Artificial Intelligence Masterclass

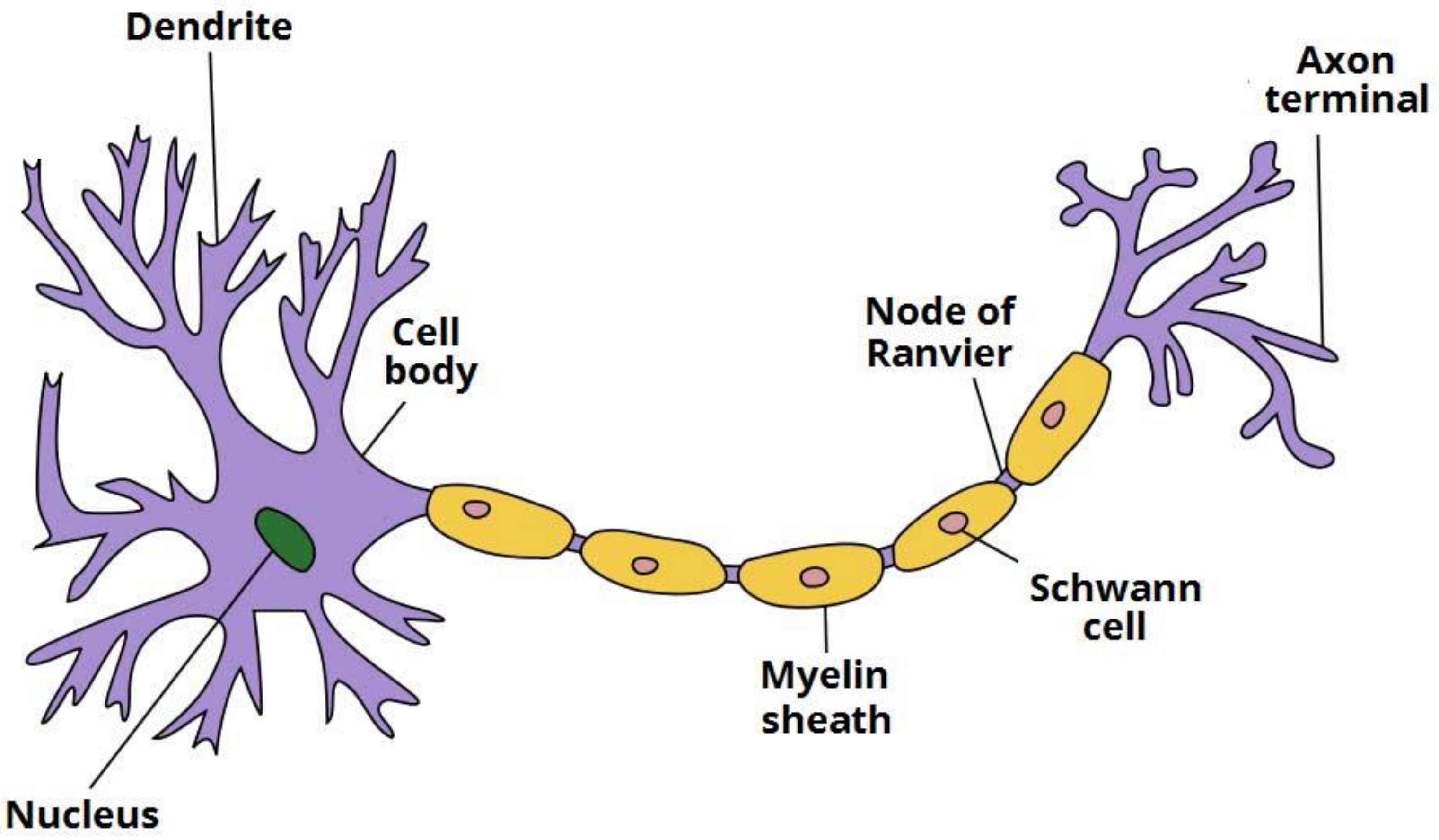
Perceptron Neural Network

H.M. Samadhi Chathuranga Rathnayake

M.Sc in CS (SU), PG.Dip in SML (Othm), PG.Dip in HRM (LRN), B.Sc (Hons) in IS (UOC), B.Eng (Hons) in SE (LMU), P. Dip EP & SBO (ABE), Dip SE, Dip IT, Dip IT & E-Com, Dip B.Mgt, Dip HRM, Dip Eng

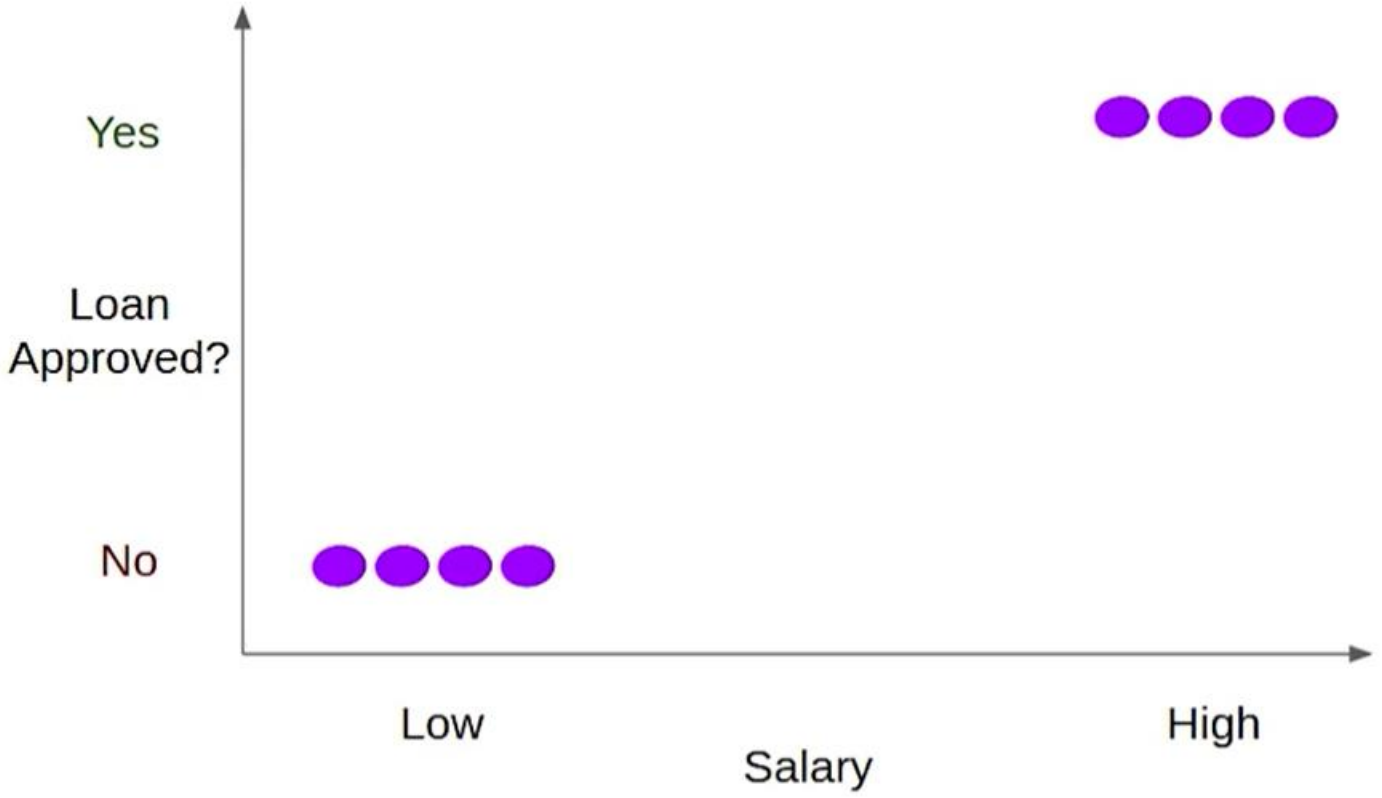
Perceptron

This the general process of decision making inside a biological neuron.



