S21CSE471: Statistical Methods in AI **Spring 2021**

Support Vector Machine

Prepared by: Team 7 - Abhijeet Ashish Darshit

1 Introduction

Support Vector Machine is a supervised learning algorithm with an objective to find a hyper-plane in an N-dimensional space that distinctly analyzes the data for classification. Let's take an example of linear classification in 2D to understand it in a simpler way.

Suppose we have a 2 labeled classes red circle and blue circle on a graph which is separated by a hyper-plane (Fig 1). This hyper-plane is basically a line here which dividing a plane in two parts where each class is lay in either side.

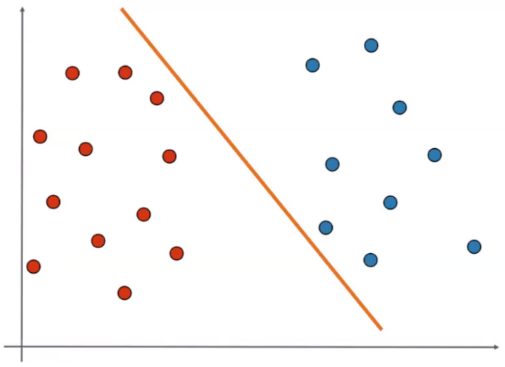

Figure 1: Data Separated by hyperplane

Any point that is in a left side of the plane is belongs to red circle class and on right it belongs to blue circle class. Make it in a mathematical way, our equation for Hyperplane is - $w^T x = 0$

So, for our 2D line it will be, $w_0 + w_1 * x_1 + w_2 * x_2 = 0$
Let's put the value $w_0 = -6, w_1 = 1, w_2 = 1,$
Our equation of line will be,$x_1 + x_2 - 6 = 0 \implies x + y–6 = 0$

If we put the coordinates of any point in above equation and if the value is +ve it falls under blue class and if the value is –ve falls into red class. And if the value is 0 it's on the line. So, the sign is the classification rule here for the binary class.
Now, the questions arise for the selection of hyperplane. How we select the best hyperplane?
To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find plane that has the maximum margin (Fig - 2), i.e. the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

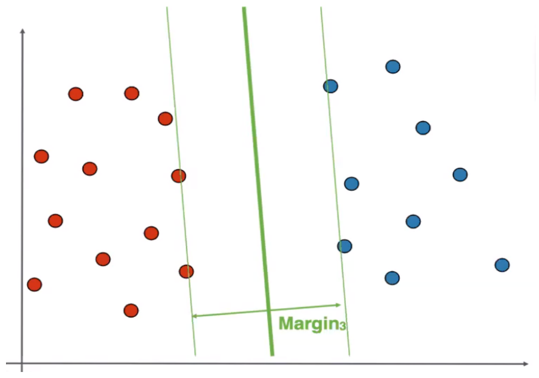

Figure 2: Hyperplane maximizes margin

2 Vapnik–Chervonenkis dimension

$$R(\alpha) <= R_{\text{tein}}\ (\alpha) + \frac{\sqrt{f \mid h}}{N}$$

Where h is the VC dimension, and f(h) given by:

$$f(h) = h + h\log(2N) - h\log(h) - c$$

To reduce test error, keep training error low (say 0), and minimize the VC-dimension, h.
Where h is the relative margin.

$$\text{Relative Margin}: \frac{\rho}{D}$$
$$\mathrm{V}C - D, h \leq \min\left(h, \left\lceil \frac{D^2}{\rho^2} \right\rceil\right) + 1.$$

So, in general we can say the larger the margin lowers the generalization error of the classifier.

