# Programming Assignment 6: Support Vector Machines

Srihari Akash Chinam
USC ID - 2953497706
Net ID - chinam

Sayan Nanda
USC ID - 2681592859
Net ID - snanda

## Implementation

- Language used - Python version 2
- Libraries used - Numpy (arrays and matrices), SciKit-Learn, scipy, matplotlib.pyplot

### Support Vector Machines - Linearly Separable Case

Using the 100 2D data points provided, alpha values were maximized. The maximization was done using an in-built optimizer from the library scipy. The equation maximized to find alpha is shown below

$$-\frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s (\boldsymbol{x}^t)^T\boldsymbol{x}^s + \sum_t \alpha^t$$

with the constraints as

$$\sum_t \alpha^t r^t = 0, \text{ and } \alpha^t \geq 0, \forall t$$

After alpha was computed, the weights of the equation were calculated based on alpha, labels and the data points using the equation below.

$$\boldsymbol{w} = \sum_t \alpha^t r^t \boldsymbol{x}^t$$

After the weights matrix w (w1,w2) was calculated, the constant w0 was calculated using the equation shown below.

$$w_0 = r^t - \boldsymbol{w}^T\boldsymbol{x}^t$$

To calculate w0 on of the support vectors was used. The values obtained for w and w0 are shown below in Table 1.

**Table 1** Weights obtained for Linearly Separable Case

| w1 | w2 | w0 |
|---|---|---|
| 7.2495774 | -3.86186094 | -0.106872871116 |

This made the equation of the line obtained as

$$7.2495774x1 - 3.86186094x2 = 0.106872871116$$

The same data was passed to the SVM implementation of SciKit-Learn, for which the weights obtained are shown below in Table 2.

**Table 2** Weights obtained for Linearly Separable Case by SciKit-Learn

| w1 | w2 | w0 |
|---|---|---|
| 3.04461355 | -1.6886085 | 0.12013405 |

These equations were plotted along with the Data points and the results are shown in Figure 1 below. The black line represents the output obtained by our implementation while the green line represents the result of SciKit-Learn's SVM. The difference of results occurs as SciKit-Learn's implementation uses a Soft Margin approach with a C value.

