SNUP
Project assignment

EMARO

The images belong to one of two classes $\mathcal{S}_+$, $\mathcal{S}_-$ and are characterized by features that have independent two-dimensional normal distributions $N_{\mathbf{m},\mathbf{C}}$, where for class $\mathcal{S}_-$

$$\mathbf{m} = \begin{bmatrix} -1-a \\ -1-b \end{bmatrix}$$
$$\mathbf{C} = 0.3\,c\,\mathbf{I}$$

while for class $\mathcal{S}_+$

$$\mathbf{m} = \begin{bmatrix} 1+a \\ 1+b \end{bmatrix}$$
$$\mathbf{C} = 0.6\,c\,\mathbf{I}$$

where (i) $c = 0.5$, (ii) $c = 1$, (iii) $c = 2$, and $\mathbf{I}$ is the unit 2×2 matrix. For each case (i), (ii), (iii) create a set of 200 vectors (100 in each class) for classification purposes and another 200 vectors for testing purposes. Choose
$a =$ (the sum of the last three digits of student ID)/60,
$b =$ (the product of your last two digits of your student ID)/200.

Classify the images for all three cases (i), (ii), (iii) using different neural methods

Rosenblatt's Perceptron

1. Use one-layer Rosenblatt's Perceptron with Basic Learning Algorithm. You may additionally also apply Gallant's algorithm.

   Check the classification quality after $1000\,i$ steps of the algorithm, for $i = 1, 2, \ldots$ ('checkpoints'), by finding the number $M(m)$ of images that have been classified incorrectly for the weights at the moment $1000\,i$. Finish the training if at any checkpoint all images are classified correctly, otherwise continue up to $i = i_{\max} = 500$.

   Plot the biases and both weights against i, $i = 1, \ldots, i_{\max}$. For the final bias and weights, plot the decision line on the plane together with the classified points of both classes, both for the training as well for the testing set. Discuss the results.

Support vector machine (SVM)

2. Use one of the following SVMs

   - polynomial machine with $K(\nu, \nu') = (1 + \nu^T \nu')^c$ for $c = 2$, $c = 3$ (choose one).
   - radial machine with $K(\nu, \nu') = \exp(-\frac{1}{2\,c^2}\|\nu - \nu'\|^2)$, for $c = 0.3$, $c = 0.9$, $c = 2.7$ (choose one).

   Plot the decision line together with the classified points both for the training as well for the testing set. Discuss the classification quality, compare with Rosenblatt's algorithm.

Multi-layer Perceptron

3. Use the perceptron with sigmoidal n-neuron hidden layer (n of your choice, e.g. $n = 10$) and sigmoidal output layer. Assume that the desired output is d for class $\mathcal{S}_+$, and $-d$ for class $\mathcal{S}_-$ ($0 < d \leq 1$, e.g. $d \geq 0.8$). Plot the decision line together with the classified points both for the training as well for the testing set. You may additionally test the dependence of the results on d and n.

STUDY OF PERCEPTRONS, MULTILAYER PERCEPTRONS AND SUPPORT VECTOR MACHINES in BINARY CLASSIFICATION

MANIKANDAN BAKTHAVATCHALAM

January 5,2009

Abstract:

The aim of this project is the study and analysis of Rosenblatt perceptron, multilayer perceptrons(MLP) and Support Vector Machines(SVM) for binary classification tasks. For Rosenblatt perceptron, the classification is performed with and without using Gallant's pocket algorithm. For Support Vector Machines with Radial Basis Function, the effect of sigma on the hyperplane and for Polynomial kernels, the effect of the degree of the polynomial d are studied. In addition, the effect of the hyperparameters on the classification ability of support vector machines is also investigated.

Software used:

Matlab with SPIDER Support Vector Machine toolbox for Matlab

Dataset Generation

3 datasets with each containing 200 training vectors and 200 testing vectors were created as per the given specifications. The separability of the 2 classes was linear for dataset1 and non-linear for the datasets 2 and 3. That is, dataset1(both training and testing) has 2D feature vectors that are linearly separable. The 2nd and 3rd dataset also have 2d vectors but cannot be separated with a straight line into two classes.(non-linear separability). Among dataset2 and dataset3, dataset3 is more non-linear than dataset2. All these datasets with training and testing data have been used to test the performance of the 3 classifiers with the training data to train the classifier and testing data to test the performance. Except for the multilayer perceptron(MLP) where the training and testing datasets have been combined together and 50% of the total combination is used for training,25% for testing and the remaining 25% for validation.

PART 1
ROSENBLATT PERCEPTRON

IMPLEMENTATION:

The code for the Rosenblatt's perceptron was implemented from scratch **without using any libraries or Neural Network Toolbox functions** in Matlab.

The reasons behind this choice were:

• To experiment with perceptron for function approximation and later modify it to use it for classification.

• Approach of writing the basic code from scratch, though "re-inventing the wheel", allows a greater degree of flexibility for experimentation. I wanted to experiment with function approximation apart from classification. I also wanted to experiment with learning rate η and test how it could slow down or increase the speed of learning.

• In fact,it was quite easier to understand the necessity for and implement the Gallant's pocket learning algorithm. Further, the weight and bias corrections needed to be plotted against the number of iterations.

RESULTS:

The Rosenblatt perceptron is trained using the basic learning algorithm and the classification was tested on both the training and testing datasets. Several trial runs were made both on the training and testing datsets. Interesting facts were revealed about the limitations of the classification ability of the perceptron. They are outlined below:

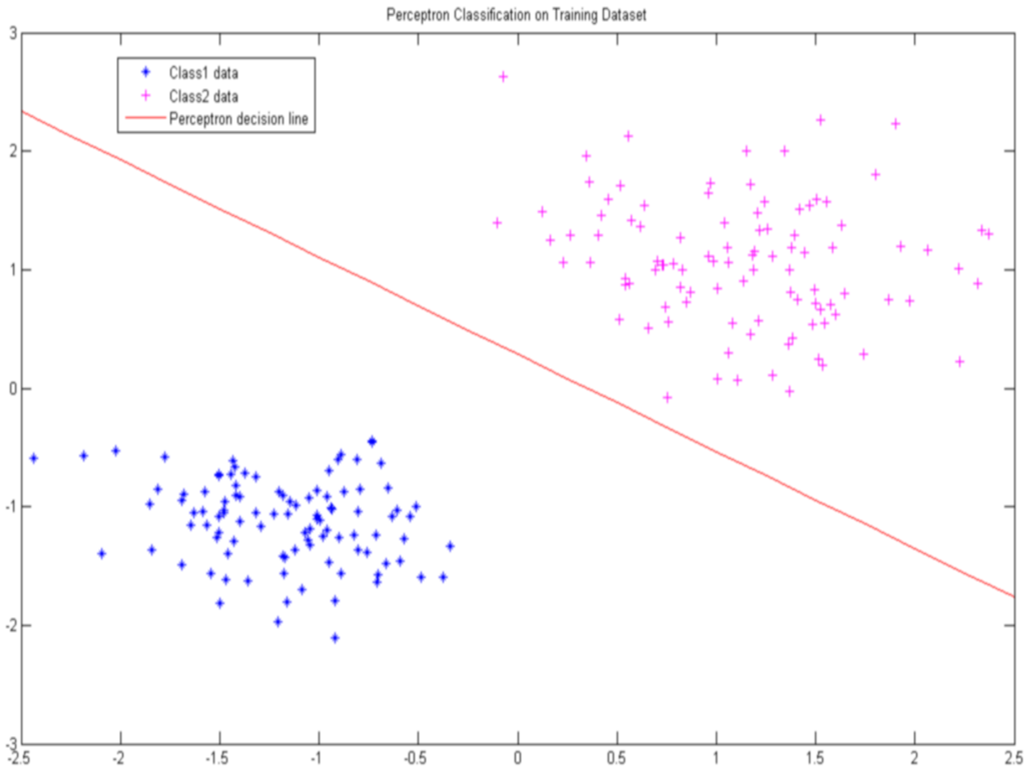
DATASET	No. of misclassifications in Trial 1	No. of misclassifications in Trial 2
Testing dataset 1(Linearly separable)	0	1
Testing dataset 2 (Non-linear separability)	13	35
Testing dataset3(Very high non-linear separability)	100	100

Perceptron for Linearly separable data points:

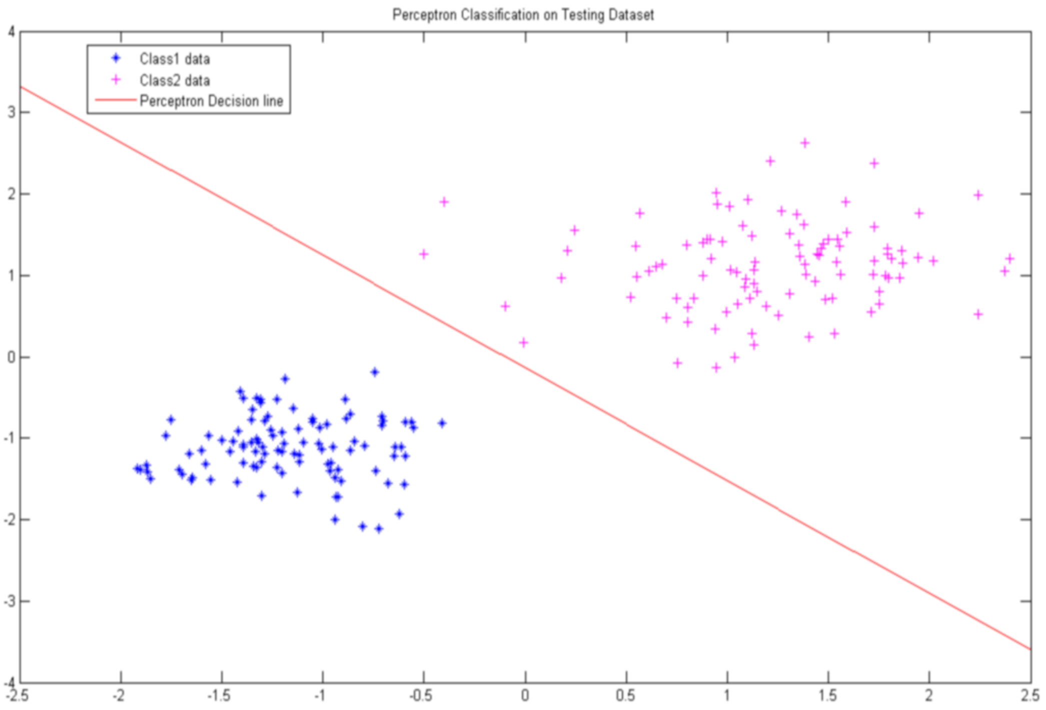
1) When used with the linearly separable dataset,the basic learning algorithm converges in a finite number of steps. To be precise, the algorithm implemented for this project converged in just 2 complete passes over the input vectors. One pass is a complete set of iterations over all the 2D points in the dataset.

