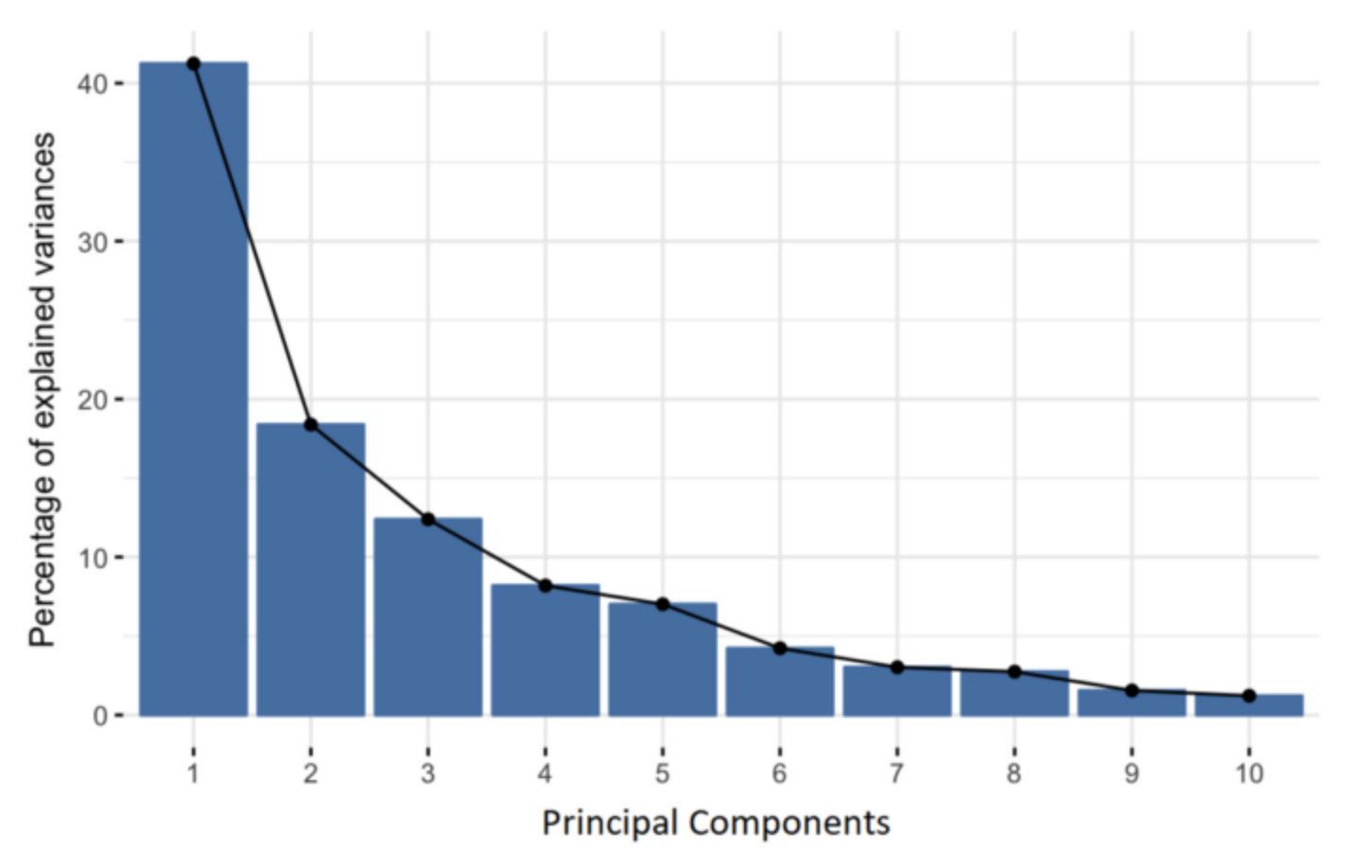
PRINCIPAL COMPONENT ANALYSIS

\*What is PCA?

Principal component analysis (PCA) is a dimensionality reduction and machine learning method used to simplify a large data set into a smaller set while still maintaining significant patterns and trends.

\*What Are Principal Components?

Principal components emerge as new variables through linear

combinations of original variables.

These combinations ensure uncorrelatedness among the principal

components.

The majority of information from the initial variables condenses into the

initial principal components.

PCA aims to maximize information in each component successively.

It facilitates dimensionality reduction while retaining crucial data aspects by discarding less informative components.

Interpretability of principal components is limited due to their linear

combination nature.

Geometrically, principal components denote directions explaining

maximum variance in the data.

The variance along each principal component signifies the information it holds.

They act as new axes providing optimal perspectives for assessing data distinctions.

\*How PCA Constructs the Principal Components?

