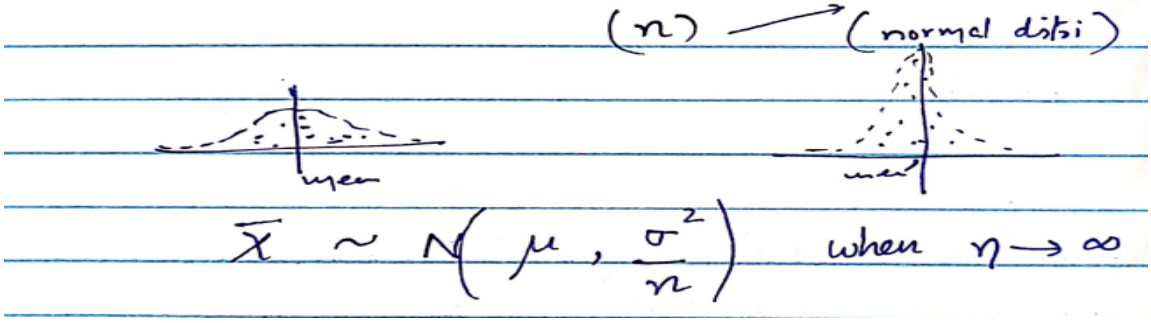
Machine learning Assignment 01

Name: Raman Kumar
Reg NO: BL.EN.P2DSC22009
Course: Mtech Data Science

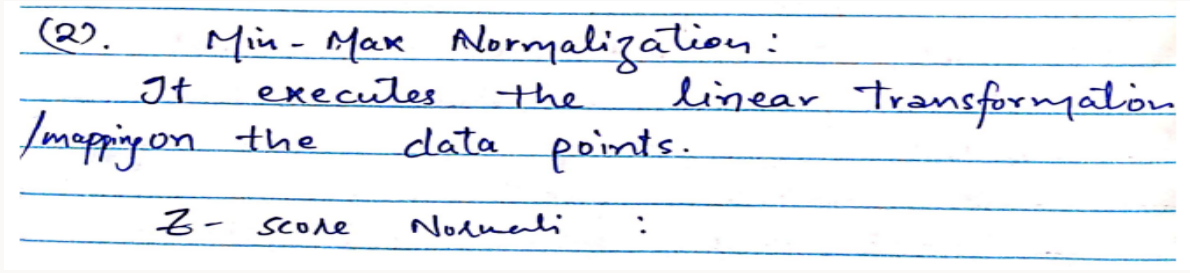
Q1. Define & explain the Central Limit Theorem. Mention its applicability in machine learning.

Ans. In the central limit theorem, we experiment with the samples from the whole population on a dataset. We try to find the primary and useful patterns from the data sample & then discover the pattern to the population while making predictions.
In that way, the central limit theorem helps us to make inferences about the sample parameters and learn better.

- The central limit theorem tells us if we take the mean of the samples of the population & plot the frequencies of their mean, we get a gaussian or normal distribution.
- The more the data 'n' will be it slowly tends to → ∞
-



Q2. Mention the similarities and differences between z-score and min-max normalization.

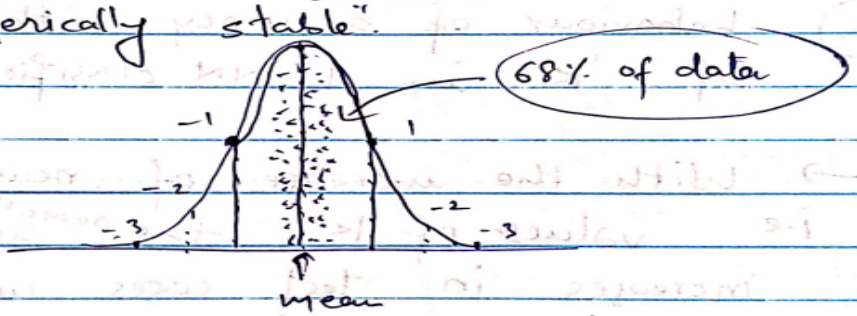


It is the process of rescaling the features so the features get the properties of gaussian distri with μ and σ.

μ ↓ mean

σ ↓ S.D from the mean

→ Primary goal of normalization is to bring the data near to zero. That makes the "optimization" more "numerically stable".



So shifting the mean to '0' ensures that most components of most vectors are close to '0's.

(min-max)	(z-score)
① min. values & max. value of features used for scaling.	① Mean & standared daviation are main keypoints
② Here main affection is because of outliers.	② It is less effected by outliers.
③ Shrinks the values to 0 to 1 (-1 to 1 in case of negative values)	③ No range bounds.

- MIN-MAX guarantees all features will have the exact scale but doesn't handle outliers.
- Z-score , handles outliers well , but doesn't produce normalized data with the exact same scale.

Q3. Explain the behavior of resultant accuracy with a variation of 'k' in k-NN classifier.

With the increase of nearest neighbor i.e value of 'k', the overall accuracy increases in test cases up to a point after that accuracy starts declining again with a large number of k's.
The decision boundary will be smooth with a large value of 'k' and able to execute well on test data.

But the training error rate grows sometimes when 'k' grows, but not the same for test data.
So k should be in a certain balanced range to avoid overfitting & underfitting.

Q4. Compare perceptron learning with back-propagation learning.

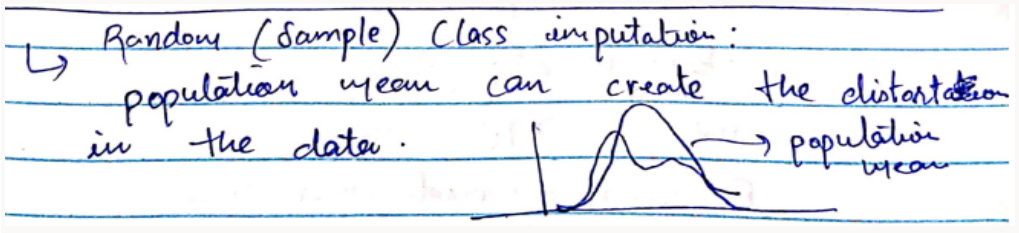
Perceptron networks can be single-layer or multilayer networks of perceptrons.
The data flow in a single layer that is the forward direction from the input to the output layer.
Backpropagation is a technique where a multilayer perceptron receives "feedback" on the error in its results and the multilayer perceptron adjusts its weights according to to make more accurate predictions in the future.

→ single layer perceptron classifiers can only deal with a linearly separable set of patterns.
The multilayer network deals with both.

Q5. The missing values for a continuous dataset (training) may be replaced with the population mean or the class mean. Explain which is better with justifications.

Ans: Median: In case of "outliers"
Mode: In case of categorical values
Mean is the most common method for replacing the missing values, but not in case of outliers.
Sample Mean: We use this in case of low accuracy.
Population Mean: In case of high accuracy.
Imputation of the population mean in place of missing values can lead to overfitting.

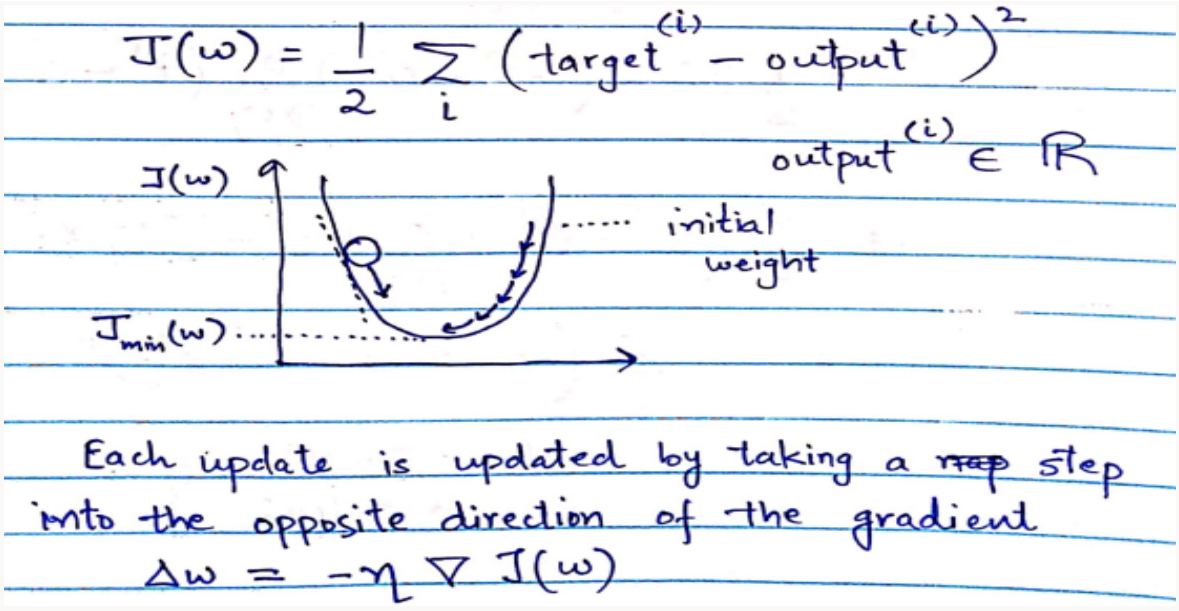
- Random (sample) class imputation of mean into missing values.
- Population mean can create the distortion in the data.



