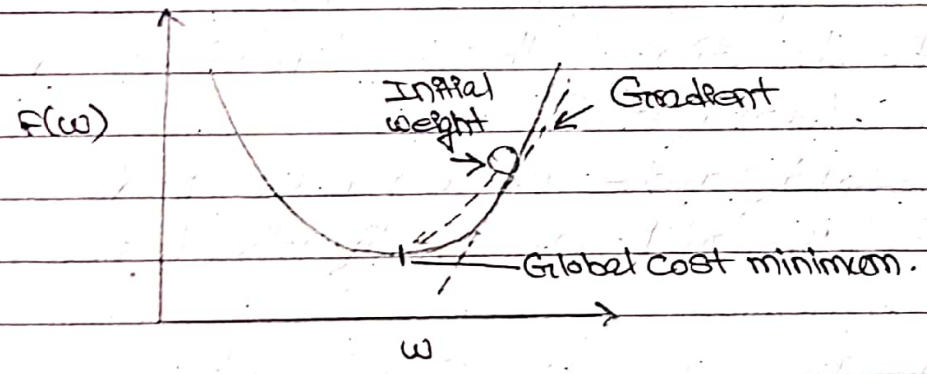
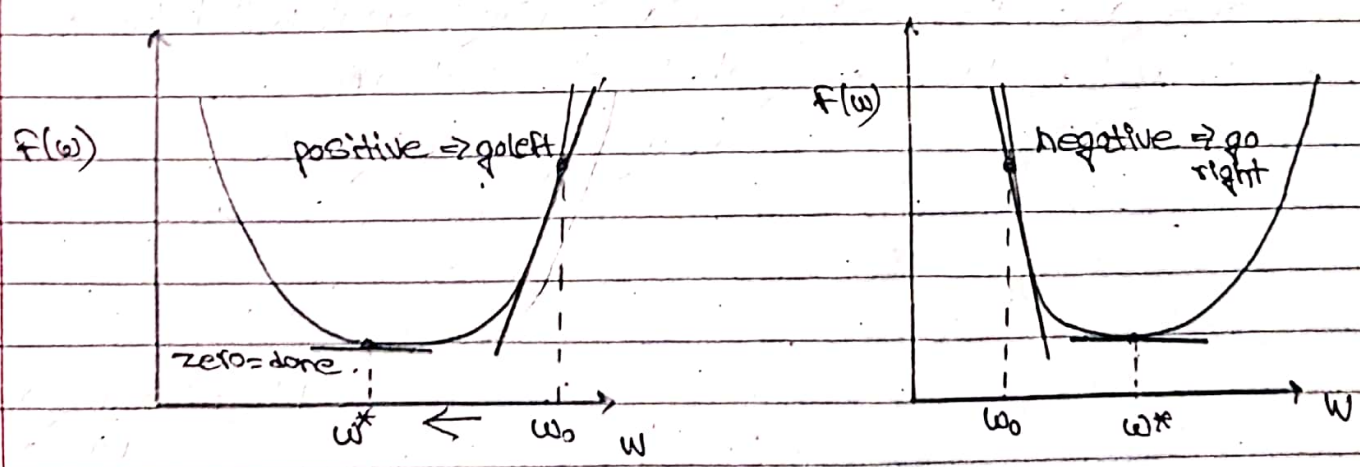
Page No.
Date : / /

Unit: 2
Supervised Learning

* Gradient Descent:

- It is an optimization algorithm used to minimize some convex function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.
- In machine learning, we use gradient descent to update the parameters of our models. Parameters refers to coefficients in Logistic Regression & weights in neural network.





Note: We can also think of a gradient as the slope of a function.

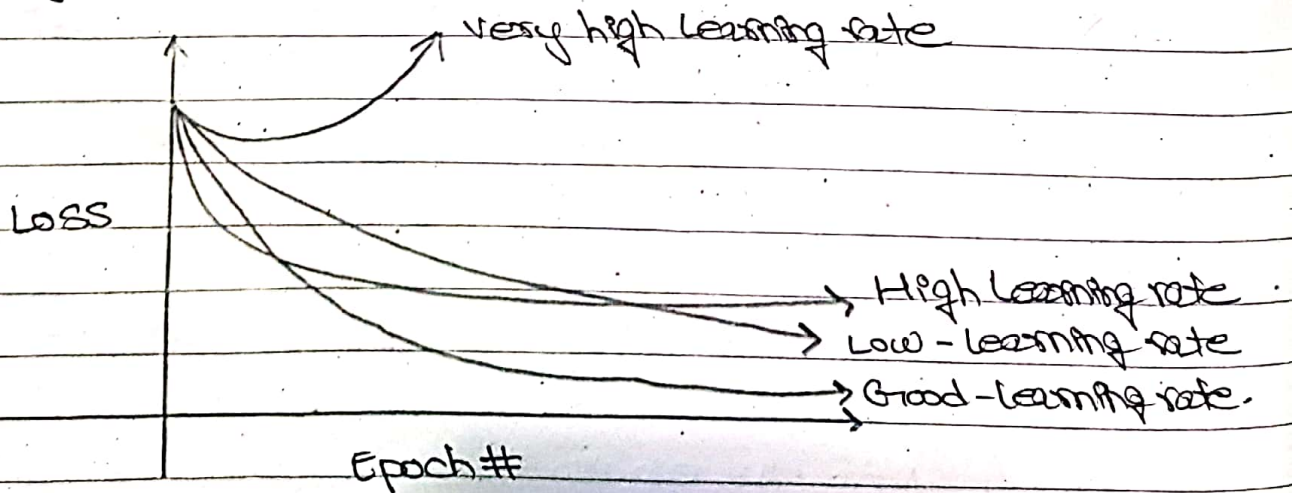
Page No.
Date : / /

- If f is function to be minimized (cost function), Gradient descent changes the parameters of learning model as :

$$w = w - \alpha \frac{df}{dw}$$

where, α is learning rate & w is parameter to be optimized.

- The higher the gradient, the steeper the slope and the faster a model can learn. But if the slope is zero, the model stops learning.
- How big the steps are that Gradient Descent takes into the direction of the local minimum are determined by learning rate.
- With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point.
- A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.



Page No.
Date : / /

Gradient Descent variations:

- Gradient descent is an optimization algorithm which is used to find model parameters (e.g. coefficients or weights) that minimize the error of the model on the training dataset.
- It uses the error on the predictions to update the model in such a way as to reduce the error.
- It does this by making changes to the model that moves it along a gradient or slope of errors down towards a minimum error value.
- The three main flavors of gradient descent are batch, stochastic, and mini-batch.

① Batch-Gradient Descent:

- Batch-Gradient descent, computes the gradient of the cost function w.r.t. the parameters w for the entire training dataset:

$$w = w - \frac{df(w, x, y)}{dw}$$

- As we need to calculate the gradients for the whole dataset, it perform just one update,
- Batch gradient descent is computationally fast. However it is intractable for datasets that don't fit in memory.
- It doesn't allow us to update our model online, i.e. with new examples on-the-fly.
- Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces & to a local minimum for non-convex surfaces.

Page No.
Date : / /

② Stochastic Gradient Descent (SGD):

- Stochastic gradient descent (SGD) in contrast performs a parameter update for each training example $x^{(i)}$ and label $y^{(i)}$. Therefore, learning happens on every example.

$$w = \frac{w - \alpha \, \partial f(w, x^{(i)}, y^{(i)})}{\partial w}$$

- The term stochastic indicates that one example comprising each batch is chosen at random
- It allow us to update our model online.
- It updates the model much frequently, which is more computationally expensive than other GD variations.
- Thus, it takes significantly longer to train models on large datasets. At the same time we loose speedup due to vectorization.
- These frequent updates can result in a noisy gradient signal, which may cause the model parameters jump around. (higher variance over training epochs)
- At the same time this behaviour helps to jump to another minima.
- This ultimately complicates convergence to the exact minimum, as SGD will keep overshooting.

Page No.
Date : / /

③ Mini-batch Gradient Descent:

- Mini-batch Gradient descent splits the training datasets into small batches that are used to calculate model error and update model coefficients.

$$w = w - \alpha \frac{\partial f\left(w, x^{(i:i+n)}, y^{(i:i+n)}\right)}{\partial w}$$

- Mini-batch gradient descent takes the best of both Batch Gradient and SGD.
  - i) Reduces the variance of the parameter updates, which can lead to more stable convergence.
  - ii) performs less frequent parameter updates & hence is not much time consuming.

- Mini-batch gradient descent is the recommended variant of gradient descent for most applications.
- Mini-batch requires the configuration of an additional batch size Hyperparameter for the learning algorithm
  - parameter = (आपने Learn किये)
  - Hyperparameter = (Need to Find experimentally)

- Small values results in faster learning process at the cost of noise in the training process.
  (Similar to batch gradient descent)
- Large values results in slow learning process with accurate estimates of the error signal.
  (Similar to stochastic Gradient descent)

Page No.
Date : / /

* Linear Regression:

- Regression analysis is the process of curve fitting in which the relationship between the independent variables and dependent variables are modeled in the m^{th} degree polynomial.
- Polynomial Regression models are usually fit with the method of least mean square (LMS) rule or Widrow–Hoff rule.
- If we assume that the relationship is a linear one and only one variable, then we can use linear equation as.

$$y = f(x) = w_0 + w_1 x$$

(y: dependent variable; x: independent variable; w_0, w_1: coefficients)

- Let us suppose that training set contains n data points. Error function or cost function for the n data points is given by:

Let, dataset $= (x^1, y^1), (x^2, y^2) \ldots (x^n, y^n)$

$$y = f(x) = w_0 + w_1 x$$

$$y' = w_0 + w_1 x^i \quad \text{(predicted output)}$$

$$e = y^i - y = y^i - w_0 - w_1 x^i$$

where, y^i = Actual data

y = Predicted data.

Page No.
Date : / /

Error Function,

$$E = \frac{1}{2n} \sum_{i=1}^{n} e_i^2$$

$$E = \frac{1}{2n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x^i)^2$$

— now, Coefficients can be determined or updated using gradient decent method as below.

① $w_0 = w_0 - \alpha \frac{\partial E}{\partial w_0} = w_0 - \alpha \cdot \frac{\partial}{\partial w_0} \left(\frac{1}{2n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x^i)^2 \right)$

$= w_0 + \alpha \frac{1}{n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x^i)$

② $w_1 = w_1 - \alpha \frac{\partial E}{\partial w_1} = w_1 + \alpha \frac{1}{n} \sum_{i=1}^{n} (y_i - w_0 - w_1 x^i) x^i$

* Example: Fit a straight line through following data using SGD. Show one epoch of training.

x	1	2	3	4
f(y)	3	5	7	9

Sol

General Form of linear regression equation : $y = w_0 + w_1 x$

Let us assume that initial values of parameters are:

$w_0 = w_1 = 0$

Iteration 1: $x = 1$, $y = f(x) = 3$, $\alpha = 0.01$

$w_0 = w_0 + \alpha (y - w_0 - w_1 x) = 0 + 0.01 (3 - 0 - 0.1) = 0.03$

$w_1 = w_1 + \alpha (y - w_0 - w_1 x) x_1 = 0 + 0.01 (3 - 0 - 0.1) = 0.03$

Page No.
Date: / /

Iteration 2: $x = 2,\ y = f(x) = 5,\ w_0 = 0.03$ & $w_1 = 0.03$

$w_0 = w_0 + \alpha(y - w_0 - w_1 x)$
$= 0.03 + 0.01((5 - 0.03 - (0.03 \times 2))$
$= 0.0791$

$w_1 = w_1 + \alpha(y - w_0 - w_1 x)x$
$= 0.03 + 0.01(5 - 0.03 - (0.03 \times 2)) \times 2$
$= 0.1252$

Iteration 3: $x = 3,\ y = f(x) = 7,\ w_0 = 0.0791$
$w_1 = 0.1252$

$w_0 = w_0 + \alpha(y - w_0 - w_1 x)$
$= 0.0791 + 0.01(7 - (0.0791 \times 3) - (0.1252 \times 3))$
$= 0.14446$

$w_1 = w_1 + \alpha(y - w_0 - w_1 x)x$
$= 0.1252 + 0.01(7 - 0.0791 - (0.1252 \times 3))3$
$= 0.324289$

Iteration 4: $x = 4,\ y = f(x) = 9,\ w_0 = 0.14446$
$w_1 = 0.324289$

$w_0 = w_0 + \alpha(y - w_0 - w_1 x)$
$= 0.14446 + 0.01(9 - 0.14446 - (0.324289 \times 4))$
$= 0.22$

$w_1 = w_1 + \alpha(y - w_0 - w_1 x)x$
$= 0.324289 + 0.01(9 - 0.14446 - (0.324289 \times 4)) \times 4$
$= 0.63$

Page No.
Date : / /

* Example:

Derive weight update rule for Linear Regression for two variables.

$$y = w_0 + w_1 x_1 + w_2 x_2$$

cost function for linear regression,

$$E = \frac{1}{2n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x_1 - w_2 x_2)^2$$

now, coefficient can be updated using G.D as

① $$w_0 = w_0 - \alpha \frac{\partial E}{\partial w_0} = w_0 - \alpha \frac{\partial}{w_0} \left(\frac{1}{2n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x_1 - w_2 x_2)^2 \right)$$

$$= w_0 + \alpha \left(\frac{1}{n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x_1 - w_2 x_2) \right)$$

② $$w_1 = w_1 - \alpha \frac{\partial E}{\partial w_1} = w_0 + \alpha \frac{1}{n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x_1 - w_2 x_2) x_1$$

③ $$w_2 = w_2 - \alpha \frac{\partial E}{\partial w_2} = w_0 + \alpha \frac{1}{n} \sum_{i=1}^{n} (y^i - w_0 - w_1 x_1 - w_2 x_2) x_2$$

* Example: Fit a straight line through the following data using SGD - show on epoch of training.

x_1	1	2	1	3
x_2	1	1	2	3
$y = f(x)$	3	5	2	5

Iteration 1: $x_1 = 1, x_2 = 1, y = f(x) = 3$

Let, $w_0 = w_1, w_2 = 0$, and $\alpha = 0.01$ then,

$w_0 = w_0 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2)$

$= 0 + 0.01(3 - 0 - 0.1 - 0.0)$

$= 0.03$,

$w_1 = w_1 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_1$

$= 0 + 0.01 (3 - 0 - 0*1 - 0*1).1$

$= 0.03$

$w_2 = w_2 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_2$

$= 0 + 0.01 (3 - 0 - 0*1 - 0*1) 1$

$= 0.03$

Iteration 2: $x_1 = 2, x_2 = 1, y = 5, w_0 = 0.03$

$w_1 = 0.03$

$w_2 = 0.03$

$w_0 = w_0 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2)$

$= 0.03 + 0.01 ((5 - 0.03 - (0.03*2) - (0.03*1))$

$= 0.0788$

$w_1 = w_1 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_1$

$= 0.03 + 0.01 ((5 - 0.03 - (0.03*2) - (0.03*1)).2$

$= 0.1276$

$w_2 = w_2 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_2$

$= 0.03 + 0.01 ((5 - 0.03 - (0.03 \times 2) - (0.03 \times 1)) \times 1$

$= 0.0786$

Iteration 3: $x_1 = 1, x_2 = 2, y = 2, w_0 = 0.0788$

$w_1 = 0.1276$

$w_2 = 0.0788$

$w_0 = w_0 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2)$

Page No.
Date : / /

$w_0 = 0.0788 + 0.01((2 - 0.0788 - (0.1276 * 1) - (0.0788 * 2)$

$= 0.09516$

$w_1 = w_1 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_1$

$= 0.1276 + 0.01((2 - 0.0788 - (0.1276 * 1) - (0.0788 * 2)) 1$

$= 0.14396$

$w_2 = w_2 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_2$

$= 0.0788 + 0.01(2 - 0.0788 - (0.1276 * 1) - (0.0788 * 2)) * 2$

$= 0.11152$

Iteration 4: $x_1 = 3$, $x_2 = 3$, $y = 5$, $w_0 = 0.09516$

$w_1 = 0.14396$

$w_2 = 0.11152$

$w_0 = w_0 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2)$

$= 0.09516 + 0.01 (5 - 0.09516 - (0.14396 * 3) - (0.11152 * 3)$

$= 0.136544$

$w_1 = w_1 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_1$

$= 0.14396 + 0.01 (5 - 0.09516 - (0.14396 * 3) - (0.11152 * 3)) * 3$

$= 0.268112$

$w_2 = w_2 + \alpha (y - w_0 - w_1 x_1 - w_2 x_2) x_2$

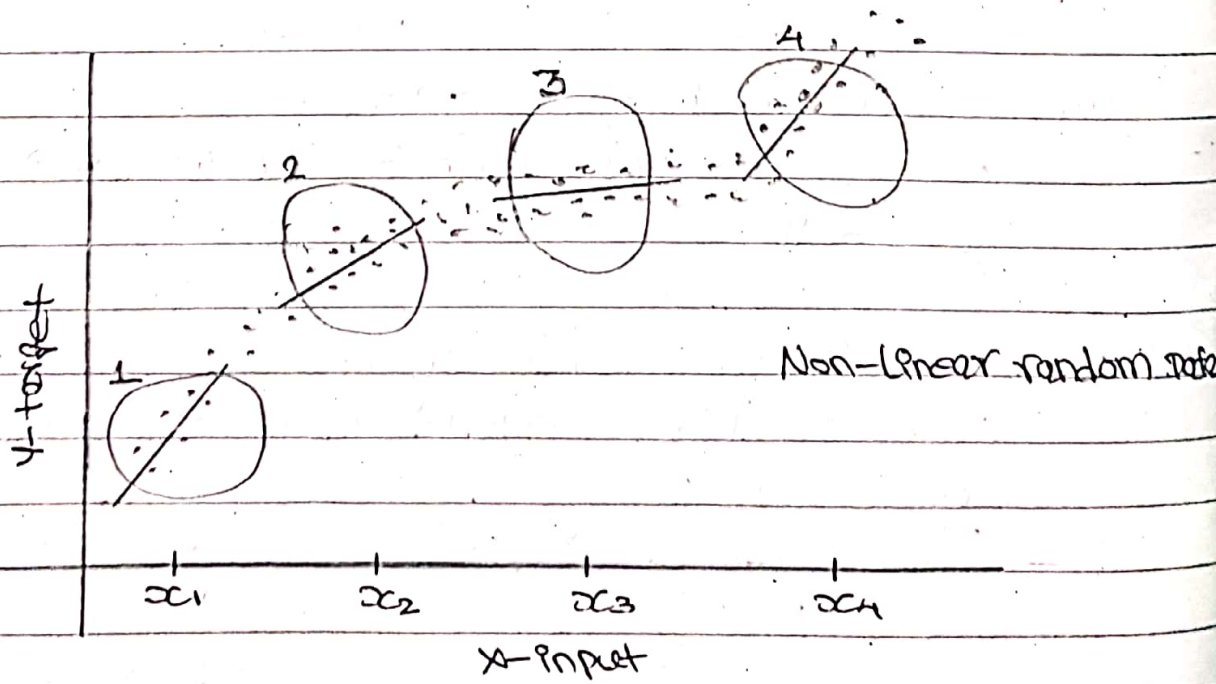
$= 0.11152 + 0.01 (5 - 0.09516 - (0.14396 * 3) - (0.11152 * 3)) * 3$

$= 0.235672$

Page No.
Date : / /

* Locally weighted Regression:

- Linear regression is a supervised learning algorithm used for computing linear relationships between input (x) and output (y).
- This algorithm cannot be used for making predictions when there exists a non-linear relationship between x and y. In such case, Locally weighted Linear regression is used.
- Locally weighted linear regression is a non-parametric algorithms, that is, the model doesnot learn a fixed set of parameters as it is done in ordinary linear regression.
- rather the parameters are computed individually for each query point x.
- While computing parameters a higher preference is given to the points in the training set lying in the neighborhood of x than the points lying far away from x. i.e.



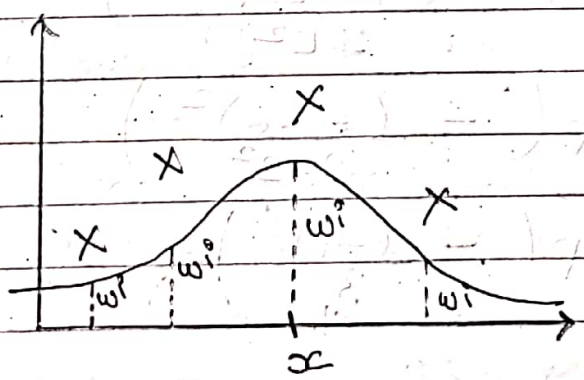
- This results in the model fitting a straight line only to the data which is near or close to the query point.
- The cost function for the locally weighted linear regression algorithm is given as below.

$$E = \frac{1}{2n} \sum_{i=1} w_i (y_i - a_0 - a_1 x_i)^2$$

- Here, w_i is a non-negative weight associated with training point x_i. (The value of w^i lies between 0-1).
- For x^i lying closer to the query point the value of w_i is large, while for x^i lying away to the query point value of w_i is small. A typical choice w_i of is:

$$w_i = \exp\left(\frac{-(x^{(i)} - x)^2}{2\tau^2}\right)$$

where, τ called the bandwidth parameter & controls the rate at which w_i falls with distance from x.



- clearly, if $\tau \cdot |x^i - x|$ is small w_i is close to 1 and if $|x^i - x|$ is large w_i is close to 0. Thus, the Training-set-points lying closer to the query point x contribute more to the cost function than the points lying far away from x.
- Once we have cost function, we can use gradient descent Algorithm to train the LWR algorithm.

Page No.
Date : / /

* Example: Consider a query point $x = 6$ and let $x^1 = 5$, $x^2 = 4$, and $x^3 = 3$ are three points in the training set. Find the cost function for the locally weighted linear regression. (Let $\tau = 2$)

= Cost function for the locally weighted linear regression.

$$E = \frac{1}{2n} \sum_{i=1}^{n} w^i \left(y^i - a_0 - a_1 x^{(i)} \right)^2$$

Initially, we need to evaluate the value of w^i for training point $x^1 = 5$, $x^2 = 4$ and $x^3 = 3$

$$w^{(i)} = \exp\left(- \frac{(x^{(i)} - x)^2}{2\tau^2} \right)$$

① For $x^1 = 5$, $x = 6$, Let $\tau = 2$

$$w^1 = \exp\left(- \frac{(x^{(i)} - x)^2}{2\tau^2} \right)$$

$$= \exp\left(- \frac{(5-6)^2}{2 \cdot 2^2} \right)$$

$$= \exp\left(- \frac{(-1)^2}{8} \right)$$

$$= 0.88 \text{ //}$$

② For $x^2 = 4$, $x = 6$, Let $\tau = 2$

$$w^2 = \exp\left(- \frac{(4-6)^2}{2 \cdot 2^2} \right)$$

$$= 0.60 \text{ //}$$

Page No.
Date : / /

③ For $x^3 = 3, x = 6$, let $\tau = 2$

$$w^3 = \exp\left(-\frac{(x^{(i)} - x)^2}{2 \cdot \tau^2}\right)$$

$$= \exp\left(-\frac{(3-6)^2}{2 \cdot 2^2}\right)$$

$$= 0.32$$

So, the cost function is

$$E = \frac{1}{2n} \sum_{i=1}^{n} w^i (y^i - a_0 - a_1 x^i)^2$$

$$= \frac{1}{2n} \{ 0.88 (y^1 - a_0 - a_1 x^1)^2 + 0.61 (y^2 - a_0 - a_1 x^2)^2 + 0.32 (y^3 - a_0 - a_1 x^3)^2 \}$$

Thus, the weights fall exponentially as the distance between x & $x^{(i)}$ increases

* Coefficient update of LWR:

- The coefficient update rule for the locally weighted linear regression is given below:

i.e.

$$a_0 = a_0 - \frac{\partial E}{\partial a_0} = a_0 + \alpha \frac{1}{n} \sum_{i=1}^{n} w^i (y^i - a_0 - a_1 x^i)$$

$$a_1 = a_1 - \frac{\partial E}{\partial a_1} = a_1 + \alpha \frac{1}{n} \sum_{i=1}^{n} w^i (y^i - a_0 - a_1 x^i) x^i$$

Page No.
Date : / /

* Example: Consider following dataset & show one epoch of Training using LWR.

x	3	4	5
y	10	15	24

= Iteration 1: $x^{(i)} = 3$, $w^{(i)} = 0.32$ (from previous example), $y = 10$, $a_0 = 0$, $a_1 = 0$ & $\alpha = 0.1$

then,

$a_0 = a_0 + \alpha w^i (y^i - a_0 - a_1 x^i)$

$= 0 + 0.1 * 0.32 (10 - 0 - 0 * 3)$

$= 0.32$

$a_1 = a_1 + \alpha w^i (y^i - a_0 - a_1 x^i) x^i$

$= 0 + 0.1 * 0.32 (10 - 0 - 0) * 3$

$= 0.96$

Iteration 2: $x^{(i)} = 4$, $w^{(i)} = 0.60$, $y = 15$, $a_0 = 0.32$, $a_1 = 0.96$, $\alpha = 0.1$

$a_0 = a_0 + \alpha w^i (y^i - a_0 - a_1 x^i)$

$= 0.32 + 0.1 * 0.60 (15 - 0.32 - 0.96 * 4)$

$= 0.97$

$a_1 = a_1 + \alpha w^i (y^i - a_0 - a_1 x^i) * x^i$

$= 0.96 + 0.1 * 0.60 (15 - 0.32 - 0.96 * 4) * 4$

$= 3.60$

Page No.
Date : / /

Iteration 3: $x^{(i)} = 5$, $w^{(i)} = 0.88$, $y = 24$, $a_0 = 0.97$

$a_1 = 3.60$

$\alpha = 0.1$

$$a_0 = a_0 + \alpha w^i (y^i - a_0 - a_1 x^i)$$
$$= 0.97 + 0.1 * 0.88 (24 - 0.97 - 3.60 * 5)$$
$$= 1.41$$

$$a_1 = a_1 + \alpha w^i (y^i - a_0 - a_1 x^i) x^i$$
$$= 3.60 + 0.1 * 0.88 (24 - 0.97 - 3.60 * 5) * 5$$
$$= 5.81$$

* Logistic Regression:

- Logistic regression is one of the most popular machine learning algorithms for binary classification. This is because it is a simple algorithm that performs very well on a wide range of problems.
- We want to predict a variable $\hat{y} \in \{0, 1\}$, where 0 is called negative class, while 1 is called positive class. Such task is known as binary classification.
- The heart of the logistic regression technique is logistic function and is defined as:

$$f(x) = \frac{1}{1+e^{-x}}$$

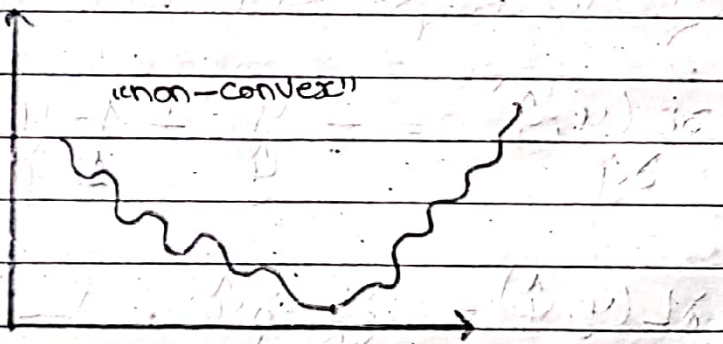
- Logistic function transforms the input into the range $[0, 1]$. Smallest negative numbers results in values close to zero and the larger positive numbers results in values close to one.
- If there are two input variables, logistic regression has two coefficients just like linear regression. i.e.

$$y = w_0 + w_1 x_1 + w_2 x_2$$

- Unlike linear regression, the output is transformed into a probability using the logistic function:

$$\hat{y} = \sigma(y) = \frac{1}{1+e^{y}}$$

- If the probability is >0.5 we can take the output as a prediction for the class 1, otherwise the prediction is for the class 0.
- The job of the learning algorithm will be to discover the best values for the coefficients (w_0, w_1, and w_2) based on the training data.
- Unfortunately, we can't use the cost function MSE in logistic regression. It's because our prediction function is non-linear (due to sigmoid transform).
- Squaring this prediction as we do in MSE results in a non-convex function with many local minimums. If our cost function has many local minimums, gradient descent may not find the optimal global minimum.



- Given the training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \ldots (x^{(m)}, y^{(m)})\}$ we want $\hat{y}^{(i)} \approx y^{(i)}$
- For logistic regression, we use the following loss function or error function

$$L(\hat{y}, y) = \begin{cases} -\log(\hat{y}) & \text{if } y=1 \\ -\log(1-\hat{y}) & \text{if } y=0 \end{cases}$$

- In case $y=1$, the output (the cost to pay) approaches to 0

as $\hat{y}$ approaches to 1. Conversely, the cost to pay grows to infinity as $\hat{y}$ approaches to 0.

- This is a desirable property : we want a bigger penalty as the algorithm predicts something far away from the actual value. The same intution applies when $y=0$.

- Thus, cost function for logistic regression is given as:

$$L(y,\hat{y}) = -y \log \hat{y} - (1-y) \log (1-\hat{y})$$

* Logistic Regression GD / Derivatives:

$$\frac{\partial L(y,\hat{y})}{\partial \hat{y}} = -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$\frac{\partial L(y,\hat{y})}{\partial y} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial y} = \left\{ -\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}} \right\} \{ \hat{y}(1-\hat{y}) \}$$

$$= \hat{y} - y$$

$\therefore$ if $g(x)$ is logistic function $g'(x) = g(x)(1-g(x))$

$$\frac{\partial L}{\partial w_0} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial y} \frac{\partial y}{\partial w_0} = (\hat{y} - y_0)$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial y} \frac{\partial y}{\partial w_1} = x_1 (\hat{y} - y)$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial y} \frac{dy}{dw_2} = x_2(\hat{y} - y)$$

Thus, parameters of logistic regression are updated as below:

$$w_0 = w_0 - \alpha \frac{\partial L}{\partial w_0} = w_0 - \alpha \cdot (\hat{y} - y)$$

$$w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1} = w_1 - \alpha (\hat{y} - y) x_1$$

$$w_2 = w_2 - \alpha \frac{\partial L}{\partial w_2} = w_2 - \alpha (\hat{y} - y) x_2$$

* Example: Fit the logistic regression model through the following data. Show one epoch of training.

x_1	x_2	class(y)
0.78	0.69	1
0.67	1.00	1
0.00	0.00	-1
0.22	0.14	-1

= Sol

General form of logistic regression equation is

$$y = w_0 + w_1 x_1 + w_2 x_2$$

Let us assume that initial values of parameters are:

$$w_0 = w_1 = w_2 = 0$$

Iteration 1: $x_1 = 0.78$, $x_2 = 0.69$, $y = 1$, $\alpha = 0.1$

$$y = w_0 + w_1 x_1 + w_2 x_2 = 0 + 0 * 0.78 + 0 * 0.69 = 0$$

So,

$$\hat{y} = \frac{1}{1 + e^{-y}} = \frac{1}{1 + e^{-0}} = 0.5$$

Page No.
Date : / /

$w_0 = w_0 - \alpha(\hat{y} - y) = 0 - 0.1(0.5 - 1) = 0.05$
$w_1 = w_1 - \alpha(\hat{y} - y)x_1 = 0 - 0.1(0.5 - 1) * 0.78 = 0.039$
$w_2 = w_2 - \alpha(\hat{y} - y)x_2 = 0 - 0.1(0.5 - 1) * 0.69 = 0.0345$

Iteration 2:

$x_1 = 0.67, x_2 = 1, y = 1, \alpha = 0.1, w_0 = 0.05$
$w_1 = 0.039$
$w_2 = 0.0345$

$y = w_0 + w_1x_1 + w_2x_2$
$= 0.05 + (0.039 * 0.67) + (0.0345 * 1)$
$= 0.11063$

$\hat{y} = \frac{1}{1+e^{y}} = 0.5276$

$w_0 = w_0 - \alpha(\hat{y} - y) = 0.05 - 0.1(0.5276 - 1) = 0.097$
$w_1 = w_1 - \alpha(\hat{y} - y)x_1 = 0.039 - 0.1(0.5276 - 1)0.67$
$= 0.0706$

$w_2 = w_2 - \alpha(\hat{y} - y)x_2 = 0.0345 - 0.1(0.5276 - 1) * 1$
$= 0.08174$

Iteration 3: $x_1 = 0, x_2 = 0, y = 0, \alpha = 0.1, w_0 = 0.097$
$w_1 = 0.0706$
$w_2 = 0.08174$

$y = w_0 + w_1x_1 + w_2x_2$
$= 0.097 + (0.0706 * 0) + (0.08174 * 0)$
$= 0.097$

Page No.
Date : / /

$$\hat{y} = \frac{1}{1+e^{-y}} = 0.5243$$

$w_0 = w_0 - \alpha(\hat{y}-y) = 0.097 - 0.1(0.5243-0) = 0.0445$
$w_1 = w_1 - \alpha(\hat{y}-y)x_1 = 0.0706 - 0.1(0.5243-0)0 = 0.0706$
$w_2 = w_2 - \alpha(\hat{y}-y)x_2 = 0.08174 - 0.1(0.5243-0)\times 0 = 0.08174$

Iteration 4:

$x_1 = 0.22, x_2 = 0.14, y = 0, \alpha = 0.1, w_0 = 0.0445$
$w_1 = 0.0706$
$w_2 = 0.08174$

$y = w_0 + w_1x_1 + w_2x_2$
$= 0.0445 + (0.0706 * 0.22) + (0.08174 * 0.14)$
$= 0.0715$

$$\hat{y} = \frac{1}{1+e^{-y}} = 0.5178$$

$w_0 = w_0 - \alpha(\hat{y}-y) = 0.0445 - 0.1(0.5178-0) = -0.0072$
$w_1 = w_1 - \alpha(\hat{y}-y)x_1 = 0.0706 - 0.1(0.5178-0)0.22 = 0.0592$
$w_2 = w_2 - \alpha(\hat{y}-y)x_2 = 0.08174 - 0.1(0.5178-0)\times 0.14 = 0.0744$

Page No.
Date : / /

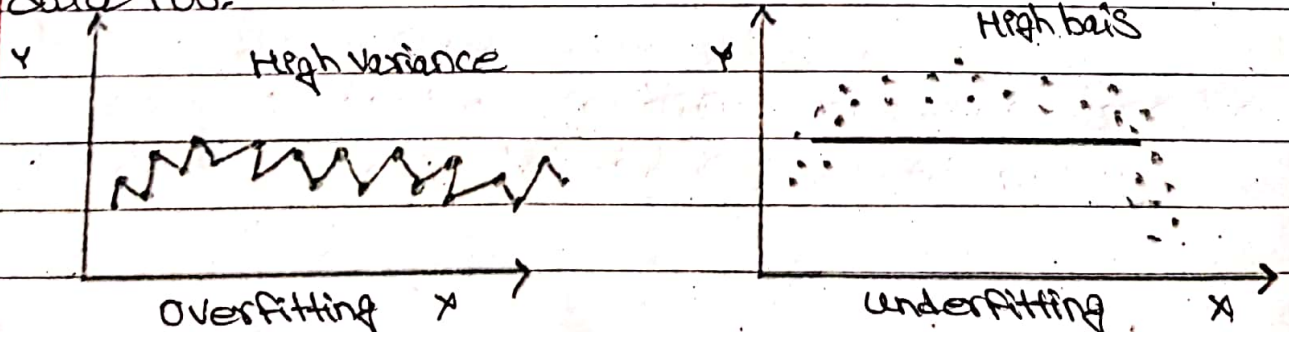
* Overfitting and Underfitting:

① Overfitting:

- Overfitting is the result of using an excessively complicated model.
- It happens when the model learns the details and noise of training data to the extent that it negatively impacts the performance of the model on new data.
- This means that the model learns noise or random fluctuations in the training data as concepts.
- ~~The~~ This negatively impacts the model's ability to generalize because these concepts do not apply to new data.

② Underfitting:

- Underfitting is the result of using an excessively simple model or using very few training samples.
- In such situations, a machine learning algorithm cannot capture the underlying trend of the data.
- Thus, it refers to a model that can neither model the training data nor generalize to new data.
- Obviously, an underfitted machine learning model is not a suitable model for making predictions because it has poor performance on the training data too.



Page No.
Date : / /

* Performance Measures of classification:

- Before selecting a classification algorithm to solve a problem, we need to evaluating its performance. Confusion matrix is widely used for computing performance measures of classification algorithm.

* Confusion Matrix:

- A Confusion matrix is an NXN matrix used for evaluating the performance of a classification model, where N is the number of target classes.
- The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.
- For a binary classification problem, we would have a 2X2 matrix as shown below with 4 values:

		Actual classes	
		Positive	Negative
predicted classes	positive	TP	FP
	Negative	FN	TN

① True Positive (TP):

It represents correctly classified positive classes. Both actual and predicted class are positive here.

② False Positive (FP):

It represents incorrectly classified positive classes. These are the positive classes predicted by the model that were actually negative. This is called Type I error.

③ True Negative (TN):

It represents correctly classified Negative classes. Both actual & predicted class are negative here.

④ False Negative (FN):

It represents incorrectly classified negative classes. These are the negative classes predicted by the model that were actually positive. This is called Type II error.

* Performance measures of classifications:

Four widely used performance measures used for evaluating classification models are:

① Accuracy,
② Recall,
③ Precision,
④ F1-Score.

① Accuracy:

It is the percentage of correct predictions made by the model and is given as below:

Page No.
Date : / /

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{\text{Total no. of Instances}}$$

② **Precision:**

It is percentage of predicted positives that are actually positive and is given by:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{Total Number of predicted positive}}$$

③ **Recall:**

It is the percentage of actual positives that are correctly classified by the model and is given as below:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{Total number of actual positives.}}$$

④ **F1-Score:**

It is the harmonic mean of recall and precision. It becomes high only when both precision and recall are high. This score is given by:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

* Example: Suppose that we have to classify 100 people as pregnant or not pregant. This includes 40 pregnant women & the remaining 60 are not pregnant. out of 40 pregnant women 30 pregnant women are classified correctly and the remaining 10 percent women are classified as not pregnant by the machine learning Algorithm. on the other hand, out of 60 people in the not pregnant category, 55 are classified as not pregnant & the remaining 5 are classified pregnant. Compute accuracy, precision, recall and F1-Score for above example.

= Sol

Let, the Pregnant women belongs to Positive class and non-pregnant women belongs to Negative class then,

True positive = 30
False positive = 5
True Negative = 55
False Negative = 10.

		Actual class	
		pregnant	Non-pregnant
predicted class	pregnant	30 (T.P)	5 (F.P)
	Non-pregnant	10 (F.N)	55 (T.N)

Page No.
Date : / /

① Accuracy = $\frac{TP + TN}{TP+TN+FP+FN} = \frac{30+55}{100} = 0.85$

② Precision = $\frac{TP}{TP+FP} = \frac{30}{30+5} = 0.857$

③ Recall = $\frac{TP}{TP+FN} = \frac{30}{30+10} = 0.75$

④ F1-Score = $\frac{2 \times Recall \times Precision}{Recall + precision}$

$= \frac{2 \times 0.75 \times 0.857}{0.75 + 0.857}$

$= 0.799$

Page No.
Date : / /

* Matrix Derivatives:

- Let A be a $m \times n$ matrix and $f : R^{m \times n} \to R$ be a mapping function from m-by-n matrices to a real number. i.e.

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & & & \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix}$$

- Now, Gradient of f with respect to A is also a $m \times n$ matrix and is given as below:

$$\nabla_A f_A = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} & \cdots & \frac{\partial f}{\partial A_{1m}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} & \cdots & \frac{\partial f}{\partial A_{2m}} \\ \vdots & & & \\ \frac{\partial f}{\partial A_{n1}} & \frac{\partial f}{\partial A_{n2}} & \cdots & \frac{\partial f}{\partial A_{nm}} \end{bmatrix}$$

Page No.
Date : / /

* Example: Consider the following 2×2 matrix,

$$\begin{bmatrix} w & x \\ y & z \end{bmatrix}$$

- Let $f(w,x,y,z) = 2w + 3x + yz$

- $\nabla_A f_A = \begin{bmatrix} 2 & 3 \\ z & y \end{bmatrix}$,,

* Trace of Matrix:

- For a $n \times n$ matrix A, trace of A is defined as sum of diagonal elements

  i.e $tr.A = \sum_{i=1}^{n} a_{ii}$

- For a real number a $tr.a = a$.
- If multiplication of A & B is square matrix then

  $tr\,AB = tr\,BA$

Verify that $tr.AB = tr.BA$

Let, A & B are matrices of order 2×3 & 3×2 respectively,

i.e, $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$

$B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$

Page No.
Date : / /

$$AB = \begin{bmatrix} a_{11}\cdot b_{11} + a_{12}\cdot b_{21} + a_{13}\cdot b_{31} & a_{11}\cdot b_{12} + a_{12}\cdot b_{22} + a_{13}\cdot b_{32} \\ a_{21}\cdot b_{11} + a_{22}\cdot b_{21} + a_{23}\cdot b_{31} & a_{21}\cdot b_{12} + a_{22}\cdot b_{22} + a_{23}\cdot b_{32} \end{bmatrix}$$

$$\text{tr.}AB = a_{11}\cdot b_{11} + a_{12}\cdot b_{21} + a_{13}\cdot b_{31} + a_{21}\cdot b_{12} + a_{22}\cdot b_{22} + a_{23}\cdot b_{32}$$

Similarly

$$BA = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

$$= \begin{bmatrix} b_{11}\cdot a_{11} + b_{12}\cdot a_{21} & b_{11}\cdot a_{12} + b_{12}\cdot a_{22} & b_{11}\cdot a_{13} + b_{12}\cdot a_{23} \\ b_{21}\cdot a_{11} + b_{22}\cdot a_{21} & b_{21}\cdot a_{12} + b_{22}\cdot a_{22} & b_{21}\cdot a_{13} + b_{22}\cdot a_{23} \\ b_{31}\cdot a_{11} + b_{32}\cdot a_{21} & b_{31}\cdot a_{12} + b_{32}\cdot a_{22} & b_{31}\cdot a_{13} + b_{32}\cdot a_{23} \end{bmatrix}$$

$$\text{tr.}BA = \begin{bmatrix} b_{11}\cdot a_{11} + b_{12}\cdot a_{21} + b_{21}\cdot a_{12} + b_{22}\cdot a_{22} + b_{31}\cdot a_{13} + b_{32}\cdot a_{23} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}\cdot b_{11} + a_{12}\cdot b_{21} + a_{13}\cdot b_{31} + a_{21}\cdot b_{12} + a_{22}\cdot b_{22} + a_{23}\cdot b_{32} \end{bmatrix}$$

$\therefore$ tr.AB = tr.BA.

Page No.
Date : / /

* Corollaries of tr. AB = tr. BA

tr. ABC = tr. CAB = tr. BCA

tr. ABCD = tr. DABC = tr. BCDA

* Prove that tr. ABC = tr. CAB = tr. BCA

i.e.

$$\text{tr. } ABC = \text{tr. } \overset{A}{(}AB\overset{B}{)}C$$
$$= \text{tr. } CAB$$

Similarly, $\text{tr. } ABC = \text{tr. } \overset{A}{A}\ \overset{B}{(BC)}$
$$= \text{tr. } BCA$$

Thus, tr. ABC = tr. CAB = tr. BCA proved.

