


# Data Science/Machine Learning/Deep Learning Workshop

*Nueva Ecija University of Science and Technology  
(NEUST)*

Rodolfo C. Raga Jr., PhD CS

*November 21-23, 2021*

A solid orange shape that starts as a thin wedge at the bottom left and expands diagonally upwards to the right, filling the bottom right corner of the slide.

# Day 1 : Basic Concepts, Intuitions, and Foundations

## **Topic 1: Introduction to Data Science, Machine Learning, and Deep Learning**

- ☐ What is Data Science
- ☐ What is Machine Learning
- ☐ What is Deep Learning

## **Topic 2: Intro to Python, Jupyter Notebook, Google Colab, Scikit-learn, Tensorflow, and Keras**

- ☐ Why use Python
- ☐ Jupyter Notebook vs. Google Colab
- ☐ Understanding sklearn, Tensorflow and Keras

## **Topic 3: Fundamentals of Data Analysis using sklearn**

- ☐ Correlation and Distribution Analysis
- ☐ Data Preprocessing (Feature Selection, Data Normalization, Data Splitting)
- ☐ Building and Training of ML prediction models using sklearn

## **Topic 4: Fundamentals of Neural Networks**

- ☐ Definition and NN Architecture
- ☐ Perceptron and Multi-layer Perceptron Architecture
- ☐ Building NN prediction models using sklearn

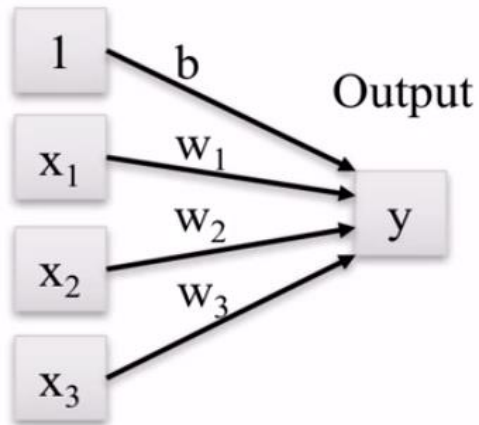
# Outline

- ◆ Definition and Architecture of Neural Networks
- ◆ Perceptron and Multi-layer Perceptron
- ◆ Building NN prediction models using sklearn

# Review of Linear and Logistic Regression...

## Linear regression

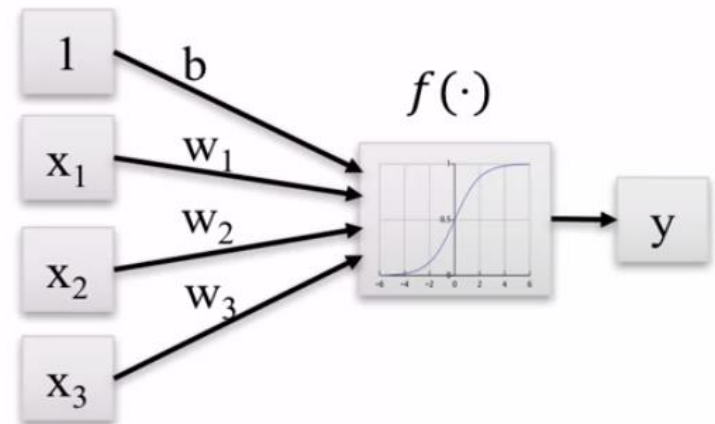
Input features



$$\hat{y} = \hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n$$

## Logistic regression

Input features



$$\hat{y} = \text{logistic}(\hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n)$$

# Linear vs. highly non-linear functions

- The objective of both LinReg and LogReg is to find the linear function

$$f: X \rightarrow Y$$

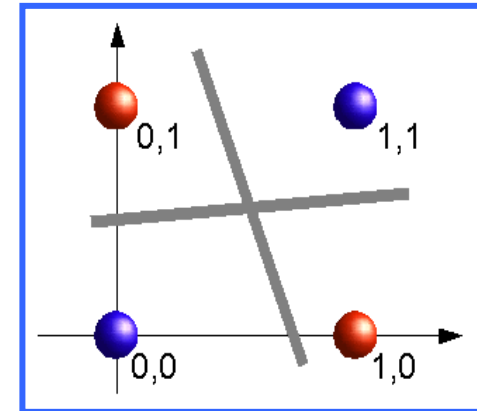
Where:

X (vector of) continuous and/or discrete vars

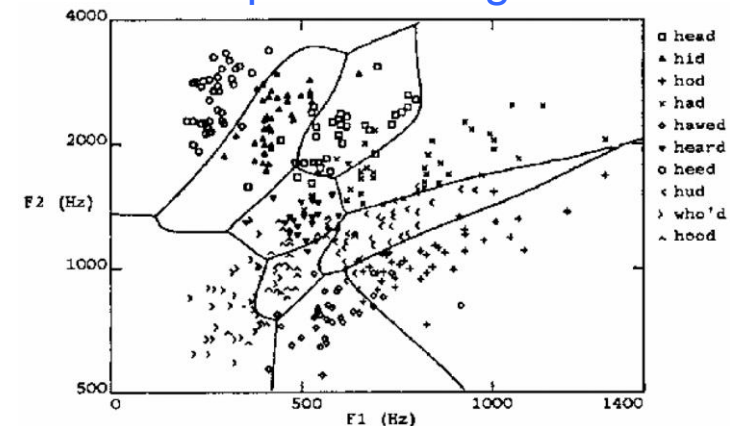
Y (vector of) continuous and/or discrete vars

- But what if  $f$  requires a non-linear function?
- In this case, we would need a new class of decision functions.

The XOR gate



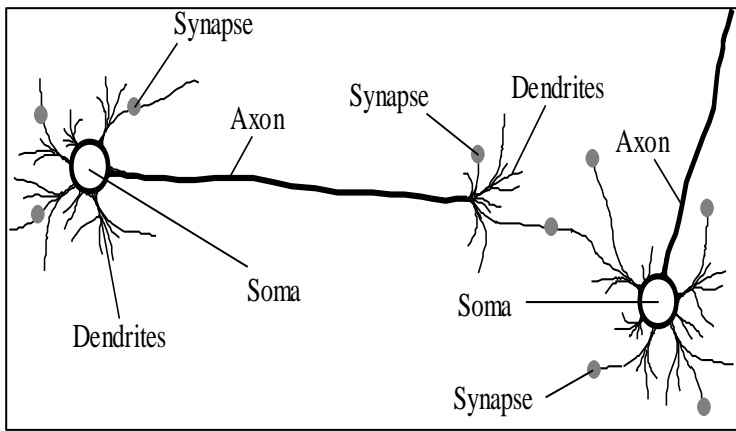
Speech recognition



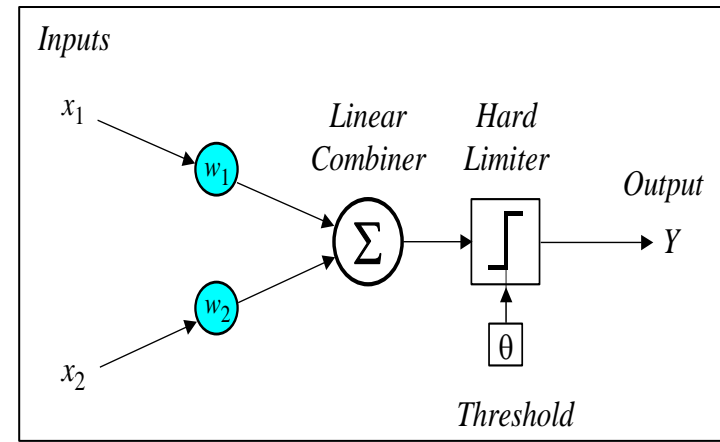
# About Artificial Neurons

Artificial neurons was inspired by the biological functions of neurons in the human brain.