The first principal component is constructed to capture the

highest variance in the data.

In a scatter plot, it corresponds to a line passing through the

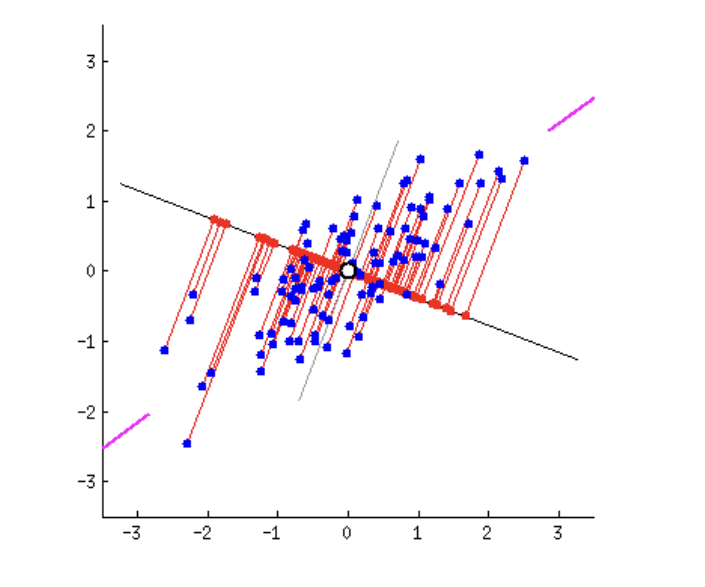
origin that maximizes the spread of the points.

Mathematically, it maximizes the variance, which is the average squared distance of points from the origin.

The second principal component is perpendicular to the first and captures the next highest variance.

This process continues until all p principal components are

derived, equaling the original number of variables.

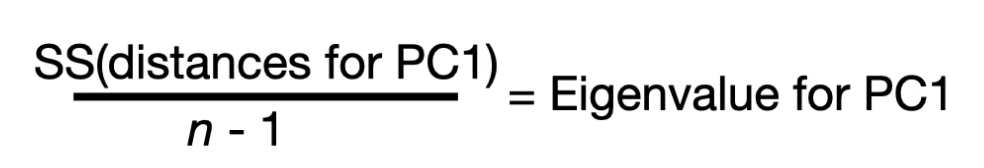


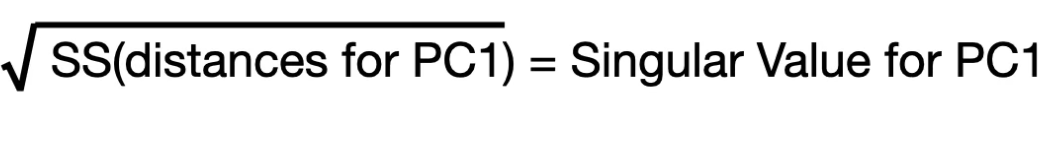
\*Terminologies

**—>**The unit vector along PC 1 line is called **singular vector /eigenvector.**

**—>**The proportion of data sets is called **loading scores.**

**—>**The average of sum of squared distance on PC 1 is called

**Eigenvalue.**

**—>**The square root of SS(distance) is called **singular value.**

Lets take example of plot given below-

Eigenvalues are just the measure of variations.

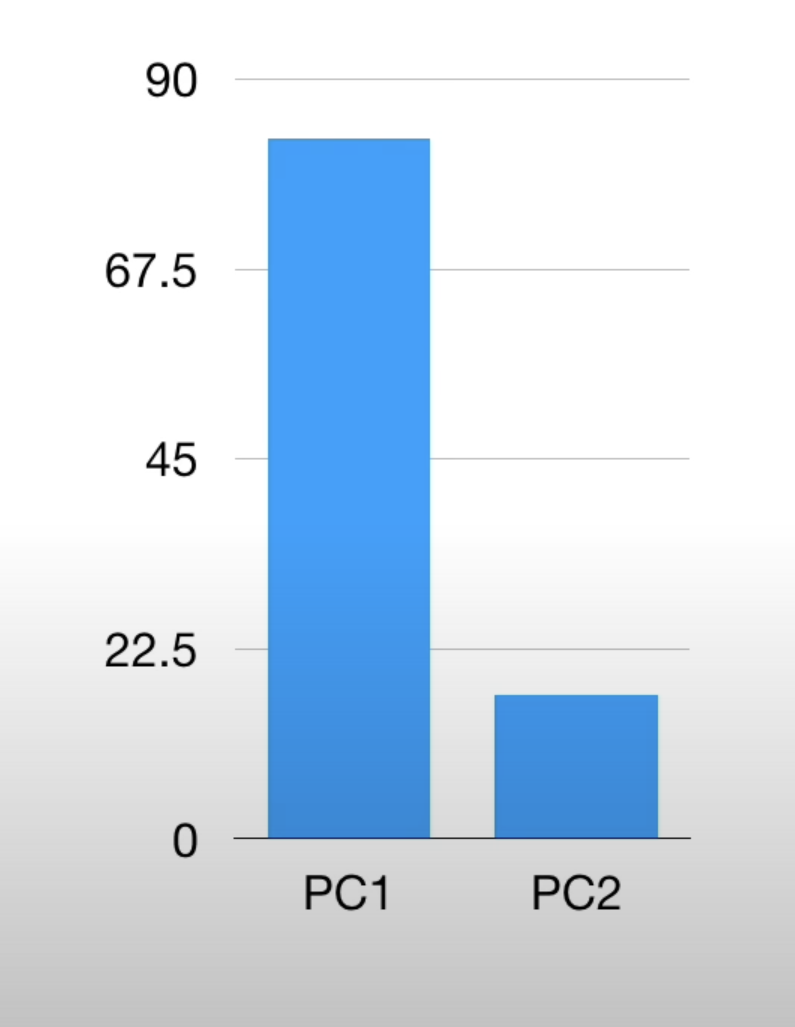
The variation for PC1 is 15 and for PC2 is 3.

