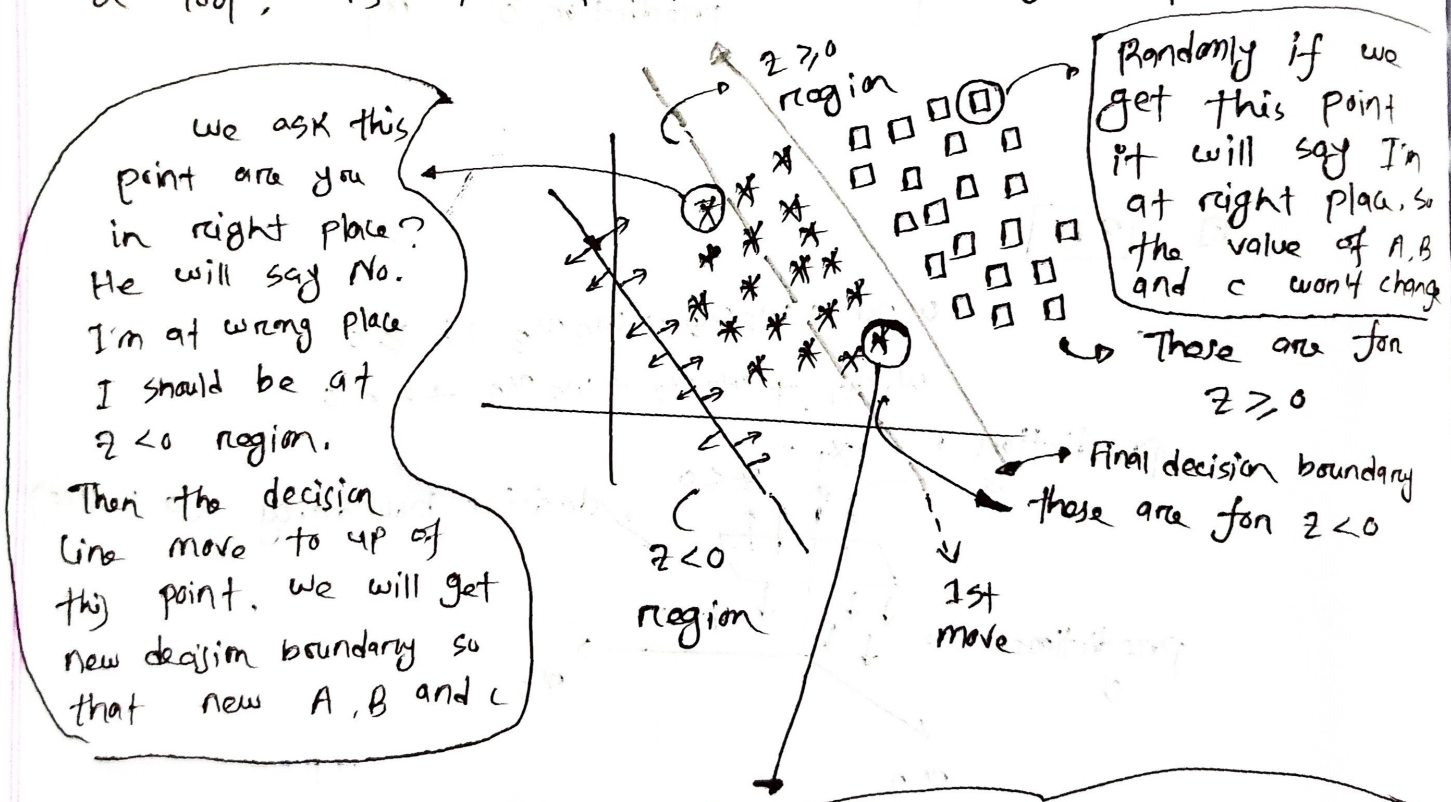
Day-06 : Perceptron Trick. How to train a perceptron.

Our purpose is to find the values of weights and bias such that the decision boundary fit at the proper place.

Initially we start taking the random values of weights like for $Ax_1 + Bx_2 + C = z$, $A = 1, B = 1$ and $C = 0$ then we ask every point by running a loop, is the point in the right place?



This is called epochs

After looping like 1000 or more times we will get (possibility) the best fitting decision boundary line. Consequently the values of A, B and C.

Randomly If we get this point it will say I'm at the wrong place and come up to me (decision line) then the line will be change so that A, B and C.