Biological Neurons

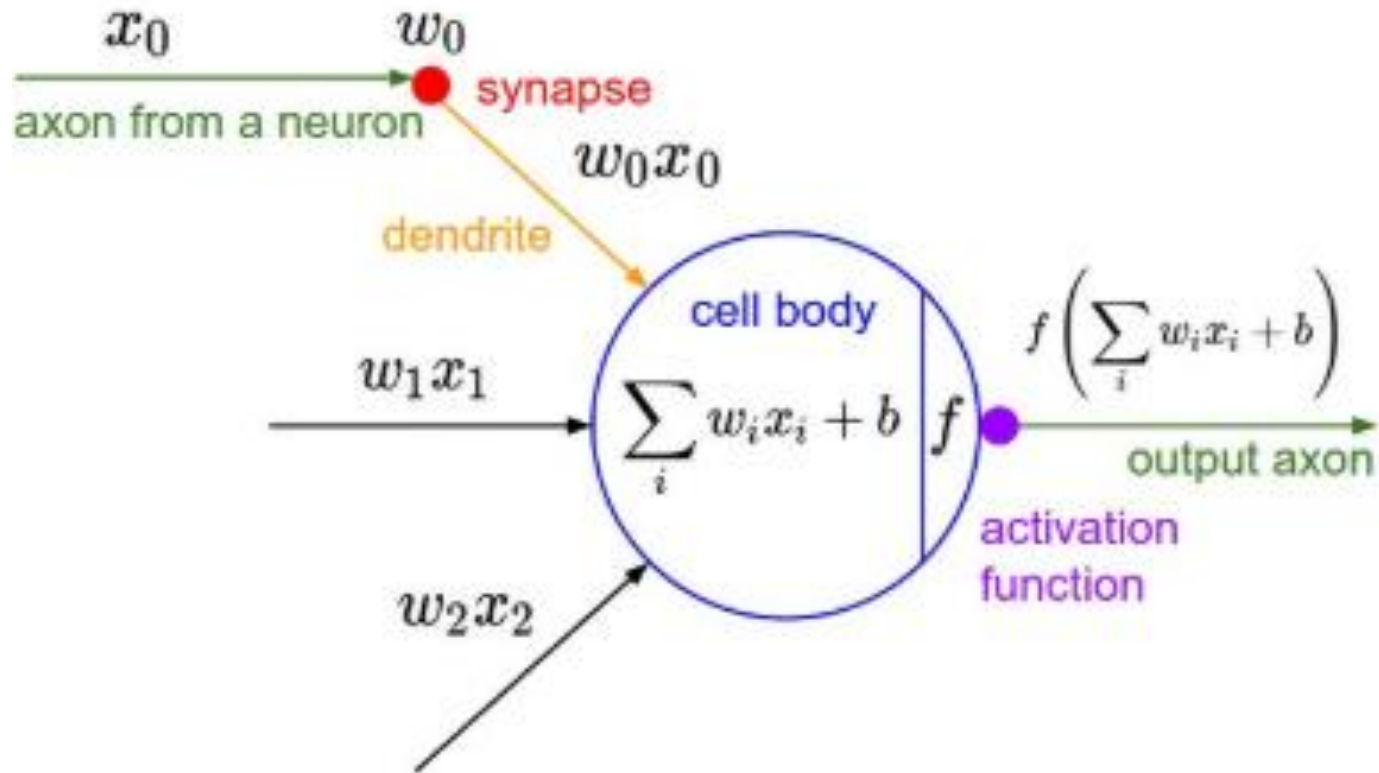


Artificial Neuron (Perceptron)

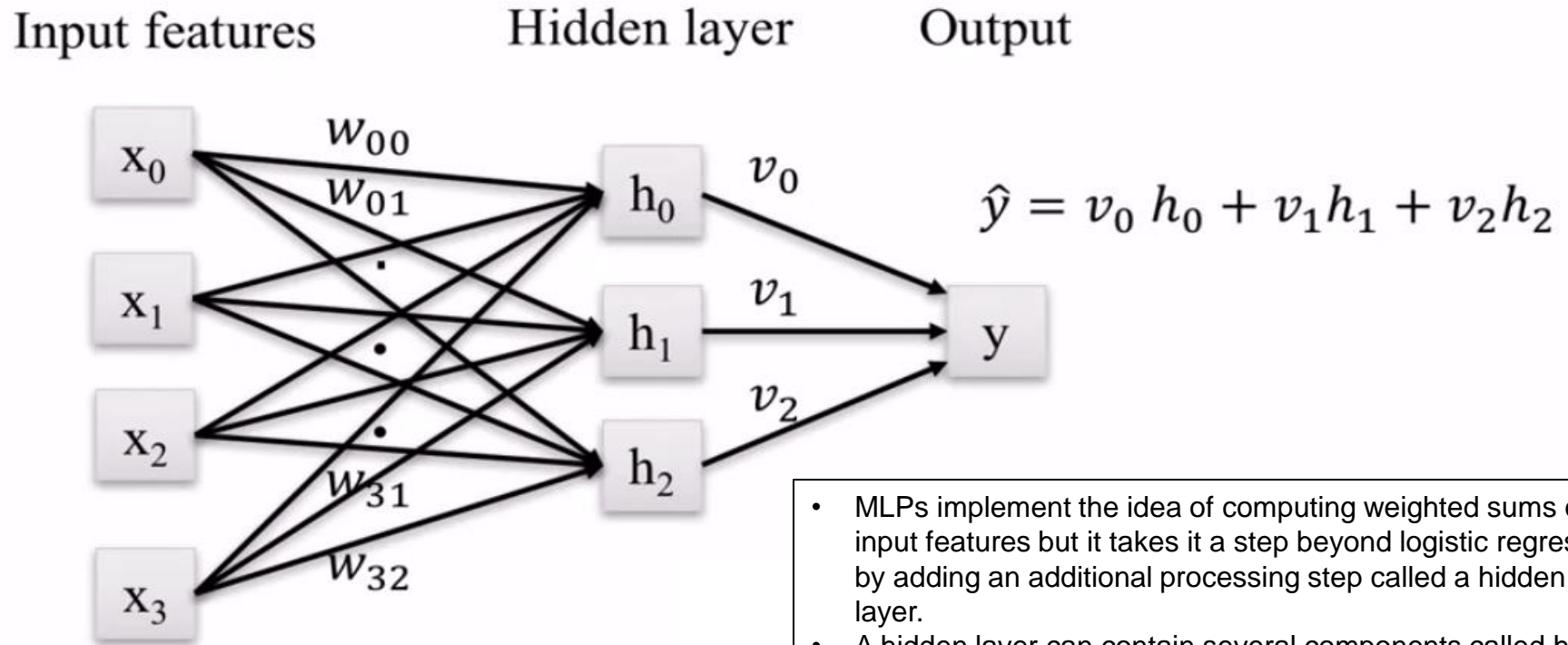


These artificial neurons potentially offers modelling procedures for non-linear data, which can discover more complex relationships and thereby provide more accurate and effective predictive models.

