Kernel Method for Pattern Analysis

CS6011

IIT Madras

Assignment 2

By:
Group 6:
Arun Baby (CS15S016)
Vishal Subbiah (MM12B035)
Nikhil Thomas Stephen (CS15M032)

Contents

1 Task 1: Classification using SVMs

Classification is a general process related to categorization, the process in which ideas and objects are recognized, differentiated and understood.

1.1 Datasets

1.1.1 Linearly separable data

The dataset is a 2 dimensional linearly separable data. It has 3 classes of 500 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.1.2 Nonlinearly separable data

The dataset is a 2 dimensional nonlinearly separable data. It has 3 classes of 300, 600 and 900 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.1.3 Overlapping data

The dataset is a 2 dimensional overlapping separable data. It has 4 classes of 500 samples each. We used 50% for training, 30% for validation and 20% for testing.

1.2 Models and Kernels

In this task we use two models

- C - SVM
- ν - SVM

And two kernels

- Polynomial
- Gaussian

1.3 Linearly separable classes

1.3.1 Inferences

- We notice that as we vary the parameters the error for the kernel gram matrix varies in the form of an upwards parabola.
- For polynomial kernel we did not need a degree greater than 2 to achieve high accuracy.
- As we decrease C and ν we see the number of support vectors decreases.
- By varying the non constant term we see the error drops initially and then remains constant

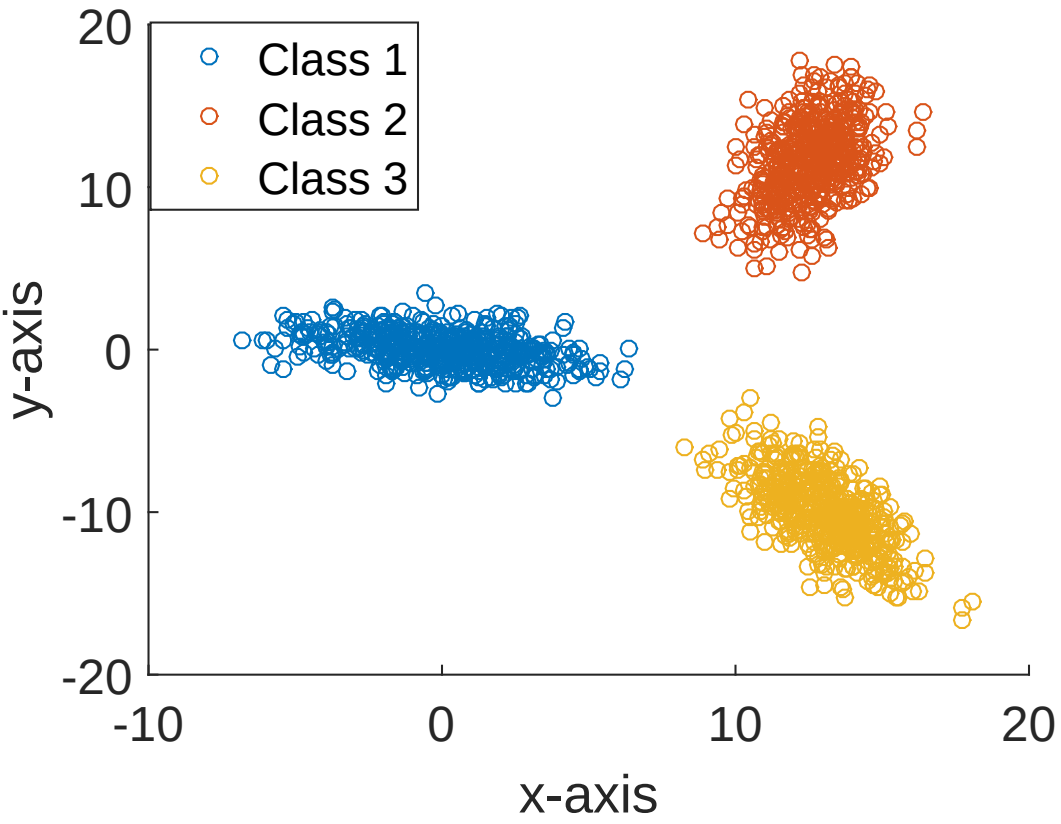
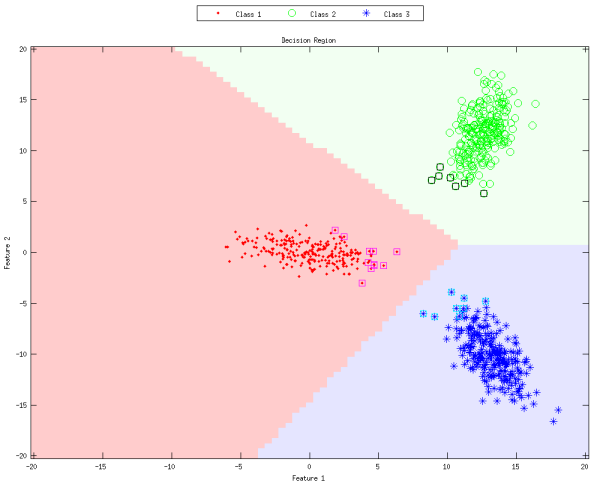
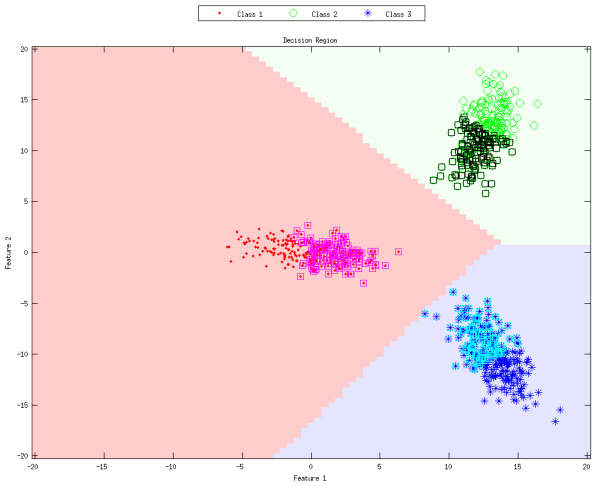


Figure 1: Scatter Plot

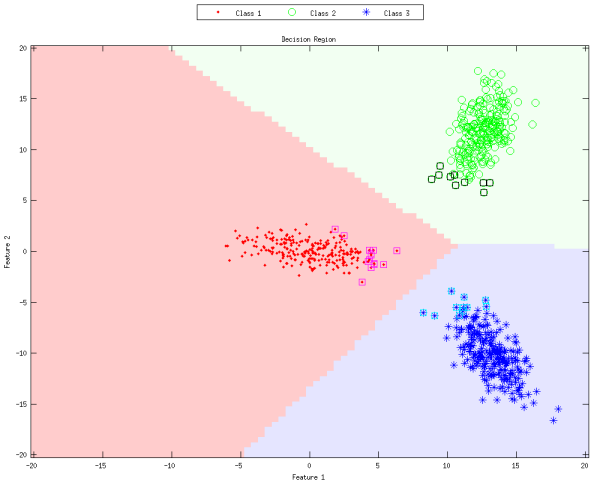
1.3.2 Decision Boundaries



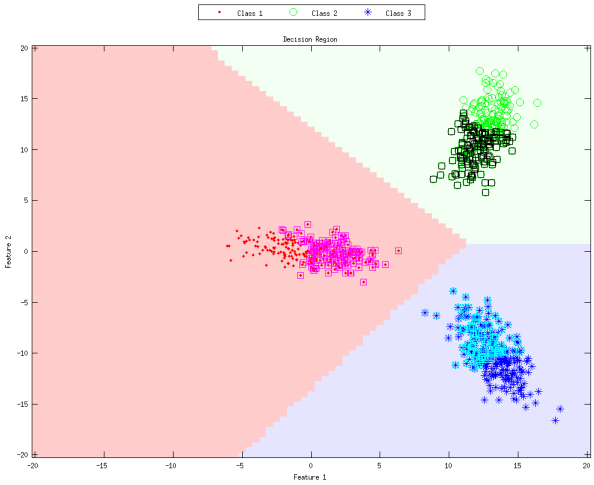
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



