Contents

1 Introduction

Support Vector Machine (SVM) is one of the most powerful supervised machine learning algorithm. Most commonly used for solving classification problems and is also referred to as **Support Vector Classification**. There is also a subset of SVM called SVR which stands for **Support Vector Regression** which uses the same principles to solve regression problems. SVM is used for both linear separable data and non-linear separable data. For non-linear data, kernel functions are used basically helps in solving the non-linearity of the equation in a very easy manner.

2 Quintessential Topics for SVM

2.1 Decision Boundary

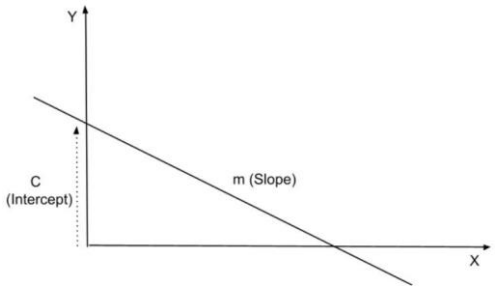
Decision Boundary is the main separator for dividing the points into their respective classes. Simply means to seperate the datpoints by means of line or plane known as **Hyperplanes**.

2.2 Hyperplane

A Hyperplane is a decision boundary that differentiates two classes in SVM. A datapoint falling in either side of hyperplane can be attributed to different class. The dimension of the hyperplane depends on the number of input features in the dataset.
In general a hyperplane is **n-1** dimensional subspace in an **n-** dimensional space.

- SVC is a single point in a one-dimensional space
- SVC is a one-dimensional line in a two-dimensional space.
- SVC is a two dimensional plane in a three-dimensional space.
- If there is more than three parameters or dimensions, SVC is hyperplane.

Equation of Hyperplane



$$y = m * x + x \tag{1}$$

Hence, The hyperplane equation dividing the points (for classifying) can be written as

$$H : w^T(x) + b = 0 \tag{2}$$

This is 2D space so the hyperplane is straight line. In n dimensional space hyperplane would always be n-1 operators.

2.3 Linear vs Non-linear separability

In case of linear seperable, SVM is trying to find a hyperplane that maximizes the margin(the distance between the hyperplane and the observations closest to the hyperplane) with a condition that it classified both the classes. But in general, dataset are generally not linearly seperable. Below figure shows two-dimensional data for linear and non-linear seperable data points

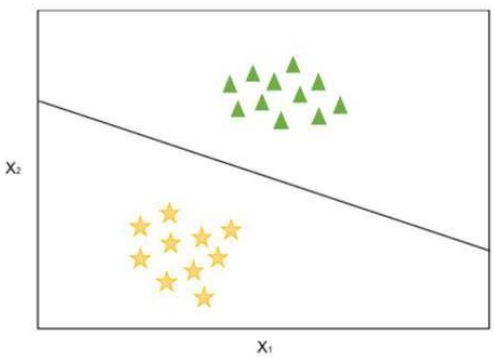


Fig 1a. Linear Separable

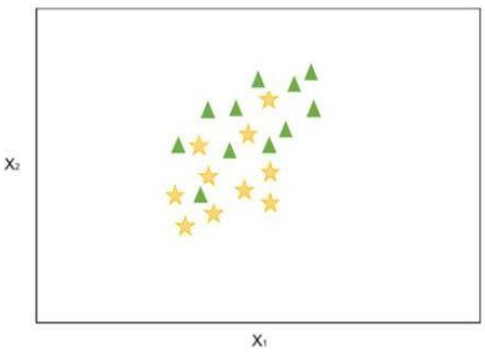


Fig 1b. Non-linear Separable

2.4 Classifier

Once we have the hyperplane, we can then use the hyperplane to make predictions. The hypothesis function h is:

$$h(x_i) = \begin{cases} +1 & \text{if } \omega \cdot x + b \geq 1 \\ -1 & \text{if } \omega \cdot x + b < 1 \end{cases} \tag{3}$$

3 Mathematics Behind Support Vector Machine Algorithm

3.1 Metrics to compare hyperplanes

Considering the equation of hyperplane $\omega \cdot x + b = 0$. If a point (x, y) is not on the hyperplane then $\omega \cdot x + b = 0$ could be positive or negative.

