Support Vector Machines - Binary Classification

Project Report

Taha Adeel Mohammed CS20BTECH11052, Shambhu Kavir CS20BTECH11045

Abstract*— *Support Vector Machines (SVMs) are a supervised machine learning technique utilized for classification and regression analysis. It can be viewed as a convex optimization problem, where the goal is to find the hyperplane which maximizes the margin between the two classes of data points. In the case of non-linearly separable data, we introduce the concept of slack variables(error) to allow for some misclassification (soft-margin SVM). In this report, we will be focusing on the theory of SVMs and analysis of the underlying convex optimisation problems. We also implement the SVM algorithm from scratch using the CVXPY library and apply it to our sample dataset.

I. Origin of the Problem

The basic idea of any (Supervised) Machine Learning model is to predict a certain output 'y' for an input 'x' correctly, when a certain amount of training data is given.

For our classification problem, given a set of vectors $(\vec{x_1}, \vec{x_2}, ..., \vec{x_m})$, where $\vec{x_i} \in \mathbb{R}^n$, and a set of labels $(y_1, y_2, ..., y_m)$, we need to find a function f such that $f(\vec{x_i}) = y_i$ for all $i = 1, 2, ..., m$.

The SVM tries to solve this problem by assuming f to be a separating hyperplane (assuming the input is linearly separable). It operates on the binary classification problem class, i.e., each y_i can be either -1 or 1. More specifically, the SVM tries to find a plane: $\vec{w}^T\vec{x} + b = 0$ such that

$$f(\vec{x_i}) = \begin{cases} 1 & \text{if } \vec{w}^T\vec{x_i} + b \geq 0 \\ -1 & \text{if } \vec{w}^T\vec{x_i} + b < 0 \end{cases} \tag{1}$$

We can theoretically find a separating hyperplane by solving the following feasaibility problem:

$$y_i(\vec{w}^T\vec{x_i} + b_i) \geq 0, \forall\ i = 1, 2, ..., m$$

The problem with setting the following constraints is that the values $\vec{w}, b = (\vec{0}, 0)$ statisfy the constraints and the solver will return these value to us. This will yield $(\vec{w}, b) = (\vec{0}, 0)$.

However, for linearly separable data, since a non-trivial solution $(\vec{w}, b)$ exists, $(k\vec{w}, kb)$ would also be a solution for any $k > 0$. Hence we can update our constraints to the following without affecting the existence or validity of the solution:

$$y_i(\vec{w}^T\vec{x_i} + b_i) \geq 1, \forall\ i = 1, 2, ..., m$$

To remove the redundancy of multiple solutions, we can add the following constraint:

$$\min_{i=1...m} |\vec{w}.\vec{x} + b| = 1$$

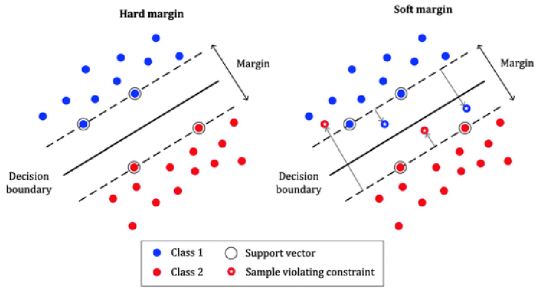

Fig. 1: Hard-margin and soft-margin SVMs

Continuing from the above normalisation, we will get two hyperplanes, as show in Fig. 1, such that (for some i, j):

$$\vec{w}.\vec{x_i} + b = \quad 1$$
$$\vec{w}.\vec{x_j} + b = -1$$

These are two parallel hyperplanes and the distance between them, called the **maximal margin** is given by

$$\text{maximal margin} = \frac{2}{\|\vec{w}\|} \tag{2}$$

Now we frame our optimisation problem under the hypothesis that a "good" classifier will maximise the "margin" between the two classes of data.

$$\vec{w}^* = \arg\max_{\vec{w},\ b} \frac{2}{\|\vec{w}\|} \tag{3}$$
$$\text{s.t. } y_i(\vec{w}.\vec{x} + b) \geq 1, \quad \forall\ i = 1...m$$

This can also be framed as

$$\vec{w}^* = \arg\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \tag{4}$$
$$\text{s.t. } y_i(\vec{w}.\vec{x} + b) \geq 1, \quad \forall\ i = 1...m$$

The above **hard-margin SVM** is a **convex optimisation problem**, specifically, a quadratic problem with linear constraints.

