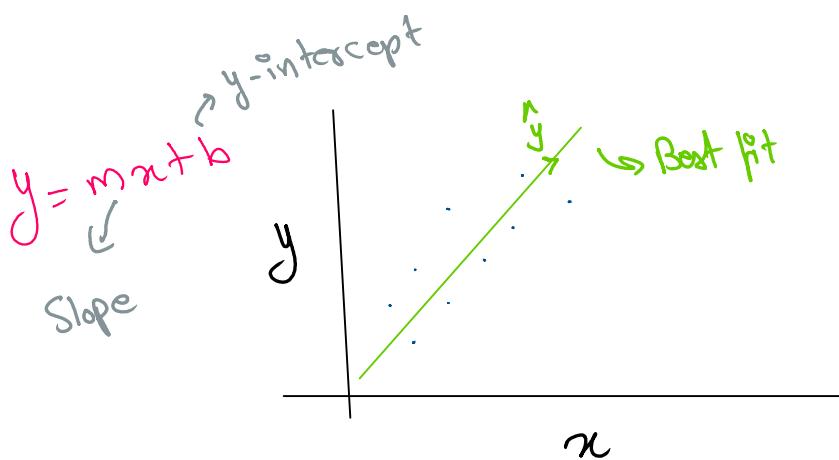
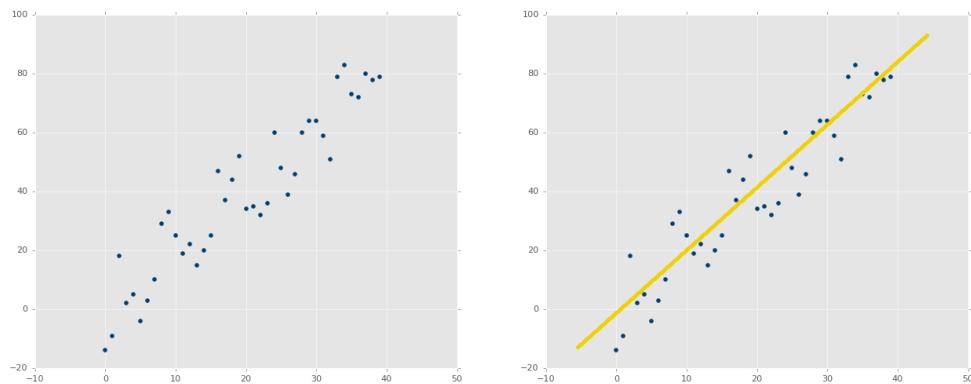


Simple Regression is used to find the best fit line of a dataset.



$$\text{Slope } (m) \therefore m = \frac{\bar{x} \cdot \bar{y} - \bar{x}\bar{y}}{(\bar{x})^2 - (\bar{x}^2)}$$

$$\text{y-intercept } (b) \therefore b = \bar{y} - m\bar{x}$$

$$\left\{ \because y = mx + c \right. \therefore \hat{y} = \frac{\bar{x}\bar{y} - \bar{x}\bar{y}}{(\bar{x})^2 - (\bar{x}^2)} \times x + (\bar{y}) - \left(\frac{\bar{x}\bar{y} - \bar{x}\bar{y}}{(\bar{x})^2 - (\bar{x}^2)} \bar{x} \right)$$

Equation of Regression Line (\hat{y})

→ R-Squared

The distance between Regression line Values (\hat{y}) & data's y values is the error, which we then sq.

Line's Squared error is either a mean or sum of this values.

$$R^2 = 1 - \frac{\text{Squared Error } \hat{y}}{\text{Squared Error } \bar{y}}$$

Coefficient of determination

→ regression line

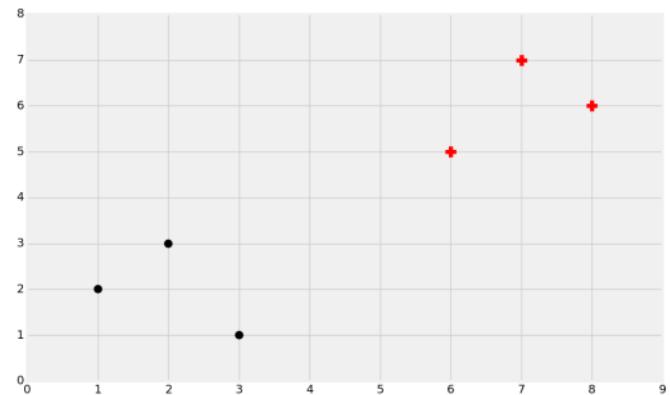
→ mean of y data points

Q) Why are we squaring errors? Why not just adding them up?

Ans) First, we want a way to normalize the error as a distance, so the error might be -5, but, when squared, that's a positive number. Another reason is to further punish for outliers. The "further" off something is, the more it affects the error. This is simply the standard people use.

