Fully Connected Neural Networks

Audun Wigum Arbo, Christian Echtermeyer, Even Dalen

Perceptron

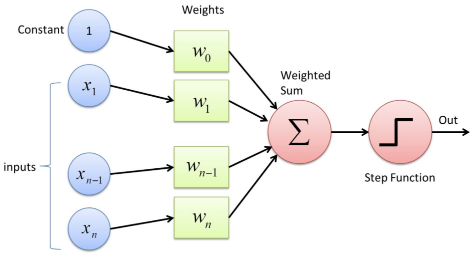
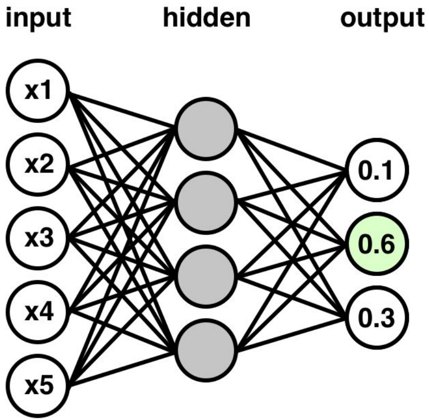


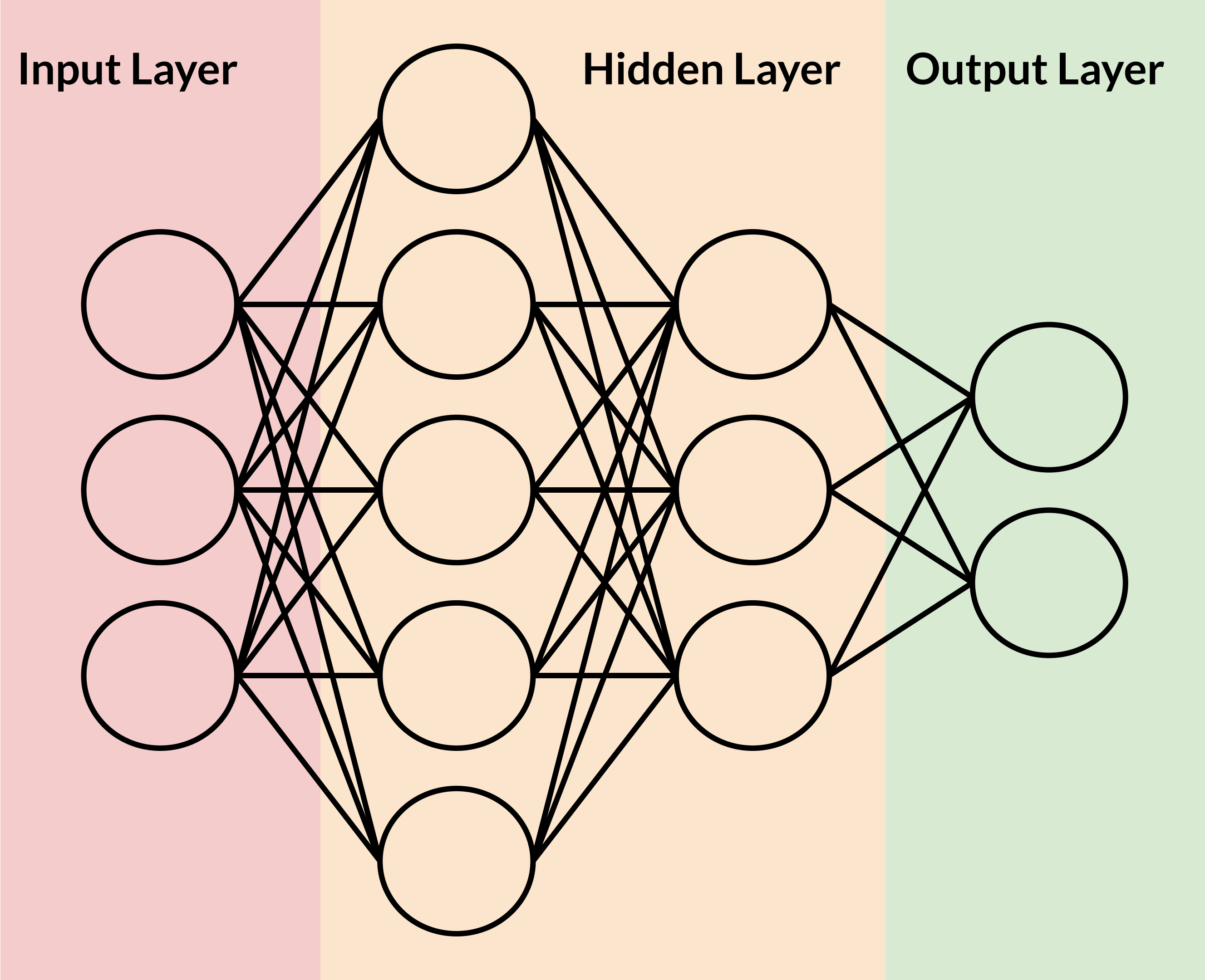
Fig : Perceptron

- The building block of a artificial neural network
- A simple binary classifier
- The input is a set of values (x_0 to x_n)
- Each edge has a weight that is multiplied with the input at that edge.
- A weighted sum is calculated and sent to an activation function.
- The result of the activation function is the output of a perceptron.

Artificial Neural Network



- One input and one output layer. Layers between are called hidden layers.
- Every unit on each layer is connected to every unit of the next layer (fully connected).
- Works similar to the perceptron. Each edge has a weight, and output of each neuron is the result of an activation function.
- The output of each neuron propagates forward through the network to the output layer.


Input Layer
Hidden Layer
Output Layer

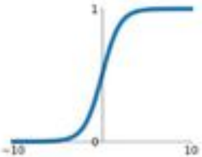
Prediction - forward propagation

- A neural network can represent an estimate **h(x)** of a concept **c(x)**
- The values of the input layer acts as the input **x** of **h**
- The values of the output layer acts as the output, **h(x)**

Activation Functions

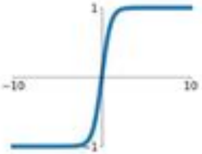
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



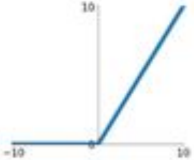
tanh

$\tanh(x)$



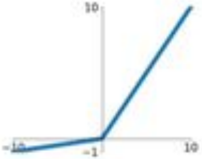
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

