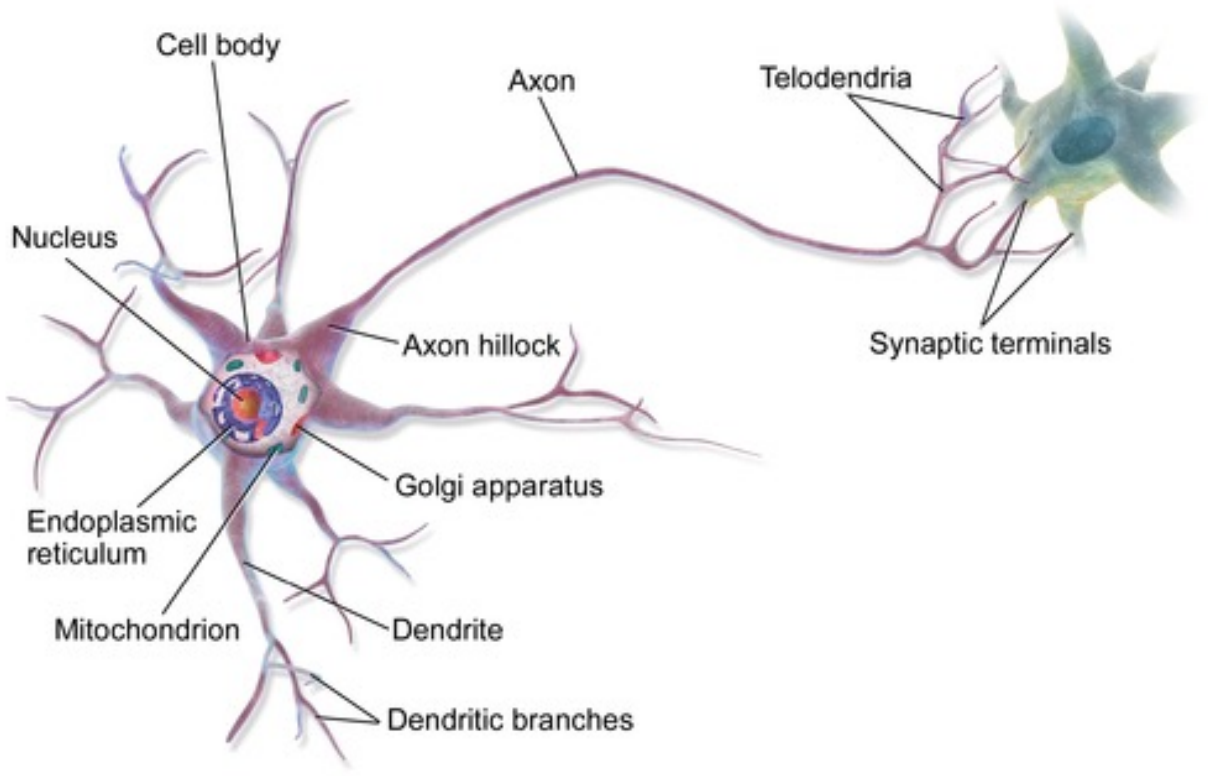
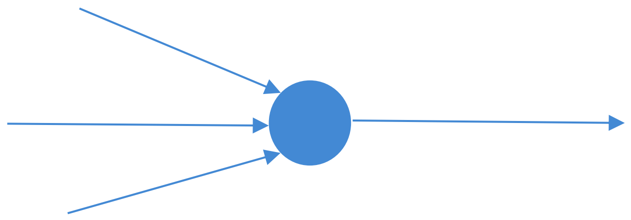
Introduction to Neural Networks