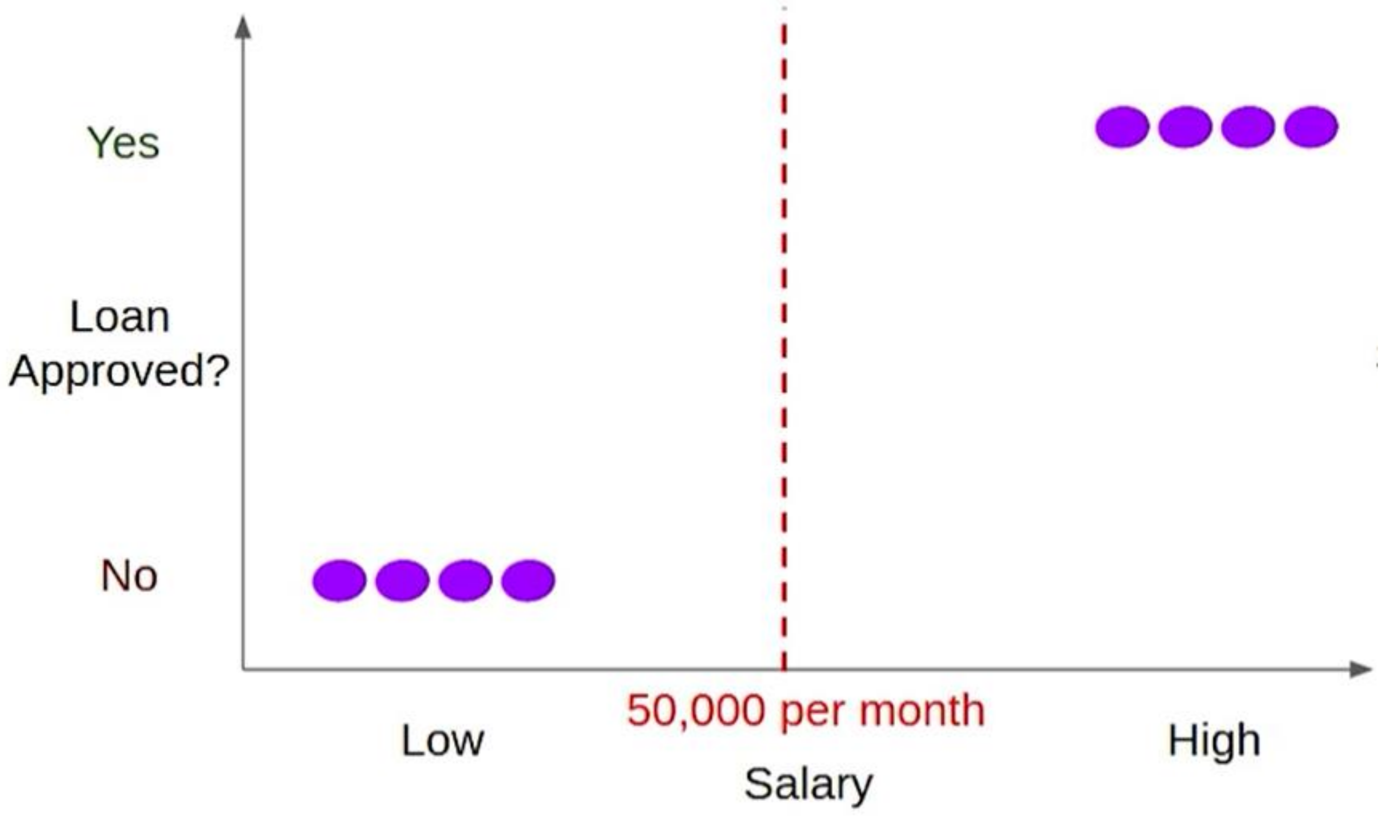
Perceptron

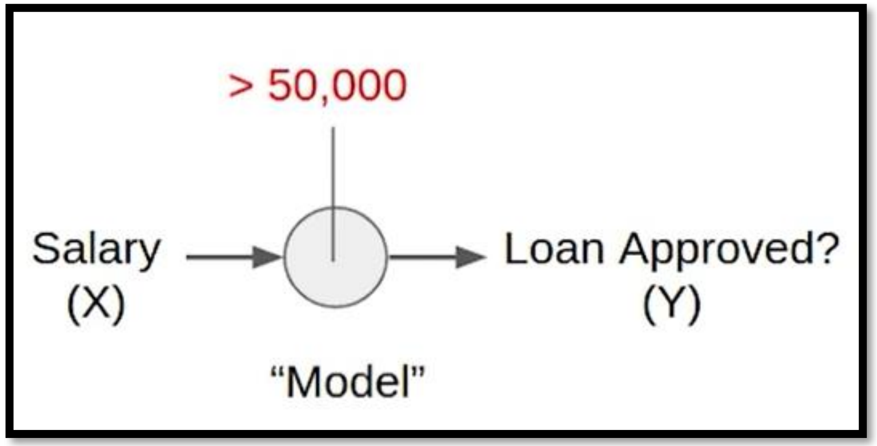
Think about following question. Here with the salary, it will check whether the loan is approved or not.



Perceptron

Think that the loan is going to approved when the salary is greater than 50000.



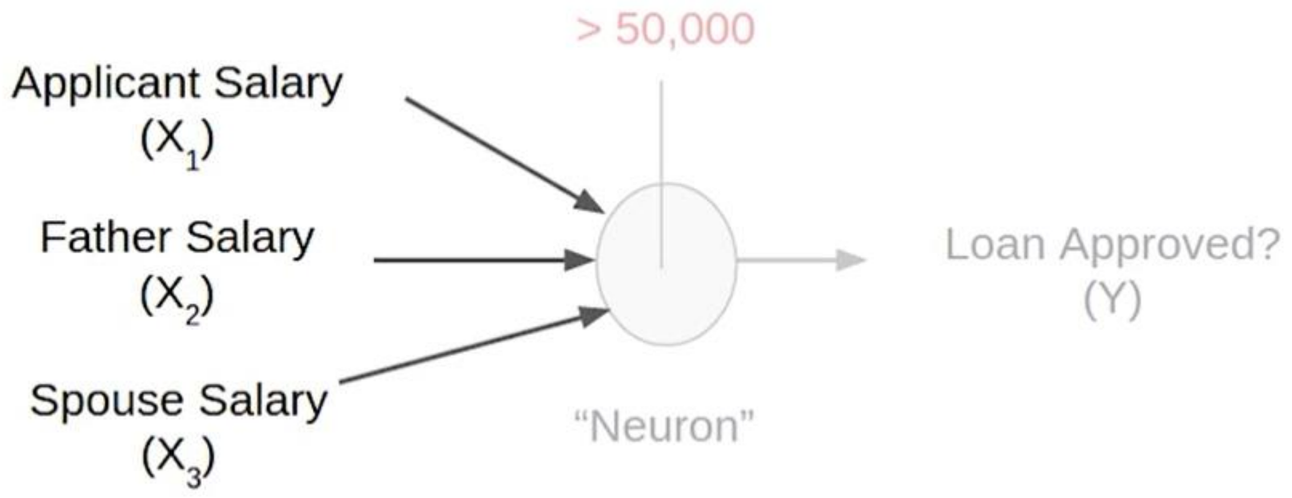


Tasks:

