	30.74	26.22	26.06	26.07
2	80.61	70.12	56.56	41.68	38.58	36.31	30.86	31.21	30.86
3	79.44	66.83	52.90	38.97	36.55	34.99	29.66	29.47	29.65
Dimension → **K ↓**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
1	25.87	26	26	25.98	25.87	26.07	26.08	25.96	26
2	30.99	30.82	30.81	30.86	30.79	30.97	30.83	30.79	31.01
3	29.71	29.45	29.64	29.55	29.59	29.71	29.44	29.63	29.43
Dimension → **K ↓**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
1	25.89	25.91	26.02	25.90	26.06	25.94	26.13	25.92	26.06
2	30.75	30.96	31.01	30.78	30.95	30.91	30.78	30.72	30.86
3	29.59	29.44	29.54	29.59	29.47	29.48	29.54	29.58	29.52
Dimension → **K ↓**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
1	25.92	25.92	25.82	25.98	25.79	25.96	25.75	25.92	26.16
2	30.79	30.97	30.71	31.03	31.05	30.90	30.73	30.91	30.71
3	29.49	29.56	29.46	29.70	29.50	29.37	29.53	29.71	29.51
Dimension → **K ↓**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**
1	26.06	26.08	25.87	25.99	25.85	25.99	25.87	25.85	25.94

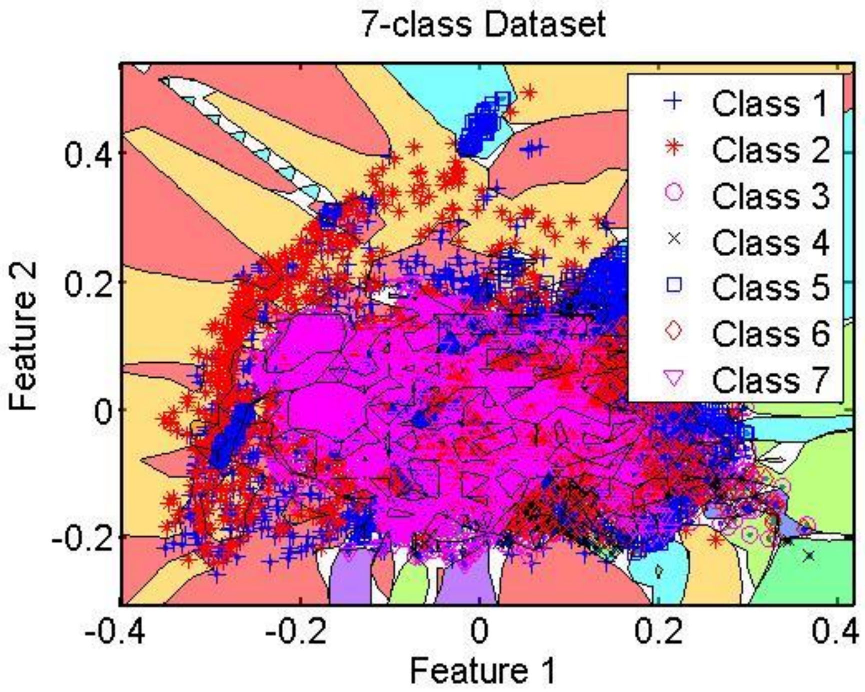
2	30.63	30.86	30.89	30.87	30.76	30.95	31.02	30.66	30.84
3	29.43	29.55	29.50	29.36	29.56	29.72	29.60	29.83	29.62
Dimension →	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**
K ↓									
1	25.98	25.93	25.77	26.01	26.08	26.19	25.84	25.92	25.83
2	30.92	30.90	31.01	30.95	30.73	30.81	30.99	30.73	30.89
3	29.45	29.68	29.56	29.49	29.51	29.62	29.61	29.57	29.72

The minimum error rate in cross validation came out to be **25.75±16%** for dimension 34.

The error rate of the classifier on intermediate testing data reduced to dimension 34 is **81.67%,** so the accuracy **18.33%**

The error rate of the classifier on final testing data reduced to dimension 34 is **76.57%,** so the accuracy is **23.43%**

2D scatter plot of training data with decision region as found by knnc



Even though the cross validation accuracy was pretty decent, following may be the reasons that knnc failed to give as good accuracy in the intermediate and final testing set:

1. One possible reason could be different distribution of data in training and testing set
2. Another reason, as you can see in the 2D scatter plot, could be overfitting

Even though the cross validation accuracy is way better than our baseline classifier, the performance of knnc on intermediate testing dataset and final dataset is totally unimpressive and clearly it performs poorly than our baseline classifier on these fronts.

b) **Perceptron**

perlc is the perceptron classifier in prtools.