DataScience@SMU

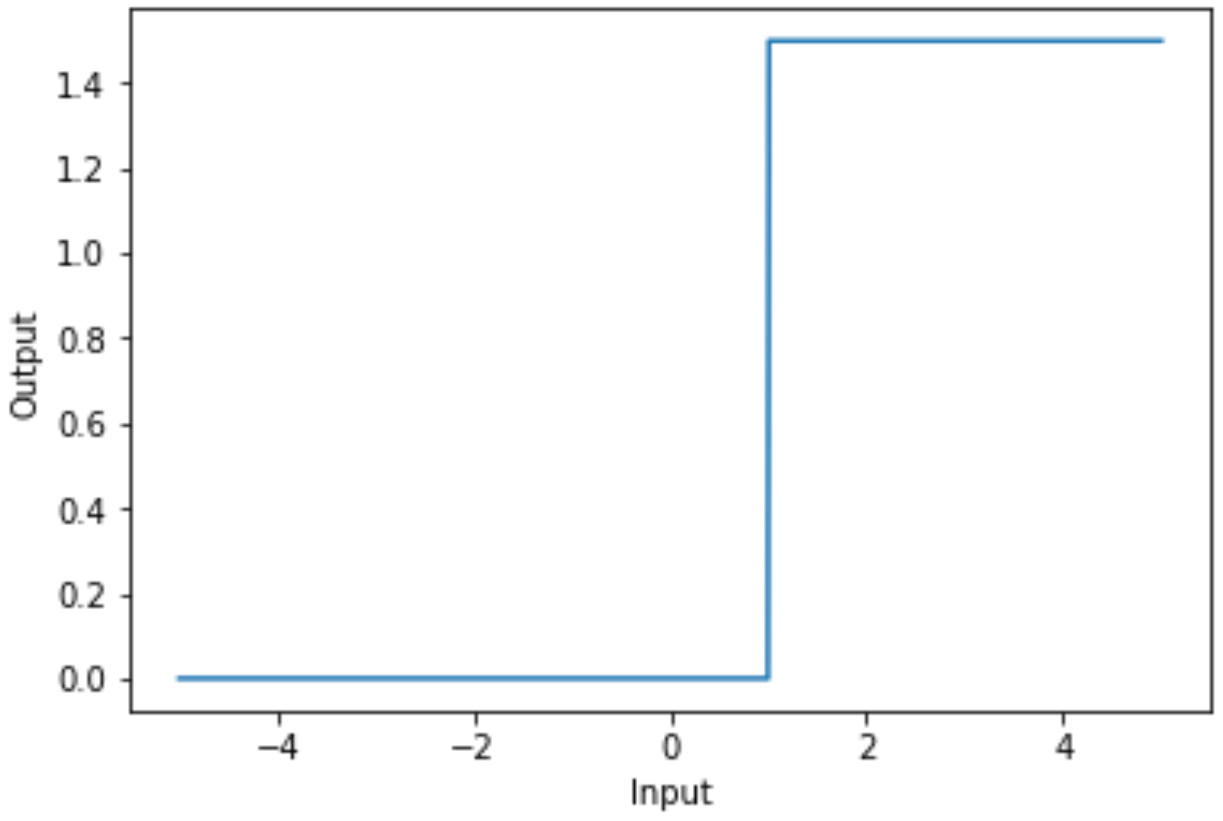
Neural Network Is Based on Your Brain





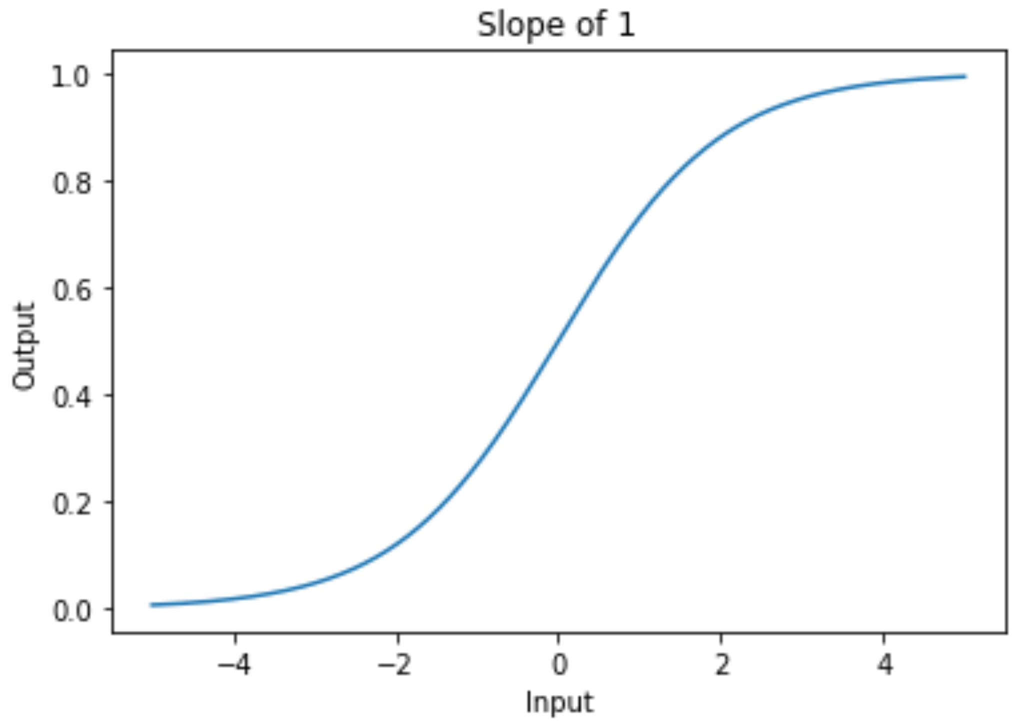
How Does a Neuron Work?

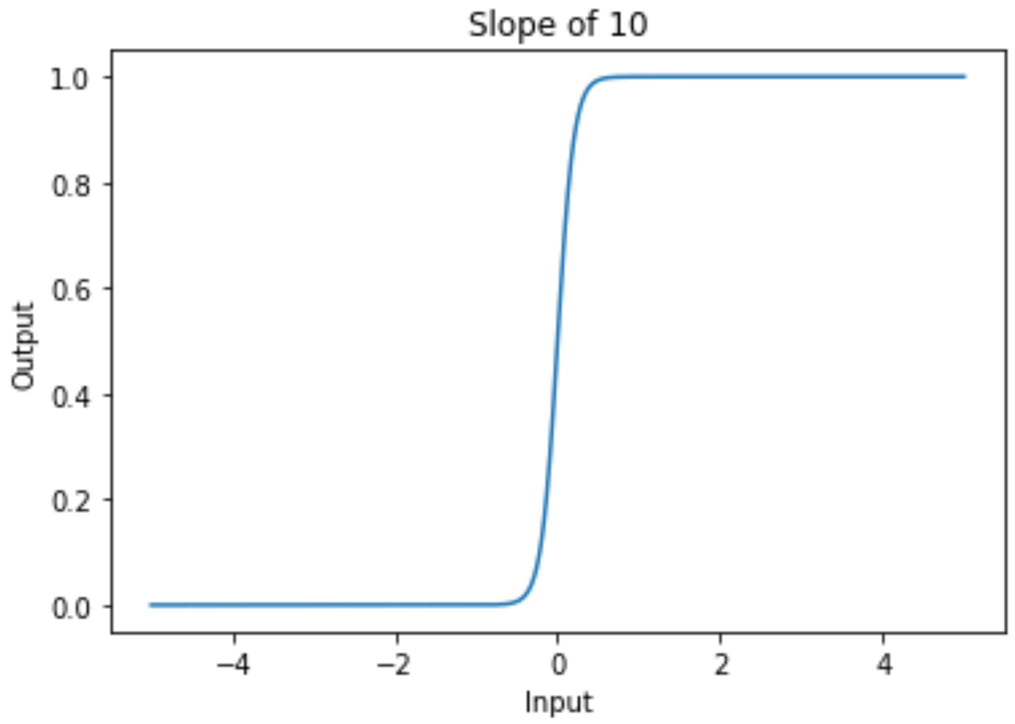
A neuron receives signals. Once the combine input of the signals reaches a threshold, the neuron will fire off a signal.



Model of the Step Function

The step function is not a smooth function. But the sigmoid is a good approximation of a step function!



