MACHINE LEARNING ASSIGNMENT 4

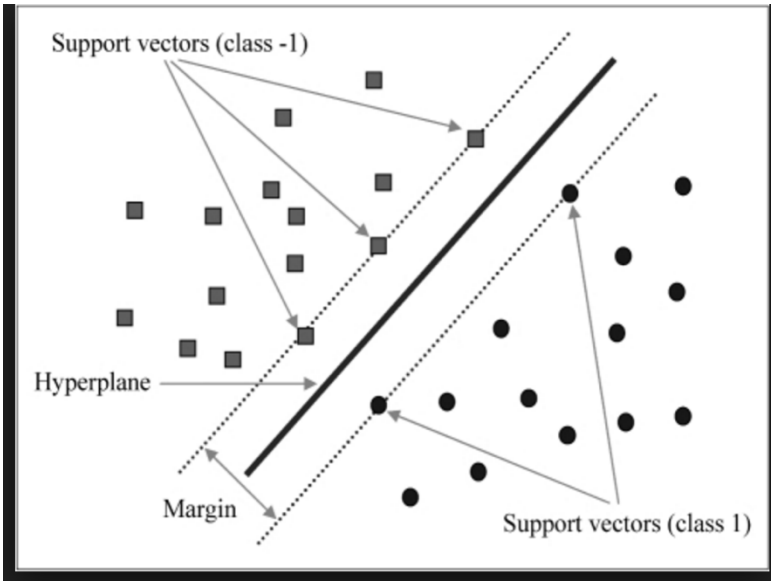
PRATEEK AGRAWAL

A20358932

Table of Contents

Problem Statement

We are trying to solve the problem of classification using Support Vector Machine. Classification is the problem of identifying a class for a given feature vector. We use different dataset to train our machine for this task. The basic approach that we follow here is drawing a line between different data points discriminating one class from other. SVM is different from Discriminative learning in the sense that it generalizes better as it tries to draw a line for the clusters are far as possible.



In the above diagram the SVM tries to maximize the margins and identify the support vectors. These support vectors will be used for the purpose of classifying a new example.

Following are the datasets used:

- Banknote authentication data set from UCI data base.
- Artificial linearly separable data set is created.
- Artificial non linearly separable set is created.
- Balance scale date set from UCI date base.

Proposed Solution

As shown in the above diagram our main aim here is to identify the support vectors and dram a hyper plane to discriminate the 2 labeled classes. The line drawn to discriminate the classes has the following equation.

$$d(X) = Wt * X + W°$$

Here $W°$ is the bias or the distance from the origin. 'Wt' is the normal to the discriminating line. Our main aim here is to maximize the geometrical margin shown the above diagram. To do this we minimize the primal problem given below.

$$Lp = \frac{1}{2} tranpose(W) * W$$

Subjected to the constraint that $y(i)(Wt * X(i) + W°) > 1$

We redefine $Lp = \frac{1}{2} tranpose(W) * W - \sum_{i=1}^{m} \alpha i(y(i)(Wt * X(i) + W°) - 1)$

Here αi is the LaGrange multiplier and is defines as the penalty we give for each wrong classification.

Next we minimize *Lp* with respect to W and W0, and then maximize the result with respect to αi. Taking partial derivative of *Lp* with respect to W and W0 we get the following results.

$$W = \sum_{i=1}^{m} \alpha i * yi * xi$$

$$\sum_{i=1}^{m} \alpha i * yi = 0$$

Substituting W is the equation of *Lp* we get,

$$Ld = -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha i * \alpha i * yi * yj * treanpose(Xi) * Xj + \sum_{i=1}^{m} \alpha i$$

Now is maximize this *Ld with respect to* αi using the Qp solver. After solving this we get all the values of the αi. Using the value to αi we identify all the support vectors. All the other points should have a value equal to zero but support vectors would be greater than zero. Using this αi we can compute the value of Ws.

Same can be used to compute the W0 with the following formulae.

$$W° = \frac{1}{\#SVs} \sum_{X \, \epsilon \, SVs} (Yi - transpose(W) * Xi)$$

Using these values, we classify a new given feature vector.

$$\text{If: } Wt * X + W° > 0, \text{ we classify as class 1 else class 0}$$

The algorithm discussed above is known as hard margin. We can also give some slack to the classifier and decrease the penalty it gives every time a wrong classification is done. For doing this our basis equation changes to:

$$Lp = \frac{1}{2} transpose(W).W + C \sum_{i=1}^{m} Si$$

Here C is the weight we give to each slack variable and Si is the slack variable for each example.