In order to overcome the hurdle of linearly inseparable vectors, we allow some vectors to be misclassified by allowing some error ξ_i for each vector $\vec{x_i}$. The new **convex optimisation problem** is called the **soft-margin SVM** problem and is given by

$$\vec{w}^* = \arg\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 + C\left(\sum_{i=1}^{m} \xi_i\right)^k \tag{5}$$
$$\text{s.t. } y_i(\vec{w}.\vec{x} + b) \geq 1 - \xi_i, \quad \forall\ i = 1...m$$
$$\xi_i \geq 0, \quad \forall\ i = 1...m$$

II. Analysis of the problem

A. Hard-margin SVM

The hard margin SVM convex optimization problem, as we've seen above, is given by

$$\vec{w}^* = \arg\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \quad \text{s.t. } y_i(\vec{w}.\vec{x}+b) \geq 1, \quad \forall\, i = 1...m \tag{6}$$

Now this can be solved using the Lagrangian method which is given by

$$L(\vec{w}, b, \vec{\lambda}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \lambda_i (y_i(\vec{w}.\vec{x}+b) - 1) \tag{7}$$

Here λ is the vector of non negative Lagrange multipliers. Differentiating the Lagrangian with respect to $\vec{w}$ and b and equating to zero, we get

$$\frac{\partial L(\vec{w}, b, \vec{\lambda})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{m} \lambda_i y_i \vec{x}_i = 0 \tag{8}$$

$$\frac{\partial L(\vec{w}, b, \vec{\lambda})}{\partial b} = \sum_{i=1}^{m} \lambda_i y_i = 0 \tag{9}$$

We impose the condition $\vec{\lambda}.\vec{y} = 0$ and $\lambda_i \geq 0$ and substitute the value of w to get the dual problem

$$\vec{\lambda}^* = \arg\max_{\vec{\lambda}} \ \vec{\lambda}\cdot\vec{1} - \frac{1}{2}\vec{\lambda}^T D \vec{\lambda} \tag{10}$$

$$\text{s.t. } \vec{\lambda}\cdot\vec{y} = 0, \quad \vec{\lambda} \geq \vec{0}$$

$$\text{where } D_{ij} = y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{11}$$

The complementary slackness conditions are given by:-

$$\lambda_i (y_i(\vec{w}.\vec{x}+b) - 1) = 0, \forall i = 1...n \tag{12}$$

From the above equation, we can infer that the points in the dataset corresponding to $\lambda_i > 0$ are forced to be on the margin. These points are called the **support vectors**. The points corresponding to $\lambda_i = 0$ are not on the margin and do not affect the solution.

Once we solve the KKT conditions to obtain the values of the Lagrange multipliers $\vec{\lambda}$, the parameters of the hyperplane $(\vec{w^*}^T \vec{x} + b^* = 0)$ can be obtained as

$$\vec{w}^* = \sum_{i=1}^{m} \lambda_i^* y_i \vec{x}_i \tag{13}$$

$$b^* = y_i - \vec{w}^* \cdot \vec{x}_i \tag{14}$$

B. Soft-margin SVM

We now consider the case when the data is not linearly separable. In such a case, we need to allow some points to be on the wrong side of the margin. We introduce a slack variable ξ_i for each point in the dataset. The slack variable is the distance by which the point is on the wrong side of the margin. The optimisation problem is given by:-

$$\vec{w}^* = \arg\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 + C\left(\sum_{i=1}^{m} \xi_i\right)^k \tag{15}$$

$$\text{s.t. } y_i(\vec{w}.\vec{x}+b) \geq 1 - \xi_i, \quad \forall\, i = 1...m$$

$$\xi_i \geq 0, \quad \forall\, i = 1...m$$

Here we allow our classifier to commit ξ_i error for sample i, but we penalize the total error using the addition of the second term in the objective function. We can control C and k so as to control how we want to penalize the error.

Therefore, the Lagrangian for the soft-margin problem is given by

$$L(\vec{w}, b, \vec{\lambda}, \vec{\mu}) = \frac{1}{2}\|\vec{w}\|^2 + C(\sum_{i=1}^{n} \xi_i)^k - \sum_{i=1}^{n} \lambda_i (y_i(\vec{w}\cdot\vec{x}+b) - 1 + \xi_i) - \sum_{i=1}^{n} \mu_i \xi_i \tag{16}$$