Q6. Derive the formula for the gradient descent algorithm used in perceptron learning.

Being a continuous function, one of the biggest advantages of the linear activation functions over the unit step function is that it is differentiable. This property allows us to define a cost function J(w) that we can minimize in order to update our weights.

In the case of the linear function (activation), we can define the cost functions J(w) as the "sum of the squared errors" (SSE) which is similar to the cost function that is minimized in ordinary least squares (OLS) linear regression.



Thus we have to compute the partial derivative of the cost function for each weight in the weight vector.

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

Partial derivative of SSE cost function for a particular weight can be calculated as

$$\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \left(\sum_i \left(t^{(i)} - o^{(i)} \right)^2 \right.$$

$$= \frac{1}{2} \sum_i \frac{\partial}{\partial w_j} \left(t^{(i)} - o^{(i)} \right)^2$$

$$= \frac{1}{2} \sum_i 2 \left(t^{(i)} - o^{(i)} \right) \frac{\partial}{\partial w_j} \left(t^{(i)} - o^{(i)} \right)$$

$$= \sum_i \left(t^{(i)} - o^{(i)} \right) \frac{\partial}{\partial w_j} \left(t^{(i)} - \sum_j w_j x_j^{(i)} \right)$$

$$= \sum_i \left(t^{(i)} - o^{(i)} \right) \left(-x_j^{(i)} \right)$$

$t \rightarrow$ target
$o \rightarrow$ output

And if we plug the results back into the learning rule, we get

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = -\eta \sum_i \left[t^{(i)} - o^{(i)} \right] (-x_j^{(i)})$$

$$= \eta \sum_i \left(t^{(i)} - o^{(i)} \right) x_j^{(i)}$$

Eventually, we can apply a simultaneous weight update similar to the perceptron rule.

$$w := w + \Delta w$$

↳ The parameter 'w' is iteratively updated by taking steps proportional to the negative of the gradient:

$$w(k+1) = w(k) - \Delta w(k)$$

$w(k)$ → the value of 'w' at iteration 'k'

$$\Delta w = \mu \frac{\partial E}{\partial w}$$

$\frac{\partial E}{\partial w}$ → gradient of Loss function w.r.t.

μ → "Learning rate"

For each sample i this gradient can be splitted according to the "chain rule"

$$\frac{\partial E}{\partial w} = \frac{\partial E_i}{\partial y_i} \cdot \frac{\partial y_i}{\partial w}$$

$E_i \longrightarrow$ squared error loss

$$\frac{\partial E_i}{\partial y_i} = \frac{\partial (t_i - y_i)^2}{\partial y_i}$$