Given a dataset $D = \{(x_i, y_i) | x_i \in R^n, y_i \in \{-1, 1\}\}^m_{i=1}$ we can compute $\beta = |\omega \cdot x + b|$ for each point in the set and calculate

$$B = \min_{i=1...m} \beta \tag{4}$$

If we have S hyperplanes, each of them will have a B_i value, we select the hyperplane with the largest B_i value

$$H = \max_{i=1...S} \{h_i | B_i\} \tag{5}$$

This metric however needs adjustments, since we're taking absolute value of $\omega \cdot x + b$ we may get the same values for a correct and incorrect hyperplane. The problem of finding the values of ω and b is called optimization problem.

3.2 Derivation of SVM optimization problem

To find the values of ω and b of the optimal hyperplane, we need to solve the following optimization problem, with the constraint that the geometric margin of each example should be greater than or equal to M:

$$\begin{aligned} &\max_{\omega,b} M \\ &\text{subject to } \gamma_i \geq M, i = 1...m \end{aligned} \tag{6}$$

We also know that, $M=\frac{F}{\|\omega\|}$ and if we rescale ω and b, we are still maximizing M and the optimization result will not change. Let's rescale ω and b and make F=1, the above problem can be rewritten as:

$$\max_{\omega,b} \frac{1}{||\omega||} \quad \text{subject to } f_i \geq 1, i = 1...m \tag{7}$$

This maximization problem is equivalent to the following minimization problem:

$$\min_{\omega,b} ||\omega|| \quad \text{subject to } f_i \geq 1, i = 1...m \tag{8}$$

This minimization problem is equivalent to the following minimization problem:

$$\min_{\omega,b} \frac{1}{2}||\omega||^2 \quad \text{subject to } [y_i(\omega \cdot x + b) - 1] \geq 0, i = 1...m \tag{9}$$

The above statement is the SVM optimization problem. It is called a convex quadratic optimization problem. And we'll talk about how to solve this problem in the next section.

3.3 Solving SVM optimization problem – Hard Margin SVM

SVM Lagrange Problem

Lagrange stated that if we want to find the minimum of f under the equality constraint g, we just need to solve for:

$$\nabla f(x) - \alpha \nabla g(x) = 0 \tag{10}$$

where α is called the lagrange multiplier

In terms of SVM optimization problem $f(\omega) = \frac{1}{2}\|\omega\|^2, g(\omega, b) = y_i(\omega \cdot x + b) - 1, i = 1...m.$

The Lagrangian function is then $L(\omega, b, \alpha) = \frac{1}{2}\|\omega\|^2 - \Sigma_{i=1}^{m}\alpha_i[y_i(\omega \cdot x = b) - 1]$

Equivalently, we need to solve the following Lagrangian primal problem:

$$\min_{(\omega,b)} \max L(\omega, b, \alpha) \text{subject to } \alpha_i \geq 0, i = 1...m \tag{11}$$

Wolfe dual problem

The Lagrangian function is

$$L(\omega, b, \alpha) = \frac{1}{2}\omega \cdot \omega - \Sigma_{i=1}^{m}\alpha_i[y_i(\omega \cdot x + b) - 1] \tag{12}$$

For the dual problem:

$$\nabla_\omega L(\omega, b, \alpha) = \omega - \Sigma_{i=1}^{m}\alpha_i y_i x_i = 0 \nabla_b L(\omega, b, \alpha) = -\Sigma_{i=1}^{m}\alpha_i y_i = 0 \tag{13}$$

From the above two equations, we get $\omega = \Sigma_{i=1}^{m}\alpha_i y_i x_i$ and $\Sigma_{i=1}^{m}\alpha_i y_i = 0$. We substitute them into the Lagrangian function L The dual problem is thus stated as:

$$\begin{aligned} &\max_{\alpha} \Sigma_{i=1}^{m}\alpha_i - \frac{1}{2}\Sigma_{i=1}^{m}\Sigma_{j=1}^{m}\alpha_i \alpha_j y_i y_j x_i x_j \\ &\text{subject to } \alpha_i \geq 0, i = 1...m, \Sigma_{i=1}^{m} a_i y_i = 0 \end{aligned} \tag{14}$$

Note because the constraints are inequalities, we actually extend the Lagrange multipliers method to the KKT (Karush-Kuhn-Tucker) conditions. The complementary slackness condition of KKT conditions states that:

$$\alpha_i[y_i(\omega \cdot x^* + b) - 1] = 0 \tag{15}$$

x^* are the point/points where we reach the optimal

Compute ω and b

After we solve the Wolfe dual problem, we obtain a vector of α containing the Lagrangian multiplier value for every example. We can then proceed to compute ω and b, which determines the optimal hyperplane.