# Internal Structure of Artificial Neurons



# Multi-layer Perceptron (MLP) : A simple Artificial Neural Network



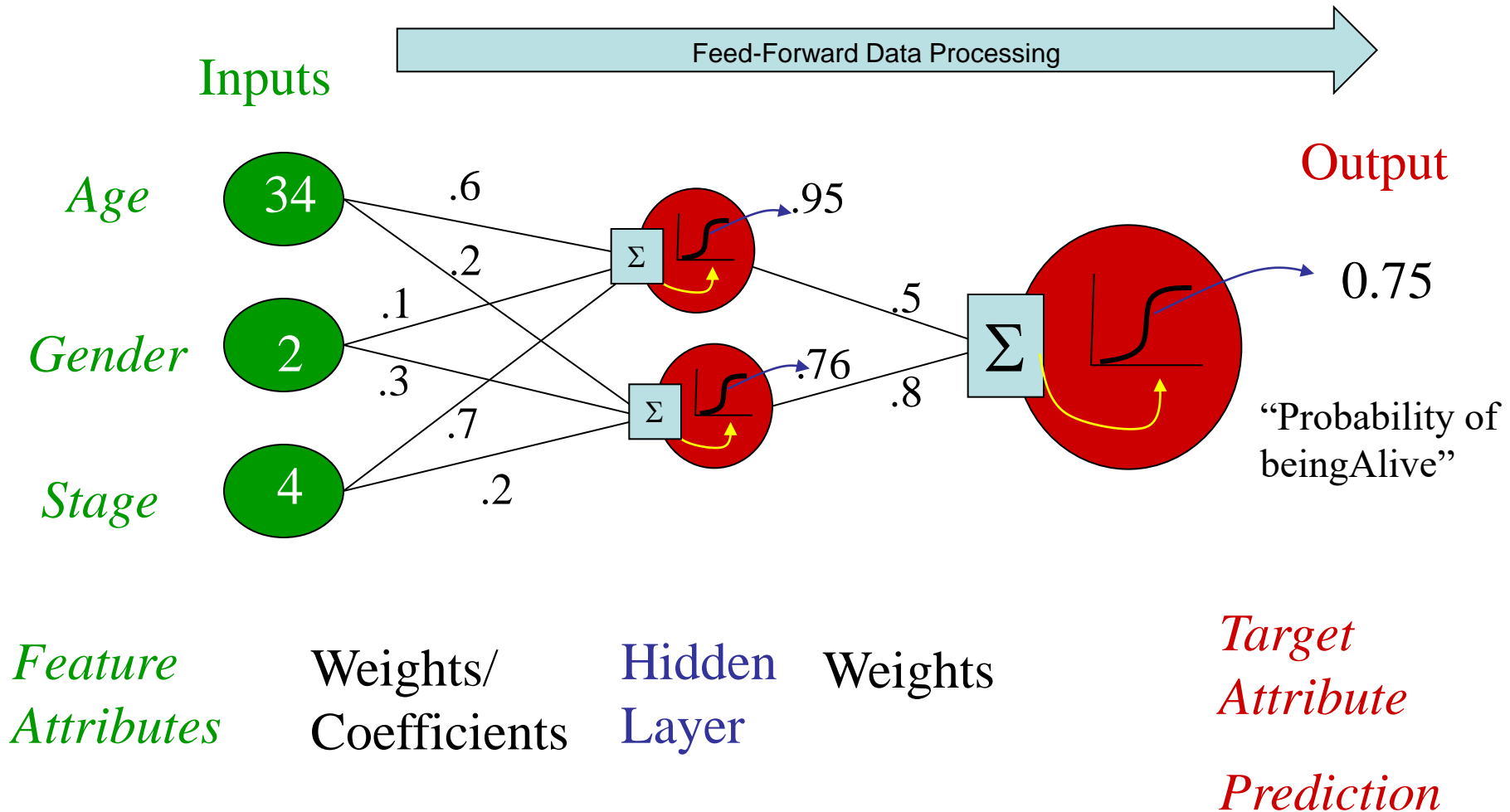
**Universal Approximation Theorem:** a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units

- Hornik, 1991

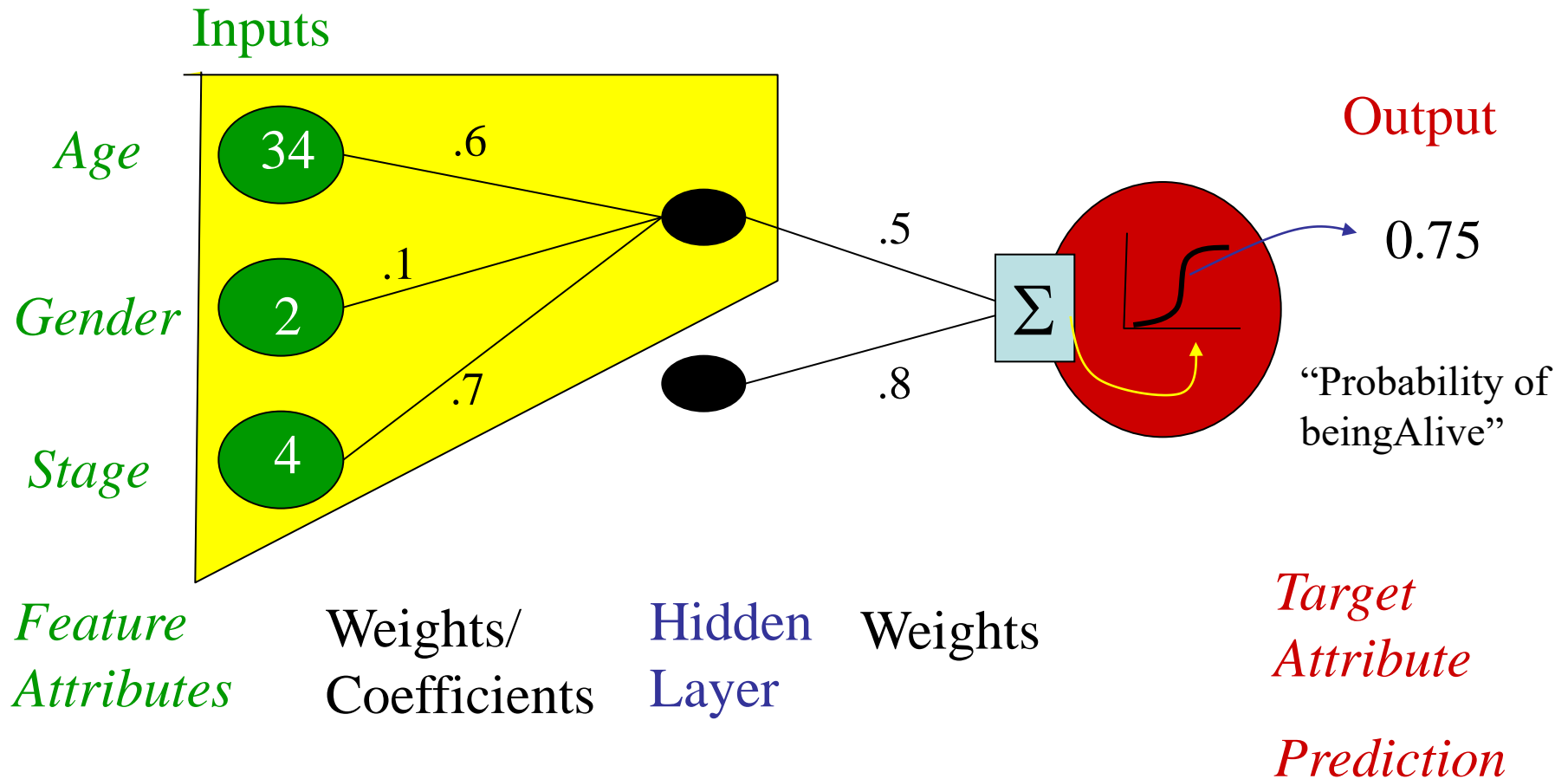
- MLPs implement the idea of computing weighted sums of the input features but it takes it a step beyond logistic regression by adding an additional processing step called a hidden layer.
- A hidden layer can contain several components called hidden units or neurons.
- Each neuron computes the weighted sums of the input features and then applies a nonlinear function called the **activation function**. This generates intermediate output values, i.e.,  $v_0, v_1, v_2$  which is then fed to the nodes in the next layer through a feed-forward propagation process.



