Classification

Monday, June 21, 2021 6:21 PM

1. KNN

Algorithm :

1. Initialize the List of Distances = [ ].
2. For each of xj in the dataset:
   Compute the distance d(xi, xq) = di
   Store them in list as (xi, yi, di)

3. Sort the list on the basis of distances.

4. Take the smallest 'k' distances and store them in list 'KNN' .

5. For each xi in KNN list:
   if yi is true:
   count_pos = +1
   else:
   count_neg = +1

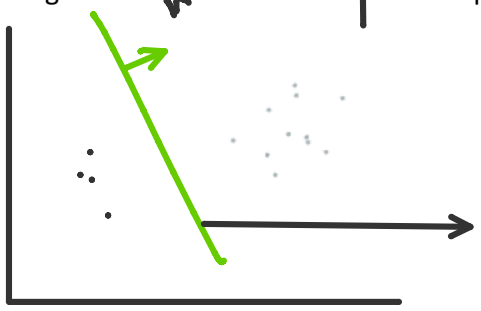
6. If (count_pos > count_neg) :
   print('Positive class')
   else:
   print('Negative class')

There is no training in KNN algorithm.

1. Logistic Regression

Linear Regression

- Supervised Learning
- Regression Task
- $D_n = \{ (x_i, y_i)_{i=1}^{n} \mid y_i \in R \}$
- Task : To find the line that best fits The training data.



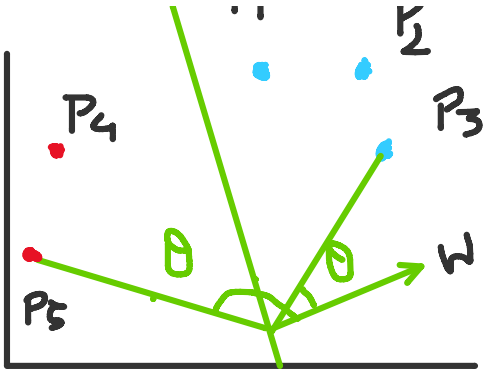
Logistic Regression

Supervised Learning
Classification Task
$D_n = \{ (x_i, y_i)_{i=1}^{n} \mid y_i \in \{-1, 1\} \}$
Task : Find the line that BEST
Separates Line between +ve & -ve

W = Norm of the Hyperplane/ plane
i.e. It always perp. to the line

y = mx + c ≅ $w^T x + w_0$

$\vec{w} . p1 = +ve$



$\overrightarrow{w}.p1 = +ve$
$\overrightarrow{w}.p2 = +ve$
$\overrightarrow{w}.p3 = +ve$
$\overrightarrow{w}.p4 = -ve$
$\overrightarrow{w}.p5 = -ve$

We have assumed that the data is separable by line/plane.

So the formula is,

$$\boldsymbol{Signed\ Distance} = \boldsymbol{\Sigma_i^n}\ (\boldsymbol{y_{actual}} * \boldsymbol{Y_{predicted}})$$

Where $\boldsymbol{Y_{predicted}}$: $|\boldsymbol{w}|.|\boldsymbol{p}|\boldsymbol{cos\theta}$ $\boldsymbol{or}$ $\boldsymbol{mx+c}$

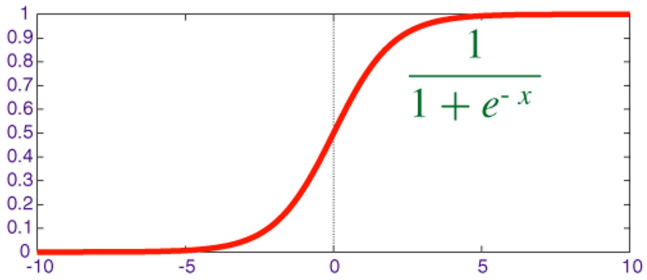
The vectors which gives +ve values from above formula are considered to be correctly classified whereas vectors which gives overall -ve values are considered to be misclassified.