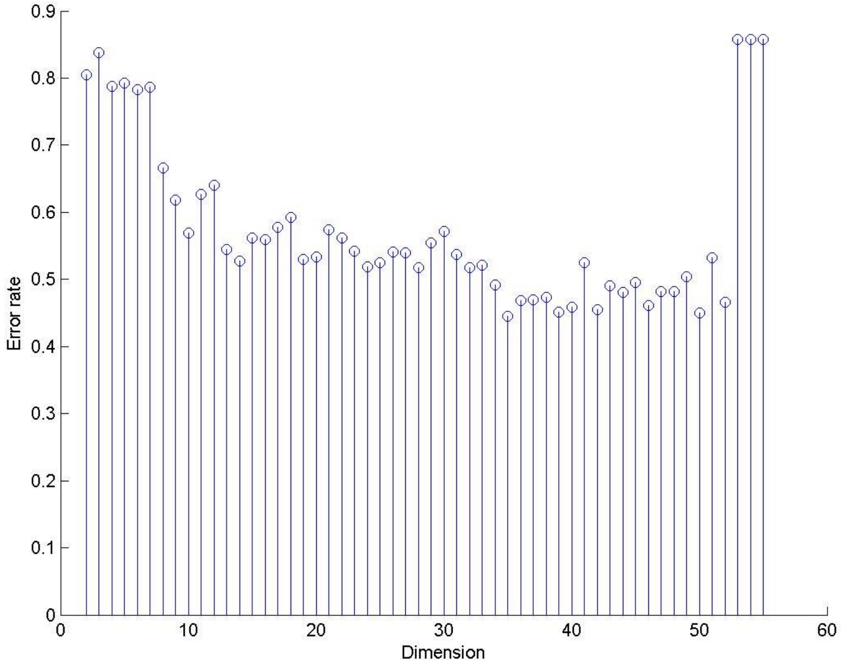
Setting of the experiment:

i. The training data from training_set and testing data from testing_set_int_labeled was extracted and normalized using the norm method (p=1), then PCA was selected as dimension reduction tool.

ii. 5-fold cross validation (`prcrossval' method of prtools) repeated 5 times was used to get the optimal dimension.

iii. For this optimal dimension, error rate on testing data was calculated.

Cross validation accuracy for different dimensions:

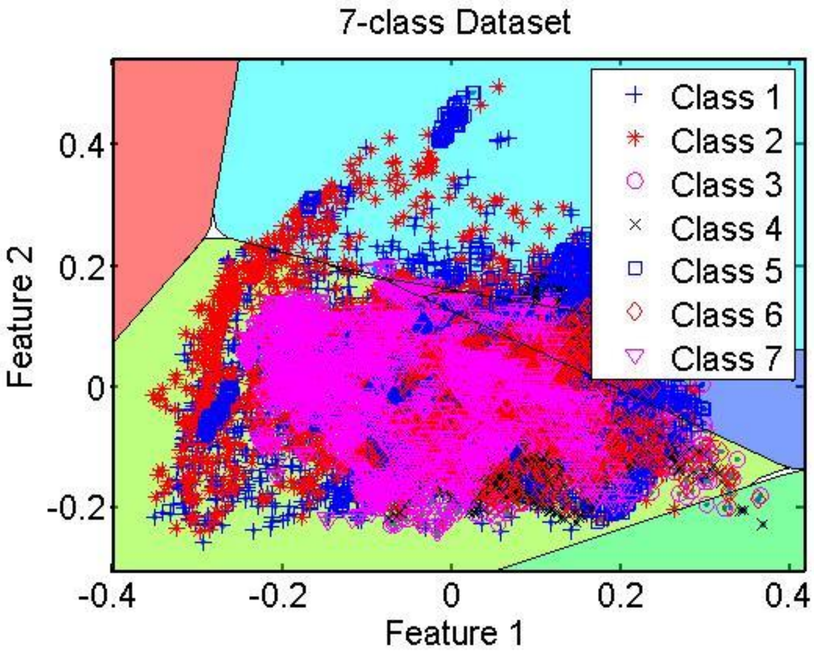


The minimum error rate in cross validation came out to be **44.47±23.21%** for dimension 34.

