# Support Vector Machines

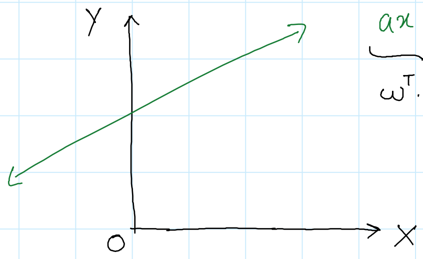
14 October 2021 18:46

1. In an n-dimensional space, the equation of a "hyperplane" is given by:

$$\Rightarrow \quad \omega^T x + b = 0$$

where $\vec{\omega}$ and $\vec{x}$ are n dimensional vectors and b is the bias term. $\in \mathbb{R}$

Eg: 2-D space (1-D line)




$$\underbrace{ax + by}_{\omega^T \cdot x} + c = 0$$

where:

$$\omega = \begin{bmatrix} a \\ b \end{bmatrix} \quad \& \quad x = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$(1\times 2) X (2\times 1) \Rightarrow (1\times 1) \in \mathbb{R}$$

So, we can always characterize or define a d-dimensional hyperplane with $\vec{\omega}$ & b.
(Think of ω as the slope of a 1-D line and b as the intercept.)

Thus, for a line –

$$ax + by + c = 0 \iff \omega_2 x_1 + \omega_1 x_0 + \omega_0 = 0$$

$$\Updownarrow$$

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} + \omega_0 = 0$$

for a plane –

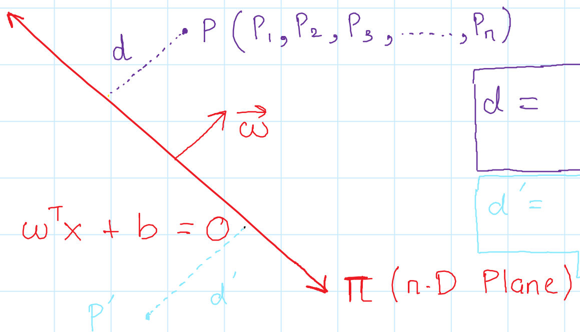
$$ax + by + cz + d = 0 \iff \omega_3 x_2 + \omega_2 x_1 + \omega_1 x_0 + \omega_0 = 0$$

$$\Updownarrow$$

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^T \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \omega_0 = 0$$

2. To 'cut' or distribute an n-dimensional space, we need an (n-1) dimensional 'hyperplane'.
$\Rightarrow$ Divide a line with a point.
$\Rightarrow$ Cut a cubical space with a plane sheet.

3. Distance of a point from a plane (n - D)




$$\boxed{d = \frac{\vec{\omega} \cdot \vec{P}}{\|\omega\|}} = \|\vec{P}\| \cos\theta$$

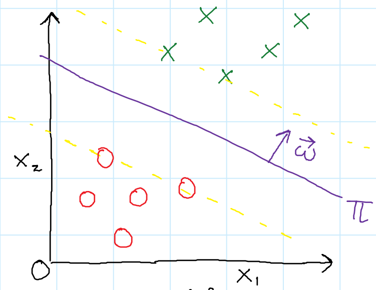
$$\boxed{d' = \frac{\vec{\omega} \cdot \vec{P'}}{\|\omega\|}} = \|\vec{P}'\| \cos\theta'$$

This is pretty intuitive, we take the projection of $\vec{P}$ on the normal $\vec{\omega}$ and divide by its norm to get d.

## ARMED WITH THIS KNOWLEDGE, WE CAN ATTACK SUPPORT VECTOR MACHINES !

**AIM :** To create a Maximum Margin Classifier for some linearly separable data. or else use Kernel Function(s) to create higher dimensional inner products.

**HYPOTHESIS :** Once your model is ready, it ideally should classify all + samples & - samples correctly. So, it should lie such that all points on one side are + samples and all points on the other side are - samples. with the widest gutter width and "cushioning" on both the sides to maximize generalization.




A sample "crosses" the hyperplane if

$\vec{\omega} \cdot \vec{x} \geqslant c$ (covered properly in my notes)

$\Rightarrow \vec{\omega} \cdot \vec{x} - c \geqslant 0$

$\Rightarrow \vec{\omega} \cdot \vec{x} + b \geqslant 0$

$\Rightarrow \vec{\omega} \cdot \vec{x}_+ + b \geqslant 0$ (for all + samples)

$\vec{\omega} \cdot \vec{x}_- + b \leq 0$ (for all − samples)

$$\Rightarrow \quad \vec{\omega} \cdot \vec{x}_+ + b \geq 0 \quad \text{for} \quad y = +1$$
$$\vec{\omega} \cdot \vec{x}_- + b \leq 0 \quad \text{for} \quad y = -1$$

$$\Rightarrow \quad y_i(\vec{\omega} \cdot \vec{x}_i + b) \geq 0 \quad \forall \ i \in [1, 2, \dots, m]$$
(samples)