- Salary as input
- Check if it is at least 50000 or not?
- If the condition is True the approve the loan

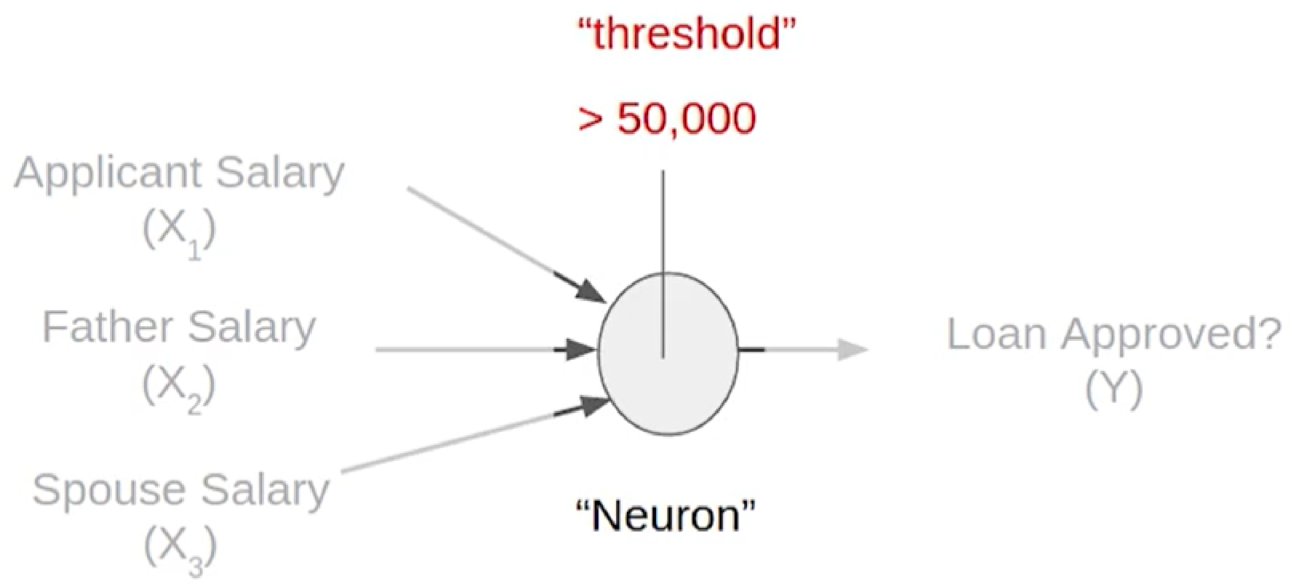
Perceptron

More than one input variables can be considered.



Perceptron

Consider that the addition of these input values is going to be compared with the threshold.



$$X_1 + X_2 + X_3 > T$$

$$X_1 + X_2 + X_3 - T > 0$$

$$X_1 + X_2 + X_3 + Bias > 0$$

Total Income = Applicant Salary (X_1) + Father Salary (X_1) + Spouse Salary (X_3)

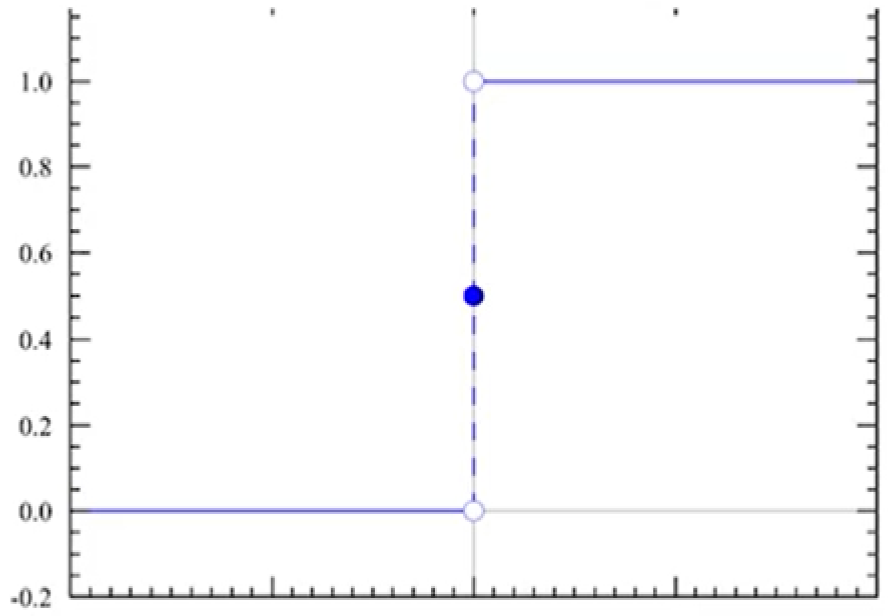
Total Income > threshold ?

$$X_1 + X_2 + X_3 + Bias > 0 \quad \boxed{1}$$

$$X_1 + X_2 + X_3 + Bias < 0 \quad \boxed{0}$$

Perceptron

This follows the step function.