* Prove that tr. ABCD = tr. DABC = tr. BCDA

i.e.

$$\text{tr. } ABCD = \text{tr. } \overset{A}{(}ABC\overset{B}{)}D$$
$$= \text{tr. } DABC$$

$$\text{tr. } ABCD = \text{tr. } \overset{A}{A}\ (BCD\overset{B}{)}$$
$$= \text{tr. } BCDA$$

Thus, tr. ABCD = tr. DABC = tr. BCDA proved.

Page No.
Date : / /

* If A and B are square matrices & a is a real number then trace operator exhibits following properties.

i.e, ① $tr.A = tr\ A^T$

② $tr(A+B) = tr\,A + tr\,B$

③ $tr\ a.A = a\ tr\,A$

① $tr.A = tr.A^T$

= Sol

Let, $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$

$$tr.A = \sum_{i=1}^{n} a_{ii}$$

Again,

$$A^T = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \cdots & \vdots \\ a_{1n} & a_{2n} & & a_{nn} \end{bmatrix}$$

$$tr.A^T = \sum_{i=1}^{n} a_{ii}$$

Thus, $tr.A = tr.A^T$ i.e. $\sum_{i=1}^{n} a_{ii}$

Page No.
Date : / /

② $tr(A+B) = tr.A + tr.B$

= Let,

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & & & \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}$$

now,

$$A+B = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \vdots & & & \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \dots & a_{nn}+b_{nn} \end{bmatrix}$$

$$tr(A+B) = [a_{11}+b_{11} + a_{22}+b_{22} + \dots + a_{nn}+b_{nn}]$$

$$tr.(A+B) = \sum_{i=1}^{n} a_{ii}+b_{ii}$$

Then,

$$tr.A = a_{11} + a_{22} + \dots + a_{nn} = \sum_{i=1}^{n} a_{ii}$$

$$tr.B = b_{11} + b_{22} + \dots + b_{nn} = \sum_{i=1}^{n} b_{ii}$$

$$tr.A + tr.B = \sum_{i=1}^{n} a_{ii} + \sum_{i=1}^{n} b_{ii}$$

$$= \sum_{i=1}^{n} a_{ii}+b_{ii}$$

Thus, $\boxed{tr.(A+B) = tr.A + tr.B}$

Page No.
Date : / /

③ $tr.\, aA = a\, tr.\, A$

= Let,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

$$a \cdot A = \begin{bmatrix} a \cdot a_{11} & a \cdot a_{12} \\ a \cdot a_{21} & a \cdot a_{22} \end{bmatrix}$$

$$\boxed{tr.\, aA = a \cdot a_{11} + a \cdot a_{22}}$$

now,

$$\boxed{tr.\, A = a_{11} + a_{22}}$$

Thus,

$$\boxed{a\ tr.\, A = a \cdot a_{11} + a \cdot a_{22}}$$

* Some facts about matrix derives:

① $\nabla_A\, tr\, AB = B^T$

② $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

③ $\nabla_A\, tr\, ABA^TC = CAB + C^TAB^T$

④ $\nabla_A |A| = |A| (A^{-1})^T$

Page No.
Date : / /

① Prove that $\nabla_A \text{tr.} AB = B^T$

= Let order of A is 3×2 & order of B is 2×3.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

$$AB = \begin{bmatrix} a_{11}\cdot b_{11} + a_{12}\cdot b_{21} & a_{11}\cdot b_{12} + a_{12}\cdot b_{22} & a_{11}\cdot b_{13} + a_{12}\cdot b_{23} \\ a_{21}\cdot b_{11} + a_{22}\cdot b_{21} & a_{21}\cdot b_{12} + a_{22}\cdot b_{22} & a_{21}\cdot b_{13} + a_{22}\cdot b_{23} \\ a_{31}\cdot b_{11} + a_{32}\cdot b_{21} & a_{31}\cdot b_{12} + a_{32}\cdot b_{22} & a_{31}\cdot b_{13} + a_{32}\cdot b_{23} \end{bmatrix}$$

$$\text{tr.} AB = a_{11}\cdot b_{11} + a_{12}\cdot b_{21} + a_{21}\cdot b_{12} + a_{22}\cdot b_{22} + a_{31}\cdot b_{13} + a_{32}\cdot b_{23}$$

now,

$$\nabla_A \text{tr.} AB = \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \\ b_{13} & b_{23} \end{bmatrix}$$

$$B^T = \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \\ b_{13} & b_{23} \end{bmatrix}$$

Thus, $\boxed{\nabla_A \text{tr.} AB = B^T}$ proved.

Page No.
Date : / /

* Deriving Normal Equation For Linear Regression:

= Given, a training set, let x be the mxn matrix that contains input values of training examples in its rows and let $\vec{y}$ be a m dimensional vector containing target values.

$$X = \begin{bmatrix} \text{---} (x^1)^T \text{---} \\ \text{---} (x^2)^T \text{---} \\ \vdots \\ \text{---} (x^m)^T \text{---} \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

We know that,

$y = F(x) = (x^i)^T w$, where w is coefficient vector.

now,

$$\text{error} = F(x) - \vec{y}$$
$$= xw - \vec{y}$$
$$= \begin{bmatrix} (x^1)^T w \\ (x^2)^T w \\ \vdots \\ (x^m)^T w \end{bmatrix} - \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

Page No.
Date : / /

now, Loss function can be written as :

$$L = \frac{1}{2} \sum_{i=1}^{m} (f(x^i) - y^i)^2$$

$$= \frac{1}{2} (xw - \vec{y})^T (xw - \vec{y})$$

to minimize loss, we need to calculate derivatives/ gradient of L w.r.t. w.

$$\nabla_w L = \frac{1}{2} \nabla_w \underbrace{(xw}_{A\,B} - \vec{y})^T (xw - \vec{y})$$

$$= \frac{1}{2} \nabla_w (w^T x^T - \vec{y}^T)(xw - \vec{y}) \quad [\because (AB)^T = B^T A^T]$$

$$= \frac{1}{2} \nabla_w (w^T x^T xw - w^T x^T \vec{y} - \vec{y}^T xw + \vec{y}^T \vec{y})$$

$$= \frac{1}{2} \nabla_w \, tr(w^T x^T xw - w^T x^T \vec{y} - \vec{y}^T xw + \vec{y}^T \vec{y}) \quad [\because tr.a = a]$$

$$= \frac{1}{2} \nabla_w (tr\, w^T x^T xw - 2\, tr\, \vec{y}^T xw) \quad [\because \nabla_w \vec{y}^T \vec{y} = 0$$

$$tr\, A = tr\, A^T$$

$$tr(w^T x^T \vec{y}) = tr(w^T x^T \vec{y})^T$$

$$= tr(\vec{y}^T xw)]$$

$$= \frac{1}{2} (x^T xw + x^T xw - 2 x^T \vec{y}) \quad [\because \nabla_A tr\, ABA^T C$$

$$= B^T A^T C^T + BA^T C]$$

$$= x^T xw - x^T \vec{y}$$

At minima, gradient / derivatives is zero :

thus, $x^T xw - x^T \vec{y} = 0$

$$\Rightarrow w = \frac{x^T \vec{y}}{x^T x}$$

$$= (x^T x)^{-1} \cdot x^T \vec{y}$$

⇒ This is called normal eqn for linear regression.

Page No.
Date : / /

Note: We can use above equation to find values of coefficients.

* Example:

x_1	1	2	1	3
x_2	1	1	2	3
y	3	5	2	5

Use normal equation to find coefficient of linear regression equation fitted through above example.

= Sol

$$x = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 1 & 2 \\ 3 & 3 \end{bmatrix} \quad \vec{y} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 5 \end{bmatrix}$$

Augment x for w_0 i.e, $y = w_0 + w_1x_1 + w_2x_2$

Padding for w_0 value.

$$x = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \\ 1 & 3 & 3 \end{bmatrix}$$

$$\vec{y} = \begin{bmatrix} 3 \\ 5 \\ 2 \\ 5 \end{bmatrix}$$

Page No.
Date : / /

we have, normal equation i.e.

$$w = (x^T x)^{-1} x^T \vec{y}$$

$$= \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 3 \\ 1 & 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \\ 1 & 3 & 3 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 3 \\ 1 & 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 5 \\ 2 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} 1+1+1+1 & 1+2+1+3 & 1+1+2+3 \\ 1+2+1+3 & 1+4+1+9 & 1+2+2+9 \\ 1+1+2+3 & 1+2+2+9 & 1+1+4+9 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 3 \\ 1 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \\ 2 \\ 5 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 7 & 7 \\ 7 & 15 & 14 \\ 7 & 14 & 15 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 3+5+2+5 \\ 3+10+2+15 \\ 3+5+4+15 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 7 & 7 \\ 7 & 15 & 14 \\ 7 & 14 & 15 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 15 \\ 30 \\ 27 \end{bmatrix}$$

now,

$$\begin{bmatrix} 4 & 7 & 7 \\ 7 & 15 & 14 \\ 7 & 14 & 15 \end{bmatrix}^{-1} = \frac{1}{\det(A)} \text{Adj.}(A)$$

$$\det(A) = 4 \begin{bmatrix} 15 & 14 \\ 14 & 15 \end{bmatrix} - 7 \begin{bmatrix} 7 & 14 \\ 7 & 15 \end{bmatrix} + 7 \begin{bmatrix} 7 & 15 \\ 7 & 14 \end{bmatrix}$$

Page No.
Date : / /

$= 4(225-196) - 7(105-98) + 7(98-105)$

$= 116 - 49 - 49$

$= 18 \neq 0$ then, matrix has inverse.

now,

Adj A =

$$\begin{bmatrix} 15 & 14 \\ 14 & 15 \end{bmatrix} \quad \begin{bmatrix} 7 & 14 \\ 7 & 15 \end{bmatrix} \quad \begin{bmatrix} 7 & 15 \\ 7 & 14 \end{bmatrix}$$

$+ = +29 \qquad - = -7 \qquad + = -7$

$$\begin{bmatrix} 7 & 7 \\ 14 & 15 \end{bmatrix} \quad \begin{bmatrix} 4 & 7 \\ 7 & 15 \end{bmatrix} \quad \begin{bmatrix} 4 & 7 \\ 7 & 14 \end{bmatrix}$$

$- = -7 \qquad + = 11 \qquad - = -7$

$$\begin{bmatrix} 7 & 7 \\ 15 & 14 \end{bmatrix} \quad \begin{bmatrix} 4 & 7 \\ 7 & 14 \end{bmatrix} \quad \begin{bmatrix} 4 & 7 \\ 7 & 15 \end{bmatrix}$$

$+ = -7 \qquad - = -7 \qquad + = 11$

$$\text{Adj}A = \begin{bmatrix} 29 & -7 & -7 \\ -7 & 11 & -7 \\ -7 & -7 & 11 \end{bmatrix}^T = \begin{bmatrix} 29 & -7 & -7 \\ -7 & 11 & -7 \\ -7 & -7 & 11 \end{bmatrix}$$

now,

$$A^{-1} = \frac{1}{|A|} \text{Adj. } A$$

$$= \frac{1}{18} \begin{bmatrix} 29 & -7 & -7 \\ -7 & 11 & -7 \\ -7 & -7 & 11 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{29}{18} & \frac{-7}{18} & \frac{-7}{18} \\ \frac{-7}{18} & \frac{11}{18} & \frac{-7}{18} \\ \frac{-7}{18} & \frac{-7}{18} & \frac{11}{18} \end{bmatrix}$$

$$= \begin{bmatrix} 1.61 & -0.38 & -0.38 \\ -0.38 & 0.61 & -0.38 \\ -0.38 & -0.38 & 0.61 \end{bmatrix}_{3\times3} \cdot \begin{bmatrix} 15 \\ 30 \\ 27 \end{bmatrix}_{3\times1}$$

$$w = \begin{bmatrix} 2.49 \\ 2.34 \\ -0.63 \end{bmatrix}$$

Page No.
Date : / /

✳ Parametric vs Non-Parametric Algorithms:

① Parametric Algorithms:

- General form of learning model is assumed or General form of mapping function is assumed.
- Number of parameters are fixed, We just need to adjust values of parameters.
- number of parameters is independent of size of training data.
- Fast to train and predict Compared to non-parametric
- Can perform satisfactorily with small data size,
- example:- Linear regression, locally weighted logistic regression, i.e $y = w_0 + w_1 x_1 + w_2 x_2$

② Non-parametric Algorithm:

- No assumption about mapping function
- parameters are not only adjustable rather size of the parameters may change.
- Slow Compared to parametric algorithms.
- Need a large volume of training data to learn.
- example:- KNN (K-nearest neighbor) algorithm or classifier.

U.V.I

* Probabilistic Implementation of Least-square Regression

In case of least square regression, input and output variables are related via following equations.

$$y^i = w^T x^i + \varepsilon^i$$

where, ε^i is the error term that captures unmodeled effect & random noise.

Let us assume that ε^i follows IID (Independently and Identically distributed), Thus, according to Gaussian distributed distribution with zero mean & σ^2 standard deviation, probability distribution function for ε^i can be written as below:

$$p(\varepsilon^i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(\varepsilon^i)^2}{2\sigma^2}\right)$$

$$\Rightarrow p(y^i / x^i, w) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - w^T x^i)^2}{2\sigma^2}\right)$$

If we view above function as function of w, it is called likelyhood function and is given by:

$$L(w) = p(\vec{y} / x, w)$$

$$= \prod_{i=1}^{m} p(y_i / x^i, w)$$

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^i - w^T x^i)^2}{2\sigma^2}\right)$$

Page No.
Date : / /

Taking log on both sides,

$$\log(L(w)) = \log\left(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^i - w^T x^i)^2}{2\sigma^2}\right)\right)$$

$$\Rightarrow l(w) = \sum_{i=1}^{m} \log\left(\underbrace{\frac{1}{\sqrt{2\pi}\sigma}}_{a} \underbrace{\exp\left(\frac{-(y^i - w^T x^i)^2}{2\sigma^2}\right)}_{b}\right)$$

$$\left[\because l(w) = \log(L(w))\right]$$

$$= m \cdot \log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^i - w^T x^i)^2$$

$$\boxed{= m \cdot \log\frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \left(\frac{1}{2} \sum_{i=1}^{m} (y^i - w^T x^i)^2\right)}$$

Thus, maximizing above equation is equivalent to minimizing the term $\boxed{\frac{1}{2} \sum_{i=1}^{m} (y^i - w^T x^i)^2}$

i.e, we can say that maximizing likelihood is equivalent to minimizing squared sum of error (i.e. least square method).