→ Or we can use the method of convergent for all the points

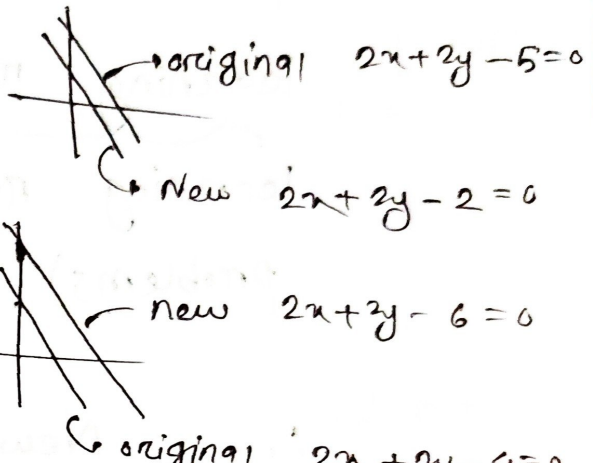
Use desmos.com for real experience

generally we are discussing on 2D

Transformation of decision boundary line is depend on the value of A, B and C

C For increasing c the new line parallele go to the lower position.

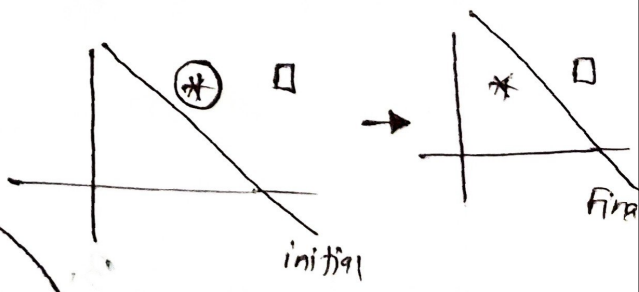
and for decreasing c the new line paralelle go to upper position



For A and B we know the fact.

Important: For moving the decision line we need the point which is in the wrong place. our goal is to move the decision boundary in such way such that the point gets the right region.

we can do this mathematically



new coefficient = old coefficient − coordinates

Here c = 1

But the perceptron doesn't do this large transformation at a time. It does the change/transformation of the line a little at a time

After keeping transformation like this way, finally it classify the misclassfied point.

This small changes happened fixing / multiplying the coordinates by a small values — called learning rate. (initially we fix the value of learning rate 0.01 or 0.001 depending on the problems):

$$\therefore \text{New coef} = \text{old coef} - \eta \cdot \text{coordinate of misclassified point}$$

(learning rate)

we can divide it in two parts:

(i)

i) Perceptron training loop:

1. Input a feature vector x (eg $[x_1, x_2]$)
2. predict using initial weights w and bias b,

$$y_{pred} = \text{sign}(w_1 x_1 + w_2 x_2 + b)$$

3. Compare prediction y_{pred} with actual y_{true} (Keep in mind label +1 or -1)
4. If $y_{pred} \neq y_{true}$ update the weights and bias

we can write $z = Ax + By + C$

$z = w_0x_0 + w_1x_1 + w_2x_2$ where $w_0 = b$ and $x_0 = 1$

$= \sum_{i=0}^{2} w_i x_i$ = [illegible] . $[w_0, w_1, w_2] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$. [+1 or -1]

$w_1 \text{ new} = w_1 \text{ old} + \eta \cdot \hat{y}_{true} \cdot x$

$w_2 \text{ new} = w_2 \text{ old} + \eta \cdot y_{true} \cdot x$

$b_{new} = b_{old} + \eta \cdot y_{true}$

যদি label [+1 or -1] না হয়ে [+1 or 0] হয় তাহলে আমাদের loop এর মধ্যে if condition চালাতে হবে।

অথবা :

(ii)

1. Input a feature vector x_i and bias
2. predict with initial weights $w = [w_1, w_2, b]$

$$y_{pred} = \sum w_i x_i$$

3. If $y_{pred} \geq 0$ and $x_i \in$ Negative region

$$w_{new} = w_{old} - \eta x_i$$

If $y_{pred} < 0$ and $x_i \in$ positive region

$$w_{new} = w_{old} + \eta \cdot x_i$$

এই loop ততক্ষণ পর্যন্ত চলবে যতক্ষণ না decision line এর best fit হয় (অর্থাৎ misclassification শেষ না হওয়া পর্যন্ত অথবা epochs শেষ না হওয়া পর্যন্ত).

* epochs শেষ? misclassification আছে. epochs বাড়িয়ে দাও

এতোকিছু না করে আমরা (ii) বাবের জন্যও অর্থাৎ যেখানে label [1 or 0] এর জন্যও প্রথম বাবের loop চালাতে পারি। সেক্ষেত্রে,

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

যদি $y_i = \hat{y}_i$ অর্থাৎ or $y_{actual} = y_{pred}$ হয় আমাদের কোনো দরকার নেই w কে আপডেট করার কারণ in this case,

$$w_{new} = w_{old}$$

কিন্তু যদি $y_i \neq \hat{y}_i$ and $y_i = 0$ but $\hat{y}_i = 1$

then $$w_{new} = w_{old} - \eta \cdot x_i$$

and if $y_i \neq \hat{y}_i$ but $\hat{y}_i = 0$ but $y_i = 1$ then

$$w_{new} = w_{old} + \eta \cdot x_i$$