Total variation is 18.

PC1-83% variation

PC2-17% variation

**Scree plot-**the graphical representation of percentages of

variances that each PC accounts for.

NEURAL NETWORKS

Neural networks are machine learning models inspired by the brain’s

structure.

\*Forward propagation

During forward propagation, input data moves sequentially through the network from one layer to the next, undergoing processing based on the activation function at each hidden layer before being forwarded to the next layer.

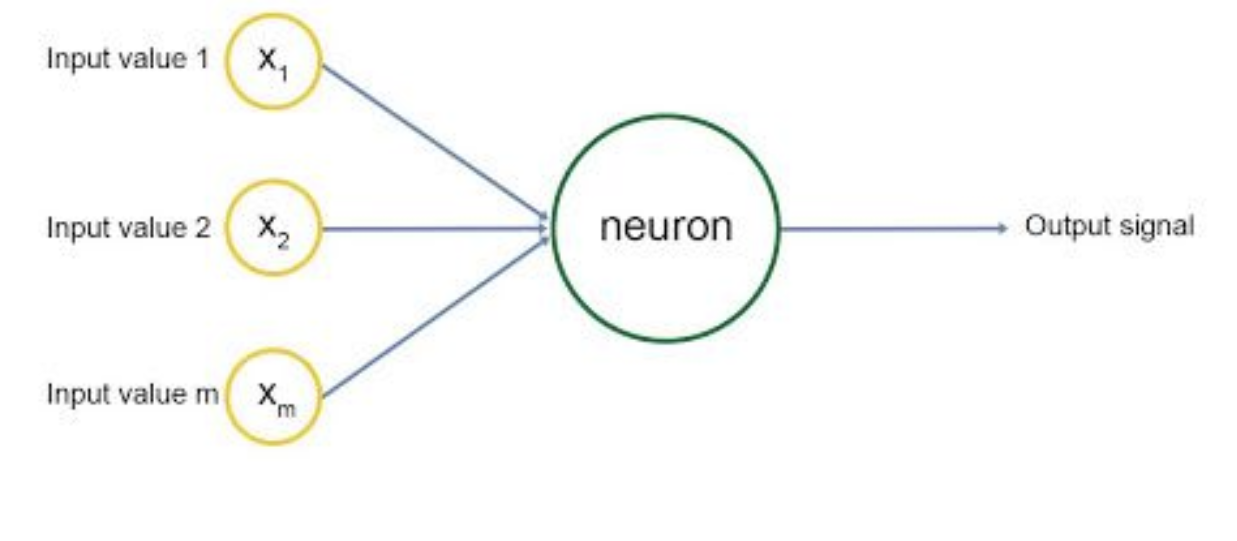
\*Components of forward propagation model