2) The perceptron can classify the linearly separable dataset with no errors at almost all the trials. However, there were training runs when 1 misclassification was observed. This can be seen from Table 1. The reason is clear that the final optimal weights obtained after training differed for various trail runs. The results of the perceptron were dependent on the order of presentation of the input vectors. And the weights at the final iteration of a training is taken as the optimal weights. This is clearly shown in figures 2 and 3 where for the same linearly separable dataset, the decision line in figure3 has misclassified a point.

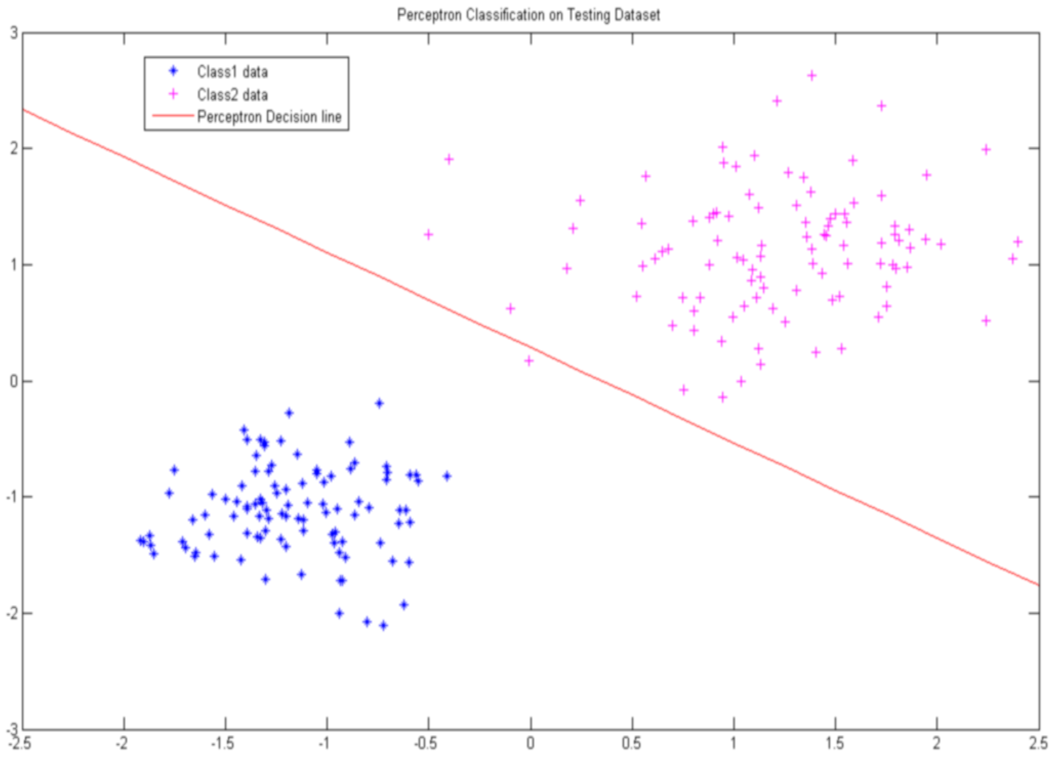
3) There are many solutions or separating planes that could perform a binary classification on the given data points when they are linearly separable.



Figure(1) Perceptron Classification on Training Dataset-1 (Linearly separable)



Figure(2) Perceptron Classification on Testing Dataset-1 (Linearly separable)
All the points are correctly classified for a linearly separable dataset.



Figure(3) Perceptron Classification on Testing Dataset-1(linear) after re-training
Note that a point in class2 is wrongly classified as Class1.This shows that the final weight vectors have changed when the training was performed again and hence reproduction of the same results cannot be expected from perceptron. Figure(2) shows all points are correctly classified after the same number of iterations the earlier training cycle.

Performance on the Non-Linearly separable data points:

The perceptron algorithm does not converge when it is used on the non-linear dataset. In this case, the learning was forced to a stop after 10000 iterations. At the end of 10000 iterations, the perceptron could not classify all the data points correctly. The plot of the decision line along with the misclassified points is shown in This holds true even if the number of iterations or passes was increased to 20000. This shows that this is independent of the number of iterations. The weights move randomly in the weight space and never converges.

Performance using Gallant's pocket learning algorithm:

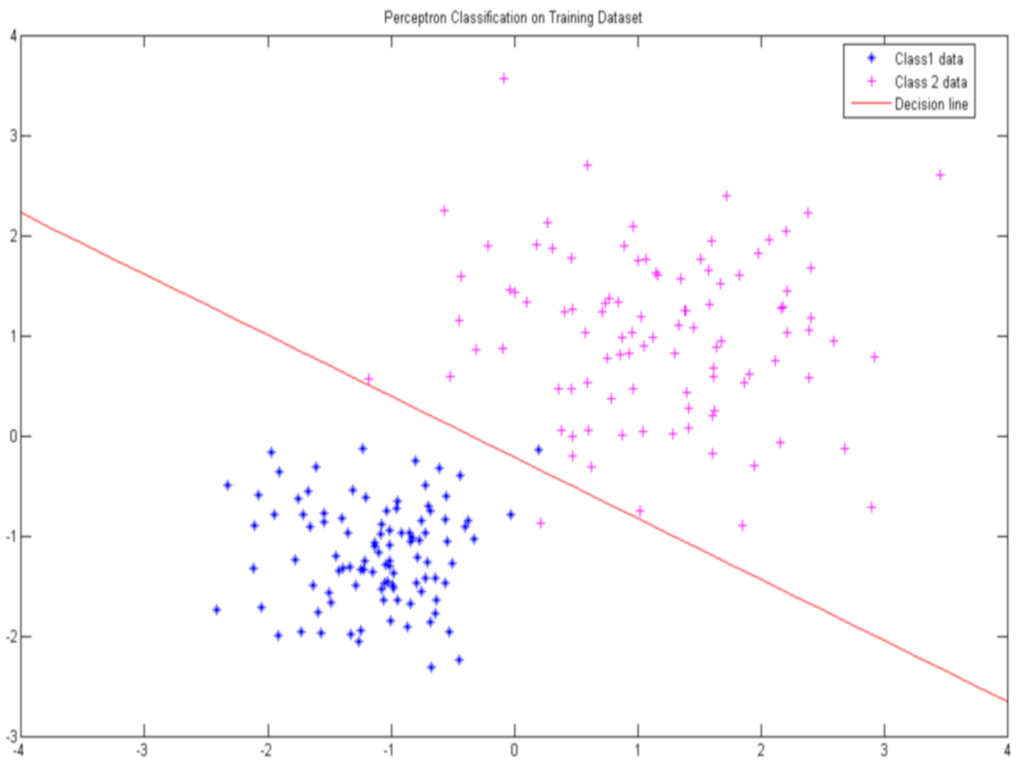
Method:

When using the Gallant's algorithm, we keep track of the number of correctly classified points and the best weights (that cause maximum number of points to be correctly classified – weights in“pocket”). During an iteration, the current weights are updated only if they classify more points correctly than the earlier chosen best vector in the pocket. The weight vector that causes the maximum number of correct classifications is chosen to be the optimal weights or solution weights.

Results Obtained:

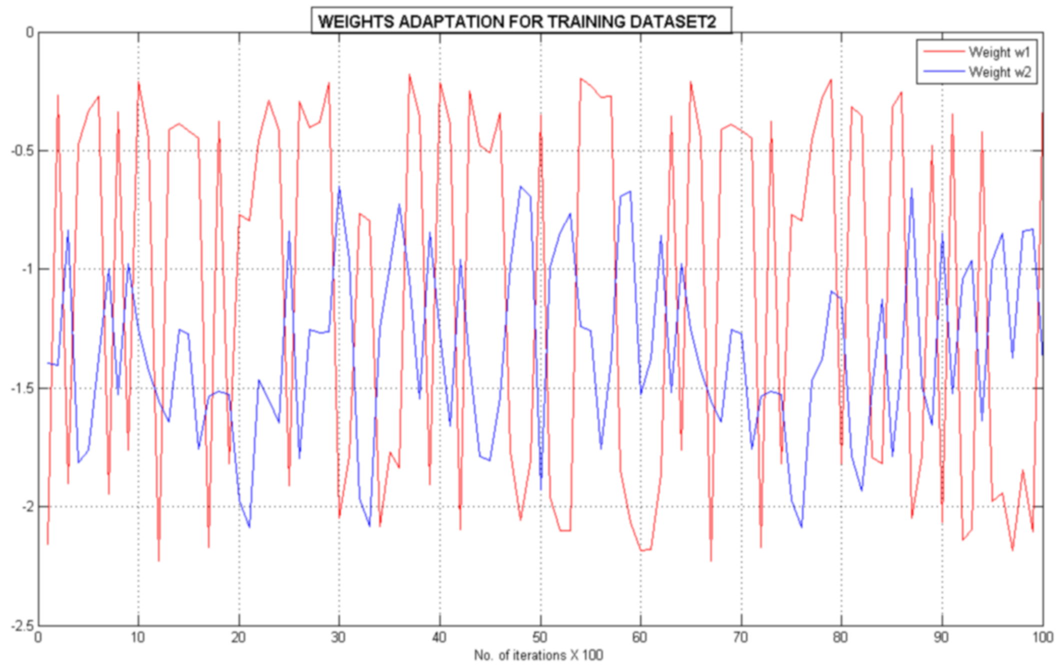
DATASET	Misclassifications after 1000 iterations		Misclassifications after 3000 iterations		Misclassifications after 10000 iterations	
	Without Gallants algorithm	After Gallant's pocket algorithm	Without Gallants algorithm W	After Gallant's pocket algorithm	Without Gallants algorithm	After Gallant's pocket algorithm
Testing dataset 1(Linearly separable)	1	0	0	0	0	0
Testing dataset 2 (Non-linear separability)	13	3	54	3	31	3
Testing dataset3(Very high non-linear separability)	100	100	100	100	100	100

As seen from the table, Gallants algorithm results in better classification results in the case of the dataset2 (non-linear). However, on dataset3 in which the separability is highly non-linear, the perceptron cannot perform well and half of the data is wrongly classified, even when Gallant's algorithm is used.