You could use the power of 4, 6, 8, or whatever. You could also just use the absolute value of the errors. If you have a unique challenge, maybe where some extreme outliers do exist, but you don't care to map them, you could consider doing something like an absolute value. If you care a lot about outliers, you could use much higher exponents. We'll stick with squared, as that is the standard almost everyone uses.

- It is a type of classification algo.
- Type of Supervised Learning.
- Datasets are labeled
- Lazy learner, no learning phase
- Algo:-



Step-1: Select the number K of the neighbours

Step-2: Calculate the Euclidean distance of **K number of neighbors**

Step-3: Take the K nearest neighbours as per the calculated Euclidean distance.

Step-4: Among these k neighbours, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbour is maximum.

Step-6: Our model is ready.

$$\rightarrow \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

→ Working:-

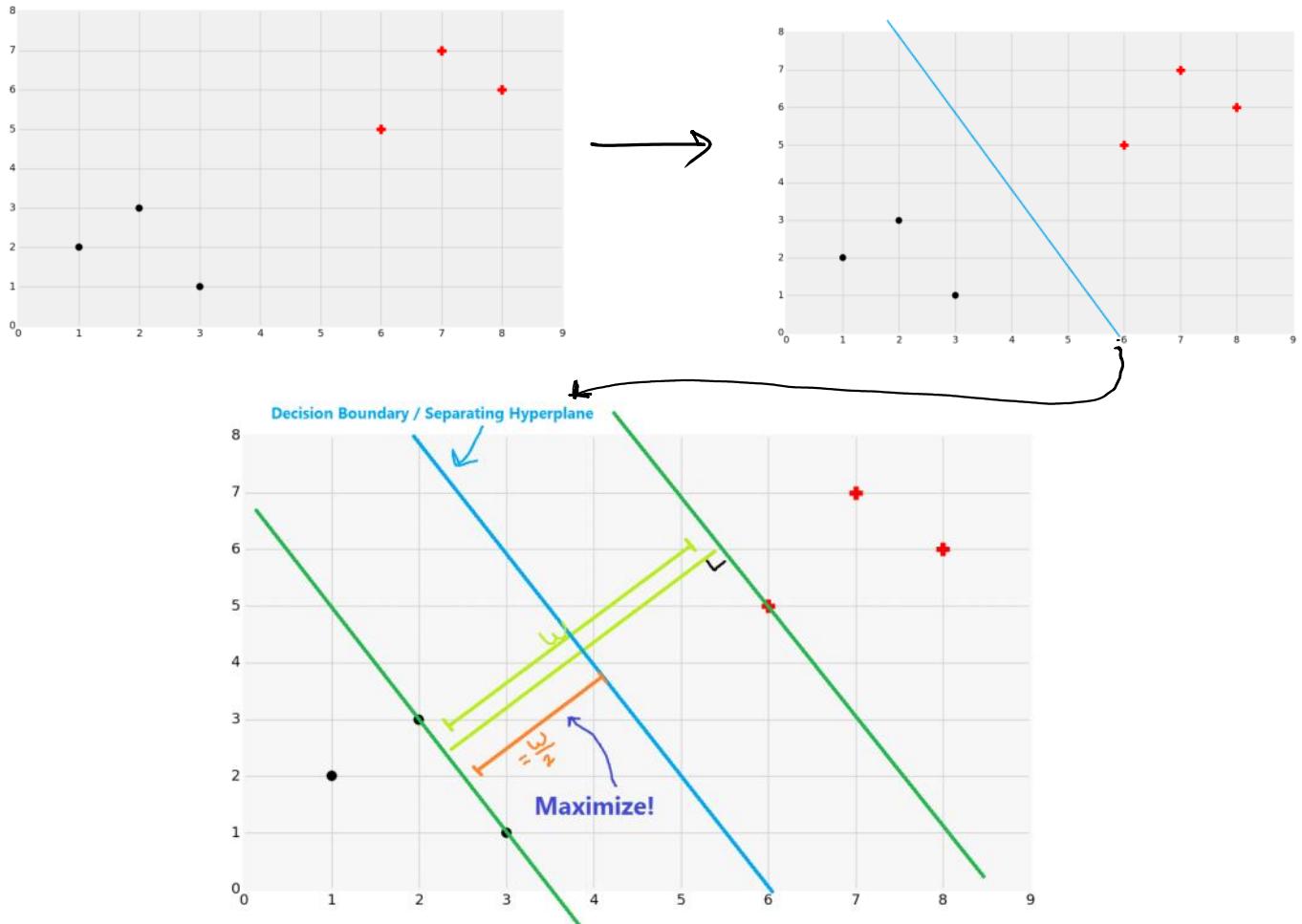
① Pass train_data & single data point from test_Set.