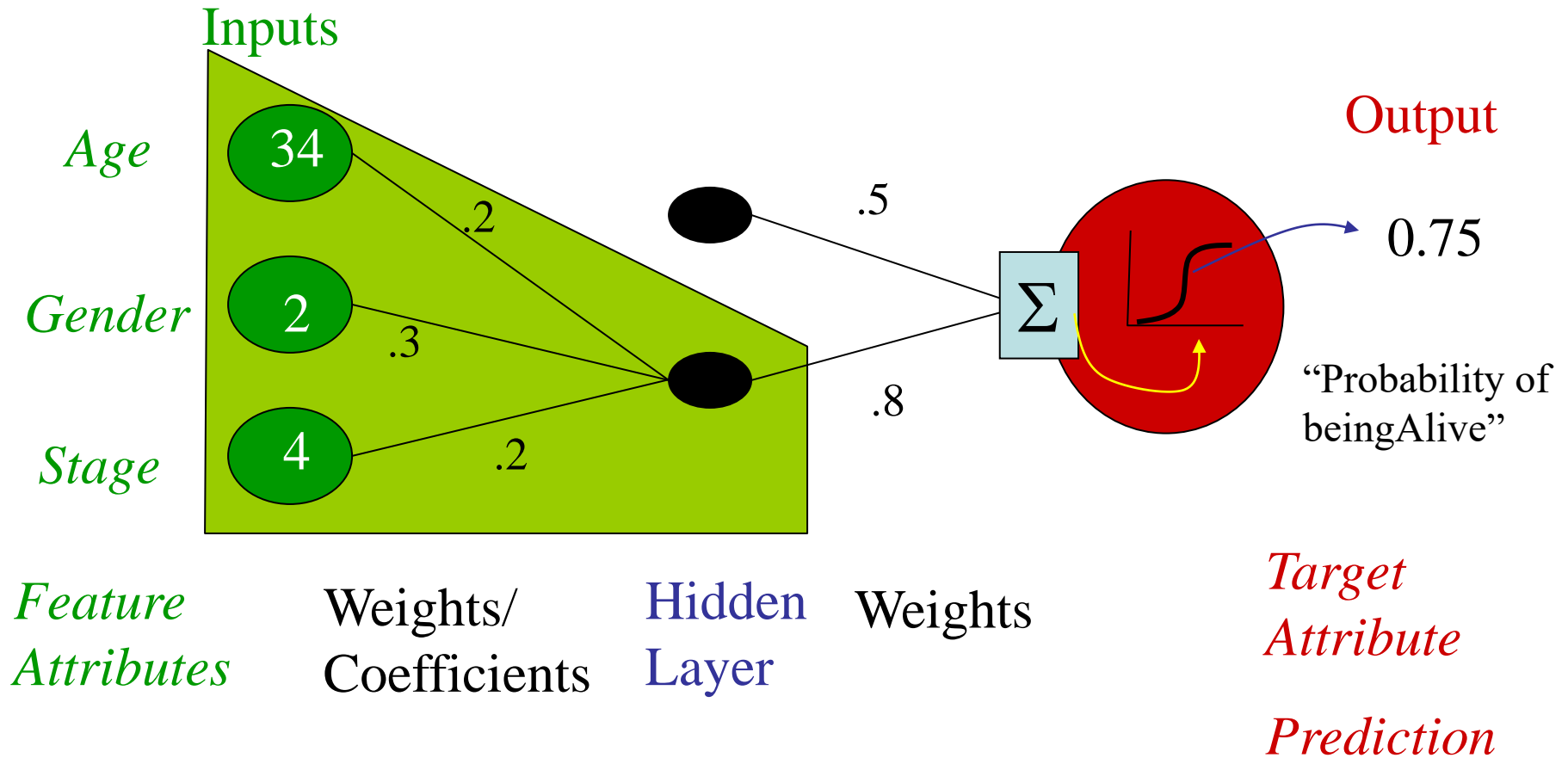
# Data Processing Architecture of a Neural Network