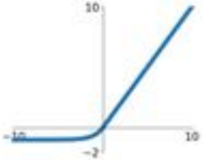


Maxout

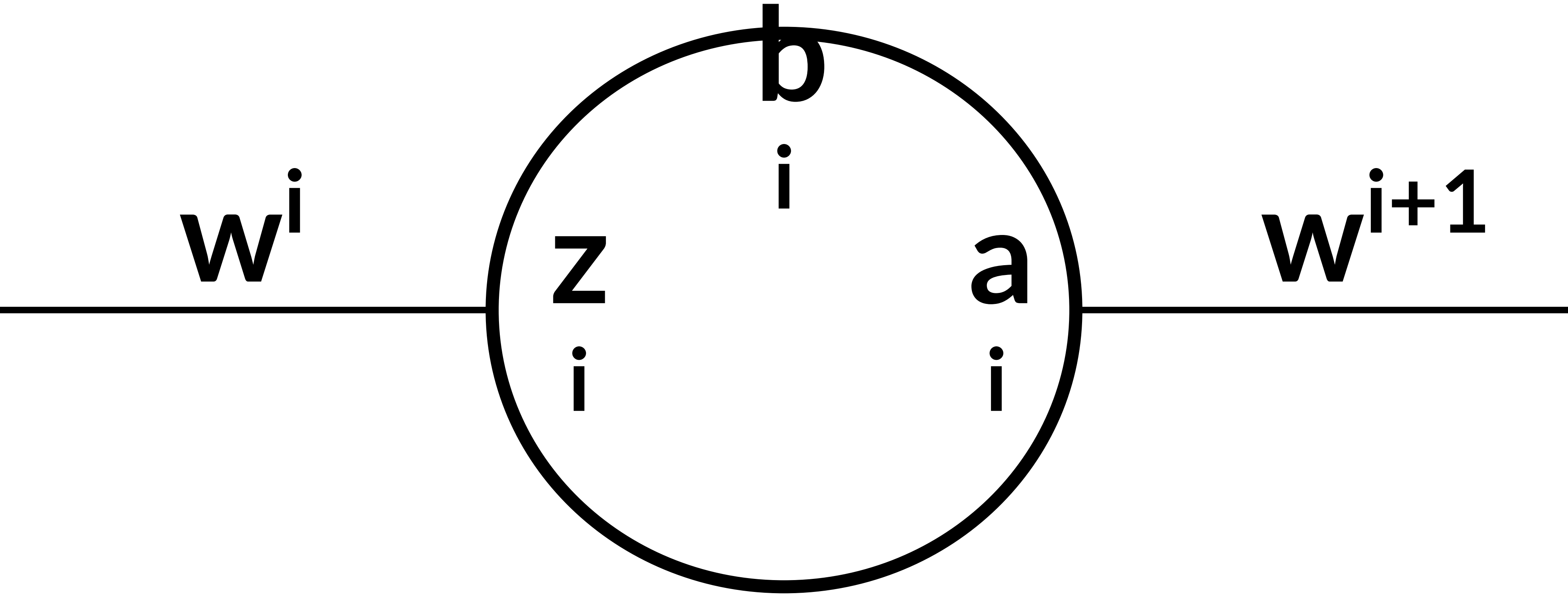
$\max(w_1^T x + b_1, w_2^T x + b_2)$

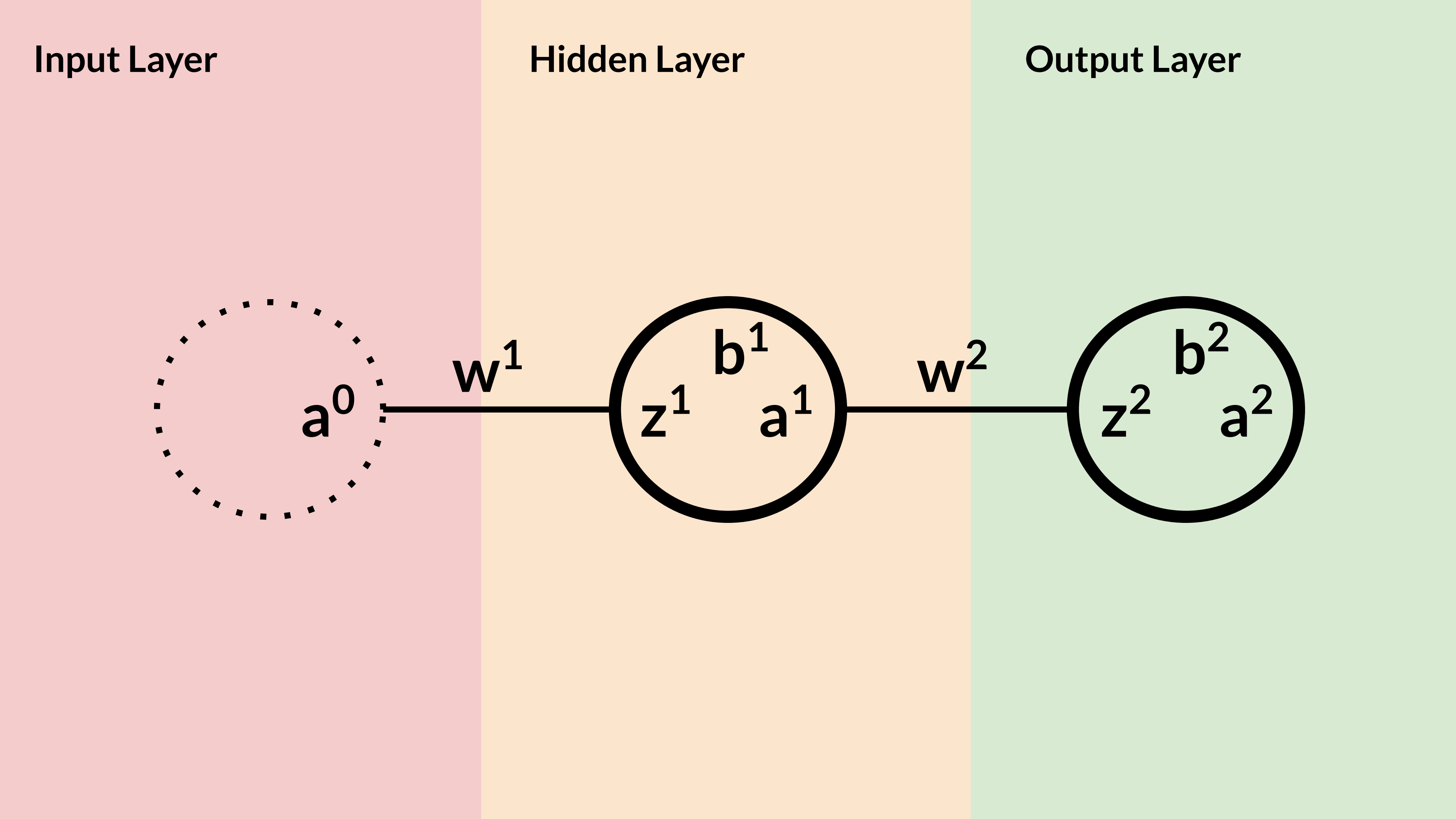
ELU

$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

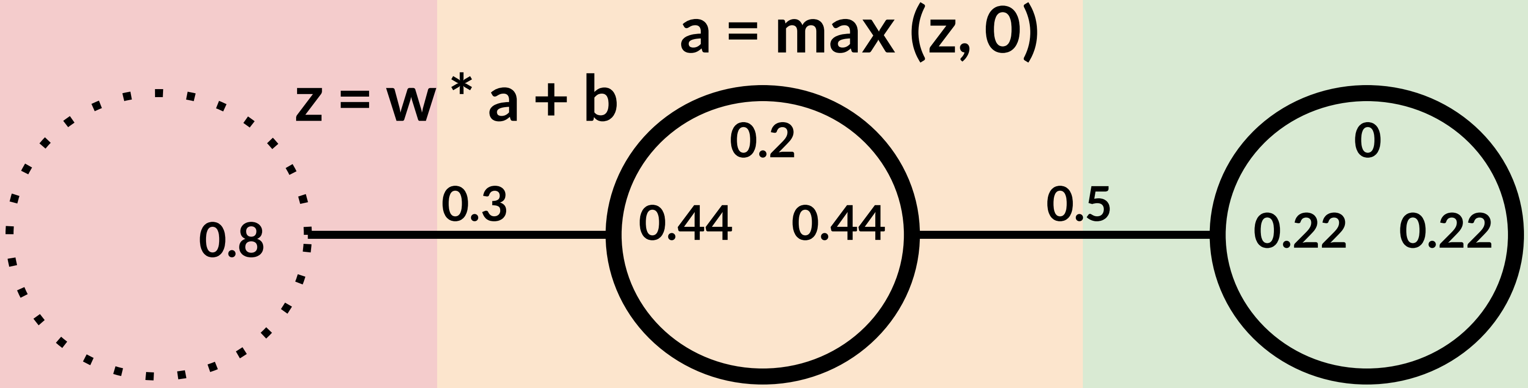


Quick Notation Guide:




Input Layer
Hidden Layer
Output Layer
a0
w1
b1
z1
a1
w2
b2
z2
a2

ReLU Example



a = max (z, 0)

Training

- The network learns by backpropagating the error of the network
- The error is a function of the difference between the expected result and the actual result

