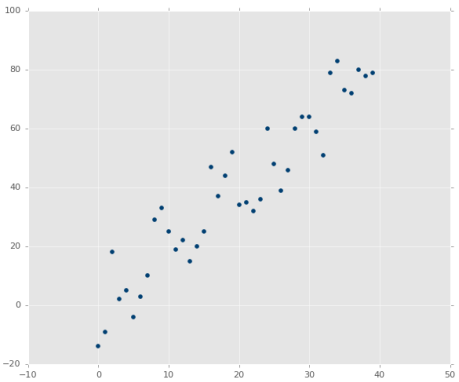
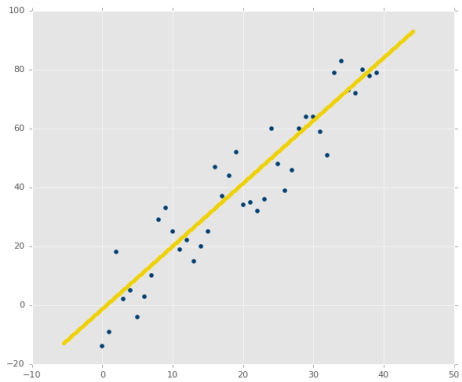
Regression

03 June 2024 18:57

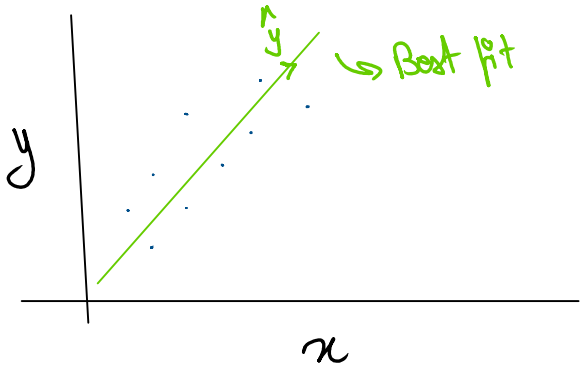
Simple Regression is used to find the best fit line of a dataset.





$y = mx + b$ ← y-intercept

Slope



Slope(m): $m = \dfrac{\bar{x}\cdot\bar{y} - \overline{xy}}{(\bar{x})^2 - (\overline{x^2})}$

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

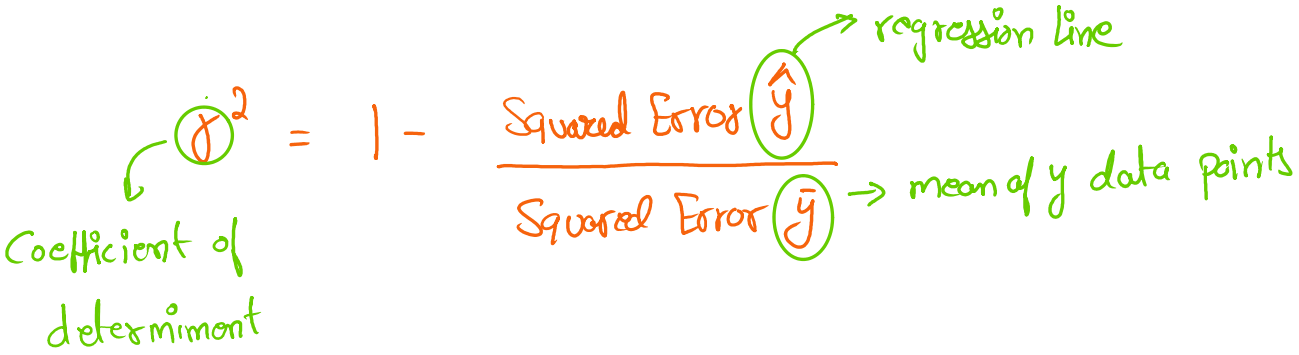
$\because y = mx + c \;\therefore\; \hat{y} = \dfrac{\bar{x}\cdot\bar{y} - \overline{xy}}{(\bar{x})^2 - (\overline{x^2})} \times x + (\bar{y}) - \left(\dfrac{\bar{x}\cdot\bar{y} - \overline{xy}}{(\bar{x})^2 - (\overline{x^2})}(\bar{x})\right)$

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 Squared error is either a mean or Sum of this values.



