Introduction to Neural Networks

Feedforward, Backpropagation, and Gradient Descent

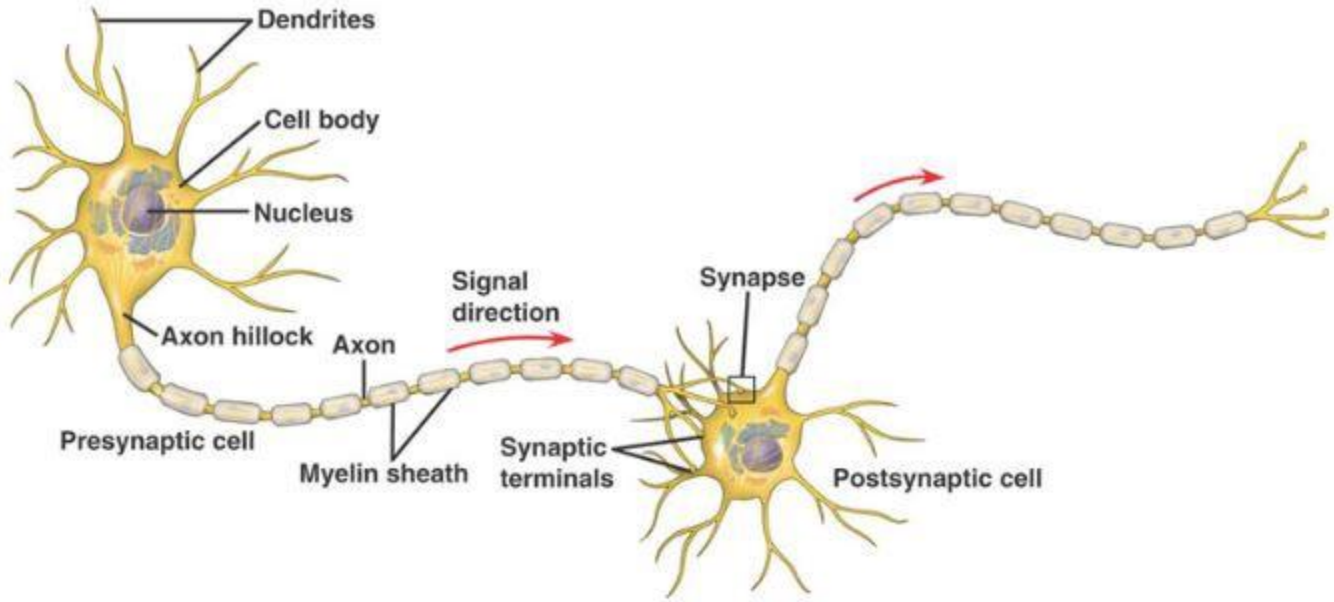
By

Oddy Virgantara Putra, M.T.

