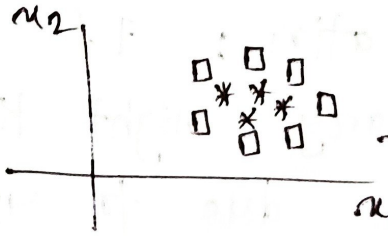
Day-07: Perceptron loss Function

Problem with perceptron Trick:

i) only works for linearly separable data

- If the data can't be separated by a straight line or plane or hyperplane the perceptron will never stop updating, it will never converge (we will not get fitted line for 2D data).

- Example: If points of class A and B are mixed like circle, perceptron fails



fails here, can't draw a line which can separate the two class

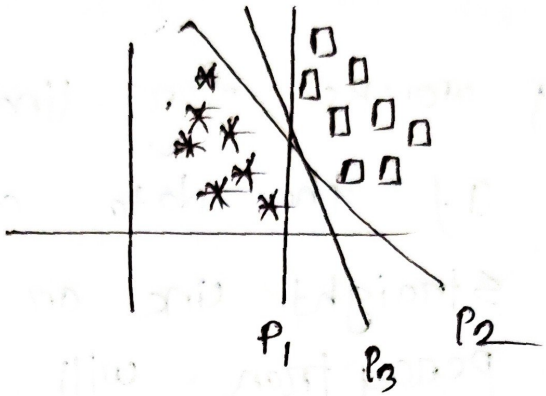
* It's also the problem of all types of perceptron not only for peceptron trick.

main problem

ii) No margin (minimum distance between decision/perceptron line and data) consideration.

- It just tries to get the classification right
- It doesn't care how far the points are

from the boundary - this can load to poor generalization.



Between P_1, P_2 and P_3 lines: which one is considered perceptron trick has no idea about this, it randomly choose one without considering margin. this creates problem.

iii) Oscillation: If the data is noisy, the boundary might keep shifting back and forth due to updates — it becomes unstable

Loss Function: A loss function is a function of parameters of weights and bias (in perceptron) which measure how wrong its predictions are.

Our goal is to minimize the functional value to get the better model.

There are a simple analogy: Imagine you're throwing darts at a bullseye

- Each throw lands somewhere on the board
- The distance from the centre is your "loss"
- You adjust your aim (model weights and bias) to reduce that distance (loss) over time.

Common loss function

i) MSE — Linear Regression/ Regression task — $(\hat{y}-y)^2$

ii) MAE — Regression line task — $\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$

iii) Hinge loss — SVM — $\max(0, 1 - y_i f(x_i))$

iv) Perceptron loss — perceptron — $\max(0, y_i f(x_i))$

v) logistic loss (log loss) — logistic regression — $\log(1+e^{-yf(x)})$

- Here y_i → true label (+1 or -1), and $f(x_i)$ is model's raw score.
  - $f(x_i)$ → $w_1x_1 + w_2x_2 + b = 0$ for 2D in perceptron

Perceptron loss function:

average error ← $E = \frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i))$

where E is the function of weights and bias → $E(w, b)$

n = number of rows / samples

y_i = actual output

$f(x_i)$ = predicted form $\sum_{i=1}^{n} w_i x_i + b$

Our goal is find those values of weights and bias such that the value of E be minimum. i.e Error minimization

min → minumum/ lowest value

argmin → Those value of the parameters (w, b) which makes the function minimum

Mathematically

$$(w, b) = \text{argmin} \left(\frac{1}{n} \sum_{i=1}^{n} \max(0, -y_i f(x_i)) \right)$$

Explanation of loss function:

Let's consider two input with two rows

$x^{(1)}$	$x^{(2)}$	y_i
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2

$y_1, y_2 \in \{+1, -1\}$

$w = [w_1, w_2]$

and bias b

$$\therefore E = \frac{1}{2} \left[\max(0, -y_1 f(x_1)) + \max(0, -y_2 f(x_2)) \right]$$

where, $f(x_1) = w_1 x_{11} + w_2 x_{12} + b$

$f(x_2) = w_1 x_{21} + w_2 x_{22} + b$

$$E = \frac{1}{2} \left[\max(0, -y_1 (w_1 x_{11} + w_2 x_{12} + b)) + \max(0, -y_2 (w_1 x_{21} + w_2 x_{22} + b)) \right]$$

since for $f(x_i) \geq 0$ we have $+1$
and $f(x_i) < 0$ we have -1

Keep in mind: If $f(x_i) \geq 0$ and $y_i = 1$ we call prediction is correct and $\max(0, -y f(x_i)) = 0$

This means for true prediction the loss becomes zero similarly for wrong prediction loss provides a value

Goal: Find the value of weight that give the lowest loss i.e $\arg\min(E)$

Gradient Descent: Gradient descent is the step by step process to reduce the loss. It helps the model finding the best values (like weights) to reduce the loss, consequently make the better prediction.

How it works?

1. start with random values for weights and bia
2. calculate the loss
3. compute the gradient (direction to reduce loss)
4. update weights by taking a small step in the opposite direction of the gradient. similarly for bias.
5. Repeat the process until the loss is very small

Formula: $w = w - \eta \frac{\partial E}{\partial w}$, $b = b - \eta \frac{\partial E}{\partial b}$

where, $w \rightarrow$ weights

$\eta \rightarrow$ learning rate (very small 0.001)

$\frac{\partial E}{\partial w}$ = gradient (slope of loss w.r.t

On Gradient Descent there is a video → (ML-51) CampusX

For 2D

$$\frac{\partial E}{\partial w} = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}\right] \text{ and } \frac{\partial E}{\partial b} = \left[\frac{\partial E}{\partial b}\right]$$

calculation part

$$E = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y_i f(x_i))$$ > For 2D

where $f(x_i) = w_1 x_{i1} + w_2 x_{i2} + b$

Here, $\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial f(x_i)} \times \frac{\partial f(x_i)}{\partial w_1}$ [chain rule since $f(x_i)$ is a function of w_1]

$$\frac{\partial E}{\partial f(x_i)} = \begin{cases} 0 & \text{if } y_i f(x_i) \geq 0 \\ -y_i & \text{if } y_i f(x_i) < 0 \end{cases}$$

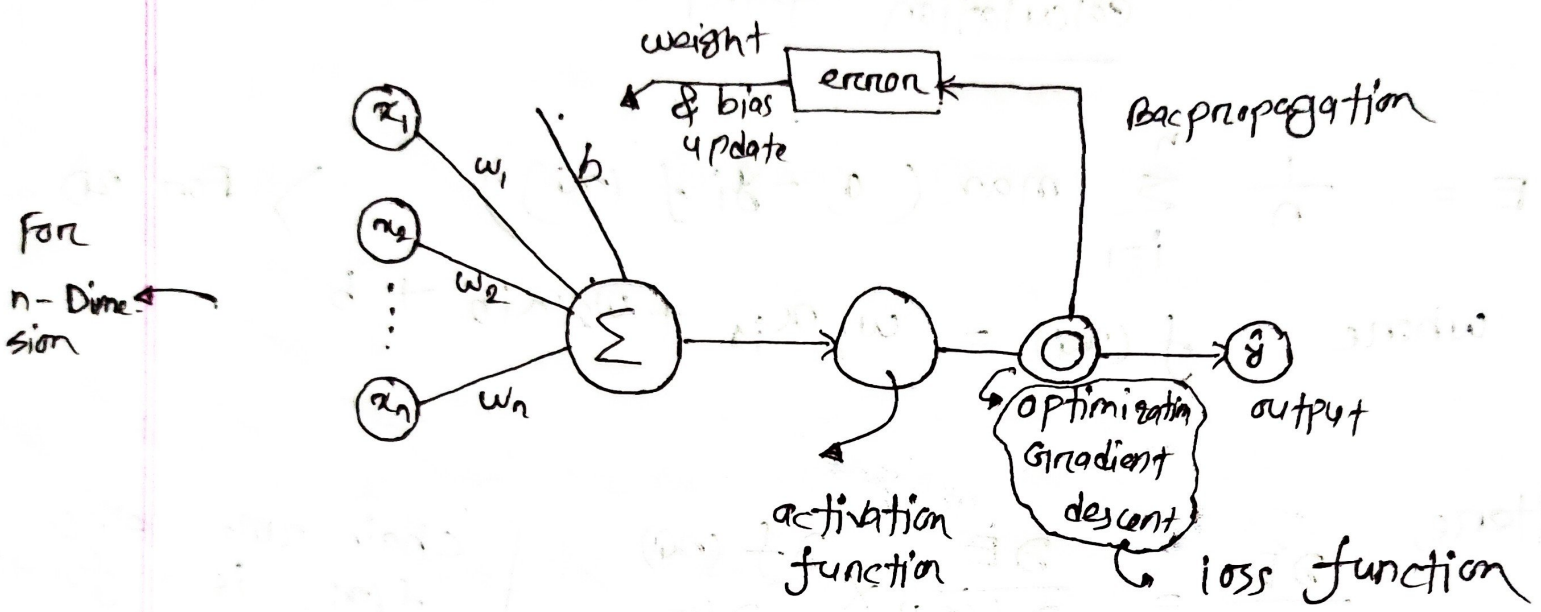
and $$\frac{\partial f(x_i)}{\partial w_1} = \frac{\partial (w_1 x_{i1} + w_2 x_{i2} + b)}{\partial w_1} = x_{i1}$$