For classifying data were the examples are not separable with a linear function we use the same approach but in addition add the kernel trick so that we do not have to map the examples to the higher dimension.

$$Xi * Xj = \emptyset(Xi) * \emptyset(Xj) = k(Xi, Xj)$$

Here the function k is the kernel function and gives us the similarity between two vectors. Several kernel functions available are radial, linear, polynomial, Gaussian etc. In this report we have used Polynomial and Gaussian.

Implementation Details

Algorithm discussed above on 4 data set. Two of the data sets are artificially prepared and one is linearly separable while the other is non linearly separable. Other two data sets are taken from UCI data repository and deals with bank note authentication and balance scale. Following are the functions used.

- **Plot_data** is used to plot the artificially prepared 2D datasets.
- **Plotting_SVs** is used to plot the support vectors identified.
- **Norm** is used to normalize the data with mean = 0 and standard deviation = 1.
- **Prediction** functions are used to predict the labels for the testing data for linear, Gaussian and polynomial SVMs.
- **Accuracy** is used to calculate the accuracy produced on the testing set.
- **Training** functions are used to train the testing data for linear, Gaussian and polynomial SVMs.
- **Gram_matrix_gaussian** creates the gram matrix using the Gaussian similarity function.

- **Gram_matrix_poly** creates the gram matrix using the polynomial similarity function.
- **Linear_SVM, Gaussian_SVM and poly_SVM** takes the data and divide it into training and testing data, performing the same on all the data.

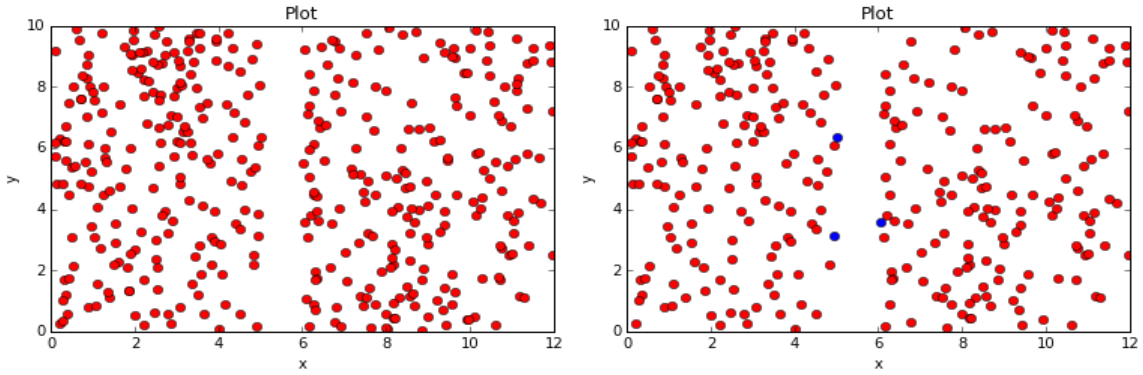
Data sets are prepared and appropriate function are called with different parameters to find the best accuracy. Artificially created data sets are formed and pickled for further used. Following are the link to the datasets.

- **/data/linear** for linearly separable data.
- **/data/non-linear** for non-linearly separable data.
- Bank authentication data set from UCI is available at **http://archive.ics.uci.edu/ml/machine-learning-databases/00267/data_banknote_authentication.txt**
- Balance scale data set from UCI is available at **http://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/balance-scale.data**
- To test the algorithm for data set for significant amount of one class data we remove some rows in the Bank authentication data set with 1's.