Best Separator will gives less misclassifications or more correct classification

Optimum value of m & c :

m*, c* = $arg_{m,c}$ $\max\{\ \boldsymbol{\Sigma_i^n}\ \boldsymbol{y_i} * (\boldsymbol{mx_i + c})\ \}$
(Maximizing the signed distance)

After removing outliers, still there are some outliers which were not detected by EDA, Hence we have to convert the Signed distance to sigmoid function



It basically more robust to outliers. It neglects the effect of outliers. It shows the tapering effect. We are doing some sort of scaling on signing distance. Its value between 0 to 1.
So,
m*, c* = $arg_{m,c}$ $\max\{\ \boldsymbol{\Sigma_i^n}\ \boldsymbol{y_i} * (\boldsymbol{mx_i + c})\ \}$

= $arg_{m,c}$ $\max\{\ \sigma\ [\boldsymbol{\Sigma_i^n}\ \boldsymbol{y_i} * (\boldsymbol{mx_i + c})]\ \}$

$$= arg_{m,c} \quad \max\{ \Sigma_i^n \frac{1}{1+exp\{ -y_i * (mx_i+c) \}} \}$$

Optimization theory :

max f ≈ min 1/f

max f ≈ - min 1/f

$$= arg_{m,c} \quad \min\{ \Sigma_i^n \; 1 + exp\{ -y_i * (mx_i + c) \} \}$$

Decision Tree :

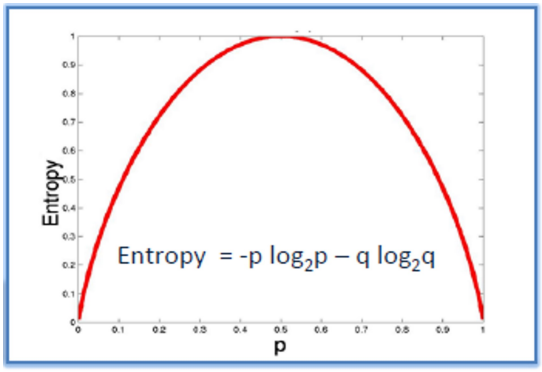
A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Entropy (H(y)) : Measure of Randomness. [y is o/p]

It captures the randomness/ uncertainty of variable. It values always lies between 0 and 1. It is 0 when all points in variable are same. If we have 50% +ve and 50% -ve, then Entropy is 1.

0 <= H <= 1

$H(y) = -\Sigma_i^k \; p(y_i). \; log_2(p\,(y_i))$



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain .

$$IG(Y, f) = \; H(y) \; - \; \Sigma_i^k \{ \frac{|D_i|}{|D|} * H_{Di}(y) \}$$

where $\Sigma_i^k \{ \frac{|D_i|}{|D|} * H_{Di}(y)$: Weighted Entropy

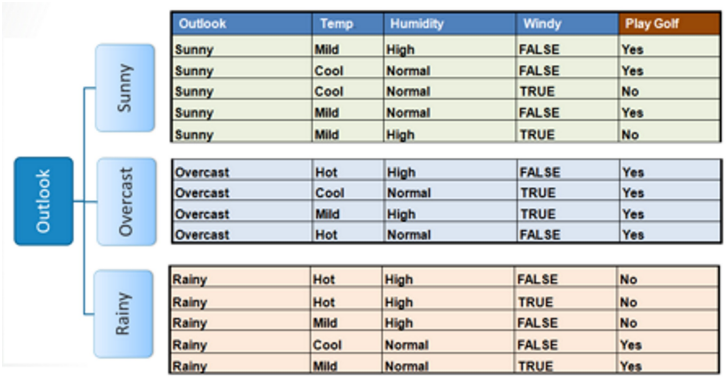
Step 1: Calculate entropy of the target.

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy

is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

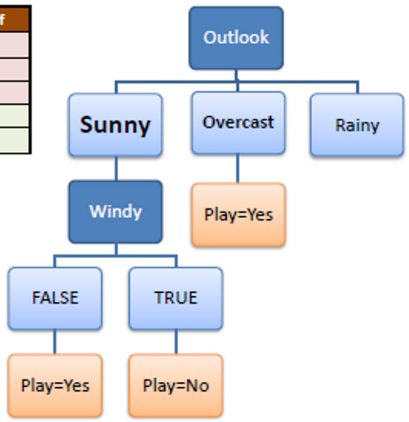
Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

Step 4 : those attributes who have more than 0 Entropy needs further split while those have 0 doesn't need.



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Advantages & Disadvantages of Decision Tree :

- We don't require to convert categorical values to one hot encoding because Decision Tree never compares the relationship between categorical values. We need to do just label Encoding and it also avoids to increase Dimensionality.
- We don't need to do Standardization/ Normalization to data because we don't need to keep all data in similar range like in other algorithms(because they have to calculate some distance in algorithm).

- Overfitting : (High Variance and low Bias)
- Real Valued Column : If model has real valued column, then for every it check entropy less than and entropy greater than (for Ex. **H < 170 and H > 170** ∀ **point**) from the whole column and if data is huge it will increases the computation time for algorithm.

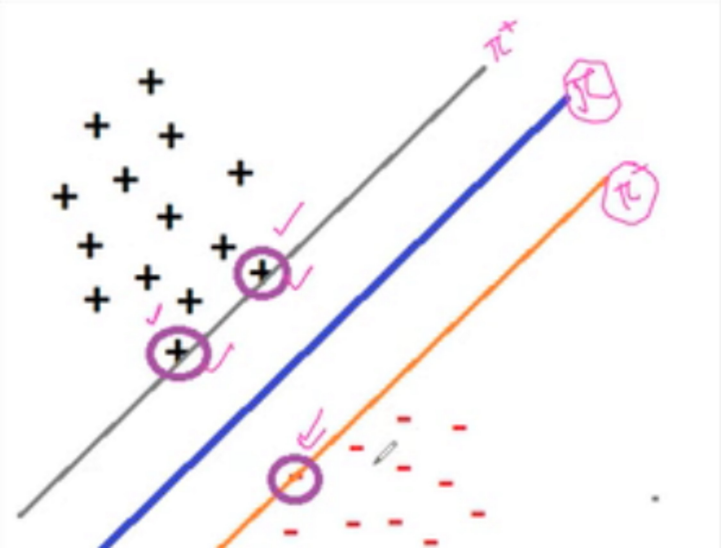
Gini Impurity : measure of randomness similar to entropy. It require less computational power than Entropy.

$$I_g = 1 - \Sigma_i^k \, p(y_i)$$

-0.5 <= Ig <= 0.5

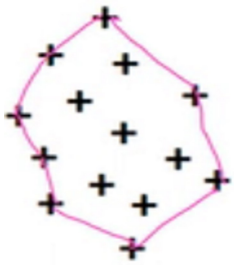
SVM (Support Vector Machine) :

SVM tries to find the hyperplane that best separate the +ve and -ve as widely as possible. Also known as MARGIN MAXIMIZING hyperplane.



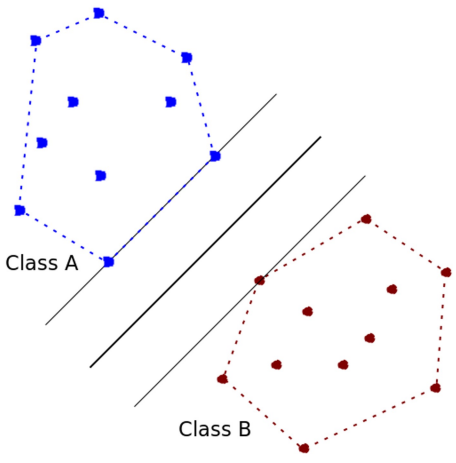
Π+ & π : Support Vectors

Convex Hull : A figure in which if we draw a line between two points in figure, the line always present inside the figure.

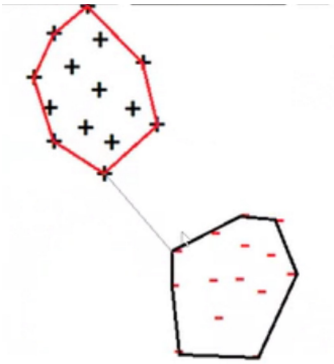


Steps for SVM :

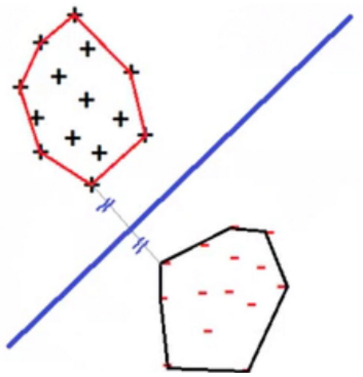
1. Draw the convex Hull for +ve and -ve points.



2. Finding the shortest line between the classes.



3.Bisect the line into two parts.



Kernel trick :
A Kernel Trick is a simple method where a Non Linear data is projected onto a higher dimension space so as to make it easier to classify the data where it could be linearly divided by a plane.

In Logistic Regression, we have to find the best line that separates the +ve and -ve .

$y_{actual} * y_{pred} \geq 0$ i.e signed dist. is +ve.

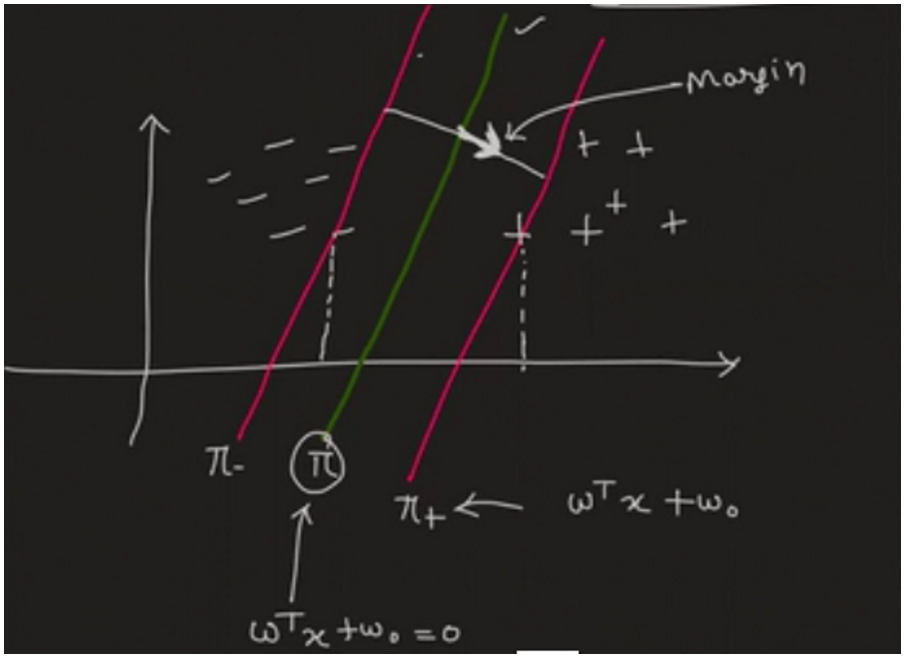
In SVM, we have to maximizing the margin distance(π_+ , π_-)

Hard Margin SVM Formulation :

Maximum Margin such that every point in the training data is correctly classified.

Max. Margin such that

$$y_i * (mx_i + c) \geq 0 \quad \forall \ i$$



Line equation for $\pi = mx + c$ or $w^T x + w_o$
Similarly for π_+ , $\pi_- = mx + c$ or $w^T x + w_o$ (+/-)k

Only difference between them that they have different intercept w_o .
So,

$\pi_+ : w^T x + w_o = 1$ (Take 1 for assumption)
$\pi_- : w^T x + w_o = -1$

Then,

$$w^T x_{pos} + w_o = 1$$
$$- \quad w^T x_{neg} + w_o = -1$$

$$= \quad w^T x_{pos} - w^T x_{neg} = 2$$

$$= \quad w^T (x_{pos} - x_{neg}) = 2$$

$$= \quad \frac{w^T (x_{pos} - x_{neg})}{|w|} = \frac{2}{|w|}$$

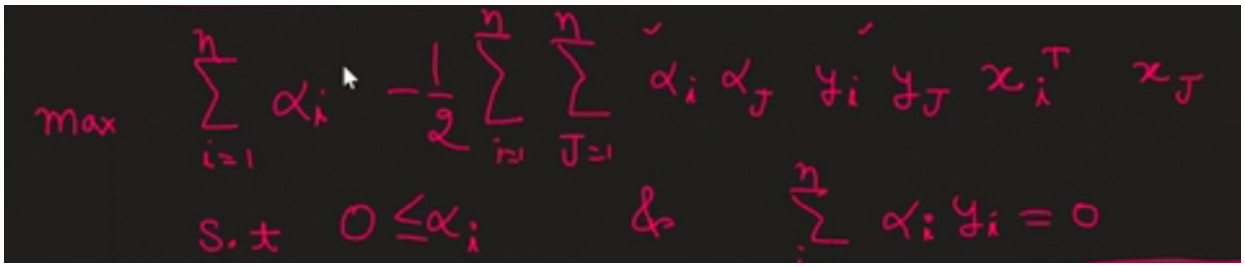
This thing is called projection of distance between margin on $\vec{w}$