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

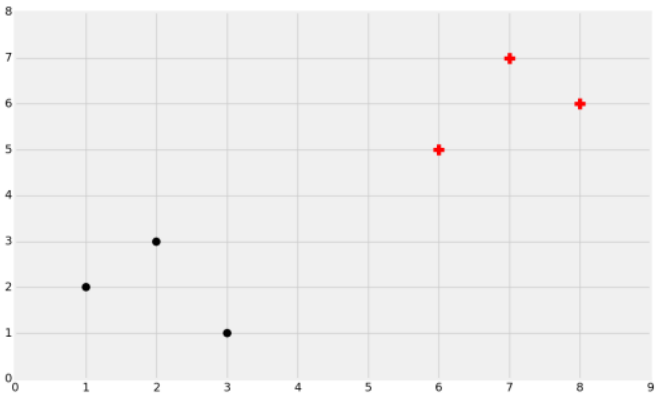
Ans) First, we want a way to normalize the error as a distance, so the error might be -5, but, when squared, that's a positive number. Another reason is to further punish for outliers. The "further" off something is, the more the it affects the error. This is simply the standard people use.

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.

KNN

10 June 2024 20:24

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



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

→ Working's —

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

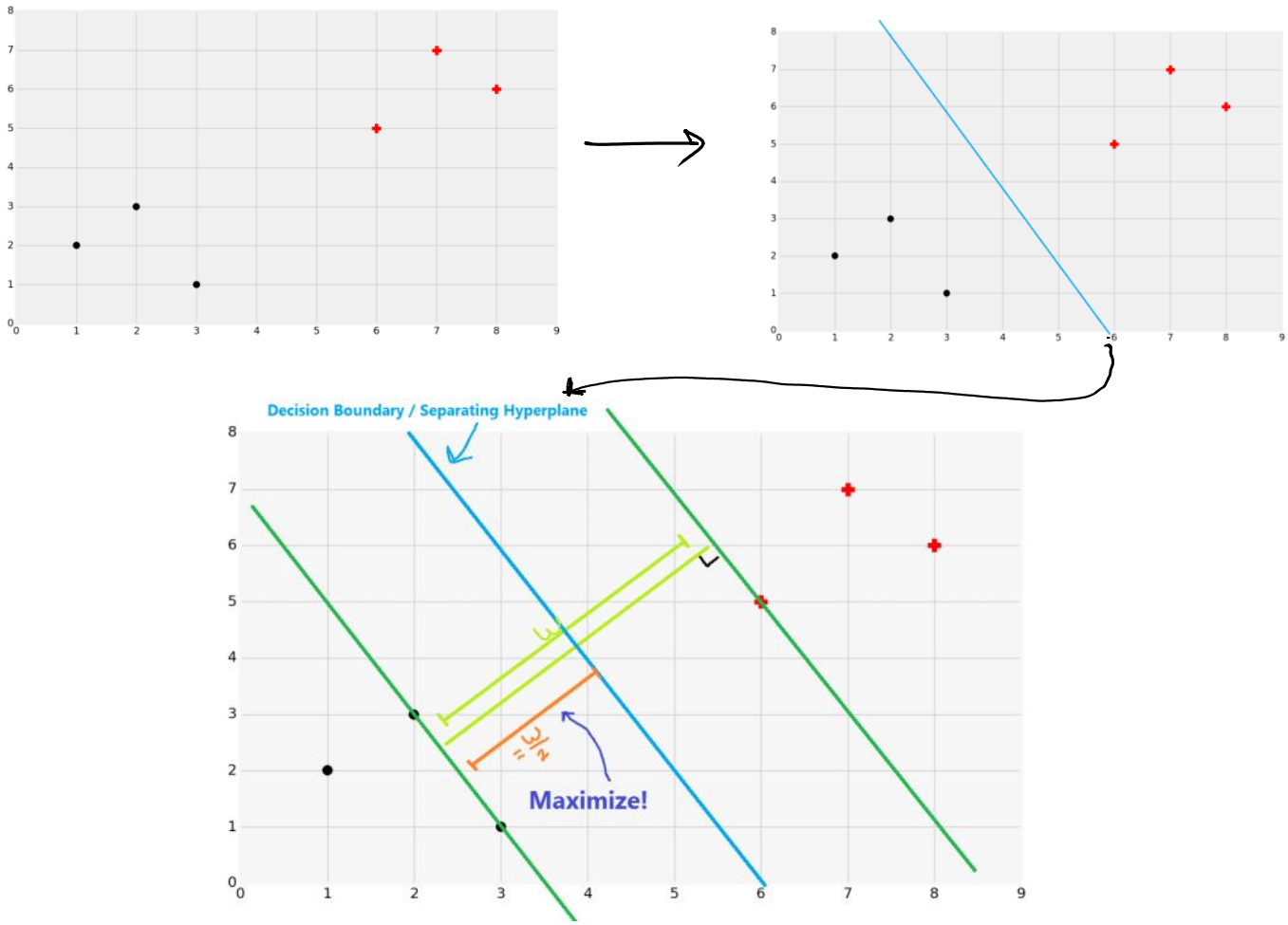
for data point to train_data.