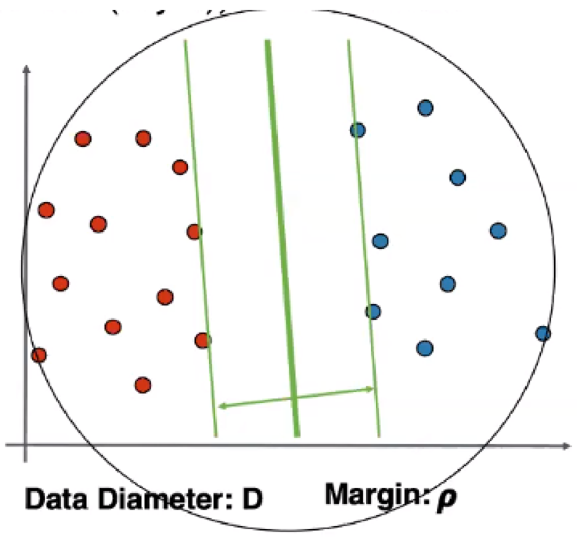

Figure 3:

3 Formulation of Margin

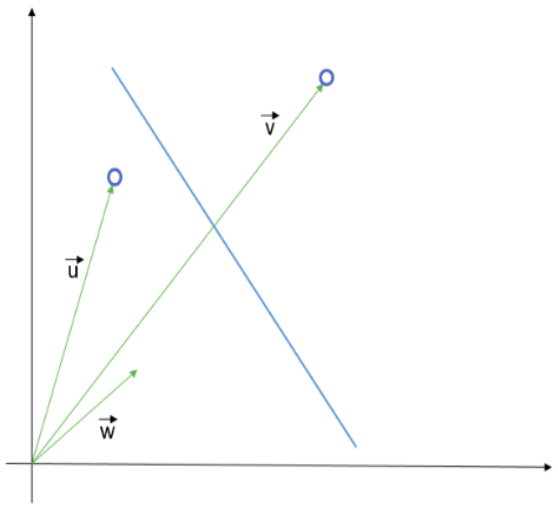

Figure 4:

Let us suppose we already have a decision boundary (blue line in Fig4) and two unknown points which we have to classify. We represent these points as vectors $\vec{u}$ and $\vec{v}$ in the 2 -d space. We also introduce a vector $\vec{w}$ which we assume is perpendicular to the decision boundary. Now, we project $\vec{u}$ and $\vec{v}$ in the direction of $\vec{w}$ and check whether the projected vector is on the left or right side of the decision boundary based on some threshold c.

Mathematically, we say that a data point $\vec{x}$ is on the right side of decision boundary (that is, in the Star class) if $\vec{w} \cdot \vec{x} \geq c$ else it is in the plus class. This means that the equation of the hyperplane that separates two classes, in terms of an arbitrary data point $\vec{x}$, is following:

$$\vec{w} \cdot \vec{x} + b = 0, \text{ where } b = -c \cdots (A)$$

It would be more convenient for us if our decision rule (i.e. $\vec{w} \cdot \vec{x} + b$) outputs quantity greater than or equal to +1 for all the data points belonging to star class and quantity less than or equal to -1 for all the data points belonging to plus class.

$\vec{x}$ should belong to class star if $\vec{w} \cdot \vec{x} + b \geq 1$ and $\vec{x}$ should belong to class plus if $\vec{w} \cdot \vec{x} + b \leq -1$ or equivalently, we can write

$$y_i \left(\vec{w} \cdot \vec{x}_i + b\right) \geq 1 \cdots (B)$$

for each point $\vec{x}_-i$, where we are considering y_i equal to -1 for plus class and equal to +1 for star class.

These two rules correspond to the dotted lines in the following diagram and the decision boundary is parallel and at equal distance from both. As we can see, the points closest to the decision boundary (on either side) get to dictate its position. Now, since the decision boundary has to be at a maximum distance from the data points, we have to maximize the distance d between the dotted lines. By the way, these dotted lines are called support vectors.