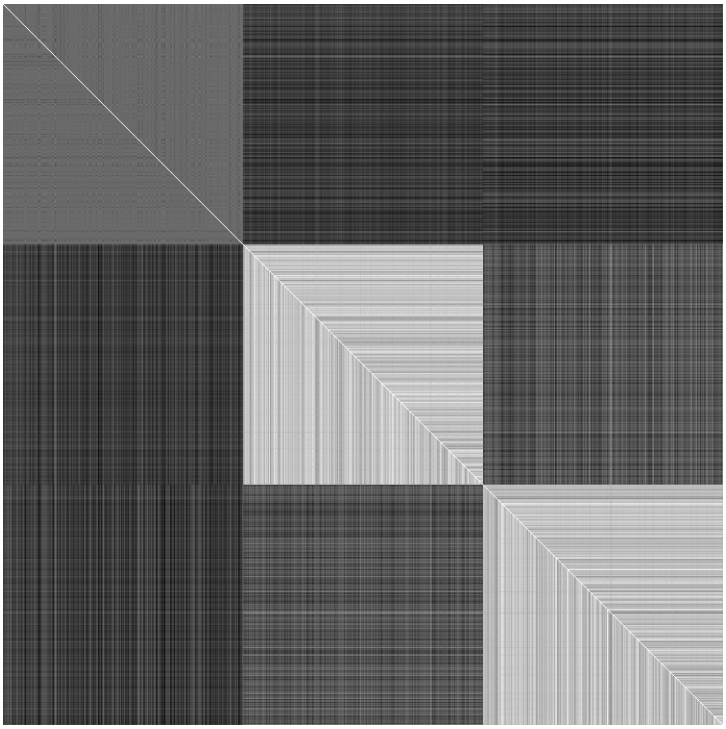
(c) C - SVM using gaussian Kernel



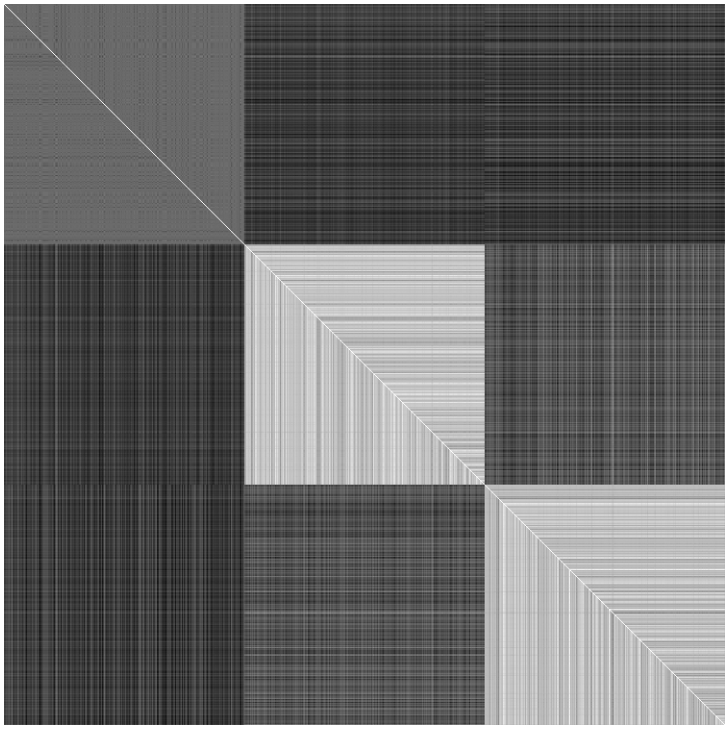
(d) ν - SVM using gaussian Kernel

Figure 2: Decision boundaries

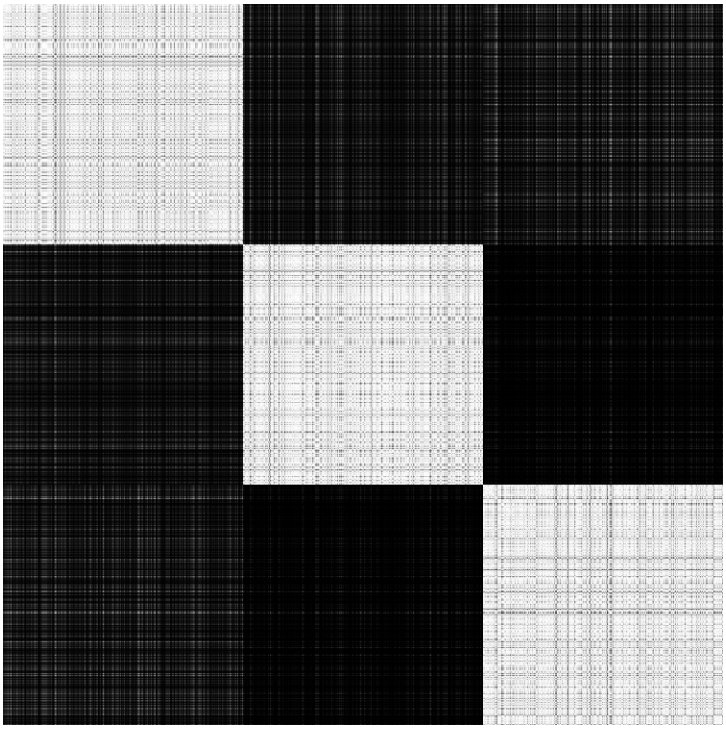
1.3.3 Kernel Gram Matrix



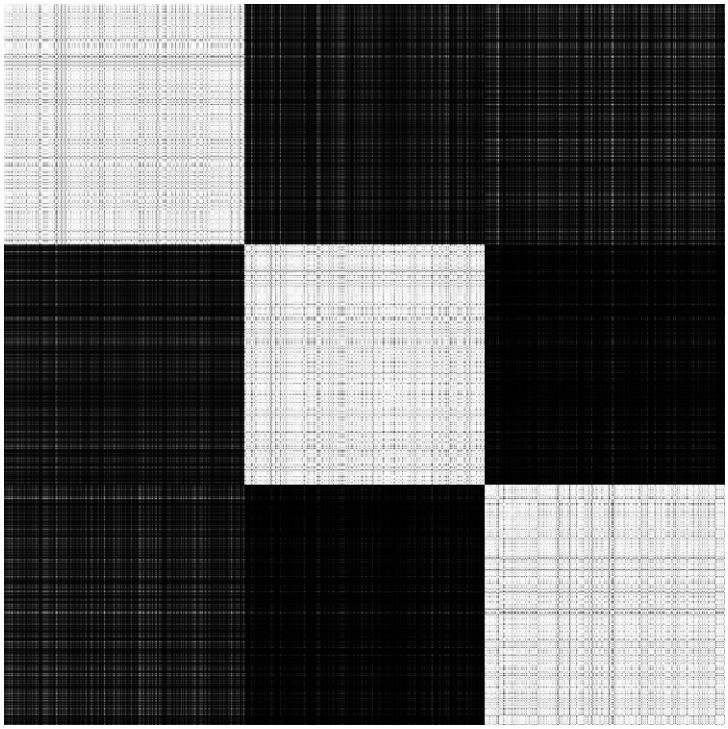
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



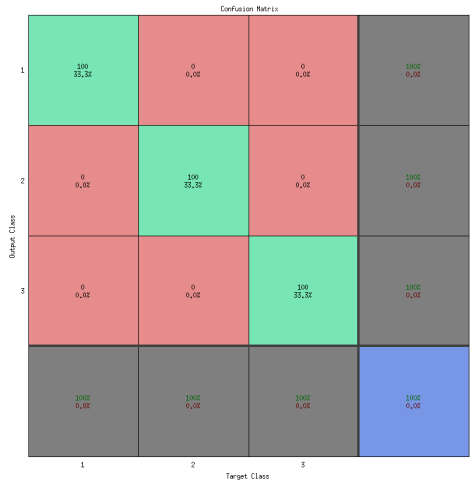
(c) C - SVM using gaussian Kernel



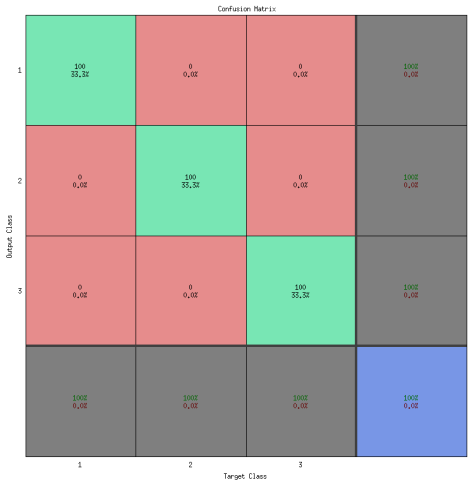
(d) ν - SVM using gaussian Kernel

Figure 3: Kernel gram matrix

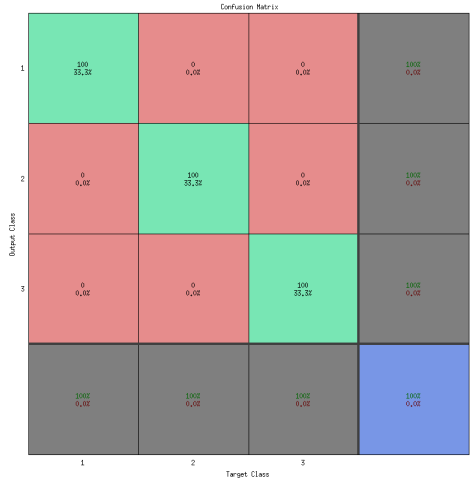
1.3.4 Confusion Matrix



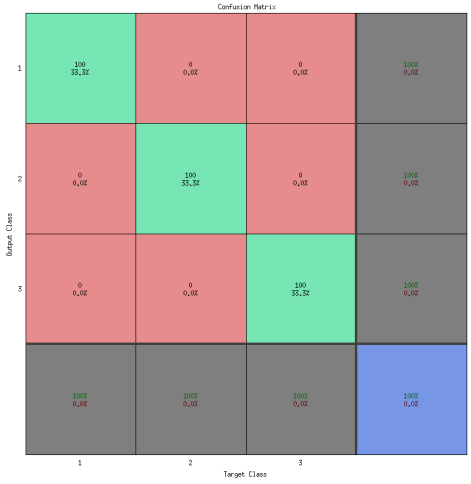
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



(c) C - SVM using gaussian Kernel



(d) ν - SVM using gaussian Kernel

Figure 4: Kernel Gram Matrix

1.4 Nonlinearly separable classes

1.4.1 Inferences

- We notice that as we vary the parameters the error for the kernel gram matrix varies in the form of an upwards parabola.
- For polynomial kernel we did not need a degree greater than 2 to achieve high accuracy.
- As we decrease C and ν we see the number of support vectors decreases.