$$= -2 (t_i - y_i)$$

$\Rightarrow \quad 2\,(y_i - t_i)$

since $y_i = x_i \cdot w$ ⟶ 1-input, 1-output linear regression

(x) —w→ (y) w → weight

↳ $\frac{\partial E_i}{\partial w} = \frac{\partial (x_i \cdot w)}{\partial w}$

$= x_i$

So, update function Δw for sample i will become

$\Delta w = \mu \cdot \frac{\partial E_i}{\partial w}$

$= \mu \cdot 2 x_i (y_i - t_i)$

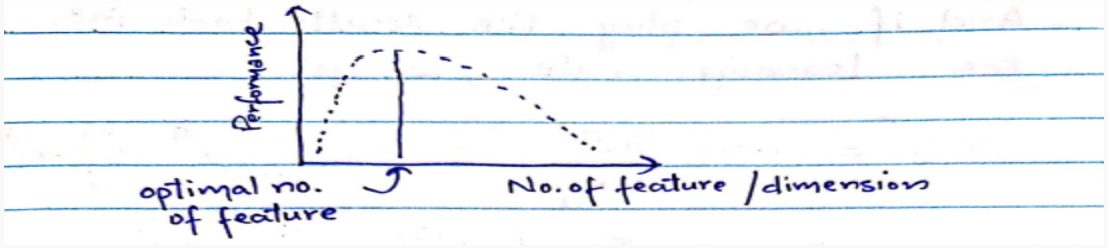
↳ In the batch Processing, we just add up all the gradients for each sample:

$\Delta w = \mu * 2 * \frac{1}{N} \sum_{i=1}^{N} x_i (y_i - t_i)$

Q7. Name and explain the curses of dimensionality.

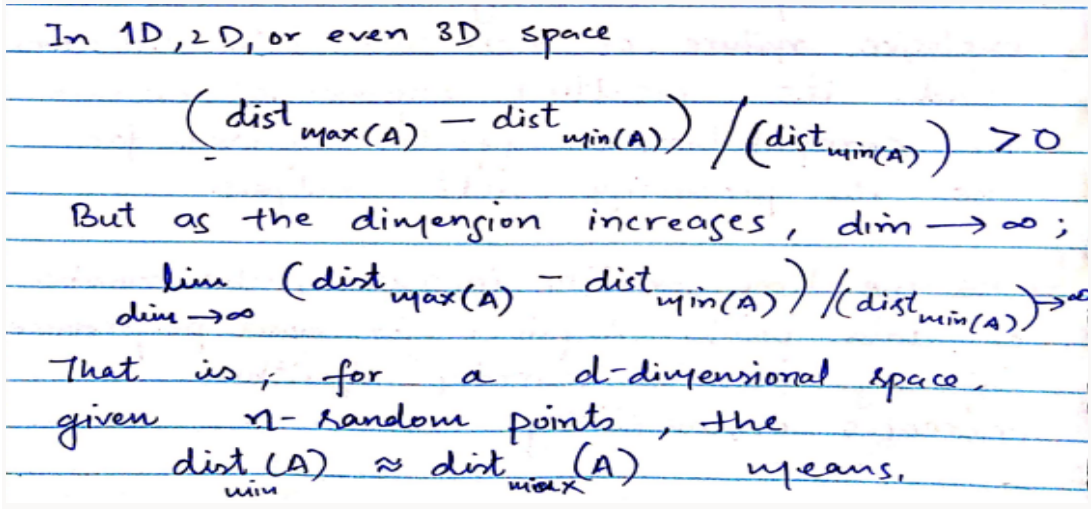
Curse of dimensionality describes the explosive nature of increasing data dimensions aand its resulting exponential increase in computational efforts required for its processing and analysis.

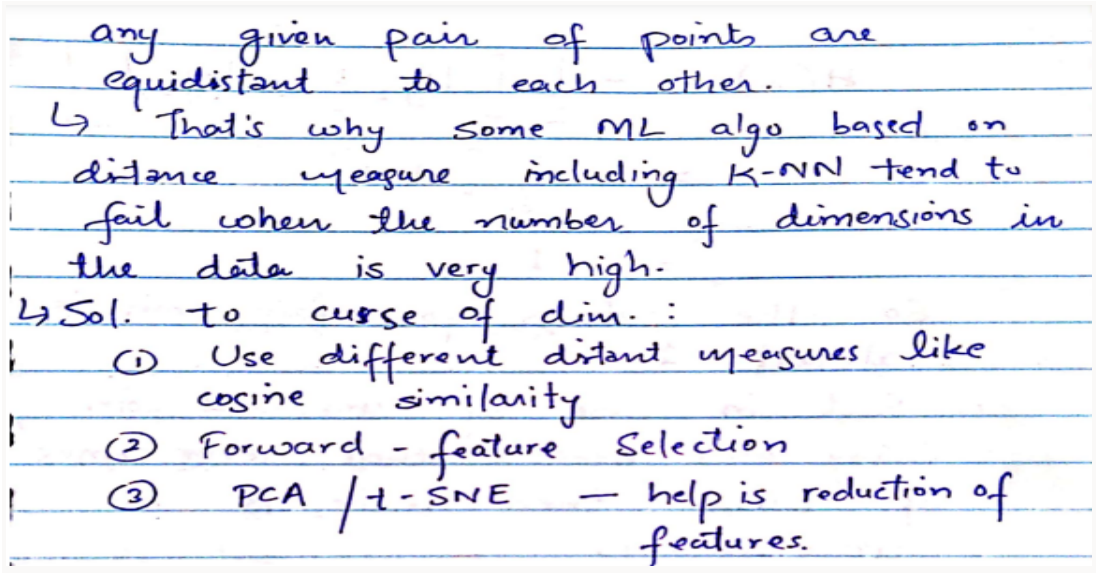
→ As the dimensionality increase, the number of data points required for good performance of any machine learning model increasing exponentially.