(4) Sort and select 'K' from the saved point.

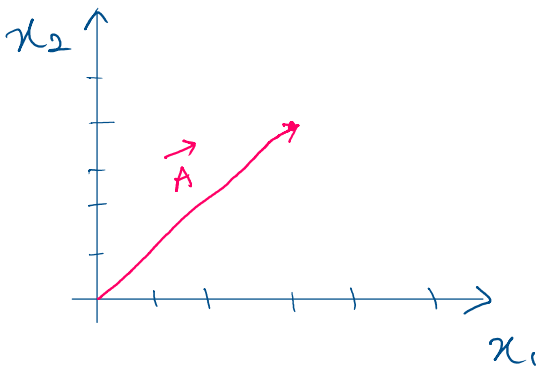
(5) From the 'K' selected groups, vote for most common classifier.

SVM https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

14 June 2024 20:55



→ Vector Basics



$\vec{A} = [3,4]$

$|A| = 5$

$\vec{A} = [1,3]$

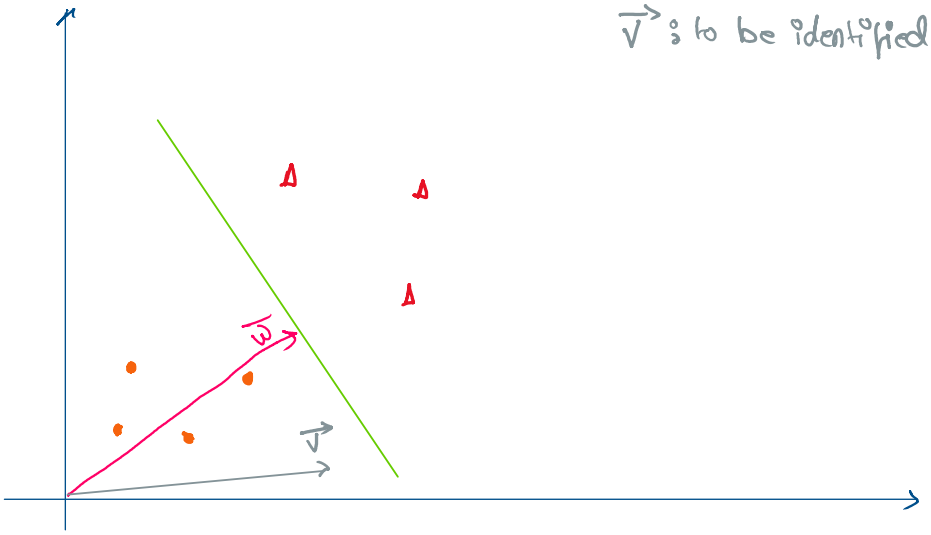
$\vec{B} = [4,2]$

$\vec{A} \cdot \vec{B} = [(1\times4)+(3\times2)]$

$= 4+6$

= 10

→ SVM Working :-



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

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

$\Rightarrow \bar{X}_{\bullet sv} \cdot \bar{w} + b = -1 \qquad \bar{X}_{\Delta sv} \cdot \bar{w} + b = 1$

y_i → $\Delta = 1$
→ $\bullet = -1$
(class)

$\Rightarrow$ Δ class $\bar{x}_i \cdot \bar{w} + b = 1 \Rightarrow y_i (x_i \cdot \bar{w} + b = 1) = (1)1$