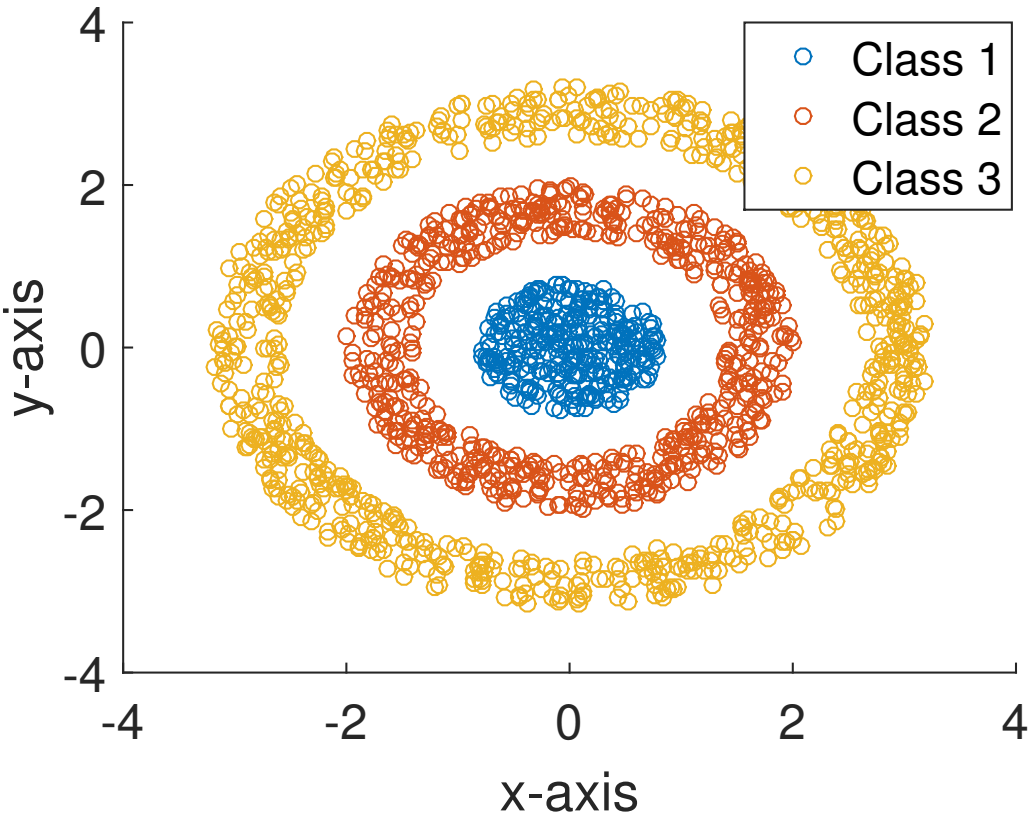
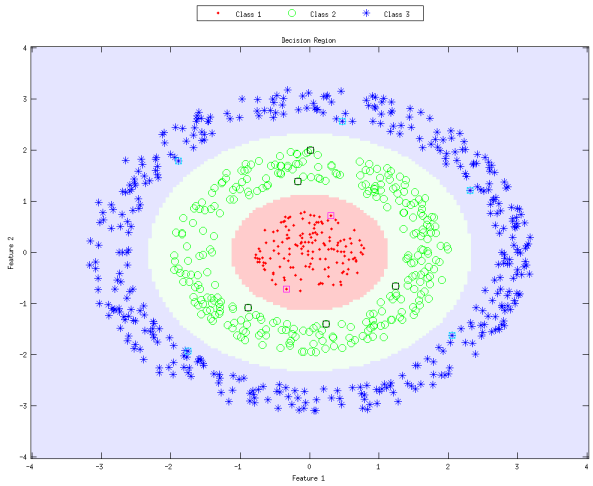
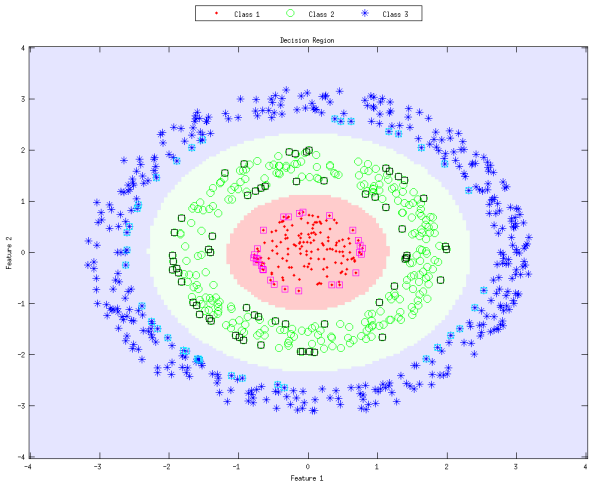


Figure 5: Scatter Plot

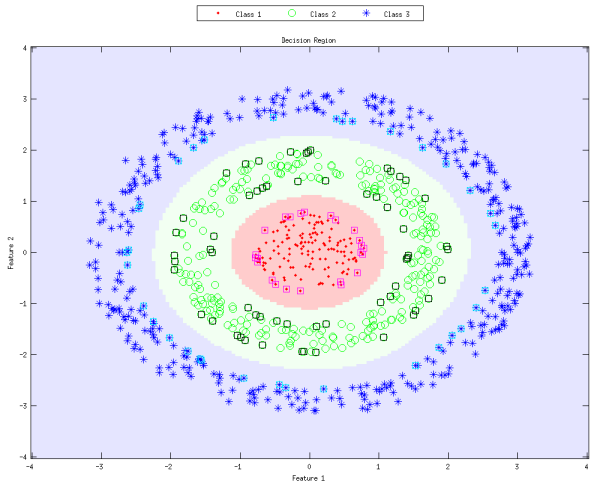
1.4.2 Decision Boundaries



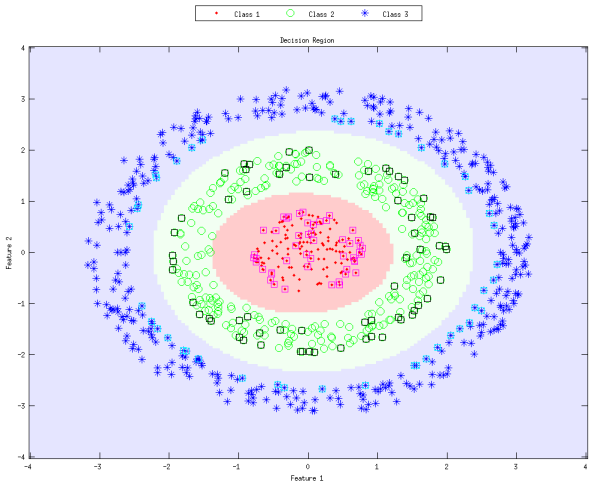
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