My Profile

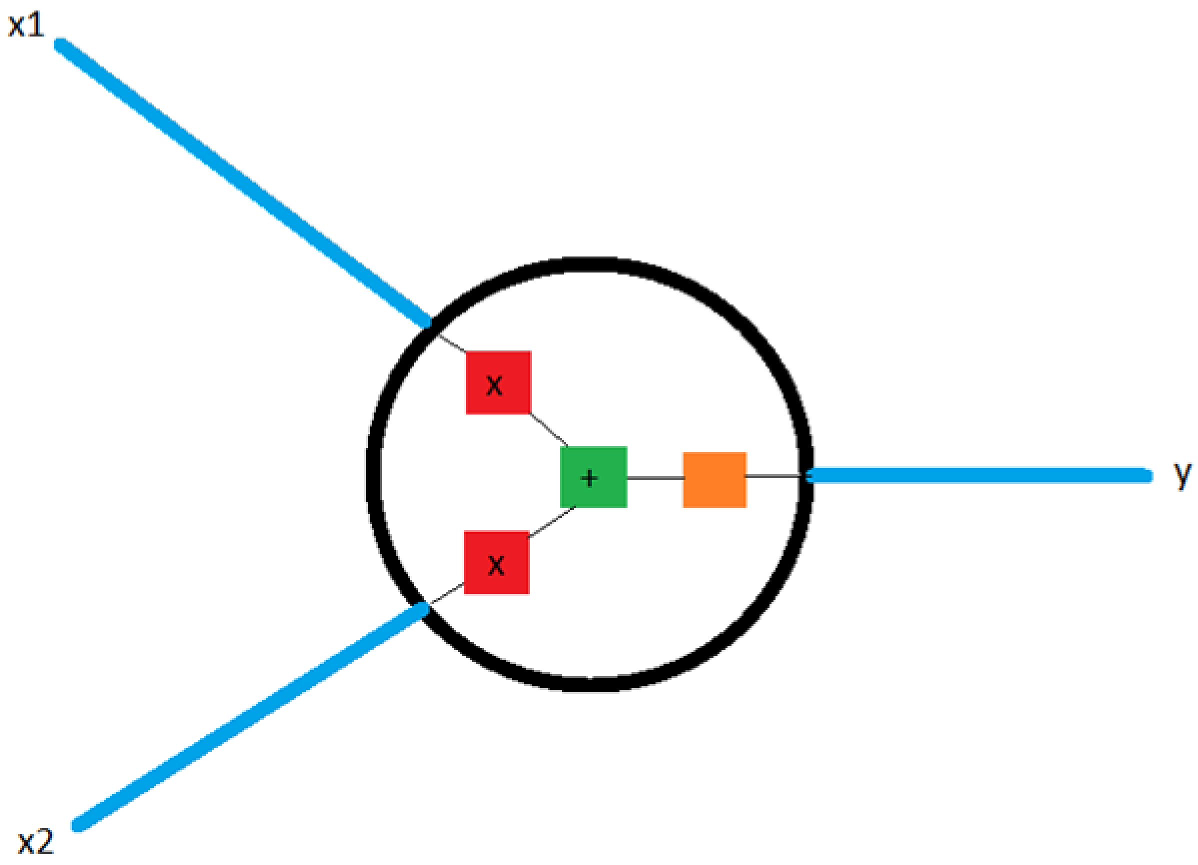
- The name is Oddy Virgantara Putra
- From Kediri
- Information System Bachelor degree from ITS – Surabaya (2007)
- Electrical Engineering Master degree from ITS – Surabaya (2015)
- Alumni LPDP PK-40
- Research Interest: Computer Vision and Artificial Intelligence
- Lecturer in Department of Informatics, Universitas Darussalam Gontor, Ponorogo

Intuition



Source: http://biomedicalengineering.yolasite.com/neurons.php

The Neuron



Three amazing things are happening in here, the **weights**, the **bias**, and the **activation function**

The Neuron – Weights

- The red dots represent **weights** $\{w_1, w_2\}$
- The inputs $\{x_1, x_2\}$ are multiplied with their respective **weights**
- We have:

$$x_1 \rightarrow x_1 * w_1$$

$$x_2 \rightarrow x_2 * w_2$$

The Neuron – Bias

- The green dot represents **bias** b
- The results from **inputs** and **weights** multiplication are added with **bias**
- We have:

$$x_1 * w_1 + x_2 * w_2 + b$$

The Neuron – Activation Function

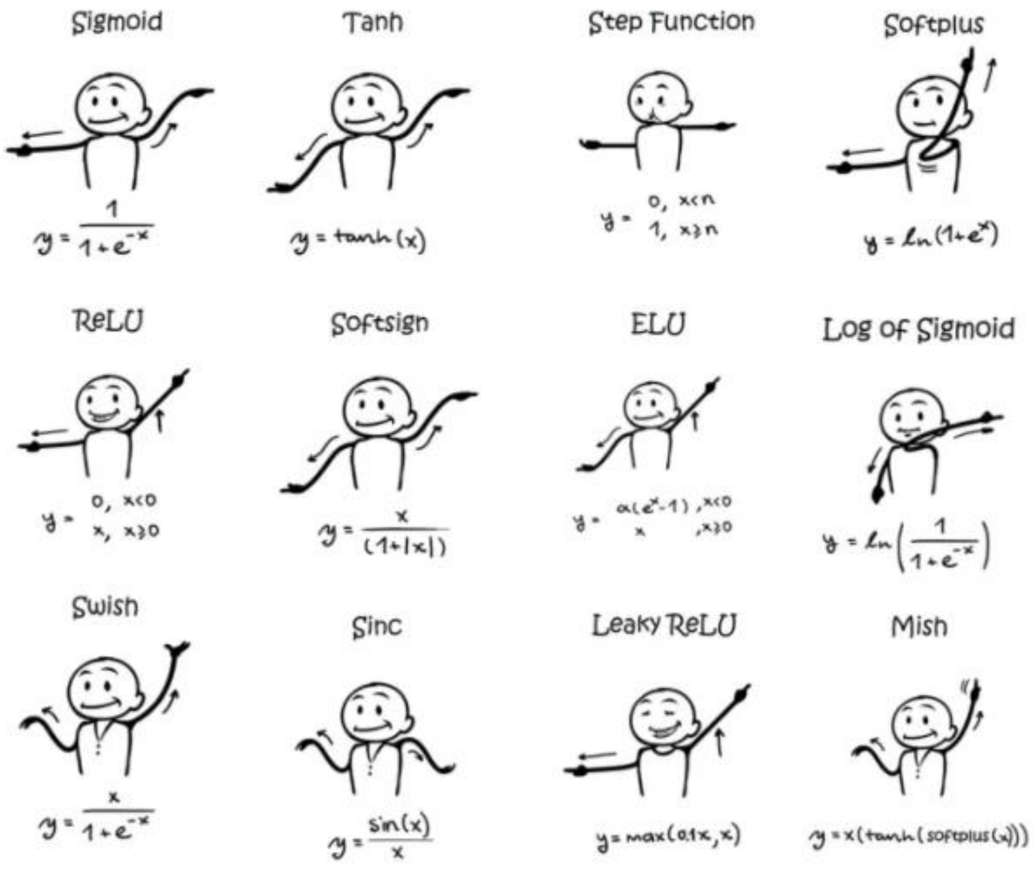
- From above equations, the sum is passed through an activation function, the orange dot.
- We have

$$y = f(x_1 * w_1 + x_2 * w_2 + b)$$

- Activation Function f is function that transforms the input into a readable, predictable, and nice presentation output.

The Neuron – Activation Function – cont'd

- There are many activation functions that are commonly used in neural network. For example



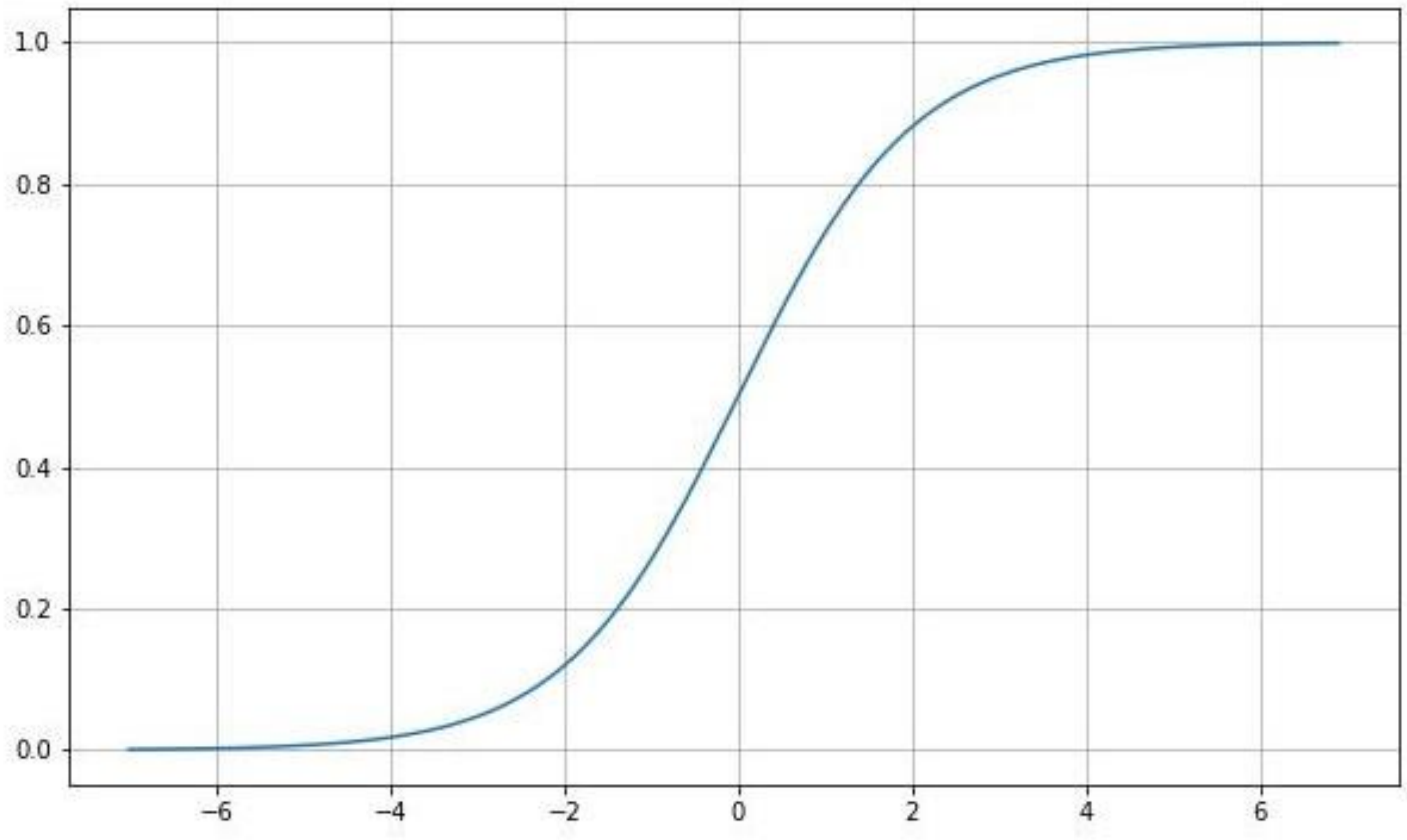
Img source : https://miro.medium.com/max/1400/1*TEhut1cV5r63K6LbvdoaaQ.png