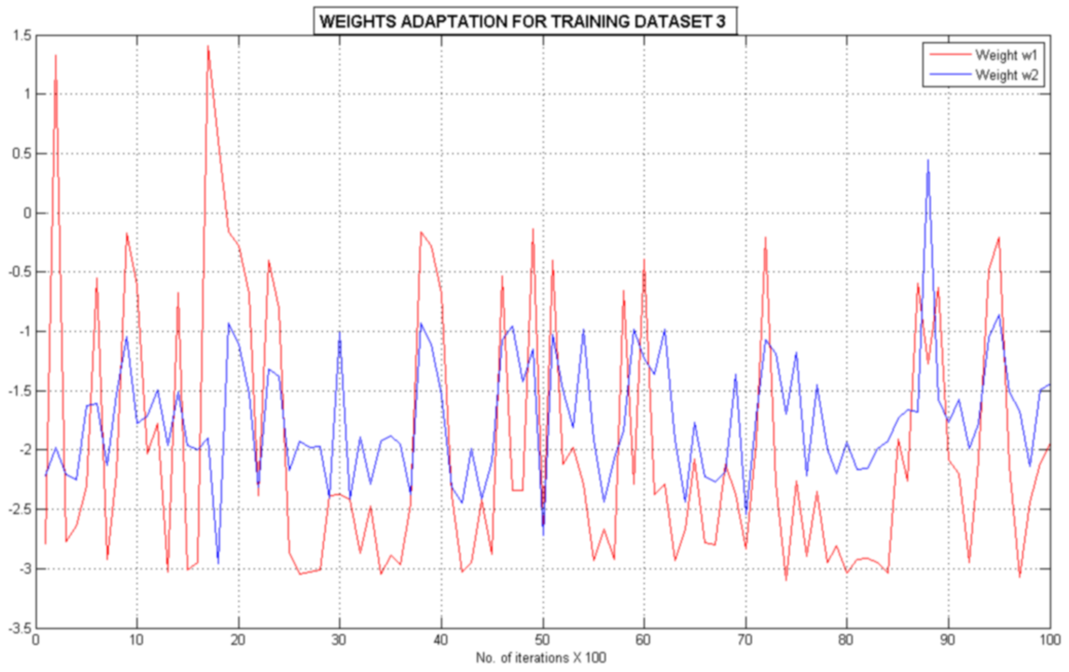


Perceptron Classification on non-linear dataset

Inability of perceptron to learn a non-linearly separable dataset. 2 data points classified incorrectly for the non-linearly separable dataset. Classification results are after 10,000 iterations.



Adaptation of weights while training perceptron on training dataset2



Adaptation of weights while training perceptron on training dataset3

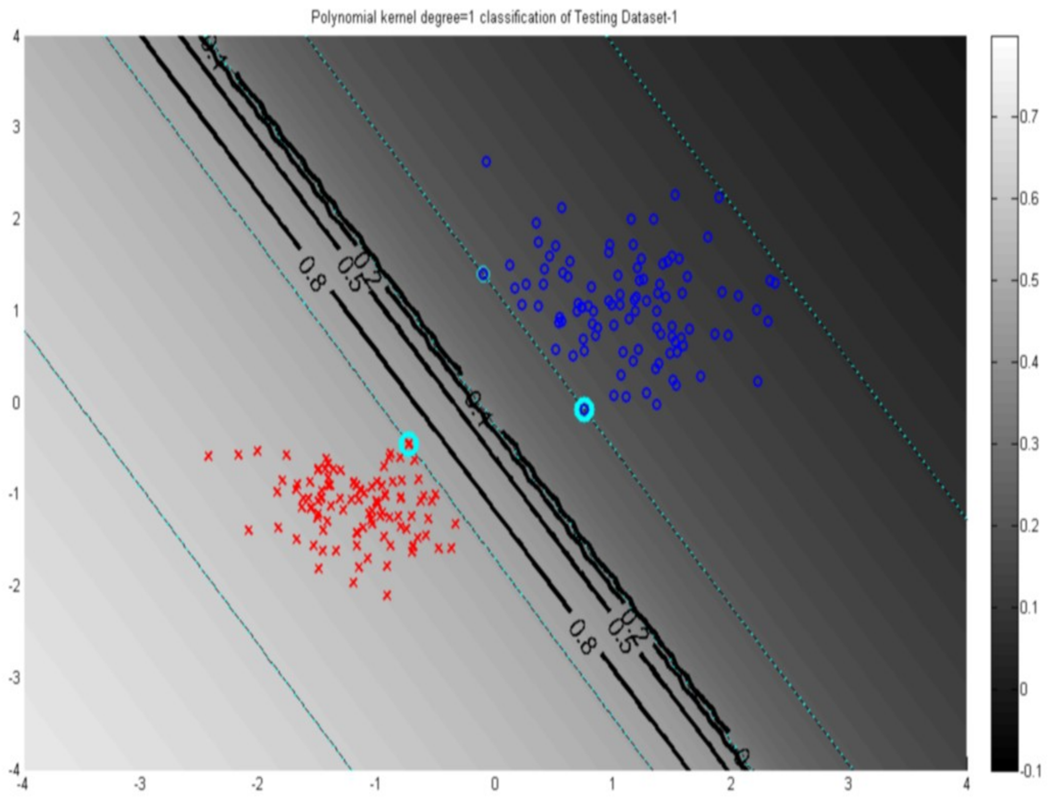
Shown above are adaptation of weights for the perceptron while training on linear datasets. (The bias is not shown here).The perceptron learns to classify the linear dataset quickly and hence would appear as a straight line in only a few iterations.(not shown here again).

PART 2

SUPPORT VECTOR MACHINES

POLYNOMIAL MACHINE:

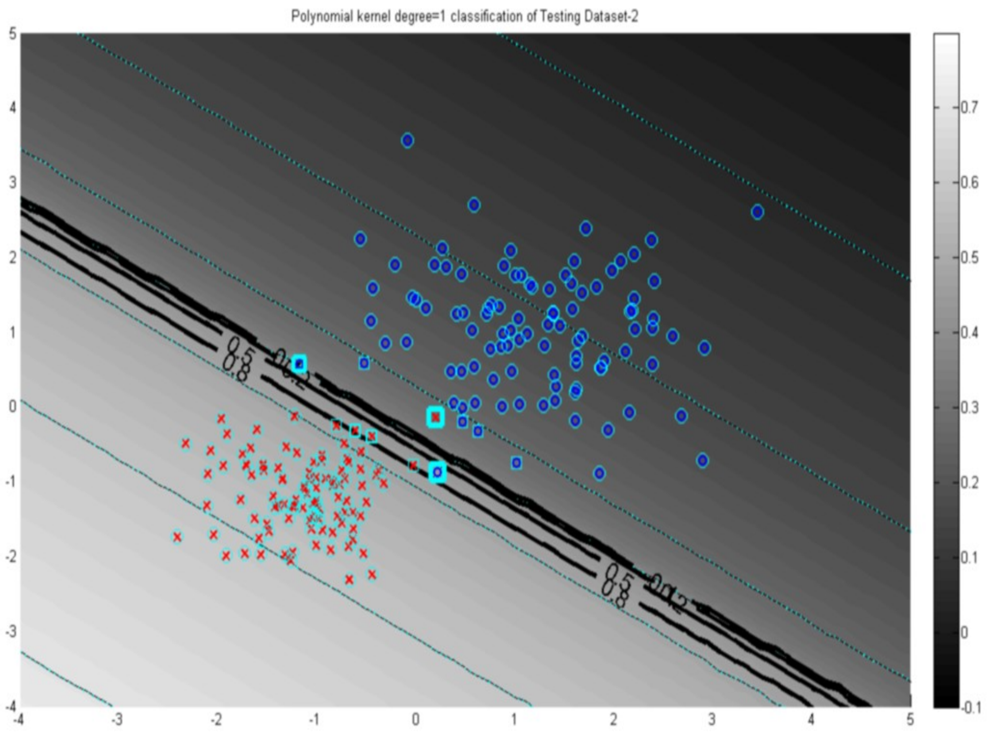
Polynomial machines of degree1,2 and 3 are used to classify the all the 3 datasets. The training was performed using the training dataset and then the classification was tested on the testing dataset. The classification results shown below are SVM performance on the testing dataset for the all the 3 kinds of datasets, as mentioned earlier.



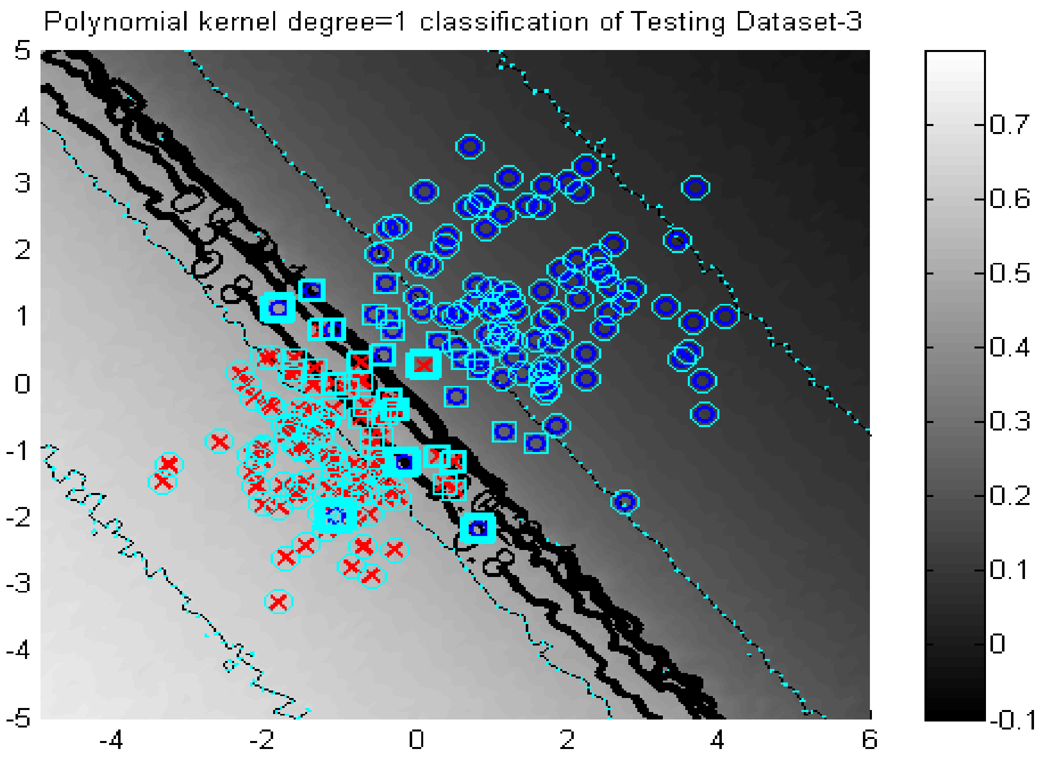
Classification of Testing dataset1 using SVM with linear Polynomial kernel (degree 1)

This is a linearly separable dataset. In this case, the SVM with linear polynomial kernel suffices to separate the two classes with a maximal margin boundary, namely the optimal *hyperplane.*

The actual support vectors are shown with fluorescent colored circles. These are the ones closest to the decision boundary.