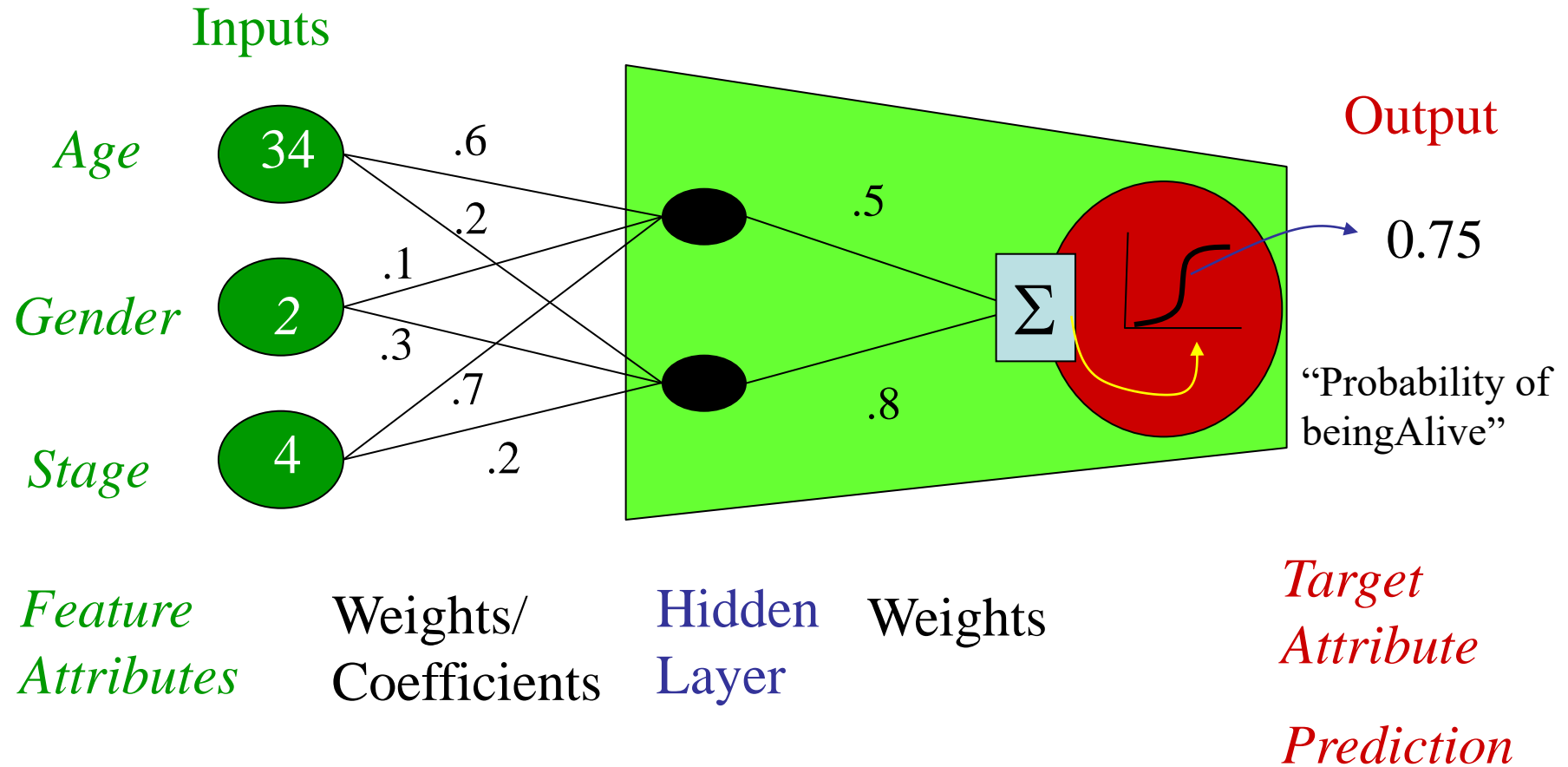
# Neural Network Architecture



# Neural Network Architecture



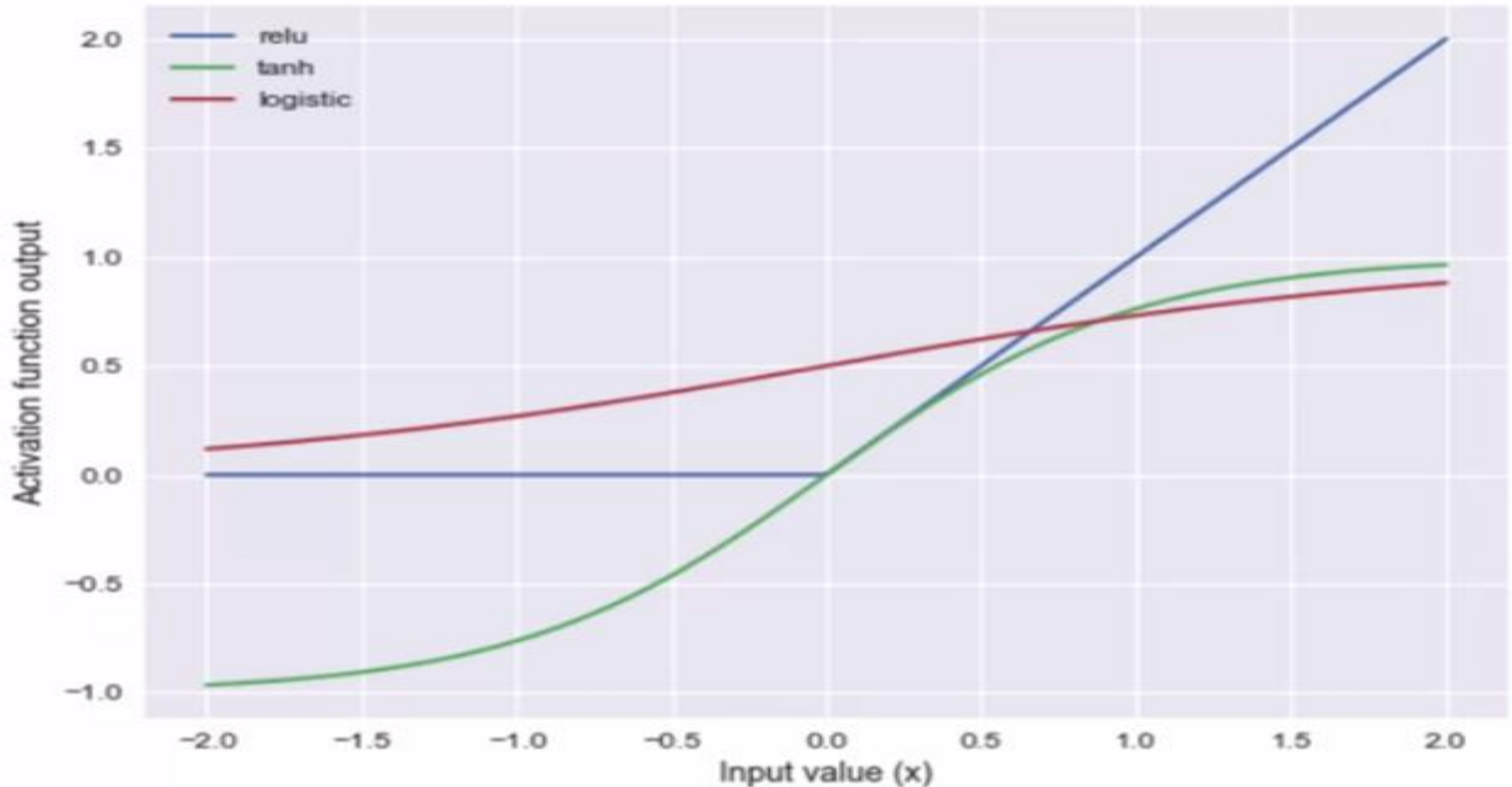
# Neural Network Architecture



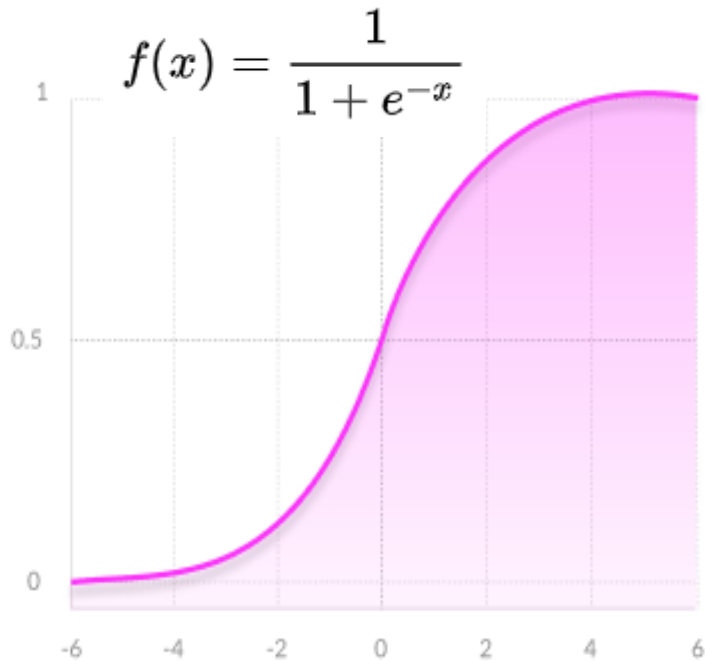
# Activation Functions

- Activation functions are mathematical equations.
- They are attached to each neuron in the network, and determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction.
- Activation functions also help normalize the output of each neuron to a particular range, i.e, between 0 and 1 or between -1 and 1.

# Activation Functions



# Sigmoid/Logistic Activation Function



## Advantages

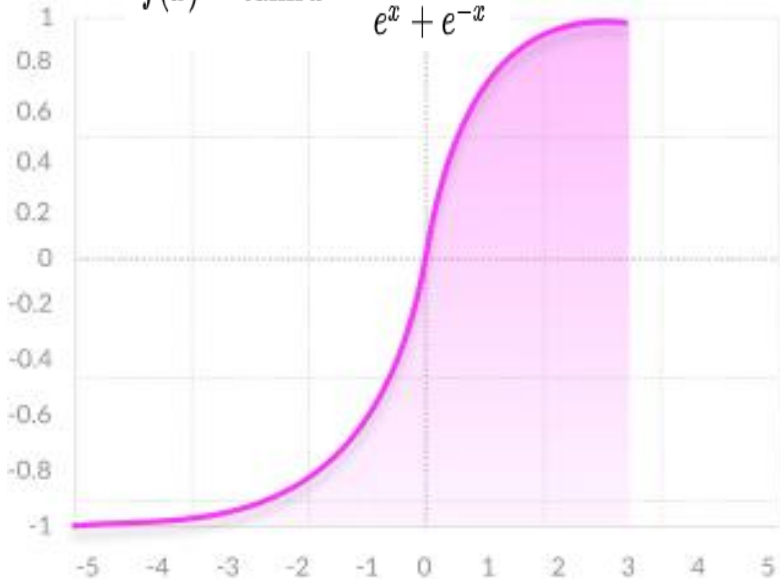
- Output values bound between 0 and 1, normalizing the output of each neuron.
- Always Positive
- Clear predictions—For X above 2 or below -2, tends to bring the Y value (the prediction) to the edge of the curve, very close to 1 or 0. This enables clear predictions.

## Disadvantages

- Vanishing gradient—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- Outputs not zero centered.
- Computationally expensive

# TanH / Hyperbolic Tangent

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



## Advantages

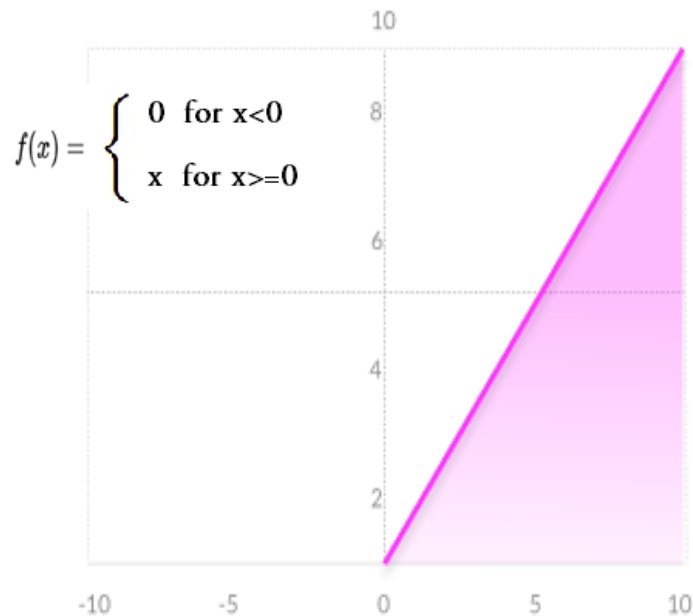
- Output values bound between -1 and 1, normalizing the output of each neuron.
- Zero centered—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.
- Otherwise like the Sigmoid function..

## Disadvantages

- Like the Sigmoid function



# ReLU (Rectified Linear Unit)



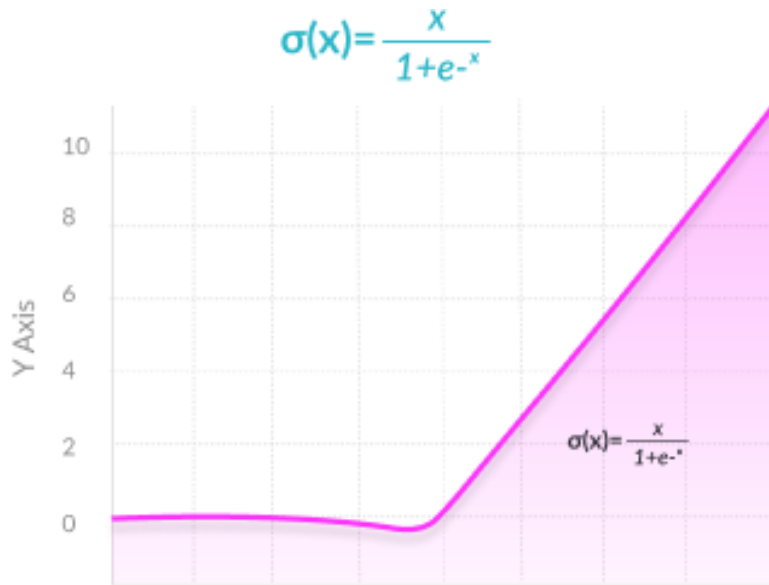
## Advantages

- Computationally efficient—allows the network to converge very quickly
- Non-linear—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation

## Disadvantages

- Not upper bounded
- The Dying ReLU problem—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn.