② Calculate Euclidean distance for data point to all the data present in train_data.

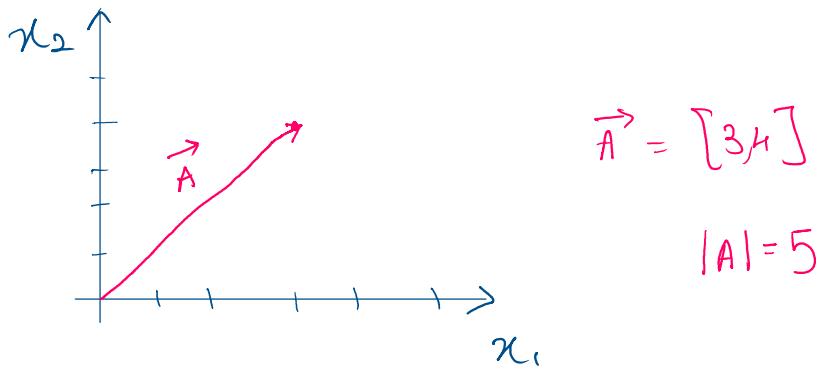
③ Save the distance and the group

for data point to train-data.

- (4) Sort and Select 'K' from the Saved point.
- (5) From the 'K' selected groups, vote for most common classifier.

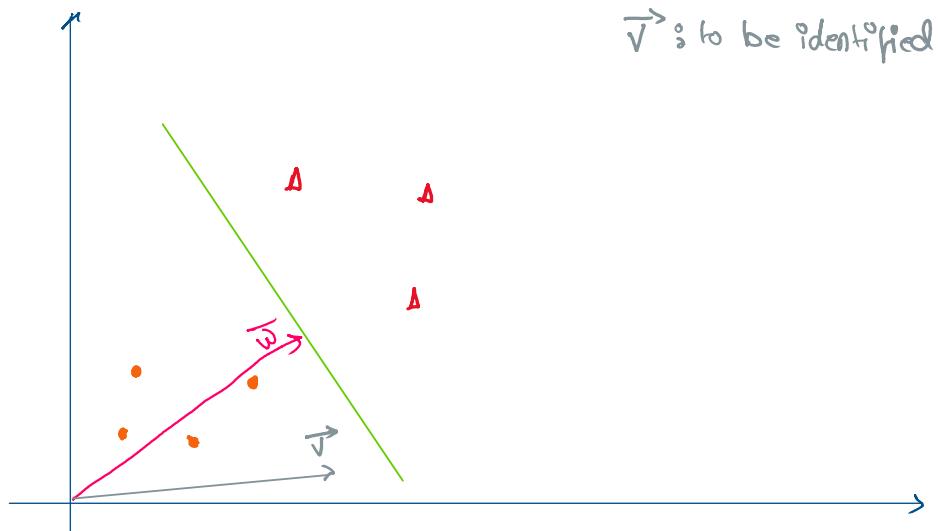


→ Vector Basics



$$\begin{aligned}\vec{A} &= [1, 3] \\ \vec{B} &= [4, 2]\end{aligned}\quad \vec{A} \cdot \vec{B} = [(1 \times 4) + (3 \times 2)] \\ = 4 + 6\end{math>$$

→ SVM Working :-



$$(\vec{v} \cdot \vec{w}) + b \quad \begin{cases} \geq 0 & \Delta \\ \leq 0 & \circ \end{cases}$$

if $\vec{v} = 0$ then \vec{v} lies on the decision boundary

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

$$\begin{aligned} y_i &\rightarrow \Delta = 1 \\ &\rightarrow \circ = -1 \\ (\text{class}) \end{aligned}$$

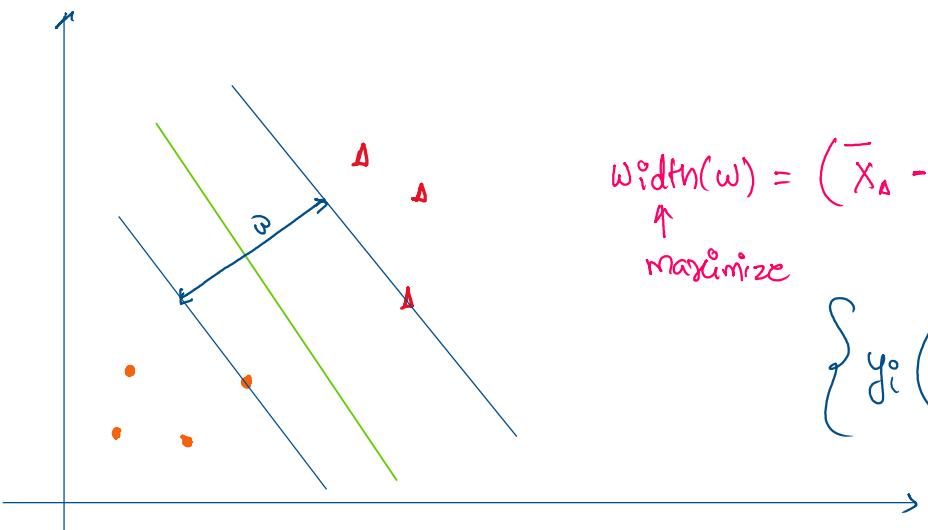
$$\Rightarrow \Delta \text{ class } \vec{x}_i \cdot \vec{w} + b = 1 \Rightarrow y_i (\vec{x}_i \cdot \vec{w} + b) = 1$$