Now, let us denote the closest plus to the decision boundary as $\vec{x}_-$ and the closest star as $\vec{x}_+$. Then, d is the length of the vector $\vec{x}_+ - \vec{x}_-$ when projected along $\vec{w}$ direction (that is perpendicular to the decision boundary)

. Mathematically, d could be written as:

$$d = (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|w\|}$$

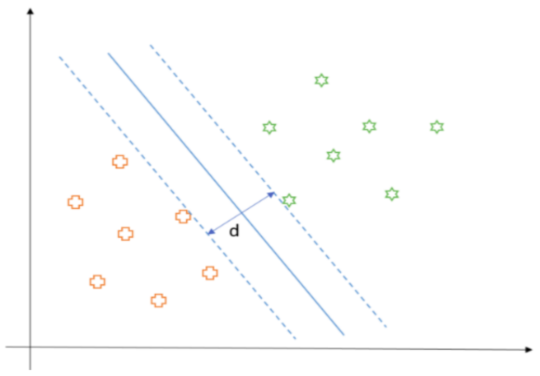

Figure 5: Hyperplane maximizes margin

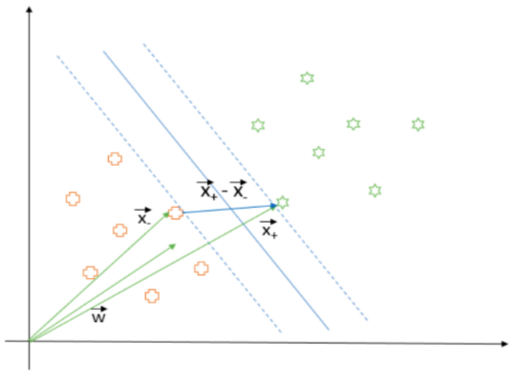

Figure 6: Calculating Margin

Since $\vec{x}_+$ and $\vec{x}_-$ are closest to the decision boundary and touch the dotted lines as mentioned earlier, they satisfy the following equations:

$$\vec{x}_+ \cdot \vec{w} + b = 1$$
$$\vec{x}_- \cdot \vec{w} + b = -1$$

Substituting $\vec{x}_+ \cdot \vec{w}$ and $\vec{x}_- \cdot \vec{w}$ in the equation of d, we get:

$$d = \frac{2}{\|\vec{w}\|} \quad \cdots (C)$$

Thus, if we have to maximize d, we can equivalently minimize $|\vec{w}|$ or minimize $\frac{1}{2}|\vec{w}|^2$. However, this optimization must be subjected to a constraint of correctly classifying all the data points. Hence, we'll make use of Lagrange Multiplier here to enforce the constraint from the equation (A). Our objective is to minimize the following objective function:

$$L = \frac{1}{2}\|\vec{w}\|^2 + \sum_i \lambda_i \left(y_i \left(\vec{w} \cdot \vec{x}_i + b\right) - 1\right) \cdots (D)$$

Differentiating L with respect to $\vec{w}$, we would obtain the optimal $\vec{w}$ as

$$\vec{w} = \sum_i \lambda_i y_i \vec{x}_i \quad \cdots (E)$$

The interesting thing to note here is that the decision vector $\vec{w}$ is a linear sum of the input vector (or data points) $\vec{x}_i$ s. Next step is to differentiate L with respect to b which would give us the following equality

$$\sum_i \lambda_i y_i = 0 \quad \cdots (F)$$

Now, we will substitute (E) into (D) and use (F) to rearrange the objective function into the following:

$$L = \sum_i \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \lambda_j y_i y_j \left(\vec{x}_i \cdot \vec{x}_j\right)$$

If you look closely, you would notice that optimization function now depends on the dot product of the input vectors . Also, this optimization function is convex so we would not get stuck in the local maxima. Now that we have everything, we could apply optimization routine like gradient descent to find the values of λ s. I would encourage you to implement it and observe the obtained values of λ s. Upon observing, you would notice that the value of λ would be zero for all the points except the ones which are closest to the decision boundary on the either side. This means that the points which are far away from the decision boundary don't get a say in deciding where the decision boundary should be. All the importance is assigned to the points closest to the boundary, which was our understanding all along.

References

[1] `https://rishabhmisra.github.io/Introduction-to-Support-Vector-Machines-Motivation-and-Basics/`

[2] `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligenc`