Islamic University of Technology
Department of Computer Science and Engineering

Report on Support Vector Machines

Course Code: CSE 4621

Course Name: Machine Learning

Section: 1

Date of Submission: 18 April, 2022

Submitted To

Dr. Md. Hasanul Kabir

Professor, Department of CSE, IUT

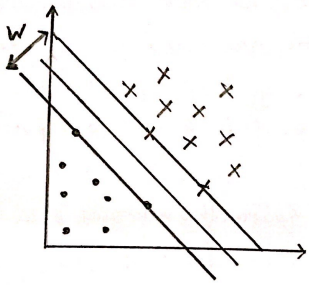
Submitted By

Md Farhan Ishmam **180041120**

Introduction:

SVMs or Support Vector Machines are supervised machine learning models widely used for data classification and regression. SVMs can generally be called as wide-margin, non-probalistic binary classifiers; although the classification can be for non-binary cases also, and instead of classification regression-based problems can also be handled. SVMs have produced great results in classification problems with limited data, often beating more sophisticated models such as deep neural networks. The key difference between SVMs and standard linear models is that SVMs try to maximize the width of the decision boundary and hence, generating a more precise decision boundary. The test examples are labeled by predicting to which side of the margin they belong to.

Theory:



We have n datapoints plotted in the graph where each data point belongs to positive or negative class. Mathematically, we represent as

$$y_i \in \{-1, 1\}, \; x_i \in R$$

For our first case, we assume the data to be linearly seperable. This means the whole dataset can be (linearly) divided using a straight line. On one side of the line, there will be positive classes and on the other side, there will be negative classes. If we look at this ~~pro~~ problem from the point of view of a linear machine or model, there can be many such linear

decision boundaries that will fit our optimization criterion. The support vector machines add another optimization criterion, it wants to maximize the width of the decision boundary. Doing so would give us a better decision boundary. By better, we mean the decision boundary can more accurately classify the data.

If we extend our 2D graph with an x-vector of single feature to an x-matrix with d-dimensions, then we need a $(d-1)$ dimension hyperplane to seperate the points in higher dimension. We can mathematically represent this hyperplane as

$$w \cdot x + b = 0$$

$-w$ represents the normal to the hyperplane

Then the perpendicular distance from hyperplane to origin is $b/\|w\|$

As seen in the graph, some datapoints are directly connected to the margin hyperplane. These are called support vectors, and they are closest to the hyperplane. The goal of support vector machine is to generate the hyperplane farthest to these support vectors, i.e maximize the width of the margin.

According to figure, the margine lines can be represented as:

$$w \cdot x^+ + b = 0 + 1, \text{ for positive support vectors} \quad \text{--- (i)}$$
$$w \cdot x^- + b = -1, \text{ for negative support vectors} \quad \text{--- (ii)}$$

Subtracting, we get, $w(x^+ + x^-) = 2 \quad [(i)-(ii)]$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|}$$

$$\therefore M = \frac{2}{|w|} \quad \text{------ (iii)}$$

In order to classify the training data,

$$wx_i + b \geq 1 \quad \text{for } y_i = +1$$
$$wx_i + b \leq -1 \quad \text{for } y_i = -1$$

Combining the equations, we get,

$$y_i(wx_i+b) \geq 1 \quad \forall_i$$

Our objective is to maximize the margin width, $M = \frac{2}{\|w\|}$ and it is to be maximized subject to the constraints

$$y_i(wx_i+b) \geq 1 \quad \forall_i \quad \text{---------(iv)}$$

The maximization criterion is equivalent to minimizing $\frac{1}{2}\|w\|^2$

We, now, have a quadratic optimization problem i.e a quadratic function that needs to be minimized subject to a linear constraint. Optimization problems with linear constraints can be solved using Lagrange multipliers α, where $\alpha_i \geq 0, \forall_i$

Now, $y_i(wx_i+b) \geq 1 \quad [(iv)]$

$$\Rightarrow y_i(wx_i+b) - 1 \geq 0$$

Hence, $L_p \equiv \frac{1}{2}\|w\|^2 - \alpha\{y_i(x_i \cdot w + b) - 1\}$

$$\equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i [y_i(x_i \cdot w + b) - 1]$$

$$\equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} \alpha_i y_i (x_i w + b) + \sum_{i=1}^{L} \alpha_i \quad \text{---------(v)}$$

We wish to find w and b which minimizes and, α which maximizes.