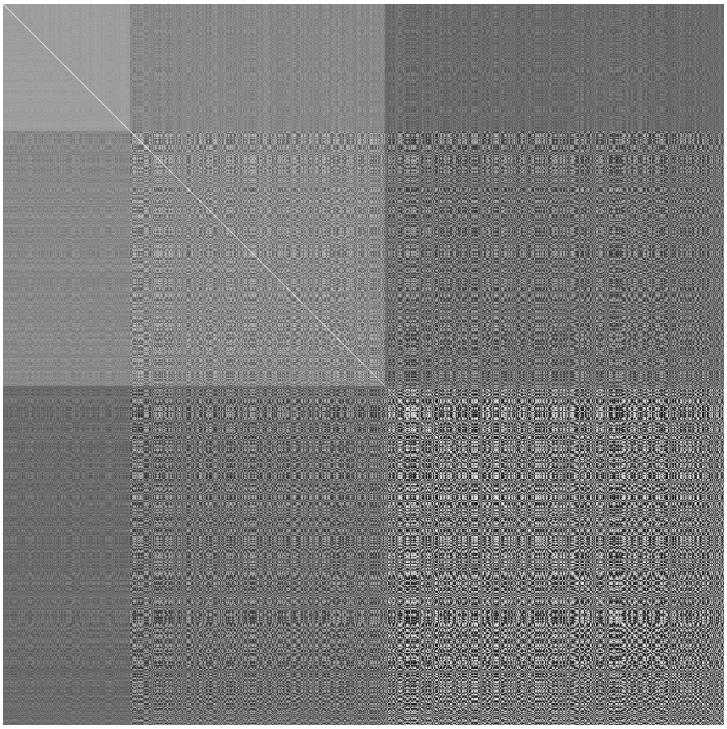
(c) C - SVM using gaussian Kernel



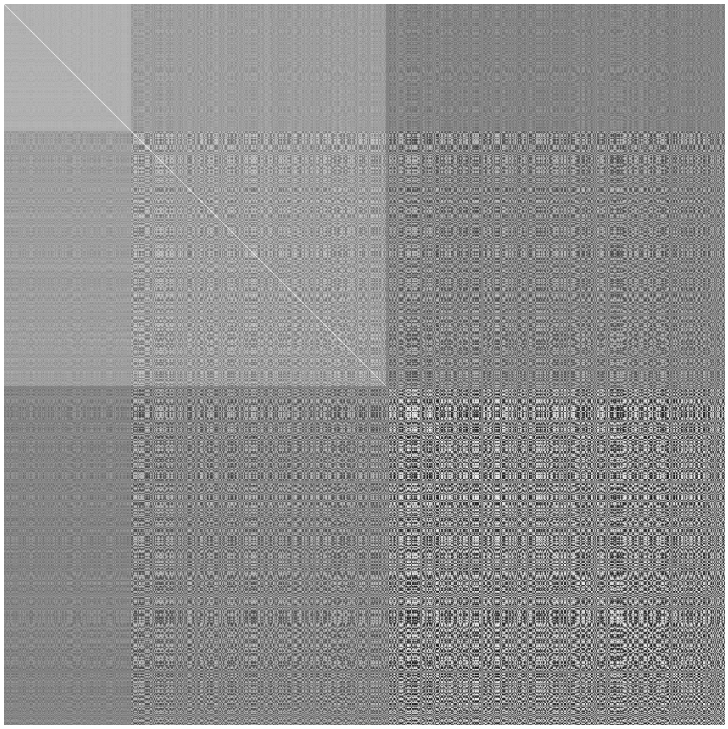
(d) ν - SVM using gaussian Kernel

Figure 6: Decision boundaries

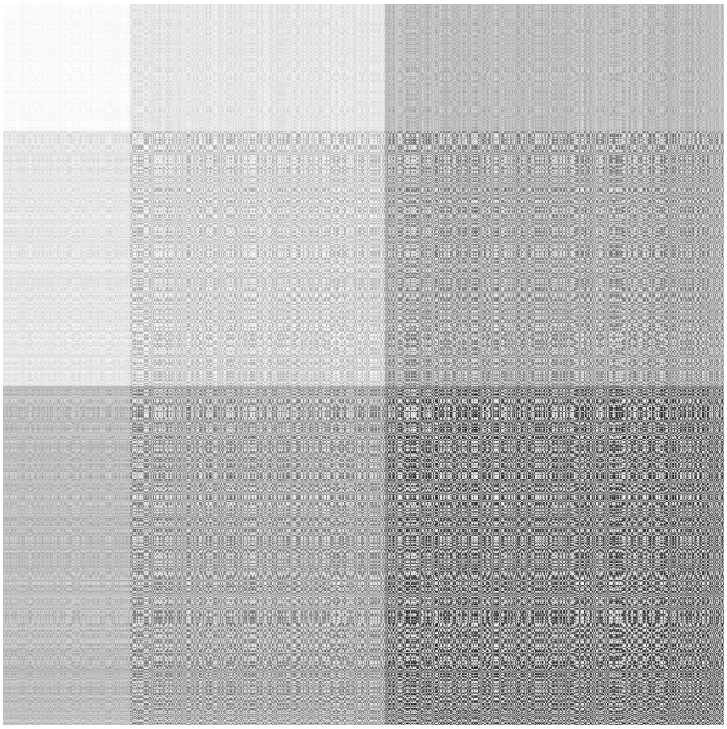
1.4.3 Kernel Gram Matrix



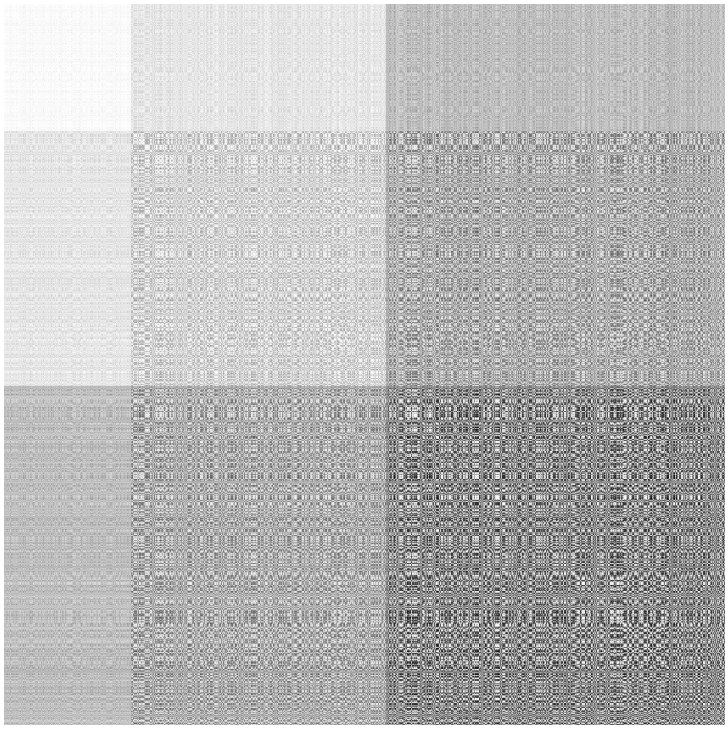
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



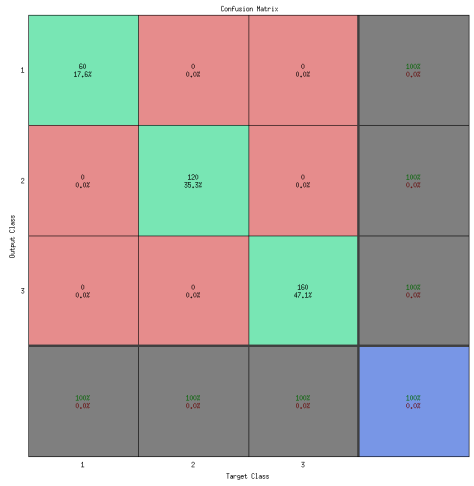
(c) C - SVM using gaussian Kernel



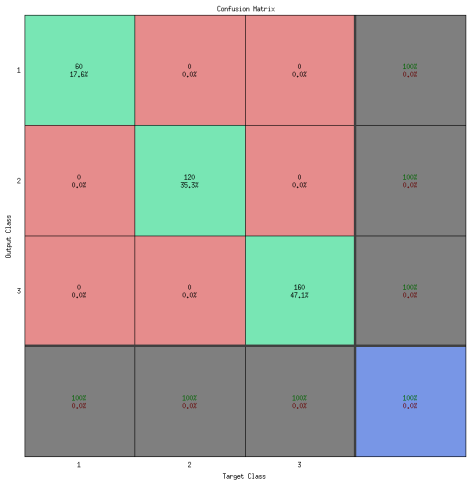
(d) ν - SVM using gaussian Kernel

Figure 7: Kernel Gram Matrix

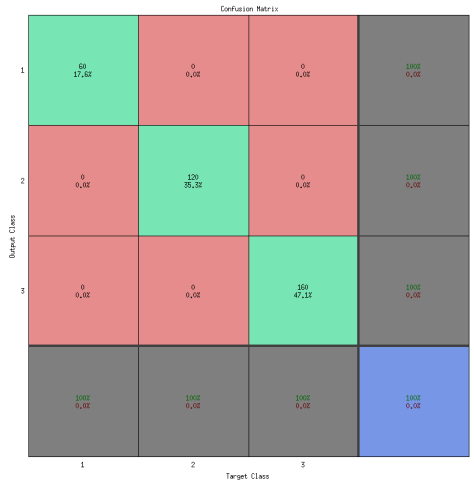
1.4.4 Confusion Matrix



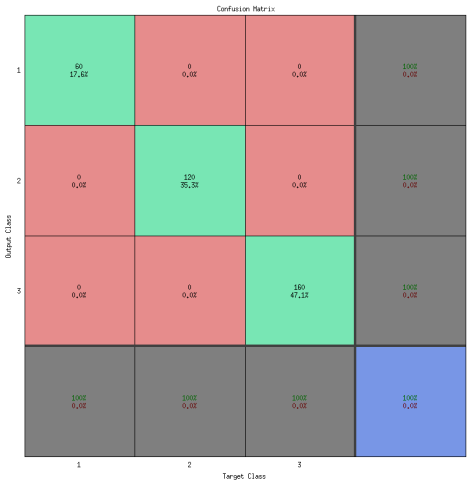
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



(c) C - SVM using gaussian Kernel



(d) ν - SVM using gaussian Kernel

Figure 8: Confusion Matrix

1.5 Overlapping classes

1.5.1 Inferences

- We notice that as we vary the parameters the error for the kernel gram matrix varies in the form of an upwards parabola.
- For polynomial kernel we did not need a degree greater than 2 to achieve high accuracy.
- As we decrease C and ν we see the number of support vectors decreases.

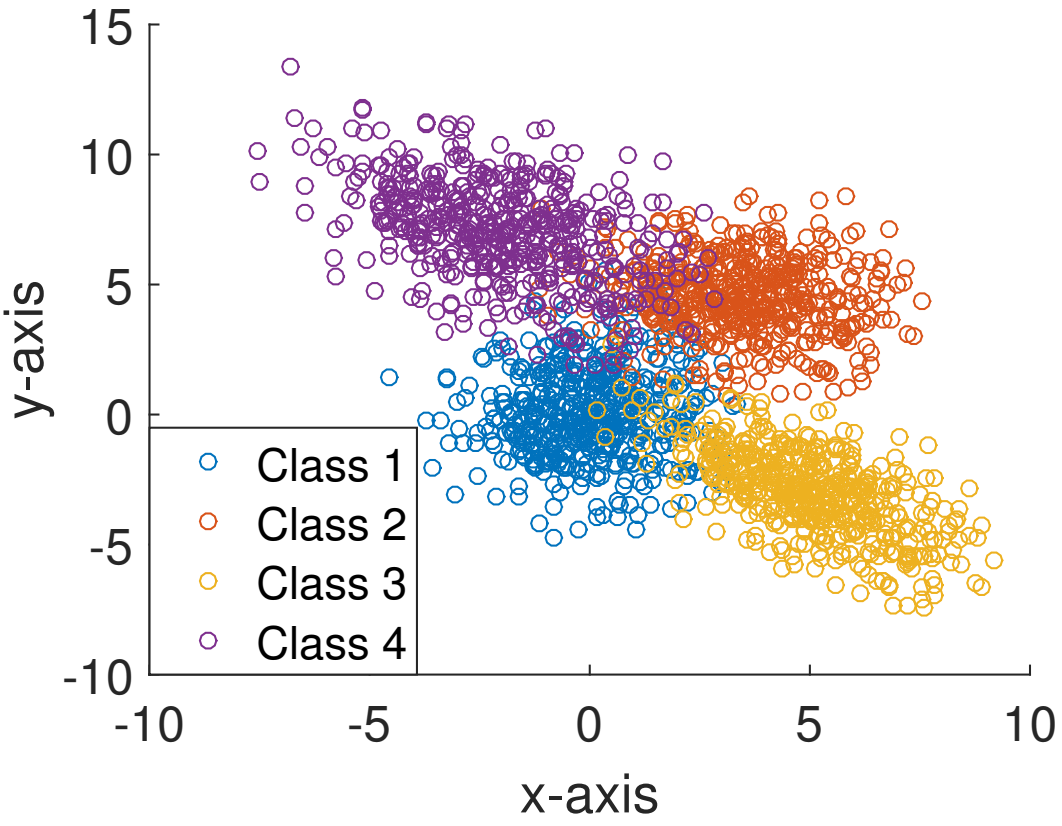
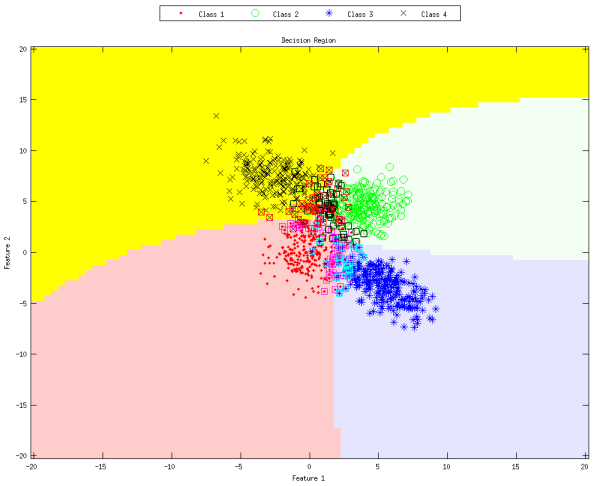
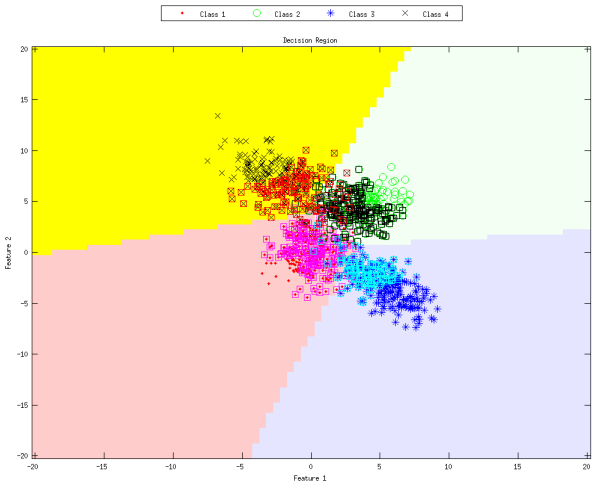