Classification of Testing dataset-2 using SVM with linear Polynomial kernel (degree 1)
(Incorrect results for non-linearly separable dataset)

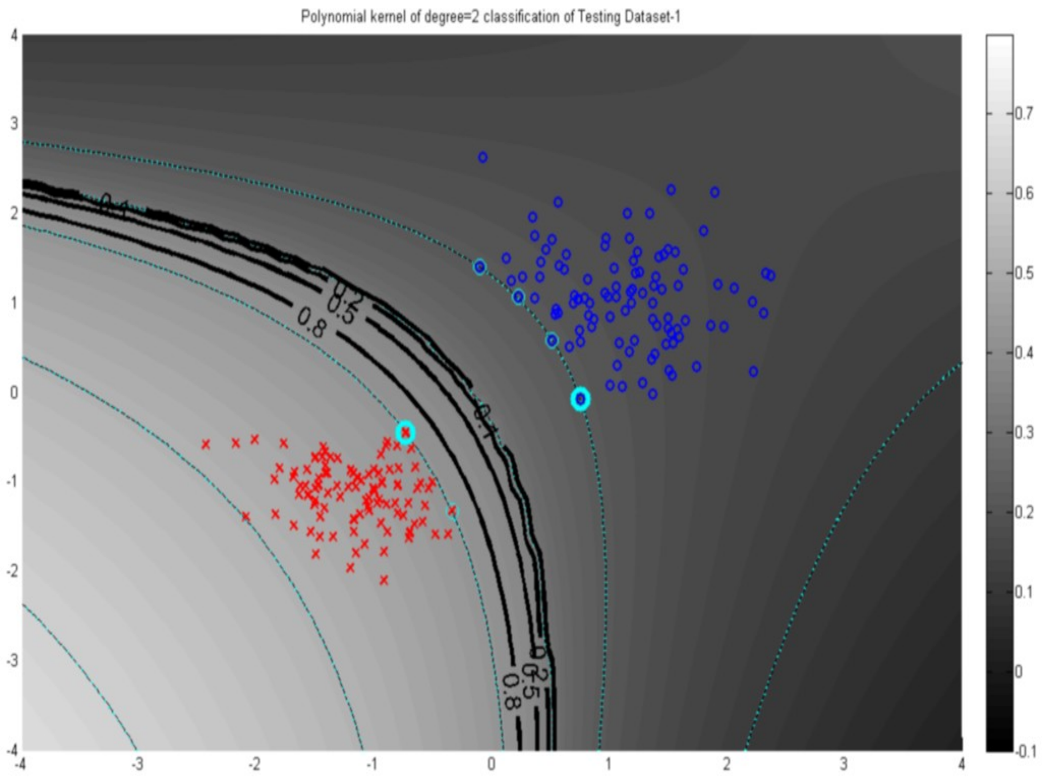


Classification of Testing dataset-3 using SVM with linear Polynomial kernel (degree 1)
(Incorrect results for non-linear dataset)

The above two figures show clearly that the SVM with linear polynomial kernel cannot separate the two classes when their separability is non-linear. In such cases where the data are not linearly separable, the SVM gives inconsistent results as shown above.

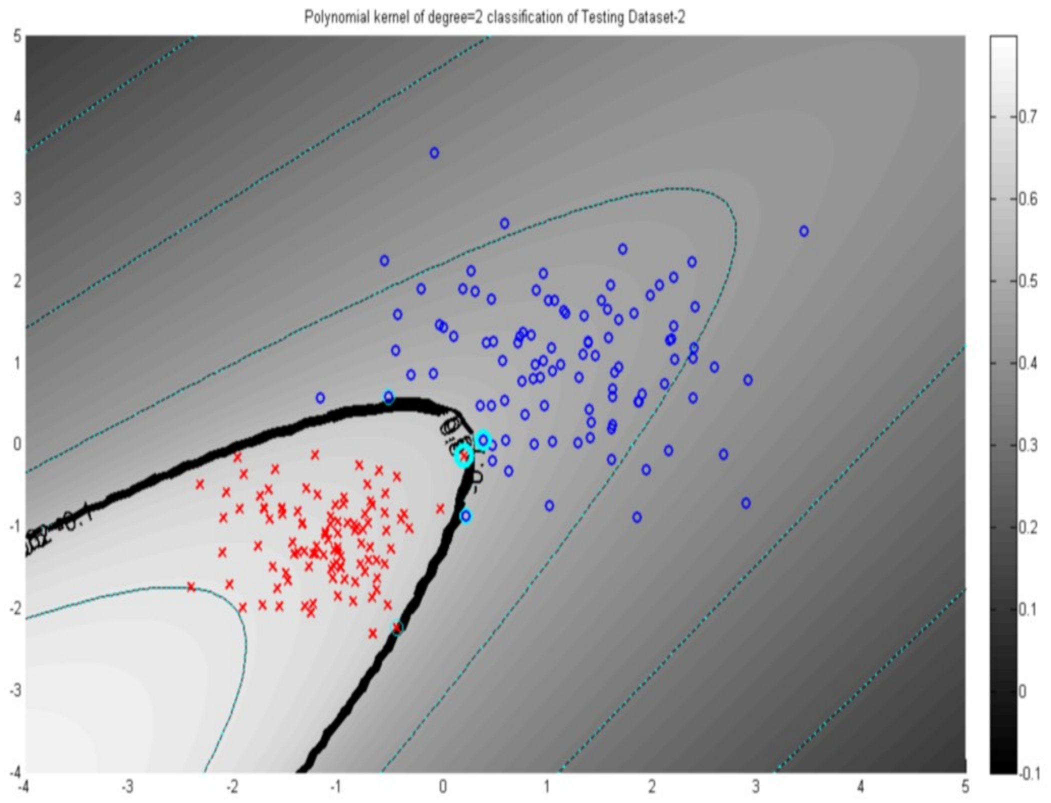
** In the Spider machine learning toolbox for matlab, the library issues a warning when trying to classify data that are not linearly separable. When using such improper kernels, there occurs a singularity in the optimization step (quadriatic proramming) inside the library. And the library issues a prompt warning that the results may be inaccurate.*

```
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 2.081708e-017.
```



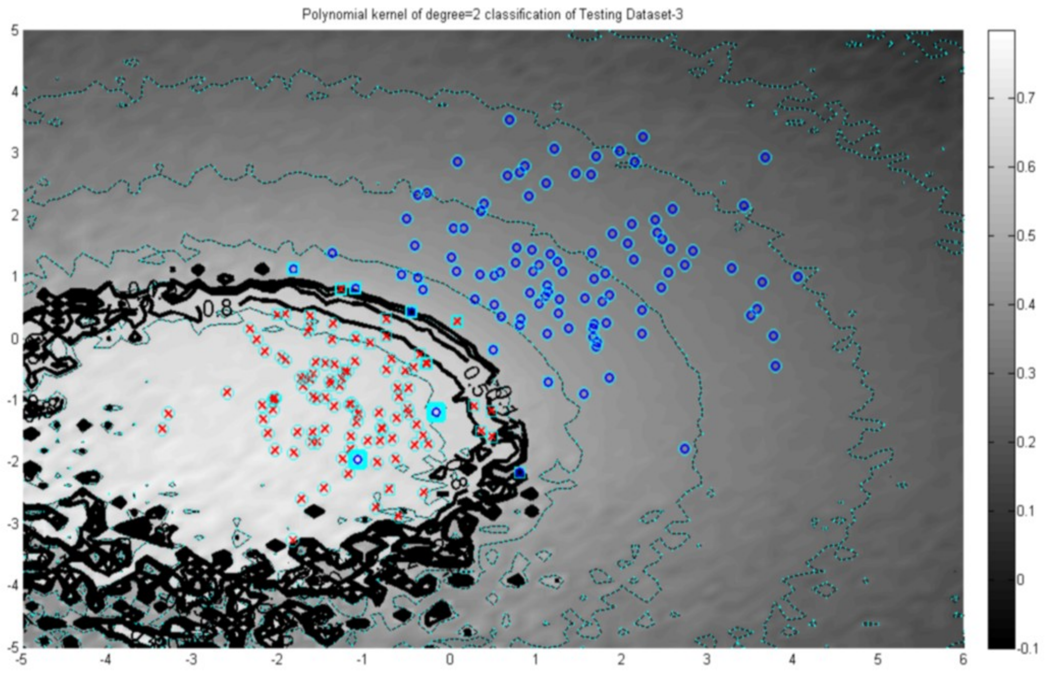
Classification of Testing dataset-1 using SVM with 2nd degree Polynomial kernel

The polynomial of degree 2 performs the same classification as the linear kernel although with a greater degree of flexibility.

