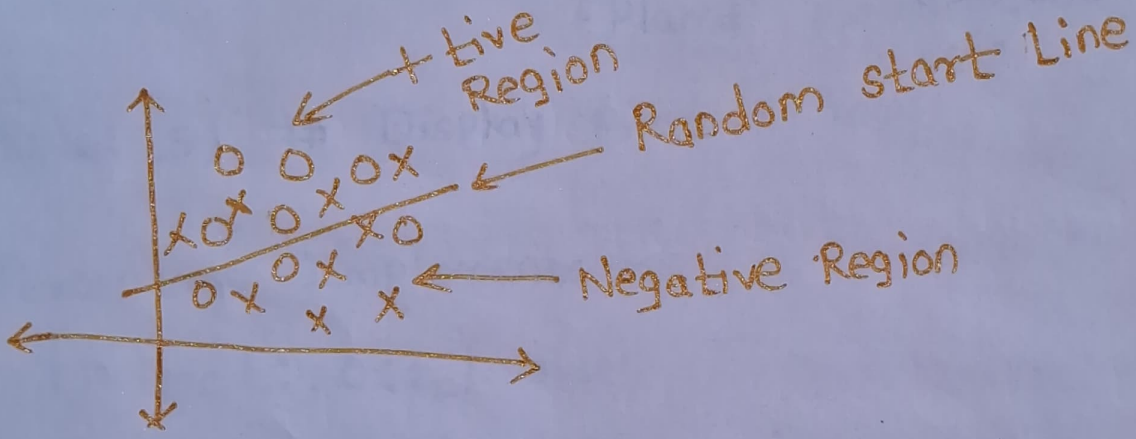
Perceptron Trick

—x—x—

* Perceptron Trick :-

We know that the 'Perceptron' tries to draw a line (decision boundary) to classify the data.

Perceptron Trick is the method to find the line that classifies the data.



Imagine we have data points on a graph that are linearly separable as shown in above fig.

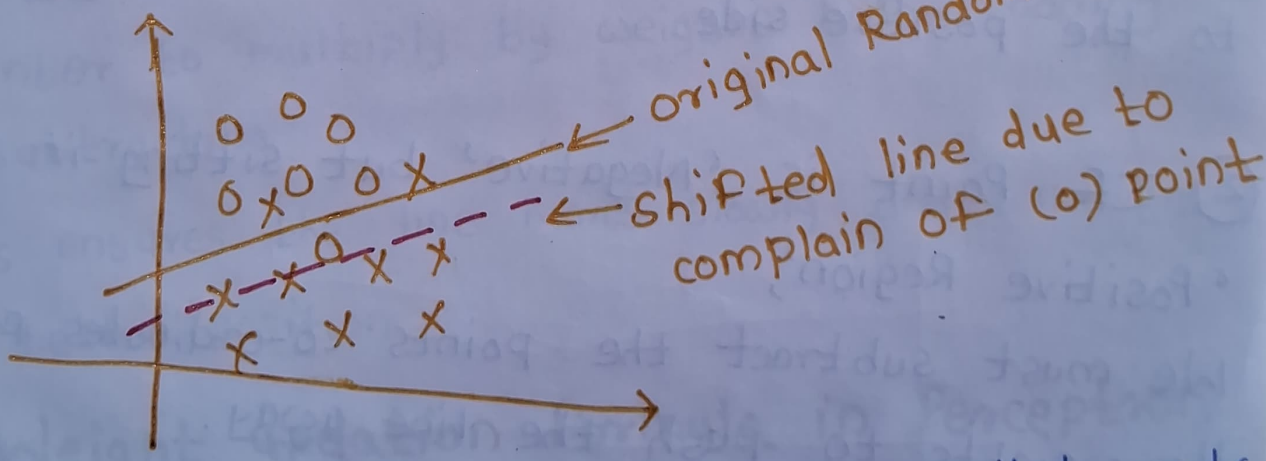
(stu. placed (o) & Not placed (x))

(1) The start :- We begin by drawing a random line that separates both regions.
As it is random the model performence is worst.

(2) We pick a random data point (student) and ask model to predict their status based on current line.

(i) Scenario 'A' (correct Prediction) - If the stude-nt is actually 'Placed' and model predicts 'Placed', the line is doing its job (Same for not Placed).
The point says "I don't have a problem' so the line does not moves.

(ii) Scenario 'B' (Wrong Prediction) - If the student is actually 'placed' but model predicts 'Not placed', the point effectively says,
"I have a problem, I~~e~~ am actually placed but your model is predicting that I am not placed"
To satisfy this point we adjust the weights in a way so this point get in "placed" region



(3) This happens for ~~every~~ ~~poi~~ the decided epochs (eg. 100) and everytime the poin is choosen randomly. or until line stops to move.

*Identification of +tive & -tive Regions:-

To understand how line moves define regions mathematically using eqn of line,

$$Ax + By + c = 0$$

This is eqn of random start line.

if $Ax + By + c > 0$ → Positive region

if $Ax + By + c < 0$ → Negative region.

*Transformation Rule:-

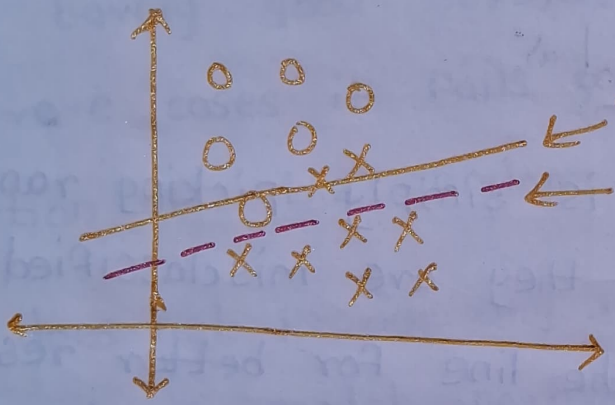
① If a point is 'positive' but sitting in 'Negative' region,
We must add the point's co-ordinates to the lines coeff. (weights) to pull the line closer to the positive side.

② If Point is 'Negative' but sitting in the 'Positive Region',
We must subtract the points co-ordinates from the weights to push the line away.

* Learning Rate & It's Role :-

(1) Learning Rate - It is the hyperparameter that controls how much a model's weights are adjusted during each step of training process.

(2) Role - If we move the line according to the selected missclassified point directly we may classify that point correctly but we will miss other



So we must define a small (generally 0.01, 0.1) number to multiply by weights before adding or subtracting.

This ensures the line moves slowly and converges to better position.

* Weight updation formula in Perceptron Trick :-

$$W_{new} = W_{old} + \eta * (y_i - \hat{y}_i) * x_i$$

Where,

$W = \sum_{i=0}^{n} W_i x_i \rightarrow$ Weights + bias term

η = Learning rate → eta

y_i = Actual values of target for ith row

$\hat{y}_i$ = Predicted value of target for ith row

x_i = ith input value

* Summary :-

"Perceptron Trick" is simply picking random points, checking if they are misclassified, and slighly moving the line for better results.