The Sigmoid

- One of the most commonly used activation function in neural network
- Transform any inputs $[-\infty, \infty]$ into $[0,1]$
- The equation is:

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

The Sigmoid-cont'd



Example 1

- Given two input features $\vec{x} = \{1,1\}$, $\vec{w} = \{2,3\}$, and $b = 1$, we can calculate them using dot product. Thus, we have:

$$\begin{aligned} \vec{w} \cdot \vec{x} + b &= (w_1 * x_1) + (w_2 * x_2) + b \\ &= (2 * 1) + (3 * 1) + 1 \\ &= 6 \end{aligned}$$

- Let's put above result into sigmoid

$$\begin{aligned} y &= f(\vec{w} \cdot \vec{x} + b) \\ &= f(6) \\ &= 0.997 \end{aligned}$$

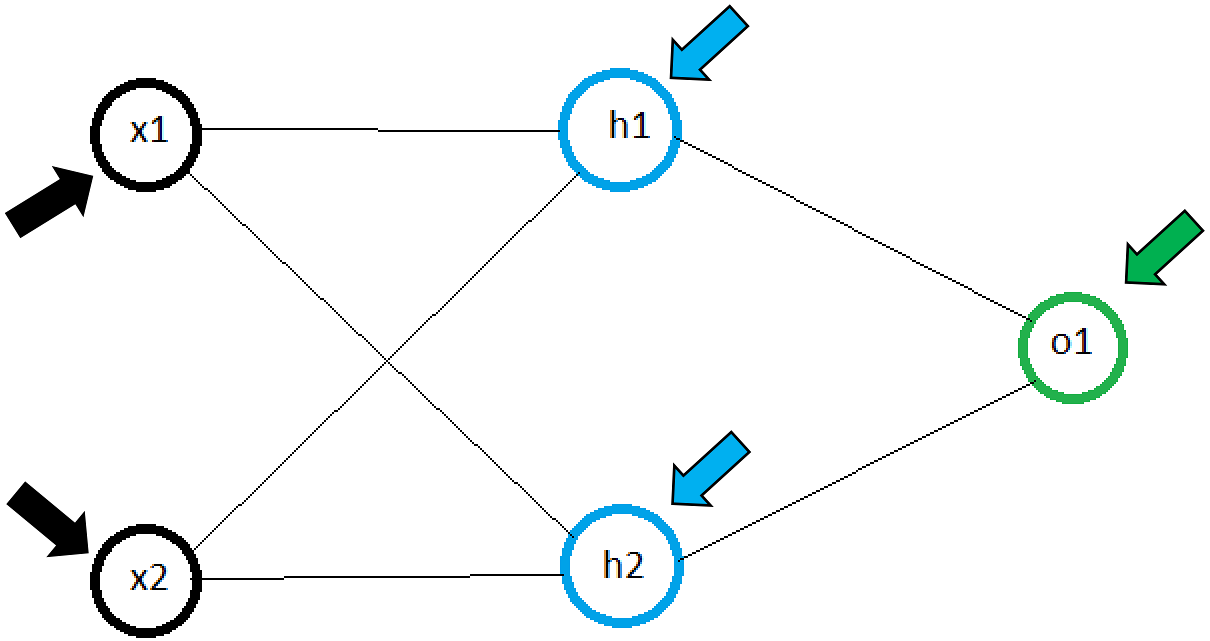
Feedforward

- From Example 1 above, the inputs $\vec{x}$ was calculated with weights $\vec{w}$ and added with bias b. Subsequently, the result was passed through a sigmoid function. Finally, we got y value at 0.997.
- This whole process is called as **Feedforward**

That was neuron

Where is the network?

Neural Network – a simple network



- It is a bunch of neurons.
- It is a fully-connected network.

It contains:

- Input nodes/neurons – Input Layer
- Hidden nodes – Hidden layer
- Output node – Output layer

Hidden Layer

- Two neurons have been added into the network, h_1 and h_2 .
- This layer is called **hidden layer**
- A neural network may contains of one or more hidden layers.

Example 2

- Given two input features $\vec{x} = \{1,1\}$, $\vec{w} = \{2,3\}$, and $b = 1$.
- If you realize, both h_1 and h_2 are equal to the y value from Example 1.
- Thus, we can formulate h_1 and h_2 as follows:

$$\begin{aligned} h_1 = h_2 &= f(\vec{w} \cdot \vec{x} + b) \\ &= f((2 * 1) + (3 * 1) + 1) \\ &= f(6) \\ &= 0.997 \end{aligned}$$

Example 3

- Given two input features $\vec{h} = \{0.997, 0.997\}$, $\vec{w} = \{1, 2\}$, and $b = 1$.
- We can calculate the output o_1 from $\vec{h}$
- From here, we have:

$$\begin{aligned} o_1 &= f(\vec{w} \cdot \vec{h} + b) \\ &= f((1 * 0.997) + (2 * 0.997) + 1) \\ &= f(3.991) \\ &= 0.981 \end{aligned}$$

That’s all for Feedforward

What now?

A Case Study

- Given a dataset containing the height and weight of some persons.
- Each data is labeled corresponding to its gender, male or female.
- The dataset as in Table 1
- We want to **predict** the gender of a person based on both **height** and **weight**
- What should we do?

Table 1. Height and Weight Data

Name	**Height**	**Weight**	**Gender**
Budi	170	80	M
Yanto	172	81	M
Susi	160	64	F
Siti	155	59	F
Agus	167	74	M
Asih	156	57	F

Let’s learn about features and class

Feature and Class

- Feature is a **characteristic** of an entity that distinguish itself from others.
- Similar terms of **feature**: attributes, independent variables, or properties.
- Class is a **predicted variable** of given data points.
- Similar terms: targets, outputs, labels, dependent variables, or categories.