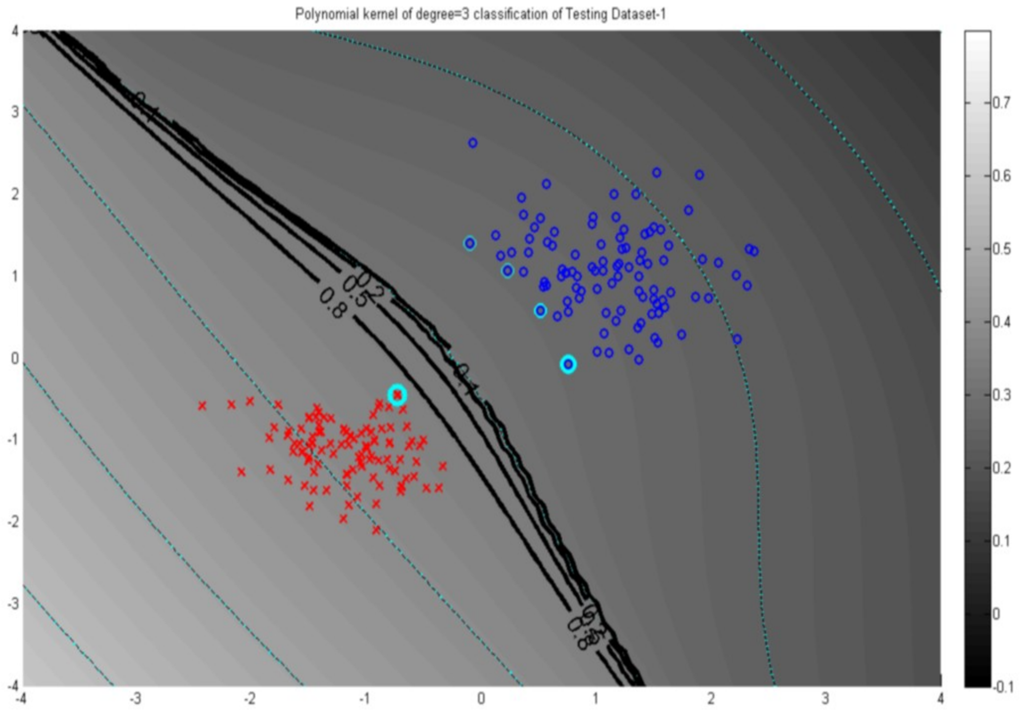


Classification of Testing dataset-2 using SVM with 2nd degree Polynomial kernel

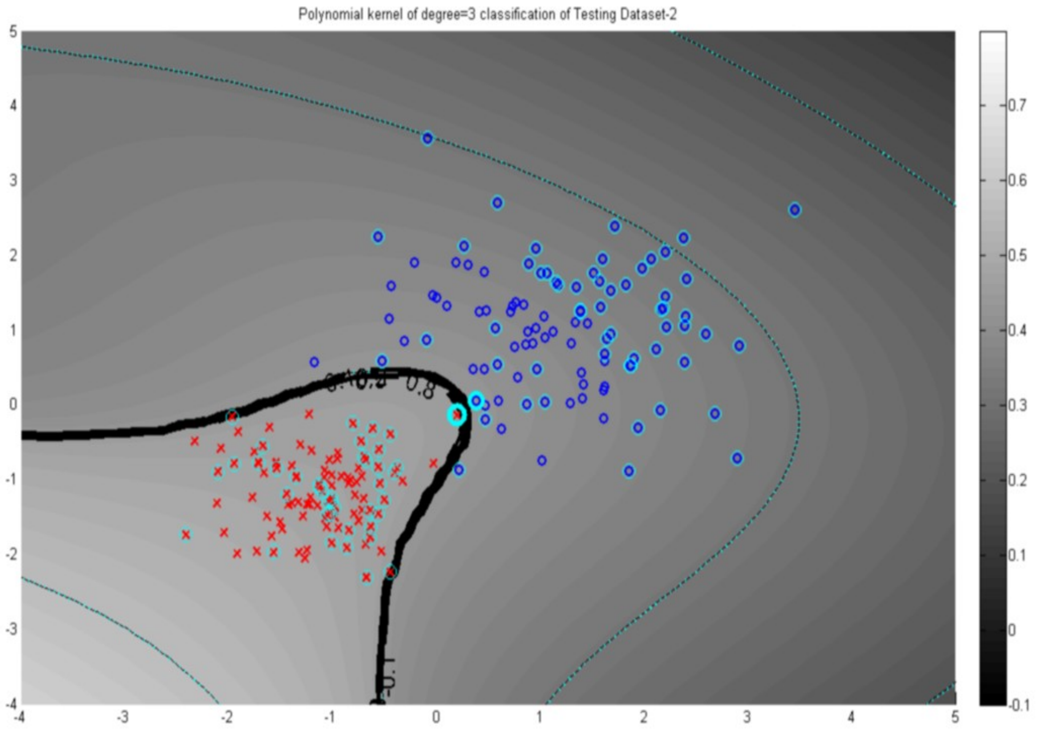
The polynomial kernel of 2nd degree separates optimally with a hyperplane the non-linear dataset that could not be classified by the linear polynomial machine.



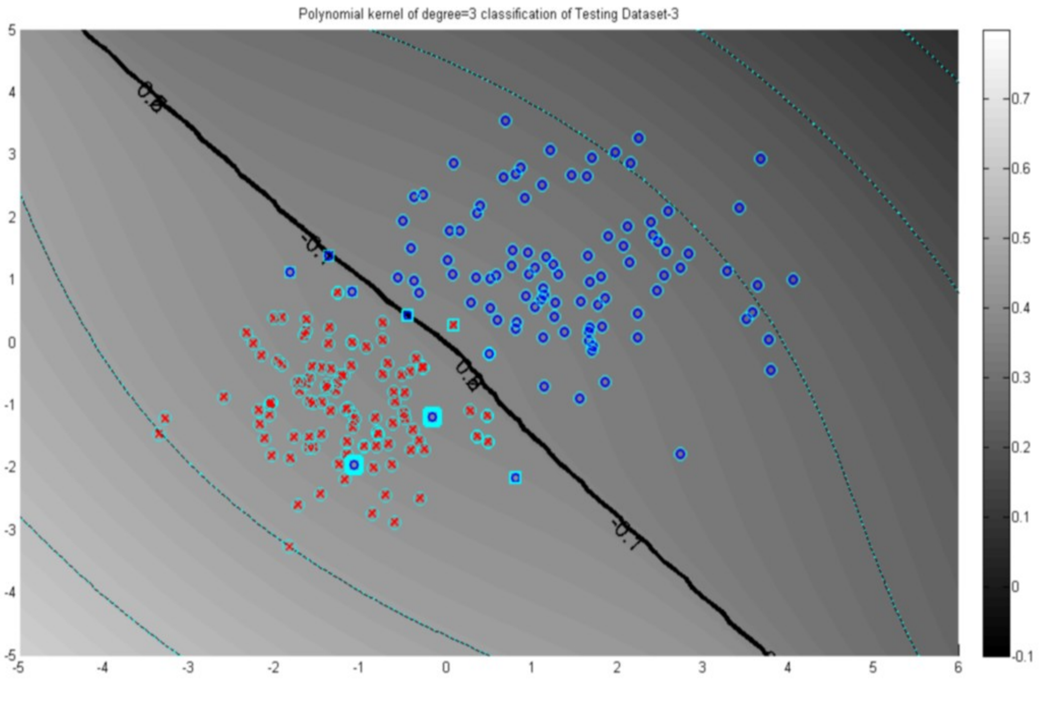
Classification of Testing dataset-3 using SVM with 2nd degree Polynomial kernel
(Incorrect results when applied on the 3rd dataset with high non-linear separability)



Classification of Testing dataset-1 using SVM with 3nd degree Polynomial kernel



Classification of Testing dataset-2 using SVM with 3nd degree Polynomial kernel



Classification of Testing dataset-3 using SVM with 3nd degree Polynomial kernel Polynomial SVM Results:

The following observations were made from the results.

1) A linear kernel is sufficient to perform a binary classification of the linearly separable data. In this case, the SVM finds the optimal hyperplane that separates the two classes.

2) The linear polynomial machine with degree1 is unable to classify testing datasets 2 and 3. Therefore, the lowest degree polynomial namely the linear kernel, which is not sufficient when a non-linear relationship between features exists.

EFFECT OF DEGREE 'd' on the Polynomial Kernel:

The polynomial machine of degree 2 separates with an optimal hyperplane the same non-linear dataset(dataset2) that the linear polynomial machine could not classify. At the same time, we can observe that the same SVM with 2nd degree polynomial cannot find a separating hyperplane for the 3rd dataset.

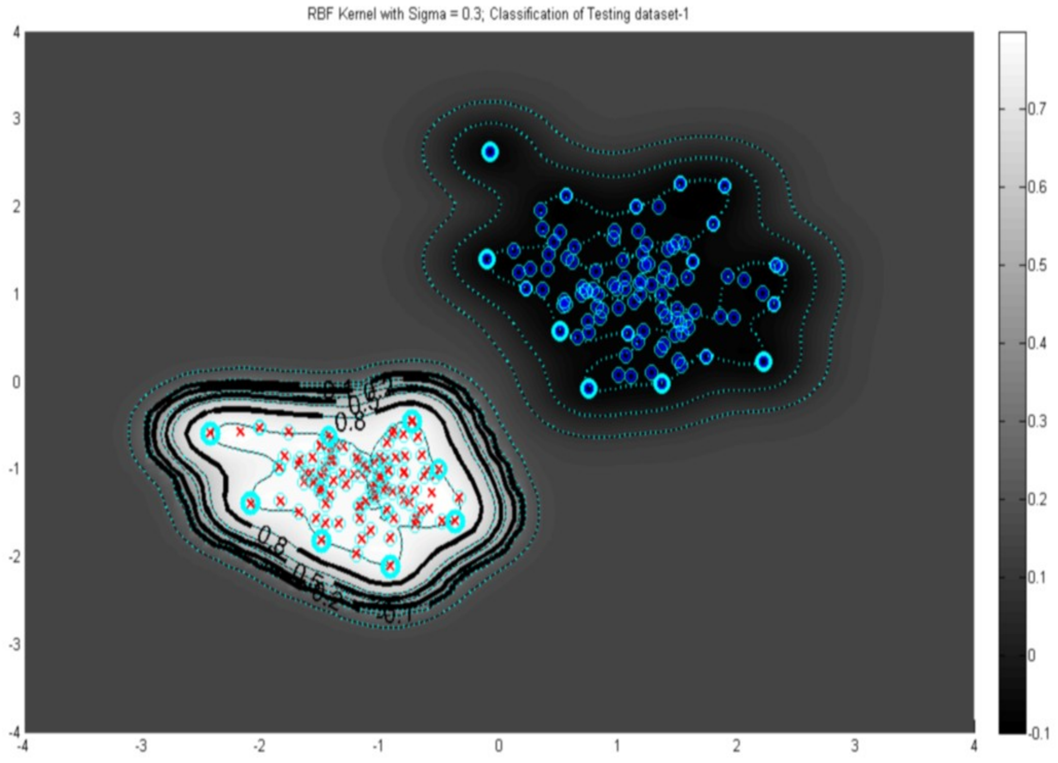
The degree of the polynomial determines the **flexibility** of the resulting classifier. For example, the polynomial machine of degree 2 and degree 3 perform the same separation as the linear polynomial kernel although with a greater degree of flexibility as shown in figures [] and [].

Overfitting problem:

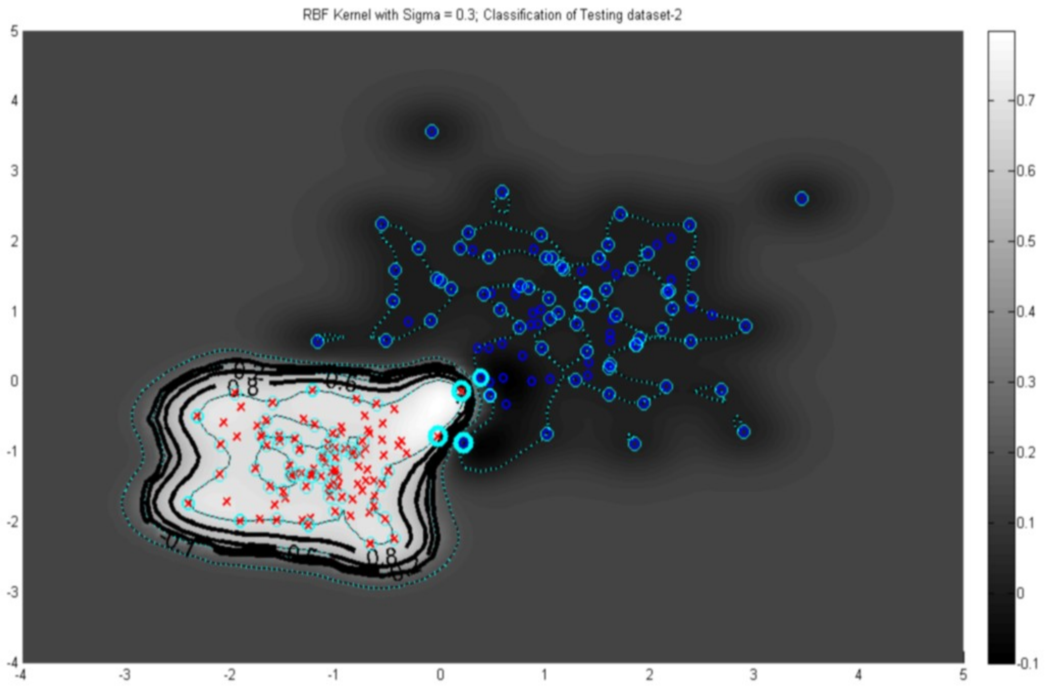
When a still higher degree kernel is used, the classic problem of overfitting to training data happens and the results are incorrect on the testing dataset.