The error rate of the classifier on intermediate testing data reduced to dimension 34 is **64.31%,** so the accuracy is **35.69%**

The error rate of the classifier on final testing data reduced to dimension 34 is **53.21%,** so the accuracy is **46.79%**

2D scatter plot of training data with decision region as found by perceptron: (Note the white regions in the graph which shows indeterminate regions)



The reasons why perlc did not turn out to be the best classifier because:

1. Perceptron has proved to be very successful to classify linearly separable data. The graph above clearly shows that the data in 2D is not linearly separable.
2. Perceptron classifiers give rise to indeterminate region (white regions in the graph) which affects the efficiency.

Perceptron, though simpler, beats accuracy of our baseline classifier on training and final test dataset and performance of perceptron is comparable to qdc on intermediate test dataset.

c) ldc or LDC

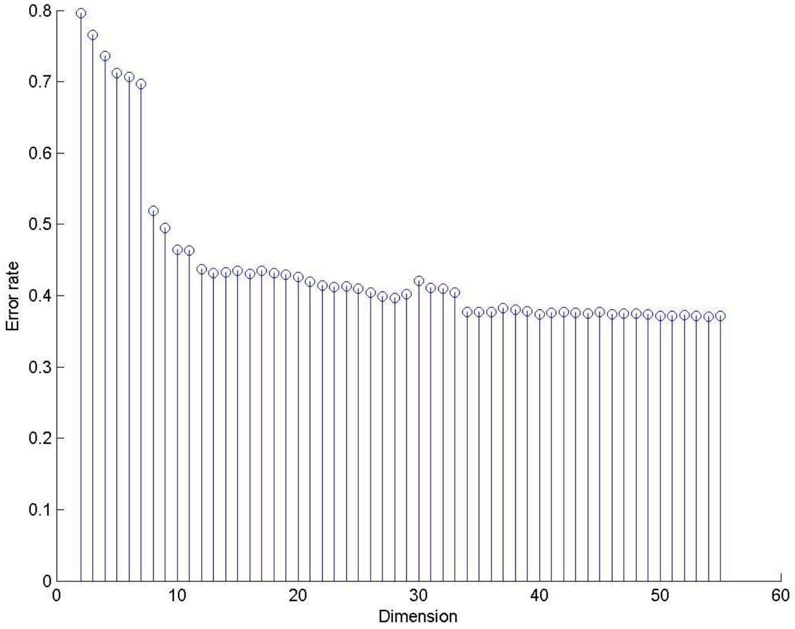
Setting of the experiment:

i. Similar to the above two methods, training data and testing data from testing_set_int_labeled was read and normalized using the norm method (p=1). PCA was used as a dimension reduction tool for this case too.

ii. 5-fold cross validation (`prcrossval' method) repeated 5 times was used to get the optimal dimension

iii. Then, for the optimal dimension, error rate was calculated on testing data.

Cross validation accuracy for different dimensions:

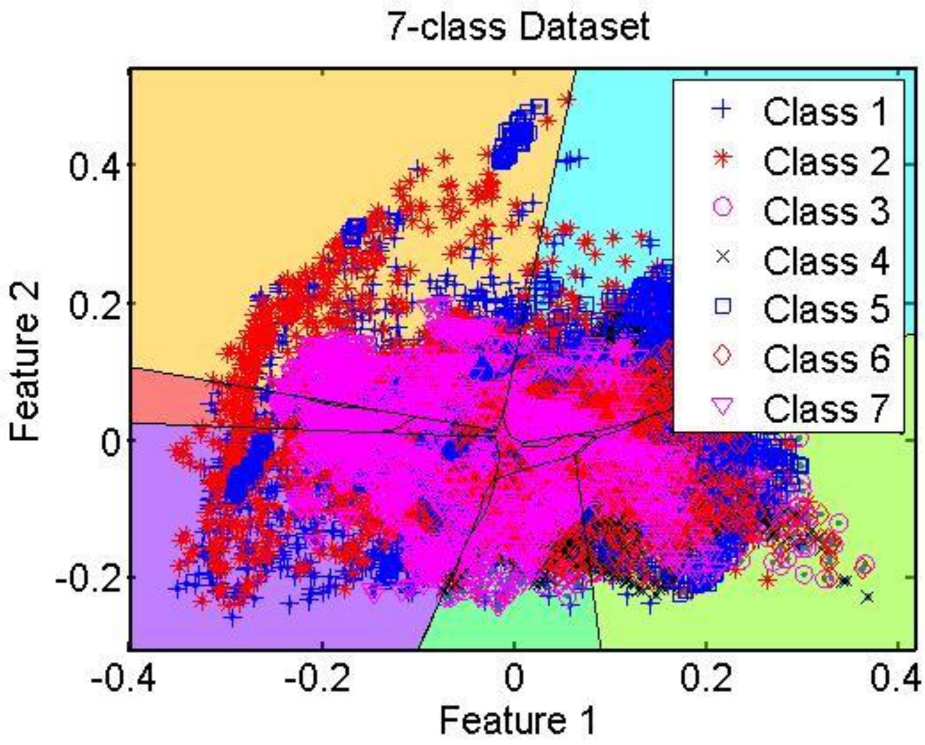


Minimum cross validation error rate of **37.10±11.53%** was achieved at dimension number 53

Corresponding error rate on intermediate testing data is **46.11%.** So the accuracy was **53.89%** after reducing the data to 53D

Error rate on final testing data is **50.82%.** So the accuracy is **49.18%.** In this case also the dimension was reduced to 53.

2D scatter plot of training data with decision boundaries:



Comment on the behavior of ldc based on above graph:

The error rate of ldc on 2D data is 91.59% which makes ldc useless in 2D for this type of data. The reasons for such poor performance by ldc can be:

1. It's a linear classifier, from the graph it is pretty clear that in 2D, the data is not linearly separable so poor performance of ldc is expected.
2. Also, reducing dimension from 54 to 2 using PCA, results in lots of lost information which makes it even harder for classifiers.

Overall, LDC beats the accuracy of baseline classifier by huge margin. Clearly LDC captures the underlying statistics of the data and so it's able to perform better than QDC.

d) SVM

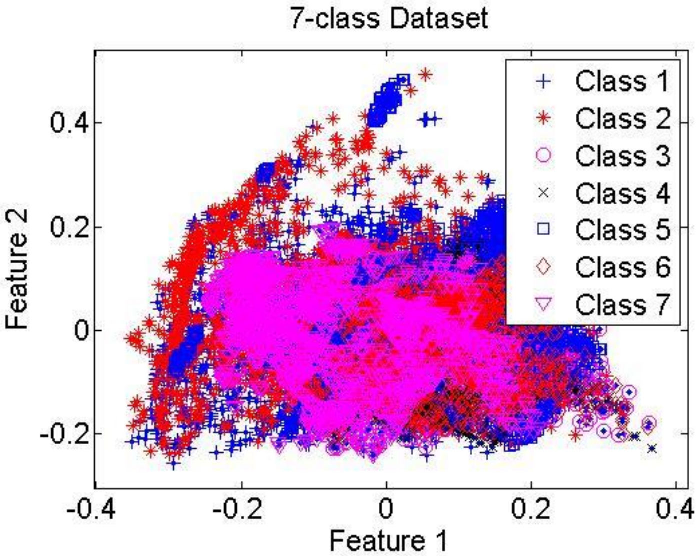
Libsvm was used to train and test svm classifier. Svmtrain and svmpredict are the commands that can be used to train and test respectively. Since linear and Gaussian svm training and testing methods are very similar, the experiment to get optimal parameters and dimension was also very similar:

i. Like all the other methods, training data and testing data from testing_set_int_labeled was read and normalized using the norm method (p=1). PCA was used as a dimension reduction tool for this case too.

ii. 5-fold cross validation (using `-v' switch in svmtrain) repeated 5 times was used to get the optimal dimension and optimal parameters, `c' in case of linear svm and `c' and `gamma' in case of Gaussian svm. One thing to note here is unlike cross validation of all the other classifiers, cross validation in svm requires three loops, one which varies dimensions and another two loops which vary `c' and `gamma'. In case of linear svm we don't need `gamma'.

iii. Then, for the optimal parameters and dimension, error rate was calculated on testing data.