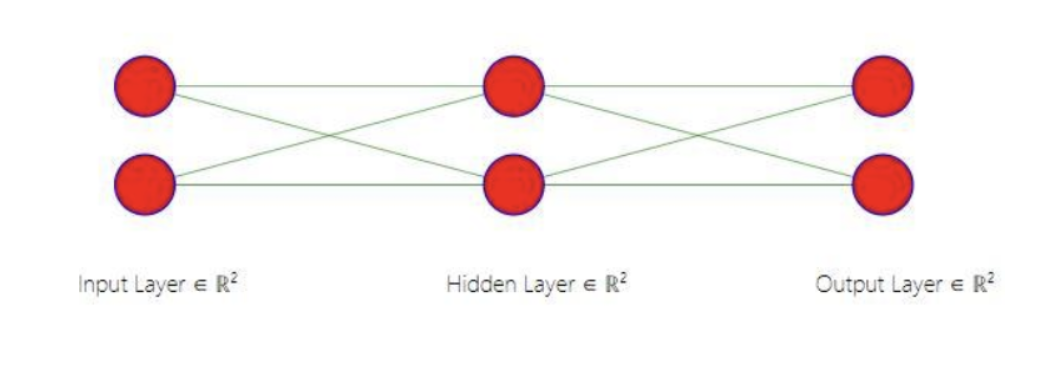
**Preactivation:** This involves calculating a weighted sum of

inputs, which is the linear transformation of weights with respect to the available inputs.

**Activation:** The computed weighted sum of inputs is then passed through an activation function.

Example-sigmoid function



Applications of forward propagation

A 3-layer neural network with 2 input units, 2 hidden layer units, and 2 output units predicts market direction.

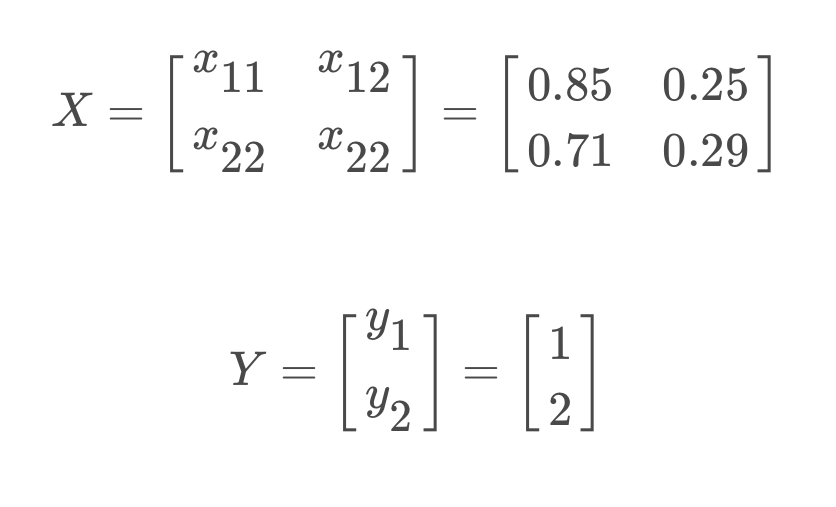
Class 0 represents the market closing down, while Class 1 indicates the market closing up.

Input data includes x1 (correlation between close prices and 10-day SMA) and x2 (difference between close price and 10-day SMA).

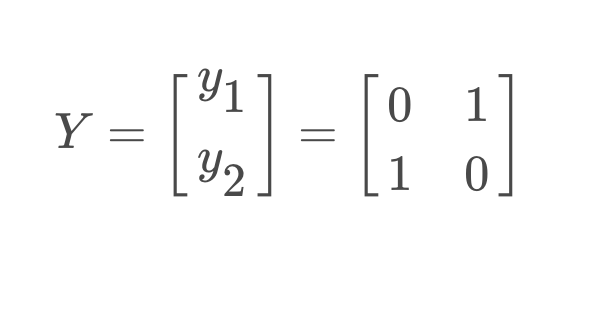
For example, a datapoint in Class 1 has inputs X = [0.85, 0.25].