RADIAL BASIS FUNCTION KERNELS

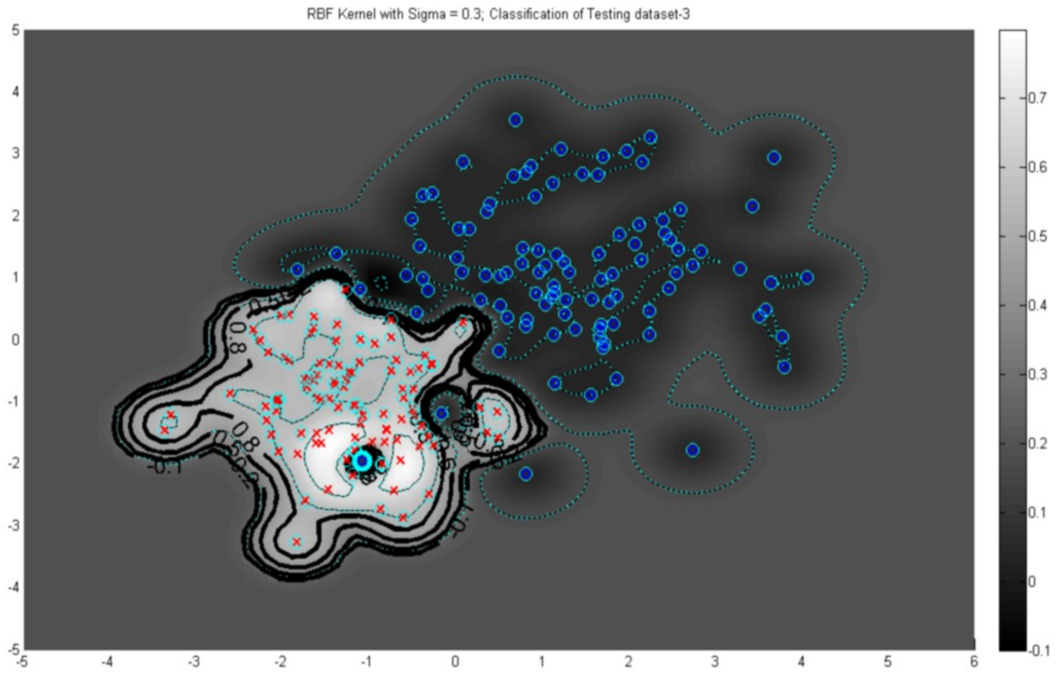
Radial basis functions with different widths namely $\sigma = 0.3$, $\sigma = 0.9$ and $\sigma = 2.7$ are used for the classification of all 3 types of datasets. The results are shown below and the effect of the parameter σ on the classication results have been discussed.



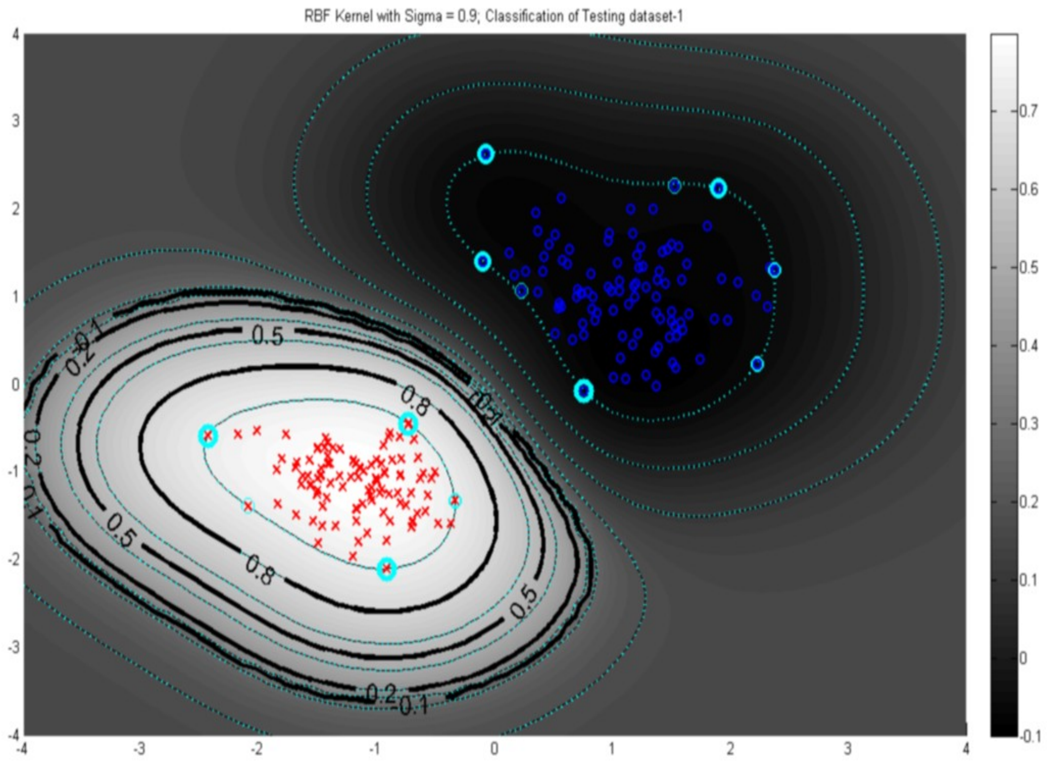
Classification of Testing dataset-1 using SVM with Radial Basis Function kernel $\sigma = 0.3$



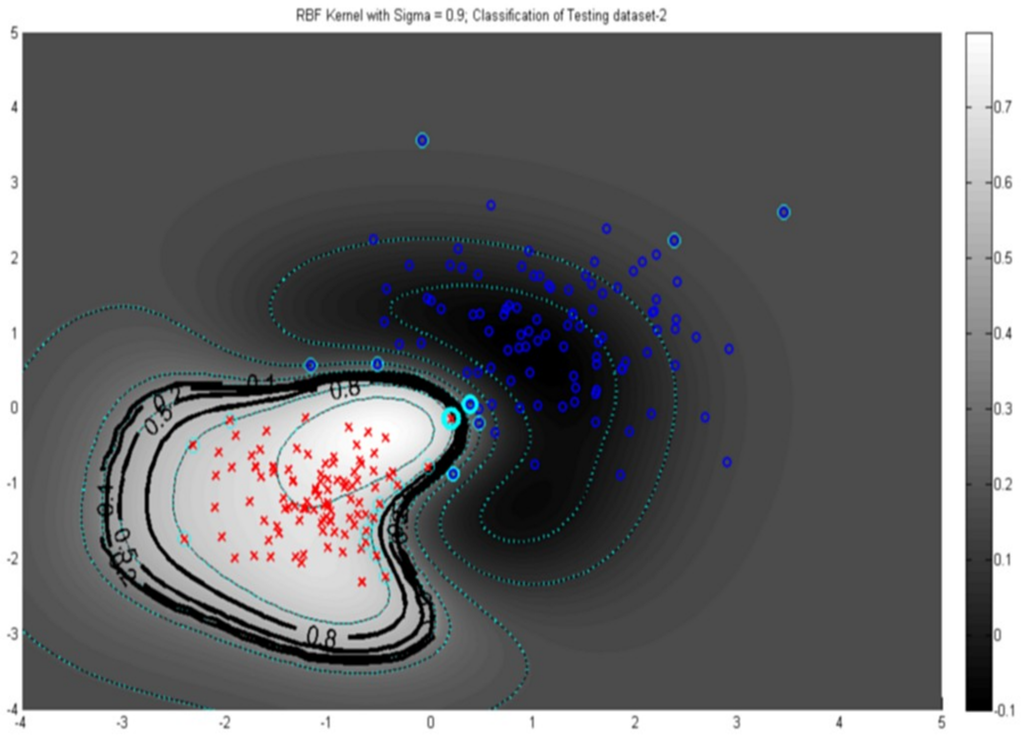
Classification of Testing dataset-2 using SVM with Radial Basis Function kernel $\sigma = 0.3$



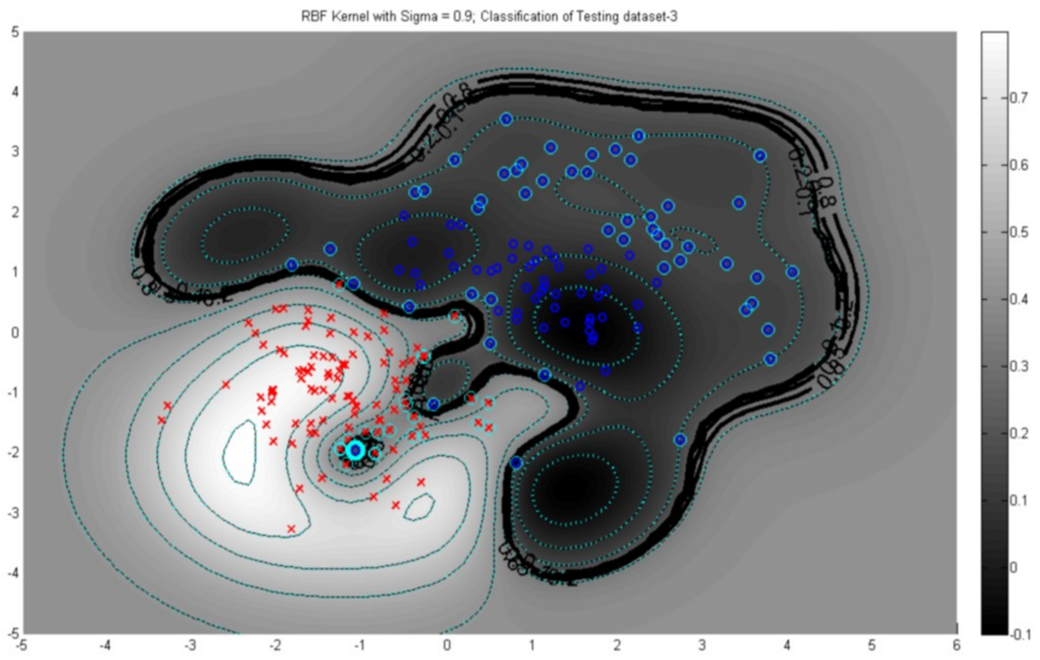
Classification of Testing dataset-3 using SVM with Radial Basis Function kernel $\sigma = 0.3$