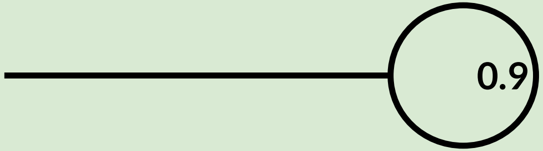
Training: Loss functions

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

Output Layer

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

Desired result: 0.5

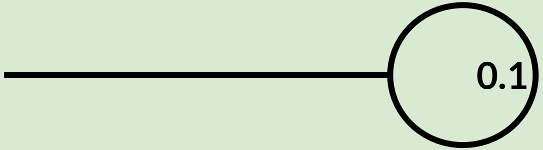


$$(0.5 - 0.9)^2 = 0.16$$

$$\boxed{= 0.16}$$

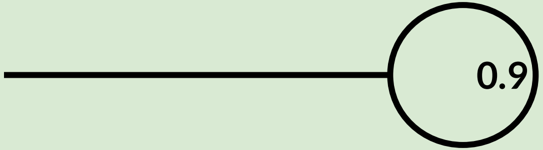
Output Layer

Desired result: 1.0



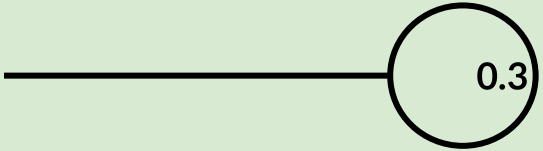
$(1.0 - 0.1)^2 = 0.81$

Desired result: 0.5



$(0.5 - 0.9)^2 = 0.16$

Desired result: 2.0



$(2.0 - 0.3)^2 = 2.89$

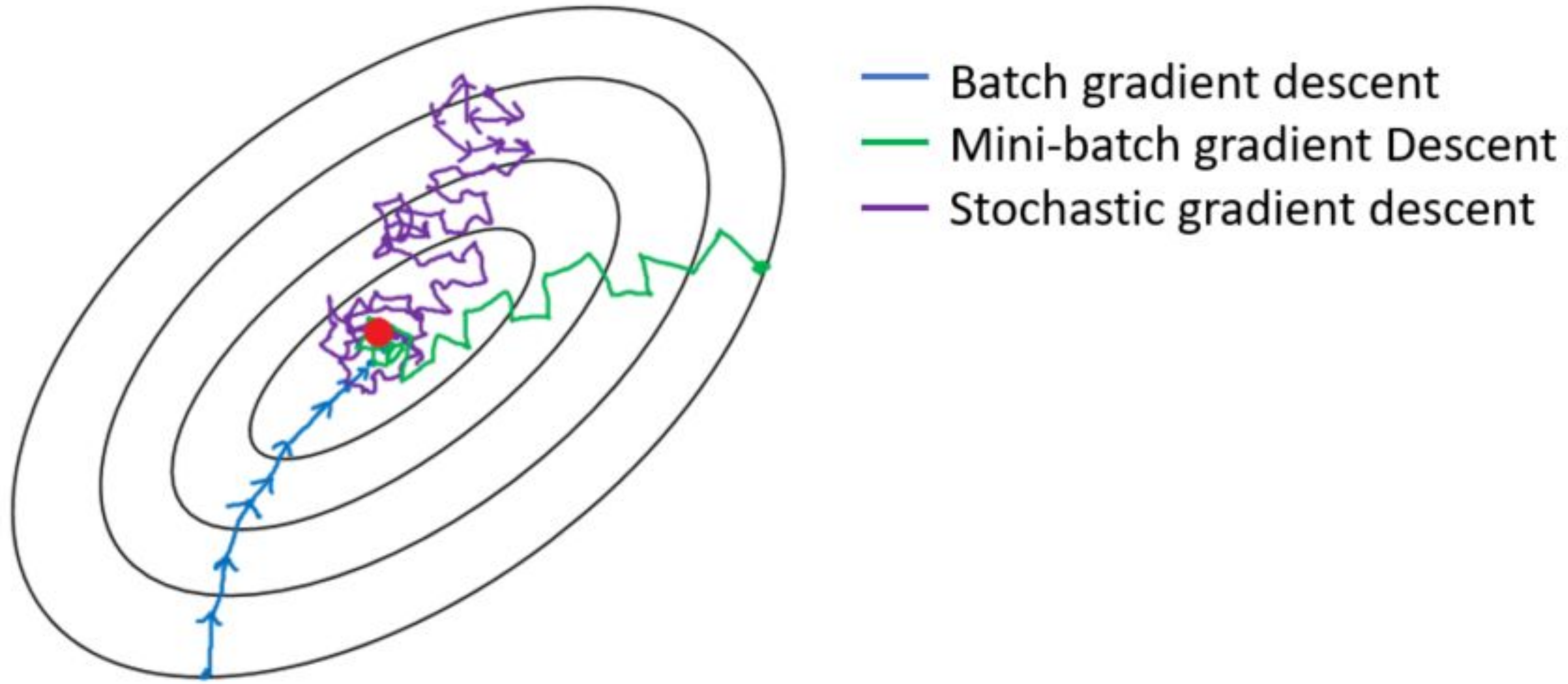
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

$$\frac{(0.81+0.16+2.89)}{3}$$

$$\boxed{= 1.286}$$

Training

- Once the network has a measure of "correctness" it can then update the weights and biases of the network to be more correct.
- This is often done with a batch of random samples averaged out before a small step is taken in the "direction" that best optimizes the samples.


Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent

Summary

- A Neural Network consists of neurons
- A neuron simply holds a number value (activation)
- In a fully connected network every neuron in one layer is connected to every neuron in the next.
- Every connection has a weight
- The activation of a neuron is decided by the weight and activation of the previous layer and the bias. This result is then passed through an activation function.
- A FCNN can be divided into three layers: Input, Hidden and Output
- The network learns by backpropagating the error

References

- http://neuralnetworksanddeeplearning.com/
- https://towardsdatascience.com/under-the-hood-of-neural-networks-part-1-fully-connected-5223b7f78528
- https://towardsdatascience.com/coding-up-a-neural-network-classifier-from-scratch-977d235d8a24