2D scatter plot of the training data:



Since SVM is one vs rest classifier drawing decision region in one diagram is not possible, we need 7 plots to show 7 one vs rest decision regions which will greatly increase the length of the report.

a. Linear SVM

To get the optimal parameters, dimension was varied from 1 to 54 and `c’ was varied as a power of 2 from 2 to 512. This makes total of 54*9=486 data points on which cross validation accuracy was measured. Allowed space is not enough to present all the results.

In the given setting, maximum accuracy of Linear SVM = 58.2% optimal dimension=24 and optimal c=1024

Accuracy of linear svm on intermediate testing dataset reduced to 24 dimensions = 44.9735%

Accuracy of linear svm on final testing dataset reduced to 24D = 45.5357%

Interesting observation:

1. If there is no dimension reduction of the normalized data, linear svm gives better accuracy.
2. Also, the accuracy of unreduced but normalized data keeps on increasing with increasing `c’
3. The best accuracy achieved was 66.2% for c = 32768 which might be due to overfitting so testing data was used to get which c was better and it was observed that after c=8192 (accuracy = 65%) the rate of increase of efficiency settles down to a constant value. So we chose c=8192 as optimal c value for this case

b. Gaussian SVM (Best classifier)

In Gaussian case, to get the optimal parameters, dimension was varied from 1 to 54, `c’ was varied as a power of 2 from 2 to 1024 (total 10) and `gamma’ was varied as a power of 2 from 0.03125 to 4 (total 8). So total of 54*10*8=4320 data points were used to calculate the optimal parameters and dimension. Due to lack of space only the optimum accuracy and parameters are given below:

Maximum accuracy of Gaussian SVM is 66.7% optimal dimension = 10 optimal c = 512 and optimal gamma = 4

Accuracy of gaussian svm on intermediate testing dataset reduced to 10 dimensions = 44.4180%

Accuracy of gaussian svm on final testing dataset reduced to 10D = 48.3214%

Interesting observation:

1. Similar to the kind of observation made in case of linear svm, in the case of Gaussian svm too it was noticed that the classifier performs better on unreduced but normalized data
2. So, `c' and `gamma' was varied to get the optimum accuracy and it was observed in this case too with increasing `c' and `gamma' the performance of the classifier improves
3. Testing data from intermediate testing dataset was used to get the optimum `c' and `gamma'. With increasing `c' and `gamma' there came a point where the accuracy on testing data started to drop. This point gave the value of optimum `c' and `gamma'. **This becomes our best performing classifier.**

Accuracy of Gaussian svm on intermediate testing dataset: 80.82% (c = 8192 and gamma = 16)

Accuracy of Gaussian svm on final testing dataset: 81.2857 %(c = 8192 and gamma = 16)

Confusion matrix of the classifier on normalized but unreduced intermediate testing data:

True Labels	**Estimated Labels**							**Totals**
	1	**2**	**3**	**4**	**5**	**6**	**7**	
1	366	91	1	0	15	1	66	540
2	127	304	24	0	59	20	6	540
3	0	5	390	32	14	99	0	540
4	0	0	13	514	1	12	0	540
5	0	24	8	0	502	6	0	540

6	0	6	45	20	10	459	0	540
7	18	2	0	0	0	0	520	540
Totals	**511**	**432**	**481**	**566**	**601**	**597**	**592**	**3780**

Confusion matrix of the classifier on normalized but unreduced final testing data:

True Labels	**Estimated Labels**							**Totals**
	1	**2**	**3**	**4**	**5**	**6**	**7**	
1	270	60	0	0	18	1	51	400
2	93	240	7	0	45	11	4	400
3	0	4	301	20	13	62	0	400
4	0	0	7	383	0	10	0	400
5	3	14	10	0	367	6	0	400
6	0	5	42	10	8	335	0	400
7	18	2	0	0	0	0	380	400
Totals	**384**	**325**	**367**	**413**	**451**	**425**	**435**	**2800**

Even though parameter selection and optimization methods in both types of svm inspects only a small subset of value of `c’ and `gamma’ due to limitation of computational resources, still, both linear and Gaussian SVM beats accuracy of our baseline classifier. This is due to the nature of svm where it maps the data to higher dimensional and then try to find the decision regions.