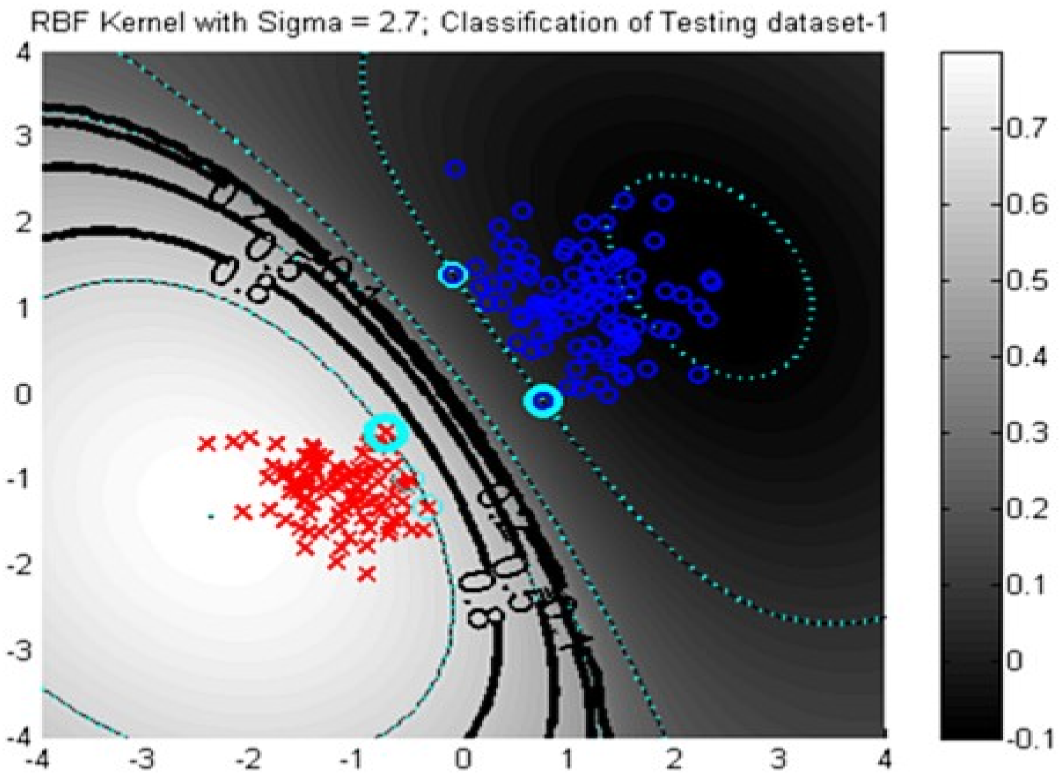
Classification of Testing dataset-1 using SVM with Radial Basis Function kernel $\sigma = 0.9$



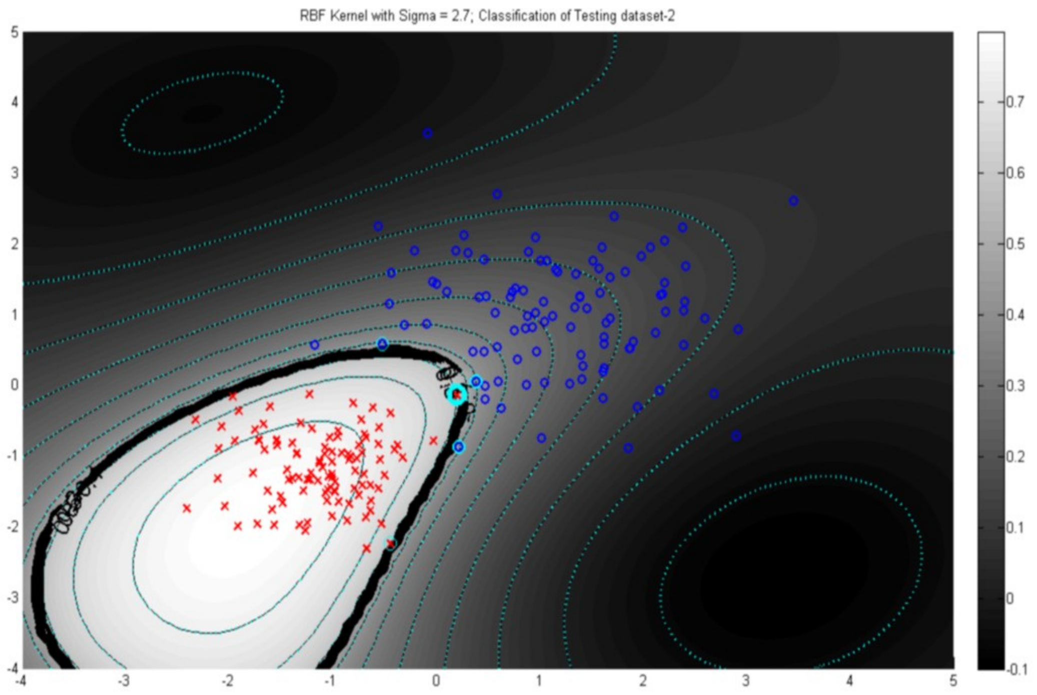
Classification of Testing dataset-2 using SVM with Radial Basis Function kernel $\sigma = 0.9$



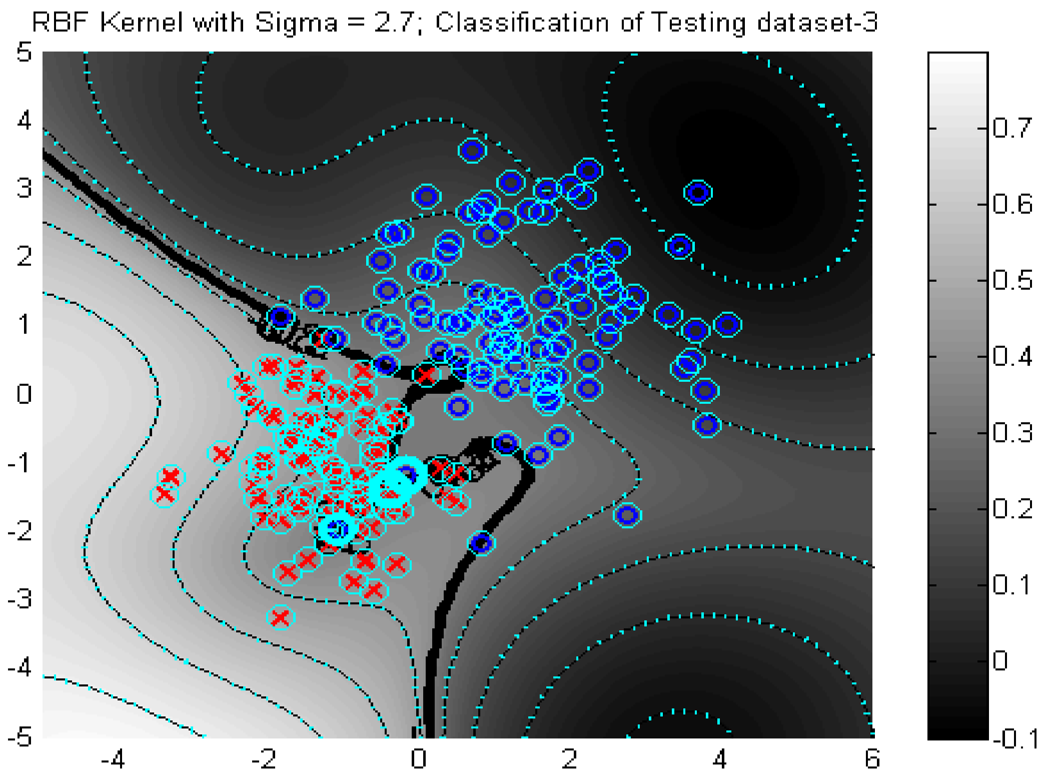
Classification of Testing dataset-3 using SVM with Radial Basis Function kernel $\sigma = 0.9$



Classification of Testing dataset-1 using SVM with Radial Basis Function kernel $\sigma = 2.7$



Classification of Testing dataset-2 using SVM with Radial Basis Function kernel $\sigma = 2.7$



Classification of Testing dataset-3 using SVM with Radial Basis Function kernel $\sigma = 2.7$

EFFECT OF WIDTH σ OF THE RBF KERNEL ON THE CLASSIFICATION:

For very small values of width, the training leads to overfitting. The inverse of the width of the Gaussian RBF kernel is called γ When this γ value is very large(i.e., σ is small) then overfitting occurs. This can be observed from the figures [],[] and []. The SVM kernel with $\sigma = 0.3$ shows the classification results obtained as a result of overfitting the data.

However, for smaller values of the inverse of width, the SVM classifies the data by finding the optimal hyperplane.

MULTILAYER PERCEPTRON

<u>IMPLEMENTATION:</u>

A 2 layer multilayer perceptron with a hidden layer of 10 sigmoidal neurons and outer layer of 1 sigmoidal neuron is used for classification. There are 2 versions of the code, one using MatLab NN toolbox and the other is a custom function that has been written from scratch.

Change in usage of training and testing data: Unlike for the perceptrons and SVM, here the training and testing datasets are merged together in order to comply with data formats required by the Matlab functions. And later the whole set has been divided like 50% of training, 25% each for testing and validation.(Validation is to test for generalization of the trained MLP). The code snippet is as below:

```
p = [U X];
[trainV,val,test] = dividevec(p,t,0.25,0.25);
```

p is the whole input vector set consisting of both the training and testing datasets.
t is the desired target values namely the class numbers.

The same procedure is followed for each of the 3 different types of datasets(linear, non-linear and highly non-linear).