Differentiating L_p with respect to w and b, we get,

$$\frac{\partial L_p}{\partial w} = 0 \qquad \text{and,} \quad \frac{\partial L_p}{\partial b} = 0$$

$$\Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i \text{ ---(vi)} \qquad \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0$$

Substituting the values in (v), we get,

$$L_p \equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2}\sum \alpha_i \alpha_j y_i y_j x_i \cdot x_j \text{ with subject to}$$

$$\sum_{i=1}^{L} \alpha_i y_i = 0 \text{ and}$$

$$\alpha_i \geq 0 \text{ for } \forall_i$$

This is a convex quadratic optimization problem and we can run a quadratic polynomial solver which will return α. Using α, we can calculate the w. Then the classifying function will have the form

$$f(x) = \sum \alpha_i y_i x_i^T x + b$$

We notice that it relies on the inner product of the test points and support vectors x_i.

Non-linear Kernels:

In most cases, it is not possible to linearly seperate the data points. However, the theory of support vector machines can be extended to non-linear decision boundaries, by mapping the input feature set to a higher dimension feature set in which the input feature set appears to be linearly seperable.

If $x \in \mathbb{R}^n$ is an input point, we define $\Phi(x)$ to be the transformation function that maps x to some higher dimensional feature space. Such a function is called kernel function which is mathematically defined as

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

i.e. it corresponds to an inner product in some expanded feature space.

Kernel Trick:

Instead of computing the inner product of all pairs of data in feature space, we can implicitly declare the function in the original feature space that will produce changes similar to the changes in higher dimensional space. Such an operation where transformations to a high dimensional space doesn't explicitly have to the performed can result in faster computational results. This operation of implicit feature mapping is termed as Kernel Trick. Modern SVMs use Kernel trick to map non-linearity instead of explicit feature mapping to a higher dimension.

Examples of some Kernel functions are-

Linear : $K(x_i, x_j) = x_i^T x_j$

Polynomial : $K(x_i, x_j) = (1 + x_i^T x_j)^p$ where p is power of the polynomial

Gaussian : $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$ [Radial Basis Function]

Sigmoid : $K(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)$

Fisher : $K(x_i, x_j) = U_{x_i}^T \mathcal{I}^{-1} U_{x_j}$ where $U_x = \nabla_\theta \log P(x|\theta)$

$\mathcal{I}$ is Fisher Information Matrix

The optimization techniques for finding α remains the same after using any kernel.

$$f(x) = \sum \alpha_i y_i K(x_i, x_j) + b$$

Strengths of SVM:

i) SVM can operate with small dataset and provide decent in accuracy, outperforming sophisticated models like neural networks which tend to require larger datasets.

ii) There's flexibility in choosing the kernel or similarity function.

iii) When dealing with large datasets, there's sparseness of solution as only support vectors are used to seperate the data by forming a hyperplane.

iv) SVM can handle large feature spaces as complexity doesn't depend on the dimensionality of the feature space.

v) Overfitting can be handled by using soft margin.

vi) Guaranteed to converge to a single global solution.

Weakness of SVM:

i) SVMs are not suitable for large datasets where Neural Networks perform better.

ii) Underperforms in noisy dataset. Also causes issue in choosing the optimization criterion - hard vs soft margin

iii) Underperforms when features exceed the number of training examples.

iv) Issues in choosing the kernel and kernel parameters as they are not automatically tuned.

References:

i) Support Vector Machine and its Applications – Mingyue Tan

ii) Support Vector Machine – Wikipedia

iii) Kernel Trick – Wikipedia

iv) Properties of Support Vector Machine – M. Pontil and A. Verri

v) Support vector Machines Explained – Tristan Fletcher

vi) Support Vector Networks – C. Cortes and V. Vapnik

vii) Support Vector Machine – Simply Explained – Lilly Chen, towardsdatascience

viii) Support Vector Machines – A simple Explanation – N. Bambrick, KDnuggets