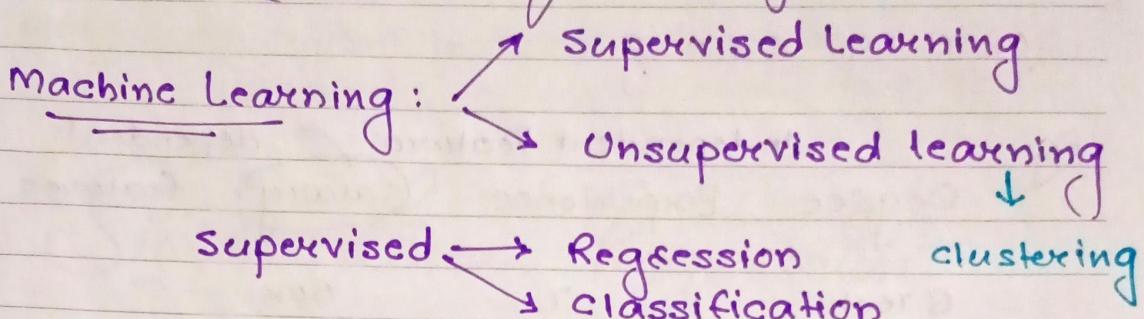


Machine Learning.



Supervised (output is known)

Regression →

1. Linear Regression
2. Polynomial Regression
3. SVR
4. Decision Tree
5. Random Forest
6. Xgboost , 7. KNN

Classification: →

1. Logistic Regression
2. SVM
3. Decision Tree
4. Random Forest
5. Naive Bayes
6. KNN

Unsupervised learning: (output is unknown)

clustering →

DBScan, kmeans;

Hierarchical,

silhouette scoring

• Download Machine Learning:

<https://t.me/AIMLDeepThaught>

Supervised Learning

Eg. Independent feature → dependent features

Degree	Experience	(Salary)
--------	------------	----------

B.Tech	7	50k
PhD	2	70k
-	-	-
!	!	!

O/p = continuous → Regression problem

Eg.

Independent Feat.	dependent	
No. of play hrs	No. of study hrs	Pass/Fail feat.

O/p = categorical = classification problem

- Flight Price Prediction → Regression problem
- Algerian Fire Forest → classification problem
- Predict Air Quality Index → Regression Problem
- Rain Prediction → classification Problem
- Buying day of a person → classification problem

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

Note: Independent features is basically input features.

Simple Linear Regression.

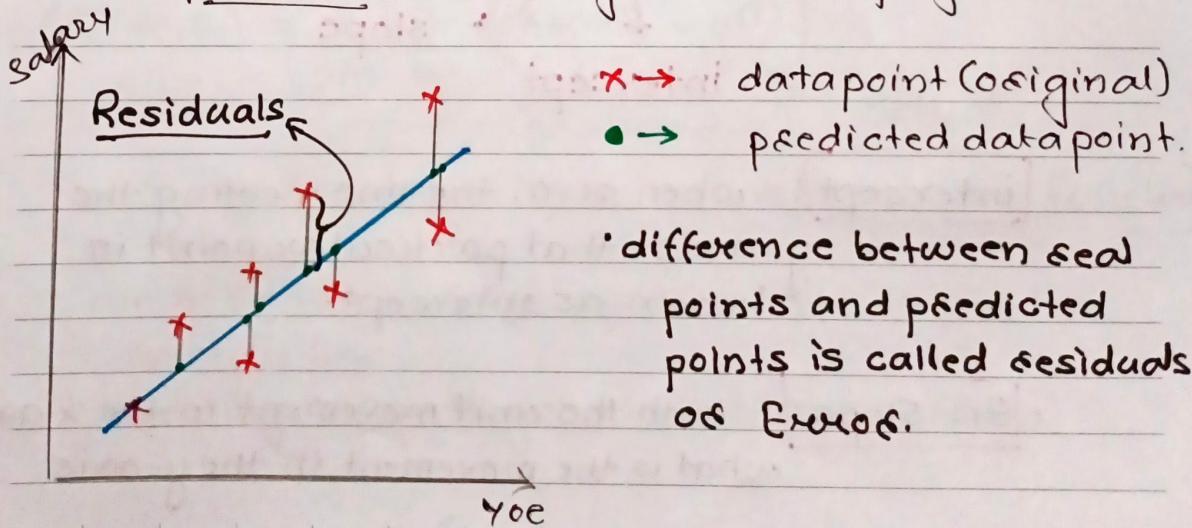
(one independent feature and one dependent feature)

Eg. aim: to create a model, which takes input as height and predict weight.

dataset: height, weight.

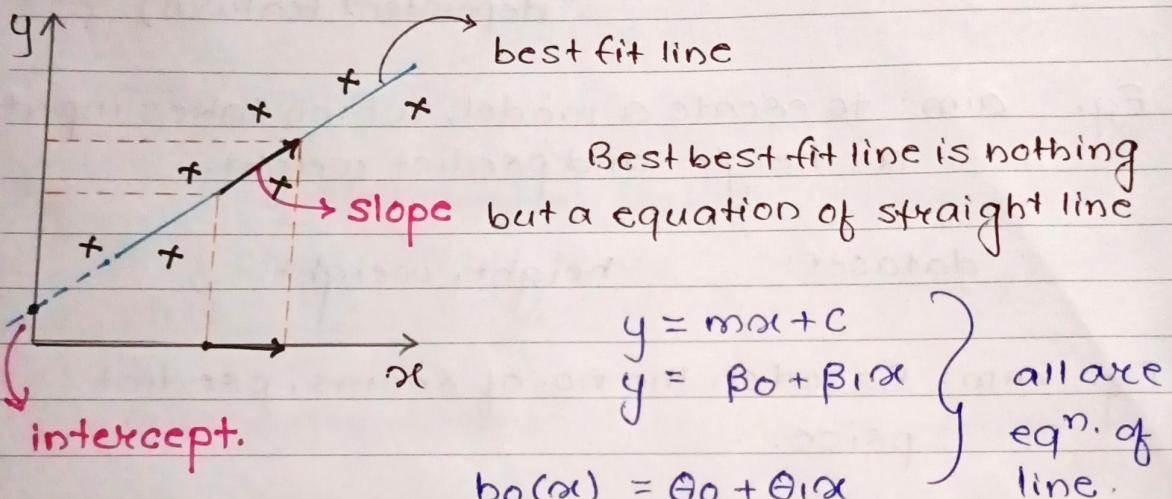
Eg. aim: Based on the no. of rooms, predict price.

Eg. model: year of experience and salary
predict: salary band on 1st year.



- Based on the training dataset, it finds the best fit line in such a way that the sum of difference between real points and predicted should be minimum.

First, we need to understand why are creating a straight line?



$$h_0(x) = \theta_0 + \theta_1 x$$

↓ ↘ slope
intercept

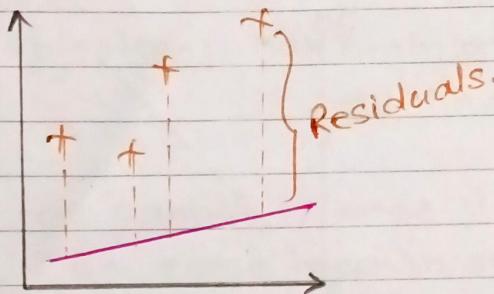
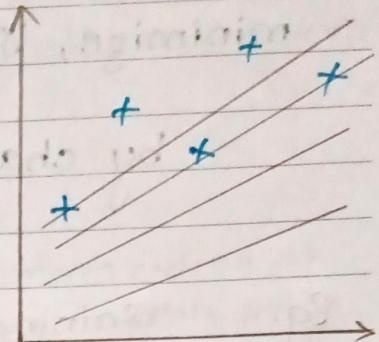
intercept: when $x=0$, the line meeting the ~~longest~~ y-axis, that particular point is known as intercept.

θ_1 : Slope: with the unit movement in the x-axis, what is the movement in the y-axis.

- By changing θ_0 and θ_1 , best fit line will be change.

θ_0, θ_1 : changing the values, will change the line.

after some iterations, we will get a best fit line which is known as training of the model.



we need to minimize this error using equation called as cost function.

Cost Function:

for easy calculations (derivatives)

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

predicted

actual.

mean square error. (one of the cost function)

- we need to minimize cost function to get the best fit line.
- dividing by m to get the average (mean).

final aim:

minimize, $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x)^{(i)} - y^{(i)})^2$

• by changing the values of θ_0 and θ_1 .

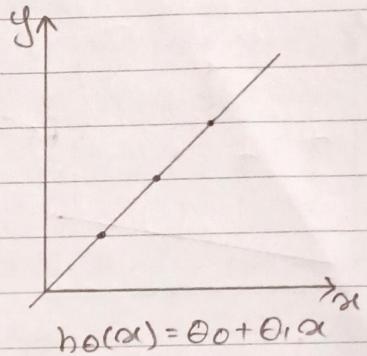
Let us consider

Eg. Training dataset.

x	y
1	1
2	2
3	3

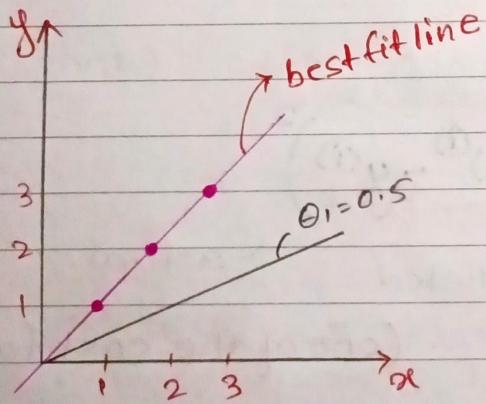
$$\theta_0 = 0$$

$$h_\theta(x) = \theta_1 x$$



Now, let's assume $\theta_1 = 1$

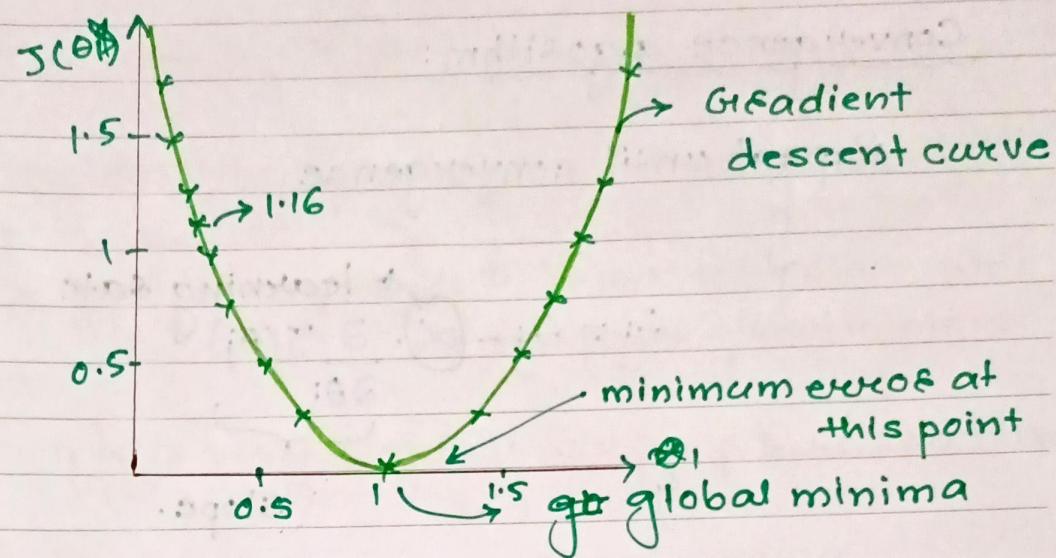
$$\therefore h_\theta(x) = x.$$



$$\begin{aligned} J(\theta_1) &= \frac{1}{m} \sum_{i=1}^m (h_\theta(x)^{(i)} - y^{(i)})^2 \\ &= \frac{1}{3} [0 + 0 + 0] \\ &= 0. \end{aligned}$$

when $\theta_1 = 0.5$

$$\begin{aligned} J(\theta_1) &= \frac{1}{3} [(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \\ &= \underline{1.16}. \end{aligned}$$



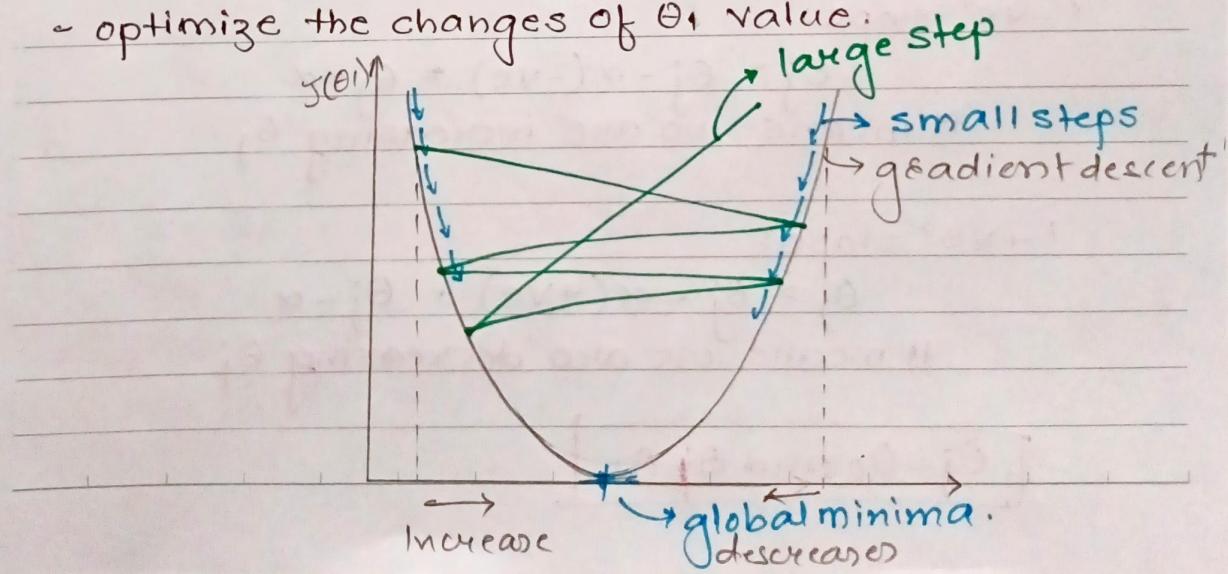
Objective: our main aim is to come near global minima.

we cannot change θ value manually, there should be some mechanism to change θ value to get the global minima.

To overcome this, we use Convergence algorithm.

Convergence algorithm:

- optimize the changes of θ_1 value.



Convergence algorithm:

Repeat until convergence,

{

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta_j)}{\partial \theta_j}$$

↑ learning rate
↓ slope.

}

+ve or -ve slope:

left side of the line facing downwards
→ -ve' slope

right side of the line facing upwards
→ +ve' slope

-ve' slope:

$$\theta_j = \theta_j - \alpha(-\text{ve}) = \theta_j + \alpha$$

It means we are increasing θ_j .

+ve' slope:

$$\theta_j = \theta_j - \alpha(+\text{ve}) = \theta_j - \alpha$$

It means we are decreasing θ_j .

$\theta_j = \theta_0 \text{ and } \theta_j \neq \theta_1$

α , Learning Rate

It decides the speed of the convergence.

If α is very small: It will take more time to reach global minima.

If α is very large: It will jump here and there, and won't reach to global minima.

α , should be around 0.001 for smaller steps.

• Download Machine Learning:
<https://t.me/AIMLDeepThaught>

convergence algorithm:

$$\text{cost function } J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2$$

↓
predicted ↓
actual θ₁ truth point

mean square error.

m = no. of datapoint.

difference between cost function and loss function.

cost function: we find error for all the points and take average of it.

loss function: we find error for observed points and if we find error for all the points then it is loss function.

$$\begin{aligned} \text{loss function} &:= \left(\underbrace{h_{\theta}(x^{(i)})}_{\text{predicted value}} - \underbrace{(y^{(i)})}_{\text{actual value}} \right)^2 \\ &= \left(\hat{y}_j - y_j \right)^2 \end{aligned}$$

* for every single point we need to find loss function.

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

To achieve global minima,

derivative w.r.t θ_0 , $j=0$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_0} \left\{ \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(\alpha)^{(i)} - y^{(i)})^2 \right\}$$

$$= \frac{1}{2m} h_{\theta}(\alpha) = \theta_0 + \theta_1 \alpha$$

$$= \frac{\partial}{\partial \theta_0} \left\{ \frac{1}{2m} \sum_{i=1}^m [(\theta_0 + \theta_1 \alpha)^i - y^i]^2 \right\}$$

$$\frac{\partial}{\partial \alpha} [(1+\alpha) - y]^2 = \frac{1}{m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i) \times 1$$

$$= \underline{2[(1+\alpha) - y]} \times 1 = \frac{1}{m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i)$$

derivative w.r.t. θ_1 , $j=1$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_1} \left\{ \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(\alpha)^{(i)} - y^{(i)})^2 \right\}$$

$$= \frac{\partial}{\partial \theta_1} \left\{ \frac{1}{2m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i)^2 \right\}$$

$$= \frac{1}{m} \sum_{i=1}^m ((\theta_0 + \theta_1 \alpha)^i - y^i) \alpha.$$

Repeat until convergence

{

$$\theta_0 := \theta_0 - \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^m (h_\theta(x)^i - y_i)^2$$

$$\theta_1 := \theta_1 - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (h_\theta(x)^i - y_i) x^{(i)}$$

}

* learning rate: speed of convergence

Types of cost function:

- ① MSE: mean square error
- ② MAE: mean absolute error
- ③ RMSE: root mean square error.