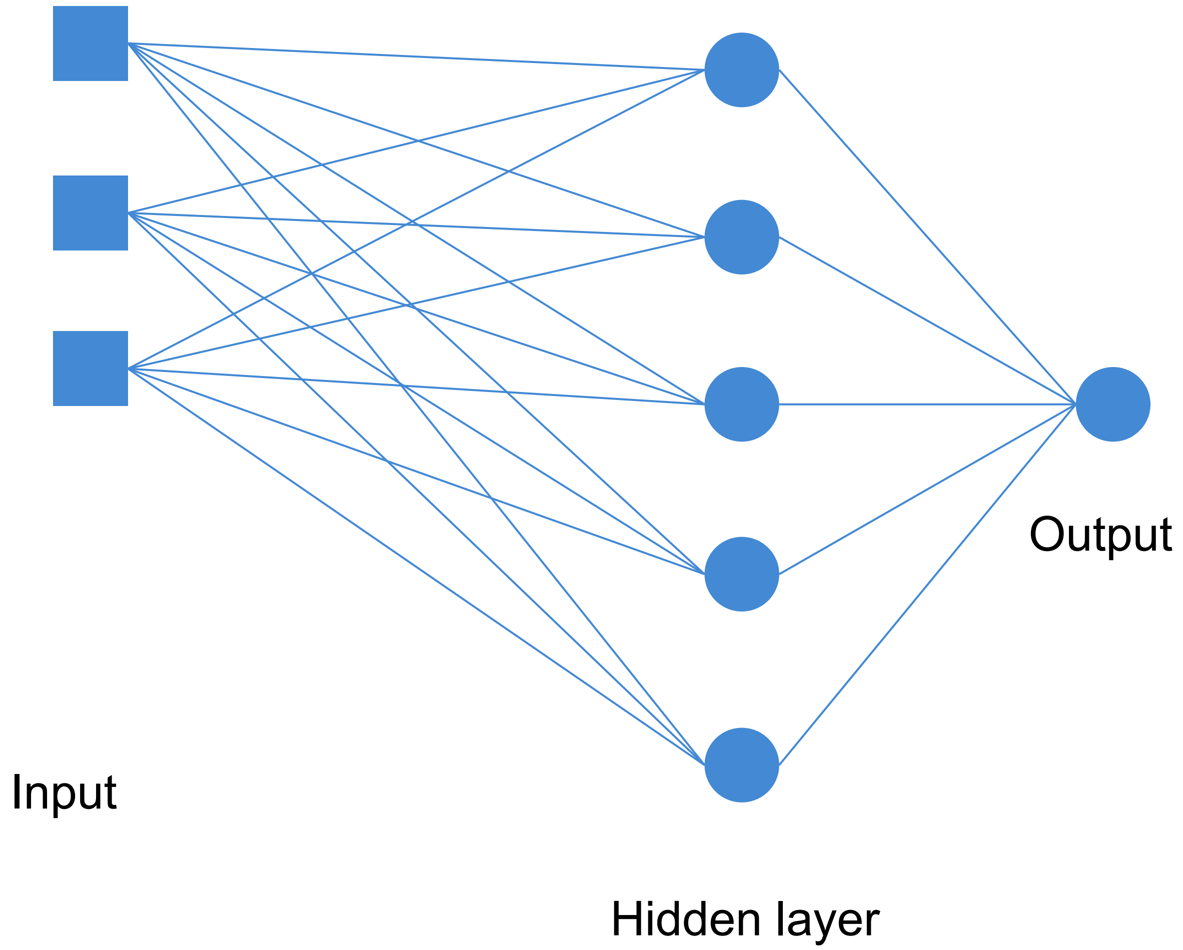


Neural Network: Ensemble of Regression

- A neural network is just a regression ensemble. The sigmoid is what we call our “activation” function.
- Activation function can be any nonlinear function.
- Each neuron is A(mx + b) where A(z) is the activation function.
- We will review activation functions in a future lesson.

Network Diagrams

This is a simple network with 1 hidden layer that contains 5 neurons. There are 3 input variables (typically not shown, but shown here for clarity). All the inputs are sent to every neuron in the hidden layer. The hidden layer outputs of each neuron are used as inputs to the next layer, in this case the output.



DataScience@SMU

Math for Neural Networks

Each Neuron Is a Regression

- Each neuron calculates:

$$z = \sigma(w_i x_i + b)$$

- Where σ is the activation function:

$$\sigma(z_1), z_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1$$

$$\sigma(z_2), z_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2$$

$$\sigma(z_3), z_3 = w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3$$

$$\sigma(z_4), z_4 = w_{41}x_1 + w_{42}x_2 + w_{43}x_3 + b_4$$

$$\sigma(z_5), z_5 = w_{51}x_1 + w_{52}x_2 + w_{53}x_3 + b_5$$

Convert to a Matrix

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

- For the next layer, Z becomes the input and there will be a new set of weights.
- Thus, one can see why w is chosen, as it stands for “weight” matrix.
- Activation is done element-wise. Each z is sent through the activation function, and the result is stored as the output.

Why Is this Important?

- Rather than a mathematical sideshow, getting matrices to the correct size is one of the methods of troubleshooting neural networks.
- Neural networks are typically built and configured. To optimize speed, calculations are done via optimized matrix multiplication.
  - If the matrices cannot be multiplied, libraries will throw errors or not work.
  - This can be as simple as failing to convert a row matrix to a column matrix.