Page No.
Date : / /

* Perceptron learning Algorithm:

- Perceptron is a single neuron that is used for binary classification.
- It can only classify linearly separable patterns.
- Perceptron rule can be ~~defined~~ derived from Logistic regression as below.
- In Logistic regression $\hat{y}$ is predicted as below:

$$\hat{y} = g(y) = \frac{1}{1+e^{-y}}$$

where, $y = w_0 + w_1 x^1 + w_2 x^2 \ldots + w_n x^n$

- If we use following definition of g then, it will be perceptron learning rule,

$$g(y) = \begin{cases} 1 & y \geq 0 \\ 0 & y < 0 \end{cases}$$

- Thus, weight update rule for Logistic regression is same as weight update rule for perceptron, which is given as below:

$$w_0 = w_0 + \alpha \frac{\partial L}{\partial w_0}$$

$$w_1 = w_1 + \alpha \frac{\partial L}{\partial w_1}$$

$$w_2 = w_2 + \alpha \frac{\partial L}{\partial w_2}$$

Page No.
Date : / /

where,

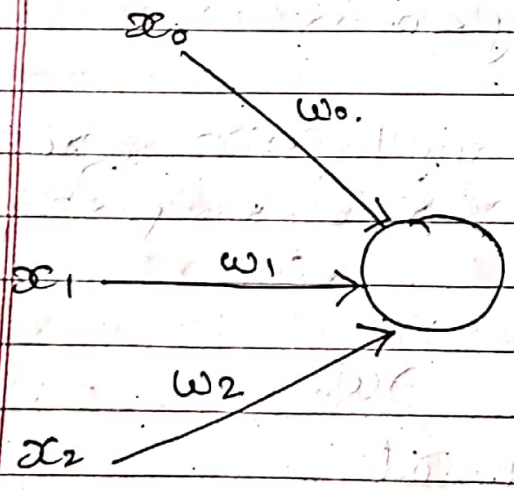
$$\frac{\partial L}{\partial w_0} = (\hat{y} - y)\, x^0 \quad (\text{where, } x^0 \text{ is always } 1)$$

$$\frac{\partial L}{\partial w_1} = (\hat{y} - y)\, x^1$$

$$\frac{\partial L}{\partial w_2} = (\hat{y} - y)\, x^2$$

Thus, General weight update rule for perceptron learning is

$$\boxed{w_i = w_i + \alpha (\hat{y} - y)\, x^{(i)}}$$



$$v = w_0 x_0 + w_1 x_1 + w_2 x_2$$

Page No.
Date : / /

* Gaussian Discriminant Analysis (GDA):

There are two broad categories of Supervised classification algorithms:

i) Discriminative (Discriminant) Analysis Algorithm
ii) Generative Learning Algorithms

i) Discriminant Analysis Algorithm:

- It finds decission boundary between different classes.
- For e.g:- Logistic regression, perceptron learning, etc.
- For the test data that lies in one side of the boundary belongs to class 1 and the data point that lies in another side of decision boundary belongs to class 2.

ii) Generative Learning Algorithm:

- It finds the distribution of each class seperately
- For e.g:- Naive Bayes classifier
- At the time of prediction, test data points is compared with each data distribution & belongs to the class with which it resembles the most.

* Naive Baye's classifier:

- It is based on Baye's therom which is given as below:

$$P(H/X) = \frac{P(X/H) \cdot P(H)}{P(X)}$$

$P(H/X)$: Posterior probability
$P(X/H)$: Likelyhood
$P(H)$, $P(X)$: Prior probability

Page No.
Date : / /

- Bayes's theorem provides the way of calculating posterior probability from prior probabilities & likelyhood.
- Let, D be the dataset and $C_1, C_2, \ldots, C_m$ are m classes.

now, Bayes's rule can be written as below:

$$P(C_i|X) = \frac{P(X|C_i) \cdot P(C_i)}{P(X)}$$

where, X is the input tuple containing n attributes i.e $X = \{x_1, x_2, \ldots, x_n\}$

The classifier predicts that tuple X belongs to class C_i having posterior probability conditioned on X.

Since, $P(X)$ is same for all classes, we need to maximize $P(X|C_i)\, P(C_i)$

Let X contains set of K attributes (i.e $X = \{x_1, x_2, x_3, \ldots, x_K\}$) which are independent of each other.

now, the probability $P(X|C_i)$ can be written as below.

$$P(X|C_i) = P(x_1|C_i) \times P(x_2|C_i) \ldots \times P(x_K|C_i)$$

$$= \prod_{J=1}^{K} P(x_J|C_i)$$

$$\Rightarrow \boxed{P(C_i|X) = \frac{P(C_i) \times \prod_{J=1}^{K} P(x_J|C_i)}{P(X)}}$$

Page No.
Date : / /

Finally, $P(X)$ is same for all classes so while making prediction, X is predicted to be in class C_i for which $P(C_i|X)$ is maximum.

* Example:

Age	Income	Student	Credit_Rating	(class) Buys_computer
youth	High	No	Fair	No
youth	High	No	Excellent	No
Middle-Age	High	No	Fair	yes
Senior	Medium	No	Fair	yes
Senior	Low	yes	Fair	yes
Senior	Low	yes	Excellent	No
Middle-Age	Low	yes	Excellent	yes
youth	Medium	No	Fair	No
youth	Low	yes	Fair	yes
Senior	Medium	yes	Fair	yes
youth	Medium	yes	Excellent	yes
Middle-Age	Medium	No	Excellent	yes
Middle-Age	High	yes	Fair	yes
Senior	Medium	No	Excellent	No

Consider the above training data & predict class label of X = (Age = youth, Income = medium, Student = yes, Credit_Rating = Fair)

= Sol

$P(Buys = yes) = 9/14 = 0.643$ } $P(C_i)$
$P(Buys = No) = 5/14 = 0.357$ }

Page No.
Date : / /

For Age: Youth

$P(Age = youth \mid Buys = yes) = 2/9$

$P(Age = youth \mid Buys = No) = 3/5$

For Income: Medicm

$P(Income = medicm \mid Buys = yes) = 4/9$

$P(Income = medicm \mid Buys = No) = 2/5$

For student: yes

$P(student = yes \mid Buys = yes) = 6/9$

$P(student = yes \mid Buys = No) = 1/5$

For credit Rating: Fair

$P(credit\_Rating = Fair \mid Buys = yes) = 6/9$

$P(credit\_Rating = Fair \mid Buys = No) = 2/5$

now,

$$P(Buys = yes \mid X) = P(Buys = yes) \times \prod_{j=1}^{n} P(x_j \mid C_i)$$

$= P(Buys = yes) \times P(Age = youth \mid Buys = yes) \times P(Income = medicm \mid Buys = yes) \times P(student = yes \mid Buys = yes) \times P(credit\_Rating = Fair \mid Buys = yes)$

$$= \frac{9}{14} \times \frac{2}{9} \times \frac{4}{9} \times \frac{6}{9} \times \frac{6}{9}$$

$= 0.028.$

pc

Page No.
Date : / /

Similarly,

$$P(Buys=No|X) = P(Buys=No) \times \prod_{i=1}^{n} (x_i|C_i)$$

$$= P(Buys=No) \times P(Age=youth|Buys=No) \times P(Income=Medium|Buys=No) \times P(Student=yes|Buys=No) \times P(Credit\ Rating=Fair|Buys=No)$$

$$= \frac{5}{14} \times \frac{3}{5} \times \frac{2}{5} \times \frac{1}{5} \times \frac{2}{5}$$

$$= 0.007$$

Since, $P(Buys=yes|X) > P(Buys=No|X)$; predicted class label for the given tuple is "BuysComputer = yes".

* Variants of Naive Bayes's classifier:

① Gaussian Naive Bayes:

- Gaussian Naive Bayes is used when features of data set have gaussian distribution.
- If features of dataset have continuous values, they follow gaussian distribution.
- In this case, probability of feature x_i belonging to class C is estimated using gaussian probability distribution function as given below.

$$p(x_i|C) = \frac{1}{\sqrt{2\pi}\,\sigma_c} \exp\left(-\frac{(x_i-\mu_c)^2}{2\sigma_c^2}\right)$$

where, μ_c & σ_c are mean & standard deviation

Page No.
Date : / /

of feature belonging to class C.

② Multinomial Naive Bayes:

- It is used when input features follows multinomial distribution. It is primarily used in text classification.
- It calculates probability of features x_i belonging to class C as below:

$$p(x_i|C) = \frac{N_{ic} + \alpha}{N_c + \alpha \times n}$$

where,

N_{ic} is no. of occurences of features x_i in class C.
N_c is the no. of samples belonging to class C.
α is smoothing factor (Normally $\alpha = 1$ is used).
n is no. of dimensions.

③ Bernoulli Naive Bayes:

- It is used when input features follows multivariate Bernoulli distribution. When input features of dataset are binary-value, they follow Bernoulli distribution.
- It calculates probability of feature x_i belonging to class C as below:

$$P(x_i|C) = P(x_i|C)x_i + (1 - P(x_i|C))(1 - x_i)$$

Page No.
Date : / /

* Laplace Smoothing:

- It is the technique used to handle zero probability in Naive Bayes classifier.
- It estimates probability of features x_i belonging to class C as below: (Case of multinomial naive Bayes)

$$p(x_i|C) = \frac{N_{ic} + \alpha}{N_c + \alpha \times n}$$

where,

- N_{ic} is no. of occurrences of features x_i in class C.
- N_c is the no. of samples belonging to class C.
- α is smoothing factor (Normally $\alpha = 1$ is used).
- n is no. of dimensions.

* Example:

Consider the following data set:

Email	Class
Send us your password	Spam
Send your Account	Spam
Review your Account	Ham
Review your password	Ham
Review our profile	Spam
Send us money	Spam
Review your profile	?

= Sol

Assume $\alpha = 1$

Vocabulary = {send, us, your, password, account, review, our, profile, money}

$p(spam) = 4/6$
$p(Ham) = 2/6$

$\left[\because p(x_i|c) = \frac{N_{ic}+\alpha}{N_c+\alpha n}\right]$

① Send:

$p(Send|spam) = \frac{3+1}{4+1\times9} = \frac{4}{13}$

$p(Send|Ham) = \frac{0+1}{2+1\times9} = \frac{1}{11}$

② us:

$p(us|spam) = \frac{2+1}{4+1\times9} = \frac{3}{13}$

$p(us|Ham) = \frac{0+1}{2+1\times9} = \frac{1}{11}$

③ your:

$p(your|spam) = \frac{2+1}{4+1\times9} = \frac{3}{13}$

$p(your|Ham) = \frac{2+1}{2+1\times9} = \frac{3}{11}$

④ password:

$p(password|spam) = \frac{1+1}{4+1\times9} = \frac{2}{13}$

$p(password|Ham) = \frac{1+1}{2+1\times9} = \frac{2}{11}$

Page No.
Date : / /

⑤ Account :

$$p(\text{Account} \mid \text{Spam}) = \frac{1+1}{4+1\times 9} = \frac{2}{13}$$

$$p(\text{Account} \mid \text{Ham}) = \frac{1+1}{2+1\times 9} = \frac{2}{11}$$

⑥ review :

$$p(\text{review} \mid \text{Spam}) = \frac{1+1}{4+1\times 9} = \frac{2}{13}$$

$$p(\text{review} \mid \text{Ham}) = \frac{2+1}{2+1\times 9} = \frac{3}{11}$$

⑦ over :

$$p(\text{over} \mid \text{Spam}) = \frac{1+1}{4+1\times 9} = \frac{2}{13}$$

$$p(\text{over} \mid \text{Ham}) = \frac{0+1}{2+1\times 9} = \frac{1}{11}$$

⑧ profile :

$$p(\text{profile} \mid \text{spam}) = \frac{1+1}{4+1\times 9} = \frac{2}{13}$$

$$p(\text{profile} \mid \text{Ham}) = \frac{0+1}{2+1\times 9} = \frac{1}{11}$$

⑨ money :

$$p(\text{money} \mid \text{spam}) = \frac{1+1}{4+1\times 9} = \frac{2}{13}$$

$$p(\text{money} \mid \text{Ham}) = \frac{0+1}{2+1\times 9} = \frac{1}{11}$$

Page No.
Date : / /

now,

$P(\text{Spam} \mid \text{"Review your Profile"}) = P(\text{Spam} \mid [0,0,1,0,0,1,0,1,0])$

$= P([0,0,1,0,0,1,0,1,0] \mid \text{Spam}) \times P(\text{Spam})$

$= \left(1-\frac{4}{13}\right) \times \left(1-\frac{3}{13}\right) \times \left(\frac{3}{13}\right) \times \left(1-\frac{2}{13}\right) \times \left(1-\frac{2}{13}\right) \times \frac{2}{13} \times \left(1-\frac{2}{13}\right) \times \frac{2}{13} \times \left(1-\frac{2}{13}\right) \times \frac{4}{6}$

$= 0.000994$

$P(\text{Ham} \mid \text{"Review your profile"}) = P(\text{Ham} \mid [0,0,1,0,0,1,0,1,0])$

$= P([0,0,1,0,0,1,0,1,0] \mid \text{Ham}) \times P(\text{Ham})$

$= \left(1-\frac{1}{11}\right) \times \left(1-\frac{1}{11}\right) \times \left(\frac{3}{11}\right) \times \left(1-\frac{2}{11}\right) \times \left(1-\frac{2}{11}\right) \times \frac{3}{11} \times \left(1-\frac{1}{11}\right) \times \frac{1}{11} \times \left(1-\frac{1}{11}\right) \times \frac{2}{6}$

$= 0.00103$

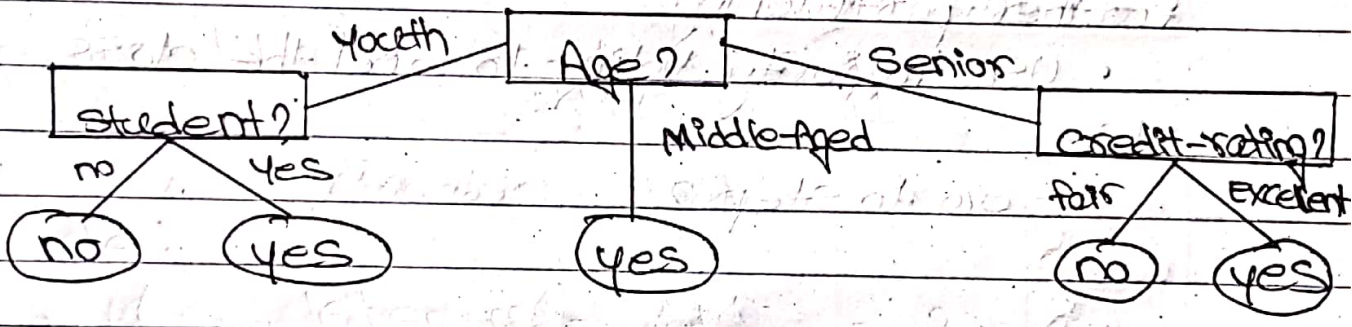
Since, $P(\text{Ham} \mid \text{"Review your Profile"}) > P(\text{Spam} \mid \text{"Review your profile"})$.

predicted class label for the given email message is "Ham".