Mean Square Error

$$MSE = \frac{1}{n} \sum_{i=1}^n (y - \hat{y})^2$$

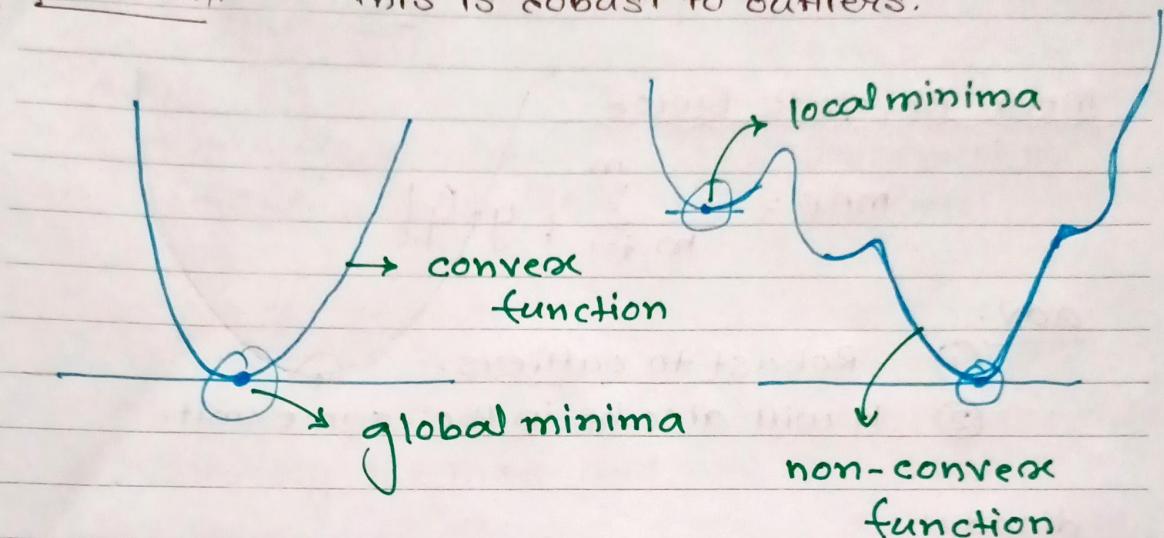
actual.
 $\hat{y} = \theta_0 + \theta_1 x$
predicted.

Adv

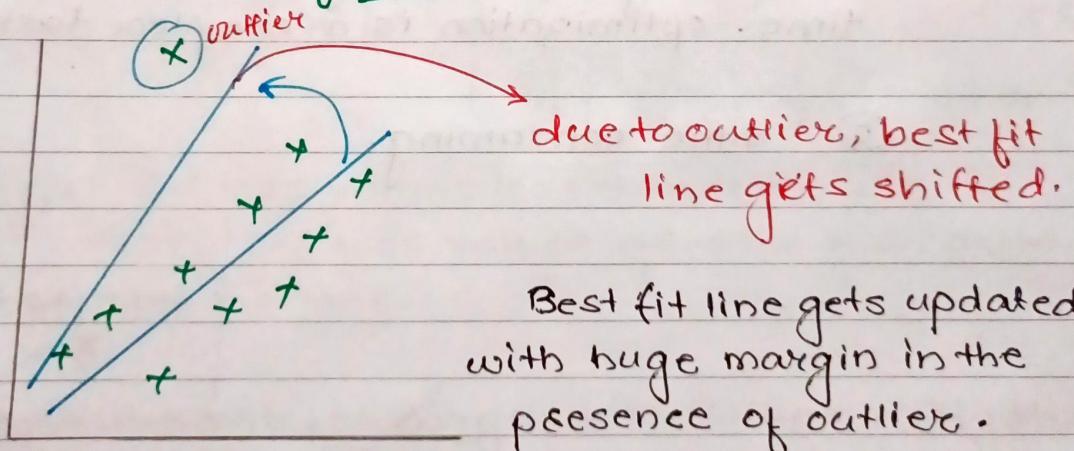
- This equation is differentiable.
- It also has one global minima.

disadv

- This is robust to outliers.



∴ main adv. of MSE that it has convex function.



Eg. (salary) \rightarrow dependent feature

Experience \rightarrow independent feature

$$(y - \hat{y})^2 \text{ (lakhs)}^2$$

\hookrightarrow unit changing \rightarrow time complexities increased.

we don't do scaling for dependent feature.

$(y - \hat{y})^2 \rightarrow$ squared \rightarrow error \rightarrow penalized
↓
Increased.

Mean Absolute Error

$$MAE = \frac{1}{m} \sum_{i=1}^m |y - \hat{y}|$$

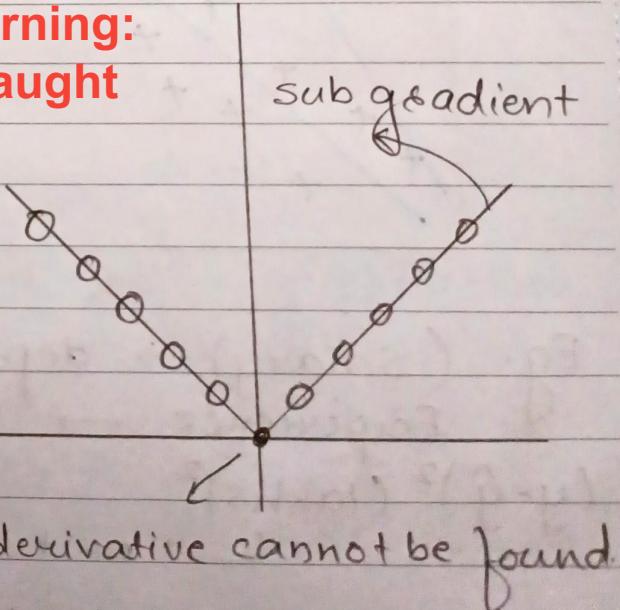
adv:

- ① Robust to outliers.
- ② It will also be in the same unit.

disadv:

- ① convergence usually takes more time. optimization is a complex task.
- ② time consuming.

• Download Machine Learning:
<https://t.me/AIMLDeepThought>



Root mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

adv:

- It is differentiable.
- unit remain same.

disadv:

- Not robust to outliers.

Huber Loss Function

The Huber loss offers the best of both worlds by balancing the MSE and MAE together.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2} (y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2} \delta^2 & \text{otherwise.} \end{cases}$$

It says: for loss values less than delta, use the MSE, for loss values greater than delta, use the MAE.

using the MAE for larger loss values mitigates the weight that we put on outliers so that we still get a well-sounded model.

At the same time, we use the MSE for the smaller loss values to maintain a quadratic function near the centre.

This has the effect of magnifying the loss values as long as they are greater than 1.

Once the loss for those data points dips below 1, the quadratic function down-weights them to focus the training on the higher-error data points.

Note: use the Huber loss any time you feel that you need a balance between giving outliers some weight, but not too much.

How can we check if a model is good or not?
→ using performance metrics.

Performance metrics

- ① R-squared
- ② Adjusted-R squared

R-squared: measures the performance of the model.

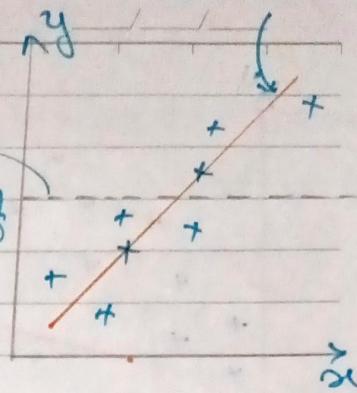
• Download Machine Learning:
<https://t.me/AlMDeepThaught>

Best fit line

$$R\text{-squared} = 1 - \frac{SS_{\text{Res}}}{SS_{\text{Total}}}$$

SS_{Res} = Sum of square Residuals

\bar{y} (average of y)



SS_{Total} = sum of square average

$$R\text{-squared} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

low value
high value

\bar{y} = average of y .

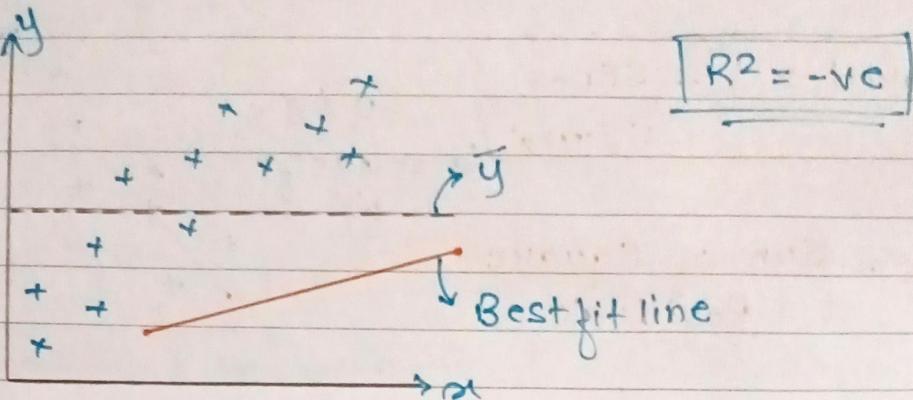
$$= 1 - \left\{ \frac{\text{small value}}{\text{Bigger value}} \right\} \rightarrow \text{small value}$$

R-squared ≤ 1

If R-squared = 0.85 = 85% accurate
= 0.75 = 75% accurate

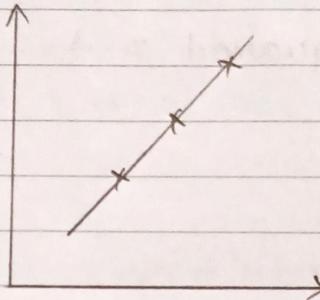
* If R-squared is '0' then the model is not working is good.

- Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>



Here $(y_i - \hat{y}) > (y_i - \bar{y}) \rightarrow R^2$ will be negative.

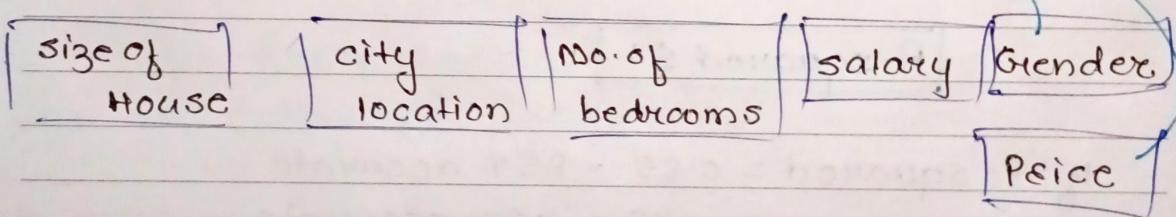
$$R^2 = 1 \rightarrow$$



Adjusted R-squared

Eg.

No direct correlation



Before, when only size of house was present as an independent feature to get the price and has some R-squared value. But with addition of city location R² got increase. after that with addition of no.of bedroom, gender the value got increase, but there is ~~not~~ no direct correlation between gender and price, the value shouldn't increased.

R^2	features added	adj. R^2	Independent feature
65%	size of house	63%	P=1
75%	location	73%	P=2
88%	no. of bedrooms	86%	P=3
90%	Gender	85%	P=4

↓ no direct correlation w/ price
 * very slightly increase, but this shouldn't happen.
 To solve this problem we use Adjusted R^2 .

$$\text{Adjusted } R^2 = \frac{1 - \frac{(1-R^2)(N-1)}{N-p-1}}{1}$$

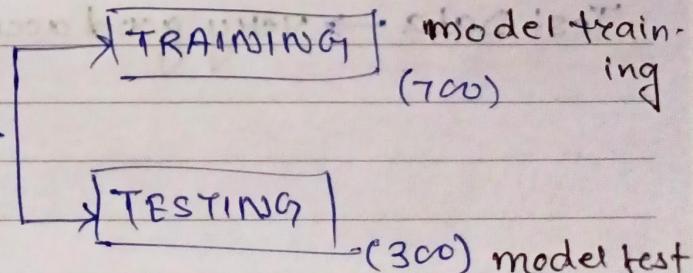
$$N = \text{No. of data points} \quad p = \text{No. of independent features}$$

* Adjusted R^2 is the best metrics to evaluate the model.

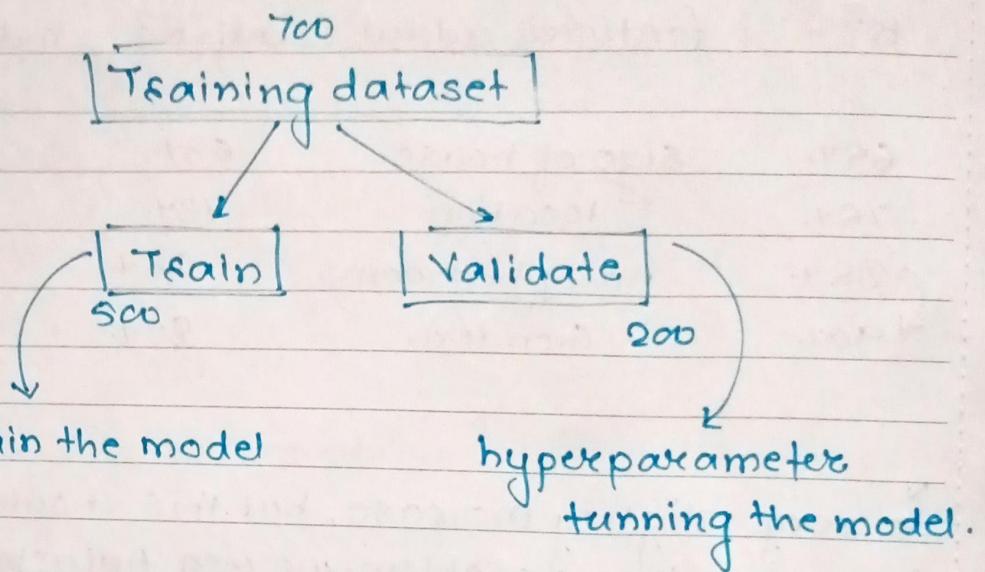
Overfitting and Underfitting

dataset

1000 datapoints



overfitting and underfitting (Bias and Variance)



MODEL

Generalize

Model

Ttrain Data → very good accuracy (90%)

Test Data → very good accuracy (85%)

our aim is to get good test and train accuracy.

Ttrain Data → very good accuracy 90% [low bias]

Test Data → very good accuracy 85% [low variance]

• Download Machine Learning:
<https://t.me/AIMLDeepThaught>

Overfitting
TRAIN → very good accuracy (90%) [low Bias]

TEST → Bad accuracy (50%) [High Variance]

Underfitting

TRAIN → model accuracy is low. [High bias]

TEST → model accuracy is low/high. [low or high variance]

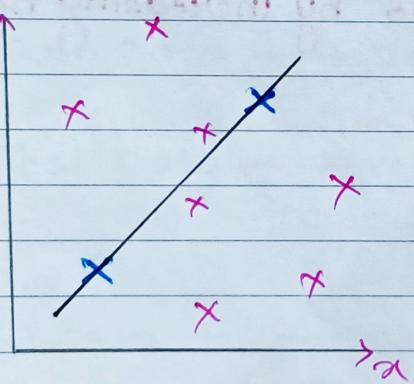
- * we can solve this issue by performing hyperparameter tuning or by increasing no. of dataset.

Regularization

- Sometimes when we train a model it will start to overfit. A way to avoid overfitting data (especially for models like linear regressions that are heavily affected by outliers) we can use **regularization**. This will lead to a more general model that is technically less accurate but generalizes to the data better.

1. Ridge / L2 Regression

(used to reduce overfitting)



cost function = 0.

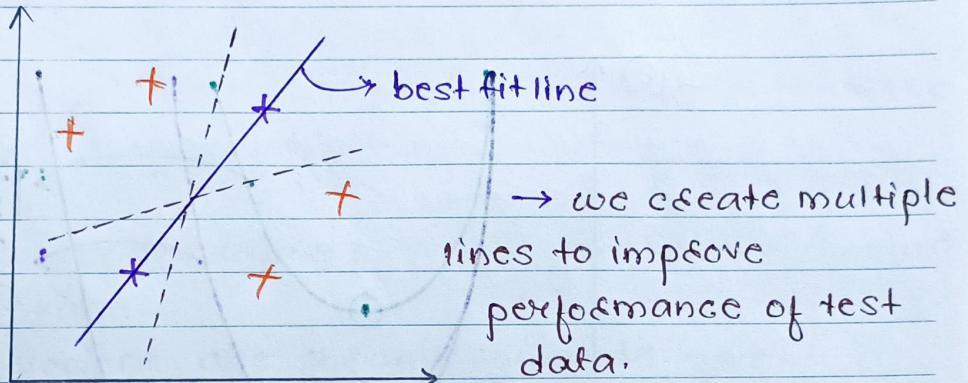
Training data: low bias

Testing data: low/high variance

- If the data (test data) is near to best fit line then performance will be good.
(low variance)

- If the test data is far (away) to best fit line then performance will be bad.
(high variance)

aim: To reduce overfitting



cost Function:

$$\text{cost function} = \frac{1}{m} \sum_{i=1}^m \left\{ h_{\theta}(x^{(i)}) - y^{(i)} \right\}^2 + \lambda (\text{slope})^2$$

λ = hyperparameter

Eg.

$$h_{\theta}(x) = \theta_0 + \theta_1 x \rightarrow \text{slope} = \theta_1$$

If multiple features are present, then

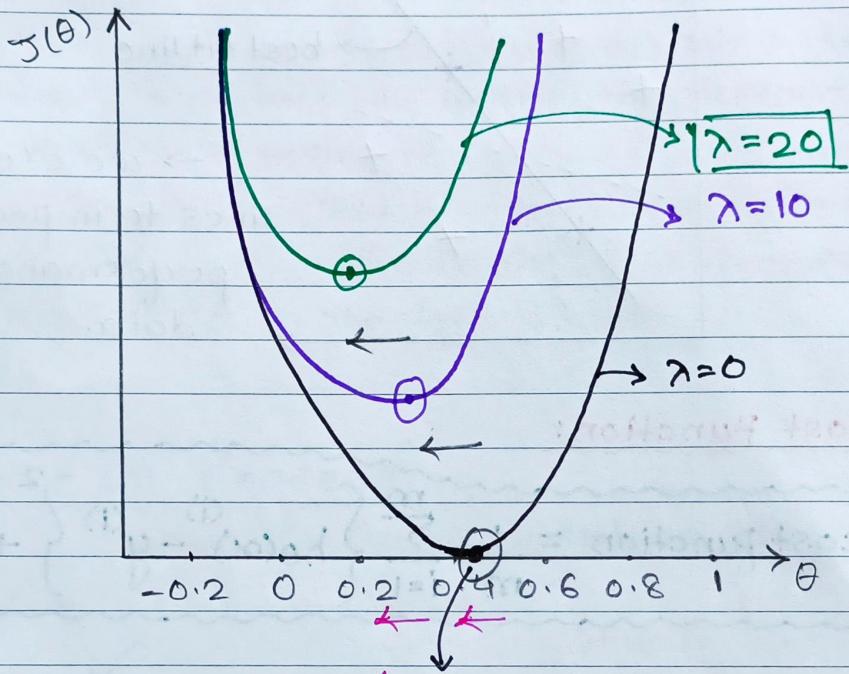
$$(\text{slope})^2 = \sum_{i=1}^n (\text{slope})^2$$

slope = slope of different lines.

If $\lambda = 0$,

cost function is same as linear regression's cost function.

Relationship between Slope and λ



"shifting towards zero" global minima

- Global minima gets shifted towards left with increase in λ .

$$\text{cost function} = \theta + (\text{slope})^2 \cdot \lambda$$

• Join me on LinkedIn for the latest updates on ML:

<https://www.linkedin.com/groups/7436898/>

- change θ value to create another best fit line.

$$\lambda \propto \frac{1}{\text{slope}}$$

* inversely proportional.

λ = It make sure that our line doesn't overfit.

$\lambda \geq 0$, is a complexity parameter that controls the amount of shrinkage:

the larger the value of λ , the greater the amount of shrinkage.

The coefficient are shrunk towards zero.

* θ value never becomes zero!

$$\begin{aligned} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + 0.95 x_1 + 0.82 x_2 + 0.10 x_3 \end{aligned}$$

It will get deleted.

∴ Ridge Regression is used to introduce bias to the data in order to generalize the data and increase bias.

This is useful if you don't have much training data.

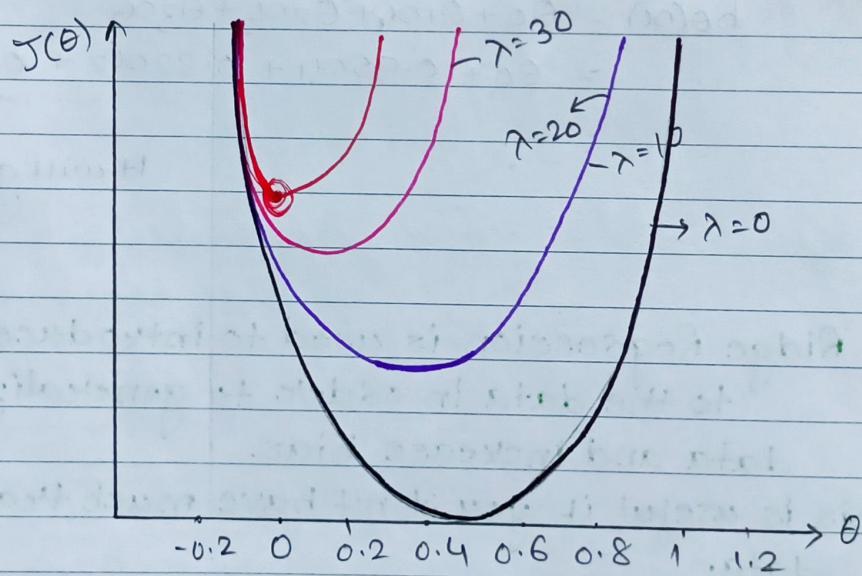
Lasso Regression

(L1 Regularization / L1 Norm)

- It is used to reduce the features. It helps in feature selection.

cost Function:

$$\text{cost function} = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{i=1}^m |\text{slope}|$$



$$\begin{aligned} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= \theta_0 + 0.54x_1 + 0.23x_2 + 0.10x_3 \end{aligned}$$

least correlated.

- If data has outliers \rightarrow use Ridge Regression.

Lasso = Least Absolute Shrinkage and Selection Operator Regression.

- Lasso regression tends to eliminate the weights of the least important features by setting their weights to zero.

Elastic Net

\rightarrow combination of L1 and L2 Regularization.

$$\text{cost function} = \frac{1}{m} \sum_{i=1}^m \{ h_{\theta}(x)^{(i)} - y^{(i)} \}^2 + \lambda \frac{(\text{slope})^2}{L2} + \lambda |\text{slope}|$$

can be changed to MAE, RMSE, MSE.

- Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

Notes taken from John Starmer of Stat Quest

Youtube Videos

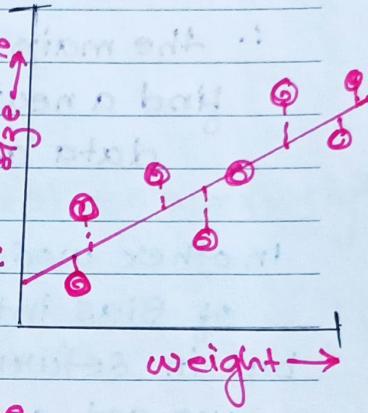
Regularization: Ridge (L2) Regression.

we find the line that results in the minimum sum of squared residuals.

∴ we end up with the eqn of the line:

$$\text{size} = 0.95 + 0.75 \times \text{weight}$$

y-intercept slope

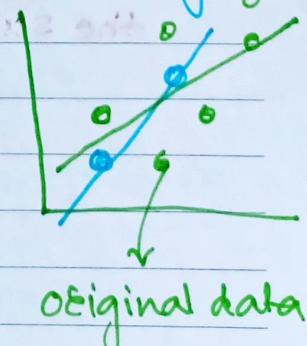


when we have of lot of measurements, we can be fairly confident that least squares line accurately reflects the relationship between size and weight.

But what if we only have two measurements?
we fit new line. since the new line overlaps the two data points, the minimum sum of squared residuals = 0.

∴ New line eqn: $\text{size} = 0.4 + 1.3 \times \text{weight}$

sum of the squared residuals for testing data is large
which means the new line has high variance.



In ML, new line (blue) is overfit to training data.

∴ the main idea behind Ridge Regression is to find a new line that doesn't fit the training data as well.

In other words, we introduce a small amount of Bias into how the new line is fit to the data but in return for that small amount of bias, we get a significant drop in variance.

∴ Ridge Regression can provide better long term predictions.

when least squares determines values for the parameters in this equation



$$\text{size} = \text{y-axis intercept} + \text{slope} \times \text{weight}$$

it minimizes...

the sum of the squared residuals.

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

In contrast,

when Ridge Regression determines values for the parameters in this equation....



$$\text{size} = y\text{-axis intercept} + \text{slope} \times \text{weight}$$

.... it minimizes
the sum of the squared residuals

+

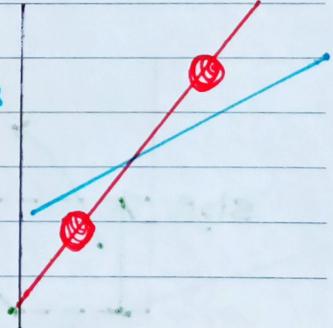
$$\lambda \times \text{slope}^2$$

→ lambda.

this part adds a penalty to
the traditional least square
method.

and λ determines how severe that
penalty is.

→ the sum of squared residuals
for the least square fit
is 0 (because the line overlaps
the data points). and the slope
is 1.3.



$$\therefore 0 + \lambda \times (1.3)^2 = 0 + 1 \times (1.3)^2 = \underline{\underline{1.69}}$$

For blue line →

$$\text{slope} = 0.8$$

$$(0.3)^2 + (0.1)^2$$

$$\lambda + (0.8)^2$$

$\lambda = 1$ all together

$$\underline{\underline{0.74}}$$

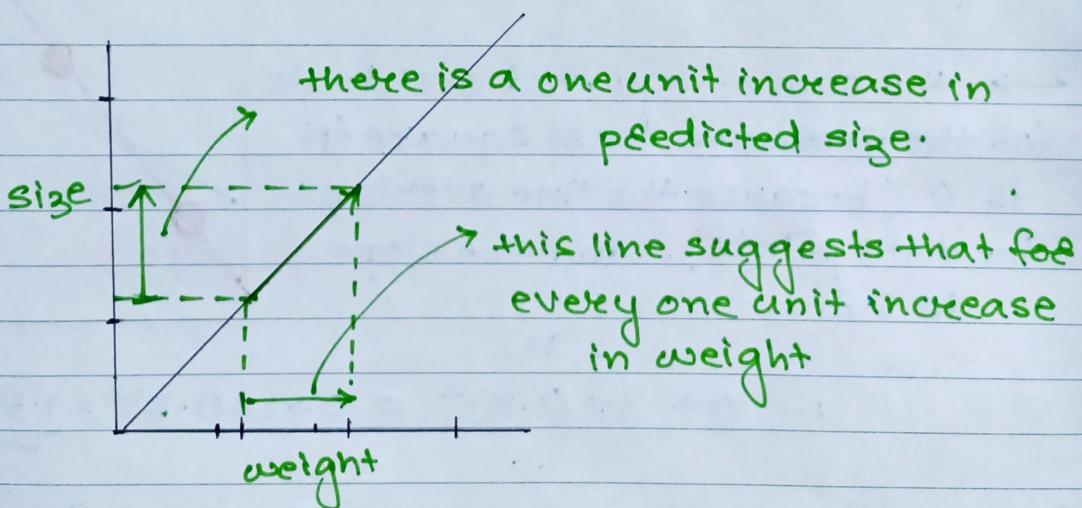
∴ Ridge Regression line :-

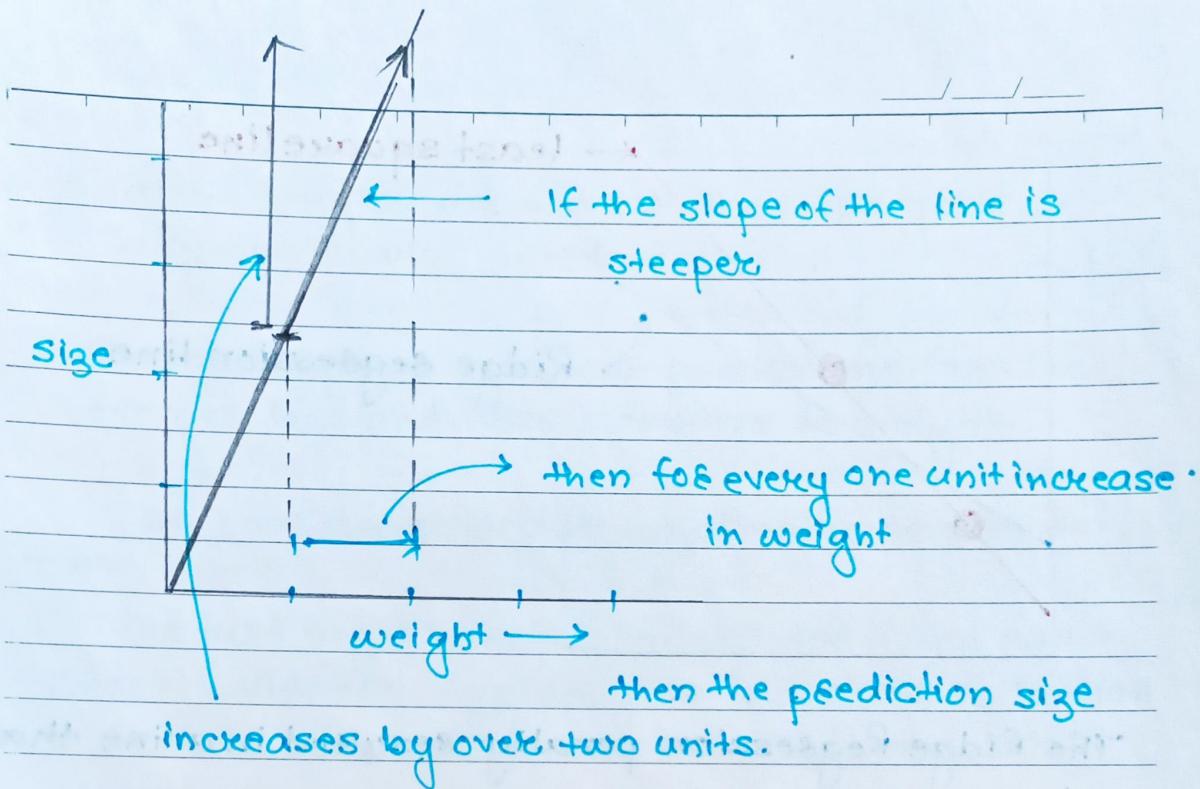
$$\text{Red} = 1.69, \text{Blue} = 0.74$$

Thus, if we wanted to minimize the sum of the squared residuals plus the Ridge Regression penalty, we would choose the Ridge Regression Line over the least square line.

without the small amount of Bias that the penalty creates, the least squares fit has a large amount of Variance.

In contrast, the Ridge Regression line, which has the small amount of Bias due to the penalty, has less variance.

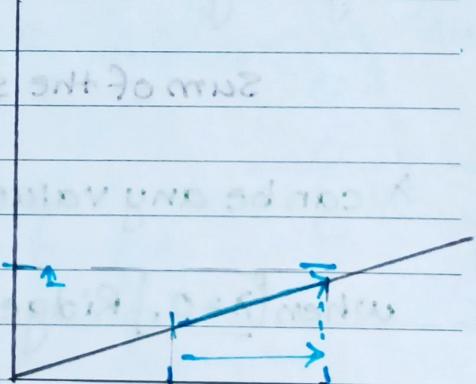




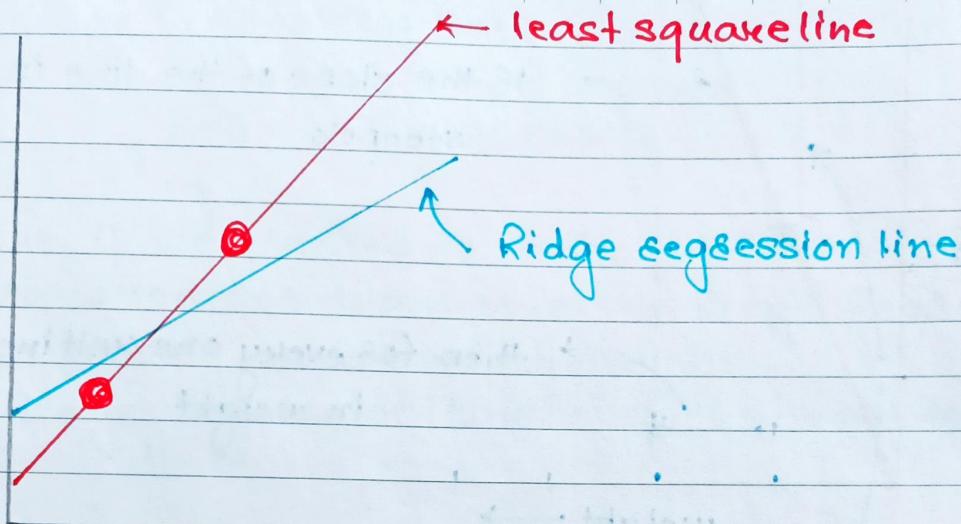
In other words, when the slope of the line is steep, then the prediction for size is very sensitive to relatively small changes in weight.

when the slope is small, then for every one unit increase in weight

the prediction for size barely increases.



In other words, when the slope of the line is small, then predictions for size are much less sensitive to changes in weight.



The Ridge Regression penalty resulted in a line that has a smaller slope ...

which means that predictions made with the Ridge Regression line are less sensitive to weight than the least square line.

Ridge Regression (RR)

$$\text{Sum of the squared residuals} + \lambda \times (\text{slope})^2$$

λ can be any value from 0 to positive infinity.

when $\lambda=0$, Ridge Regression line = least square line:

$\boxed{\lambda=1}$ RR ended up with a smaller slope than the least square line.

and the larger we make λ , the slope gets asymptotically closer to 0.

So, the larger λ gets, our predictions for size becomes less and less sensitive to weight.

→ so how do we decide what value to give λ ?

we just try a bunch of values for λ and use cross-validation, typically 10-fold cross validation, to determine which one results in the lowest variance.

..... until now RR was for continuous variable.

However, RR also works when we use discrete variable.

