UNIVERSITY of GUELPH

Introduction to Neural Networks

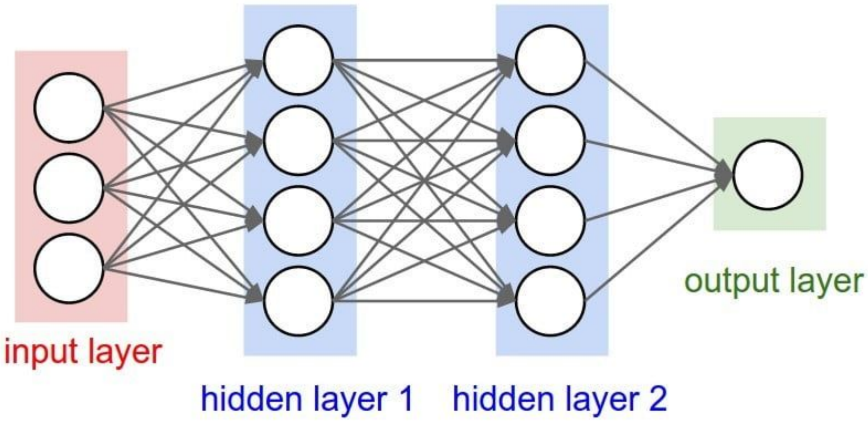
Terrance DeVries

Contents

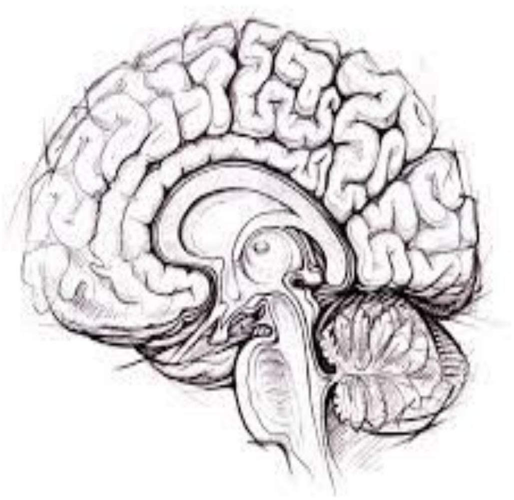
What is an Artificial Neural Network?

- Predictive model that can learn to map given inputs to desired outputs
- Mathematical function designed to mimic the brain

Artificial Neural Network



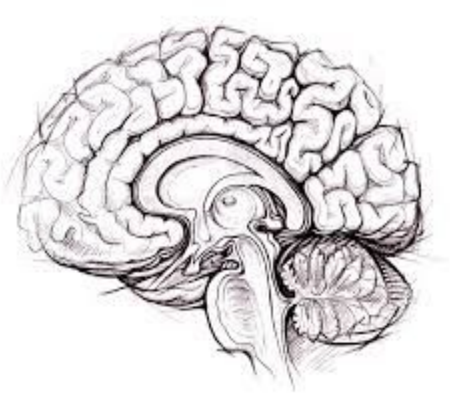
Biological Neural Network

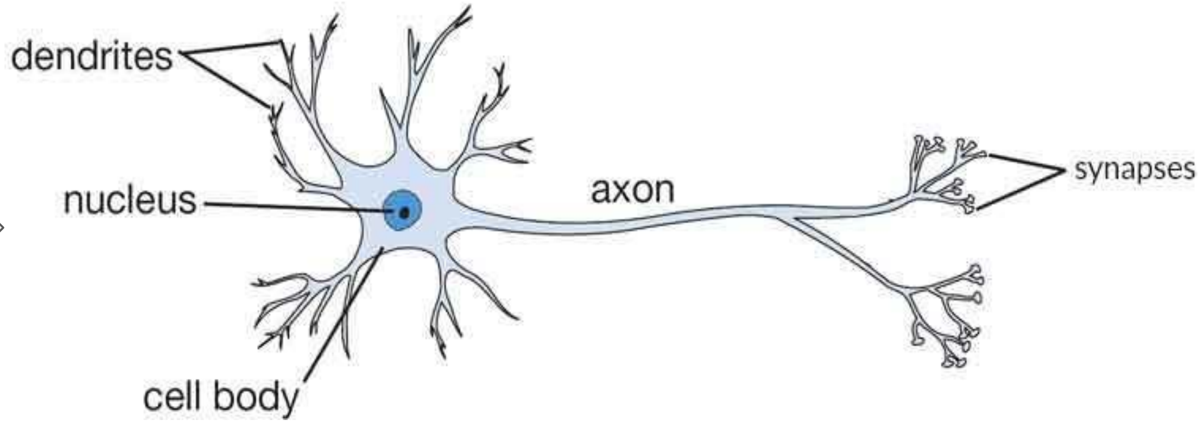


The Biological Neuron

The brain contains billions of interconnected neurons.

1. Dendrites take in inputs
2. Cell does some electrochemical processing
3. If resulting voltage is greater than some threshold, the neuron “fires”
4. Signal is sent down axon to other neurons

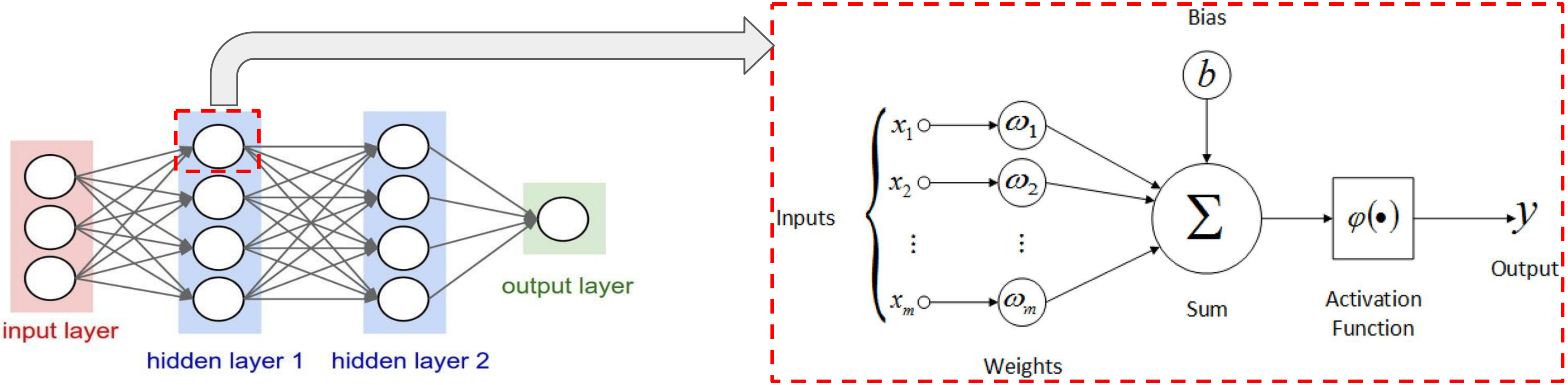




The Artificial Neuron

Artificial neural networks are composed of many artificial neurons.

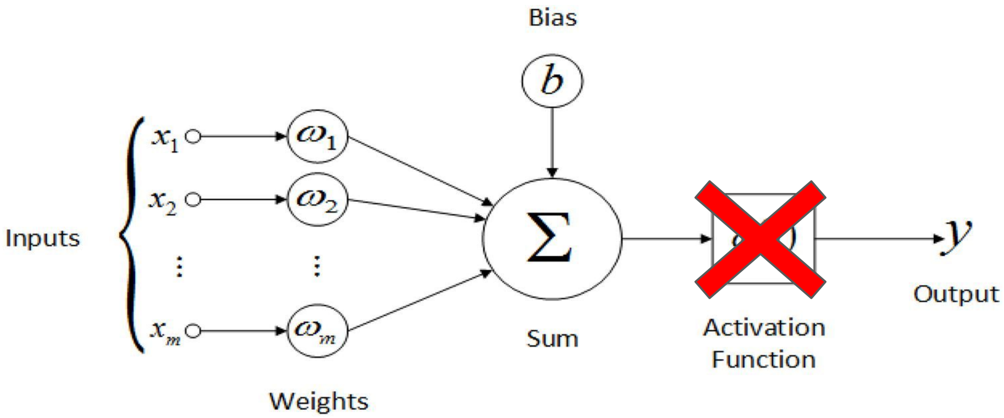
1. Neuron receives inputs
2. Each input is multiplied by some weight and then summed together
3. Pass response through an “activation function”
4. Output signal is sent to other neurons

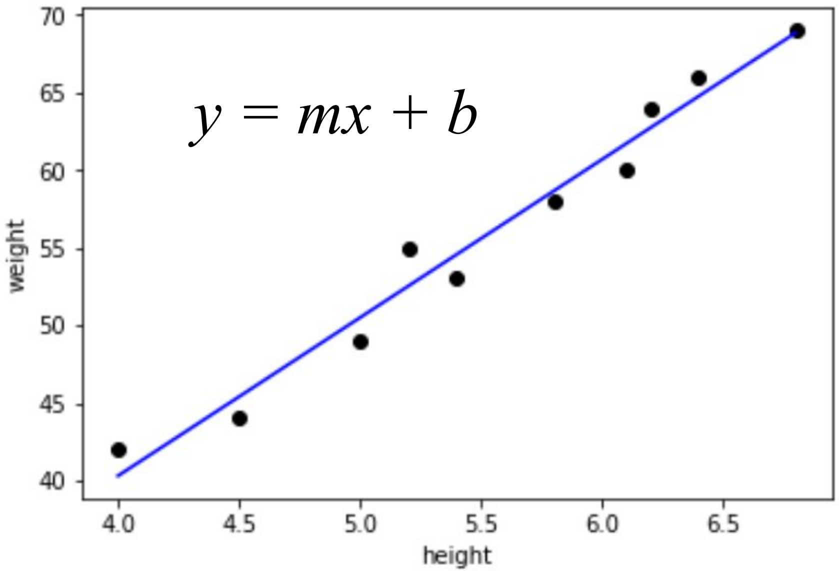


The Artificial Neuron

An artificial neuron without an activation function is simply linear regression

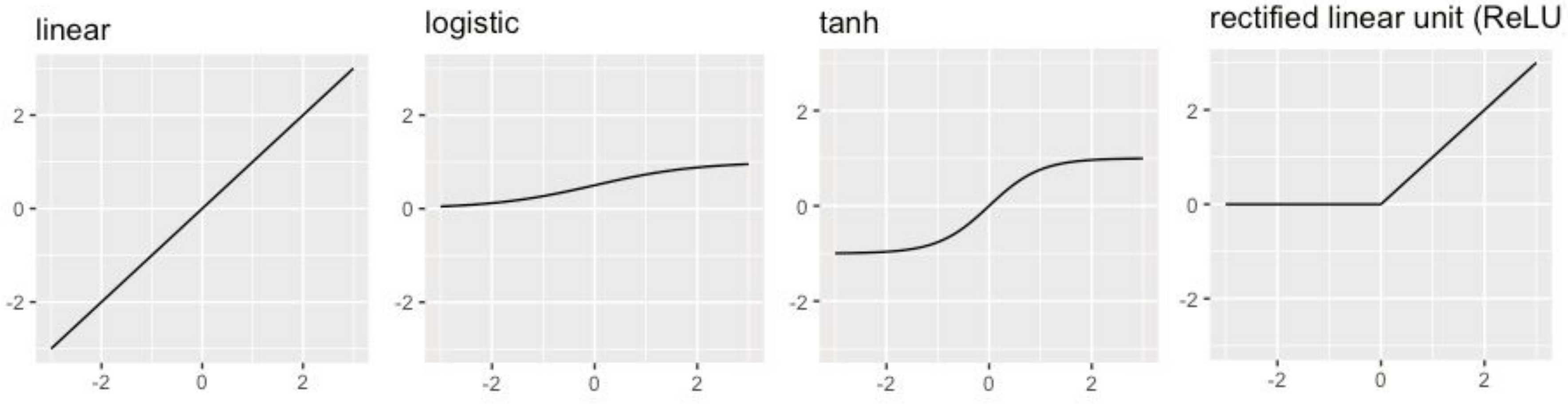
- x = input value
- y = predicted value
- m = slope of the line
- b = bias





Activation Function

- Simulates the firing of a biological neuron
- Allows the neural network to model non-linear problems (only if the activation function is also non-linear)



(Equivalent to having no activation function)

Interactive Demo

https://playground.tensorflow.org

Universal Approximation Theorem

A neural network with at least one hidden layer can approximate any continuous function.

This is very powerful: for any set of input-output pairs, there exists a neural network that can almost perfectly model them

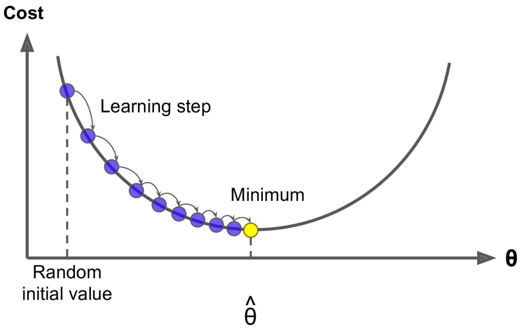
Some limitations:

- Number of neurons may be impractically large
- Generalization to new samples is not guaranteed
- It may be difficult to find the correct weights

How Do We Find the Correct Weights?