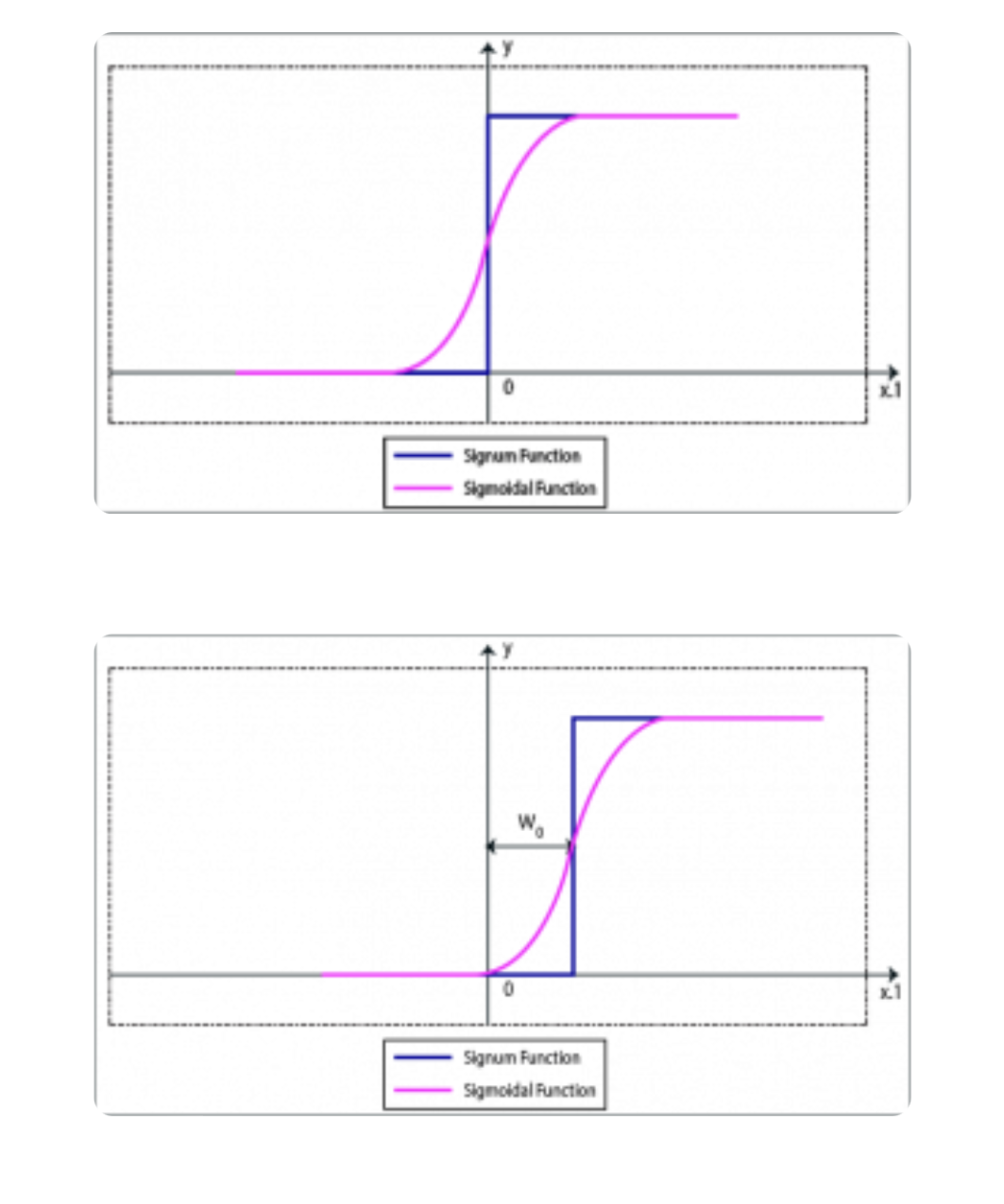
The corresponding output label y is [1], indicating Class 1.

This setup trains the neural network for market prediction tasks.

Example with two data points:

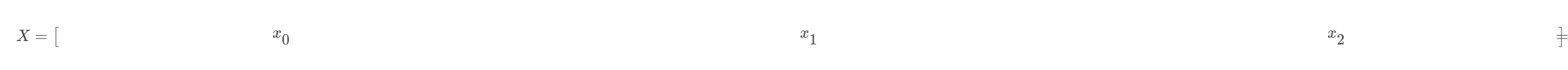
The model's output is categorical, representing discrete classes. To enable probability prediction for each class, the output data is converted into matrix form. In this matrix, columns represent classes, and rows represent input examples.



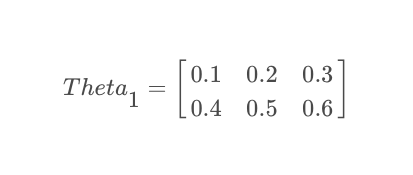
In a neural network learning, apart from the input variable, we add a bias term to every layer other than the output layer. This bias term is a constant, mostly initialized to 1. The bias enables moving the activation threshold along the x-axis.

the forward propagation process with the given inputs and weights:

**Adding Bias Column to Input Data (X):**

****

Here, x0 is introduced as a bias term with a value of 1.

**Initializing Weights (Theta1) for Hidden Layer:**

The weights are arranged such that each row corresponds to a neuron in the second layer.

**Matrix Multiplication (Forward Propagation):**

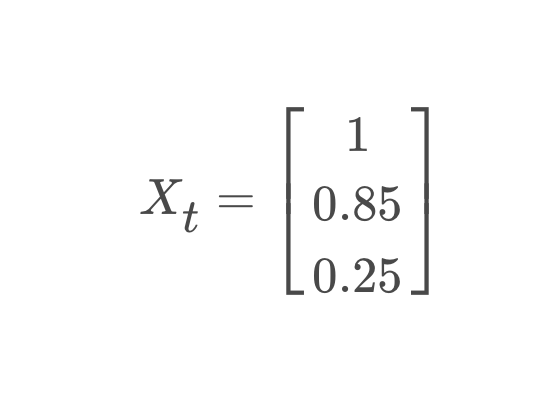
Before multiplying, transpose X to ensure correct correspondence of weights with input examples.