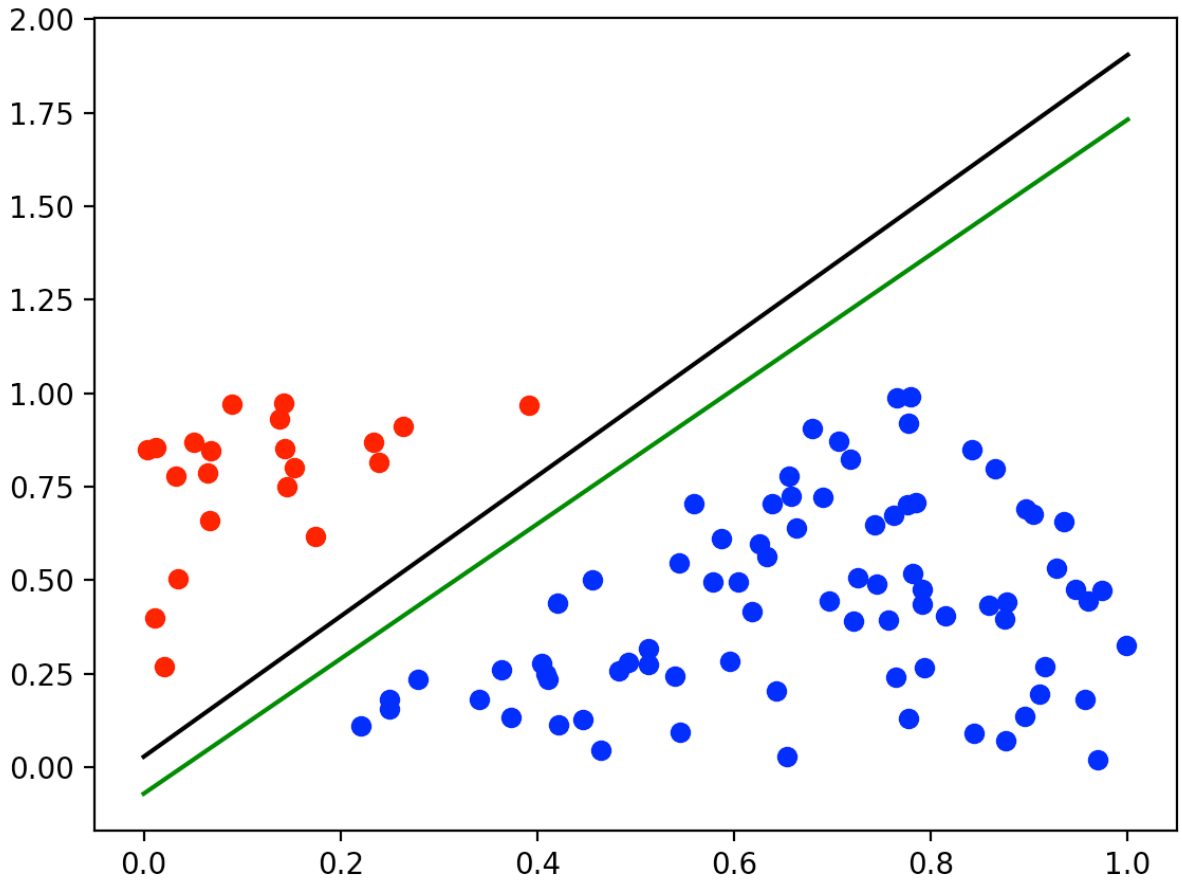


**Figure 1** SVM outputs by our implementation and SciKit-Learn's implementation

To get a solution similar to sklearn, we introduced a soft margin approach with an upper bound C on alpha, with C=1.2. This gave a very similar result to sklearn, with the weights shown in Table 3 and the new output plotted in Figure 2.