Data Preprocessing

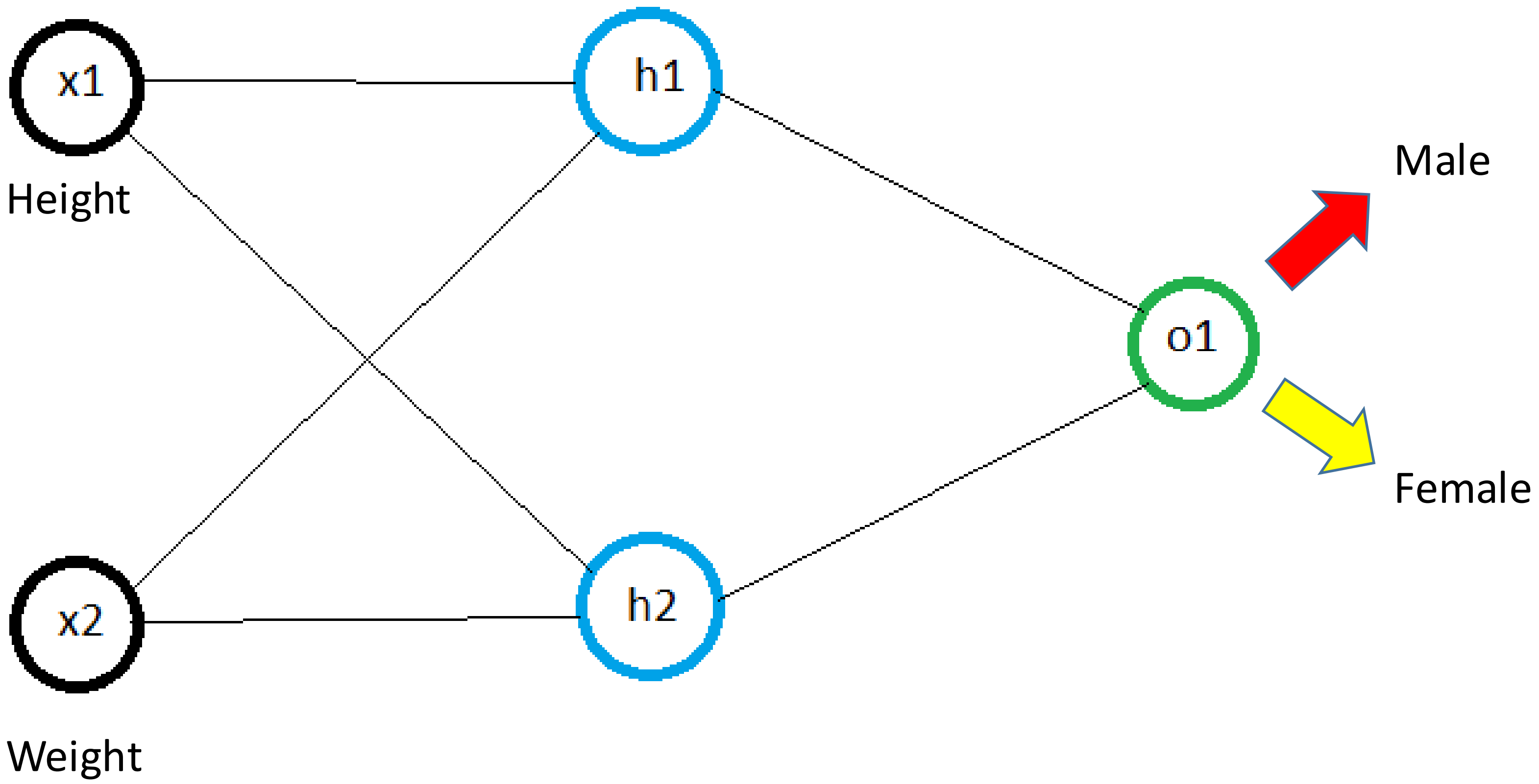
Name	Height	Weight	Gender
Budi	170	80	M
Yanto	172	81	M
Susi	160	64	F
Siti	155	59	F
Agus	167	74	M
Asih	156	57	F

Label Encoding

Min Max Scaling

Height	Weight	Gender
0.88	0.96	1
1	1	1
0.29	0.29	0
0	0.08	0
0.7	0.71	1
0.06	0	0

Network Model



Loss Function

- A model needs to “learn” from mistake.
- Evaluate the model
- Is it good or bad
- To measure the quality of how good the model learns, we can use

Loss Function

Loss Function – cont’d

- Loss function has many forms
- We use one of them, the **MSE**
- It stands for Mean Squared Error
- MSE has a form as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yt_i - yp_i)^2$$

Are you panicking?

Don’t be

Let’s break it down

MSE – in a nutshell

- n is the **number of dataset**, as in our case, we have 6 person.
- i is the ith position or index of data
- yt is the **observed class** (the **Gender** column) respective to height and weight of a person
- yp is the **predicted class** (the o_1) from the network model
- $yt - yp$ is an error
- $(yt - yp)^2$ is an error squared
- The lower squared error, the better the network model