Page No.
Date : ./ /

* Decision Tree classifier :

- It is the supervised classification algorithm that constructs the decision tree from the given training dataset.
- Decision tree is the tree structure where internal nodes (non-leaf node) denotes a test on an attributes, branches or edges represents outcomes of tests, and leaf nodes (terminal nodes) holds class labels.
- In order to make prediction for a tuple, the attributes of a tuple are tested against the decision tree. A path is traced from the root to a leaf node which determines the predicted class for that tuple.



A decision tree indicating whether a customer is likely to purchase a computer.

(i) Class-label Yes : The customer likely to buy a computer
(ii) Class-label No : The customer is unlikely to buy a computer.

Page No.
Date : / /

* Algorithm for Constructing Decision Tree classifier:

Constructing a decision tree uses greedy algorithm. Tree is constructed in a top-down-divide-and-conquer manner.

1. Initially, all the training tuples are considered in root nodes.
2. Tuples are partitioned recursively based on selected attributes.
3. If all samples of the given node belongs to the same class
   - Label the class.
4. Else if there are no remaining attributes for further partitioning
   - Use majority voting to label the class.
5. Else
   - Go to step 2

- There are many variations of ~~des~~ decision tree algorithms. Some of them are: ID3 (Iterative Dichotomiser 3), C4.5 (successor of ID3), CART (classification and Regression Tree) etc.
- There are different attribute selection measures used by decision tree classifier. Some of them are: Information Gain, Gain Ratio, Gini Index etc.

Page No.
Date : / /

ID3 Decision Tree:

- ID3 stands for Iterative dichotomiser 3. It uses top-down greedy approach to build decision tree model.
- This algorithm uses information gain as attribute selection measure and selects the attributes with the highest information gain.
- Information gain measures reduction in entropy after ~~data~~ the data transformation.
- Entropy is the measure of homogeneity of the sample. Entropy or expected information of dataset D is calculated by using equation below:

$$E(D) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

where, p_i is the probability of a tuple belonging to class C_i and
m is the number of classes.

$$p_i = \frac{|C_i, D|}{|D|}$$

where, $|C_i, D|$ is the total number of samples in D belonging to class C_i and $|D|$ is the total number of samples in training dataset.

- Suppose dataset D is partitioned on the basis of attribute A having V distinct values. Thus, ~~per~~ dataset D is partitioned into V partitions $\{D_1, D_2, \ldots, D_V\}$

now, the total entropy of the dataset after partitioning on A can be calculated as below:

$$E_A(D) = \sum_{i=1}^{v} \frac{|D_i|}{|D|} E(D_i)$$

- Information gain is reduction in entropy after partitioning - Thus, information after partitioning on A can be calculated as below.

$$Gain(A) = E(D) - E_A(D)$$

- ID3 selects attributes with highest information for partitioning.

* Example:

Age	Income	Student	Credit_Rating	Buys_Computer
youth				
youth				
Middle age				
Senior				
Senior				
Middle age				
youth				
youth				
Senior				
youth				
Middle age				

Page No.
Date : / /

Age	Income	Student	Credit_Rating	(class) Buys_computer
youth	High	No	Fair	No
youth	High	No	Excellent	No
Middle_Age	High	No	Fair	yes
Senior	Medium	NO	Fair	yes
Senior	Low	yes	Fair	yes
Senior	Low	yes	Excellent	No
Middle_Age	Low	yes	Excellent	yes
youth	Medium	No	Fair	No
youth	Low	yes	Fair	yes
Senior	Medium	yes	Fair	yes
youth	Medium	yes	Excellent	yes
Middle_Age	Medium	No	Excellent	yes
Middle_Age	High	yes	Fair	yes
Senior	Medium	No	Excellent	No

- Construct decision tree from above data using information gain.
- predict class level of the tuple: X = (age = youth, income = medium, student = yes, credit_rating = fair)

= Sol Entropy of the dataset before partitioning is given as below:

We have 9 positive & 5 negative class out of 14 class

So, S(9+, 5−):

$$E(D) = -\sum_{i=1}^{m} P_i \log_2 P_i$$

$$= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$$

$$= 0.41 + 0.53$$

$$= 0.94$$

Page No.
Date : / /

① Attribute = "Age"

Syouth [2+, 3−] Entropy (Syouth) = $-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$

= 0.971

Smiddle_Age [4+, 0−] Entropy (Smiddle_Age) = $-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$

= 0 [∵ In case of either +=0 or −=0 Ans 0]

Ssenior [3+, 2−] Entropy (Ssenior) = $-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$

= 0.971

now,

$$\text{Gain}(S, \text{Age}) = E(D) - \sum_{i=1}^{v} \frac{|D_i|}{|D|} E(D_i)$$

$$= \text{Entropy}(D) - \left(\frac{5}{14}\text{Entropy}(\text{Syouth}) + \frac{4}{14}\text{Entropy}(\text{Smiddle\_Age}) + \frac{5}{14}\text{Entropy}(\text{Ssenior})\right)$$

$$\text{Gain}(S, \text{Age}) = 0.94 - \left(\frac{5}{14}\times 0.971 + \frac{4}{14}\times 0 + \frac{5}{14}\times 0.971\right)$$

$$= 0.94 - 0.694$$

$$= 0.246$$ ✓

Page No.
Date : / /

② Attribute = "Income"

S_{High} [2+, 2−] Entropy $(S_{High}) = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}$

$= 1.0$ [∵ In case of equal +ve −ve Entropy is 1]

S_{Medium} [4+, 2−] Entropy $(S_{medium}) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}$

$= 0.9183$

S_{Low} [3+, 1−] Entropy $(S_{Low}) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}$

$= 0.811$

now,

$$\text{Gain}(S, \text{Income}) = E(D) - \sum_{i=1}^{v} \frac{|D_i|}{|D|} E(D_i)$$

$$= E(D) - \left(\frac{4}{14}\times 1 + \frac{6}{14}\times 0.9183 + \frac{4}{14}\times 0.811\right)$$

$= 0.94 - 0.904$

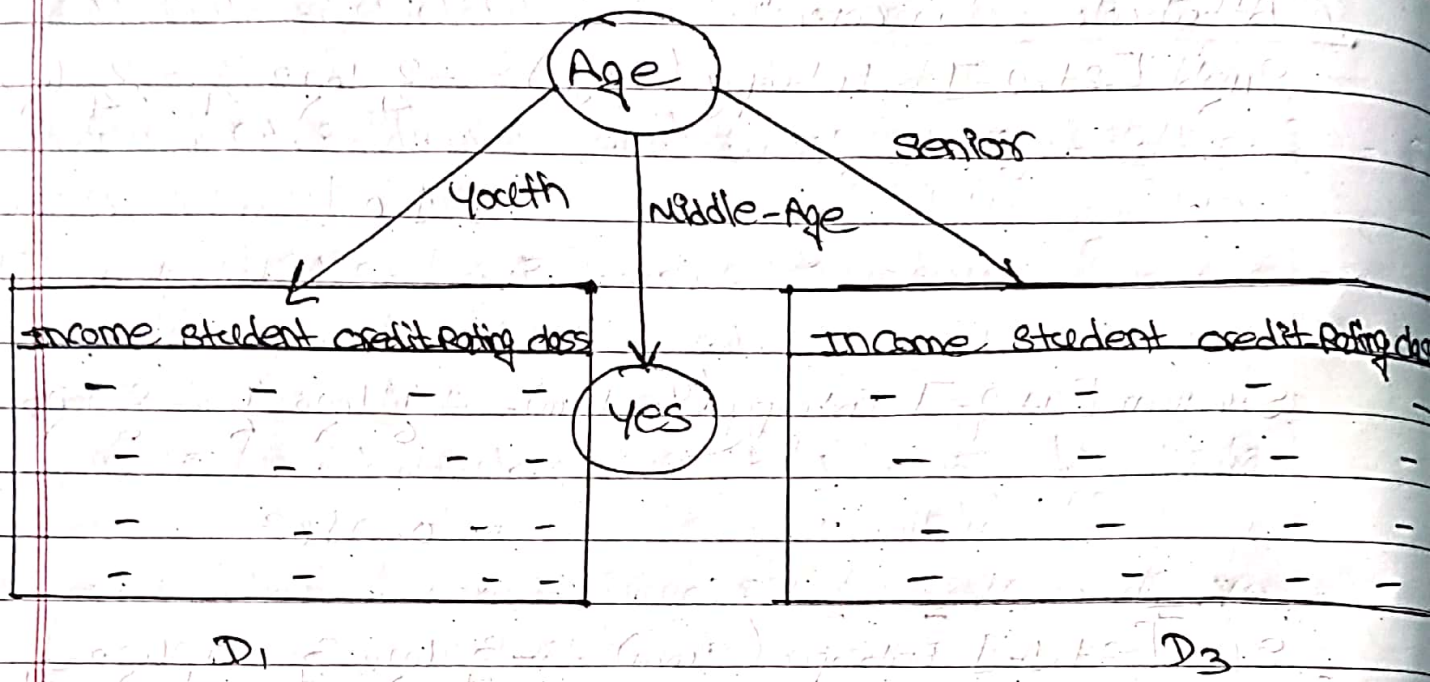
$= 0.036$ ✓

Similarly,

Gain (student) = 0.151 ✓✓

Gain (credit rating) = 0.048 ✓✓

Because age has the highest information gain among the attributes, it is selected as the splitting attribute. The decission tree now looks like below:



- Notice that the tuples falling into the partition for age = middle-age all belongs to the same class. Because they all belongs to class "yes" a leaf should therefore be created at the end of this branch & labeled "yes".

now, consider rows where, only youth is represent we get: (case of left children)

Age	Income	Student	credit_Rating	class
youth	High	No	Fair	No
youth	High	No	Excellent	No
youth	Medium	No	Fair	No
youth	Low	yes	Fair	yes
youth	Medium	yes	Excellent	yes

Page No.
Date : / /

Repeat the same process as above we get,

Entropy of the dataset before partitioning is given as

$S(2+, 3-)$ $E(D) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$

$= 0.528 + 0.442$

$= 0.97$

Attribute = "Income"

$S_{High}(0+, 2-)$ $E(S_{High}) = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2}$

$= 0$ [∵ Incase of any one is 0]

$S_{Medium}(1+, 1-)$ $E(S_{Medium}) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}$

$= 1$

$S_{Low}(1+, 0-)$ $E(S_{Low}) = -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}$

$= 0$

now,

$$\text{Gain}(S_{youth}, \text{Income}) = \text{Entropy}(D) - \sum_{i=1}^{v} \frac{|D_i|}{|D|} E(D_i)$$

$$= E(D) - \left(\frac{2}{5}\times 0 + \frac{2}{5}\times 1 + \frac{1}{5}\times 0\right)$$

$= 0.97 - 0.4$

$= 0.57$ ✓

Page No.
Date : / /

Attribute = "Student"

① S_{No} [0+, 3-] $E(S_{No}) = -\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = 0$

② S_{yes} [2+, 0-] $E(S_{yes}) = 0$

So,

$$\text{Gain}(S_{youth}, \text{student}) = E(D) - \sum_{i=1}^{v} \frac{|D_i|}{D} E(D_i)$$

$$= 0.97 - \left(\frac{3}{5}\times 0 + \frac{2}{5}\times 0\right)$$

$$= 0.97$$ Ans

Attribute = "credit Rating"

① S_{Fair} [2+, 1-] $E(S_{Fair}) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}$

$$= 0.9183$$

② $S_{Excellent}$ [1+, 1-] $E(S_{Excellent}) = 1$ [∵ both are same]

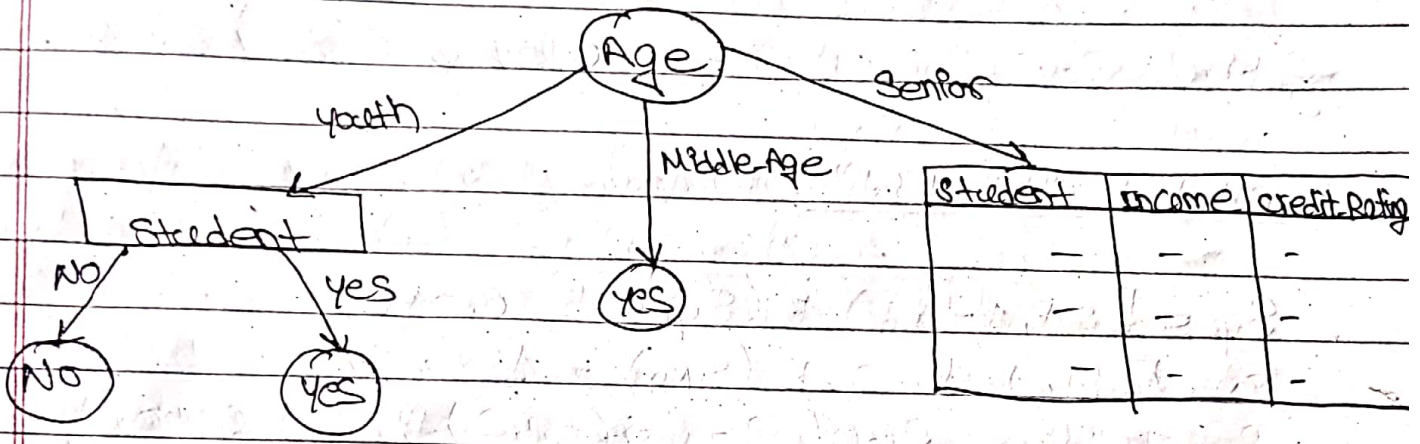
So,

$$\text{Gain}(S_{youth}, \text{credit\_Rating}) = E(D) - \sum_{i=1}^{v} \frac{|D_i|}{|D|} E(D_i)$$

$$= 0.97 - \left(\frac{3}{5}\times 0.9183 + \frac{2}{5}\times 1\right)$$

$$= 0.0192$$ Ans

now, Attribute "student" has the highest information gain among the other attributes we use as splitting attributes. So, the decision tree we get is.



now, Same method is repeated for the ~~left~~ Right children i.e Age="Senior". So, Consider only rows where senior appears i.e.