**Table 3** Weights obtained for Linearly Separable Case with Soft Margin Approach

| w1 | w2 | w0 |
|---|---|---|
| 3.83875754 | -2.16437733 | 0.14690175781 |

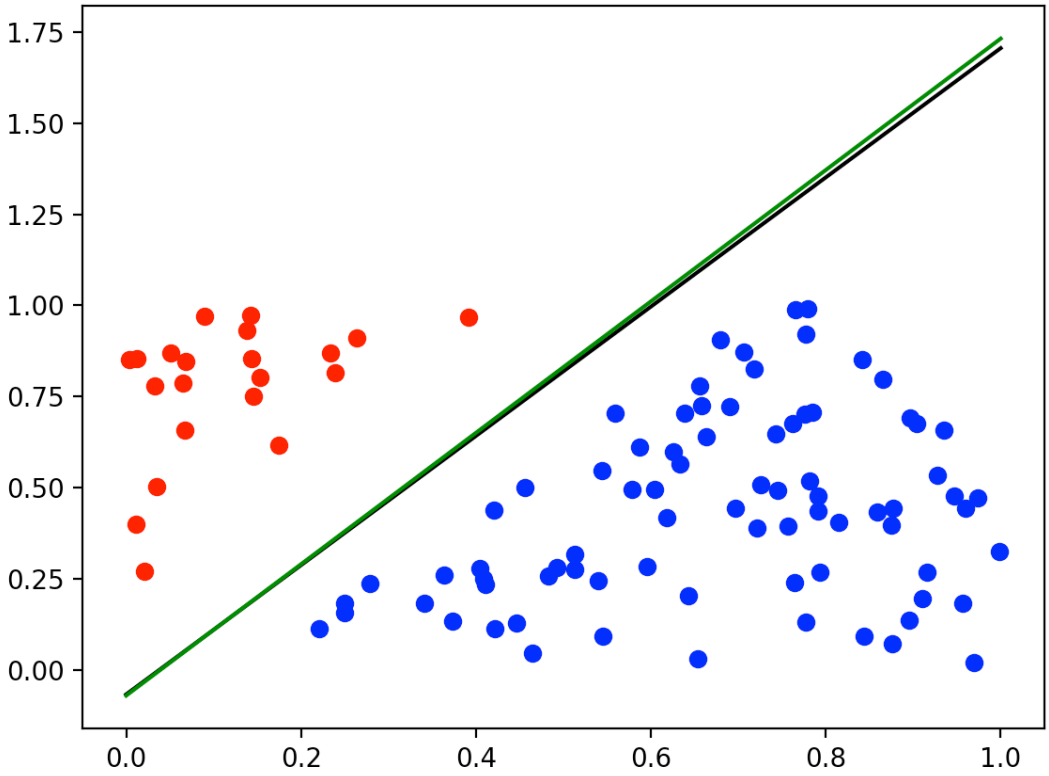


**Figure 2** Output for Soft Margin approach compared to SciKit-Learn's implementation

With the new weights, the equation of the line becomes

$$3.83875754*x1 - 2.16437733*x2 + 0.14690175781 = 0$$

## Support Vector Machines - Linearly Inseparable Case

Using the 100 2D points provided, the alpha values were maximized. The equation used to maximize alpha is shown below.

$$L_d = \sum_t \alpha^t - \frac{1}{2} \sum_t \sum_s \alpha^t \alpha^s r^t r^s K(\boldsymbol{x}^t, \boldsymbol{x}^s)$$

where K(xt,xs) is the kernel function chosen for the SVM. In this case, that is a degree 2 polynomial. The basis function $\phi(x)$ used in this case is shown below.

$$\phi(\boldsymbol{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]$$

Since there are 6 terms involved, 6 weights were collected for it. A constant (w0) was added to provide a more controlled intercept for the curve generated. After multiple initializations, the best curve was obtained with the weights shown in Table 4.

**Table 4** Weights obtained for Linearly Inseparable case

| w1 | w2 | w3 | w4 | w5 | w6 | w0 |
|---|---|---|---|---|---|---|
| -8.773 | 85.694 | -88.695 | 612.569 | -591.843 | -634.022 | 130252.124 |

When tested against the training data, the accuracy of this curve was 93%. After plugging the weights along with the basis function mentioned above, the equation of the line generated is as follows:

(130243.35121388) + (85.69403744*1.414*x) - (88.695019*1.414*y) +
(612.56998646*1.414*x*y) - (591.84378044*x*x) - (634.0220185*y*y) = 0

Upon simplifying this equation, we get the final equation as:

x*x - 1.46352*x*y - 0.204735*x + 1.07127*y*y + 0.211905*y - 220.064 = 0

When plotted, it is seen that this equation represents an ellipse as shown in Figure 3 below. The data that this curve covers is shown as a scatter plot in Figure 4.