$$\therefore \frac{\partial E}{\partial w_1} = \begin{cases} 0 & \text{if } y_i . f(x_i) \geq 0 \\ -y_i . x_{i1} & \text{if } y_i . f(x_i) < 0 \end{cases}$$

Similarly, $$\frac{\partial E}{\partial w_2} = \begin{cases} 0 & \text{if } y_i . f(x_i) \geq 0 \\ -y_i . x_{i2} & \text{if } y_i . f(x_i) < 0 \end{cases}$$

$$\frac{\partial L}{\partial b} = \begin{cases} 0 & \text{if } y_i . f(x_i) \geq 0 \\ -y_i & \text{if } y_i . f(x_i) < 0 \end{cases}$$

putting these values we will get updated weights and bias.



More on loss function: The above model of perceptron is a mathematical model

SGD - Stochastic Gradient Descent

All of the combination we can use in perceptron, everytime we have to use SGD

Activition function + loss function combination

i) classic Perceptron:

- Activation function: step function (-1)

$$f(z) = \begin{cases} 1 & \text{if } (z) > 0 \\ 0 & \text{otherwise} \\ \text{or } -1 & \end{cases}$$

classification

- Loss function: Perceptron loss (misclassifica

$$L = -y(w,x) \quad \text{if } y(w,x) \leq 0$$

This is not differentiable, gradient descent isn't standard.

ii) Sigmoid + Binary cross-entropy loss

- Activation function: sigmoid

$$\sigma(z) = \frac{1}{1+e^{-z}}$$ give value as probability (betn 0 to 1)

classification

- Loss Function: (log loss)

$$L = -[y \log(\hat{y}) + (1-y)\log(1-\hat{y})]$$

fully differentiable → gradient descent works

By default perceptron only use: Activation: sign/step function loss: perceptron loss

perceptron directly doesn't support this activation function, for this we have to manually build a model on using SGD we can train model.

3) • Activation func : tanh

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \in (-1, 1)$$

regression task

• loss function : MSE

$$L = \frac{1}{2}(y_i - \hat{y}_i)^2$$

4) Relu + MSE

• Activation func : ReLU

$$ReLU(z) = \max(0, z)$$

both regression and classification

• Loss function :

For Regression : MSE

For classification : Hinge Loss

5) Softmax + categorical crossentropy

• Activation : softmax activation

$$softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

multiclass classification task

• Loss : categorical cross entropy

$$L = -\sum y_i \log(\hat{y}_i)$$