Training the network model – part 1

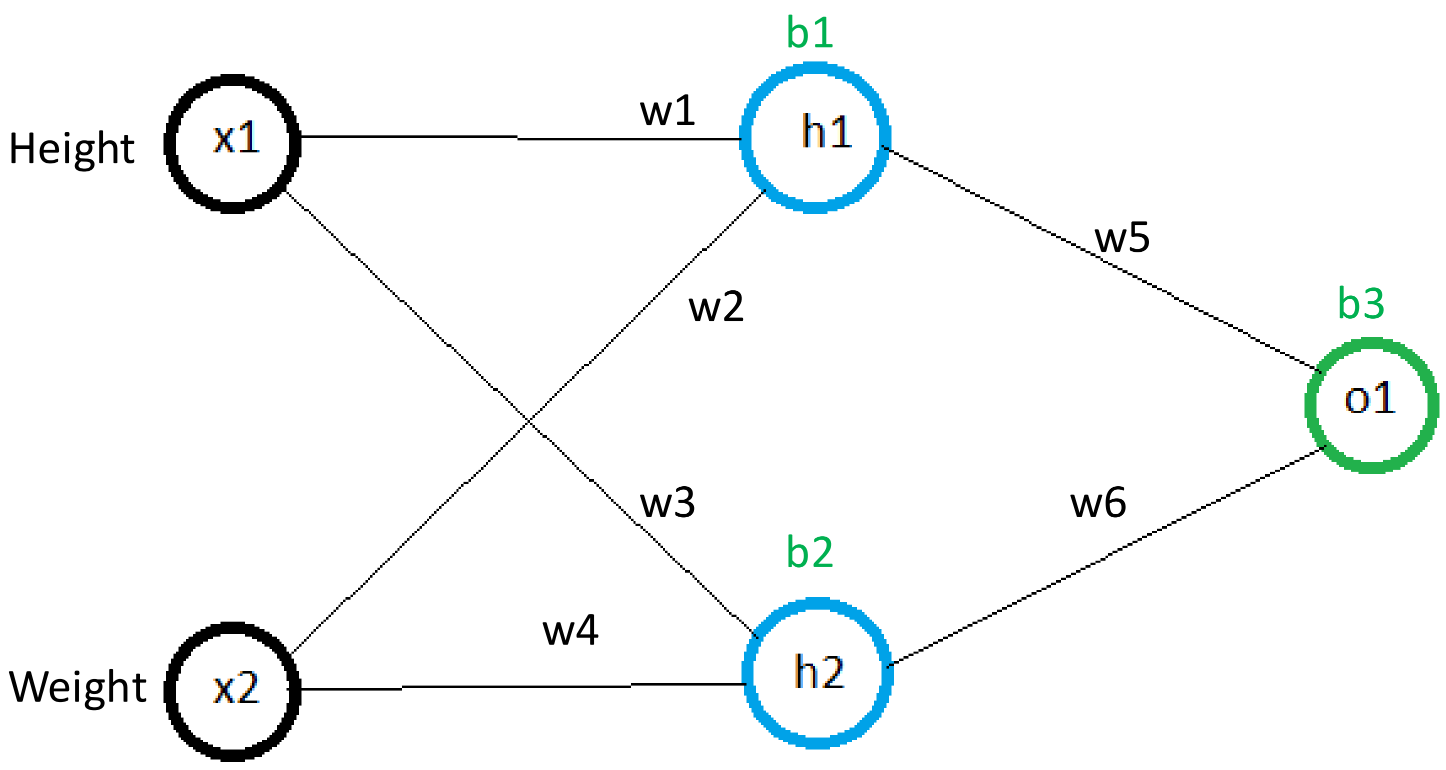
- Training is a term for model of “learning” from data
- Training in here is simply **minimizing the loss function**

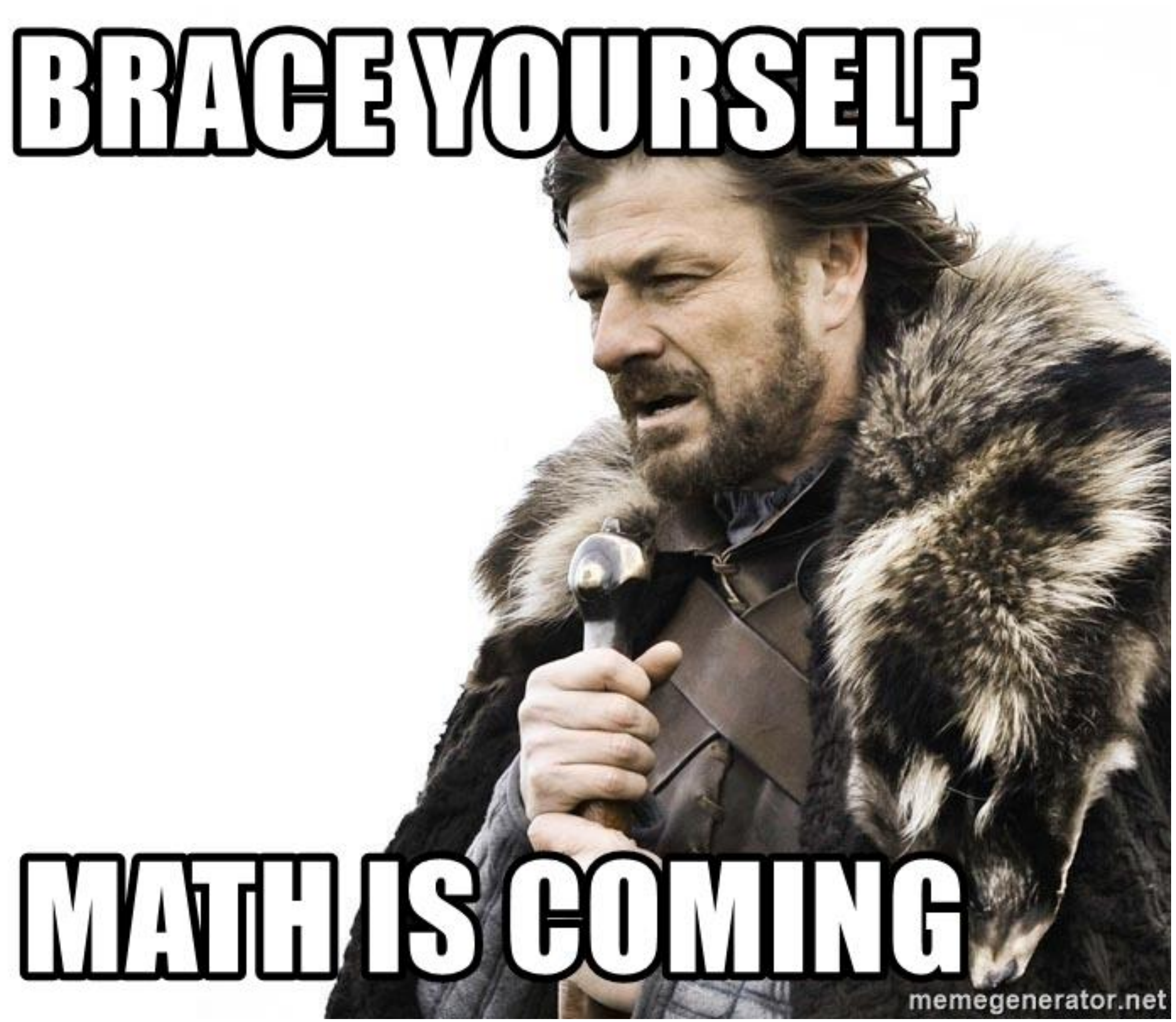
MSE

Training the network model – part 2

- The only variables in the neural network that can be changed are **weights** and **biases**
- These two variables affect the prediction performance

Neural Network Model





Img source: https://memegenerator.net/img/instances/78196533/brace-yourself-math-is-coming.jpg

Loss Function – cont'd

- Let's call Loss Function as L
- Since weights and biases only affected by L, we have 9 parameters to update as follows :

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

- First, we need to update w_1
- By using partial derivative, we can find w_1 by:

$$\frac{\partial L}{\partial w_1}$$

Loss Function – cont'd

- Let's break down $\frac{\partial L}{\partial w_1}$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial yp} * \frac{\partial yp}{\partial w_1}$$

- Remember,

$$yp = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

- Since w_1 only affected by h_1, we have:

$$\frac{\partial yp}{\partial w_1} = \frac{\partial yp}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

Loss Function – cont'd

$$\frac{\partial yp}{\partial h_1} = w_5 * f'(w_5 h_1 + w_6 h_2 + b_3)$$

- Previous step can be applied to $\frac{\partial h_1}{\partial w_1}$,

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

Then,

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 x_1 + w_2 x_2 + b_1)$$

Loss Function – cont'd

- Finally, we can update w_1, using:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial yp} * \frac{\partial yp}{\partial h_1} * \frac{\partial h1}{\partial w_1}$$

- Do this for every parameters.
- If you observe above equation, it is like calculating in backward.
- Such process is called

Backpropagation

That's all for backpropagation

What next?

How long do we need to update w1?

This problem can be solved by

Gradient Descent

We'll discuss Gradient Descent the next session

Any questions?