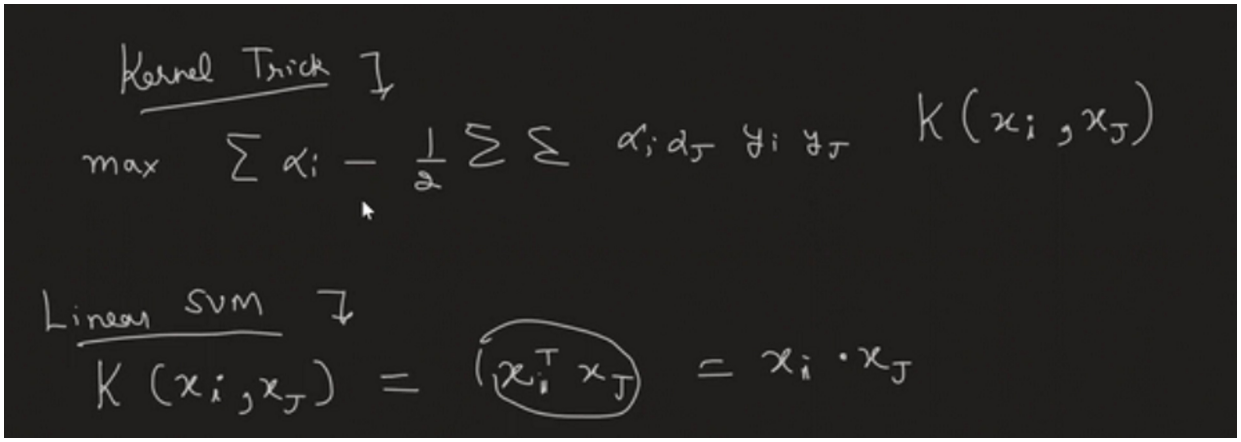
Max. $[\frac{2k}{|w|}]$
such that

$$y_i * (w^T x + w_o) \geq 0 \quad \forall \ i$$

This is called Hard Margin SVM Formulation.

Dual Form :
Lagrange Multiplier helps to achieve the Dual Form





They use quadratic equation to achieve higher dimension and to project data to higher dimension because Linear SVM is not able to classify all types of data.
More higher polynomial equations are used to classify the data.

Quadratic Kernel SVM :

$$K(x_i, x_j) = (x_i, x_j^2)$$

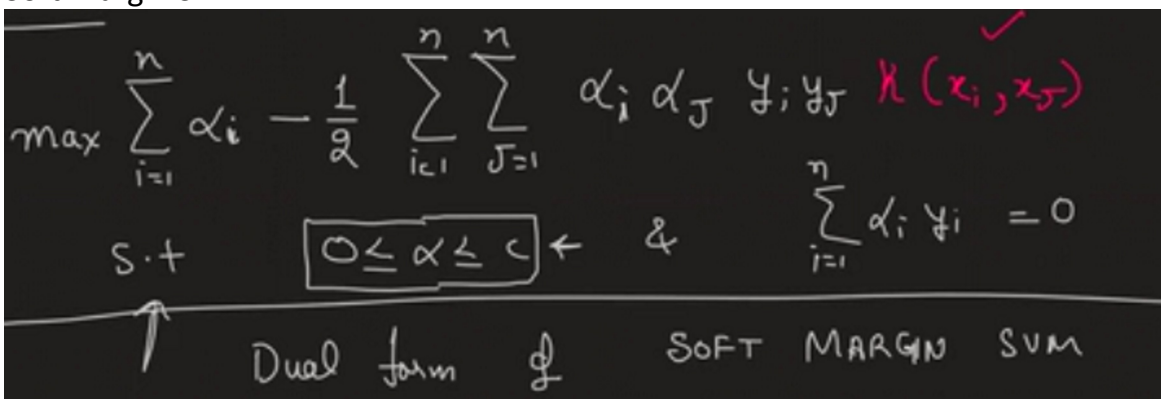
Polynomial Kernel SVM :

$$K(x_i, x_j) = (x_i, x_j^d)$$

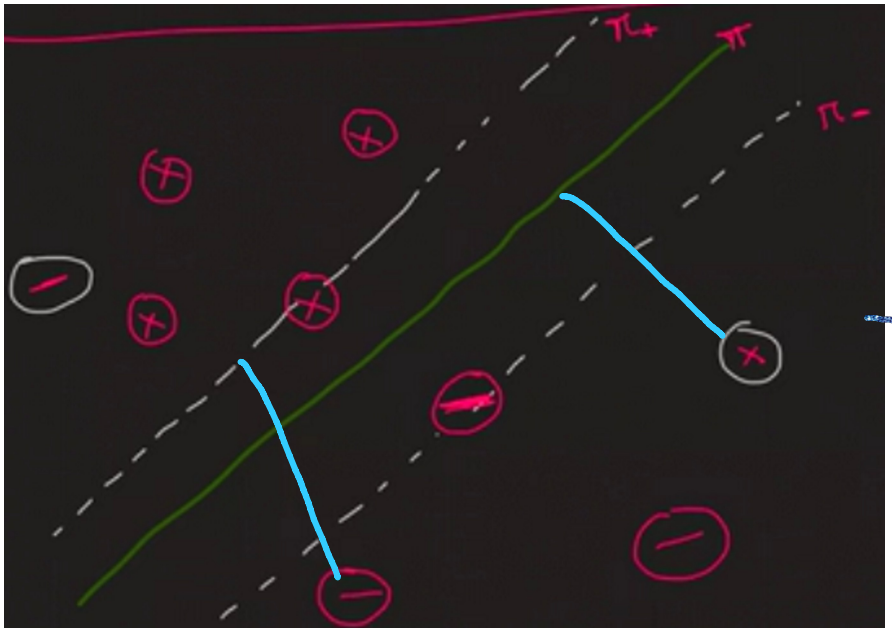
RBF Kernel(Radial basis function) :

$$K(x_i, x_j) = \exp\left\{\frac{-||x_i - x_j||}{2\sigma^2}\right\}$$

The best thing about RBF is gives results most of the time but you have to do experiments.
Kernel Trick has huge training time complexity, that's why svm has higher T.C.
Soft Margin SVM :



Only difference in $0 <= \alpha < = c$ part.



Dist = +1 * (- 2.5)
= -2.5

Dist = -1 * 2.5
= -2.5

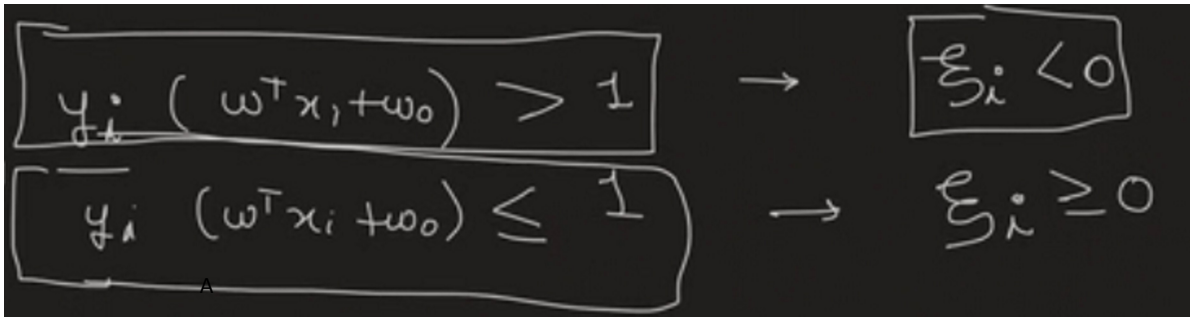
We have to use slack variable means we have to subtract it with 1 without changing its value.

(1 - (+3.5)) = -2.5 where 3.5 value is called slack variable denoted by ξ_i $(zeta)$.

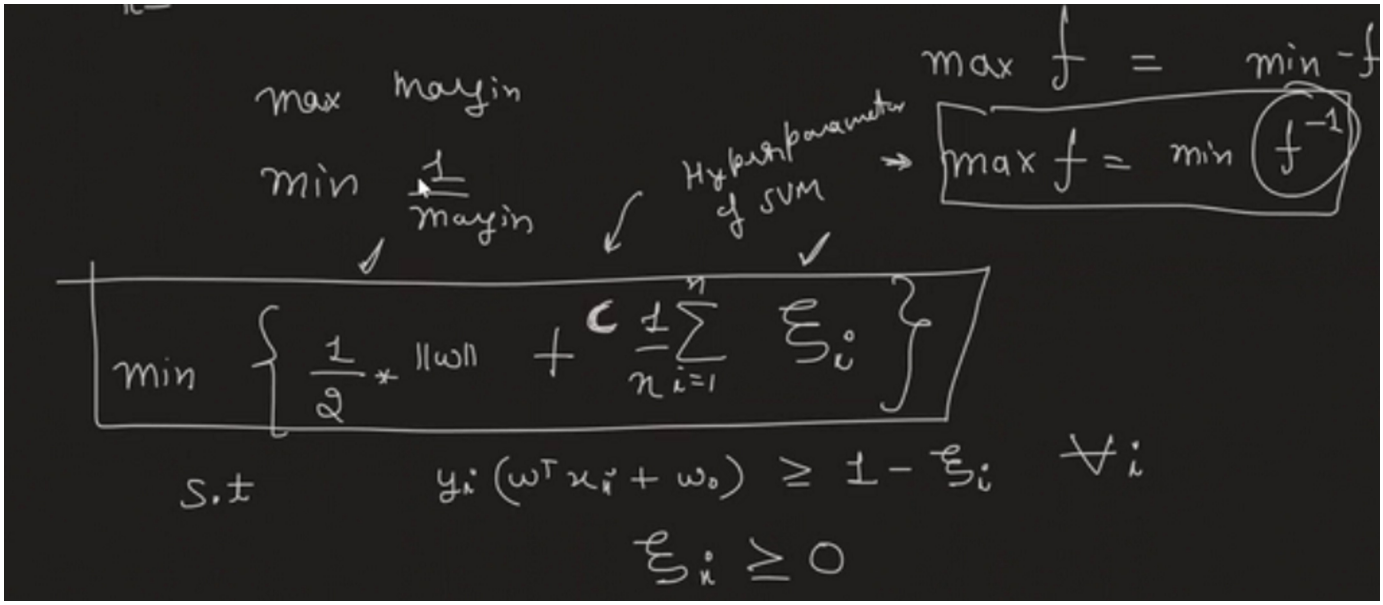
So, in general

if $\xi_i \geq 0$, $point\ is\ misclassified\ according$ to π_+ , π_-
& $\xi_i \leq 0$, $point\ is\ classified\ according$ to π_+ , π_-

∴ ξ_i ↑, then the point goes to incorrect direction(π_+ , π_-)

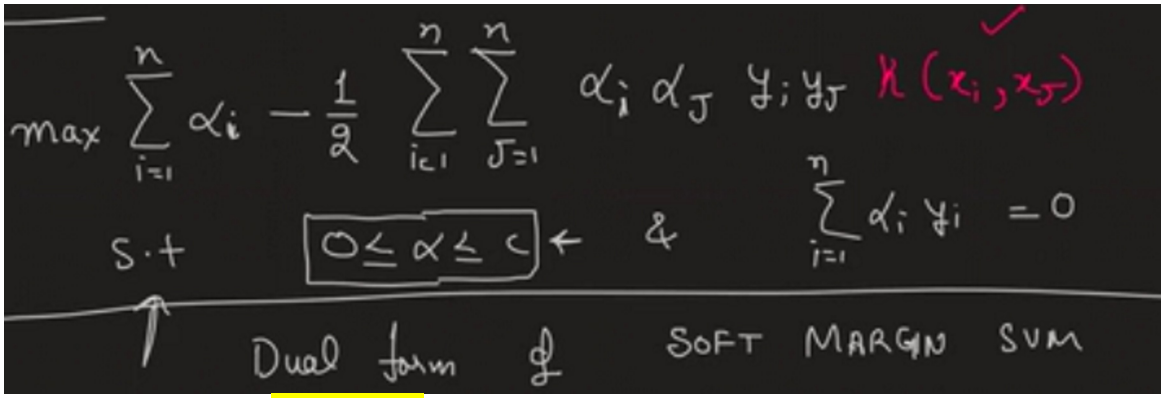


According to Optimization theory,



This is Soft Margin SVM formulation for more and more noise.
Where $\xi_i \geq 0$ *means data contains more noise*.

$c * \frac{1}{n}\sum_{i=1}^{n} \xi_i$ *is called Hinge Loss.*



Only difference in 0<= α < = c part of Hard form

KNN > SVM > LogReg in terms of time complexity.

<u>Naïve Bayes</u> :

Conditional Probability : P(E1 | E2) = $\frac{P(E1 \cap E2)}{P(E2)}$
Independent Events : P(E1 | E2) = P(E1) ∵ E2 and E1 are independent.