Now let's come back to the training phase. We want to have -

$$\vec{\omega} \cdot \vec{x}_i + b \geq +1 \quad \text{for + samples}$$
$$\vec{\omega} \cdot \vec{x}_i + b \leq -1 \quad \text{for − samples}$$

(covered properly in my notes)

A new explaination for the above eq$^{ns}$:

Distance of a point from a hyperplane

$$= \frac{\vec{\omega} \cdot \vec{x}}{\|\vec{\omega}\|}$$

So for + samples,

$$\frac{\vec{\omega} \cdot \vec{x}_+}{\|\vec{\omega}\|} \geq \nu \quad \text{(some margin threshold)}$$

for − samples,

$$\frac{\vec{\omega} \cdot \vec{x}_-}{\|\vec{\omega}\|} \leq -\nu \quad \text{(some margin threshold)}$$

$\Rightarrow$ for − samples,
For + samples,

$$\vec{\omega} \cdot \vec{x}_- \leq \{-\nu\|\omega\| = -1\}$$
$$\vec{\omega} \cdot \vec{x}_+ \geq \{\nu\|\omega\| = 1\}$$

Because our normal vector $\vec{\omega}$ is independent of scaling, changing its magnitude won't do anything because it's only meant for directing our hyperplane. (Discussed properly in my notes)

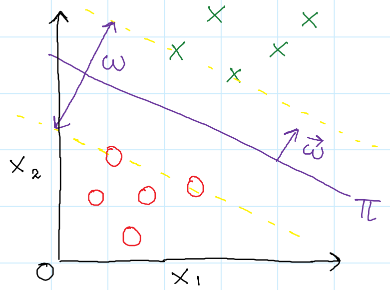
# OPTIMIZATION FOR HARD MARGIN :

So till now we discussed our expectations from the algorithm in the form of constraints and to sum them up,

$$\omega \cdot X_+ + b \geq +1 \quad \} \ (1)$$

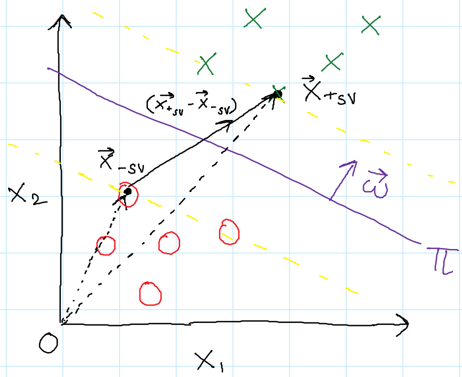
$$\left.\begin{aligned} \omega.X_+ + b &\geq +1 \\ \omega.X_- + b &\leq -1 \end{aligned}\right\} \quad (1)$$

Our motive was to get the Maximal Margin Classifier which has the widest gutter / street.




Maximize ω

Note that ω is completely different from $\vec{\omega}$.




Width of street = Projection of $(\vec{X}_{+sv} - \vec{X}_{-sv})$ on a unit vector in the direction of $\vec{\omega}$

$$= (\vec{X}_{+sv} - \vec{X}_{-sv}) \cdot \frac{\vec{\omega}}{\|\vec{\omega}\|} = \frac{\vec{\omega}.\vec{X}_{+sv} - \vec{\omega}.\vec{X}_{-sv}}{\|\omega\|}$$

$$= \frac{1 - b - (-1 - b)}{\|\omega\|} = \boxed{\frac{2}{\|\omega\|}} \quad (\text{From } (1))$$

So, we want to maximize $\frac{2}{\|\omega\|}$ or minimize $\|\omega\|$

MINIMIZE $\left\{\frac{1}{2}\|\omega\|^2\right\}$ (for mathematical ease)

(Quadratic problem with just a global minima)

subject to $y_i\{\omega.x_i + b\} = 1$ (We don't have inequality here because our model only depends on SVs)

We will use Lagrangian here to solve our optimization problem.