$\Rightarrow$ Δ class $\quad \bar{x}_i \cdot \bar{w} + b = 1 \Rightarrow y_i(\bar{x}_i \cdot \bar{w} + b = 1) = (1)1$

$y_i = 1$

• class $\quad \bar{x}_i \cdot \bar{w} + b = -1 \Rightarrow y_i(\bar{x}_i \cdot \bar{w} + b = -1) = (-1)(-1)$

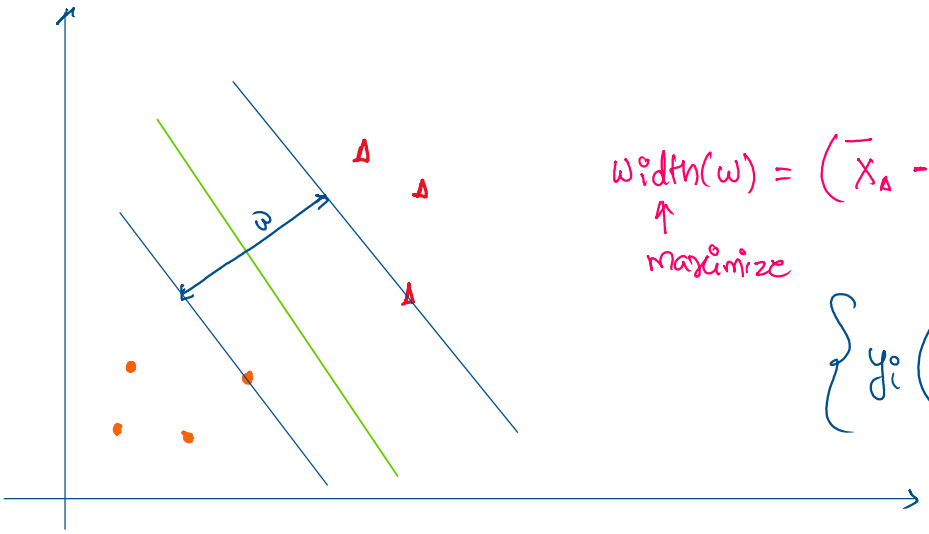
$y_i = -1$

Δ class $\Rightarrow y_i(\bar{x}_i \cdot \bar{w} + b) - 1 = 1 - 1 \to 0$

$y_i = 1$

• class $\Rightarrow y_i(\bar{x}_i \cdot \bar{w} + b) - 1 = 1 - 1 \to 0$

$y_i = -1$



$\text{width}(w) = (\bar{x}_\Delta - \bar{x}_\bullet) \cdot \frac{\bar{w}}{|w|}$

↑ maximize

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

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

*

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

$$\text{width (maximize)} = \frac{2}{|w|}$$

↳ minimize

* we want to minimize $\frac{2}{|w|}$ then we also want to minimize $|w|$.

Lets Says we want to minimize $\frac{1}{2}|w|^2$ as we alredy want to minimize $|w|$

$$\frac{1}{2}|w|^2 \; y_i (x_i . w + b) - 1$$

Lagrangian formulation:

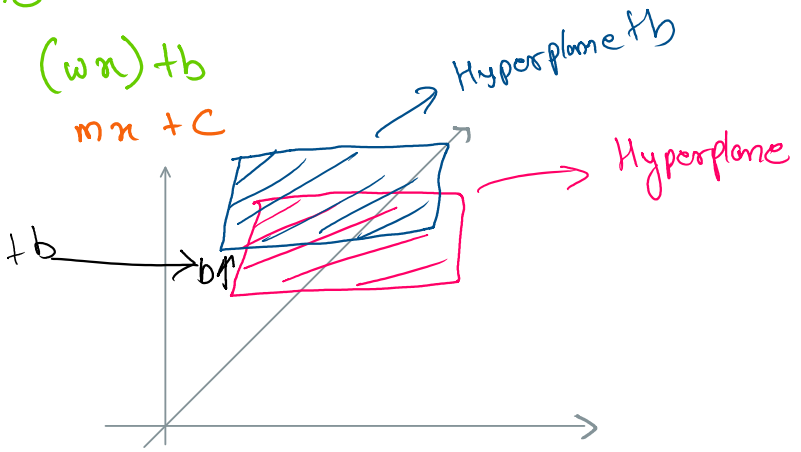
$$L(w,b) = \frac{1}{2}|w|^2 - \sum \alpha_i \left[y_i (\bar{x}_i \; \bar{w} + b) - 1 \right]$$