$$\Rightarrow \Delta \text{ class } y_i = 1 \quad \bar{x}_i^T \bar{w} + b = 1 \Rightarrow y_i (\bar{x}_i^T \bar{w} + b - 1) = 1 \rightarrow 1$$

$$\bullet \text{ class } y_i = -1 \quad \bar{x}_i^T \bar{w} + b = -1 \Rightarrow y_i (\bar{x}_i^T \bar{w} + b - (-1)) = -1 \rightarrow -1$$

$$\Delta \text{ class } y_i = 1 \Rightarrow y_i (\bar{x}_i^T \bar{w} + b - 1) = 1 - 1 \rightarrow 0$$

$$\bullet \text{ class } y_i = -1 \Rightarrow y_i (\bar{x}_i^T \bar{w} + b - (-1)) = -1 - 1 \rightarrow 0$$



$$\text{width}(w) = (\bar{x}_u - \bar{x}_l) \cdot \frac{\bar{w}}{\|w\|}$$

↑
maximize

$$\left\{ y_i (\bar{x}_i^T \bar{w} + b - 1) = 0 \right\}$$

$$\Rightarrow ((1-b) - (1+b)) \cdot \frac{\bar{w}}{\|w\|}$$

* width = $\frac{(2) \cdot \bar{w}}{\|w\|}$

$$\text{width} = \frac{2}{\|w\|}$$

(maximize) minimize

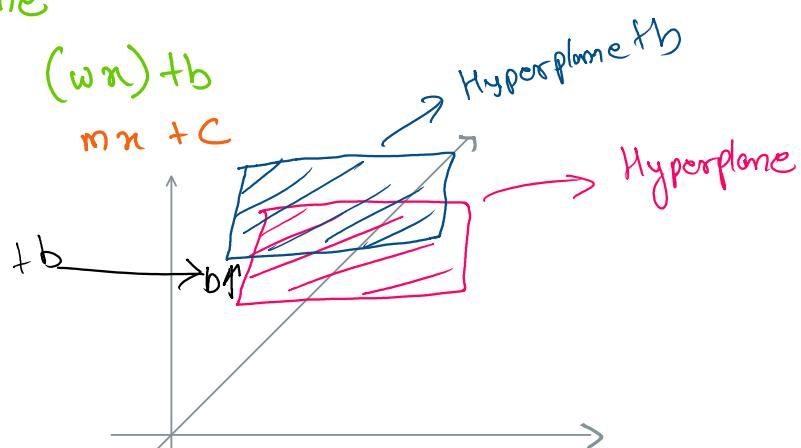
- * we want to minimize $\frac{1}{2}|\omega|^2$ then we also want to minimize $|\omega|$
- * lets say we want to minimize $\frac{1}{2}|\omega|^2$ as we already want to minimize $|\omega|$

$$\frac{1}{2}|\omega|^2 - \sum_i y_i (\bar{x}_i \cdot \omega + b) - 1$$

Lagrangian formulation

$$L(\omega, b) = \frac{1}{2}|\omega|^2 - \sum_i \alpha_i [y_i (\bar{x}_i \cdot \omega + b) - 1]$$

• Hyperplane



\Rightarrow on differentiating wrt ω & also differentiating wrt. b .

$$\frac{dL}{d\omega} = \bar{w} - \sum_i \alpha_i \bar{x}_i = 0 \Rightarrow \bar{w} = \sum_i \alpha_i y_i \bar{x}_i$$

$$\frac{\partial L}{\partial b} = -\sum_i y_i = 0 \Rightarrow \sum_i y_i = 0$$

Max

$$L = \sum_i \ell_i - \frac{1}{2} \sum_{i,j} \ell_i \ell_j y_i y_j \cdot (\bar{x}_i \cdot \bar{x}_j)$$

quadratic

→ Concept :-

$$(x \cdot w + b) \Leftarrow \text{Hyperplane Equation}$$

$= 1$ + class $= -1$ - class

$$\Rightarrow \underbrace{\text{Sign}}_{\substack{<0 \\ + class}}(x_i \cdot w + b) = 0 \quad \substack{>0 \\ - class}$$