We need to minimize the Langrangian with respect to $\vec{w}, b, \xi_i$ and maximize it with respect to $\vec{\lambda}, \vec{\mu}$. Differentiating the Langarangian and equating to zero, we get

$$\frac{\partial L(\vec{w}, b, \vec{\lambda}, \vec{\mu})}{\partial \vec{w}} = \vec{w} - \sum_{i=1}^{n} \lambda_i y_i \vec{x}_i = 0 \tag{17}$$

$$\frac{\partial L(\vec{w}, b, \vec{\lambda}, \vec{\mu})}{\partial b} = \sum_{i=1}^{n} \lambda_i y_i = 0 \tag{18}$$

$$\frac{\partial L(\vec{w}, b, \vec{\lambda}, \vec{\mu})}{\partial \xi_i} = Ck(\sum_{i=1}^{n} \xi_i)^{k-1} - \lambda_i - \mu_i = 0 \tag{19}$$

When $k > 1$, we have:-

$$\sum_{i=1}^{n} \xi_i = \left(\frac{\lambda_i + \mu_i}{kC}\right)^{\frac{1}{k-1}} \tag{20}$$

Substituting these in the Lagrangian, we get

$$L(\vec{\lambda}, \vec{\mu}) = \vec{\lambda}.\vec{1} - \frac{1}{2}\vec{\lambda}.D\lambda \tag{21}$$

From the above equation, we get the dual convex optimization problem as

$$\max_{\vec{\lambda},\vec{\mu}} L(\vec{\lambda}, \vec{\mu}) = \vec{\lambda}\cdot\vec{1} - \frac{1}{2}\vec{\lambda}^T D\vec{\lambda} - \frac{(\mu_i + \lambda_i)^{\frac{k}{k-1}}}{(kC)^{\frac{1}{k-1}}}\left(1 - \frac{1}{k}\right)$$

$$\text{s.t. } \vec{\lambda} \geq 0, \quad \vec{\mu} \geq 0, \quad \vec{\lambda}\cdot\vec{y} = 0$$

$$\text{where } D_{ij} = y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{22}$$

III. Code Implementation and Results

To implement the hard-margin and soft-margin SVM classifiers and solve the convex optimization problems, we use the `cvxpy` library in Python. To train the model, all dimensions of the input dataset are used. However, to visualize it, we use only the first two dimensions of the dataset and show the separating hyperplane.

Hard-margin SVM

Below code snippet shows setting the optimization problem and constraints for the hard-margin SVM classifier, and then solving it using the `cvxpy` library.

```
# Set up the objective function and constraints for the optimization problem
objective = cp.Minimize(-cp.sum(lambdas) + 0.5 * cp.quad_form(M @ lambdas, np.eye(M.shape[0])))
constraints = [lambdas >= 0, cp.sum(lambdas[0:50] - lambdas[50:100]) == 0]
problem = cp.Problem(objective, constraints)

# Solve the optimization problem and print the value of the Lagrange multipliers
problem.solve(solver=cp.ECOS, verbose=False)
print("Value of Lagrange multipliers:", lambdas.value)
```

Fig. 2: Code snippet for setting the optimization problem and constraints for the hard-margin SVM classifier

It outputs the following result for our input dataset. We can see that the SVM classifier is able to separate the two classes perfectly.

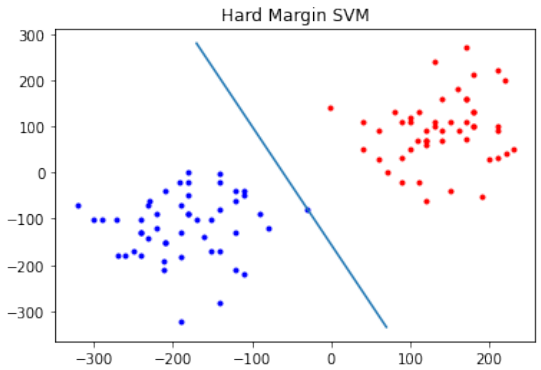

Fig. 3: Result of the hard-margin SVM classifier

Soft-margin SVM

The code for the soft-margin SVM classifier is similar to the hard-margin SVM classifier, except that we need to add the slack variables ξ_i and the corresponding constraints. The code snippet for the same is shown below.

```
# Define the objective function
k = 3
C = 0.1
objective = cp.Minimize( -cp.sum(lambd) + 0.5*cp.quad_form(lambd, D) + (lambd[0]+gamma[0])
**(k/(k-1))/(k * C)**(k/(k-1)) * (1-1/k))

# Define the constraints
constraints = [lambd >= 0, cp.sum(lambd[0:50] - lambd[50:100]) == 0, lambd <= (lambd[0]
+gamma[0]), lambd+gamma==C*np.ones(100)]

# Solve the problem using cvxpy
convexop_problem = cp.Problem(objective, constraints)
convexop_problem.solve(solver=cp.ECOS, verbose=False)
```

Fig. 4: Code snippet for setting the optimization problem and constraints for the soft-margin SVM classifier

It outputs the following result for our input dataset. We can see that the soft-margin classifier is able to separate the two classes with very high accuracy.