(min, max)

- Hyperplane

$(wx) + b$

$mx + C$



=) on differentiating wrt w & also differentiating wrt. b.

$$\frac{dL}{dw} = \bar{w} - \sum \alpha_i \; y_i \; x_i = 0 \Rightarrow \bar{w} = \sum \alpha_i y_i x_i$$

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

$$\max \; L = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j \, y_i y_j \cdot (\bar{x}_i \cdot \bar{x}_j)$$

quadratic

→ Concept :-

$(x \cdot w + b)$ ⇐ Hyperplane Equation

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

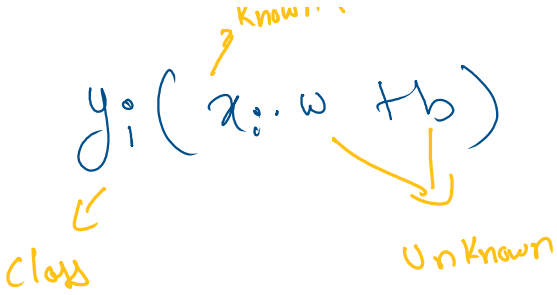
⇒ $\underline{\text{Sign}}\,(x_i \cdot w + b)$

< 0 +class 0 Decision Boundary > 0 -class

* The optimization objective is to minimize $|w|$ & maximize b. { $|w|$ & b ↑ }

Known Feature

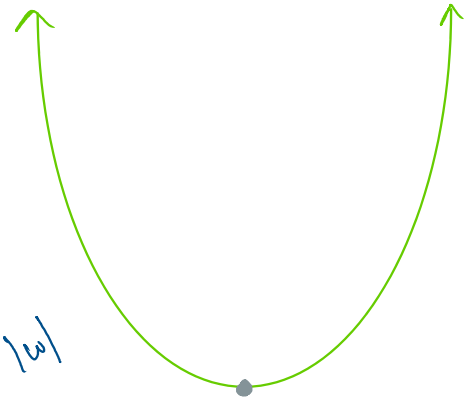
$\text{Sign}(x \cdot w + b)$



$$\max \; L = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j \, y_i y_j \cdot (\bar{x}_i \cdot \bar{x}_j)$$

quadratic

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



$\vec{w}_{1\times 2}$

$|w|$

Finding minima is the target.

Since we are trying to find minima of $|w|$ be need to check values in all the 4 quadrants for $\vec{w}$.

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

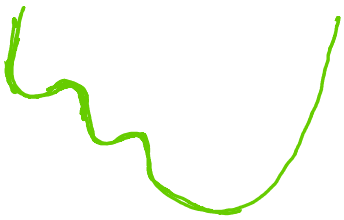
$y_i(x_i \cdot w + b) \geq 1$

$\text{mag} = \{ |w| : [\bar{w}, b] \}$

{finding lowest magnitude is target.

* local minima problem might occur

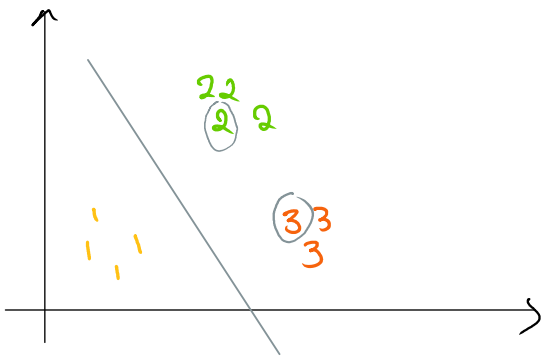
* local minima problem might occur

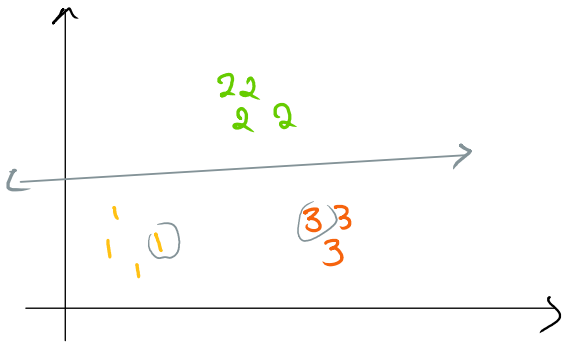


's target.