According to the equation above:

$$\omega - \Sigma_{i=1}^{m}\alpha_i y_i x_i = 0 \tag{16}$$

We get:

$$\omega = \Sigma_{i=1}^{m} \alpha_i y_i x_i \tag{17}$$

To compute the value of b:

$$y_i(\omega \cdot x^\star + b) - 1 = 0 \tag{18}$$

We multiply both sides by y_i and we know that $y_i^2 = 1$. We get:

$$b = y_i - \omega \cdot x^\star \tag{19}$$

Thus, b can be computed as:

$$b = \frac{1}{S} \Sigma_{i=1}^{S} (y_i - \omega \cdot x) \tag{20}$$

S is the number of support vectors.

3.4 Solving SVM optimization problem – Soft Margin SVM

Basically, the trick Soft Margin SVM adds slack variables ζ_i to the constraints of the optimization problem. The constraints now become:

$$y_i(\omega \cdot x_i + b) \geq 1 - \zeta_i, i = 1...m \tag{21}$$

we could use L1 regularization to penalize large values of ζ. The regularized optimization problem becomes:

$$\begin{aligned} &\min_{\omega,\zeta,b} \frac{1}{2} ||\omega||^2 + \Sigma_{i=1}^{m} \zeta_i \\ &\text{subject to } y_i(\omega \cdot x_i + b) \geq 1 - \zeta_i, i = 1...m \end{aligned} \tag{22}$$

Again, if we use Lagrange multipliers method like above, and we do all the hard math, the optimization problem could be transformed to a dual problem:

$$\begin{aligned} &\max_{\alpha} \Sigma_{i=1}^{m} \alpha_i - \frac{1}{2} \Sigma_{i=1}^{m} \Sigma_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ &\text{subject to } 0 \leq \alpha_i \leq C, i = 1...m, \Sigma_{i=1}^{m} a_i y_i = 0 \end{aligned} \tag{23}$$

Regularization parameter C

So, what does the regularization parameter C do? As we said, it determines how important ζ should be. A smaller C emphasizes the importance of ζ and a larger C diminishes the importance of ζ.

Another way of thinking of C is it gives you control of how the SVM will handle errors. If we set C to positive infinite, we will get the same result as the Hard Margin SVM. On the contrary, if we set C to 0, there will be no constraint anymore, and we will end up with a hyperplane not classifying anything. The rules of thumb are: small values of C will result in a wider margin, at the cost of some misclassifications; large values of C will give you the Hard Margin classifier and tolerates zero constraint violation. We need to find a value of C which will not make the solution be impacted by the noisy data.

3.5 Kernel or Feature Map

Now, the Soft Margin SVM can handle the non-linearly separable data caused by noisy data. What if the non-linear separability is not caused by the noise? What if the data are characteristically non-linearly separable? Can we still separate the data using SVM? The answer is of course: Yes. And we'll talk about a technique called kernel trick to deal with this. So, the kernel trick is: if you define a kernel function as $K(x_i, x_j) = x_i \cdot x_j$, we rewrite the Wolfe dual problem:

$$\begin{aligned} &\max_{\alpha} \Sigma_{i=1}^{m} \alpha_i - \frac{1}{2} \Sigma_{i=1}^{m} \Sigma_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i \cdot x_j) \\ &\text{subject to } \alpha_i \geq 0, i = 1...m, \Sigma_{i=1}^{m} a_i y_i = 0 \end{aligned} \tag{24}$$

The polynomial kernel is defined as:

$$K(x_i, x_j) = (x_i \cdot x_j + c)^d \tag{25}$$

This kernel contains two parameters: a constant c and a degree of freedom d. A d value with 1 is just the linear kernel. A larger value of d will make the decision boundary more complex and might result in overfitting.

The RBF kernel is defined as:

$$K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2) \tag{26}$$

The RBF (Radial Basis Function) kernel is also called the Gaussian kernel. It will result in a more complex decision boundary. The RBF kernel contains a parameter . A small value of will make the model behave like a linear SVM. A large value of will make the model heavily impacted by the support vectors examples.

In practice, it is recommended to try RBF kernel first cause it normally performs well.