Stochastic Gradient Descent (SGD): Iterative method for optimizing differentiable functions.

1. Randomly initialize weights ω and select learning rate η

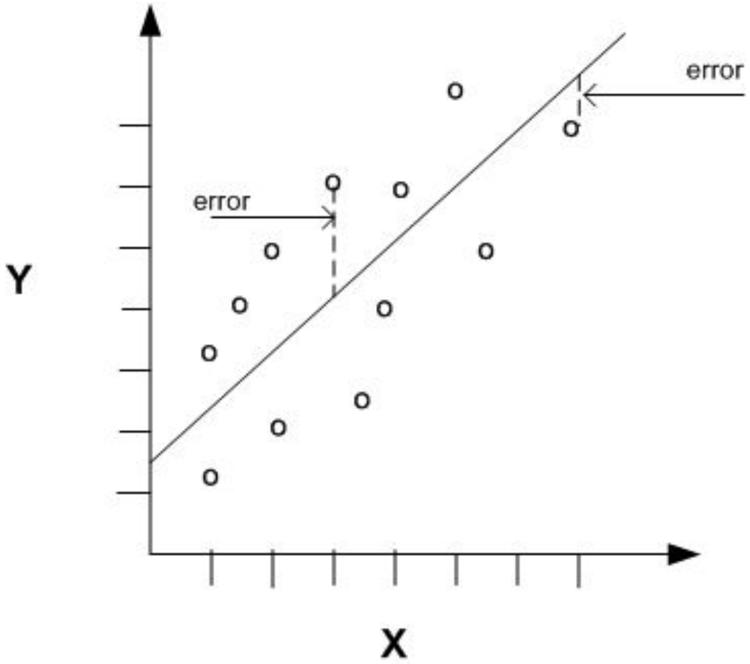


2. Repeat until convergence: $\boxed{\omega := \omega - \eta \frac{\partial E}{\partial \omega}}$

To calculate E we need a **loss function**, and to calculate $\frac{\partial E}{\partial \omega}$ we use **error backpropogation**.

Loss Function

- Loss function measures how far away the prediction is from the desired output (i.e. error)
- Use gradient descent to minimize the loss



Regression loss function:
- Mean squared error (MSE): $E = (\hat{y} - y)^2$

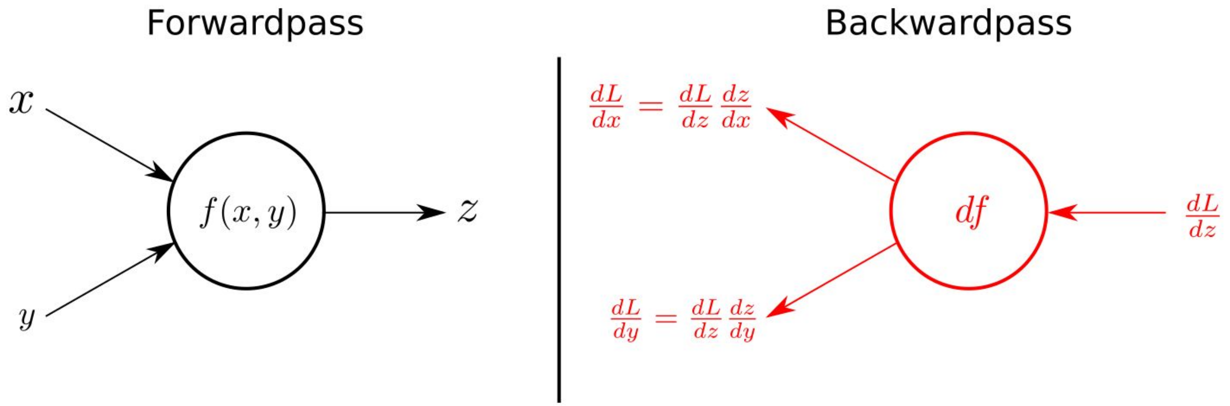
Classification loss function:
- Cross entropy:

$$E = y\log(\hat{y}) + (1 - y)\log(1 - \hat{y})$$

Error Backpropogation

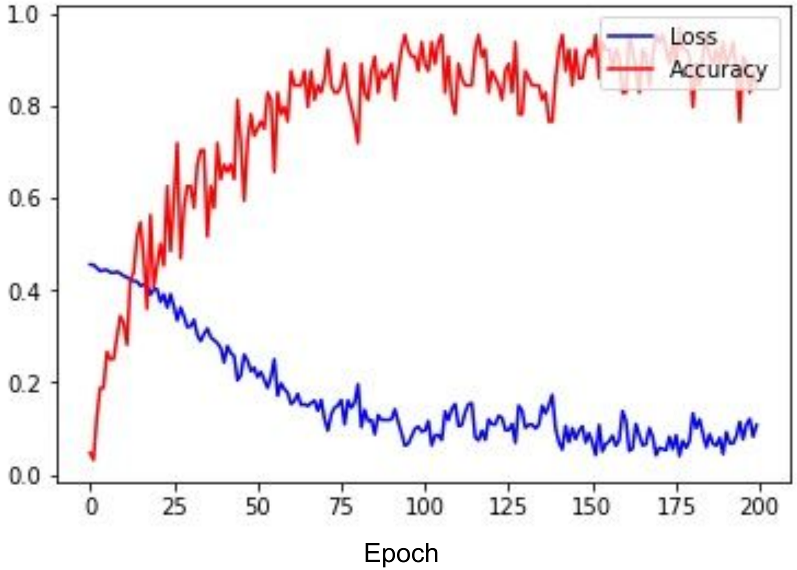
In order to calculate the error attributed to each weight we use the backpropogation algorithm:

1. Propagate forward through the network to generate an output
2. Calculate the loss (i.e. error)
3. Use chain rule to calculate the error associated with each neuron



Training Loop

1. Load batch of training inputs
2. Perform forward pass
3. Calculate loss
4. Backpropogate errors
5. Update weights
6. Repeat until convergence



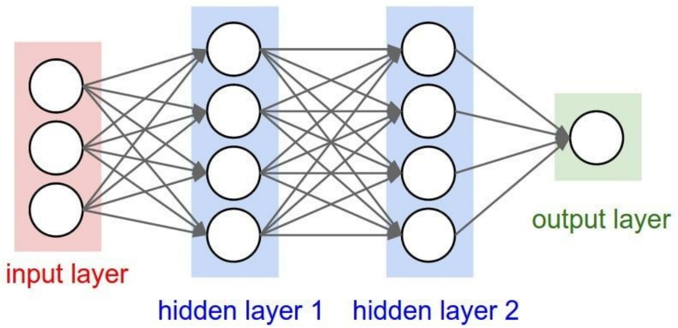
One pass through the training loop is called an **iteration.**

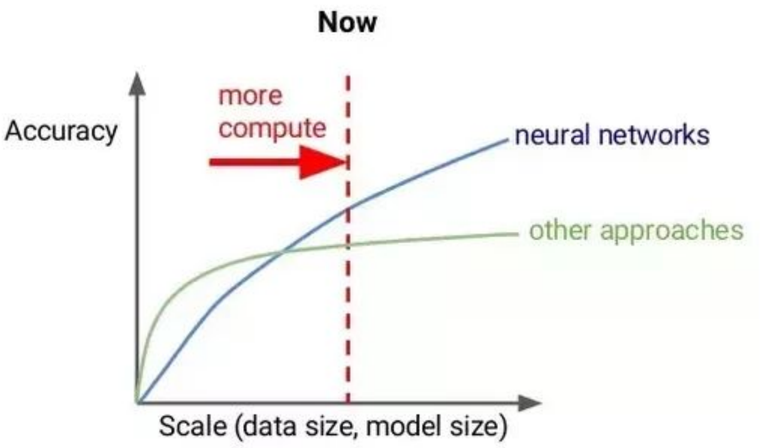
One pass through the dataset is called an **epoch.**

Multiple epochs are usually required before the model converges.

Why Neural Networks?

- Automatic feature extraction
  - No need to hand-craft features
- Extremely versatile
  - Can be adapted to a wide variety of non-standard problems
- Performance scales with the amount of data





Deep Learning Libraries

- Provides optimized implementations of common neural network building blocks
- Automatic differentiation - no need to manually calculate derivatives!
- Some libraries provide tools for deploying trained models

TensorFlow™ PYTORCH Caffe2

Keras mxnet Microsoft CNTK

Jupyter Notebook

https://jupyter.co60.ca