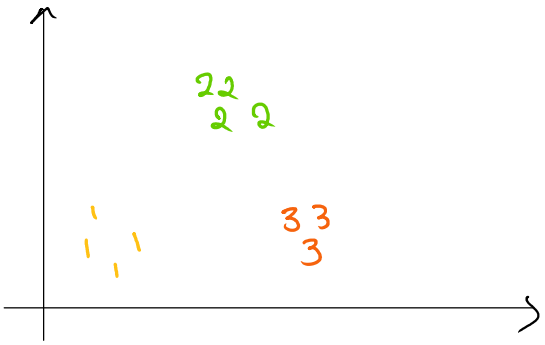
→ SVM Parameters -

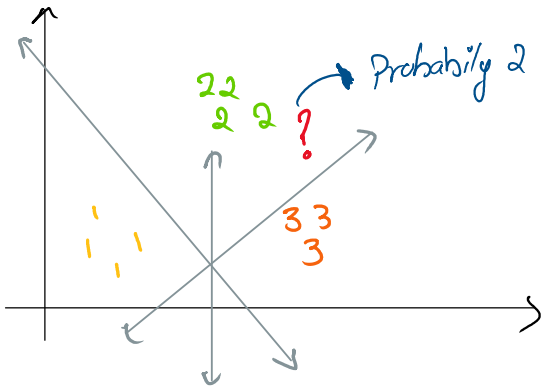
① One v/s Rest (OVR)

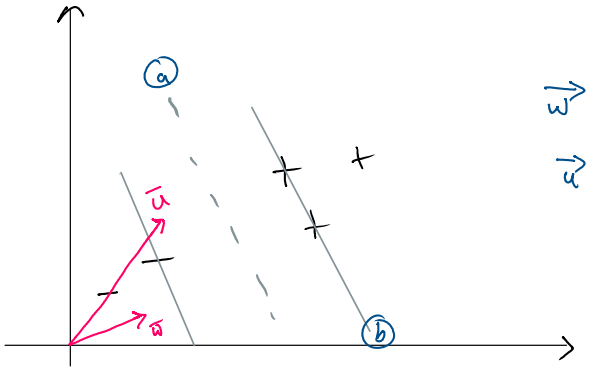




② One v/s One (OVO)-







$\vec{w}$ is $\perp$ to (a)

$\vec{u}$ Points an unknown value, lies on - or + side

$\bar{w} \cdot \bar{u} \geq c$ {c, Some Constant 'c'}

⇒ dot product has taken projection on $\bar{w}$, the bigger the projection is the further the projection lie on $\bar{w}$.

$\bar{w} \cdot \bar{u} + b \geq 0$ Then +

Decision Rule

$\bar{w} \cdot \bar{x}_+ + b \geq 1$ {for pt to lie out the the decision boundry (b)}

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

⇒ let y_i be Such that $y_i = +1$ for + Samples

$y_i = -1$ for - Samples

(+) $y_i (\bar{w} \cdot \bar{x}_i + b) \geq y_i(1)$ ⇒ $y_i(\bar{w} \cdot \bar{x}_i + b) \geq 1$

(-) $y_i (\bar{w} \cdot \bar{x}_i + b) \leq y_i(-1)$ ⇒ $y_i(\bar{w} \cdot \bar{x}_i + b) \geq 1$

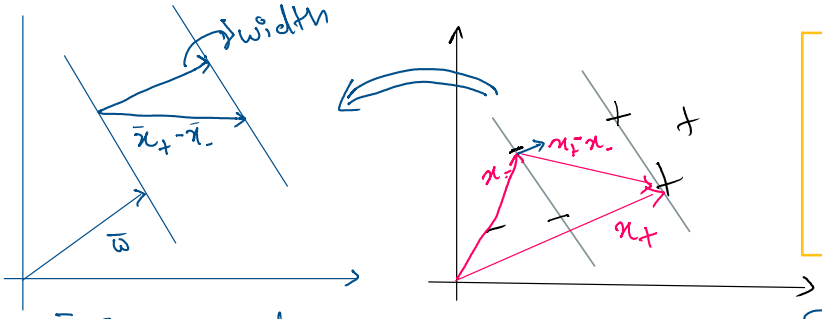
(-) $(y_i)(-1)$

(-)
(-1)
(+)