4 Application of SVM on Placement Data

Goal: To predict if a student will get placed or not

4.1 Dataset

We used dataset of placements of around **1470** students containing their Roll Number, Name, Primary Program, Primary Department, Secondary Department, Gender, CPI, Email, Class X Board, Class X Year, Class X Marks, Class XII Board, Class XII Year, Class XII Marks, Entrance Exam, Entrance Rank, Category, Friend's Name, Friend's Number, Application Count, DateOfBirth, Specilisation, Local Address, External Email, Disability, Permanent Address, Result. Out of which we removed irrelevant columns and took Primary Program, Primary Department,Secondary Department,Gender, CPI , Entrance Exam, Entrance Rank, Category and Result columns for our use.

4.1.1 Preparing Dataset

- Removed the rows containing invalid values such as negative CPI, void Entrance rank and NaN values.
- Filled missing values of Secondary department as "not opted"
- Changed datatype of categorical data(Primary Program, Primary Department, Secondary Department, Gender, Entrance Exam, Category) into *category datatype*.

- Applied One hot encoding on categorical data and normalized numerical data for faster convergence of gradient descent.

4.2 Model Making

We defined SVM class containing various defined function

- To initialise weights and bias

```
def _init_weights_bias(self, X):
        n_features = X.shape[1]
        self.w = np.random.rand(n_features)
        self.b = 0}
        }
```

- get-gradients function is used to get gradient to update weight and biases and the Maths behind this

When constraint >= 1:

$$\frac{\partial J_i}{\partial \omega_k} = \lambda \omega_k \tag{27}$$

$$\frac{\partial J_i}{\partial b} = 0 \tag{28}$$

Otherwise:

$$\frac{\partial J_i}{\partial \omega_k} = \lambda \omega_k - y_i \cdot x_i \tag{29}$$

$$\frac{\partial J_i}{\partial b} = -y_i \tag{30}$$

```
def _get_gradients(self, constrain, x, idx):
        if constrain:
            dw = self.lambda_param * self.w
            db = 0
            return dw, db

        dw = self.lambda_param * self.w - np.dot(self.cls_map[idx], x)
        db = - self.cls_map[idx]
        return dw, db
        }
```

- Where the constraint function is the implementation of below equations

$$\text{constraint} = y_i(\omega \cdot x_i + b) \tag{31}$$

```
        def _satisfy_constraint(self, x, idx):
        linear_model = np.dot(x, self.w) + self.b
        return self.cls_map[idx] * linear_model >= 1
```

- Fit function is used to put all these things together to train model.

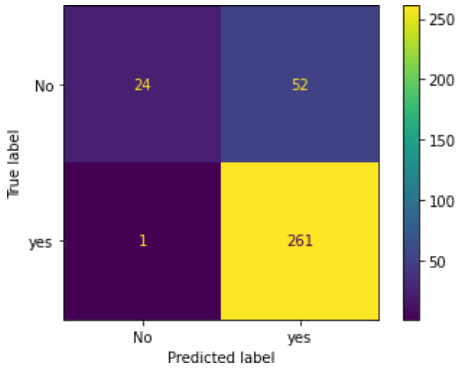
```
def fit(self, X, y):
        self._init_weights_bias(X)
        self.cls_map = self._get_cls_map(y)

        for _ in range(self.n_iters):
            for idx, x in enumerate(X):
                constrain = self._satisfy_constraint(x, idx)
                dw, db = self._get_gradients(constrain, x, idx)
                self._update_weights_bias(dw, db)
```

- predict function is used to predict the target

```
def predict(self, X):
        estimate = np.dot(X, self.w) + self.b
        prediction = np.sign(estimate)
        return np.where(prediction == -1, 0, 1)
```

4.3 Model Training and Testing

- After training model we saved predictions from test dataset and ploted confusion matrix and got the accuracy of 80.7%.
- Since hyperparameter tuning functions such as GridSearchCV can only be used for pre-built scikitlearn models so we have to write code for function that can do tuning for the SVM model we made from scratch.



Github repo : Link to full code

5 Conclusion

In this project, we built a Support Vector Mechanism (SVM) algorithm from scratch and then implemented it in a step-by-step fashion. We also learned the underlying mathematical concepts of SVM and how to solve the optimization problem using different approaches.

As a sample data set, on which our model is trained and tested, we have used placement stats of a certain collage and an accuracy of 80.7% is obtained. Furthermore, the accuracy of can be increased by using Hyper-parameter tuning.

Implementing the Support Vector Machine from scratch enables us to build a deeper understanding of the fundamental and underlying concepts. Such knowledge will prove useful when exploring some of the more complicated extensions that build on top of this algorithm.