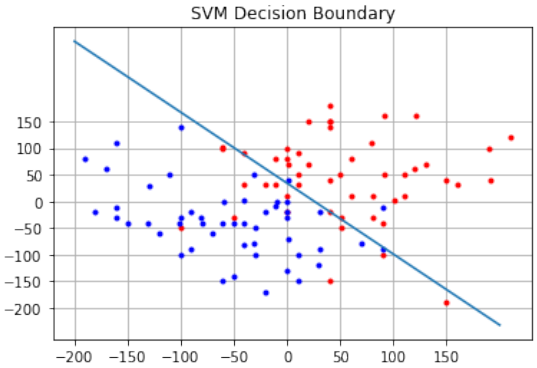

Fig. 5: Result of the soft-margin SVM classifier

The results of our model on the test dataset are shown below. We can see that the our model computes values greater than 1 for most of the blue points, and less than -1 for most of the red points. Hence, it is able to classify the points correctly.

```
SVM classifier values for blue data points:
[[ 1.98249106 -0.26897925 -0.38155281  3.31065908  0.48399125  2.02657667
  -0.2370037   3.22066998  3.36560215  0.75121093  1.06663422 -0.22794122
   1.32836704  4.59614624  0.73577341  2.20953511  2.62079451 -0.484853
   0.68653529  1.01169832  0.59670689 -0.91584294  3.3727204  -0.2826949
  -0.39447078  2.92643073  1.34454776  2.5685088   0.0528377   5.11836645
   1.12601955  0.17603202  2.99452556 -2.11383615  0.51544304 -0.22433264
  -0.01135006  4.17531496  0.91516341 -3.488595    3.95660453  0.03360939
   1.60876001  2.71874562  2.73527934 -2.08501733  2.28989646  0.97755619
  -0.88239243  5.87276459]]
SVM classifier values for red data points
[[-0.93378568 -2.0870275  -3.32451907 -2.63531315 -4.39449692 -0.06343493
  -0.55577306 -4.24956475 -1.30272149 -4.14112778 -0.33351282 -4.46241671
  -2.4779767  -3.34315343 -1.04222692 -1.82419131 -1.38577027 -2.20118513
  -3.97416786 -0.38973429 -2.75392741 -2.89990417 -2.72190458 -3.20041795
  -2.05445799 -2.79838462 -3.70459938 -2.48419533 -3.85663118  0.10971782
  -1.64693493 -0.22309865 -2.25928489 -3.15878597 -0.44201999 -2.83822163
  -0.86905555 -2.80066748 -4.72502336 -3.00888778 -2.68955026 -2.00146632
  -3.91406511 -1.92536506  1.20410949 -0.77073284 -1.82421711 -1.34092711
  -4.08475706 -4.2846065 ]]
```

Fig. 6: Performance of the model

IV. Conclusion

- In the field of machine learning, classification is one of the most fundamental tasks in data analysis. Therefore, hard and softt margin classifiers serve as a powerful tool for solving such problems.
- These classifiers aim to find the best hyperplane that can separate the input data into two different classifiers.
- This problem is interesting because it involves balancing two conflicting objectives:- maximizing the margin and minimizing the classification error.
- These classifiers have numerous applications such as image and speech recognition, text categorization, bioinformatics, etc.
- Further, this problem can be extended to handle non-linearly separable data using kernel methods. This involves mapping the input data to a higher dimensional space where it is linearly separable.

REFERENCES

[1] Support Vector Machines: Training and Applications by Edgar E. Osuna, Robert Freund, and Federico Girosi, MIT, 1997.

[2] Understanding Machine Learning: From Theory to Algorithms, Shai Shalev-Shwartz and Shai Ben-David, Cambridge University Press, 2014.

[3] Machine Learning Bert Huang, Virginia Tech, 2015.

[4] Convex Optimization, Stephen Boyd and Lieven Vandenberghe, Cambridge University Press, 2004.

[5] A Tutorial on Support Vector Machines for Pattern Recognition, Christopher J.C. Burges, Data Mining and Knowledge Discovery, 1998.