Decision
Border

- * The optimization objective is to minimize $\|w\|$ & maximize b : $\{ \|w\| \cup b \}$

w_1, w_2, \dots, w_n
 Known Feature

$$y_i(x_i \cdot w + b)$$

Known
Unknown

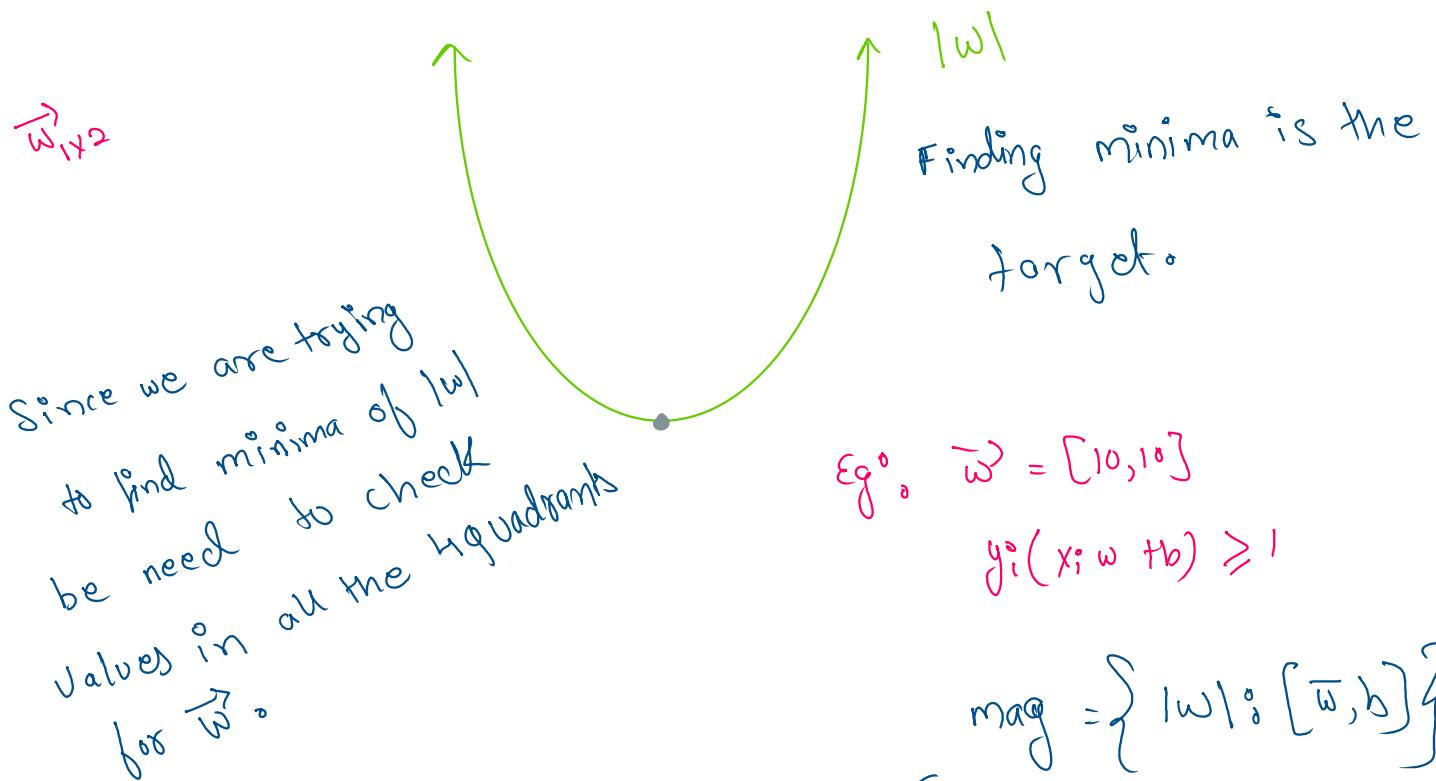
class

$$L = \sum_i \ell_i - \frac{1}{2} \sum_{i,j} \ell_i \ell_j y_i y_j \cdot (\bar{x}_i \cdot \bar{x}_j)$$

max quadratic

SVM is a optimization problem, $|w|$ is a Convex shape.