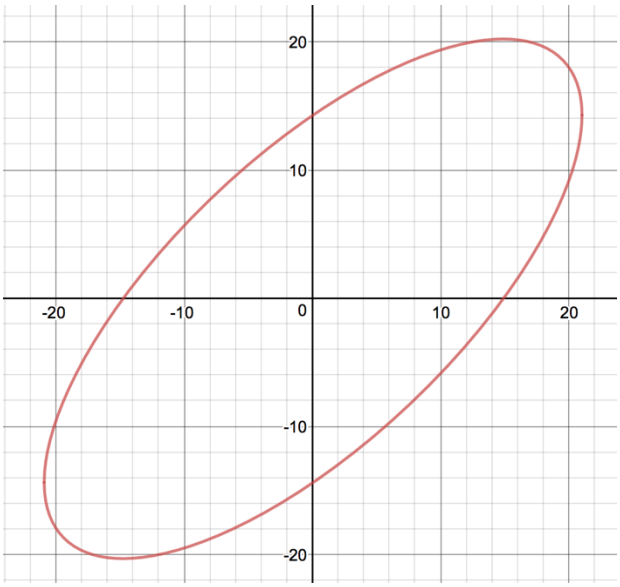


**Figure 3** Ellipse generated by Polynomial Kernel implementation

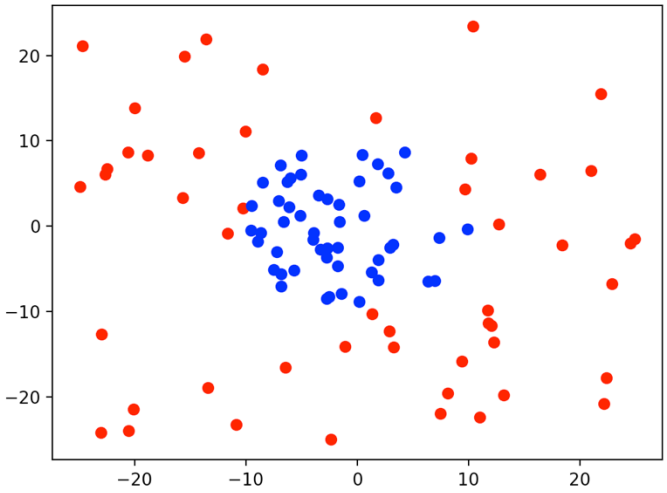


**Figure 4** Scatter plot of data provided for Linearly Inseparable Case

When SciKit-Learn's SVM was trained using a Polynomial kernel of degree 2, with the same data, the 4 weights were obtained along with one intercept. These weights are shown in Table 5 below. When tested against the training data, the SVM from SciKit-Learn scored a 100% accuracy.

**Table 5** Weights and intercept obtained using SciKit

| w1 | w2 | w3 | w4 | w0 |
|---|---|---|---|---|
| -0.04937408 | -0.08663716 | 0.109537 | 0.02647424 | -18.92792843 |

We also implemented the RBF kernel for the Linearly Inseparable case, with the kernel function being calculated as:

$$K(\boldsymbol{x}^t, \boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x}^t - \boldsymbol{x}\|^2}{2s^2}\right]$$

For this kernel, we obtained one weight and one intercept, shown below in Table 6.

**Table 6** Weights obtained with RBF kernel

| w1 | w0 |
|---|---|
| -0.22628938 | 1.01088714e-42 |

This made the equation of the curve as:

$$-0.22628938*K(xt,x) + 1.01088714e\text{-}42 = 0$$

where K(xt,x) is the kernel function for RBF mentioned previously. When tested against the training data, this curve performed with 96% accuracy.

# Software Familiarization

The python library Scikit learn offers an implementation for Support Vector Machines (SVM). To allow for the soft version of SVMs the library implementation allows the user to specify

the error term. It gives several kernel options such as radial basis function, polynomial function whose degree may be specified and a sigmoidal function. The kernel function requires a gamma value and an independent value (w0) to be specified. This is a parameter needed by some kernels.

Other options offered by the library implementation are a probability estimate, a shrinking heuristic, tolerance for a stopping criterion. The user can also specify the maximum number of iterations as well as whether the data should be taken in a random order or not. Other options include specifying the cache size and assigning class weights.

Some of the ideas from the scikit learn implementation that we can use for our implementation are including a tolerance for the stopping criterion. For the linear dataset we did include an option to include tolerance. This gave us the same result as without the tolerance. We can also take the data in a random order as well as experiment a bit with initial values of the weights. This may allow us to reach an optimized value faster. We could also have mentioned the maximum number of iterations.

# Application

Support Vector Machines are used in text and hypertext categorization. This reduces the requirement for labeled training. They are useful in many applications which earlier used Neural networks. This includes classification of images, image segmentation systems, recognizing handwriting and other such applications.

SVMs have shown promising results in bioinformatics research. They may be prove to be relevant for functional genomics and chemogenomics projects. SVM models were used to identify small organic molecules that potentially modulate the function of G-protein coupled receptors. The classifier can be used for fast filtering of compound libraries in virtual screening applications [1].

Another interesting application is in mechanical faults diagnostic. The mechanical vibrations are used along with SVMs to formulate a new intelligent method to diagnose faults. The method is used for on-line monitoring without the need for intervention from a maintenance specialist. The sensors are placed and it is found that the vibration signals occur from different directions depending on the type of fault. The optimal position for signal acquisition has also been studied [2].

# Individual Contributions

## Sayan Nanda

- Implementation of models
    - Preprocessing Data
    - Linearly Separable Case
    - Linearly Inseparable Case - RBF Kernel
- Documentation

    - Software Familiarization
    - Applications

### Srihari Akash Chinam

- Implementation of models
    - Preprocessing Data
    - Linearly Separable Case
    - Linearly Inseparable Case - Polynomial Kernel
- Documentation
    - Implementation

# References

[1] Support vector machine applications in bioinformatics, http://europepmc.org/abstract/med/15130823

[2] SVM practical industrial application for mechanical faults diagnostic, http://www.sciencedirect.com/science/article/pii/S0957417410013801