(-1) (-1)
(+1)
$\geq$ (1)

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

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



$\bar{w}$ is a normal

$\frac{\bar{w}}{|w|}$ is an unit vector

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

{using $y_i(\bar{x}_i \cdot \bar{w} + b) = 0$}

$$\Rightarrow = (\bar{x}_+ - \bar{x}_-) \cdot \frac{\bar{w}}{|w|}$$

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

$$\Rightarrow \boxed{\text{width} = \frac{2}{|w|}} \quad - (3)$$

To get the max dist b/w decision bondry we need to maximize $\frac{2}{|w|}$

↳ maximize $\frac{1}{|w|}$ → minimize $|w|$

↳ maximize $\frac{1}{|w|}$ → minimize $|w|$

↳ minimize $\frac{1}{2}|w|^2$

for mathematical convinience.

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

Lagrange Multipliers

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

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

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

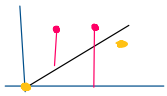
$$\boxed{L = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j}$$

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

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

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

For Non-linear Seprable Data:-



we can seprate Non linear data we change space.

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

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

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

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

28 June 2024 20:51

- Kernels are Similarity function takes two inputs & returs a Similarity using inner (dot) product.

- Kernels allows us to work in many dimensions w/o paying processing Cost to do so.

$$K(x, x') = z \cdot z' \quad \{K \text{ or } \phi\}$$

$$\begin{cases} z \rightarrow \text{function}(x) \\ z' \rightarrow \text{function}(x') \end{cases}$$

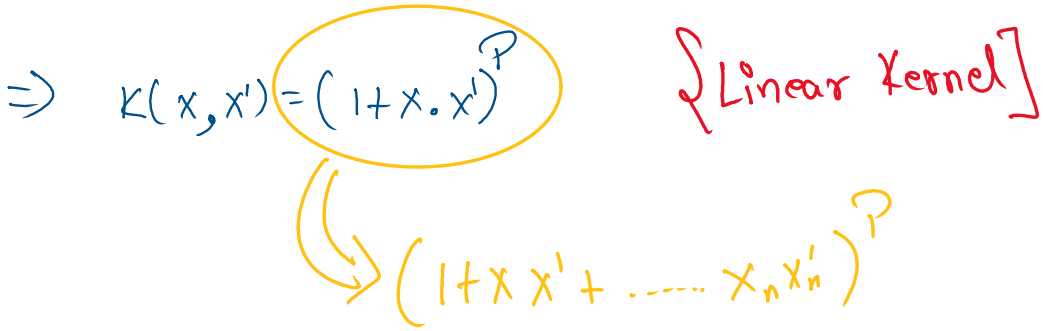
For Example, Lets assume a polynomial Kernel to 2nd order Polynomial.

$$x : [x_1, x_2] \qquad z : [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2]$$

$$z' : [1, x_1', x_2', x_1'^2, x_2'^2, x_1' x_2']$$

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

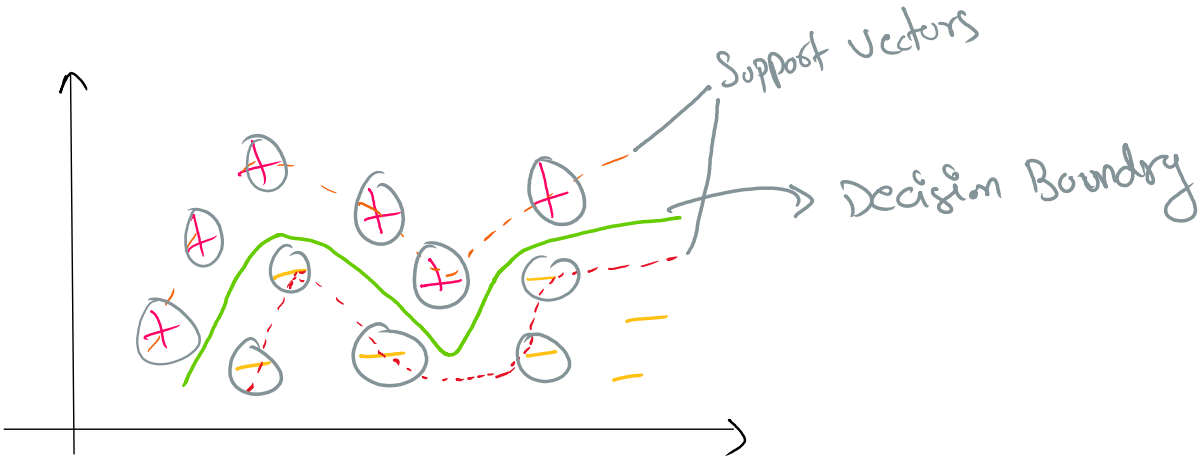
$$= 1 + x_1 x_1' + x_2 x_2' + x_1^2 x_1'^2 + x_2^2 x_2'^2 + x_1 x_1' x_2 x_2'$$