$\downarrow \{ \text{Convex} \}$



$$\text{Eg. } \vec{w} = [10, 10]$$

$$y_i(x_i \cdot w + b) \geq 1$$

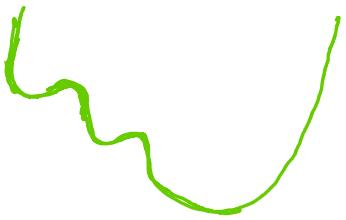
$$\text{mag} = \left\{ |w|; [\vec{w}, b] \right\}$$

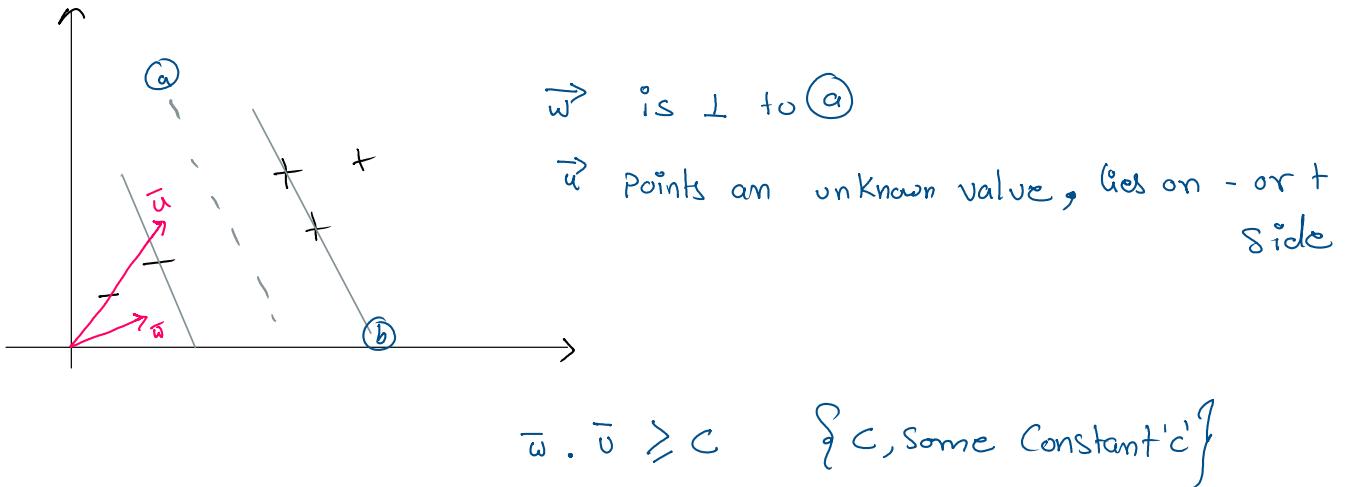
Finding lowest mag inside is target.

* local minima problem might occur

* local minima problem might occur

is target





\Rightarrow dot product has taken projection on \vec{w} , the bigger the projection is the further the projection lie on \vec{w} .

$$\boxed{\vec{w} \cdot \vec{u} + b \geq 0 \quad \text{Then +}} \\ \text{Decision Rule}$$

$$\vec{w} \cdot \vec{x}_+ + b \geq 1 \quad \left\{ \begin{array}{l} \text{for pt to lie out the the decision} \\ \text{boundary @} \end{array} \right.$$

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

\Rightarrow let y_i^o be such that $y_i^o = +1$ for + samples
 $y_i^o = -1$ for - samples

$$(+)\quad y_i^o (\vec{w} \cdot \vec{x}_i^o + b) \geq y_i^o(1) \Rightarrow y_i^o (\vec{w} \cdot \vec{x}_i^o + b) \geq 1$$

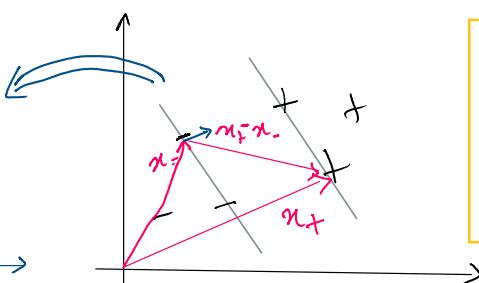
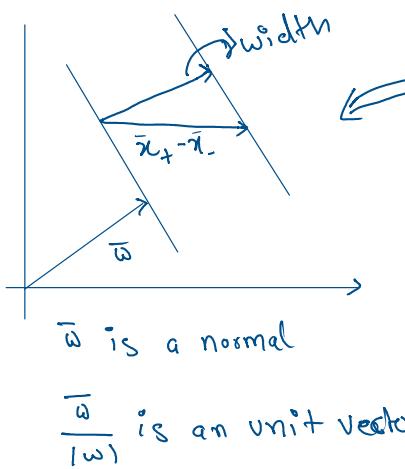
$$(-)\quad y_i^o (\vec{w} \cdot \vec{x}_i^o + b) \leq y_i^o(-1) \Rightarrow y_i^o (\vec{w} \cdot \vec{x}_i^o + b) \geq 1$$