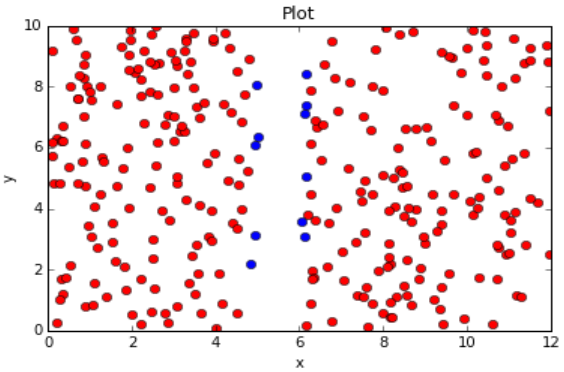
Result and Discussions

Linear SVM and Hard margin

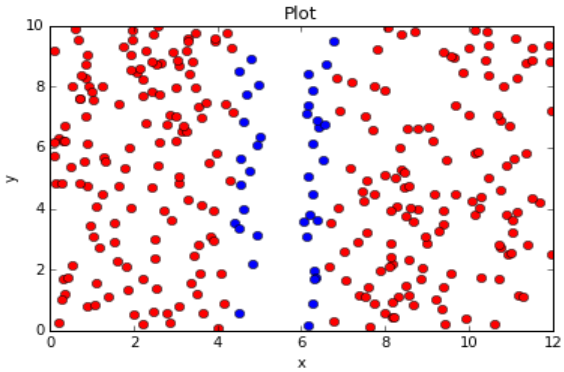
- Linear SVM was performed on the artificially prepared data set.
- Plot of the linearly separable data.



e = 0.1, c = 99999, acc = 100%

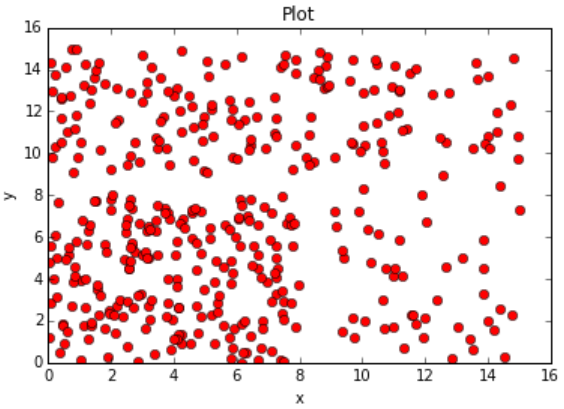


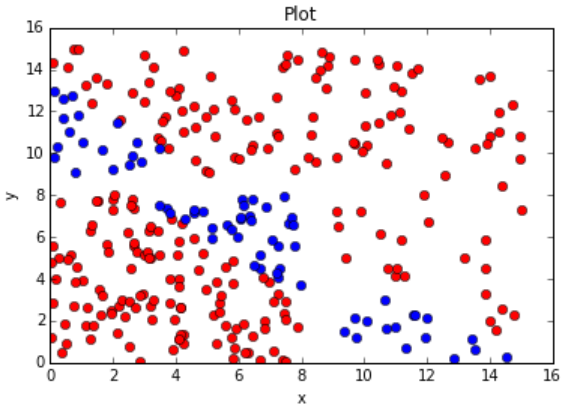
e = 0. 0000001, c = 99999, acc = 100%



e = 0. 00000001, c = 99999, acc = 100%

- As the value of e decreases the nu,ber of support vector incerases and finally overfits the data.
- Non separable data set.

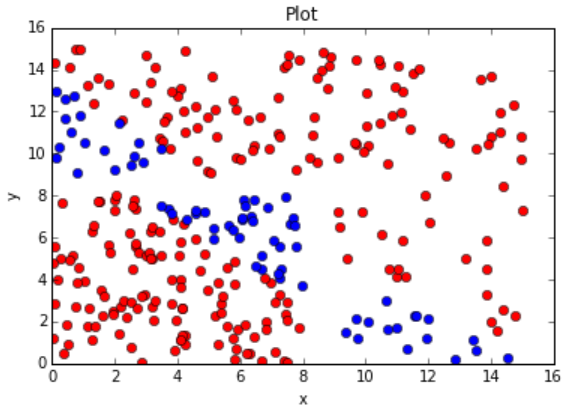




e = 0. 1, c = 99999, acc = 95%

Linear SVM with Soft margin

- Using soft margin does not make any change in the linearly separable dataset.
- Non-linearly separable data set.