Suppose P(E1 | E2) = $\frac{P(E1 \cap E2)}{P(E2)}$ and P(E2 | E1) = $\frac{P(E2 \cap E1)}{P(E1)}$

Then, According to Set Theory P(E1 ∩ E2) = P(E2 ∩ E1)
=> P(E2 | E1) = $\frac{P(E1 \cap E2)}{P(E1)}$
=> P(E1 ∩ E2) = P(E2 | E1) * P(E1) ------ (Equation : 2)

Put Equation 2 in P(E1 | E2)

=> **P(E1 | E2) =** $\frac{\mathbf{P(E2 \mid E1) * P(E1)}}{\mathbf{P(E2)}}$

This is called Bayes Theorem.

In this Algorithm we have to calculate P($C_k \mid x_q$)
Where C_k - Class Label and x_q = Query or datapoints given

We have to find the label which has highest P($C_k \mid x_q$) and Take the label as Output.

For Example in Iris Dataset :
P(Species = Setosa | x_q) = ?
P(Species = Virginica | x_q) = ?

Select the label which has maximum probability.

P($C_k \mid x_q$) = $\frac{P(x_q \mid C_k)\, P(C_k)}{P(x_q)}$ According to Bayes theorem

= $\frac{P(x_q \cup C_k)}{P(x_q)}$ or $\frac{P(x_q, C_k)}{P(x_q)}$

= $\frac{P(x_1, x_2, x_3, \ldots x_d, C_k)}{P(x_q)}$

= $\frac{P(x_1 \mid x_2, x_3, \ldots x_d, C_k) \cdot P(x_2 \mid x_3, \ldots x_d, C_k) P(x_3 \mid \ldots x_d, C_k) P(x_d \mid C_k) \cdot P(C_k)}{P(x_q)}$

This is called Chain Rule of Probability

= $\frac{P(x_1 \mid C_k)\ P(x_2 \mid C_k) P(x_3 \mid C_k) \ldots P(x_d \mid C_k)\, P(C_k)}{P(x_q)}$

Naïve Assumption : Input variables are independent of each other.

= $\frac{P(C_k)}{P(x_q)} \prod_i^d P(x_i \mid C_k)$

$\hat{y} = arg_k \max \{ \frac{P(C_k)}{P(x_q)} \prod_i^d P(x_i \mid C_k) \}$

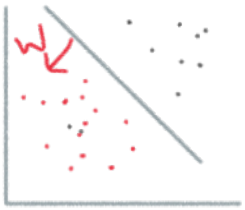
Different types of NB :

1. Multinomial NB : Text Data
2. Gaussian NB : Real Value
3. Bernoulli NB : Binary Feature

Maximum Likelihood / probabilistic approach for Logistic Regression
Xq - query point

wTx - dot product

Here y =(0,1)



Given a Xq, what is its probability that it belongs to class 1
P(y = 1 | Xq) = σ (wTx)

P(y = 0 | Xq) = 1 - σ(wTx)

P (Yi | Xq) = $(\sigma(wTx))^{Yi} + (1 - \sigma(wTx))^{1-Yi}$ also called likelihood of Xq
If Yi = 1 = $(\sigma(wTx))^{Yi}$
Yi = 0 = $(1 - \sigma(wTx))^{Yi}$

We need to maximize the likelihood for each training data point.

$$max \prod_{i=1}^{n} P(Yi \mid Xi) = \max(P(Y1|X1) + P(Y2 \mid X2) \ldots\ldots + P(Yn \mid Xn))$$

According to optimization theory,
max f(x) = log(max f(x)), it is monotonically increasing function

$$\begin{aligned} max \prod_{i=1}^{n} P(Yi \mid Xi) &= \max(\log(P(Y1|X1)) + \log(P(Y2 \mid X2)) \ldots\ldots + \log(P(Yn \mid Xn))) \\ &= max \sum \log((P(Yi|Xi)) \\ &= max \sum \log((\sigma(wTx))^{Yi} + log(1 - \sigma(wTx))^{1-Yi}) \\ &= max \sum \mathrm{Yi}.\log((\sigma(wTx))^{\square} + (1 - Yi) log(1 - \sigma(wTx))^{\square}) \\ &\text{since } \log b^{a} = a log\ b \\ &= max \sum \mathrm{Yi}.\log((\sigma(wTx))^{\square} + (1 - Yi).log(1 - \sigma(wTx))^{\square}) \\ &= -min \sum \mathrm{Yi}.\log((\sigma(wTx))^{\square} + (1 - Yi) log(1 - \sigma(wTx))^{\square}) \\ \text{m*, c*} &= -min \sum \mathrm{Yi}.\log(Pi) + (1 - Yi) log(1 - Pi) \end{aligned}$$