$$(-)\quad (y_i^o)(-1)$$

$$\begin{array}{c} (-) \\ (-1) \\ (+) \end{array} \geq \begin{array}{c} (-1) \\ (+) \\ (+) \end{array}$$

$$y_i(\bar{x}_i \cdot \bar{\omega} + b) - 1 \geq 0$$

$$y_i(\bar{x}_i \cdot \bar{\omega} + b) = 0; \text{ for } x_i \text{ in gutter}$$



$$\text{width} = (\bar{x}_+ - \bar{x}_-) \cdot \frac{\bar{\omega}}{|\omega|}$$

$$\left\{ \text{using } y_i(\bar{x}_i \cdot \bar{\omega} + b) = 0 \right\}$$

$$\Rightarrow -(\bar{x}_+ - \bar{x}_-) \cdot \frac{\bar{\omega}}{|\omega|},$$

$$(1-b) - (1+b) \cdot \frac{1}{|\omega|}$$

$$\Rightarrow \text{width} = \frac{2}{|\omega|} - \textcircled{3}$$

To get the max dist b/w decision boundary we need

$$\text{to maximize } \frac{2}{|\omega|}$$

$$\hookrightarrow \text{maximize } \frac{1}{|\omega|} \rightarrow \text{minimize } |\omega|$$

for mathematical convenience.

$$\hookrightarrow \text{maximize } \frac{1}{\|\omega\|} \rightarrow \text{minimize } \|\omega\|$$

$$\hookrightarrow \text{minimize } \frac{1}{2} \|\omega\|^2$$

$$L = \frac{1}{2} \|\omega\|^2 - \sum \alpha_i [y_i (\bar{\omega} \cdot \bar{x}_i + b) - 1]$$

Lagrange multipliers

$$\frac{\partial L}{\partial \bar{\omega}} = \bar{\omega} - \sum \alpha_i y_i x_i = 0 \Rightarrow \boxed{\bar{\omega} = \sum \alpha_i y_i x_i}$$

$$\frac{\partial L}{\partial b} = -\sum \alpha_i y_i = 0 \Rightarrow \boxed{\sum \alpha_i y_i = 0}$$

$$L = \frac{1}{2} (\sum \alpha_i y_i \bar{x}_i) (\sum \alpha_j y_j \bar{x}_j) - \sum \alpha_i y_i x_i (\sum \alpha_j y_j x_j) - \underbrace{\sum \alpha_i y_i b}_{0} + \sum \alpha_i$$

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

*

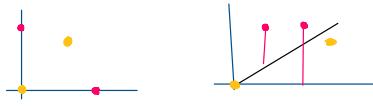
$$\sum \alpha_i y_i \bar{x}_i \cdot \bar{u} + b \geq 0 \quad \text{Then} +$$

*

Decision Rule only depends on the dot product
of Sample Vectors (\bar{x}_i) & unknown vector (\bar{u})

Note: this is a convex Space, so we can't get stuck to a local maxima.

For Non-linear Separable Data:-



we can separate Non linear data we change space.

$$\phi(\bar{x}_i) \cdot \phi(\bar{x}_j) \text{ To maximize}$$

$$\phi(\bar{x}_i) \cdot \phi(\bar{x})$$

$$K(x_i, x_j) = \phi(\bar{x}_i) \cdot \phi(\bar{x}_j)$$

Kernal function

① Linear Kernal - $(\bar{u} \cdot \bar{v} + 1)^n$

② Radial Basis - $e^{-\frac{|\bar{x}_i - \bar{x}_j|}{\sigma}}$

- Kernels are similarity function takes two inputs & returns a similarity using inner (dot) product.
- Kernels allows us to work in many dimensions w/o paying processing cost to do so.

$$K(x, x') = z \cdot z' \quad \left\{ \begin{array}{l} K \text{ or } \phi \\ z \rightarrow \text{function}(x) \\ z' \rightarrow \text{function}(x') \end{array} \right\}$$

for Example, Lets assume a polynomial Kernel to 2nd order polynomial.

$$X = [x_1, x_2] \quad Z = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$$

$$Z' = [1, x_1, x_2, x_1^2, x_2^2, x_1^2 x_2^2]$$

$$\phi(x, x') = z \cdot z'$$

$$= 1 + x_1 x'_1 + x_2 x'_2 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + x_1 x'_1 x_2 x'_2$$