$\Rightarrow K(x, x') = (1 + x \cdot x')^p$
[Linear Kernel]
$(1 + x x' + \ldots x_n x'_n)^p$

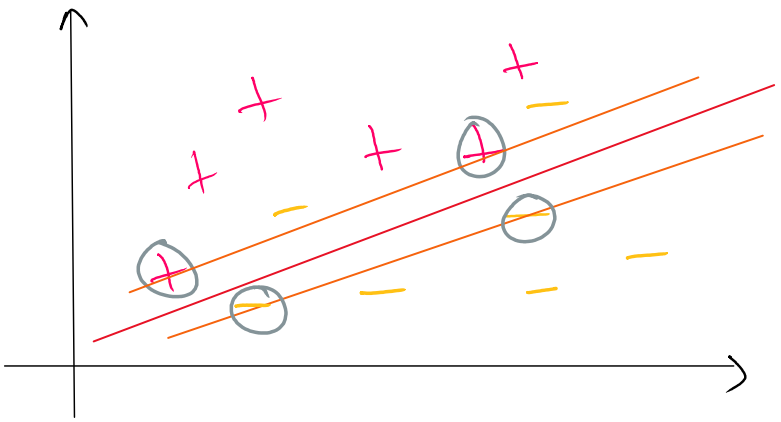
Soft Margin SVM

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 data identification; this will lead to offer fitting



Less / No overfitting

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

* Slack (ξ_i) :

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

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 1/2||w||^2 + (C * The sum of all of the slacks used).With that, we brough 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.

K-Means

02 July 2024 18:22

The objective of clustering is to find relationship & meaning.

Clustering is mostly used as a step in "Semi-Supervised machine learning".

Big-data is prime area for clustering.

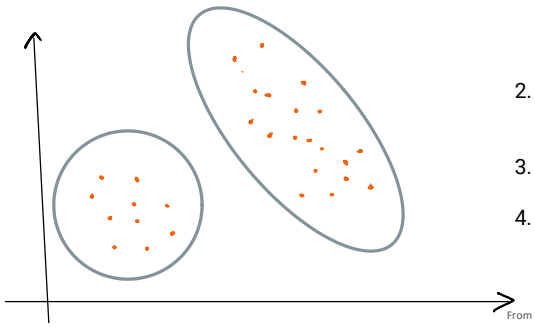
Finding patterns in data given.

Clustering

Type of unsupervised Learning

- Flat
  - we define no. of classes to be formed
- Hierarchial
  - machine figures out the no. of groups to be formed

→ K-Means



K=2

Flat

K-Means algorithm:

1. Take entire dataset, and set, randomly, K number of centroids. Centroids are just the "centres" of your clusters. To start, I typically just take the first K values in my dataset and have those as the start, but you could also randomly select them if you wanted. It should not matter, but, if you are not optimizing for some reason, it may make sense to try to shuffle the data and try again.
2. Calculate distance of each feature set to the centroids, and classify each feature set as the centroid class closest to it. Centroid classes are arbitrary, you will likely just call the first centroid 0, the second centroid 1...and so on.
3. Once you have classified all data, now you take the "mean" of the groups, and set the new centroids as the mean of their associated groups.
4. Repeat #2 and #3 until you are optimized. Typically, you measure optimization by movement of the centroid. There are many ways to do this, we're just going to use percent change.

From <https://pythonprogramming.net/machine-learning-clustering-introduction-machine-learning-tutorial/?completed=/support-vector-machine-parameters-machine-learning-tutorial/>

K-Mean, downside is that it tries to creat equal size clusters, hard time diffeent size groups because of euclidean distance.