Figure 9: Scatter Plot

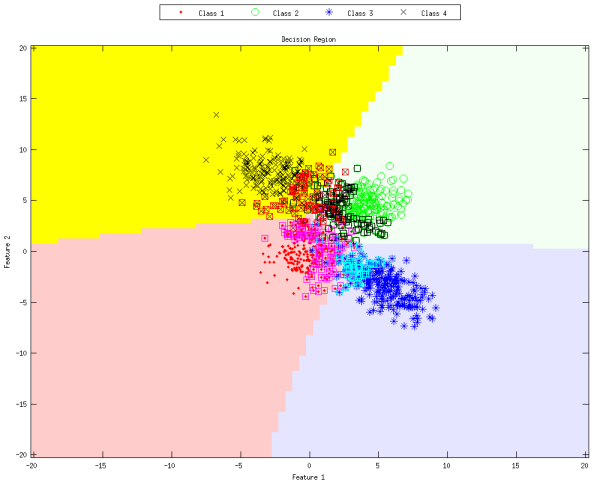
1.5.2 Decision Boundaries



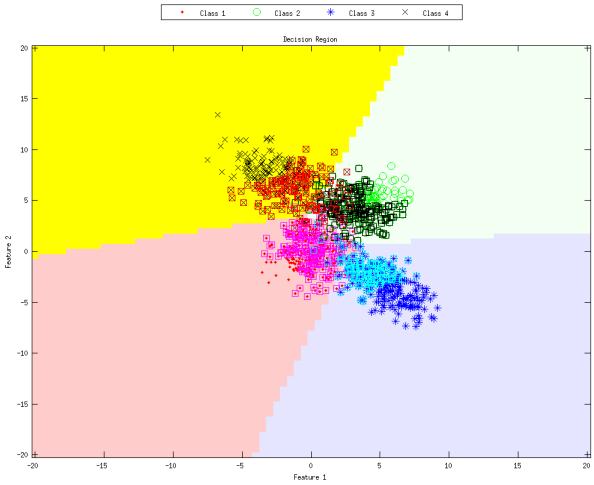
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



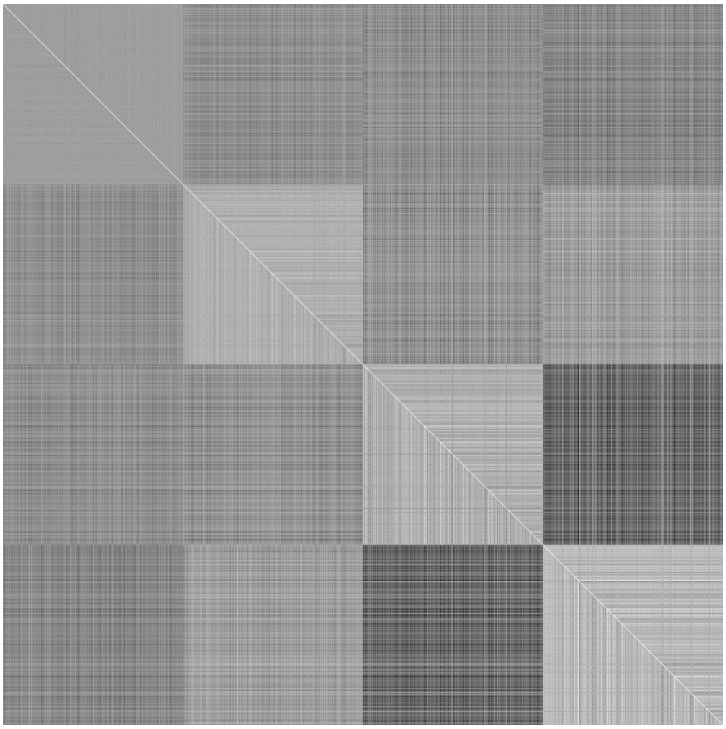
(c) C - SVM using gaussian Kernel



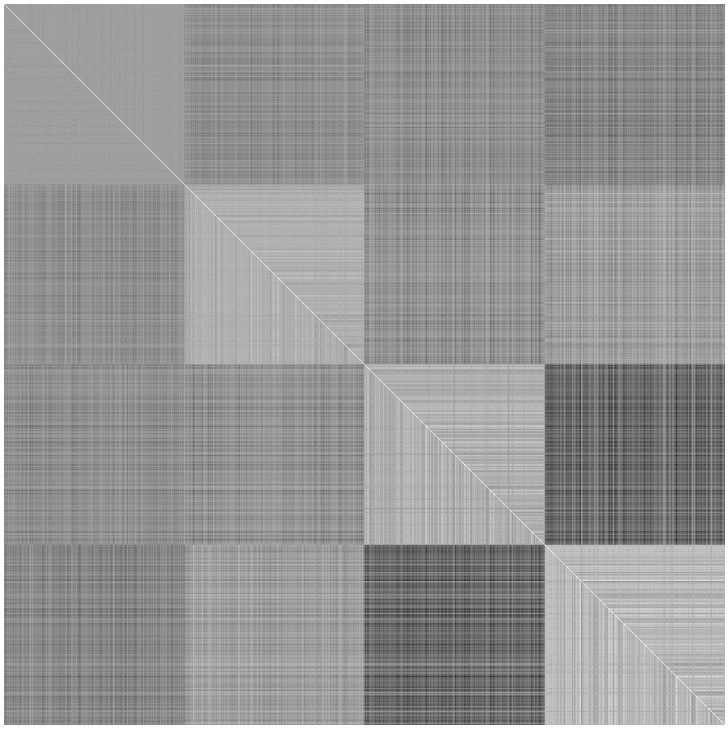
(d) ν - SVM using gaussian Kernel

Figure 10: Decision boundaries

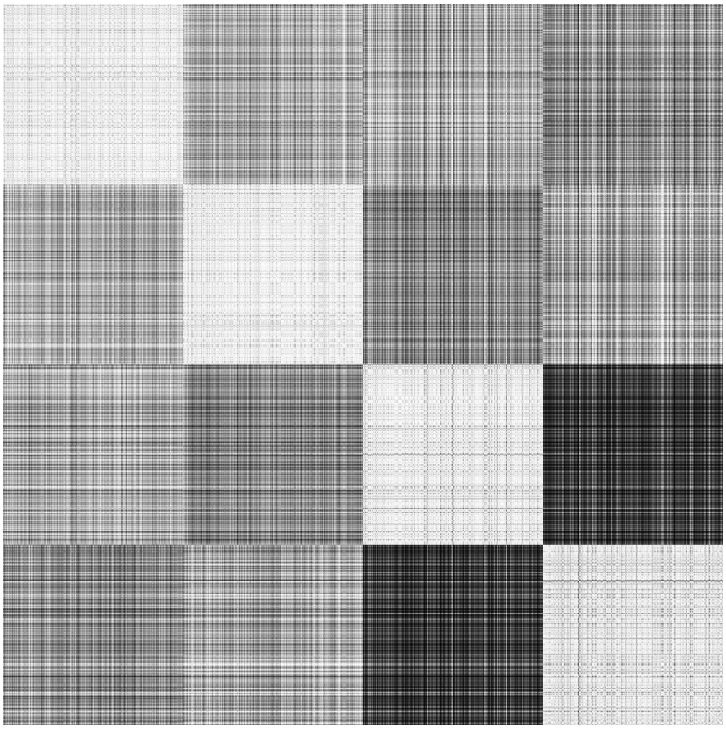
1.5.3 Kernel Gram Matrix



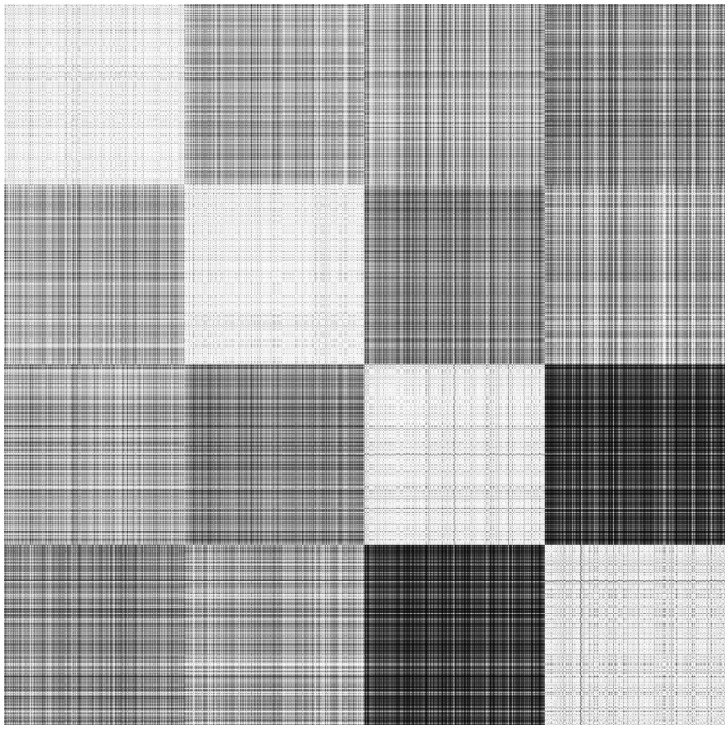
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