$$\mathcal{L}(\omega, b) = \frac{1}{2}\,\omega.\omega - \sum_{i=1}^{m} \alpha_i\left[y_i(\omega.x_i + b) - 1\right]$$

where $\alpha_i$ : $i^{th}$ Lagrangian Multiplier $(\alpha_i \geq 0)$

{More details about Lagrangian are in my NOTES ¿}

$$\frac{\partial \mathcal{L}}{\partial \omega} = \omega - \sum_{i=1}^{m} \alpha_i [y_i x_i] = 0$$

$$= 0 -$$

$$\Rightarrow \boxed{\omega_m = \sum_{i=1}^{m} \alpha_i y_i x_i} \quad - (2)$$

$$\Rightarrow \boxed{\sum_{i=1} \alpha_i y_i = 0} \quad - (3)$$

$$\sum^{m} \alpha_i y_i = 0$$

$\partial b$

Using (2) & (3) in our Primal Lagrangian Problem,

$$\mathcal{L}(\omega, b) = \frac{1}{2} \omega \cdot \omega - \sum_{i=1}^{m} \alpha_i y_i \omega . x_i - b \underbrace{\sum_{i=1}^{m} \alpha_i y_i}_{0} + \sum_{i=1}^{m} \alpha_i$$

$$\mathcal{L}(\omega, b) = \frac{1}{2} \omega . \omega - \sum_{i=1}^{m} \alpha_i y_i \omega . x_i + \sum_{i=1}^{m} \alpha_i$$

$$\mathcal{L}(\omega, b) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{m} \alpha_i$$

$$\boxed{\mathcal{L}_D(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)}$$

where $\alpha_i \geq 0$

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

Now, why did we do this ?

Well, our primal lagrangian had 3 variables to tune in order to find the minima of the objective (and the lagrangian itself). We converted that to maximizing our New Lagrangian over $\alpha_i$'s by substituting $\omega$ and $b$ as a function of $\alpha$. This was done by using the property that derivatives at min = 0.

$$\left( \frac{\partial \mathcal{L}}{\partial \omega}, \frac{\partial \mathcal{L}}{\partial b} \right)$$

Refer to 'Introduction to Machine Learning : Support Vector Machines' for understanding about Lagrangian.

Eg: $\underset{x,y}{\text{minimize}} \quad 2 - x^2 - 2y^2$

subject to : ① $x + y - 1 = 0$ (Equality constraint)

② (You can also have inequality constraints)

$$\mathcal{L}(x,y,\alpha) = (2 - x^2 - 2y^2) - \alpha(x + y - 1)$$

and now we have an unconstrained problem with respect to $x, y$ and $\alpha$ (Lagrangian multip.)

$\min \mathcal{L}(x,y,\alpha)$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial x} = 0, \quad \frac{\partial \mathcal{L}}{\partial y} = 0, \quad \frac{\partial \mathcal{L}}{\partial \alpha} = 0$$

$-2x - \alpha = 0$ — ①

$-4y - \alpha = 0$ — ②

$x + y - 1 = 0$ — ③

$\Rightarrow x + y = 1, \quad \alpha = -2x = -4y \Rightarrow \boxed{x = 2y}$

$$\boxed{y = 1/3, \quad x = 2/3, \quad \alpha = -4/3}$$

Similarly, you can have $m$ constraints.

Eg 2:

$\text{extr.}(f(x,y)) = 8x^2 - 2y$

$g(x,y) = x^2 + y^2 - 1 = 0$

$\text{extr.}(\mathcal{L}(x,y,\alpha)) = 8x^2 - 2y - \alpha(x^2 + y^2 - 1)$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial x} = 16x - 2\alpha x = 0$$

$$\frac{\partial \mathcal{L}}{\partial y} = -2 - 2y\alpha - 0$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} = -(x^2 + y^2 - 1) = 0$$

$x^2 + y^2 = 1$ — ①

$x(16 - 2\alpha) = 0$ — ②

$1 + \alpha y = 0$ — ③

i) $x = 0$ -

$y^2 = 1 \Rightarrow y = \pm 1$

$\alpha = \mp 1$

$x = 0, y = 1, \alpha = -1$

$x = 0, y = -1, \alpha = +1$

ii) $\alpha = 8$ :-

$y = -1/8 \Rightarrow x = \sqrt{1 - \frac{1}{64}} = \pm\frac{\sqrt{63}}{8}$

$x = \pm\frac{\sqrt{63}}{8}, y = \frac{-1}{8}, \alpha = 8$

Now check which of the 4 solns. give us our maxima/minima.

Once you solve the Dual Problem, you will get your support vectors (those having non-zero $\alpha_i$). After getting your support vectors, you can derive your $\omega^*$.

$$\omega^* = \sum_{i=1}^{m} \alpha_i y_i x_i$$

[ This raised a question in my mind, if we used a kernel function to get the $\alpha$'s, then we won't be able to use this formula because we won't have the $x_i$'s in the higher dimensions. ]

Once we get $\omega$, we can plug in that into:

$$\omega^* . x_{sv} + b^* = y_{sv} \quad \text{— (4)}$$

for some SV lying exactly on our margin.

$$\Rightarrow \boxed{b^* = y_{sv} - \omega . x_{sv}}$$