Discrete variable:

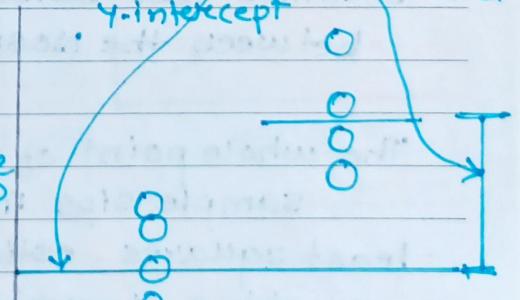
y_1 intercept \rightarrow corresponds to the average size of the mice on the Normal diet.

$$\text{size} = 1.5 + 0.7 \times \text{High fat disease}$$

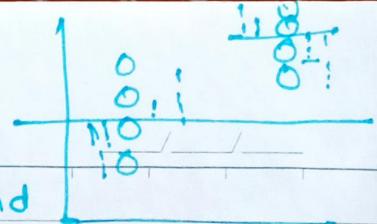
sum of these two is the prediction for the size of the mice on the High fat diet.

$$\text{size} = 1.5 + 0.7 \times \text{High fat diet}$$

y_1 -intercept



Normal diet High fat diet



these distances between the data and the means are minimized.

when RR determines values for the parameters in the equation...

... it minimizes →

the sum of the squared residuals

+

$$\lambda \times (\text{diet difference})^2$$

$\lambda = 0$, least squares error = RR line

$\lambda = 1$, then only way to minimize the whole eqn is to shrink diet distance down.

In other words, as λ gets larger, our prediction for the size of the mice on the high fat diet becomes less sensitive to the difference between the normal diet and high-fat diet.

The whole point of doing RR is because small sample size like these can lead to poor least squares estimates that result in terrible machine learning predictions.

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

Ridge Regression can also be applied to Logistic Regression.

$$= \text{the sum of the likelihoods} + \lambda(\text{slope})^2$$

Note: when applied to Logistic Regression, Ridge Regression optimizes the sum of the likelihoods instead of the squared residuals because Logistic Regression is solved using maximum likelihood.

" Ridge Regression helps reduce variance by shrinking parameters and making our predictions less sensitive to them "

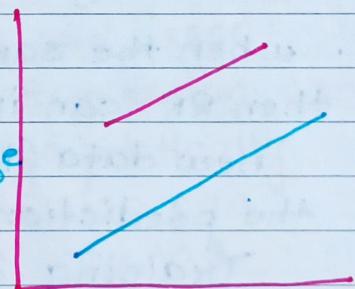
In general, RR penalty contains all of the parameters size except for the y-intercept.

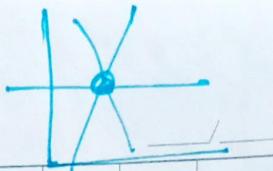
the sum of the squared residuals

+

$$\lambda(\text{slope}^2 + \text{diel distance}^2)$$

weight





least squares can't find a single optimal solution, since any line that goes through the dot will minimize the sum of the squared residuals.

but RR can find a solution with cross validation and the RR ~~penalty~~ penalty that favours smaller parameter values.



sum of the squared residuals

+

$\lambda \times (\text{slope})^2$

Summary: ~~multiple linear regression~~

- when the sample sizes are relatively small, then RR can improve predictions made from new data (i.e. reduce variance) by making the predictions less sensitive to the Training data.

RR penalty itself is λ times the sum of all squared parameters, except for the y-intercept and λ is determined using cross validation.

Lasso Regression: (L1)

Ridge Regression Penalty = $\lambda \times (\text{slope})^2$

" Lasso Regression \rightarrow

sum of all the squared residuals
+ $\lambda \times |\text{slope}|$

Lasso Regression Penalty contains all of the estimated parameters except for the y-intercept.

\therefore Ridge and Lasso Regression shrink parameters, they don't have to shrink them all equally.

Big difference between Ridge and Lasso Regression is that Ridge Regression can only shrink the slope asymptotically close to 0 while Lasso Regression can shrink the slope all the way to 0.

LR can exclude useless variables from equations, better than RR at reducing the variance in models that contain a lot of useless variables.

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

Elastic Net Regression:

Elastic-Net Regression starts with least-squares then combines the Lasso Regression penalty with the Ridge Regression penalty.

sum of the squared residuals

+

$\lambda_1 | \text{variable}_1 | + \dots + | \text{variable}_n |$

+

$\lambda_2 (\text{variable}_1)^2 + \dots + (\text{variable}_n)^2$

Note: LR and RR penalty get their own λ s.

The hybrid Elastic Net Regression is especially good at dealing with situations when there are correlations between parameters.

This is because on its own, Lasso Regression tends to pick just one of the correlated terms and eliminates the others whereas RR tends to shrink all of the parameters for the correlated variables together.

By combining LR and RR,

Elastic-Net Regression groups and shrinks the parameters associated with the correlated variables and leaves them in eqn or removes them all at once.

Logistic Regression

- classification problems

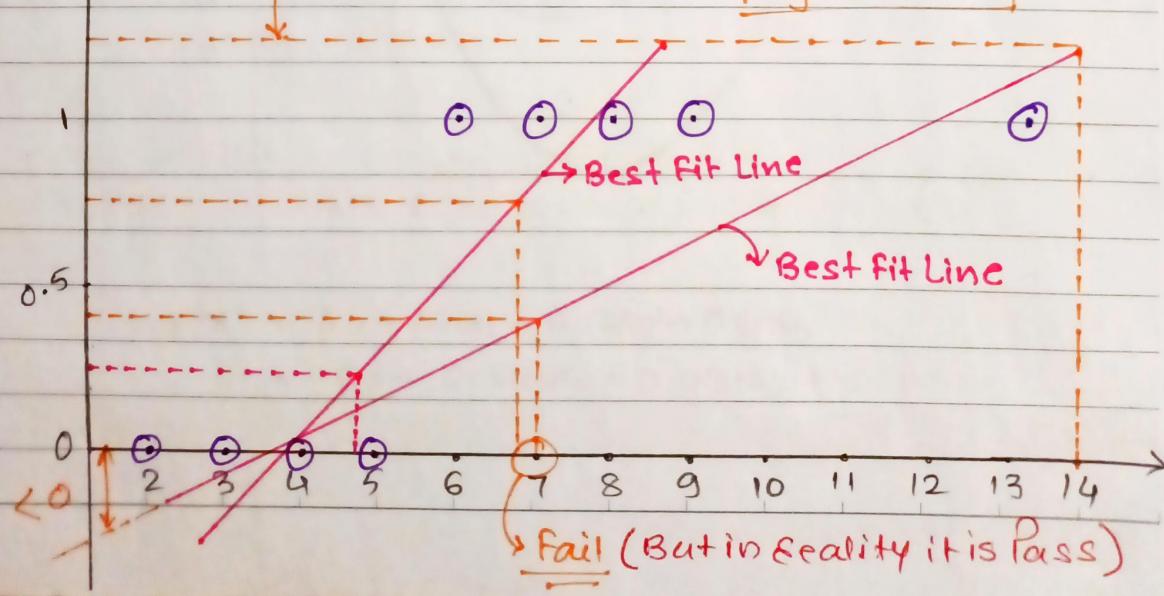
Eg.
dataset:

Study hours	Play hours	O/P (Pass/Fail)
2	8	Fail
3	7	Fail
6	3	Pass
Outliers → 1	4	Pass

* we cannot perform regression, we need to perform classification into Pass/Fail.

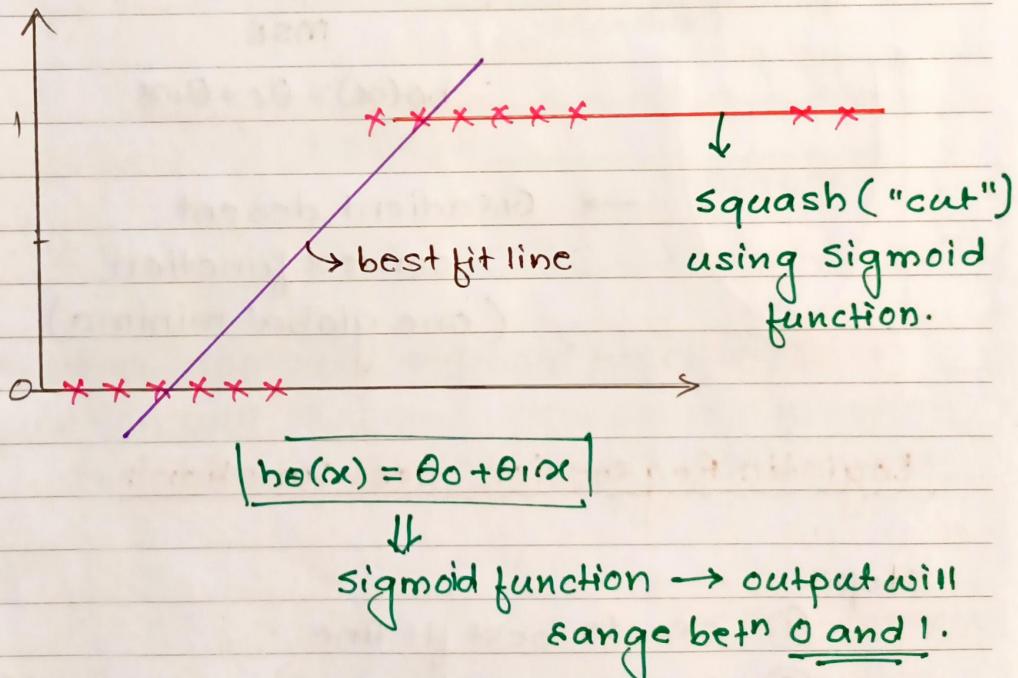
output ≥ 1 , but we have two conditions only.

$$0.5 = \text{threshold}$$
$$\begin{cases} y \leq 0.5 = 0 \\ y > 0.5 = 1 \end{cases}$$



why we use logistic Regression when we can solve classification problem using Linear Regression?

- due to outliers best fit line gets change and results will be wrong.
- we cannot remove outliers always.
- Threshold can't be changed, once fixed.
- In logistic we squash ("cut") the line. we will not change the line.



• Download Machine Learning:
<https://t.me/AIMLDeepThought>

so, even if a person study for 7 hrs

$y < 0.5$

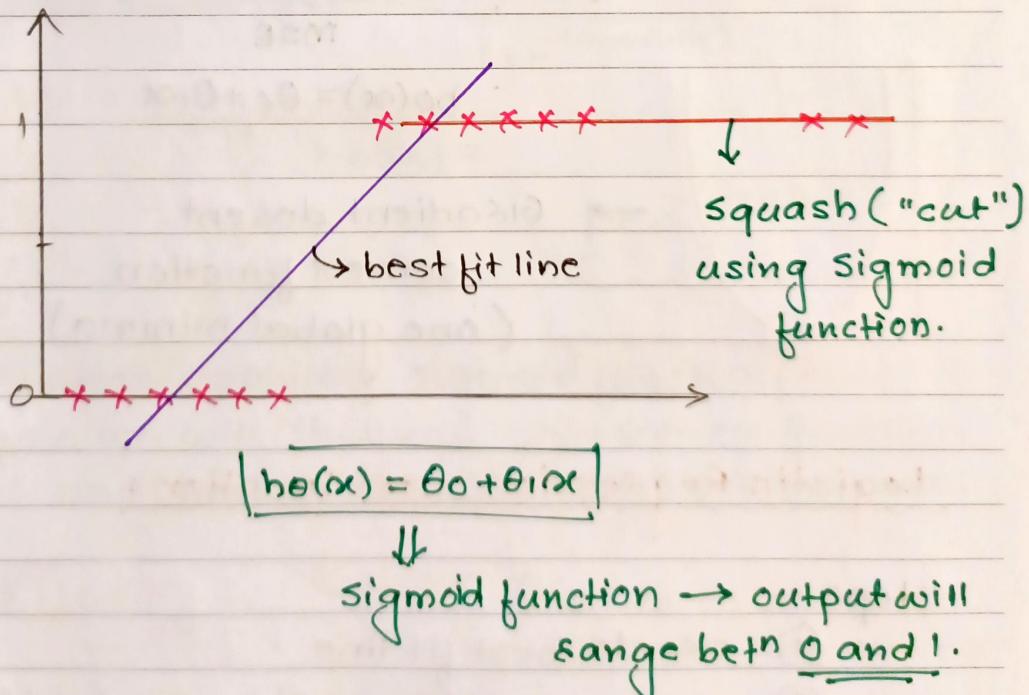
∴ the model will show Fail.

∴ we cannot used Linear Regression in this type of problem statement.

in we can sing

change and

- we cannot remove outliers always.
- Threshold can't be changed, once fixed.
- In logistic we squash ("cut") the line. we will not change the line.



Sigmoid Function:

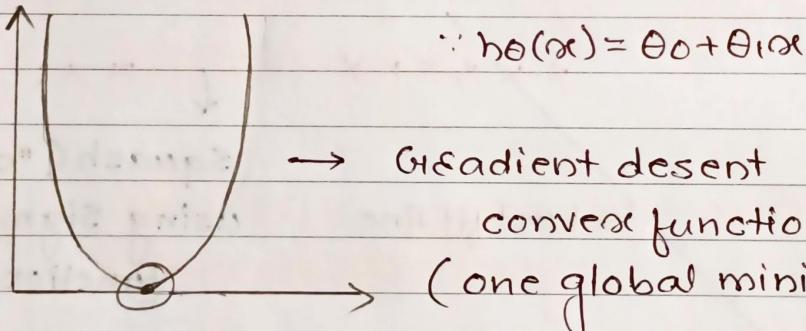
$$\text{sigmoid function} = \frac{1}{1+e^{-x}} \rightarrow \text{betn } 0 \text{ & } 1.$$

1. Create a best fit line.
2. Squashing \rightarrow sigmoid function

Linear Regression cost function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m \left\{ h_\theta(x^{(i)}) - y^{(i)} \right\}^2$$

MSE



Logistic Regression cost function:

Steps:

- ① Create best fit line
- ② apply squashing using sigmoid function.

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m \left\{ h_{\theta}(x^{(i)}) - y^{(i)} \right\}^2$$

$h_{\theta}(x) = \sigma(\theta_0 + \theta_1 x)$ sigmoid function

Let $z = \theta_0 + \theta_1 x$

$$\text{let } z = \theta_0 + \theta_1 x$$

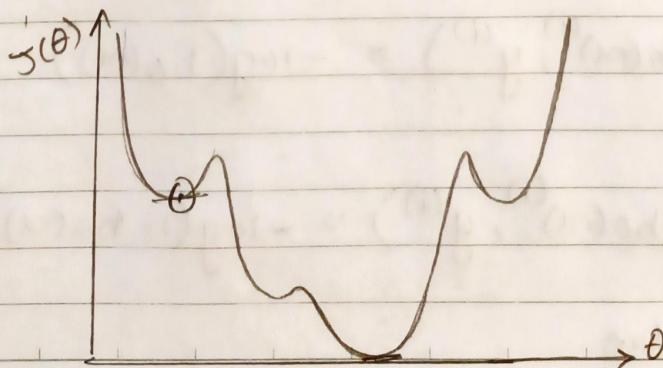
$$h_{\theta}(x) = \sigma(z)$$

$$h_{\theta}(x) = \frac{1}{1 + e^{-z}}$$

$$\therefore h_{\theta}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

$$\therefore h_{\theta}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

but, after applying sigmoid function, cost function will become non-convex function and have chances to get local minima.



← Non-convex function.

- change the cost function to solve the convexity problem.

Log Loss Cost Function

cost function = $\begin{cases} -\log(h_\theta(x)), & \text{if } y=1 \\ -\log(1-h_\theta(x)), & \text{if } y=0 \end{cases}$

converse function

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_0+\theta_1x)}}$$

$$\text{cost}(h_\theta(x)^{(i)}, y^{(i)}) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

→ This will never give local minima.

If $y=1$,

$$\text{cost}(h_\theta(x)^{(i)}, y^{(i)}) = -\log(h_\theta(x))$$

If $y=0$,

$$\text{cost}(h_\theta(x)^{(i)}, y^{(i)}) = -\log(1-h_\theta(x))$$

y = truth value

minimize cost function $J(\theta_0, \theta_1)$ by changing θ_0, θ_1 .

Convergence Algorithm:

Repeat Convergence

{

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

}

for $j = 0$ and $j = 1$

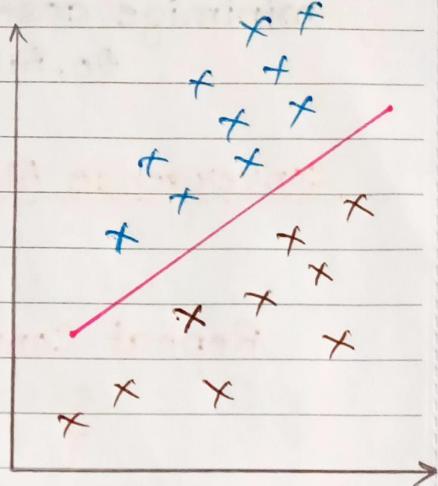
- by default take threshold = 0.5
using ROC and TOC curve, we can define threshold.

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

Performance Metrics

1. Confusion matrix
2. Accuracy
3. Precision
4. Recall
5. F- β eta score



dataset.

f_1	f_2	Op	$P = \hat{y}$ (model prediction)
-	-	0	1
-	-	1	1
-	-	0	0
-	-	1	1
-	-	1	1
-	-	0	1
-	-	1	0

Confusion matrix

1 → correct prediction
0 → wrong prediction

		← Actual value (y)	
		1	0
predicted value (\hat{y})	1	3	2
	0	1	1

		Actual value	
		1	0 (y)
Predicted value (\hat{y})	1	TP FP	correct } TP: True Positive match } TN: True Negative
	0	FN TN	wrong } FP: False Positive match } FN: False Negative

Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Eg. accuracy = $\frac{3+1}{3+2+1+1} = \frac{4}{7} \approx 57\%$

dataset \rightarrow Binary classification



1000 datapoints



900 100
(1) (0)

\leftarrow Imbalanced dataset

dumb model \rightarrow 1 \Rightarrow we get 90% accuracy.