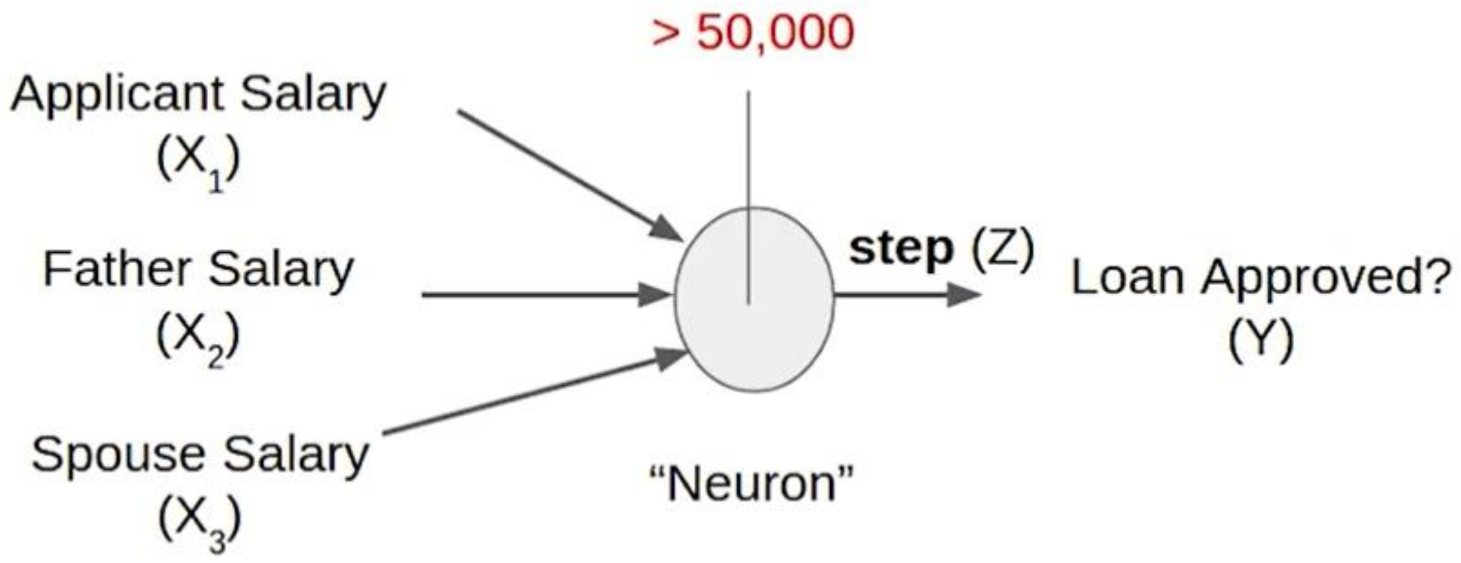


$$Z = X_1 + X_2 + X_3 + Bias$$

$$Step(Z) = Y = \begin{cases} 1 & Z > 0 \\ 0 & Z \leq 0 \end{cases}$$

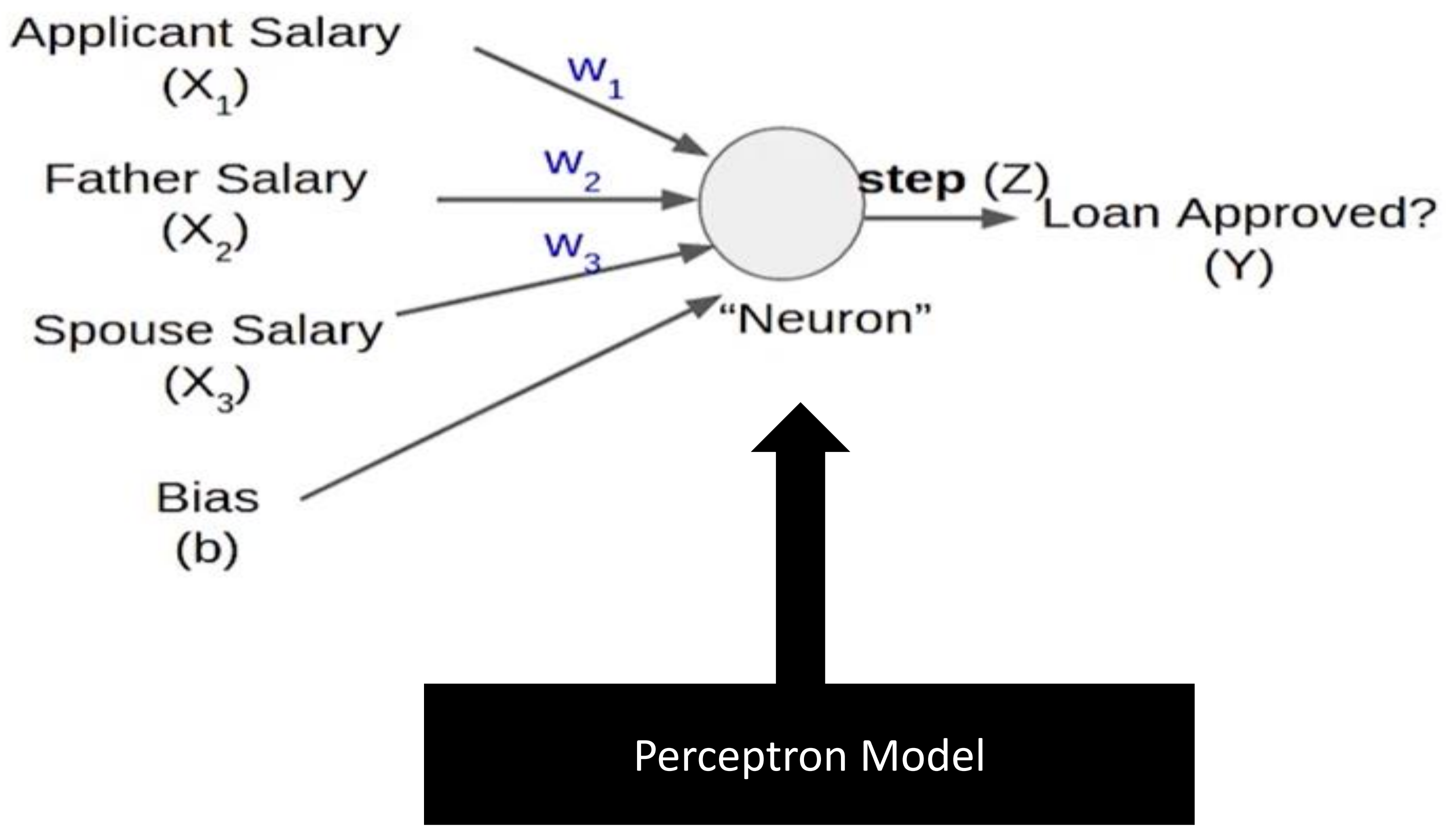
Perceptron

Finally, the system can be obtained as,



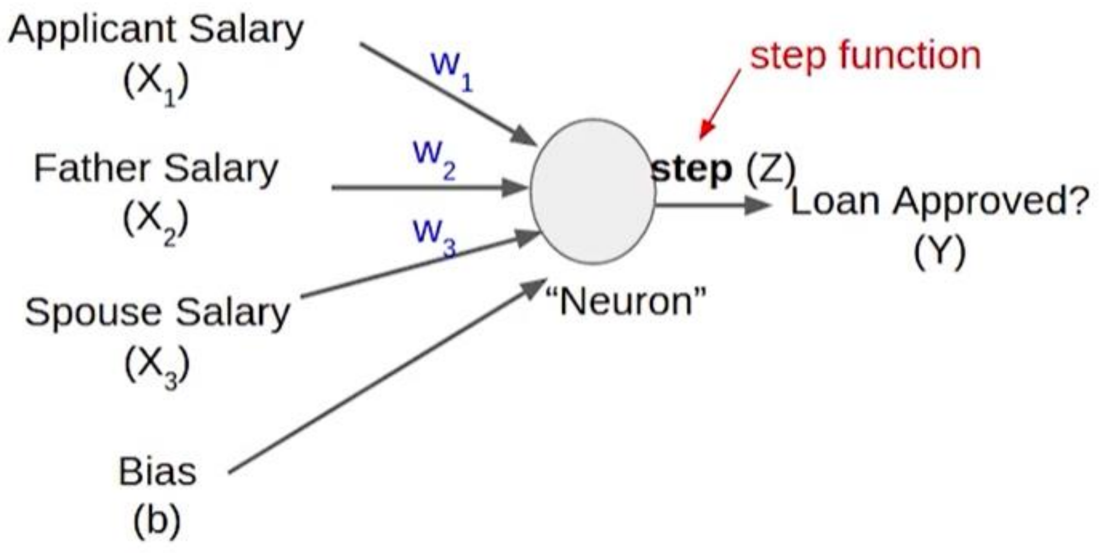
Perceptron

The weight can be changed for each input.



