(*) Perceptron model

↓

first model with weights

↓

goal in ML in general: to find a function which satisfactorily maps input to output. These functions have parameters that need to learnt inorder for the accurate mapping, for example weights and bias.

↓

MP neuron had simple function with only one parameter. (b)

↓

a function with a higher degree parameter(s) have more freedom in the trajectory and are thus better able to generalize.

↓

MP neuron fails even on basic data when the the points are at (0,0) and (1,1), for same class and at (0,1) and (1,0) for the other class.

↓

Issues with MP neuron: (i) linear model → non linear mappings are very hard to approximate

(ii) fixed slope

(iii) few possible intercept values

data { (iv) boolean ips
(v) boolean ops }

(vi) learning algorithm: brute force (okay here, but is not scalable to real world problems) in perceptron → real valued inputs

boolean outputs (binary classification)

weights for every i/p → slopes can be varied.

linear model / function

$$loss = \sum_i \max(0, 1 - y_i * \hat{y}_i)$$

specific to perceptron model

sum over all samples.

0 when correct prediction
1 when wrong prediction

(i) Data and Task

↓

real valued inputs, $x \in \mathbb{R}^n$.

↓

the feature's can take values from different ranges & scales, which might give some features (with bigger values), an inherent advantage of representation during aggregation, that can be avoided, by data preparation techniques like standardization, and normalization.

↓

feature scaling

↓

data scaling.

$$z = \frac{x - \mu}{\sigma}$$

μ → mean

σ → standard deviation

(*) Normalization

i) max (min-max scaling)
ii) min

$$x' = \frac{x - min}{max - min}$$

↓

all values b/w 0 & 1

↓

max → 1
min → 0

difference b/w standardization & normalization.

↓

Normalization → min-max scaling

standardization → z-score normalization

↓

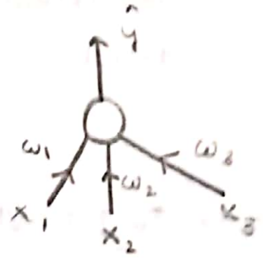
features rescaled to have mean = 0 and $\sigma = 1$.

Task: Binary classification task

(ii) Model → weights associated to each i/p.

things to specify while describing a model:

- kind of i/ps → real values
- kind of o/ps → boolean
- parameters → weights assigned to each feature
- function:
$$\begin{cases} \hat{y} = 1 \text{ if } \sum_{i=1}^{n} w_i x_i \geq b \\ \hat{y} = 0 \text{, otherwise} \end{cases}$$



↓

this model is still linear.

↓

weight is the importance or the weightage attached to different features to the aggregation which is used to achieve the desired o/p.

↓

Desirable features can have +ve weights, varying in magnitude based on the extent of desirability, and similarly for undesirability.

↓

The aggregation can be seen as the dot-product of the feature vector & weight vector and then activation fn. can be applied.

↓

threshold

$\hat{y} = 1$ (if $x.w \geq b$)

$\hat{y} = 0$ (otherwise)

since x's can take real values

(•) geometric interpretation: since b can take real values, ∴ the intercept is much freerier to move, and similarly, as the slopes of the features are not fixed, the line can take more varied forms, alowing it to take more points in its stride.

↓

$$w_1 x_1 + w_2 x_2 - b = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{b}{w_2}$$ → varying intercept

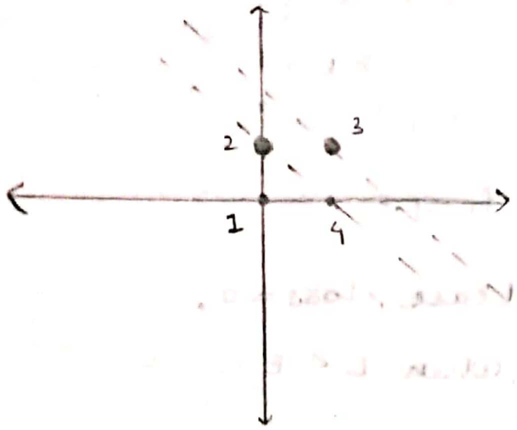
slope is variable and depends on w_1 & w_2

$$x_1 + x_2 - b = 0$$

$$x_2 = -x_1 + b$$

-1 : slope

only intercept could vary



If 2 and 3 are of same class and 1, and 4 from the other class. There is no way for MCP to classify this accurately, whereas perceptron owing to its freedom on the slope side can.