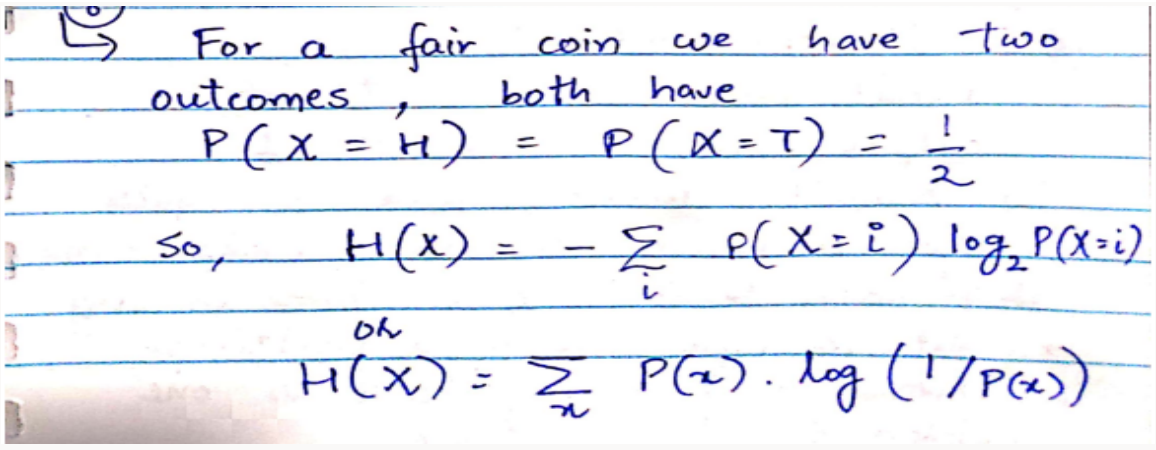
Effects of curse of dimensionality on distance function:

For any point A lets say $dist_{min}(A)$ is the minimum distance between A and its nearest neighbour and $dist_{max}(A)$ is the max distance between A and farthest neighbour.





Q8. When a biased coin is being tossed alongside a fair one, which shall have more entropy? Why?



$$H(X) = -\frac{1}{2}\left[\log_2 \frac{1}{2} + \log_2 \frac{1}{2}\right]$$

$$= -\frac{1}{2}[-1-1]$$

$$= 1$$

So, the entropy for fair coin is almost 1.

↳ But in case of unbiased coin (say only heads) that never comes up tails, then their is no uncertainity.

The entropy = 0, as each toss of coin delivers no new information as the outcome of each coin toss is always certain.

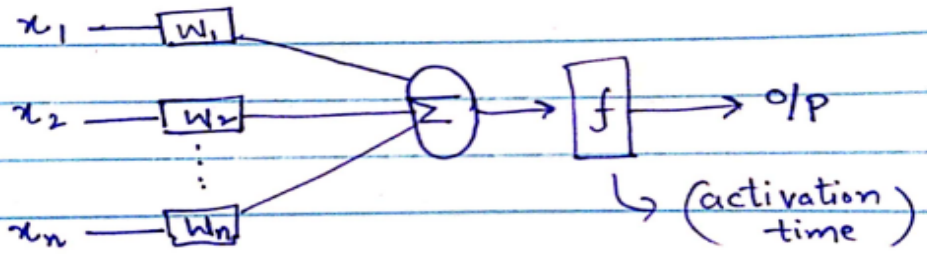
Q9. Explain the role of bias and activation functions in a neural network.

(9). Bias : Bias in neural network can be defined as the constant which is added to the product of features and weights. It is used to offset the result. It helps the model to shift the activation function towards positive or negative side. Also it increases the accuracy. It shift the straight line from ~~line~~ origin.

Types : Sample bias confirmation bias
Algorithm bias
Anchoring bias

Activation function :

This decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of activation function is to add non-linearity into the output of a neuron.



Types : Step function
Sigmoid function
ReLU
Leaky ReLU

Q10. Explain the triangle inequality of a distance metric.

(10). Triangle inequality of a distance metric :

In a metric space M, with metric d, the triangle inequality is a measurement upon distance.

$$d(x, z) \leq d(x, y) + d(y, z)$$

for all x, y, z belongs to M.

Also distance between x and z is atmost as large as the sum of distance between x & y and y & z.

→ It is responsible for convergence in a metric space. For normal vector spaces, the inequality reduces to,

$d(x, y) := \|x - y\|$, with $x - y$ being the vector pointing from y to x.

Q11. Find the rank of the below-provided matrix (provide the steps). Is the matrix invertible? Explain the reason for your answer.

(11)

$$A = \begin{bmatrix} 11 & 2 & 13 & 7 \\ 2 & 2 & 5 & 19 \\ 2 & 5 & 12 & 15 \\ 20 & 8 & 35 & -13 \end{bmatrix} \quad r_2 - r_1$$

$$\begin{bmatrix} 2 & 2 & 5 & 19 \\ 11 & 2 & 13 & 7 \\ 2 & 5 & 12 & 15 \\ 20 & 8 & 35 & -13 \end{bmatrix} \quad \begin{array}{l} r_2 \to 2r_2 - 11r_1 \\ r_3 \to r_3 - r_1 \\ r_4 \to r_4 - 10r_1 \end{array}$$

$$\begin{bmatrix} 2 & 2 & 5 & 19 \\ 0 & -18 & -29 & -195 \\ 0 & 3 & 7 & -4 \\ 0 & -12 & -15 & -203 \end{bmatrix} \quad \begin{array}{l} R_3 \to 6R_3 + R_2 \\ R_4 \to R_4 + 4R_3 \end{array}$$

$$\begin{bmatrix} 2 & 2 & 5 & 19 \\ 0 & -18 & -29 & -195 \\ 0 & 0 & 13 & -219 \\ 0 & 0 & 13 & -219 \end{bmatrix} \quad R_4 \to R_4 - R_3$$

$$\begin{bmatrix} 2 & 2 & 5 & 19 \\ 0 & -18 & -29 & -195 \\ 0 & 0 & 13 & -219 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$n = 4$

Rank of matrix = 3

one row is completely zero.

$\therefore \quad |A| = 0$

it is not invertible.

$A^{-1} = \frac{1}{|A|} \text{ adj. } A \quad ;$

$\Rightarrow \quad |A| \neq 0$, for invertible matrix.

(10). Triangle inequality of a distance metric :

Q12. Comparing the input and output ranges of sigmoid and step activation functions, explain the behavior of these functions. Explain which one should be preferred over the other with justification.

Sigmoid Function:

(12). Sigmoid function :

input range : $-\infty$ to ∞

output range : 0 to 1.

↳ Sigmoid function used for multi-layer perceptron networks. (backpropagation)

~~that could lead~~ ↳ small change in any weight in input layer of a multilayer perceptron could possibly lead to one neuron to flip $0 \rightarrow 1$.

↳ which change the hidden layer's behaviour & results also.

↳ Sigmoid fun. have no or less shortcomings and it is useful in MLP network so is better than step function.

↳ But for binary classification step fun. is useful.

Step Function:

Step function :

input range :

output range : $y = \begin{cases} 1 & , \text{if } x \geq 0 \\ 0 & , \text{if } x \leq 0 \end{cases}$

↳ It is more useful with single layer perceptron, where the linear data is seperable.

↳ Step function is non-differentiable at $x=0$, and its derivative is 0 elsewhere. So it won't be of much use in backpropagation as the requirement for them is differentiable activation fun.