- If the accuracy is 90%, it is not sufficient. Model is not good. To overcome this problem we can use Recall and Precision.

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

Precision:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Actual	
1	0
↑	1
TP	FP
FN	TN

out of all the actual values
how many are correctly
predicted.

• our aim is to reduce False Positive (FP)

Eg. ① mail → spam or ham

TP: actual → spam and predicted → spam
(Good)

FN: actual → spam and predicted → ham
(not good)

FP: actual → ham and predicted → spam
(mail is not spam) (critical problem)
focus on reducing FP.

② Diabetes or Not diabetes

Actual → diabetes and Predicted → not diabetes
critical problem → reduce FN.

Recall :

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

out of all the predicted values how many are currently predicted.

Eg. Tomorrow the stock market is going to crash.

Two points → consumers → FN ↓

→ companies → FP ↓

(can take certain decision)

		Actual	
		TP	FP
Predicted	0	FN	TN
	1		

consumer

for FP companies can take action.
company (action: Sale shares
and discounted price)

F-Beta Score

$$F\text{-}\beta\text{ Beta Score} = \frac{(1+\beta^2) \text{Precision} \times \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

① If FP and FN are both important

$$\beta = 1$$

∴
$$\text{F1 score} = \frac{2 \cdot P \times R}{P + R}$$

② If FP is more important than FN

$$\beta = 0.5$$

$$\text{F-0.5 score} = \frac{(1+0.25)P \times R}{0.25 \times P + R}$$

③ If $\text{FN} \gg \text{FP}$, (FP is less important than FN),

$$\beta = 2$$

$$\text{F2 score} = \frac{(1+4)P \times R}{4 \times P + R}$$

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

In-depth Insights

Logistic Regression

Classification

Classification problem, is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values.

$y \in \{0, 1\}$
Binary class

0: " -ve class"

1: " +ve class"

Hypothesis Representation

Intuitively, it doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$. To fix this, let's change the form for our hypothesis $h_\theta(x)$ to satisfy $0 \leq h_\theta(x) \leq 1$.

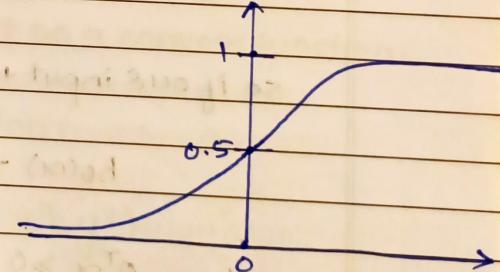
This is accomplished by plugging $\theta^T x$ into the logistic function.

Our new form uses the "sigmoid function" also called the "logistic function".

$$h_\theta(x) = g(\theta^T x)$$

$$\hat{x} = \theta^T x$$

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



Function $g(\hat{x})$, shown here, maps any real number to the $(0, 1)$ interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification.

$h_\theta(x)$ will give us the probability that our output is 1.

Eg. If $h_\theta(x) = 0.7 \rightarrow$ output: 1

thus $h_\theta(x) = 0.3 \rightarrow$ 0

$$h_{\theta}(\alpha) = P(y=1|\alpha; \theta) = 1 - P(y=0|\alpha; \theta)$$

$$P(y=0|\alpha; \theta) + P(y=1|\alpha; \theta) = 1$$

Decision Boundary

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows.

$$h_{\theta}(\alpha) \geq 0.5 \rightarrow y=1$$

$$h_{\theta}(\alpha) < 0.5 \rightarrow y=0$$

The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

$$g(z) \geq 0.5 \quad \text{when } z \geq 0$$

Remember:

$$z=0, e^0=1 \Rightarrow g(\alpha)=1/2$$

$$z \rightarrow \infty, e^{\infty} \rightarrow \infty \Rightarrow g(\alpha)=1$$

$$z \rightarrow -\infty, e^{-\infty} \rightarrow 0 \Rightarrow g(\alpha)=0$$

so if our input to g is $\theta^T \alpha$, then that means

$$h_{\theta}(\alpha) = g(\theta^T \alpha) \geq 0.5 \quad \text{when } \theta^T \alpha \geq 0$$

$$\therefore \theta^T \alpha \geq 0 \rightarrow y=1$$

$$\theta^T \alpha \leq 0 \rightarrow y=0$$

The decision boundary is the line that separates the area where $y=0$ & $y=1$. It is created by our hypothesis function.

E.g.

$$\Theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$

$$y = 1 \text{ if } 5 + (-1)x_1 + 0 \cdot x_2 \geq 0$$

$$5 - x_1 \geq 0$$

$$x_1 \leq 5$$

$$\underline{x_1 \leq 5}$$

In this case, our decision boundary is a straight vertical line placed on the graph where $x_1 = 5$, and everything to the left of that denotes $y=1$, while everything to the right denotes $y=0$.

again, the input to the sigmoid function $g(x)$ (e.g. $\Theta^T x$ doesn't need to be linear, could be a function that describes a circle (e.g., $x = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit our data).

Cost Function

we cannot use the same cost function that we use for linear regression because the logistic function will cause the output to be wavy, causing many local optima.

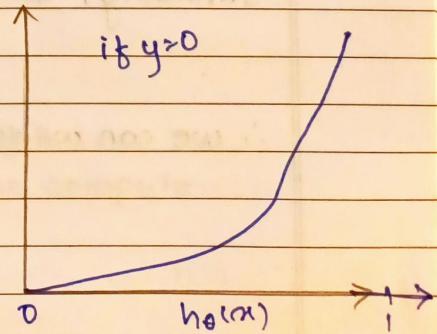
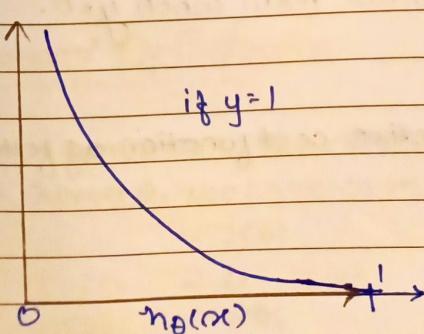
In other words, it will not be a convex function.

Cost function for logistic function:

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^m \text{cost}(h_\Theta(x^{(i)}), y^{(i)})$$

$$\text{cost}(h_\Theta(x^{(i)}), y^{(i)}) = -\log(h_\Theta(x^{(i)})) \text{ if } y^{(i)}=1$$

$$\text{cost}(h_\Theta(x^{(i)}), y^{(i)}) = -\log(1-h_\Theta(x^{(i)})) \text{ if } y^{(i)}=0$$



$$\text{cost}(h_{\theta}(x), y) = 0 \text{ if } h_{\theta}(x) = y$$

$$\text{cost}(h_{\theta}(x), y) \rightarrow \infty \text{ if } y=0 \text{ and } h_{\theta}(x) \rightarrow 1$$

$$\text{cost}(h_{\theta}(x), y) \rightarrow \infty \text{ if } y=1 \text{ and } h_{\theta}(x) \rightarrow 0$$

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Note that writing the cost function in this way guarantee that $J(\theta)$ is convex for logistic function regression.

Simplified Cost function and Gradient Descent

We can compress our cost function's two conditional cases into one case.

$$\text{Cost}(h_{\theta}(x), y) = -y \cdot \log(h_{\theta}(x)) - (1-y) \cdot \log(1-h_{\theta}(x))$$

Thus, when we substitute $y=1$ in abv eqn,

$$\text{cost}(h_{\theta}(x), y) = -y \cdot \log(h_{\theta}(x))$$

similarly, we get another term when $y=0$.

∴ We can write our entire cost function as follows:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \cdot \log(h_{\theta}(x^{(i)}) + (1-y^{(i)}) \cdot \log(1-h_{\theta}(x^{(i)}))]$$

A vectorized implementation is:

$$\begin{aligned} h &= g(x\theta) \\ J(\theta) &= \frac{1}{m} \cdot (-y^T \log(h) - (1-y)^T \cdot \log(1-h)) \end{aligned}$$

Gradient Descent:

General form of gradient descent is:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

we can workout the derivative part using calculus to get:

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

}

Notice that this algorithm is identical to the one we used in linear regression. we still have to simultaneously update all values in theta. ($h_{\theta}(x)$ is different)

A vectorized implementation is:

$$\theta := \theta - \alpha \frac{x^T(g(x\theta) - \bar{y})}{m}$$

Advanced Optimization

cost function $J(\theta)$. want min $J(\theta)$.

Given θ , we have code that can compute

- $J(\theta)$

- $\frac{\partial}{\partial \theta_j} J(\theta)$ for $j = 0, 1, \dots, n$

θ_j

Optimization algorithms:

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS.

advantages :-

- (1) No need to manually pick α .
- (2) often faster than gradient descent

disadv :- more complex to implement.

Multiclass classifications

Now we will approach the classification of data when we have more than two categories.

Instead of $y \in [0, 1]$ we will expand our definition so that
 $y \in [0, 1, \dots, n]$

since $y \in [0, 1, \dots, n]$, we divide our problem into $n+1$ (+1 bcz the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$$y \in \{0, 1, \dots, n\}$$

$$h_{\theta}(x) = P(y=0|x; \theta)$$

$$h_0(x) = P(y=1|x; \theta)$$

!

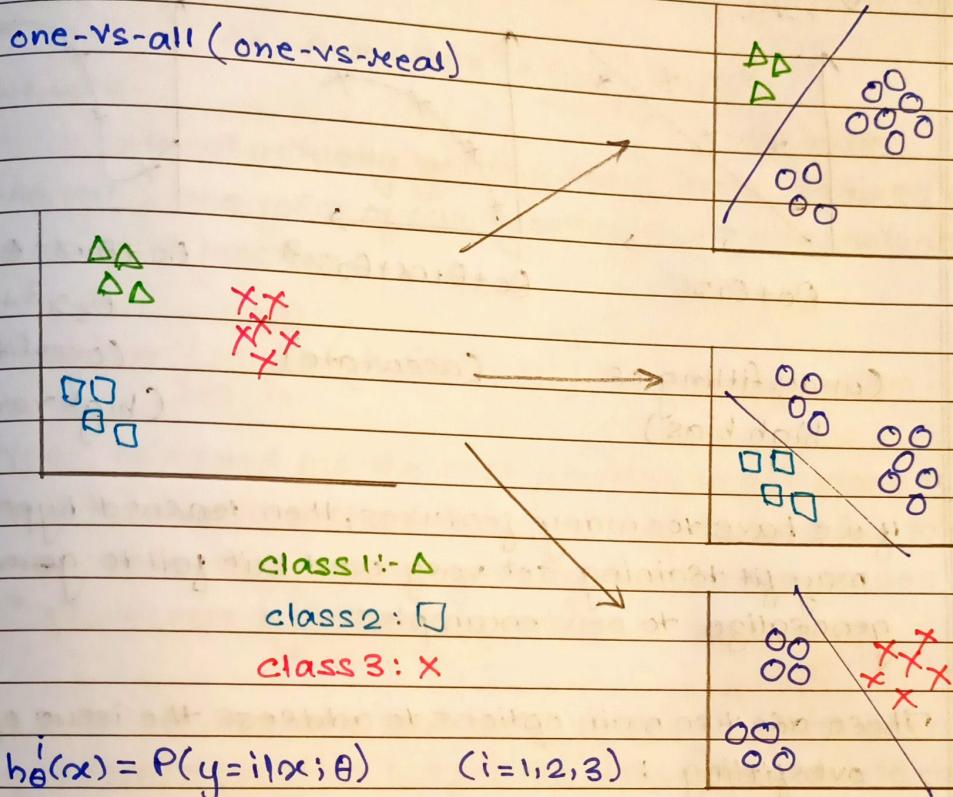
$$h_n(x) = P(y=n|x; \theta)$$

$$\text{prediction} = \max_f(h_i(x))$$

we are basically choosing one class and then lumping all the others into a single second class. we do this repeatedly;

applying binary logistic regression to each case, and thus use the hypothesis that returned the highest value as our prediction.

one-vs-all (one-vs-rest)



To summarize:

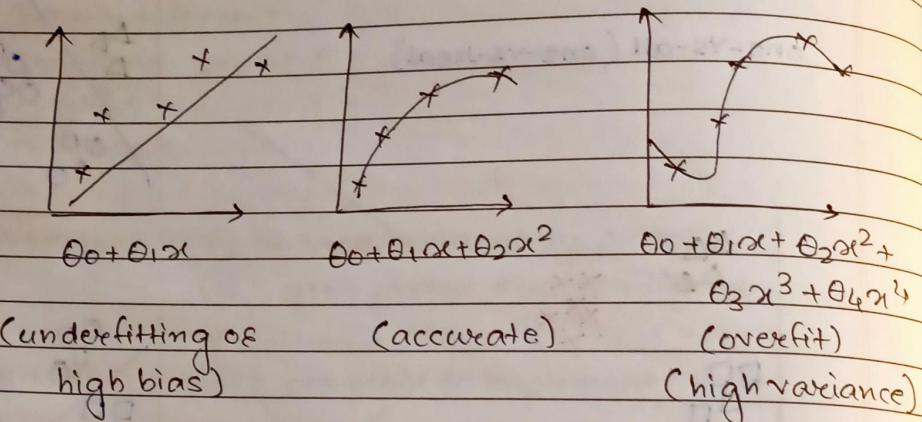
Train a logistic regression classifier $h_{\theta}(x)$ for each classifier to predict the probability that $y=i$

on a new input x , to make a prediction, pick the class that maximizes

$$\max_i h_{\theta}^{(i)}(x)$$

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

Regularization



If we have too many features, then learned hypothesis may fit training set very well but fail to generalize to new examples.

These are two main options to address the issue of overfitting:

1. Reduce the number of features:

- manually select which features to keep.
- use a model selection algorithm

2. Regularization

- keep all the features, but reduce the magnitude of parameters θ_j .

• Regularization works well when we have a lot of slightly useful resource features.

Cost function

If we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our

function costly by increasing their cost.

Eg.

We wanted to make the following function more quadratic:

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

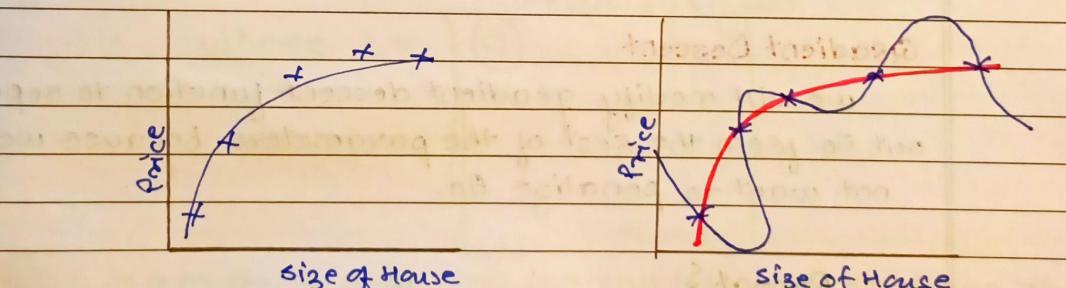
we will

without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our cost function.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (\hat{h}_{\theta}(x_i) - y_i)^2 + 1000 \cdot \theta_3^2 + 1000 \cdot \theta_4^2$$

Now, in order for the cost function to get close to zero, we will have to reduce the values of θ_3 and θ_4 to near zero. This will in turn greatly reduce the values of $\theta_3 x^3$ and $\theta_4 x^4$ in our hypothesis function.

As a result, we see that new hypothesis looks like a quadratic function but fits the data better due to extra small items $\theta_3 x^3$ and $\theta_4 x^4$. (Pink graph)



Suppose we penalize and make θ_3, θ_4 really small.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (\hat{h}_{\theta}(x_i) - y_i)^2 + 1000 \theta_3^2 + 1000 \theta_4^2$$

Pink graph shows this complete equation. ↑

we could also regularize all of our theta parameters in a single summation as:

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2 + \lambda \sum_{j=1}^n \theta_j^2$$

The λ , or lambda, is the **Regularization parameter**. It determines how much the costs of our theta parameters are inflated.

using the above cost function with the extra summation, we can smooth the output of our hypothesis function to reduce overfitting.

If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting.

Regularized Linear Regression

Gradient Descent

we will modify gradient descent function to separate out θ_0 from the rest of the parameters because we do not want to penalize θ_0 .

Repeat {

$$\theta_0 := \theta_0 - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i) \cdot x_0^i$$

$$\theta_j := \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i) \cdot x_j^i \right) + \frac{\lambda \cdot \theta_j}{m} \right]$$

$$j \in \{1, 2, \dots, n\}$$

The term $\lambda \cdot \theta_j$ performs our regularization. With some

manipulation our update rule can also be represented as:

$$\theta_j := \theta_j \left(1 - \alpha \cdot \frac{\lambda}{m}\right) - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (h_\theta(x_i) - y_i) \cdot x_{j,i}$$

(\rightarrow always will be less than 1)

Intuitively we can see it as reducing the value of θ_j by some amount on every update.

se 2nd term is exactly same as it was before.

Normal Equation:-

To add in regularization, the equation is the same as our original, except that we add another term inside the parentheses:

$$\theta = (X^T X + \lambda \cdot I)^{-1} X^T y$$

where $I =$

$$\begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}$$

I is a matrix with 0 at the top left and 1's down the diagonal, with 0's everywhere else.

It should have dimension $(n+1) \times (n+1)$. Intuitively, this is the identity matrix (though we are not including x_0), multiplied with a single scalar no. λ .

If $m < n$, then $X^T X$ is non-invertible. However, when we add the term $\lambda \cdot I$, then $X^T X + \lambda \cdot I$ becomes invertible.

Regularized Logistic Regression

cost function:

cost function for logistic regression was:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [y_i \cdot \log(h_\theta(x^{(i)})) + (1-y_i) \cdot \log(1-h_\theta(x^{(i)}))]$$

we can regularize this equation by adding a term to the end:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y_i \cdot \log(h_\theta(x^{(i)})) + (1-y_i) \cdot \log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

means to explicitly exclude the bias term, θ_0 . i.e. the θ vector is indexed from 0 to n (holding n+1 values, θ_0 through θ_n), and this sum explicitly skips θ_0 , by summing from 1 to n, skipping 0. Thus, when computing the equation, we should continuously update the two following equations.

Gradient Descent

repeat {

$$\theta_0 := \theta_0 - \alpha \cdot \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

$\frac{\partial}{\partial \theta_j} J(\theta)$ $j = (1, 2, \dots, n)$

Support Vector Machine

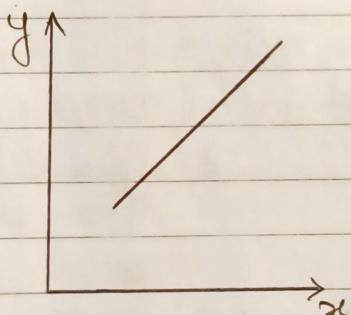
(SVM)

It can solve both classification and Regression problem.

1. classification → SVC (Support Vector classifier)

2. Regression → SVR (Support Vector Regressor)

some basics:



Equation of line:

$$y = mx + c \quad \text{OR}$$

$$y = \beta_0 + \beta_1 x \quad \text{OR}$$

$$ax + by + c = 0$$

$$\therefore y = \left(\frac{-a}{b} \right) x - \frac{c}{b}$$

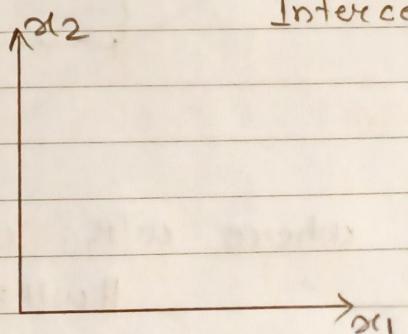
Coefficient Intercept

$$ax_1 + bx_2 + c = 0$$

$$w_1x_1 + w_2x_2 + b = 0$$

$$\therefore w^T x + b = 0$$

If line passes through origin $\therefore \underline{\underline{w^T x = 0}}$



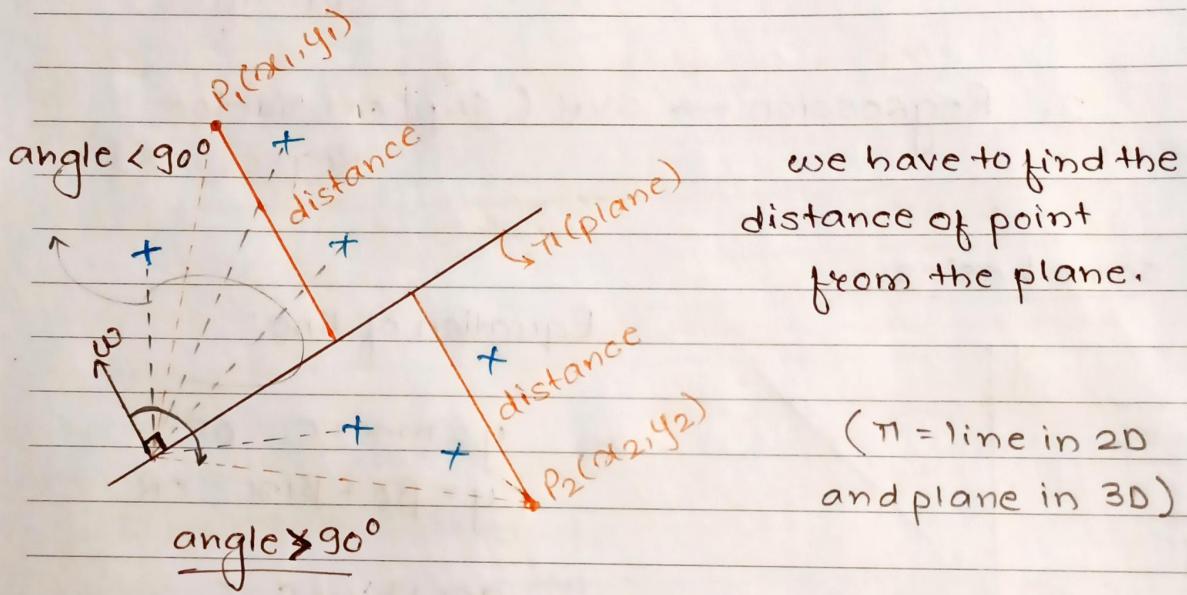
matrix multiplication:

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$$w^T x + b = 0 \quad (w^T x: w \text{ Transpose } x)$$

Equation of line passing through origin is:

$$\underline{w^T x = 0}$$



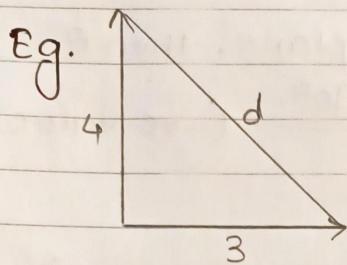
Distance of a point to the plane,

$$\text{distance}(d) = \frac{w^T p_i}{\|w\|}$$

$$= \|w\| \cdot \|p\| \cdot \cos \theta$$

where $w^T p_i$: w Transpose P_i , w : vector,
 $\|w\|$: magnitude of w .

unit vectors: A vector which has a magnitude of 1 is basically called unit vector.



Now,

$$d = \sqrt{3^2 + 4^2} \\ = \sqrt{25}$$

$$\therefore d = 5$$

where vector, $\hat{d} = \frac{d}{||d||}$

$||d||$ → magnitude

$$(3_{15}, 4_{15}) = d = \sqrt{(3_{15})^2 + (4_{15})^2} = \sqrt{25/25} \\ = 1.$$

∴ unit vector is a way to get focused on direction not on magnitude.

upward vector →

$$d = \frac{\omega \cdot P_1}{||\omega||}$$

$$d = ||\omega|| \cdot ||P_1|| \cdot \cos\theta$$

point above the plane

as $\theta < 90^\circ$

$\cos\theta$ will always be +ve.

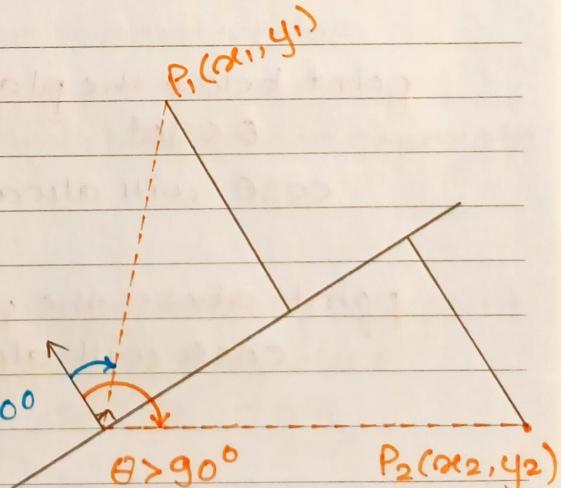
$\theta < 90^\circ$

$\theta > 90^\circ$

$P_2(x_2, y_2)$

point below the plane, as $\theta > 90^\circ$

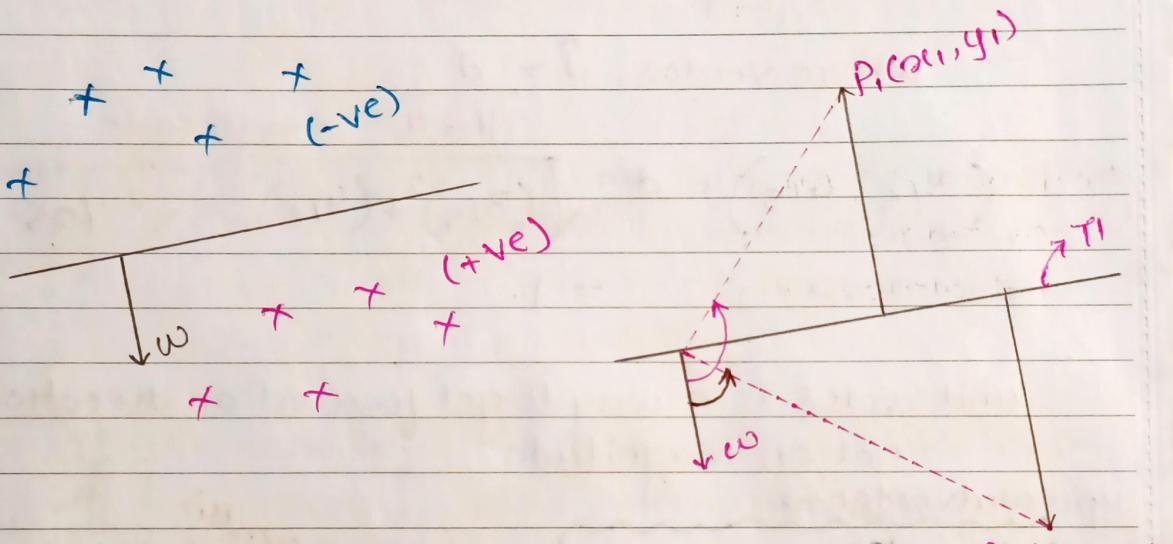
$\cos\theta$ will always be -ve.



• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

- If any point falling above the plane, then θ must be less than 90° (+value)
- If any point falling below the plane, then θ must be greater than 90° . (-ve value)

Downward vector!



point below the plane,
 $\theta < 90^\circ$

$\cos \theta$ will always be '+ve'.

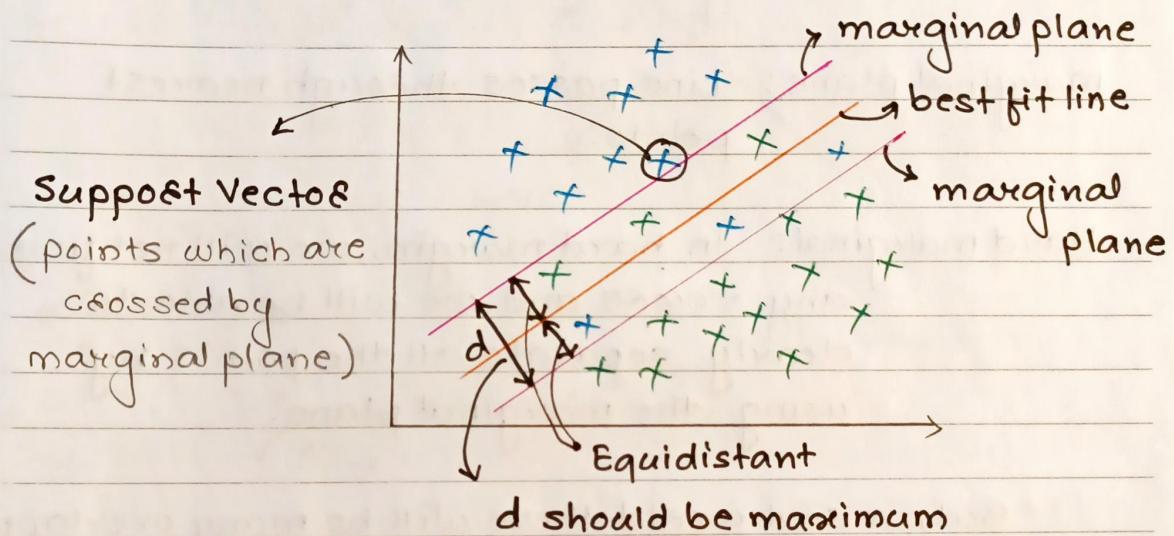
point above the plane, $\theta > 90^\circ$

$\cos \theta$ will always be '-ve'.

• Download Machine Learning:
<https://t.me/AIMLDeepThought>

Geometric Intuition Behind Support Vector Machine

Support Vector classifier (SVC)

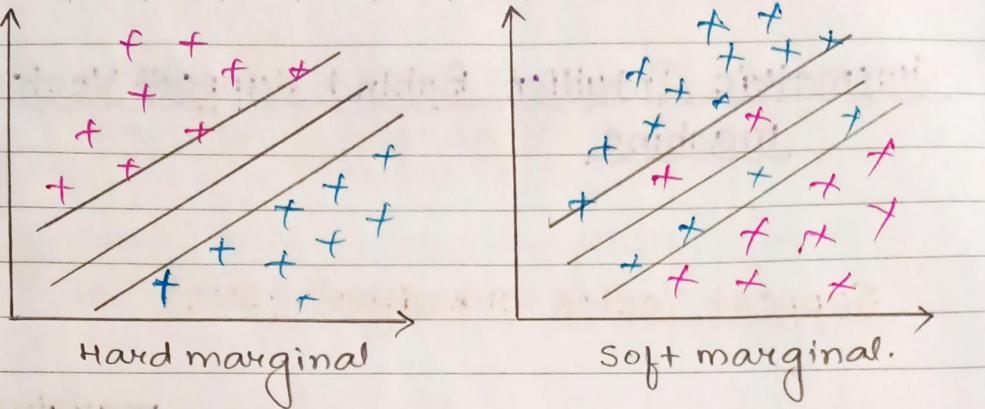


There will be many possible hyperplanes that separate different classes.

We have learnt in LR that the probability of a point belonging to any class given at very close to the hyperplane will be close to 0.5.

So, we want a hyperplane that separates (+ve) pts and (-ve) pts as far away as possible.

Key idea of SVM: such hyperplane is called margin-maximizing plane.



Marginal plane: Line passes through nearest points.

Hard marginal: In hard margin, we will not find any errors and we will be able to clearly separate all the points by using the marginal plane.

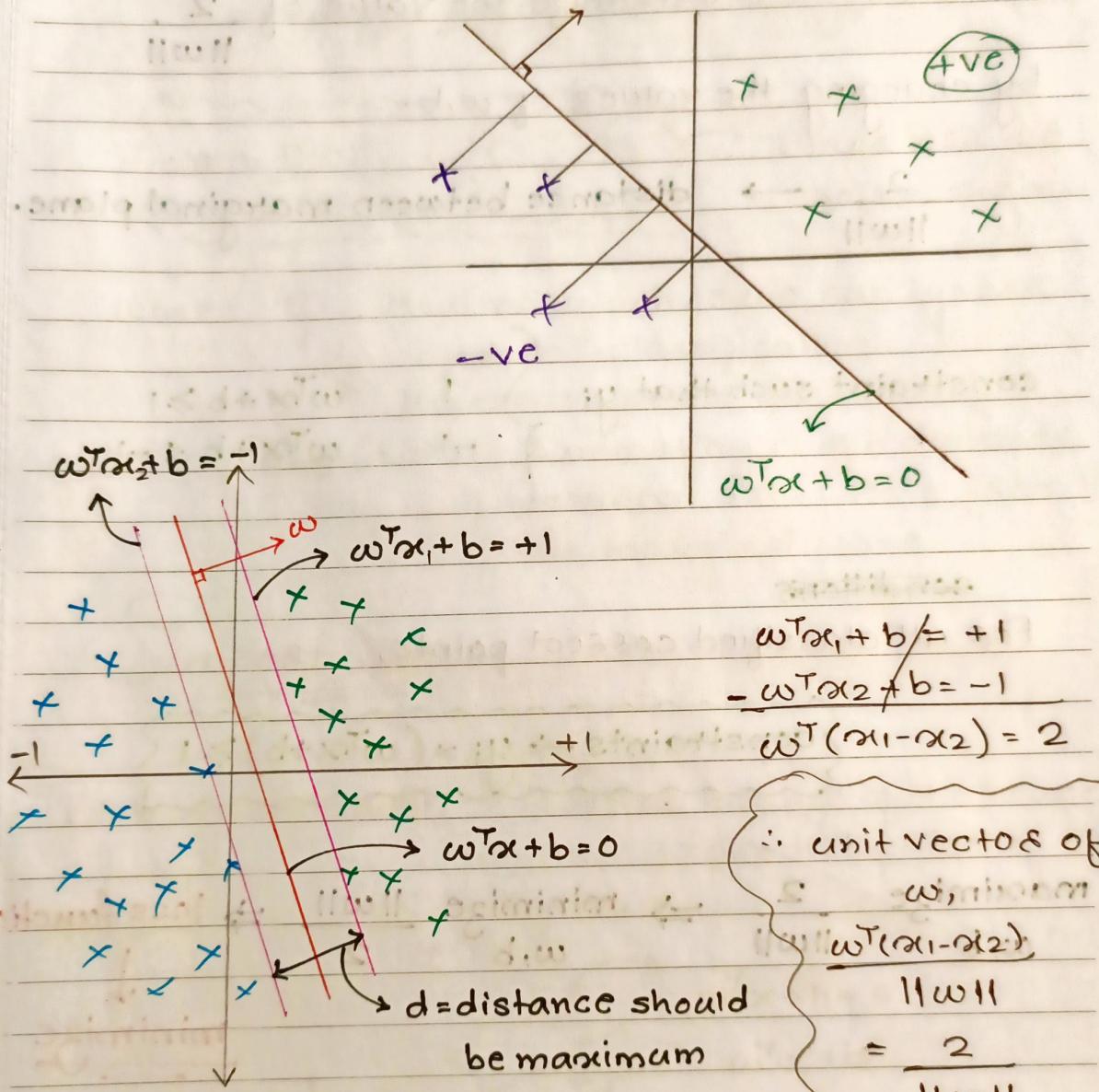
But, in real world there will be many overlapping with many errors, so marginal plane lines will be called soft margin.

→ marginal plane should be equidistance from best fit line.

• Download Machine Learning:
<https://t.me/AlMDeepThought>

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

SVM Mathematical Intuition:



$$\therefore \text{distance } (d) = \frac{w^T(x_1 - x_2)}{\|w\|} = \frac{2}{\|w\|}$$

Cost Function:

we have to maximize the value of $\frac{2}{\|\omega\|}$

by changing the values of w, b .

$\frac{2}{\|\omega\|} \rightarrow$ distance between marginal plane.

constraint such that y_i

$$\left\{ \begin{array}{l} 1 \quad w^T x + b \geq 1 \\ -1 \quad w^T x + b \leq -1 \end{array} \right.$$

conditions

For all classified correct points,

$$\{ \text{constraints} \rightarrow y_i \times (w^T x + b) \geq 1 \}$$

maximize $w, b \quad \frac{2}{\|\omega\|} \Rightarrow$ minimize $w, b \quad \frac{\|\omega\|}{2} \Rightarrow$ loss function

\downarrow
minimize

* loss function focus on minimization.

• Join me on LinkedIn for the latest updates on ML:
<https://www.linkedin.com/groups/7436898/>

Cost function:

minimize $\frac{\|w\|}{2}$ by changing w, b .

$$\min \frac{\|w\|}{2} + C_i \sum_{i=1}^n \xi_i$$

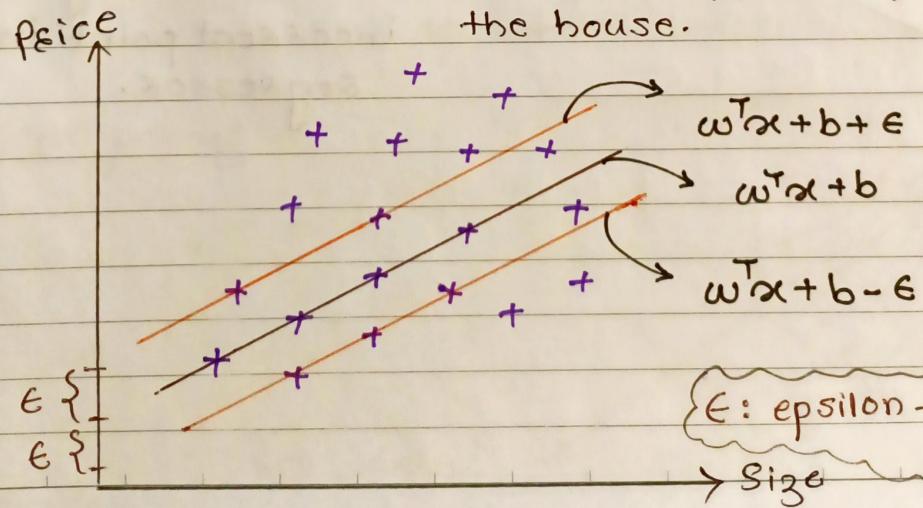
Hinge loss for soft margin.

where, C_i : How many points we can ignore
for mis-classification.
 ξ_i (Eta): Hyperparameter

ξ_i (Eta): Summation of the distance
of incorrect data points from
the marginal plane.

Support Vector Regression:

Problem Statement: Based on the size of the house, we have to predict price of the house.



ϵ : epsilon → marginal error

cost Function:

$$\text{minimize}_{w,b} \frac{\|w\|}{2} + \left\{ C_i \sum_{i=1}^n \epsilon_i \right\} \rightarrow \text{Hinge loss}$$

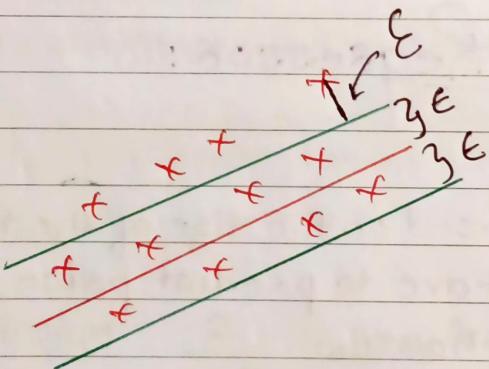
MAE

constraint: $|y_i - w^T x_i| \leq \epsilon + \epsilon_i$

Truth point predicted point epsilon

ϵ : margin of error (to decide original plane)

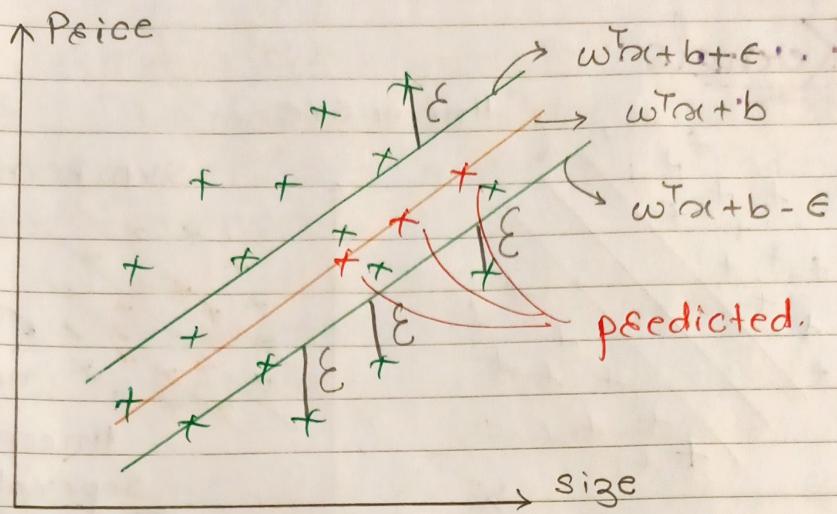
ϵ_i : errors above the margin.



Hyperparameter:

- keep adjusting ϵ to get best margin
- we can't say incorrect point in regressions.

• Download Machine Learning:
<https://t.me/AIMLDeepThought>



In Regression, no complete incorrect value, because it will be continuous value.

Q.

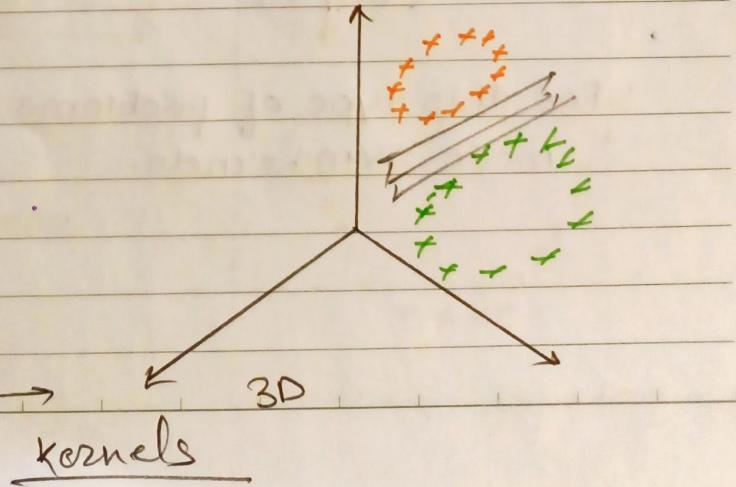
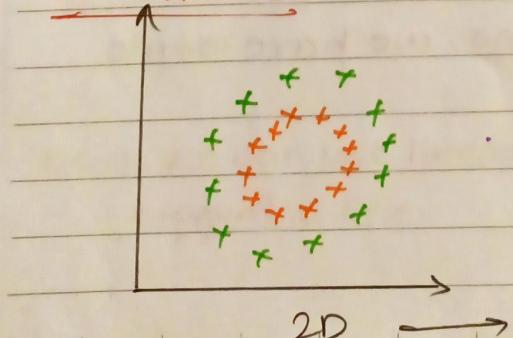
Is SVM impacted by the outliers?

Yes, SVM is impacted by the outliers.

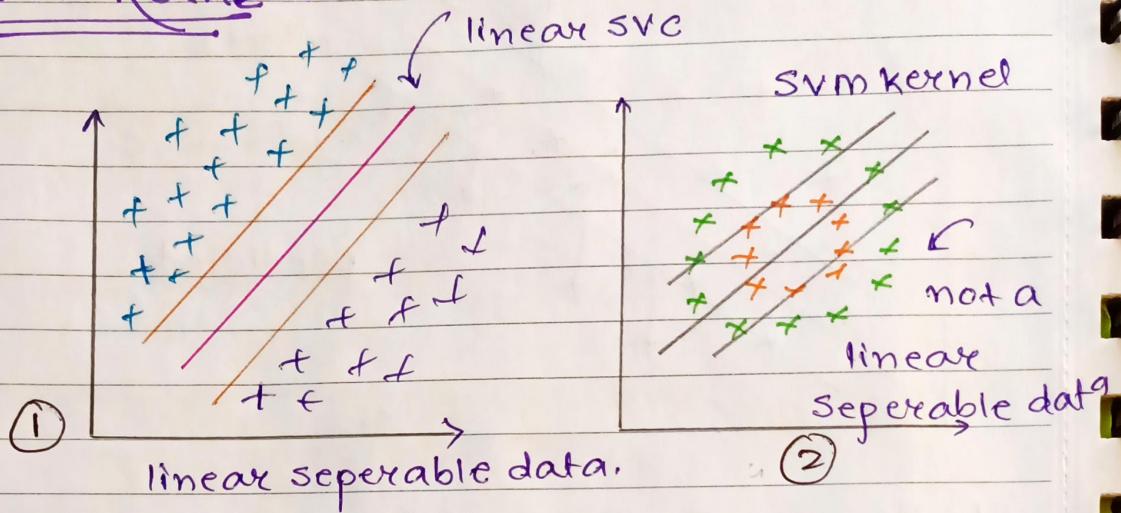
Does Standardization is need in SVM?

Yes, we need to perform Normalization and standardization.

SVM kernel:



SVM Kernels:



- when we create this (1) type of best fit line and marginal plane, we are actually solving the linear separable data.
- → called as Linear SVC. (Fig 1)
- If data is not a linear separable data, you will not be able to create best fit line and not able to create a marginal plane even though we create it, the accuracy will be very low. (Fig 2)
- For this type of problems, we have some more SVM kernels.

• Download Machine Learning:
<https://t.me/AIMLDeepThaught>

what does SVM kernels do?

→ The main aim is to apply some transformation technique. (some mathematical formula) on the dataset.

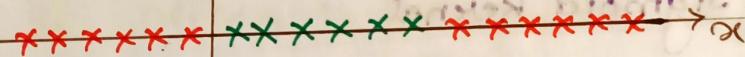
This transformation increases the dimension of the data.

SVM kernels → Transformation → Increasing the dimension of the data

(mathematical formula)

linearly separable line

→ will give error

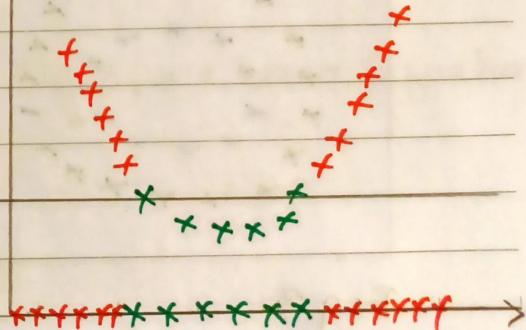


∴ we will transform the data from 1D \rightarrow 2D.

$$y = x^2$$

After \longrightarrow

Now, we can use linear separable line.



$$y = x^2$$

so if $x = -7 \quad y = 49$
 $x = -3 \quad y = 9$ and so on.

what is the advantage of doing this transformation?

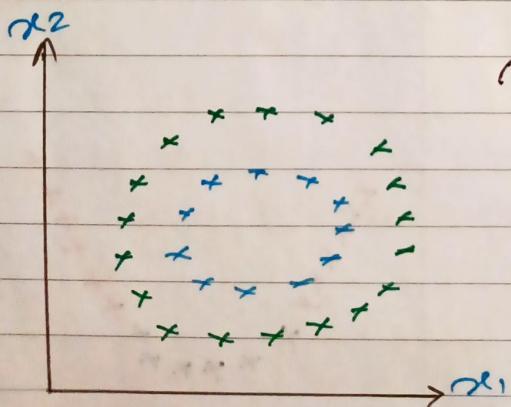
→ after transformation, we can apply linear SVM or SVC.

* when we divide convert $1D \rightarrow 2D$ then we can divide all the points using single line which is called Linear SVC.

Types of SVM Kernel:

1. Polynomial kernel
2. RBF kernel
3. Sigmoid kernel

1. Polynomial kernel



not separable with best fit line.

so, we need to convert $2D \rightarrow 3D$.

Fig 3.

Our main aim was to increase dimension,

$$2D \rightarrow 3D$$

so, hyperplane is created.

Formula for Polynomial kernel:

$$\{ f(x_1, x_2) = (x_1^T x_2 + 1)^d \}$$

Fig(4)

$d = \text{dimension}$

If we are converting $2D \rightarrow 3D$, the value of $d=3$.

$$\therefore x_1^T x_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1^2 & x_1 \cdot x_2 \\ x_1 \cdot x_2 & x_2^2 \end{bmatrix}$$

3 unique values: $x_1^2, x_1 \cdot x_2, x_2^2$

Now, initially at the time of Fig 3, we have 3 features

$x_1 \ x_2 \ y$

• Join me on LinkedIn for the latest updates on Machine Learning: <https://www.linkedin.com/groups/7436898/>

Now, after the transformation / formula of polynomial kernel we have 6 features

$$x_1 \ x_2 \ \{x_1^2 \ x_1 \cdot x_2 \ x_2^2\} y$$

∴ These new features can be plotted as the 3D in fig 4., which means that

In fig 4, x_1 will be x_1^2

x_2 will be x_2^2

z will be $x_1 \cdot x_2$

and once we have all these points, we will be able to clearly separate the points.

→ use polynomial kernel, to get better accuracy.

(2) Radial Basis Function Kernel (RBF Kernel)

$$k(\vec{x}, \vec{x}_i) = e^{-\frac{\|\vec{x} - \vec{x}_i\|}{2\sigma^2}}$$

hyperparameter

(3) Sigmoid Kernel

It can be used as the proxy for neural networks.

$$k(x, x_i) = \tanh(6x^T x_i + \gamma)$$

IN-DEPTH SVM

Mathematical formulation of SVM.



Objective: To find a hyper-plane that does margin maximization.

we need to find a hyperplane
for margin-maximization

$\Pi_1: \mathbf{w}^T \mathbf{x} + b = 0$ (w not necessarily unit vector).

$$\Pi_+: \mathbf{w}^T \mathbf{x} + b = 1$$

$$\Pi_-: \mathbf{w}^T \mathbf{x} + b = -1$$

By simple coordinate concepts we can get

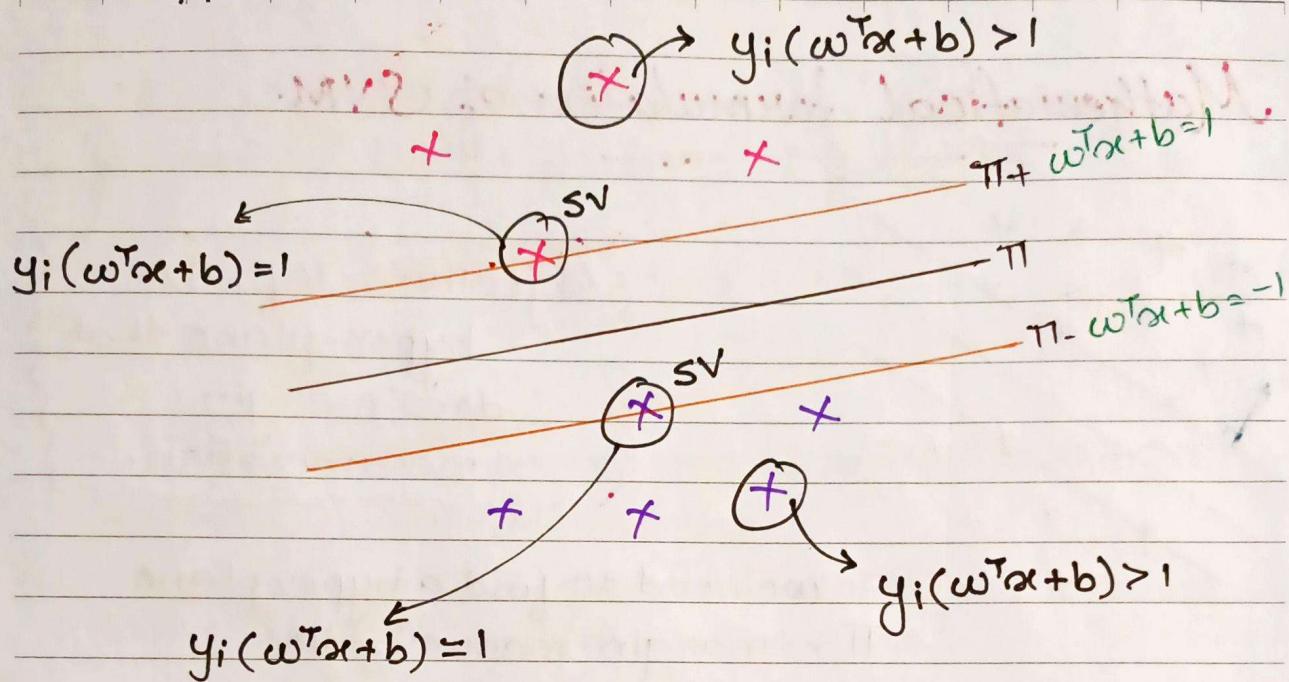
$$d = \frac{2}{\|\mathbf{w}\|}$$

So, we have to maximize 'd' with the constraint that all (+ve) points will lie above Π_+ & all (-ve) pts below Π_- .