↓

But perceptron still can't completely operate on the XOR example, due to its linear nature.

↓

perceptron model can only deal with linearly separable data.

(iii) Loss function

$$L = \mathbb{1}(Y \neq \hat{Y})$$ → Indicator function

↓

whenever the condition is satisfied, the fn. takes value = 1.
else takes value = 0.

↓

squared error loss is same for MCP (binary o/ps)

(iv) Learning algorithm

typical recipe: a) initialize w_1, w_2, b

b) iterate over data, get predictions associated with current parameters of the model, and accordingly calculate loss associated with the predictions.

c) update (w, b) based on the loss

(updation depends on the learning algorithm)

d) repeat b, c, till satisfied

in ideal case, loss = 0,

but practically, when $L < \epsilon$

some small quantity.

or the loss seems to have hit a minima (cannot be decreased further)

•) The equations of the model can be generalised over the threshold, such that

$$w_n x_n + w_{n-1} x_{n-1} + \cdots + w_1 x_1 + \underset{-b}{w_0} \underset{1}{x_0} \geq 0$$

i.e. $w \cdot x \geq 0$

or $\sum_{i=0}^{n} w_i x_i \geq 0$

where x_0 is fixed and equal to 1.

perceptron learning algorithm :

$P \leftarrow$ inputs with label 1 ;

$N \leftarrow$ inputs with label 0 ;

initialize w ~~random~~ randomly ;

while ! convergence do : → when all i/ps are classified correctly.

Pick random $x \in P \cup N$;

if $x \in P$ and $\sum w_i x_i < 0$, then

$w = w + x$

end

if $x \in N$ and $\sum w_i x_i \geq 0$, then

$w = w - x$

end

end

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

b

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} - \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

•) why it works?

$w = [w_1 \ w_2 \ . \ . \ . \ . \ w_n]$

$x = [x_1 \ x_2 \ . \ . \ . \ . \ , x_n]$

$$\cos\theta = \frac{w \cdot x}{||w|| \cdot ||x||}$$ → sign of cos θ depends on the dot product

↓

For $x \in P$ if $w \cdot x < 0$ then it means that the angle (α) between this x and the current w is greater than 90°, and we need it to be less than 90°

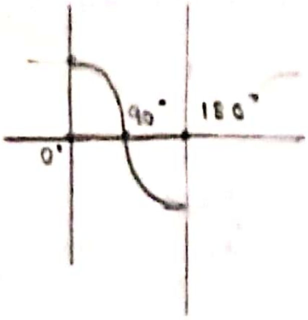
↓

$w_{new} = w + x$

↓

if angle(w, x) > angle(w_{new}, x)

the angle b/w the weights & x should dec. if it can't be made acute straightaway



$\cos(\alpha_{new}) \propto w_{new}^T \cdot x$

$\propto (w+x)^T \cdot x$

$\propto (w^T \cdot x) + x^T \cdot x$

+ve always

$\propto \cos\alpha + x^T \cdot x$

if $\cos(a) < \cos(b)$

$a > b$ (monotonically decreasing)

$\therefore \alpha_{new} < \alpha$

since $\cos(\alpha_{new}) > \cos(\alpha)$

$\cos\alpha + x^T \cdot x > \cos(\alpha)$ ✓

↓

movement to the required condition.

↓

similar argument can be made for the negative or the 0 class.

*) The perceptron algorithm will only converge if the data is linearly separable

*) Defn. of perceptron for linearly separable: Two sets P & N of points in an N dimensional space are called absolutely linearly separable, if n+1 real numbers $w_0, w_1, \ldots, w_n$ exist such that every point $(x_1, x_2, \ldots, x_n) \in P$ satisfies $\sum_{i=1}^{n} w_i \cdot x_i \geq w_0$ and every point $(x_1, x_2, \ldots x_n) \in N$ satisfies $\sum_{i=1}^{n} w_i \cdot x_i < w_0$.

(v) evaluation : checking the performance of the model.

↓

accuracy of the model on the test data.

↓

$$\text{accuracy} = \frac{\text{no. of correct predictions}}{\text{total no. of predictions}}$$