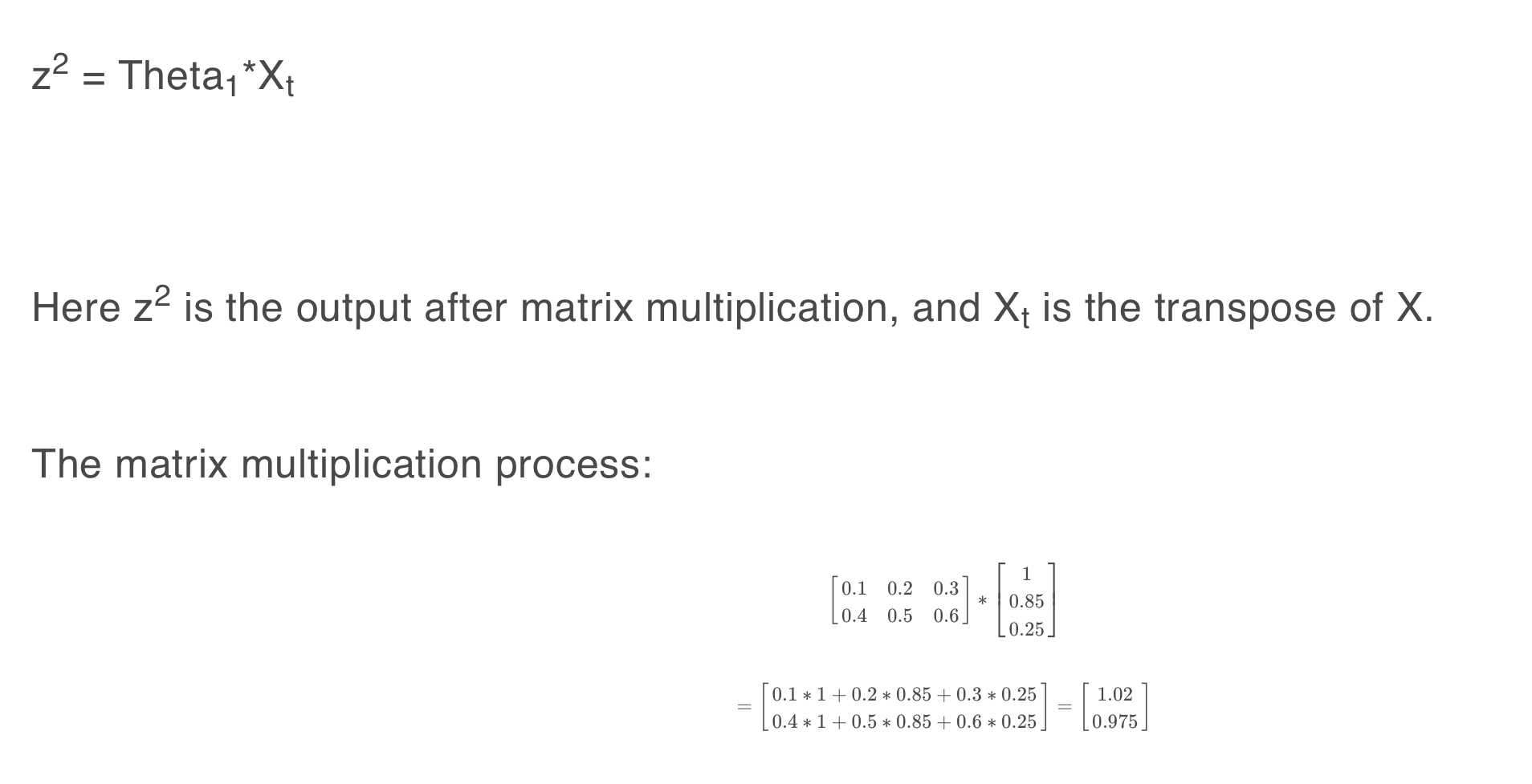
**X \* Theta1 = Xt\* Theta1**

Each element of the product is the dot product sum of a row in Xt with each column in Theta1.