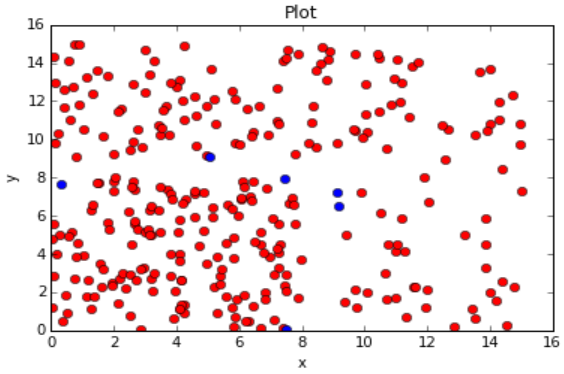


e = 0. 1, c = 1, acc = 95%

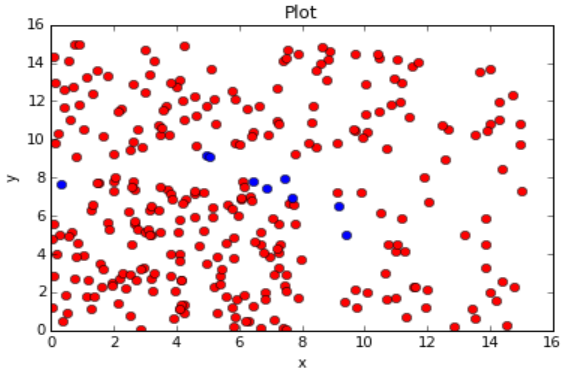
- Soft Margin with linear SVM could not help incrreasing the accuracy in the case of Non Separable data.

Gaussian Kernel SVM

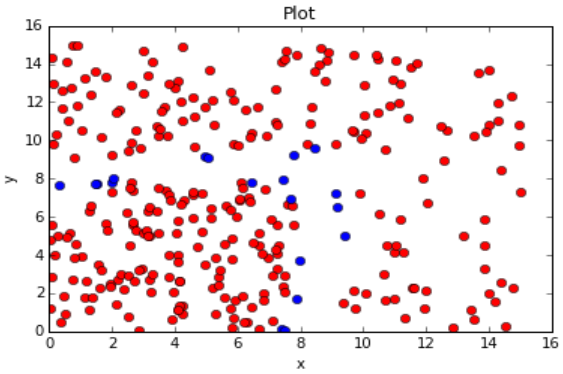
- Non Linearly separable data set.



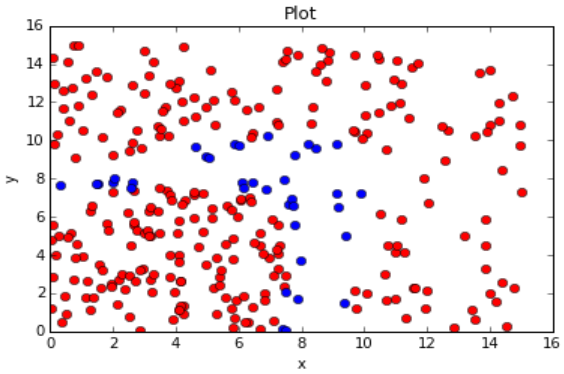
e = 0. 1, c = 99999, acc = 50%



e = 0. 1, c = NAN, acc = 64%



e = 0.1, c =99 , acc = 50%



e = 0.000001, c =NAN , acc = 100%

Data set	Epsilon	C	SVs	Accuracy
NL separable	0.1	NAN	6	50
NL separable	0.1	999	7	50
NL separable	0.1	99	17	50
NL separable	0.1	10	42	51
NL separable	0.1	5	54	60
NL separable	0.1	4	58	76
NL separable	0.1	3.95	58	77
NL separable	0.00001	3.85	61	80
NL separable	0.00001	3.79	61	82
NL separable	0.000001	NAN	36	100
Banknote	0.1	NAN	14	46.3
Banknote	0.1	99	15	53.6
Banknote	0.1	75	15	58.01
Banknote	0.1	70	19	70.5
Banknote	0.1	65	19	81.3
Banknote	0.1	60	19	88.9
Banknote	0.1	55	19	96.2
Banknote	0.1	50	20	46.93
Banknote				
Banknote	0.0001	NAN	18	46.35
Banknote	0.0001	9999	15	100.00
Banknote	0.0001	999	18	46.355
Balance	0.1	NAN	24	49.3
Balance	0.1	99	22	65.97
Balance	0.0000001	NAN	30	50.6
Balance	0.0000001	875000	113	81.94
Balance	0.00000001	NAN	107	76.38
Balance				

- As the value of E decreases we get more and more support vectors.
- Also, we need to move more and more towards softer margin by decreasing the value of C to get better accuracy.
- Decreasing C further decreases the accuracy as the data gets over fit.