# Swish Activation Function



- This is a new, self-gated activation function discovered by researchers at Google. According to their paper, it performs better than ReLU with a similar level of computational efficiency. In experiments on ImageNet with identical models running ReLU and Swish, the new function achieved top -1 classification accuracy 0.6-0.9% higher.

# Softmax Activation Function

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

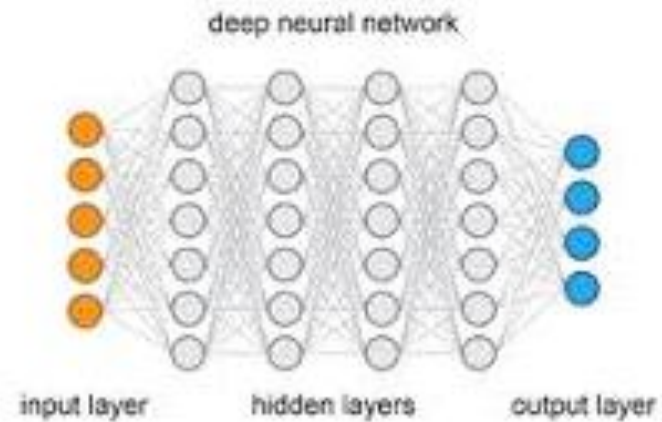
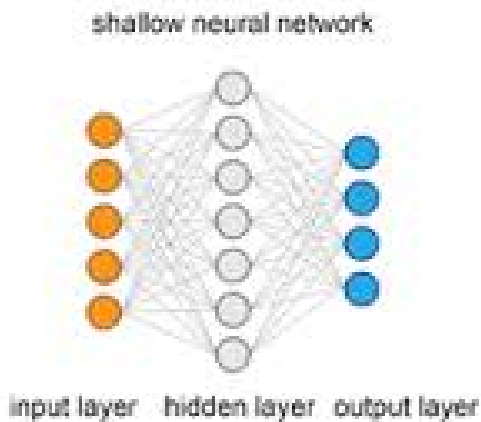
## Advantages

- A more generalized logistic activation function
- Able to handle multiple classes only one class in other activation functions—normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class.
- Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

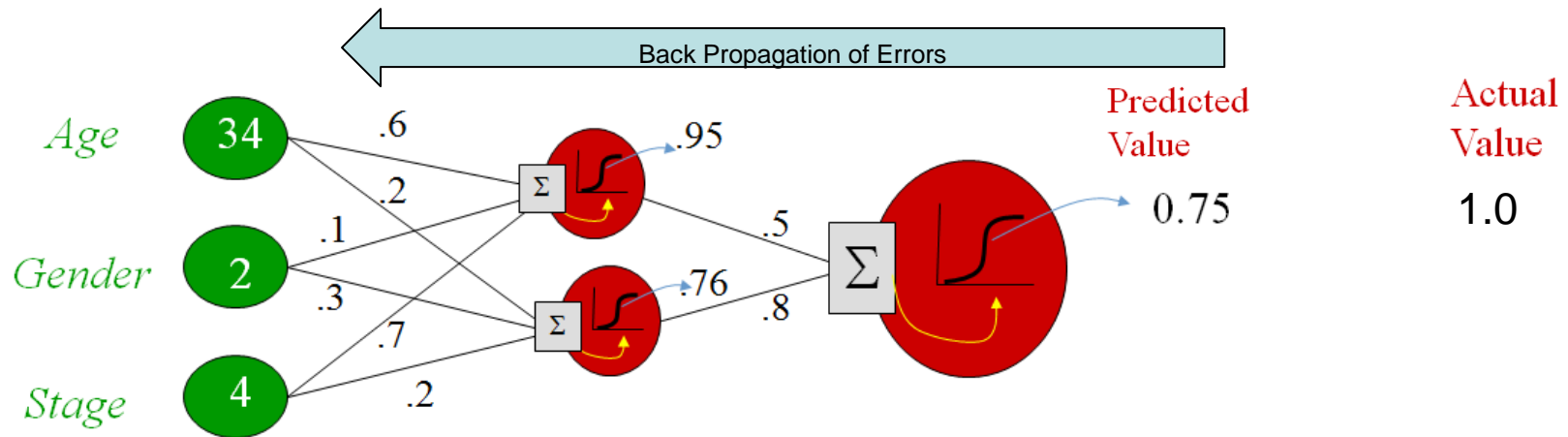
# Vanishing/Exploding Gradients

The **vanishing gradient** problem is encountered when training artificial neural networks with gradient-based learning methods and backpropagation. Each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight. The problem is that in some cases, **the gradient will be vanishingly small**, effectively preventing the weight from changing its value.

The **exploding gradient** problem is encountered when large error gradients accumulate and result in **very large updates to neural network model weights** during training. This has the effect of your model being unstable and unable to learn from your training data.

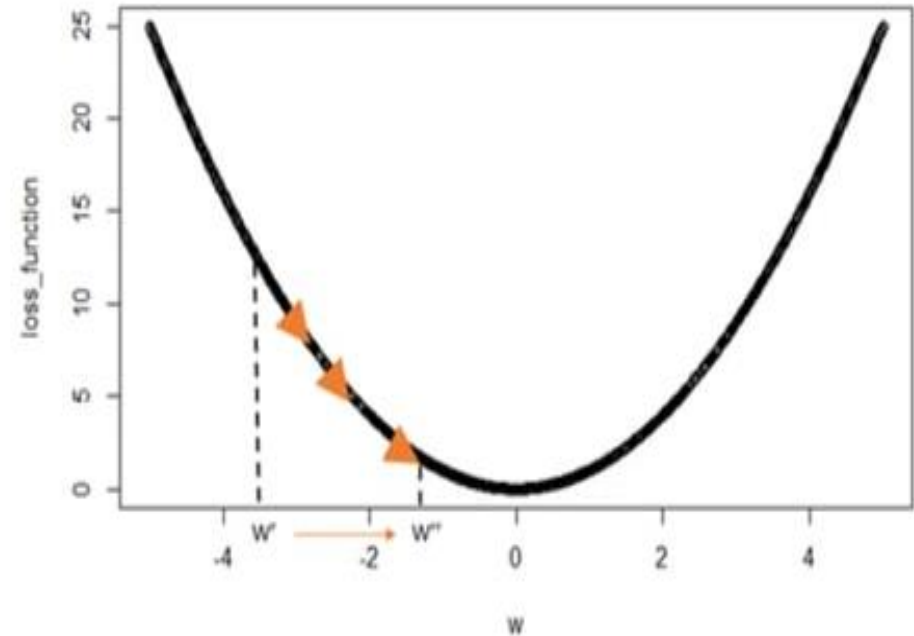
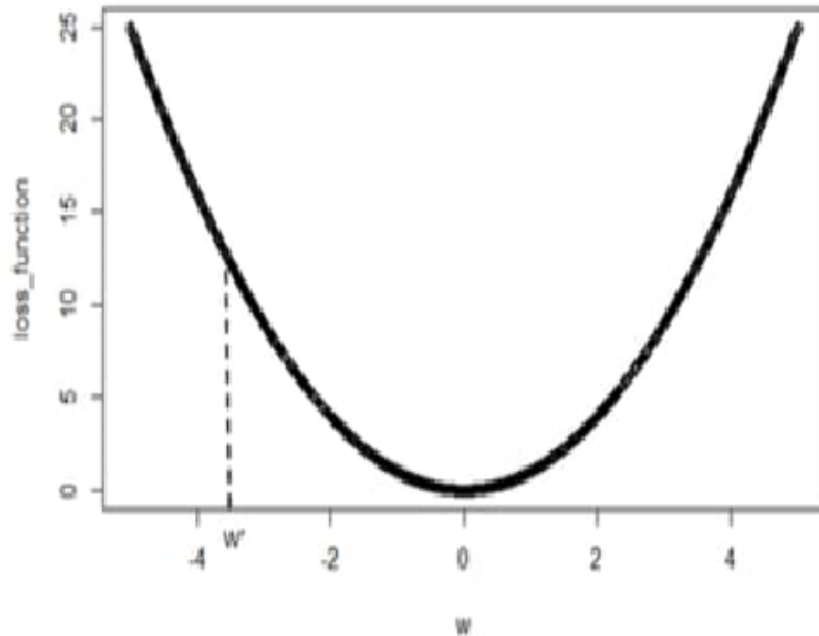