$$\Rightarrow K(x, x') = (1 + x \cdot x')^P$$

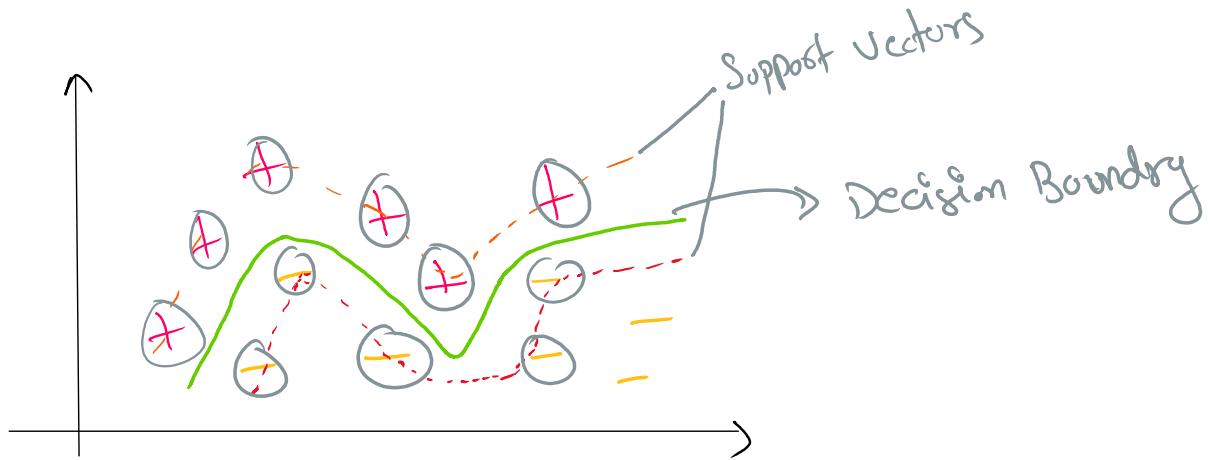
{Linear Kernel}

$$\Downarrow (1 + x_0 x'_0 + \dots + x_n x'_n)^P$$

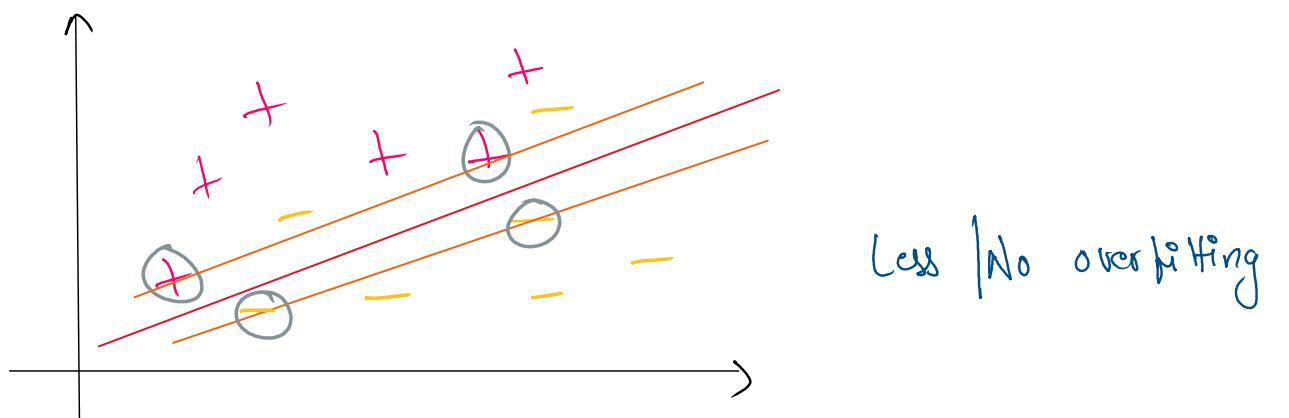
Soft Margin SVM

28 June 2024 22:23

There are two major reasons why the soft-margin classifier might be superior. One reason is your data is not perfectly linearly separable, but is very close and it makes more sense to continue using the default linearly kernel. The other reason is, even if you are using a kernel, you may wind up with significant over-fitment if you want to use a hard-margin.



Except for two data points all other acts as Support Vectors | margin for date identification; this will lead to over fitting



"soft margin" classifier, which allows for some "slack" on the errors that we might get in the optimization process.

* Slack (ξ) :

$$y_i(x_i \cdot w + b) \geq 1 - \xi$$

Our new optimization is the above calculation, where slack is greater than or equal to zero. The closer to 0 the slack is, the more "hard-margin" we are. The higher the slack, the more soft the margin is. If slack was 0, then we'd have a typical hard-margin classifier. As you might guess, however, we'd like to ideally minimize slack. To do this, we add it to the minimization of the magnitude of vector w :

$$\text{Minimize} = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i$$

Thus, we actually want to minimize $\frac{1}{2} \|w\|^2 + (C * \text{The sum of all of the slacks used})$. With that, we brought in yet another variable, C . C is a multiplier for the "value" of how much we want slack to affect the rest of the equation. The lower C , the less important the sum of the slacks is in relation to the magnitude of vector w , and visa versa. In most cases, C will be defaulted to 1.