Polynomial Kernel SVM

- Non linearly separable data set

Data set	Epsilon	C	SVs	Accuracy
Non separable	0.1	NAN	7	50

Non separable	0.1	1	13	55
Non separable	0.1	0.9	15	83
Non separable	0.1	0.8	15	79
Non separable	0.000000001	NAN	9	50
Non separable	0.000000001	0.1	95	100
Banknote	0.1	NAN	9	45.1
Banknote	0.1	0.2	21	51.020
Banknote	0.01	NAN	12	54.8
Banknote	0.01	0.2	27	59.7
Banknote	0.000000001	NAN	25	53.93
Banknote	0.000000001	9999	26	62.68
Banknote	0.0000000001	NAN	134	45.189
Balance	0.1	NAN	-	51.0
Balance	0.01	0.2	38	61.0
Balance	0. 0000001	0.2	43	93.05

- The best accuracy was achieved with Epsilon as 0.000000001 and c as 0.1 and 0.2 for the no separable and balance dataset.
- Epsilon and C follow the same relation as in the Gaussian Kernel SVM.
- I concluded that Gaussian SVM preforms better than the polynomial function.
- Decreasing the value of Epsilon and the slack variable over fits the model.

References

- Stackoverflow.com
- Pythonprogramming.net
- Coursera.com
- Element of statistical learning.

3.) Primal objective (L_p) for soft-margin is given by:-

$$L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i (y^i(w^T x_i + w_0) - 1 + \xi_i) - \sum_{i=1}^{m} \beta_i \xi_i$$

⇒ To minimized L_p we differenciate L_p with respect to the variables w_1, w_0 and ξ_i

$$\Rightarrow \frac{\partial L_p}{\partial w_1} = 0 \Rightarrow \frac{\partial L_p}{\partial w_1} = \frac{1}{2}\cdot 2\cdot w + 0 - \sum_{i=1}^{m} \alpha_i \cdot y^i \cdot x^i$$

$$\Rightarrow \boxed{w = \sum_{i=1}^{m} \alpha_i \cdot y^i \cdot x^i}$$

$$\Rightarrow \frac{dL_p}{dw_0} = -\sum_{i=1}^{m} \alpha_i \cdot y^i = 0 \Rightarrow \boxed{\sum_{i=1}^{m} \alpha_i \cdot y^i = 0}$$

$$\Rightarrow \frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \Rightarrow \boxed{C = \alpha_i + \beta_i}$$

Now, we substitute the value of w in the equation of L_p; we get:-

$$\Rightarrow L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i (y^i(w^T x_i + w_0) - 1 + \xi_i) - \sum_{i=1}^{m} \beta_i \xi_i$$

$$= \frac{1}{2}\left(\sum_{i=1}^{m} \alpha_i y^i x^{iT}\right)\left(\sum_{j=1}^{m} \alpha_j \cdot y^j \cdot x^{jT}\right) + C\cancel{\sum_{i=1}^{m} \xi_i}^{0} \# - \sum_{i=1}^{m} \alpha_i y^i \left(\sum_{j=1}^{m} \alpha_j \cdot y^j \cdot x^{jT}\right) x^i$$

$$- \cancel{\sum_{i=1}^{m} \alpha_i \cdot y^i \cdot w_0}^{0} + \sum_{i=1}^{m} \alpha_i - \cancel{\sum_{i=1}^{m} \alpha_i \cdot \xi_i}^{0} - \cancel{\sum_{i=1}^{m} \beta_i \cdot \xi_i}^{0}$$

Some of terms equate to zero, we are left with:-

$$\boxed{L_D = \frac{-1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \cdot \alpha_j \cdot y^i \cdot y^j \cdot x^{iT} \cdot x^j + \sum_{i=1}^{m} \alpha_i}$$

⇒ This is called the Dual objective and is maximized further.