# Learning Architecture of a Neural Network

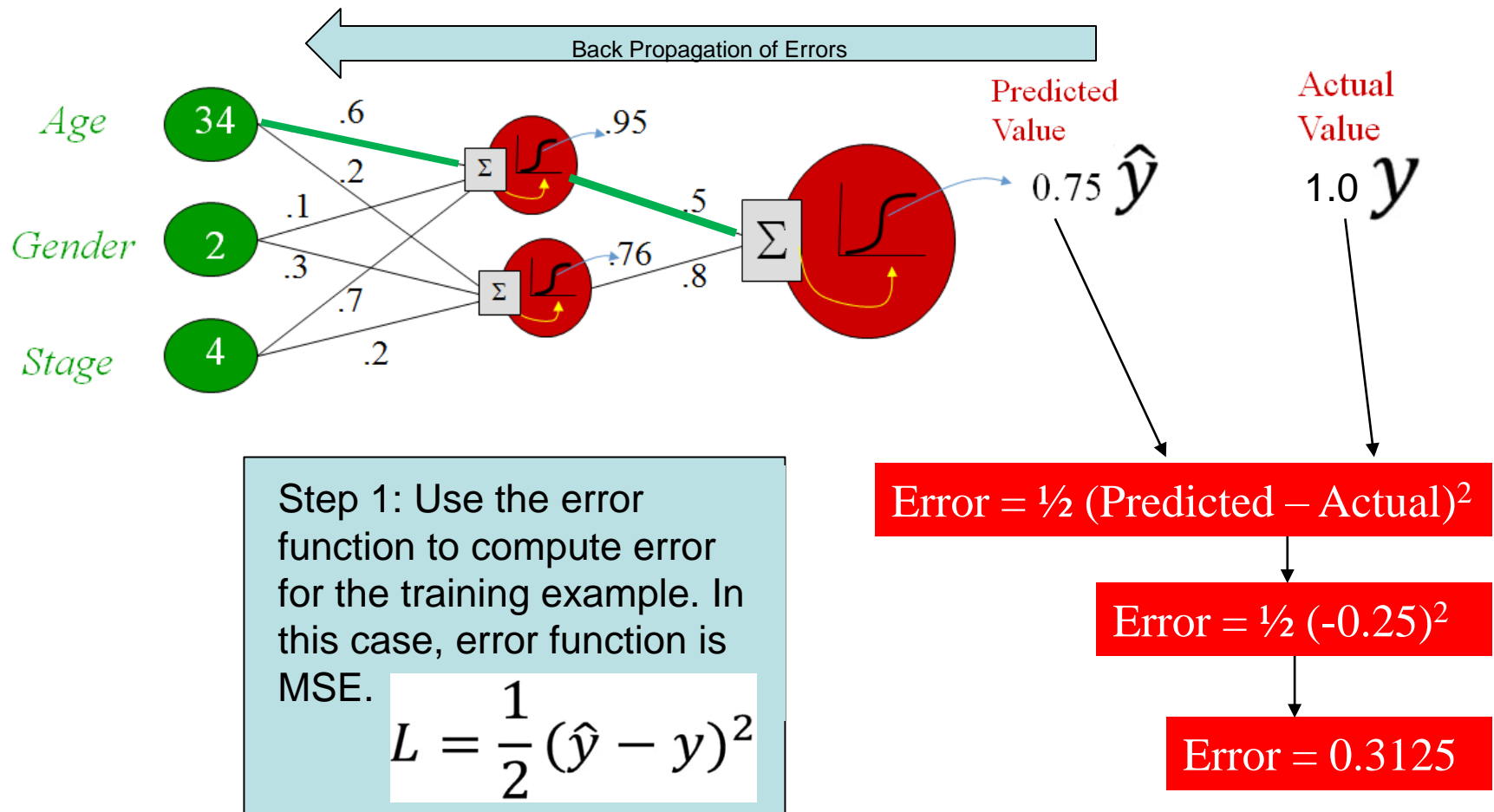


- Back-propagation is the process of optimizing prediction performance (i.e., minimize errors) by adjusting weight parameters using the error gradients.
- Gradient descent calculates how much adjustments to the weights are needed so as to achieve the most efficient decrease to the error function.
- The error gradients gives the direction in which the error function decreases, i.e., the negative gradient.