By transposing X and performing matrix multiplication, we

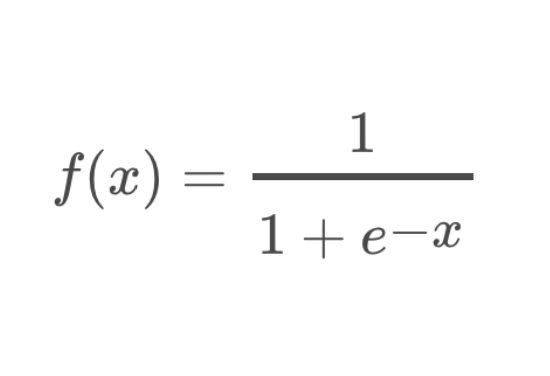
effectively multiply each weight in Theta1 with its corresponding input value, accounting for the bias and enabling the neural network to learn and make predictions based on these weighted inputs.

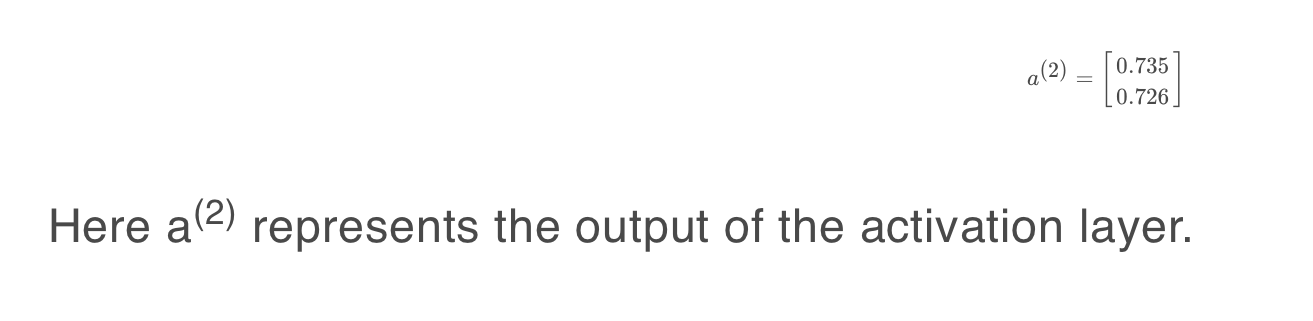




Applying sigmoid activation after the input layer-

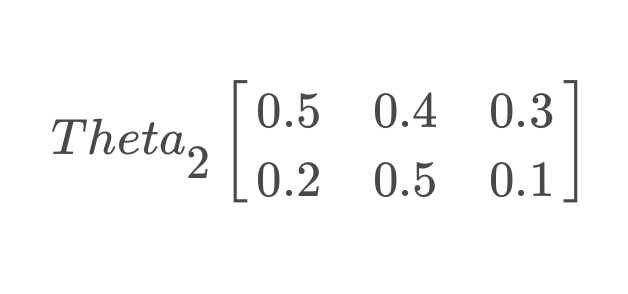
Sigmoid function —>



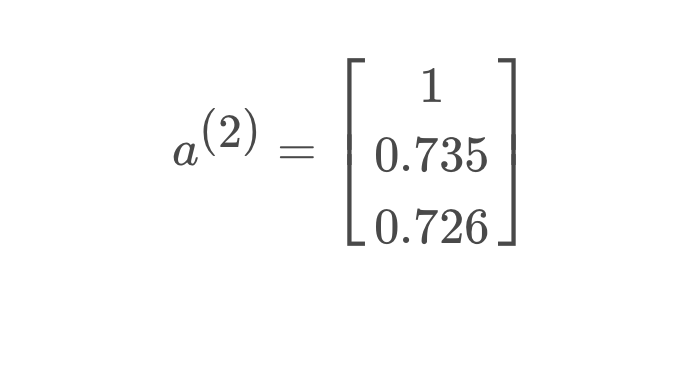
After activation function we get—

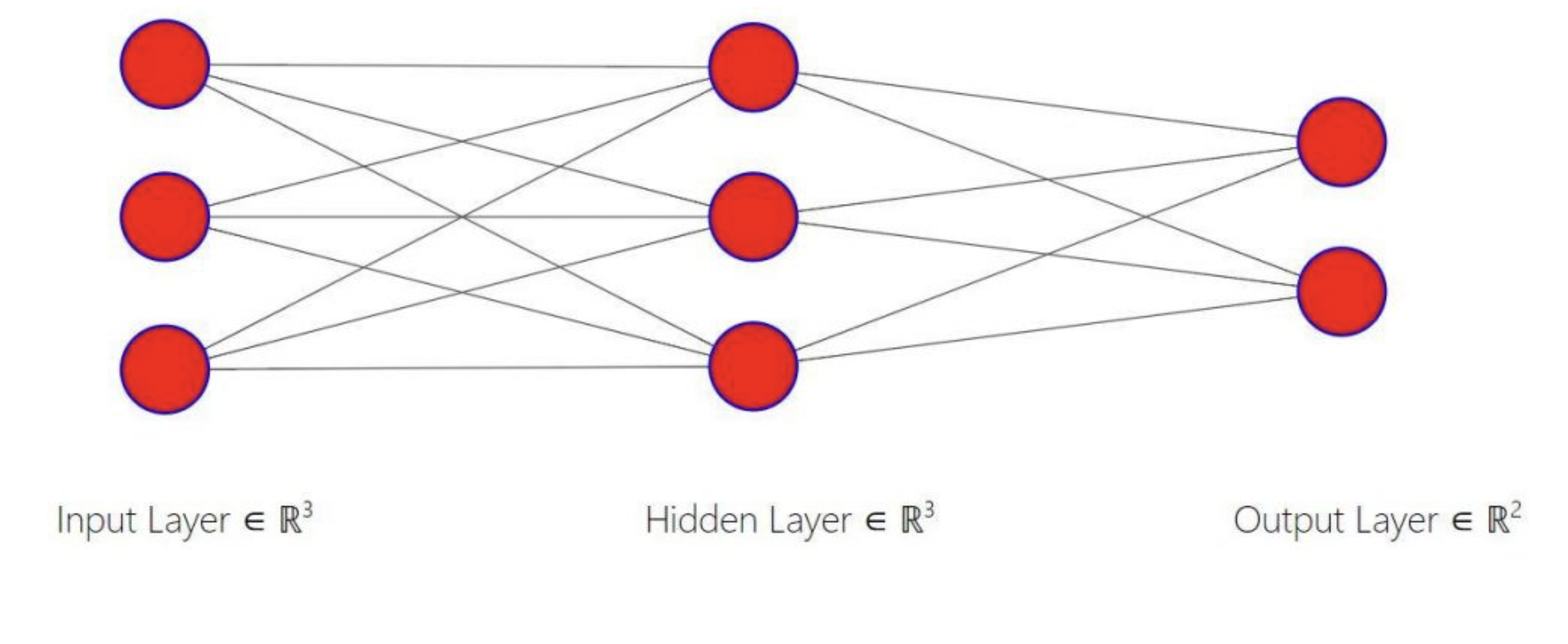