So, the optimization function becomes

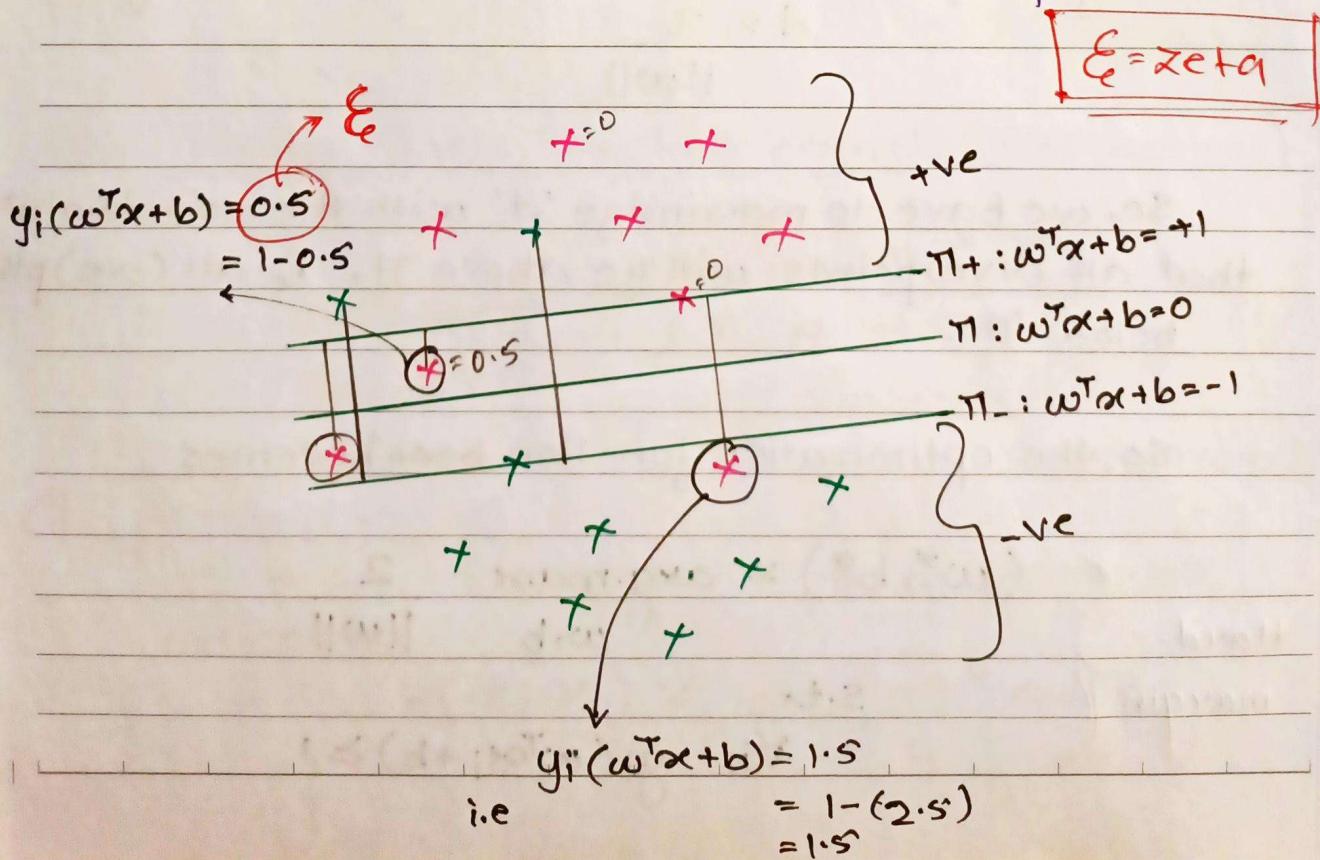
$$\left. \begin{array}{l} (\mathbf{w}^*, b^*) = \arg \max_{\mathbf{w}, b} \frac{2}{\|\mathbf{w}\|} \\ \text{Hard margin} \end{array} \right\} \quad \text{s.t.} \quad \forall i, y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

SV: support vector



But what if we have a dataset that's not linearly separable but almost linearly separable?

→ we formulate an alternative soft-margin constraint that allows some errors to persist.



$$x_i \rightarrow \xi_i$$

$$\xi = \zeta$$

$$\xi_i = 0 \quad \text{if } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

(i.e. for correctly classified points)

$\xi_i > 0$ and it is equal to some unit of distn away from the correct hyperplane in the incorrect direction.

Now, we want to minimize these errors.

so, our optimization function becomes

$$\underset{\substack{\mathbf{w}, b \\ \text{margin}}}{\arg \min} \frac{\|\mathbf{w}\|}{2} + C \cdot \frac{1}{n} \sum_{i=1}^n \xi_i$$

avg. distane of misclassified points.
(Hinge loss)

s.t.

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

$C \rightarrow$ hyperparameter

$C \uparrow \rightarrow$ tendency to make mistakes \downarrow
 \rightarrow overfit (High variance)

$C \downarrow \rightarrow$ underfit (High Bias)

y_i } correctly classified points
 $\xi_i = 0$

incorrectly classified pts $\xi_i > 0$

maximizing $\frac{2}{\|\omega\|}$ same as minimizing $\frac{\|\omega\|}{2}$

soft margin of SVM:

$$(\omega^*, b^*) = \arg \min_{\omega, b} \frac{\|\omega\|}{2} + C \cdot \frac{1}{n} \sum_{i=1}^n \xi_i$$

s.t.

$$y_i(\omega^T x_i + b^*) \geq 1 - \xi_i \quad \forall i \quad \begin{cases} \text{const.} \\ \xi_i \geq 0 \end{cases}$$

away from the correct
hyperplane in the incorrect
direction.

Now, we want to minimize these excesses.

so, our optimization function becomes

$$\arg \min_{\omega, b} \frac{\|\omega\|}{2} + C \cdot \frac{1}{n} \sum_{i=1}^n \xi_i$$

margin

avg. distance of misclassified points.
(Hinge loss)

s.t.

$$y_i(\omega^T x_i + b) \geq 1 - \xi_i$$

$C \leftrightarrow$ hyperparameter

$$\xi_i \geq 0$$

$C \uparrow \rightarrow$ tendency to make mistakes \downarrow

\rightarrow overfit (High variance)

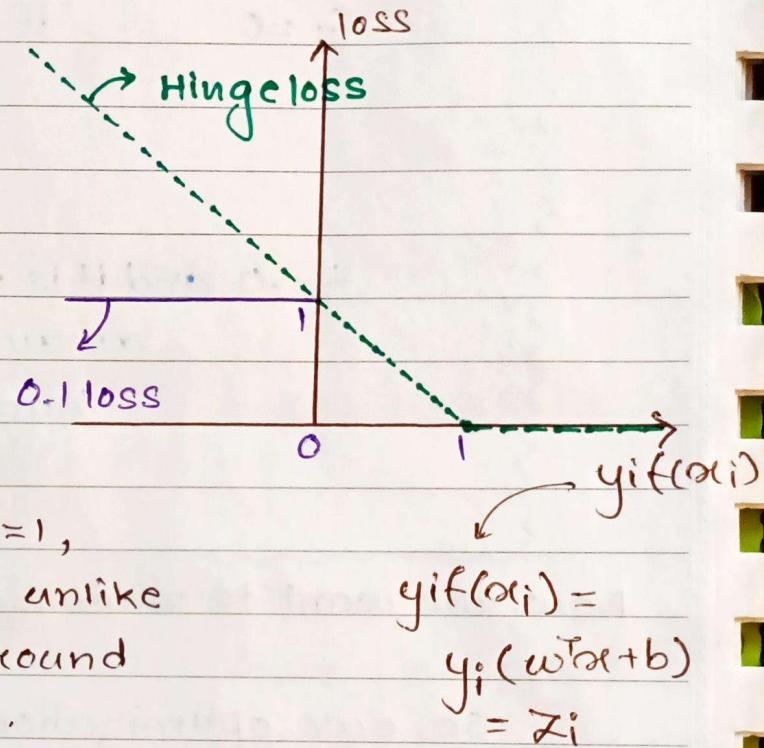
y_i correctly classified points
 $\xi_i = 0$

$C \uparrow \rightarrow$ underfit (High Bias)

incorrectly classified pts $\xi_i > 0$

Loss minimization : Hinge Loss

Hinge loss + Regularization
gives us
SVM.



$x_i > 0$: x_i is correctly classified

$x_i < 0$: x_i is incorrectly classified.

$$\text{Hinge loss} = \begin{cases} x_i \geq 1 ; \text{ Hinge loss} = 0 \\ x_i < 1 ; \text{ Hinge loss} = 1 - x_i \end{cases}$$

alternatively,

$$\text{Hinge loss} = \max(0, 1 - x_i)$$

- Andrew Ng

Lagrange Duality

consider a problem of the following form:

$$\min_{\omega} f(\omega)$$

$$\text{s.t. } b_i(\omega) = 0, i=1, \dots, l.$$

we define the Lagrangian to be

$$L(\omega, \beta) = f(\omega) + \sum_{i=1}^l \beta_i b_i(\omega)$$

Here, β_i 's = Lagrange multipliers.

To find, set L 's partial derivatives to 0;

$$\frac{\partial L}{\partial \omega_i} = 0; \quad \frac{\partial L}{\partial \beta_i} = 0,$$

and solve for ω and β .

consider the following, called primal optimization problem:

$$\min_{\omega} f(\omega)$$

$$\text{s.t. } g_i(\omega) \leq 0, i=1, \dots, k$$

$$b_i(\omega) \leq 0, i=1, \dots, l.$$

To solve it, we start by defining ..

generalized lagrangian.

$$L(\omega, \alpha, \beta) = f(\omega) + \sum_{i=1}^k \alpha_i g_i(\omega) + \sum_{i=1}^l \beta_i h_i(\omega)$$

Here, the α_i 's and β_i 's are the Langrange multipliers.

Consider the quantity,

$$\theta_p(\omega) = \max_{\alpha, \beta: \alpha_i \geq 0} L(\omega, \alpha, \beta)$$

Here, p = "primal".

Let some ω be given. If ω violates any of the primal constraints constraints

(i.e. if either $g_i(\omega) > 0$ or $h_i(\omega) \neq 0$ for some i)

then,

$$\begin{aligned} \theta_p(\omega) &= \max_{\alpha, \beta: \alpha_i \geq 0} f(\omega) + \sum_{i=1}^k \alpha_i g_i(\omega) \\ &\quad + \sum_{i=1}^l \beta_i h_i(\omega) \\ &= \infty \end{aligned}$$

①

conversely, if the constraints are indeed satisfied for a particular value of w ,

then $\theta_p(w) = f(w)$. Hence

$$\theta_p(w) = \begin{cases} f(w) & \text{if } w \text{ satisfies} \\ & \text{primal constraints} \\ \infty & \text{otherwise} \end{cases}$$

Hence, if we consider the minimization problem

$$\min_w \theta_p(w) = \min_w \max_{\alpha, \beta; \alpha_i \geq 0} L(w, \alpha, \beta)$$

same as, primal problem.

Optimal value of the objective to be $p^* = \min_w \theta_p(w)$

value of the primal problem.

Now, we define

$$\theta_D(\alpha, \beta) = \min_w L(w, \alpha, \beta)$$

where, D = "dual"

Note: In defn θ_p we were optimizing (maximizing) w.r.t α, β here we are minimizing w.r.t. w .

• Join me on LinkedIn for the latest updates on Machine Learning: <https://www.linkedin.com/groups/7436898/>

∴ Dual optimization problem:

$$\max_{\alpha, \beta: \alpha_i \geq 0} \Theta_D(\alpha, \beta) = \max_{\alpha, \beta: \alpha_i \geq 0} \min_{\omega} L(\omega, \alpha, \beta)$$

Optimal value of the dual problem's objective:

$$d^* = \max_{\alpha, \beta: \alpha_i \geq 0} \Theta_D(\omega)$$

How are the primal and the dual problems related?

$$d^* = \max_{\alpha, \beta: \alpha_i \geq 0} \min_{\omega} L(\omega, \alpha, \beta) \leq \min_{\omega} \max_{\alpha, \beta: \alpha_i \geq 0} L(\omega, \alpha, \beta) = p^*$$

always remember:

"min max min" of a function always being less than or equal to the "min max".

for certain conditions,

$$d^* = p^*$$

i.e. there exists a_i, b_i , so that
 $h_i(\omega) = a_i \omega + b_i$

affine \rightarrow linear, except contains
extra intercept term b_i .

Suppose f and the g_i 's are convex; and the b_i 's
are affine.

Suppose that the constraints g_i are
feasible; means there exists some ω so that

$$\underline{g_i(\omega) < 0 \text{ for all } i.}$$

under above assumptions, there must exist
 $\omega^*, \alpha^*, \beta^*$ so that ω^* is the solution to the
primal problem α^*, β^* are the solution to the
dual problem, and moreover $p^* = d^* = L(\omega^*,$
 $\alpha^*, \beta^*)$

moreover, ω^*, α^* and β^* satisfy the
Karush-Kuhn-Tucker (KKT) conditions

$$\frac{\partial}{\partial \omega_i} L(\omega^*, \alpha^*, \beta^*) = 0, \quad i=1, \dots, d \quad \rightarrow ③$$

$$\frac{\partial}{\partial \beta_i} L(\omega^*, \alpha^*, \beta^*) = 0, \quad i=1, \dots, l \quad \rightarrow ④$$

$$\alpha_i^* g_i(\omega^*) = 0, \quad i=1, \dots, k \quad ⑤$$

$$g_i(\omega^*) \leq 0, \quad i=1, \dots, k \quad ⑥$$

$$\alpha^* \geq 0, \quad i=1, \dots, k$$

$\rightarrow ⑦$

moreover, if some $\omega^*, \alpha^*, \beta^*$ satisfy the KKT conditions,
then it is also a solution to the primal and dual
problems.

Eqn ⑤, which is called the KKT dual complementarity condition.

Specifically, it implies that if $\alpha_i^* > 0$, then $g_i(\omega^*) = 0$.

i.e. " $g_i(\omega) \leq 0$ " constraint is active, meaning it holds with equality rather than with inequality.

This will be key for showing that SVM has only a small number of "support vectors".

Dual Form of SVM

Following (primal) optimization problem for finding the optimal margin classifier:

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

s.t.

$$y_i(w^T x_i + b) \geq 1, i=1, \dots, n$$

—⑧

constraints as

$$g_i(w) = -y_i(w^T x_i + b) + 1 \leq 0$$

when we construct Langrangian for optimization problem we have:

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

—⑨

Note: there's only " α_i " but no " β_i " Langrange multipliers, since the problem has only inequality constraints.

• Join me on LinkedIn for the latest updates on Machine Learning: <https://www.linkedin.com/groups/7436898/>

To find the dual form, first minimize $\mathcal{L}(\omega, b, \alpha)$ w.r.t ω and b (for fixed α) to get Θ_p .

$$\nabla_{\omega} \mathcal{L}(\omega, b, \alpha) = \omega - \sum_{i=1}^n \alpha_i y_i x_i = 0$$

$$\therefore \left[\omega = \sum_{i=1}^n \alpha_i y_i x_i \right] \quad \text{--- (10)}$$

derivative w.r.t b ,

$$\frac{\partial}{\partial b} \mathcal{L}(\omega, b, \alpha) = \sum_{i=1}^n \alpha_i y_i = 0 \quad \text{--- (11)}$$

If we take the defn of ω in eqn (10) and put it into the Lagrangian eqn (9), we get

$$\begin{aligned} \mathcal{L}(\omega, b, \alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j (x_i)^T x_j \\ &\quad - b \sum_{i=1}^n \alpha_i y_i \end{aligned}$$

but from eqn (11), last term = 0

$$\therefore \mathcal{L}(\omega, b, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j x_i^T x_j$$

∴ with the constraint $\alpha_i \geq 0$, we obtain

dual optimization problem

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

$$\text{s.t. } \alpha_i \geq 0, i=1, \dots, n$$

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

Important Observations:

1. for every x_i , we have an α_i

2. α_i 's only occur in the form of $\underline{\alpha_i^T x_q}$

on solving we get

$$f(x_q) = \sum_{i=1}^n \alpha_i y_i \underline{\alpha_i^T x_q} + b$$

for non-SV, $\alpha_i = 0$

so, for $f(q) \rightarrow$ only the support vectors matters.

3. $\alpha_i > 0$ only for support vectors else 0.

since x_i always occurs only as $x_i^T x_j \rightarrow x_i \cdot x_j$
cosine-sim(x_i, x_j)

so, basically cosine similarity value are
only required to solve the optimization
and ultimately get the model.

we can replace cosine-similarity with
any other kind of similarity which makes it
more powerful.

Generalized dual form -

$$\max_{\alpha_i} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \underbrace{k(x_i, x_j)}_{\text{called Kernel}}$$

function.

(can be any kind of similarity b/w
 x_i & x_j)

Even during evaluation x_i occurs only as $x_i^T x_q$

$$f(x_q) = \sum_{i=1}^n \alpha_i y_i \underbrace{x_i^T x_q + b}_{\rightarrow k(x_i, x_q)}$$

Kernel trick → Replacing $x_i^T x_j$ with generalized similarity function $k(x_i, x_j)$.

Quadratic Programming.

The hard margin and soft margin problems are both convex quadratic optimization problems with linear constraints.

Such problems are known as Quadratic Programming (QP) problems

• Download Machine Learning:
<https://t.me/AIMLDeepThaught>

• Join me on LinkedIn for the latest updates on Machine Learning: <https://www.linkedin.com/groups/7436898/>