Perceptron

Weights and bias values can be changed and updated.



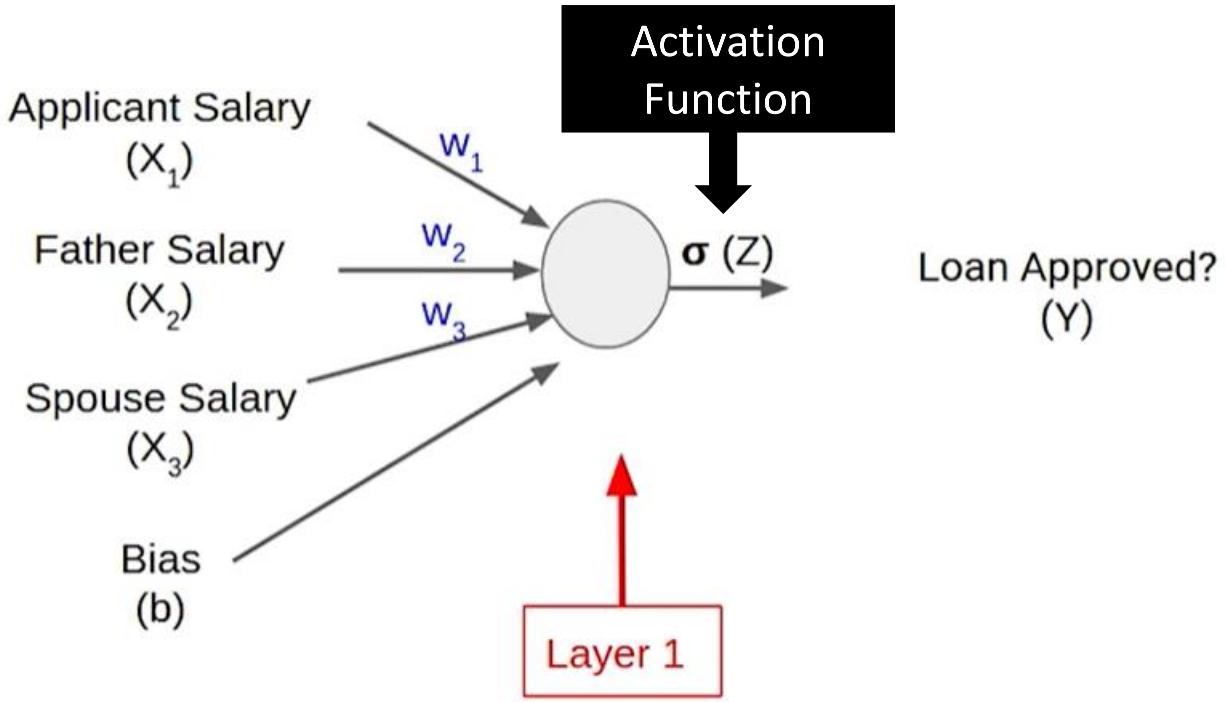
Sum of inputs = $X_1 * w_1 + X_2 * w_2 + X_3 * w_3$

$Z = X_1 * w_1 + X_2 * w_2 + X_3 * w_3 + b$ (bias)

$\hat{Y}$ (output) = **step** (Z)

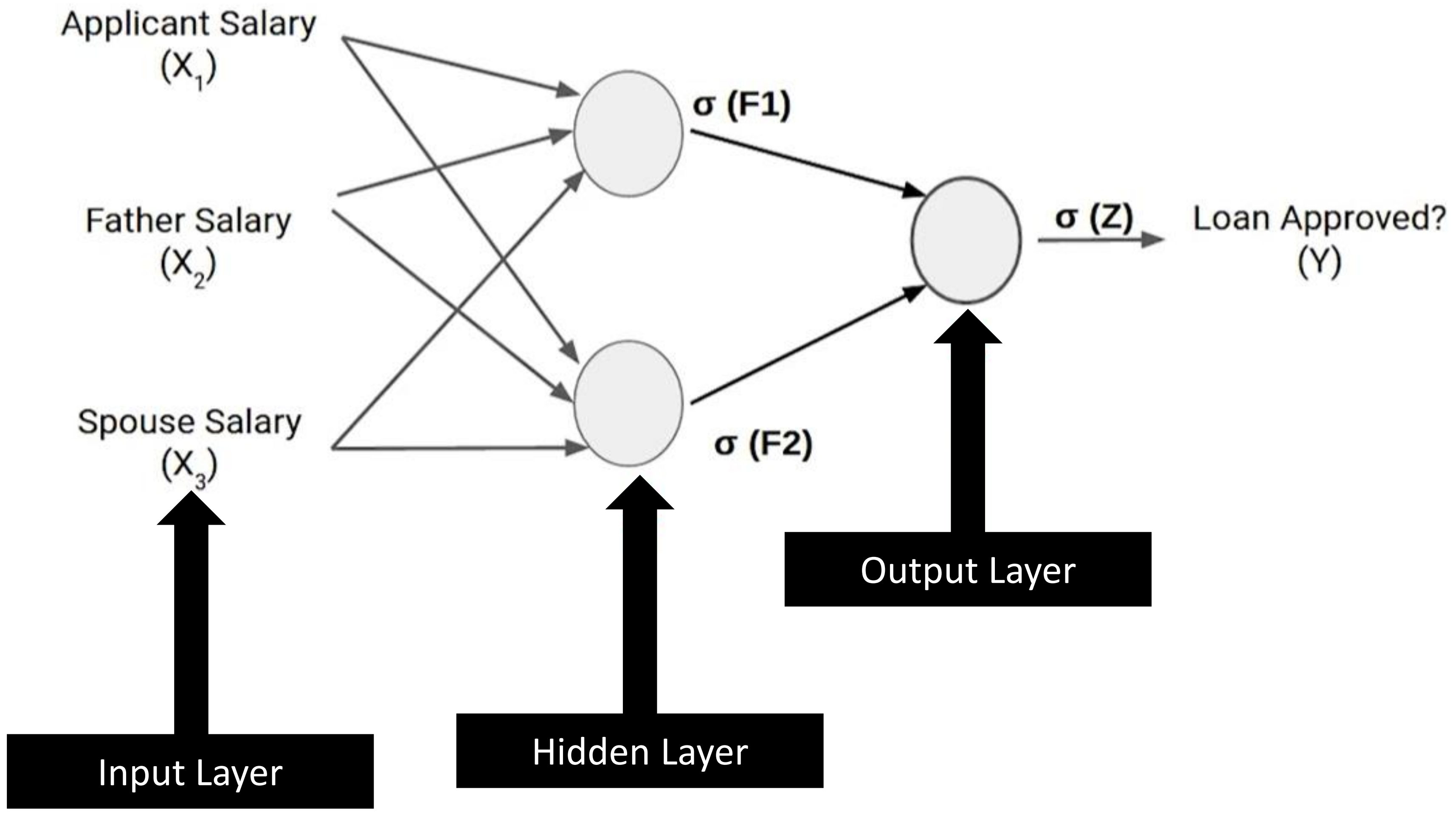
Single Layer Perceptron

Here we have only one neuron in the hidden layer.