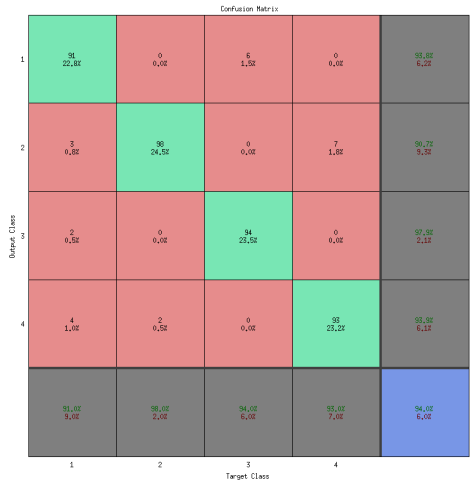
(c) C - SVM using gaussian Kernel



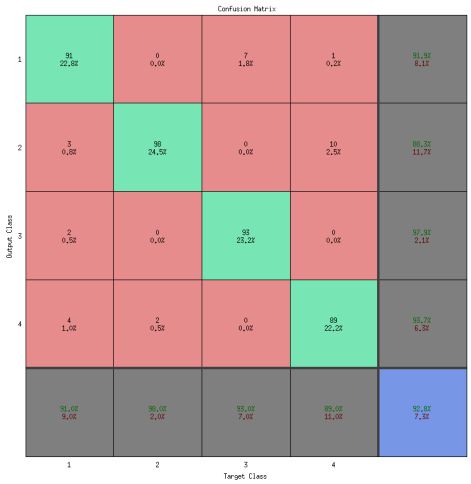
(d) ν - SVM using gaussian Kernel

Figure 11: Kernel Gram Matrix

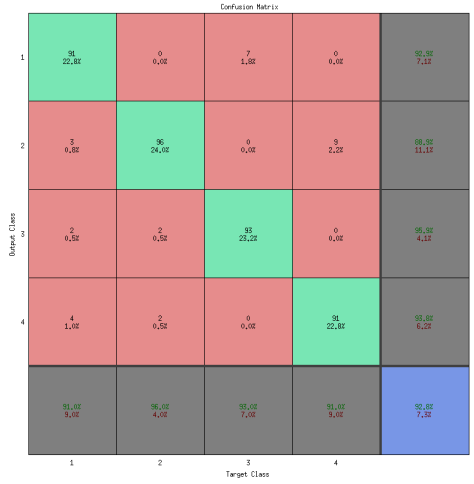
1.5.4 Confusion Matrix



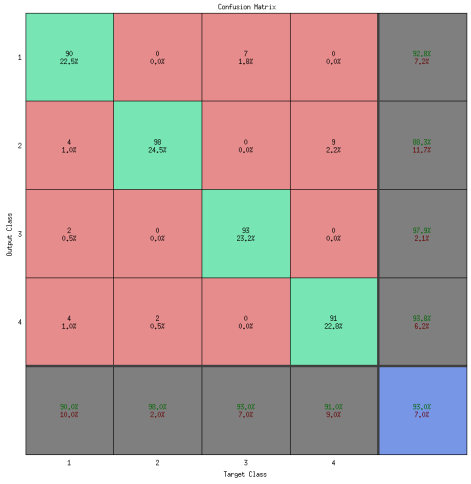
(a) C - SVM using polynomial Kernel



(b) ν - SVM using polynomial Kernel



(c) C - SVM using gaussian Kernel



(d) ν - SVM using gaussian Kernel

Figure 12: Confusionn Matrix

2 Dimension reduction using PCA, Autoencoder and Stacked Autoencoder

2.1 Experiments

- First Classification with Dimension Reduction was done and accuracy was noted
- Dimensions were reduced by using PCA to reduce dimensions to 58,25 and 10

first hidden layer	second hidden layer	third hidden layer	Accuracy
600	400	600	52.5
600	150	600	56.1
600	100	600	54.2
600	50	600	48.2
800	100	800	50.5
800	150	800	56.8
800	50	800	51.5

Table 1: Autoencoder

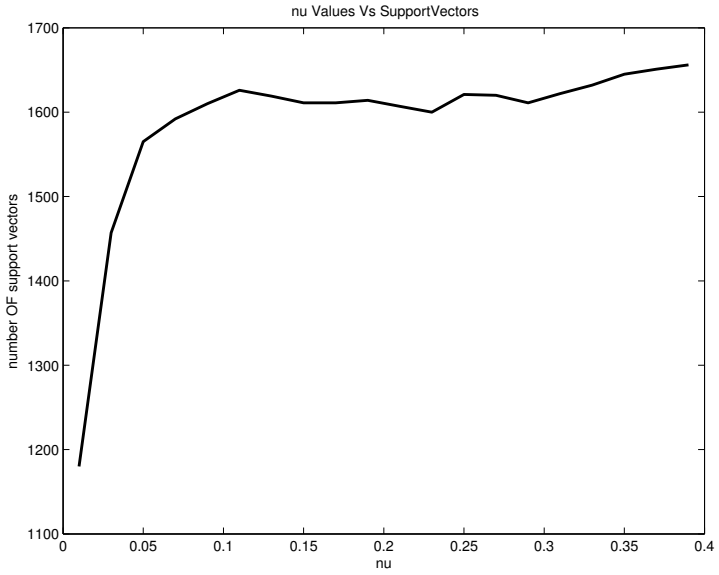
hidden layer configuration	Accuracy
512-600-400-500-275-300-150	44.5
512-800-400-600-275-300-150	43.2
512-600-400-500-200-300-100	36.2
512-800-400-600-200-400-100	48.2

Table 2: Stacked autoencoder

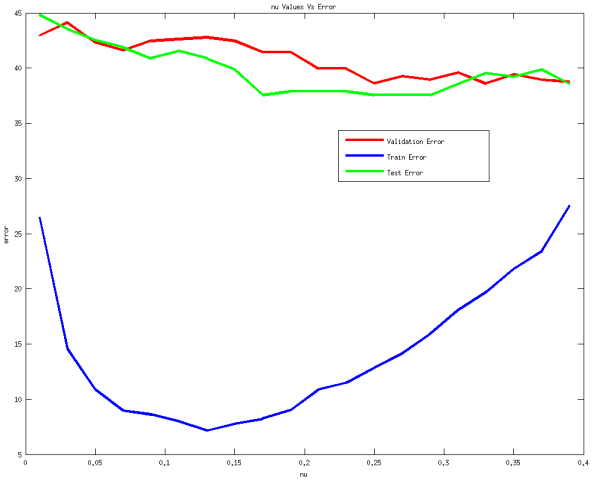
2.2 Observation and Inferences

- Without PCA accuracy obtained was 60.8 and $/nu$ value = 0.25
- With PCA when top 58 eigen values were taken accuarcy obtained was 60.1
- When top 25 eigen values were taken accuracy obtained was 59.5.
- When top 25 eigen values were taken accuracy obtained was 49.5
- Even though the dimesnions was reduced as less as 25 accuracy did not lower much. With 25 dimensions itself it was able to capture the variations in the data.
- For AutoEncoder the best accuracy was obtained for bottleneck features of 150 and accuracy obtained was 56.8
- For Stacked AutoEncoder the maximum accuracy was for the configuration 512-800-400-600-200-400-100.
- Variation of SVs in dimension reduction of PCA follows a curve seen usually in stress strain plots of metals. It is of the form $\sigma = k\epsilon^n$

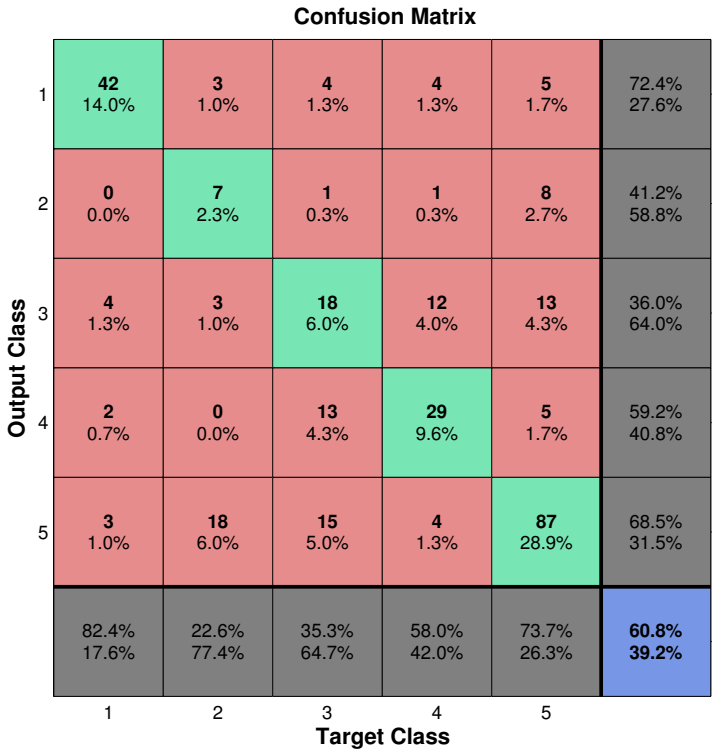
2.3 Plots



(a) Variation of number of SVs with respect to ν values

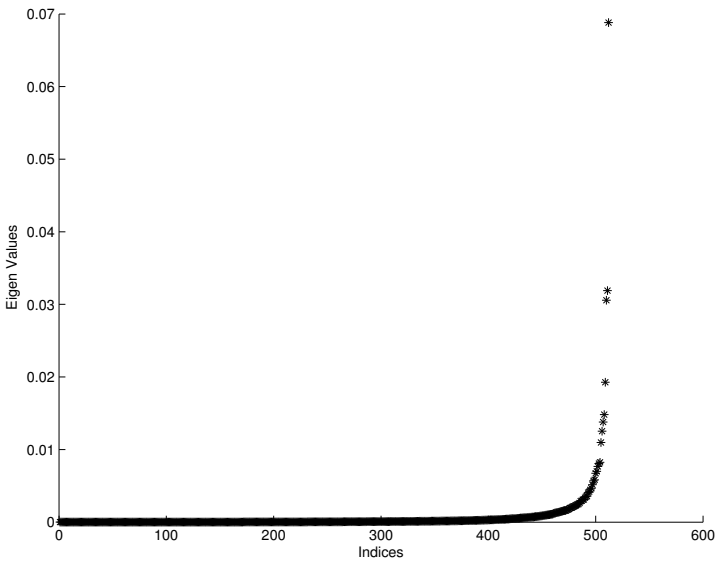


(b) varaition of Train error Test error and validation error with respect to $/nu$ values

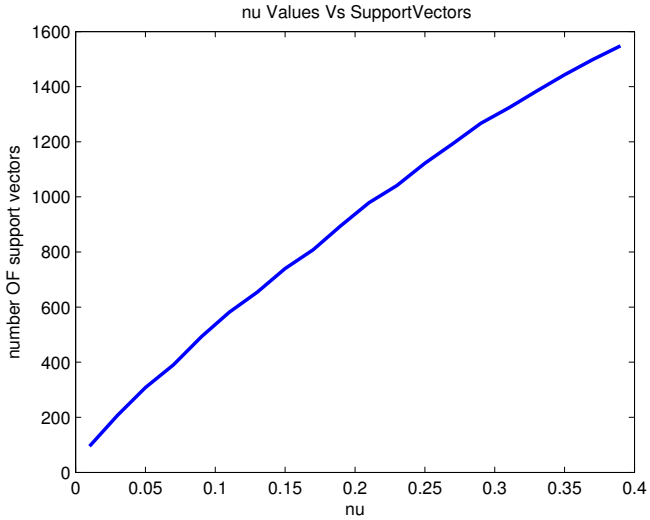


(c) Confusion Matrix

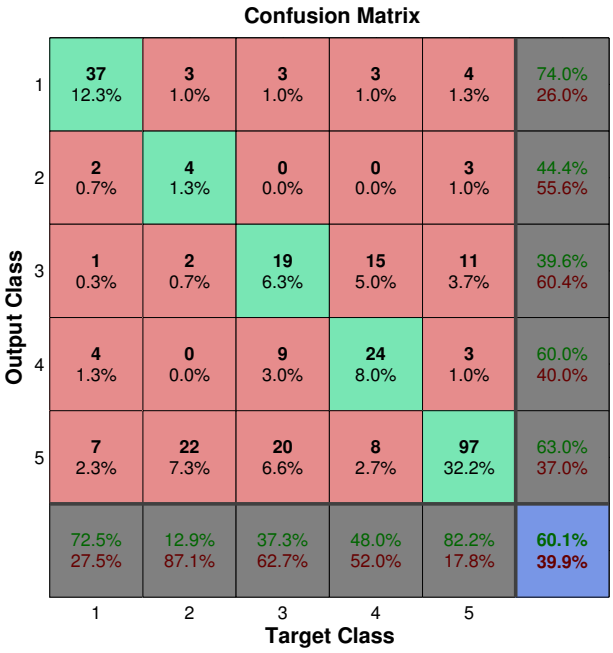
Figure 13: without PCA and gamma value 0.1



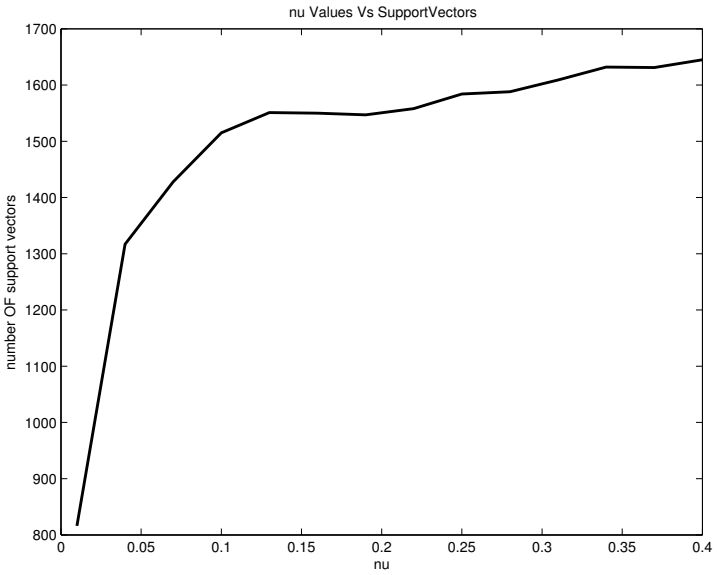
(a) Plot of Eigen Values



(b) Variation of Svs for autoencoder for configuration hidden nodes 800, and bottleneck features 150



(c) Confusion Matrix



(d) Variation of number of SVs with respect to ν values

Figure 14: With PCA taking top 58 eigen values and gama 0.1

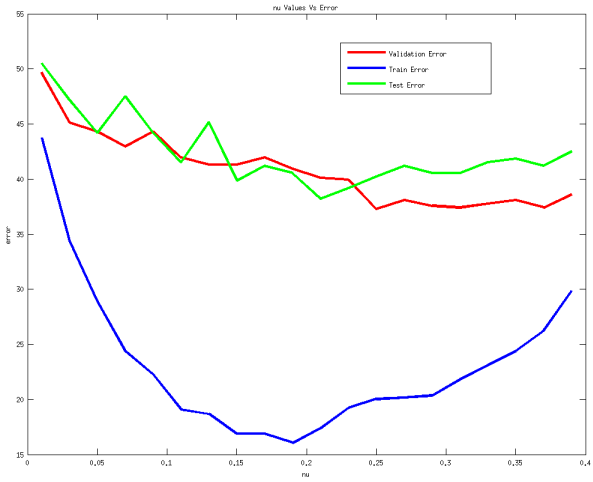
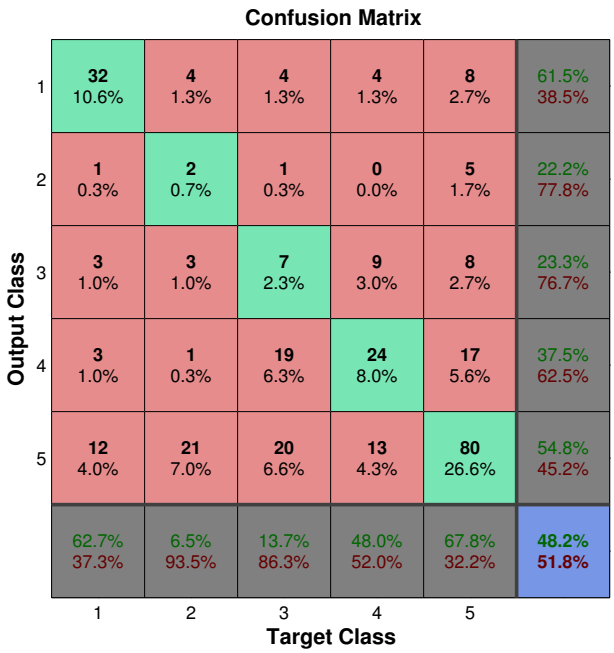
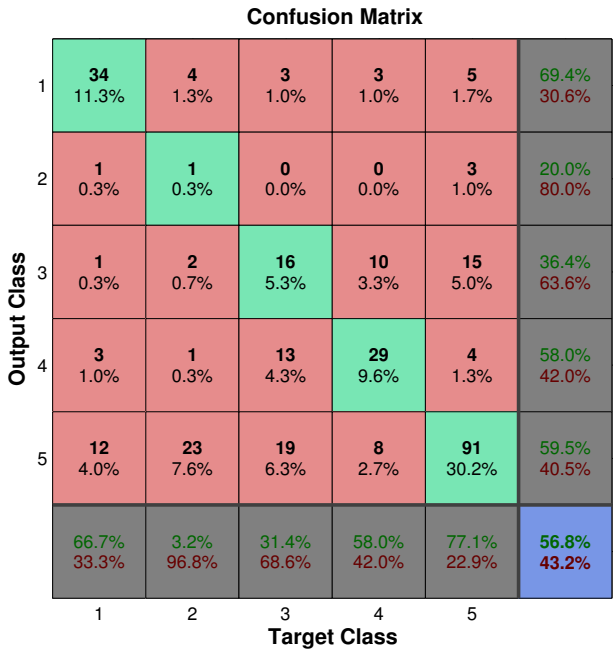


Figure 15: variation of train error test error and validation error with respect to ν values



(a) Confusion Matrix Stacked Autoencoder for configuration 512 - 800 - 400 - 600 - 200 - 400 - 100



(b) Confusion Matrix of Autoencoder with hidden layer nodes 800 and bottleneck features 150

Figure 16: Confusion matrices for stacked autoencoder and autoencoder

3 Classification using Deep Convolutional Neural Network (DCNN)

3.1 Experiments

Exp 1 (Epochs expts): First convolution layer has 6 feature maps and 3x3 window, second convolution layer has 16 feature maps and 3x3 window and third convolution layer has 26 feature maps and 3x3 window. All the pooling layer uses 2x2 window. The linear layers in the end are in the order 104, 100 and 56 output layer has 5 nodes.

Exp 2 (Linear Layer expts): First convolution layer has 6 feature maps and 3x3 window, second convolution layer has 16 feature maps and 3x3 window and third convolution layer has 26 feature maps and 3x3 window. All the pooling layer uses 2x2 window. The linear layers in the end are in the order 104, 100 and 56 output layer has 5 nodes. The nodes in the linear layers are adjusted as 100, 80 and 60.

Exp 3 (Feature map expts): First convolution layer has 6 feature maps and 3x3 window, second convolution layer has 16 feature maps and 3x3 window and third convolution layer has 26 feature maps and 3x3 window. All the pooling layer uses 2x2 window. The linear layers in the end are in the order 104, 100 and 56 output layer has 5 nodes. The number of feature maps are adjusted as 26, 32 and 64 in 3rd layer.

3.2 Inferences

- We ran the epochs experiments for 60, 120 and 200 epochs and the best results was for 120 epochs.
- For the different number of linear layer nodes as the number decreases it gives better result. for 60 we got the best accuracy.
- For different size of feature maps, as the feature maps increases the accuracy increases.

3.3 Confusion Matrices

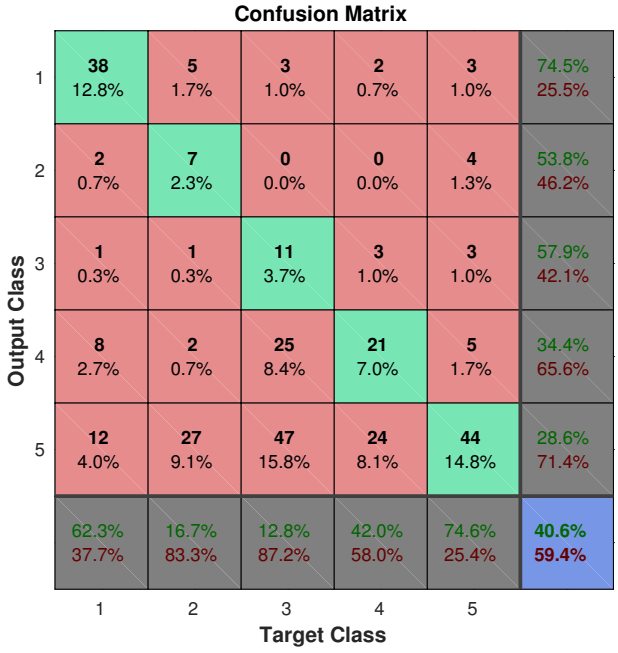


Figure 17: Classification using DCNN for 120 epochs

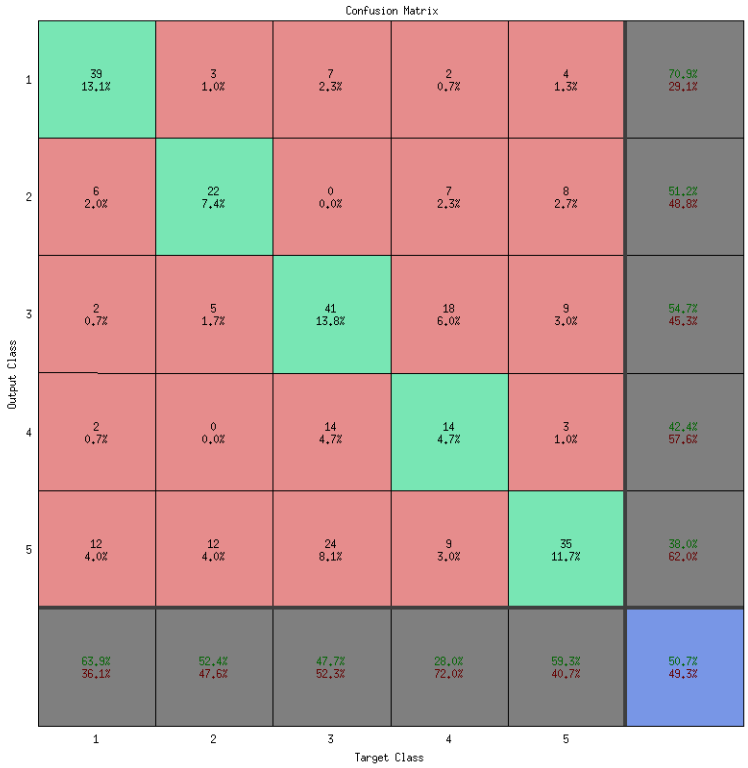


Figure 18: Classification using DCNN for different linear layer nodes

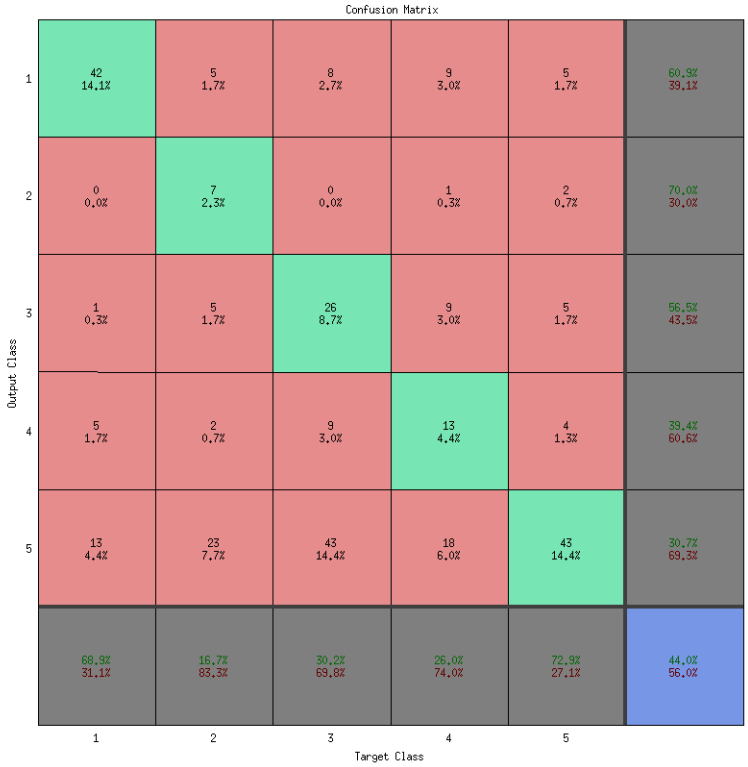


Figure 19: Classification using DCNN for different feature maps size

4 Classification using DCNN features and SVM

4.1 Experiments

First convolution layer has 6 feature maps and 3x3 window, second convolution layer has 16 feature maps and 3x3 window and third convolution layer has 26 feature maps and 3x3 window. All the pooling layer uses 2x2 window. The linear layers in the end are in the order 104, 100 and 56 output layer has 5 nodes. The nodes in the linear layers are adjusted as 100, 80 and 60. So the feature vectors of the above dimensions are fed into svm for classification.

4.2 Inferences

- For different number of linear layer nodes as the number decreases it gives better result. for 60 we got the best accuracy.
- With ν - SVM we obtained an improved accuracy than without SVM.

4.3 Confusion Matrices

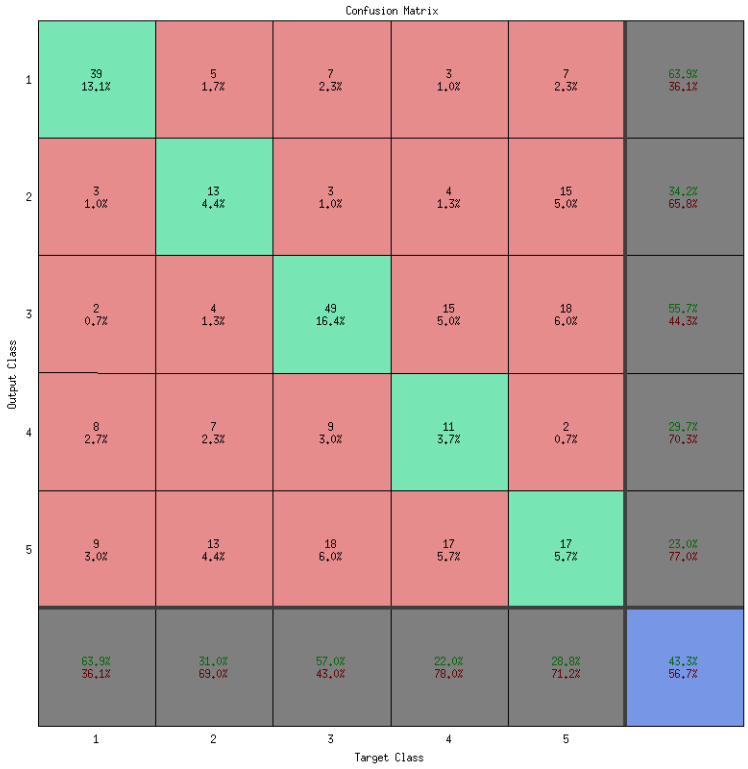


Figure 20: Classification using DCNN features and nu-svm

5 Classification using Deep Boltzmann Machine (DBM)

5.1 Confusion Matrices

13	3	2	0	0
0	19	0	2	0
1	0	19	0	0
1	0	1	15	1
1	0	0	1	15

Table 3: DBM

5.2 Inferences

- We obtained an accuracy of 86%.
- A single RBM gives an accuracy of about 73%