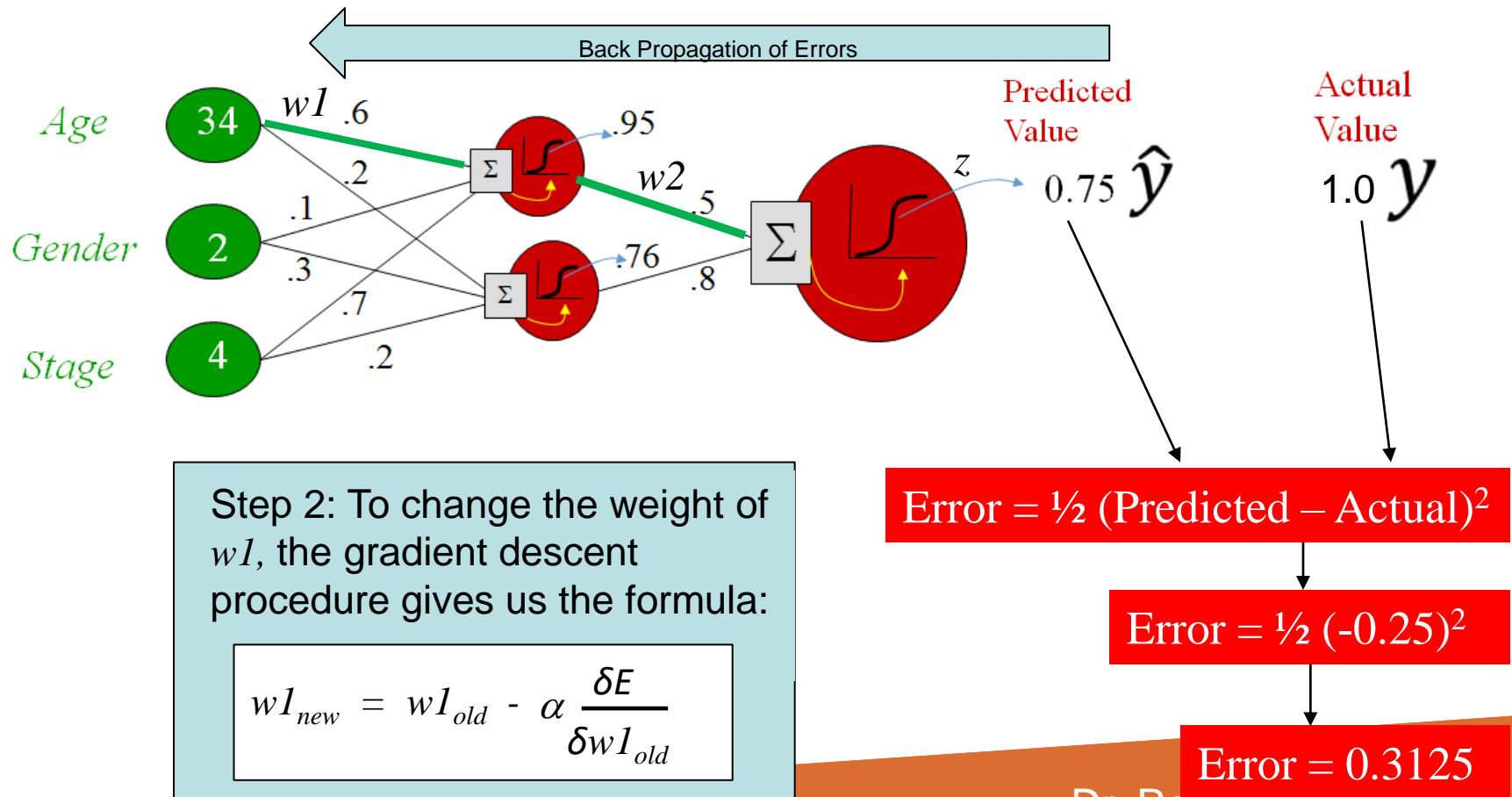
# Gradient descent and error gradient



# Learning Architecture of a Neural Network - Intuition

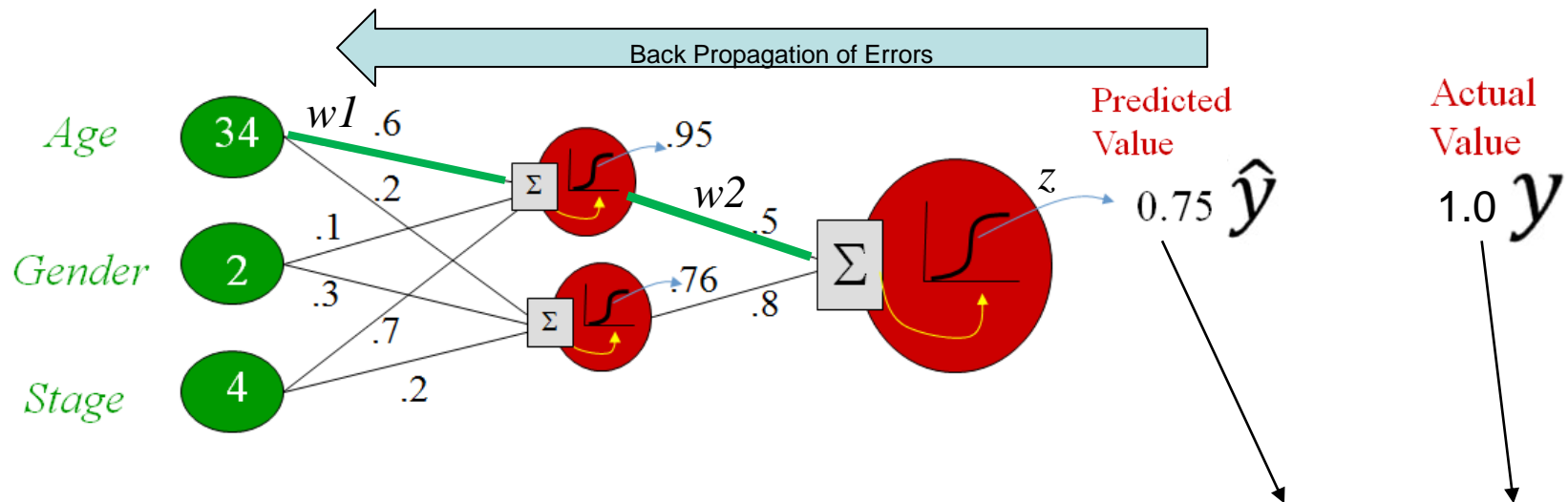


# Learning Architecture of a Neural Network - Intuition





# Learning Architecture of a Neural Network - Intuition



Step 3: Compute for the derivative of  $E$  wrt  $w1$ , which, following the chain rule will be:

$$\frac{\delta E}{\delta w1} = \frac{\delta E}{\delta \hat{y}} * \frac{\delta \hat{y}}{\delta z} * \frac{\delta z}{\delta w2} * \frac{\delta w2}{\delta w1}$$

$$\text{Error} = \frac{1}{2} (\text{Predicted} - \text{Actual})^2$$

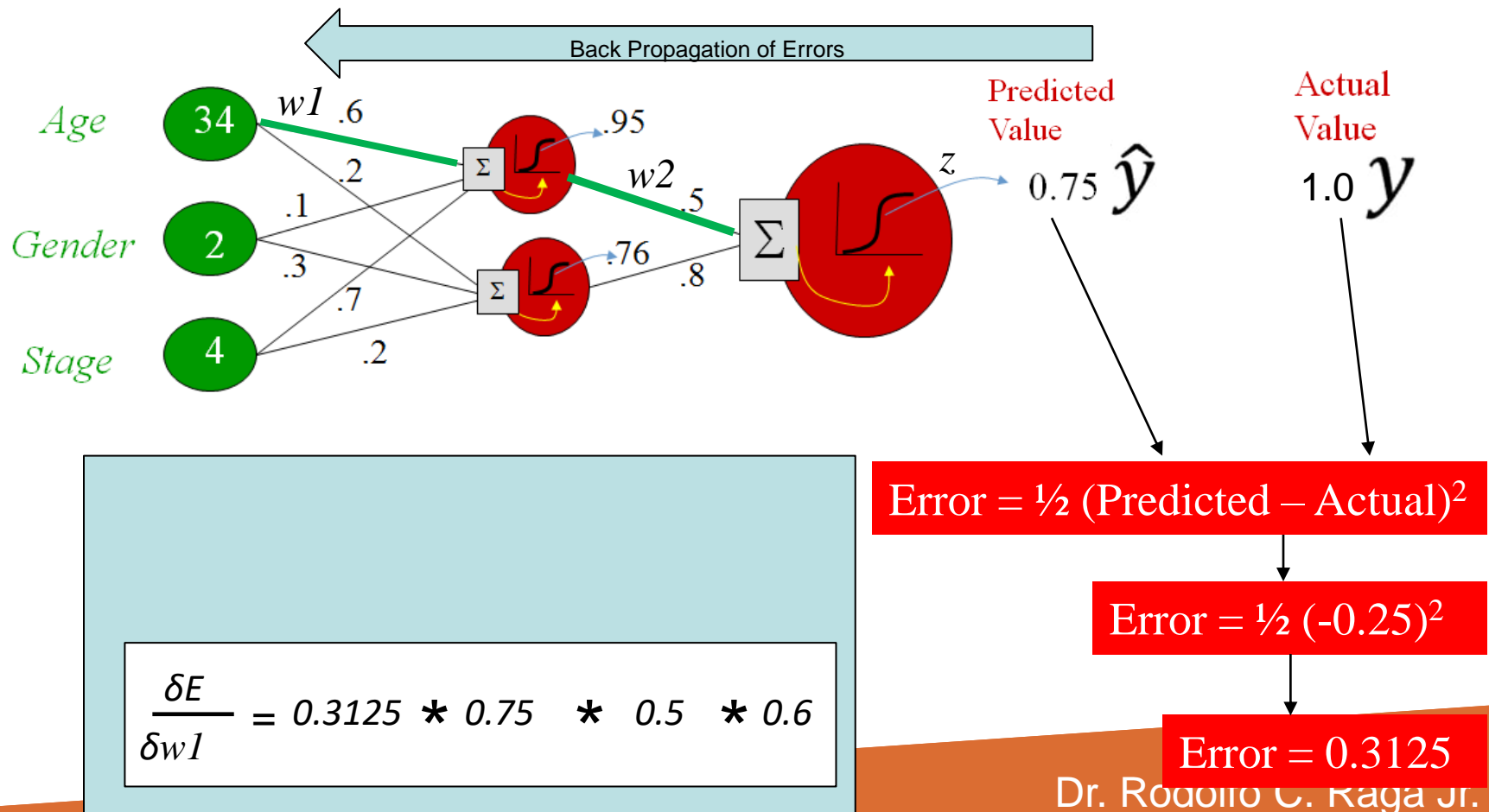
$$\text{Error} = \frac{1}{2} (-0.25)^2$$

$$\text{Error} = 0.3125$$

Dr. Rodolfo C. Raga Jr.

NEUST Workshop – Nov 8-10, 2021

# Learning Architecture of a Neural Network - Intuition



# MLP Demo

- ◆ Demo of generating an MLP prediction model using sklearn.