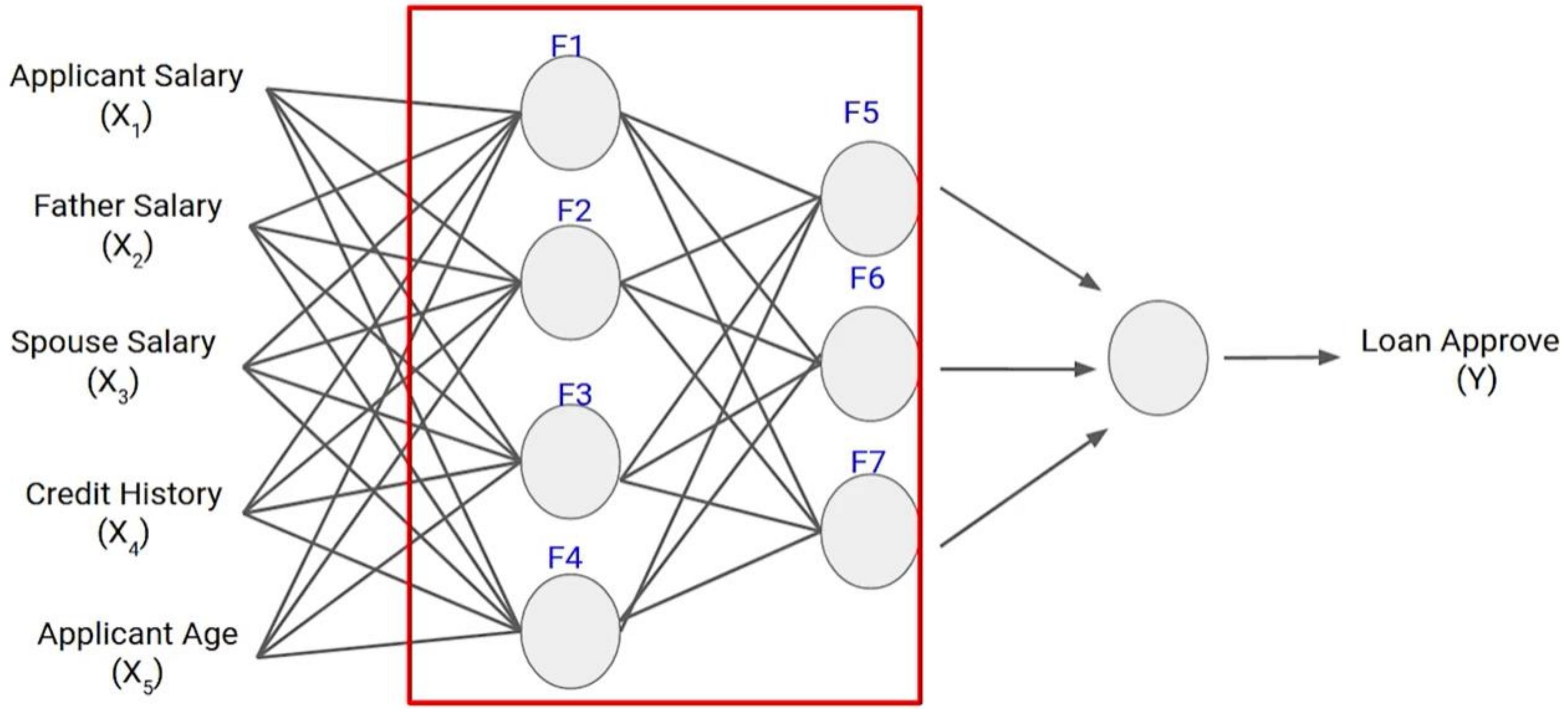
Multi Layer Perceptron

We can have more than one neuron in the hidden layer.



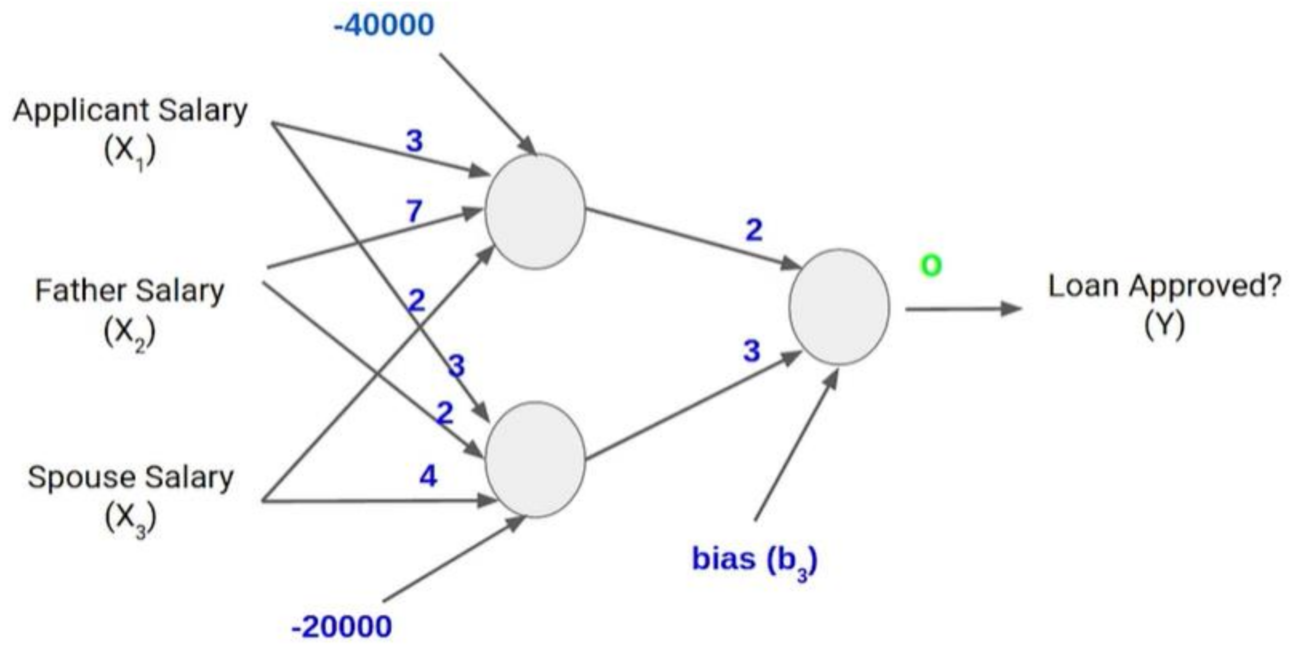
Multi Layer Perceptron

More than one layers can be added.