The activation layer's outputs become inputs for the output layer. Theta2, initialized with random weights, connects the hidden

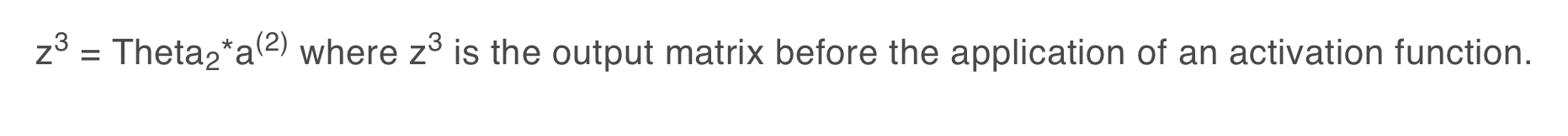
layer to the output layer. Each row in Theta2 represents the weights for the two neurons in the output layer.



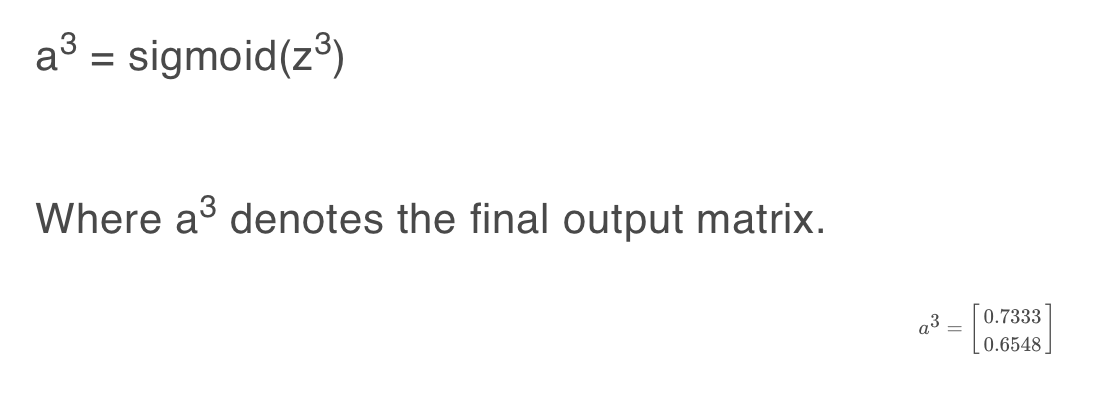
After bias addition and initialisation of weights in theta2

a(2) is-



Now we perform for -

After multiplication matrix we get-



Where row1 represents probability of class 0 and row 2

represents that of class 1.a