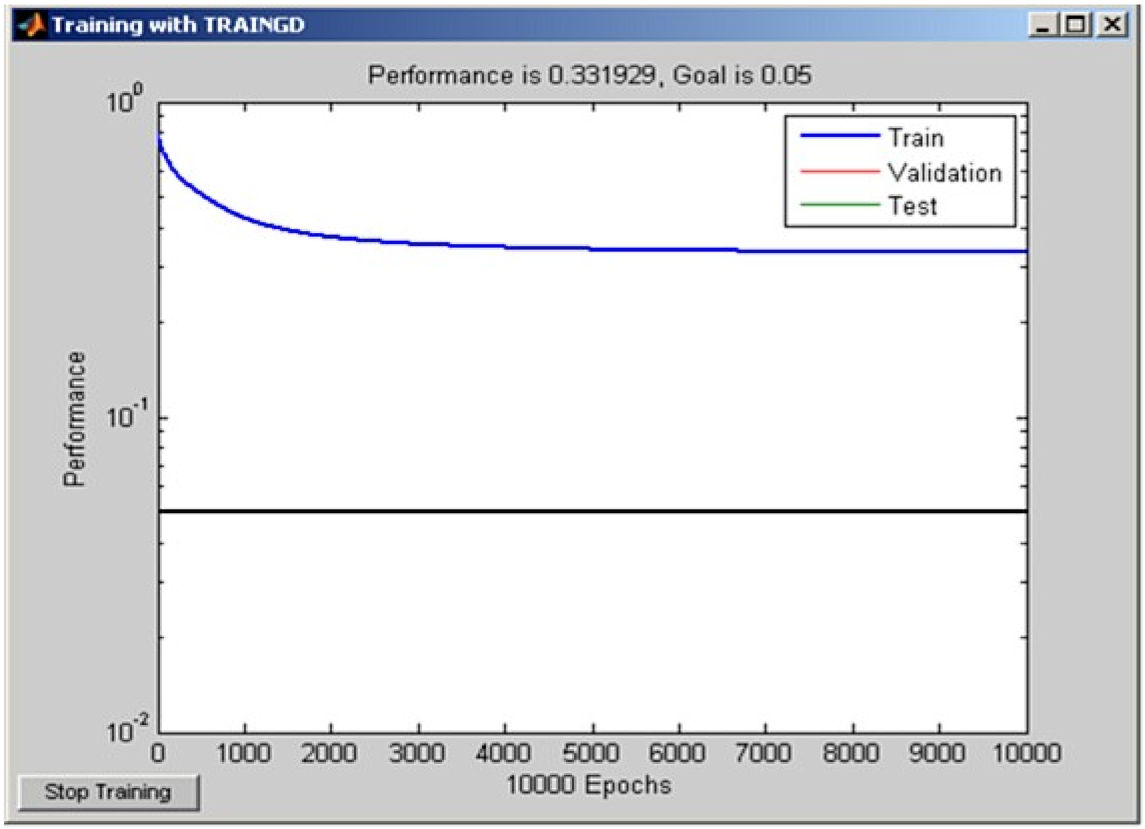
<u>RESULTS:</u>

For all 3 datasets,number of mistakes: 0. The MLP is able to approximate the underlying function and hence performs the binary classification task correctly on all the 3 datasets.

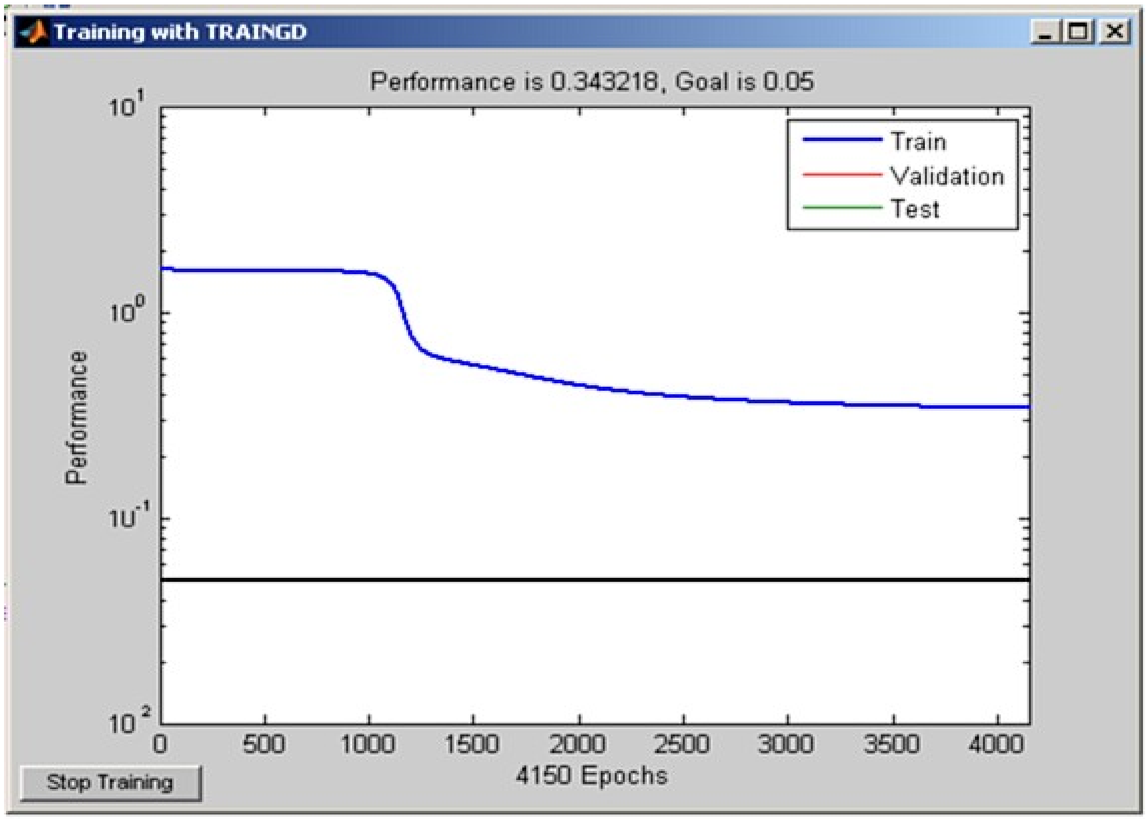
The MLP with 10 hidden neurons can learn the decisions and correctly classify all the vectors in both the linear and non-linear datasets. This confirms to the representation theorem that a MLP with sufficient number of neurons in the hidden layer can learn any arbitrary function and approximate it. In binary classification, the task becomes much simpler since the outputs are same for members of 1 class.

P.S: The decision surface is not shown here,However, two versions of the MLP code,one using Matlab NN toolbox and the other custom-written version are both included at the end of the report)

<u>EFFECT OF NUMBER OF HIDDEN LAYER NEURONS ON THE MLP LEARNING:</u>



Above: Hidden layer with 10 neurons takes 10,000 epochs to achieve an MSE = 0.33



Above: Hidden layer with 20 neurons achieves a comparable MSE (~0.33) in 4000 epochs

<u>Quick training (vs) Overfitting</u>

The concept here is that with increase in the number of neurons, the network learns the underlying function quickly.(with regards to function approximation).In the above figures, the MLP with 20 neurons in the hidden layer can learn the underlying function in even less than 50% of the iterations required in the case of an MLP with 10 neurons.

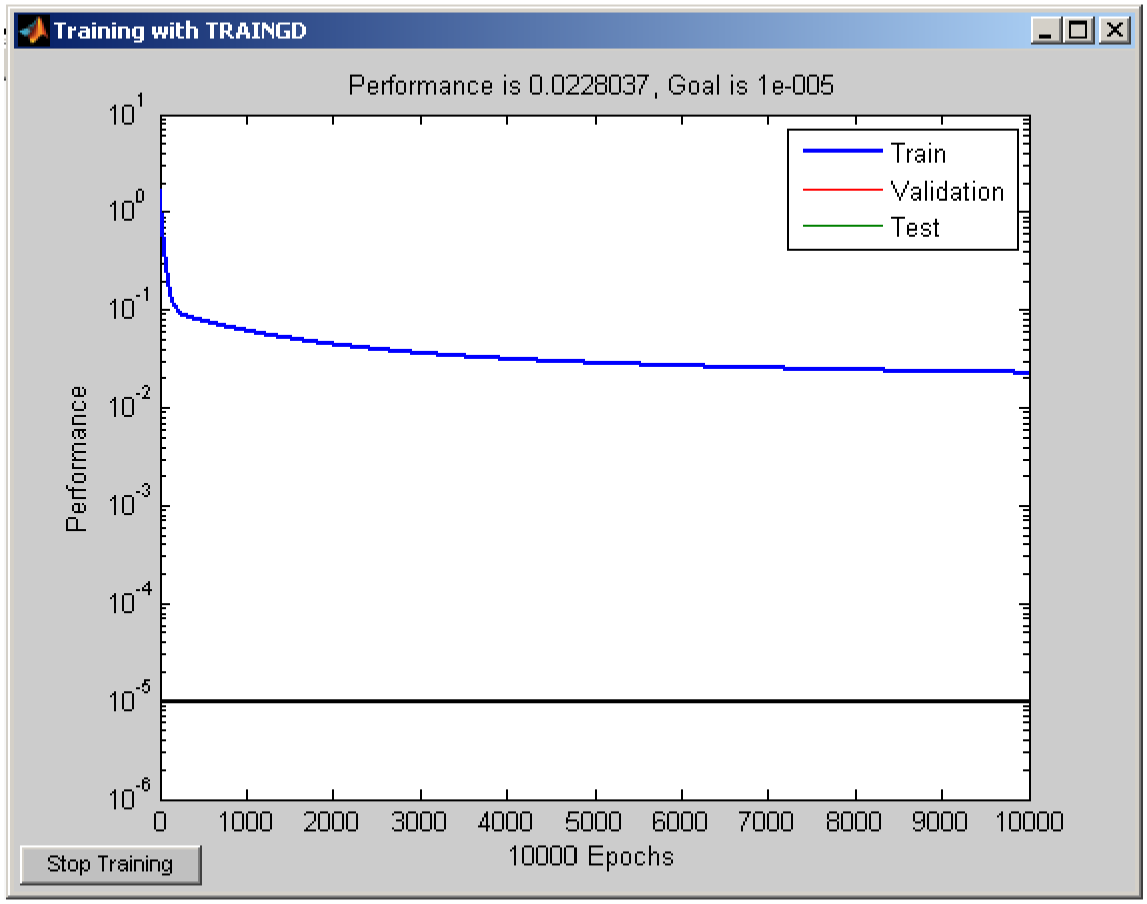
However, with a large number of neurons, the generalization performance of the MLP would be affected. That is, it overfits the training data and performs below par on unseen data.

<u>A note on classification:</u>

For classification purposes, however, such high performance in terms of MSE would not be required, since we usually set a threshold in the decision function to decide for the classes. As an example, if the two class indices are +0.8 and -0.8 for class1 and class2 respectively, then when the output of the MLP is greater than 0.4, we can decide class1. Otherwise, we can decide class2.

<u>Choice of activation functions:</u>

With a linear activation function in the output layer instead of a sigmoidal function, we can obtain much faster learning of the MLP.



MLP with linear output activation achieves a much better MSE

Shown above is the training performance curve for MLP with linear activation in the output layer. This MLP achieves a 0.02 MSE in 10000 epochs whereas the MLP with sigmoidal activation could reach only an MSE of 0.33

Some Insights:

Support vector machines prove as excellent candidates for classification exploiting the relative properties "relationship/distance" between the datapoints rather than the absolute values of the data like the perceptron. In fact, only a subset of the training data that qualify as support vectors by satisfying

$$\alpha_i > 0$$

are used for making a decision on the training data.

-- THANK YOU---

Author: Manikandan Bakthavatchalam