Forward & Backward Propagation

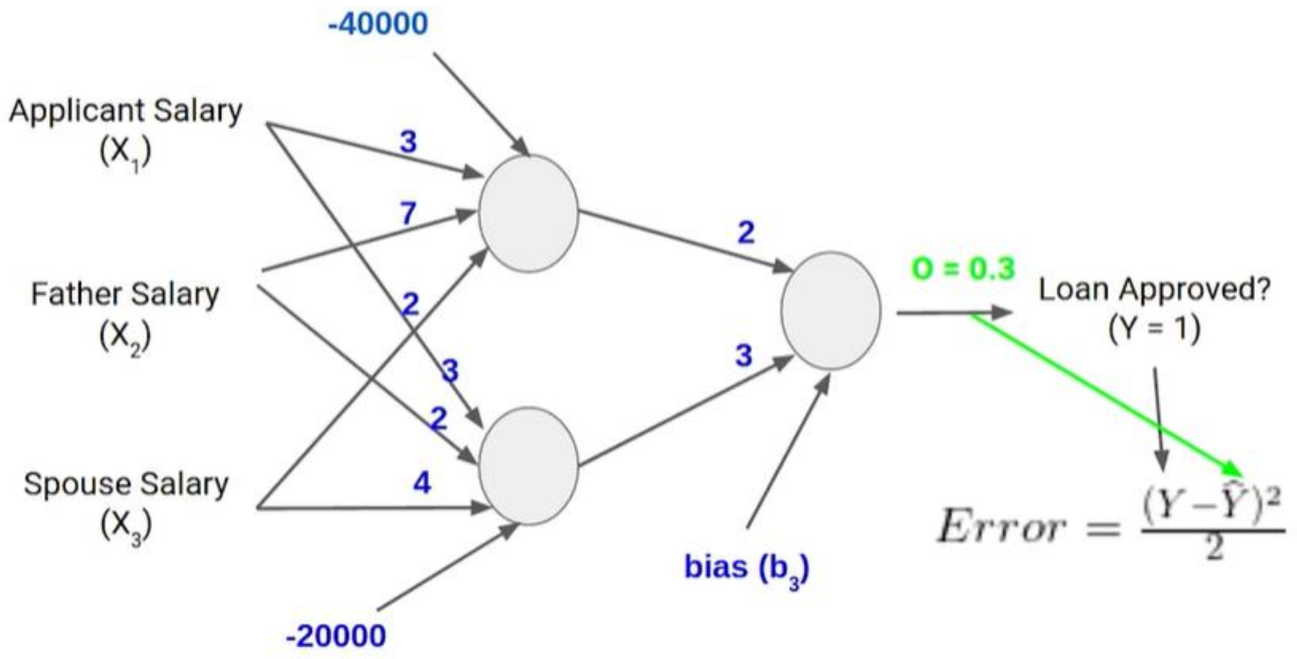
Consider the following example.



With these weights and bias values the output is obtained. This process is called the forward propagation.

Forward & Backward Propagation

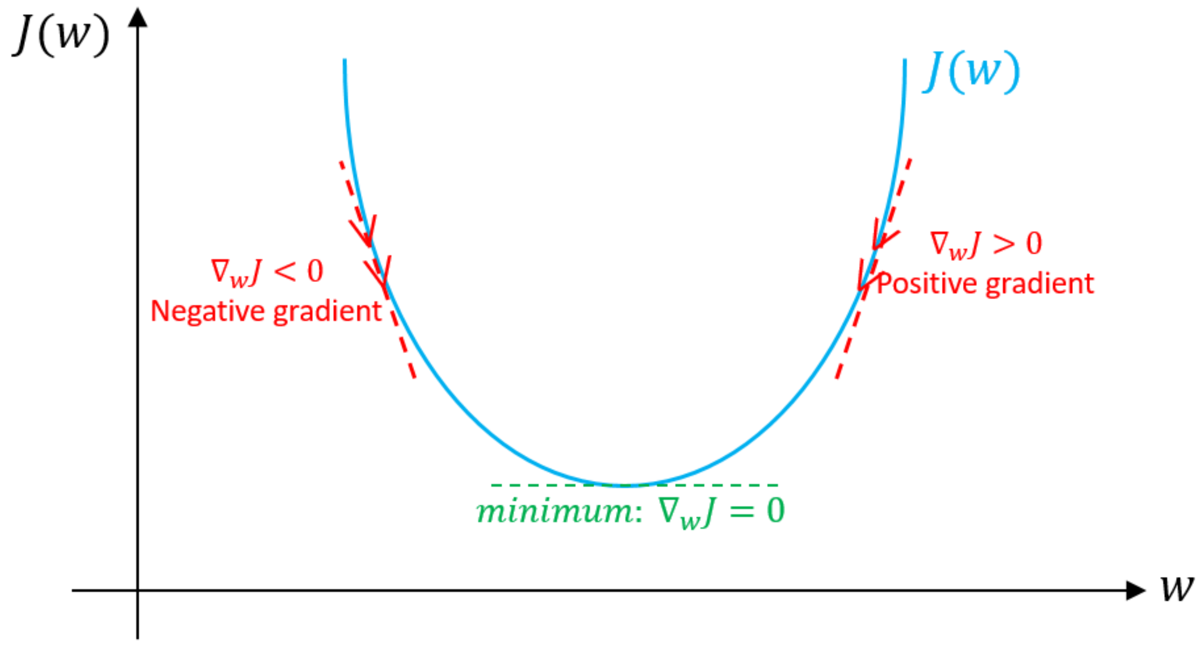
Consider the following example.



Weights and bias will be updated such that the error is minimized. This process is called the backward propagation.

Gradient Descent Algorithm

This is the algorithm which is used for minimizing the errors by adjusting the weights and bias values.



Learning Rate

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Now,

$$\frac{\partial}{\partial \theta} J_\theta = \frac{\partial}{\partial \theta} \frac{1}{2m} \sum_{i=1}^{m} \left[h_\theta(x_i) - y_i\right]^2$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i\right) \cdot \frac{\partial}{\partial \theta_j} \left(\theta x_i - y_i\right)$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \sum_{i=1}^{m} \left[\left(h_\theta(x_i) - y\right) x_i\right]$$

Therefore,

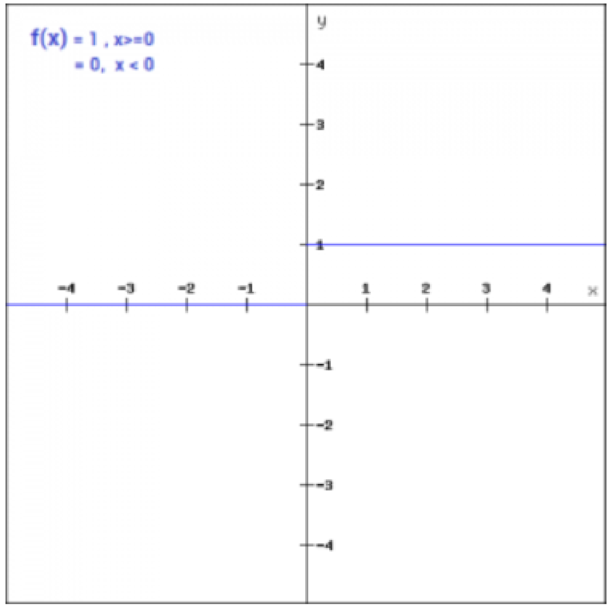
$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left[\left(h_\theta(x_i) - y_i\right) x_i\right]$$

This θ_j can be the weight or bias

This updating process stops when the error is not further minimized or when the number of iterations is completed.