Age	Income	Student	credit Rating	class
Senior	Medium	Fair	No	yes
Senior	Low	Fair	yes	yes
Senior	Low	Excellent	yes	No
Senior	Medium	Fair	yes	yes
Senior	Medium	Excellent	No	No.

now,

Entropy of dataset before partitioning is given as:

$S(3+, 2-)$ $E(D) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$

$E(Senior) = 0.97$

Attribute "Income"

$S_{Medium}(2+, 1-)$ $E(S_{Medium}) = 0.9183$

$S_{Low}(1+, 1-)$ $E(S_{Low}) = 1$

Page No.
Date : / /

$$\text{Gain}(S_{\text{Senior}}, \text{Income}) = G_I(D) - \sum_{i=1}^{v} \frac{|D_i|}{|D|} E(D_i)$$

$$= 0.97 - \left(\frac{3}{5} \times 0.9183 + \frac{2}{5} \times 1\right)$$

$$= 0.0192$$ ✓

Attribute "Student"

S_{yes} [2+, 1−] $E(S_{\text{yes}}) = 0.9183$

S_{No} [1+, 1−] $E(S_{\text{No}}) = 1$

So,

$\text{Gain}(S_{\text{Senior}}, \text{Student}) = 0.0192$ ✓

Attribute "Credit_Rating"

S_{Fair} [3+, 0−] $E(S_{\text{Fair}}) = 0$

$S_{\text{Excellent}}$ [0+, 2−] $E(S_{\text{Excellent}}) = 0$

So,

$\text{Gain} = 0.97$ ✓

now, Attribute "Credit_Rating" has high Information Gain value, we use it as a Splitting attribute so we get the decision tree as below.

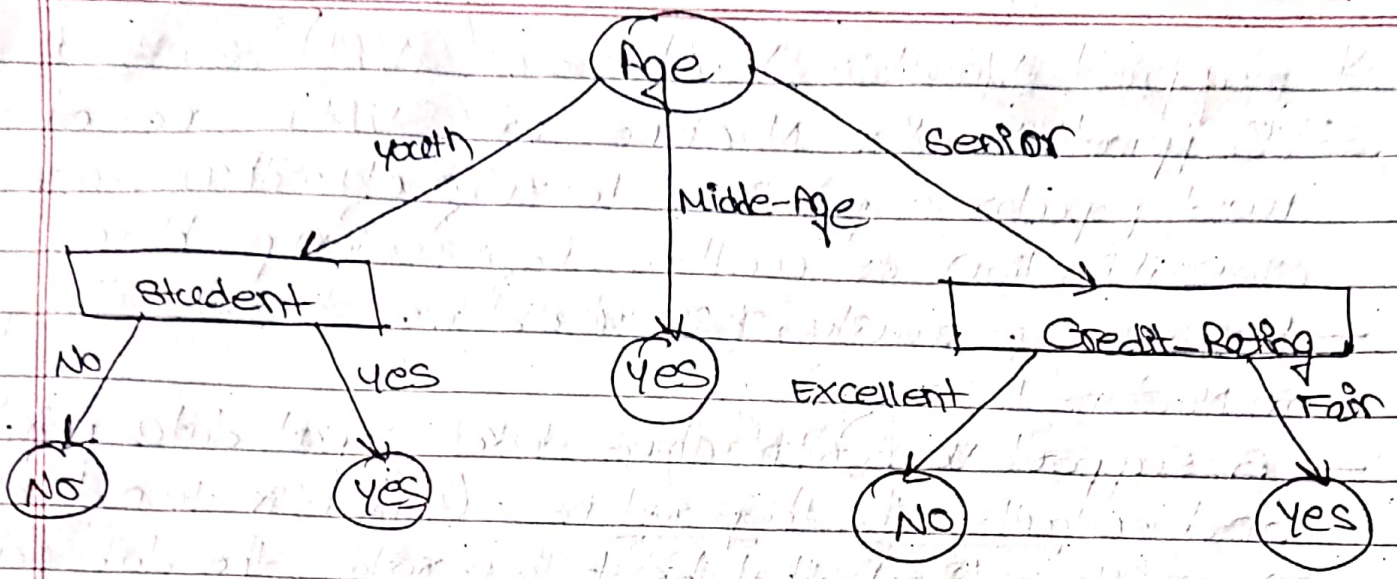


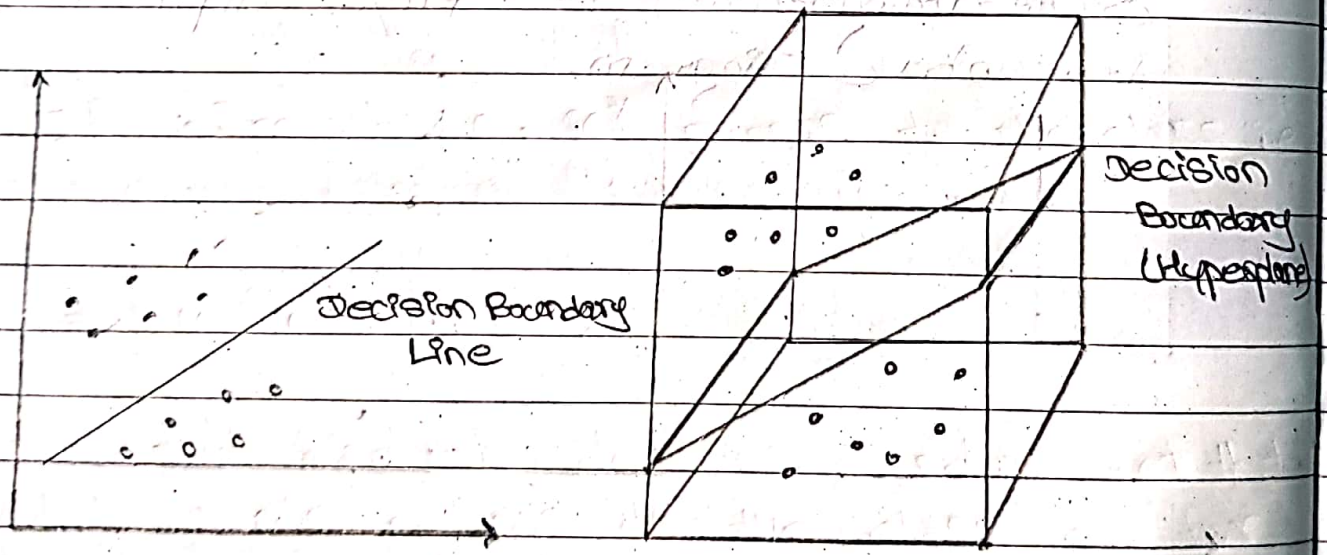
Fig of :- Final decision tree using ID3 Algo.

predicted class level of the tuple
X = (age = youth, income = medium, student = yes, credit_rating = fair) is Buys_computer = yes from above Diagram.

Page No.
Date : / /

* Support Vector Machine (SVM):

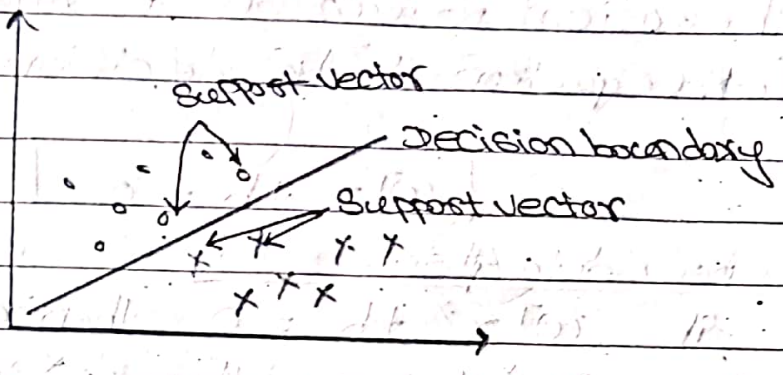
- Support Vector Machine or SVM is one of the most popular Supervised learning algorithms, used for classification as well as Regression problems.
- However, primarily, it is used for classification problems in Machine Learning.
- A Support Vector Machine takes input data points and outputs the hyperplane (which in two dimensions it's simply a line) that best seperates the data points into two classes.
- This line or hyperplane is the decision boundary any data points that falls to one side of it is classified in one class and, the data points that falls to the other of it is classified in another class.



- Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousand).

Page No.
Date : / /

- Support vectors are data points that are closest to the hyperplane and influence the position and orientation of the hyperplane.



- To seperate the two classes of data points, there are many possible hyperplanes that could be choosen. SVM algorithm selects the optimal hyperplane by choosing hyperplane with largest margin. Such hyperplane is called Maximum marginal hyperplane (MMH)

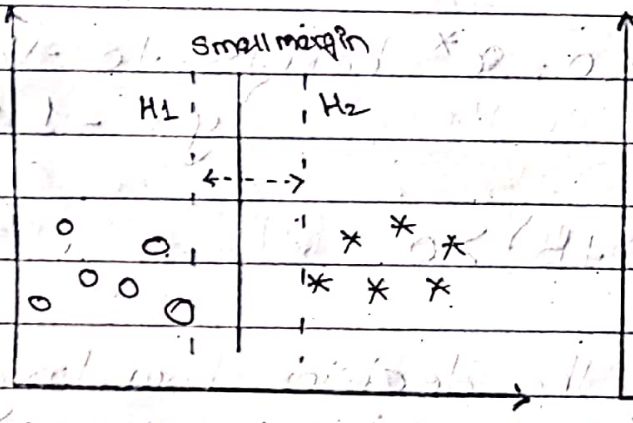


Fig:- Decision boundary with Small margin

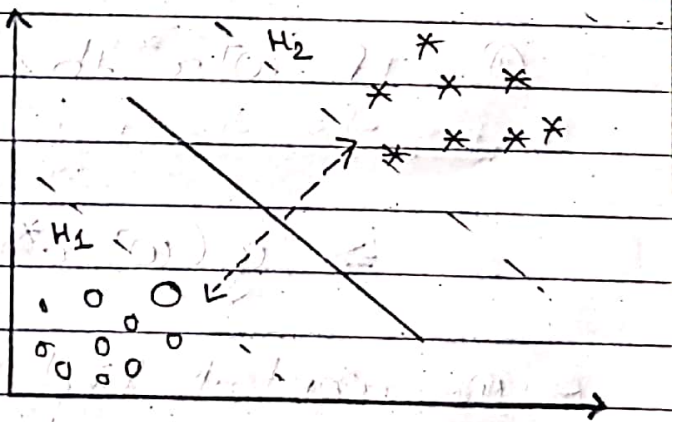


Fig of :- Decision boundary with Large Margin

- Let H1 & H2 are planes that pass through support vectors and parallel to the hyperplane of decision boundary. Margin is the distance between H1 & H2 such that distance between H1 & decision boundary

Page No.
Date : / /

must be equal to the distance between H_2 & the decision boundary.

* Mathematical formulation of SVM:

Let, equation of the decision boundary is

$$w^T x + b = 0 \quad \text{—①}$$

For new data point x^*

If $w^T x^* + b = 0$, then x^* is on the decision boundary

If $w^T x^* + b > 0$, then x^* is above the decision boundary

If $w^T x^* + b < 0$, then x^* is below the decision boundary

For classification,

① If $w^T x^* + b > 0$; x^* belongs to +ve class i.e $y = +1$

② If $w^T x^* + b < 0$; x^* belongs to -ve class i.e $y = -1$

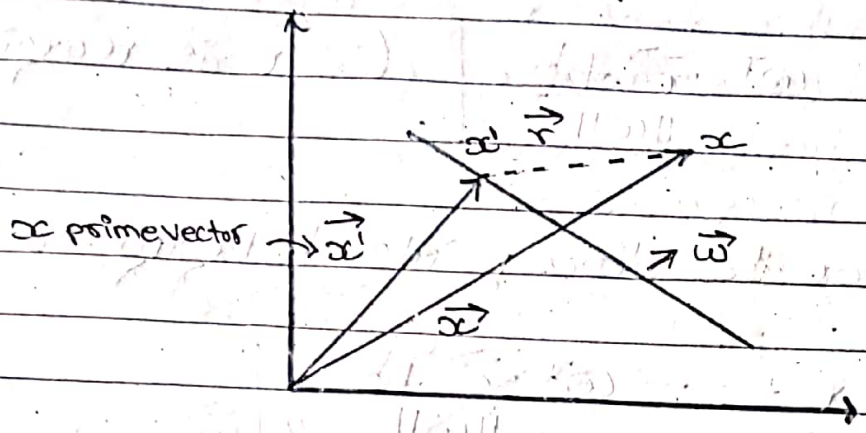
$\Rightarrow y(w^T x^* + b) > 0$

- We want to find the decision boundary such that all data points belongs to +ve class are far from the decision boundary as well as the points belonging to -ve class are also far from the decision boundary so that class label for every data point can be predicted with higher confidence.
i.e we want $|w^T x^* + b|$ is large.

Page No:
Date : / /

$\Rightarrow$ We want $y(w^T x^* + b)$ is large.

now, for the point belonging to the class.



$$\boxed{\vec{r} = \vec{x} - \vec{x'}} \quad \text{---} (2)$$

for, x', $\boxed{w^T x' + b = 0}$ ---(3) x' lies on decision boundary.