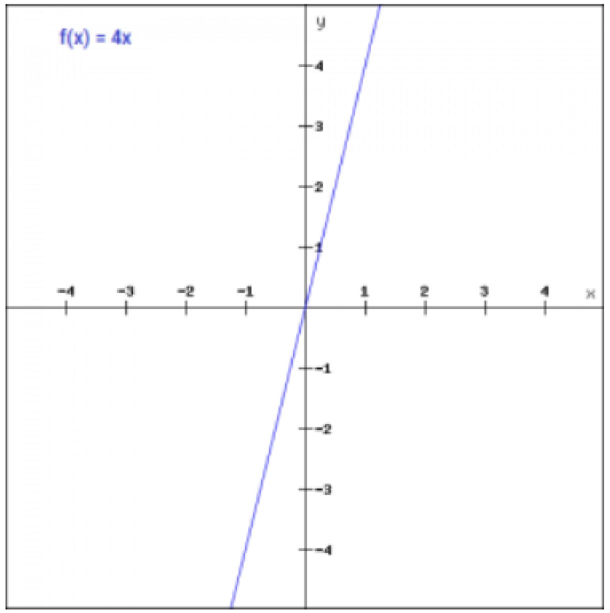
Binary stepwise function

If the input to the activation function is greater than a threshold, then the neuron is activated, else it is deactivated,



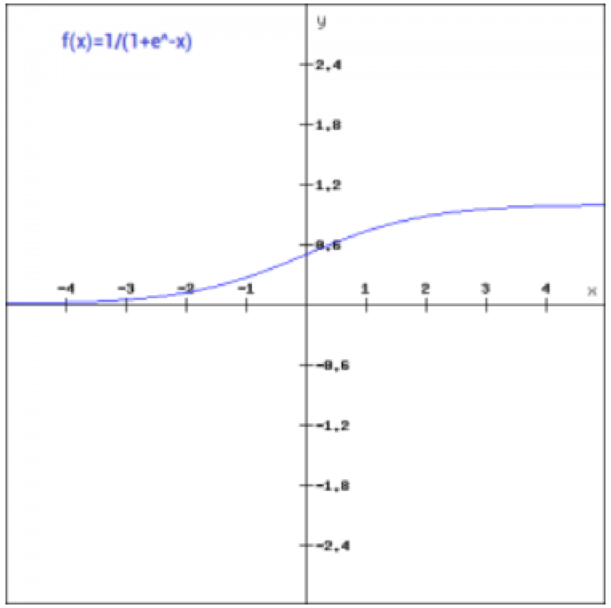
Linear activation function

The activation is proportional to the input. These activation functions are mostly used in regression problems.



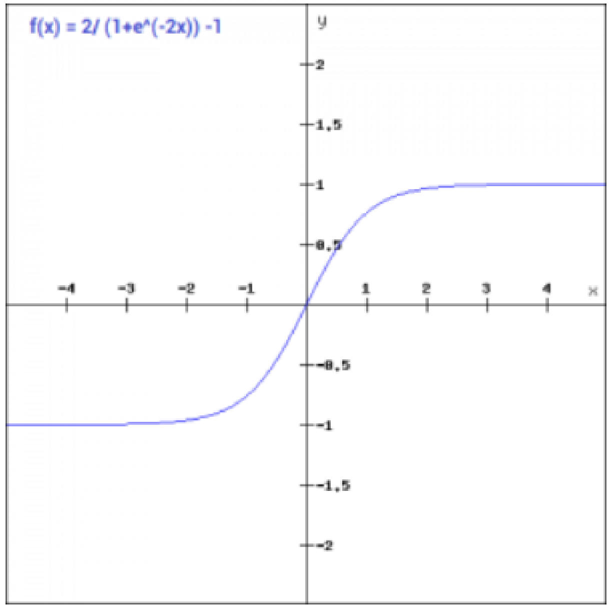
Sigmoid function

It is one of the most widely used non-linear activation function. Sigmoid transforms the values between the range 0 and 1. Usually this is used for binary classifications.



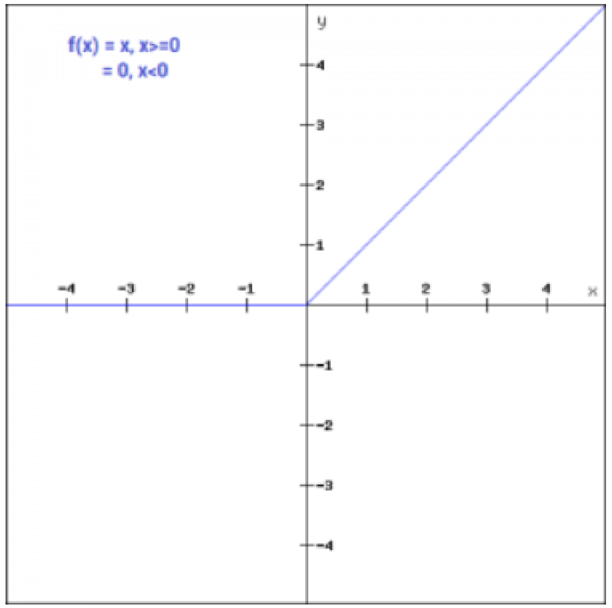
Tanh function

The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. Thus the inputs to the next layers will not always be of the same sign.



ReLU (Rectified Linear Unit) function

The ReLU function is another non-linear activation function that has gained popularity in the deep learning domain. ReLU stands for Rectified Linear Unit. The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.



This means that the neurons will only be deactivated if the output of the linear transformation is less than 0. The plot below will help you understand this better-

Softmax function

Softmax function is often described as a combination of multiple sigmoids. We know that sigmoid returns values between 0 and 1, which can be treated as probabilities of a data point belonging to a particular class. Thus sigmoid is widely used for binary classification problems.

The softmax function can be used for multiclass classification problems. This function returns the probability for a datapoint belonging to each individual class.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Loss Function

A loss function is used to optimize the parameter values in a neural network model. Loss functions map a set of parameter values for the network onto a scalar value that indicates how well those parameter accomplish the task the network is intended to do.

Regression Problem:

Loss Function: Mean Squared Error (MSE).

Binary Classification Problem:

Loss Function: Cross-Entropy, also referred to as Logarithmic loss.

Multi-Class Classification Problem:

Loss Function: Cross-Entropy, also referred to as Logarithmic loss.

Optimizer

The loss function just tells the optimizer when it's moving in the right or wrong direction. Optimizers are Classes or methods used to change the attributes of our machine/deep learning model such as weights and learning rate in order to reduce the losses. Optimizers help to get results faster.

- Adam (Adaptive Moment Estimation)
- SGD (Stochastic Gradient Decent)
- RMSprop
- Adadelta
- Adagrad
- Adamax
- Nadam
- Ftrl