Also,

$\vec{r}$ is always parallel to $\vec{w}$

Let $|\vec{r}| = r$

$$\Rightarrow \vec{r} = r \cdot \vec{w} = \frac{r \cdot \vec{w}}{\|w\|} \quad [\because \vec{w} \text{ is unit vector}]$$

from eqn (2) we have

$$\vec{x'} = \vec{x} - \vec{r} = \vec{x} - \frac{r \cdot \vec{w}}{\|w\|}$$

Substituting this value in eqn (3), we get

$$w^T\left(\vec{x} - \frac{r \cdot \vec{w}}{\|w\|}\right) + b = 0$$

$$\Rightarrow w^T \vec{x} - \frac{r \cdot \vec{w} \cdot w^T}{\|w\|} + b = 0$$

Page No.
Date : / /

Since, $w^T \cdot \vec{w} = \|w\|^2$

$\Rightarrow w^T \vec{x} - r\|w\| + b = 0$

$\Rightarrow \boxed{r = \frac{w^T \vec{x} + b}{\|w\|}}$ ($\because$ r is margin)

Similarly, for the data points belonging to -ve class

$$\boxed{r = -\frac{w^T \vec{x} + b}{\|w\|}}$$

Thus, in general margin is given as below,

$$\boxed{r = y\left(\frac{w^T \vec{x} + b}{\|w\|}\right)}$$

① For the +ve point on H_1,

$$r = \frac{w^T \vec{x} + b}{\|w\|} = \frac{1}{\|w\|} \text{ or } \frac{C}{\|w\|}$$

② For the -ve point on H_2,

$$r = -\frac{w^T \vec{x} + b}{\|w\|} = \frac{1}{\|w\|} \text{ or } \frac{C}{\|w\|}$$

$\Rightarrow$ Total margin $= \frac{C}{\|w\|} + \frac{C}{\|w\|} = \frac{2C}{\|w\|} = \frac{2}{\|w\|}$ when $C=1$

Page No.
Date : / /

our goal is to maximize above margin. Maximizing $\frac{2}{\|w\|}$ is equivalent to minimizing $\|w\|$. Thus, we can formulate following optimization problem as:

$$\underset{w,b}{\text{minimize}} = \frac{\|w\|^2}{2}$$

Subject to $y^i(w^T x^i + b) \geq 1$, $i = 1, 2, 3 \ldots n$

Above optimization problem can be solved using quadratic programming.

* Functional and Geometric Margins of SVM:

We can write SVM classifier as:

$$h_{w,b}(x) = g(w^T x + b)$$

where,

$$g(z) = 1 \quad \text{if } z \geq 0$$
$$g(z) = -1 \quad \text{otherwise.}$$

Given a training example (x_i, y_i), functional margins of the decision boundary parameterized by (w, b) is given as below:

$$\gamma_i = y^i(w^T x^i + b)$$

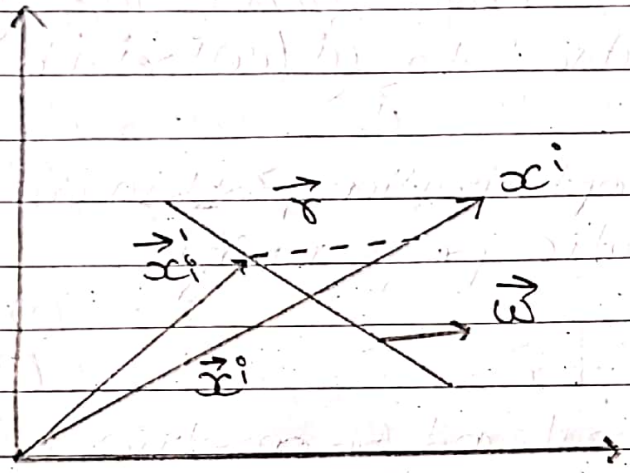
If we replace w by $2w$ & b by $2b$ then

$$g(w^T x + b) = g(2w^T x + 2b)$$

- We can make functional margin arbitrarily large

without changing meaning. This is the drawback of functional margin.

To solve this problem geometric margin is introduced for a +ve point x^i.



$$\vec{r} = \vec{x_i} - \vec{x_i'}$$

$$\Rightarrow \vec{x_i'} = \vec{x_i} + \vec{r} = \vec{x_i} + r_i \frac{\vec{w}}{\|w\|} \quad \left[\because \frac{\vec{w}}{\|w\|} \text{ unit vector}\right]$$

where, $|\vec{r}| = r_i$

since, x_i' is point on decision boundary,

$$\Rightarrow w^T x_i' + b = 0$$
$$\Rightarrow w^T (\vec{x_i} + \vec{r})$$
$$\Rightarrow w^T \left(\vec{x_i} + \frac{r_i \vec{w}}{\|w\|}\right) + b = 0$$

Page No.
Date : / /

$$\Rightarrow r_i = \frac{w^T x_i + b}{\|w\|} = \left(\frac{w}{\|w\|}\right)^T x_i + \frac{b}{\|w\|}$$

$$\text{i.e. } r_i = \left(\frac{w}{\|w\|}\right)^T x_i + \frac{b}{\|w\|}$$

Thus, geometric margin of the decision boundary parameterized by (w, b) is given as below:

$$\boxed{r_i = y_i \left(\left(\frac{w}{\|w\|}\right)^T x_i + \frac{b}{\|w\|} \right)}$$

* Lagrange Duality:

- In mathematical optimization theory, duality is the principle that optimization problem can be viewed from either of two perspective that is the primal problem and the dual problem.
- If the primal problem is minimization problem then the dual problem is maximization problem & vice versa.
- Lagrangian duality theory refers to the way to solve an primal optimization problem by looking at dual optimization problem.

Page No.
Date : / /

* Karush – Kuhn – Tucker (KKT) conditions:

- KKT conditions are tests on first derivations and other Regularity conditions that needs to be satisfied for solving a non-linear optimization problem.
- KKT conditions for solving SVM optimization problem are:

$$\frac{\partial L}{\partial w_i} = 0$$

$$\frac{\partial w}{\partial \beta_i} = 0$$

$$\left\{ \begin{array}{l} \alpha_i g_i(w) = 0 \\ g_i(w) \leq 0 \\ \alpha_i \geq 0 \end{array} \right\}$$

where,

L is Lagrangian duality function.

* Solving SVM optimization problem:

SVM optimization problem is given by

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$\text{s.t.c} \quad y_i (w^T x_i + b) \geq 1$$

$$\text{for } i = 1, 2, \ldots, m$$

Page No.
Date: / /

Above, optimization problem can also be written as below:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$\text{s.t.c} \quad g_i(w) = -y_i(w^T x_i + b) + 1 \leq 0$$

Lagrangian function for the given optimization problem is

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(w^T x_i + b) + 1] \quad \text{---} (1)$$

where, α is Lagrange multipliers.

now,

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

$$\Rightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \alpha_i y_i = 0$$

Substituting values of w in eqn (1), we get:

$$\frac{1}{2}\sum_{i=1}^{m} \alpha_i y_i x_i \cdot \sum_{j=1}^{m} \alpha_j y_j x_j - \sum_{i=1}^{m} \alpha_i \left[y_i \left(\sum_{j=1}^{m} \alpha_j y_j x_j \right) x_i + b \right) + 1$$

Solving this we get,

$$L(w,b,\alpha) = \sum \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j x_i x_j$$

Page No.
Date : / /

Thus, the task is to solve the following optimization.

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j x_i x_j$$

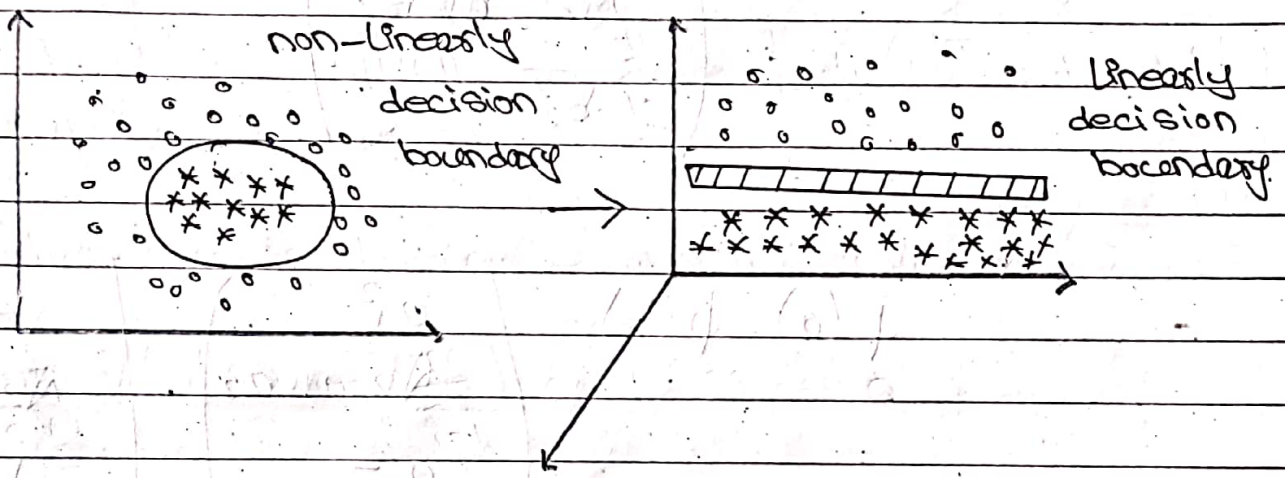
$$s.t. \quad \alpha_i \geq 0$$

$$\alpha_i \cdot y_i = 0$$

Page No.
Date : / /

* SVM Kernel:

- SVM uses a set of mathematical functions which are called Kernels or Kernel functions.
- These functions are used to transform non-linearly seperable low-dimensional data points to linearly seperable high dimensional data points i.e.



Definition: Kernel function is a mathematical function that takes vector in original space as input & returns dot product of vector in transformed (High-dimensional) Space.

i.e.

$$K(a,b) = (a \cdot b)^2$$
$$= \phi(a) \cdot \phi(b)$$

Consider, the following Kernel function.

$$K(a,b) = \phi(a) \cdot \phi(b)$$

Where,

$$\phi\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ \sqrt{2}\, x_1 \cdot x_2 \\ x_2^2 \end{pmatrix}$$

Page No.
Date : / /

here,

$$\phi(a) = \begin{pmatrix} a_1^2 \\ \sqrt{2}\, a_1 a_2 \\ a_2^2 \end{pmatrix}$$

$$\phi(b) = \begin{pmatrix} b_1^2 \\ \sqrt{2}\, b_1 \cdot b_2 \\ b_2^2 \end{pmatrix}$$

now,

$$\phi(a) \cdot \phi(b) = \begin{pmatrix} a_1^2 \\ \sqrt{2} a_1 \cdot a_2 \\ a_2^2 \end{pmatrix} \cdot \begin{pmatrix} b_1^2 \\ \sqrt{2}\, b_1 \cdot b_2 \\ b_2^2 \end{pmatrix}$$

$$= a_1^2 \cdot b_1^2 + 2\, a_1 \cdot b_1 \cdot a_2 \cdot b_2 + b_2^2 \cdot a_2^2$$

$$= (a_1 b_1 + a_2 b_2)^2$$

$$= \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \right)^2$$

$$= (a \cdot b)^2$$

Page No.
Date : / /

* Example: Let say, we have two data points.

$$a = (1, 2), \quad b = (2, 3)$$

now,

$$\phi a = (1, \sqrt{2} \times 2, 4)$$
$$\phi b = (4, \sqrt{2} \times 6, 9)$$

$$\phi(a) \cdot \phi(b) = (1, \sqrt{2} \times 2, 4) \cdot (4, \sqrt{2} \times 6, 9)$$
$$= (4 + 12 \times 2 + 36)$$
$$= 64 \text{ //}$$

using Kernel function, we have

$$K(a, b) = (a \cdot b)^2$$
$$= ((1, 2) \cdot (2, 3))^2$$
$$= (2 + 6)^2$$
$$= 64 \text{ //}$$

Conclusion: Computing dot product of vectors in transformed space is computationally expensive. Kernel functions allows us to do this in computationally efficient manner.

Page No.
Date : / /

* Types of Kernel:

① Linear Kernel:

$$K(a,b) = (a \cdot b)^2$$

② Polynomial Kernel:

$$K(a,b) = (a \cdot b + c)^d$$ (hyperparameter)

where, d is degree of polynomial.

③ Radial basis function kernel:

$$K(a,b) = \exp\left(-\underset{\text{Gamma}}{\Gamma} \|a-b\|^2\right)$$

Γ is prameter whose values ranges between 0 & 1.

④ Gaussian Kernel:

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

where, σ is bandwidth.

⑤ Sigmoid kernel:

It is equivalent to two longes perceptron model of neural network & is given as below:

$K(a,b) = \tanh(\alpha a^T b + c)$

where, α lies between 0 & 1.

⑥ Exponential Kernel:

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

* Kernel Mercer's Theorem:

- Kernel matrix is symmetric matrix that is obtained by applying kernel function K in every pair of points in the dataset.

i.e.

$K_{ij} = K(x_i, x_j)$ where, $i = 1, 2, \ldots, n$

$j = 1, 2, \ldots, n$

or,
$$K_{ij} = \begin{bmatrix} K(x_1,x_1), & K(x_1,x_2), & \ldots & K(x_1,x_n) \\ K(x_2,x_1), & K(x_2,x_2), & \ldots & K(x_2,x_n) \\ \vdots & & & \\ K(x_n,x_1), & K(x_n,x_2), & \ldots & K(x_n,x_n) \end{bmatrix}$$

- Mercers theorem determines which functions can be used as a kernel functions.
- In mathematics, Mercer's theorem states that necessary & sufficient condition for a kernel K

to be valid is that it's kernel matrix is symmetric positive semi-definite.

* The Sequential Minimization optimization (SMO) Algorithm:

- SMO algorithm can be used to solve SVM dual problem efficiently.

* Coordinate Ascent Algorithm:

- It is method of solving unconstrained optimization problem of the form.

$$\max_{\alpha} W(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n)$$

Algorithm:

Loop until Convergence
{
for i = 1 to n
{
$\alpha_i = \arg\max_{\alpha} W(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{i-1}, \hat{\alpha_i}, \alpha_{i+1}, \ldots, \alpha_n)$
}
}

* SMO Algorithm:

The SVM dual problem we have to solve is

$$W(\alpha) = \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^i y^j \alpha_i \alpha_j (x^i \cdot x^j) \quad \text{---(1)}$$

Page No.
Date: / /

such that, $\alpha_i > 0$ — (2)

$$\sum_{i=1}^{n} \alpha_i y^i = 0 \quad — (3)$$

Coordinate ascent algorithm discussed above cannot be used to solve SVM dual problem because if we optimize α_1 by keeping α_2 to α_n fixed, it leads to

$$\alpha_1 y^1 = - \sum_{i=2}^{n} \alpha_i y^i$$

$$\Rightarrow \alpha_1 = \frac{-1}{y^1} \sum_{i=2}^{n} \alpha_i y^i \quad \{ \text{but } \alpha_i > 0 \}$$

This violates constraints (2)
Thus, we must update at least two α_i simultaneously in order to keep constraint satisfying.

Now, SMO algorithm for solving SVM dual problem is given as below:

Repeat until convergence
{
- Select pair of α_i and α_j
- optimize $W(\alpha)$ w.r.t. α_i & α_j by keeping α_k, $k = 1, 2, 3, \ldots, n$, $k \neq i, i$ fixed
}