DataScience@SMU

Forward and Backwards

DataScience@SMU

Forward

- Getting a prediction is called a “forward pass”
- Data is passed through the network
  - All multiplications and activations are done.
  - We get predictions from the output layer.
  - The loss is calculated.
- Done at the batch level
  - Batch_size = 32, 32 predictions are made, and total loss for all 32 is calculated.
- Typically **very, very** fast

Backwards

- The backwards pass is when the weights are updated.
- Using the loss function supplied, the network uses backpropagation to calculate the updates to each and every weight in the network.
- Backpropagation is done by the library for you.
- Backpropagation is **slow**.
  - Done during training (or using the “fit” method)
- After every forward pass in training, there is a backwards pass to update layer weights.

Vanishing Gradient

- Vanishing gradient is a term you will see here for deep neural networks.
- As networks get more and more layers, the layers closest to the input update very slowly.
  - Part of backpropagation is that to update the weights we need the derivative. However, the early weights are deep inside our prediction equation. This means we will be multiplying them by small numbers repeatedly.
  - Small x small = very small.
- Thus, networks with more than 5–6 layers do not give much improvement.

DataScience@SMU

Building Networks

Neural Networks Need to Be Built

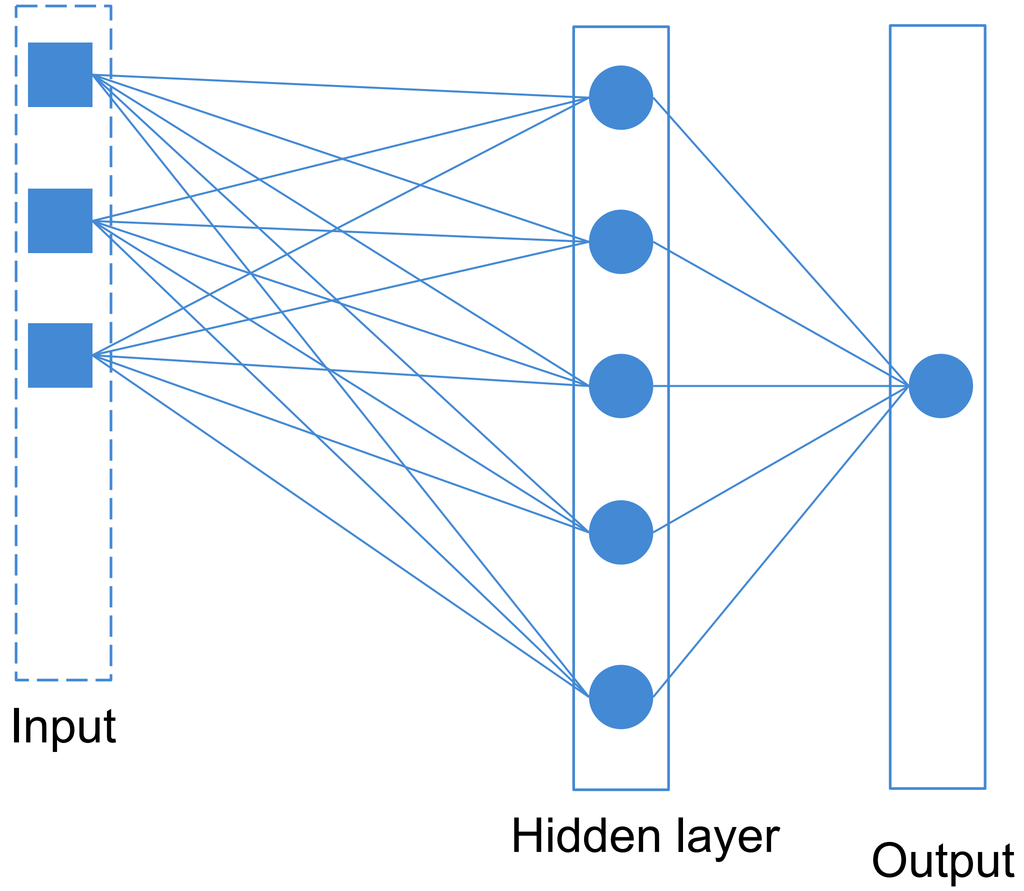
- Unlike most other algorithms, neural networks (NN) need to be built or put together before they are run.
- Building or architecture is basically another hyperparameter to tune.
- NN are put together by layers.
- Build the network layer by layer.

The Dense Layer

- The most basic layer
- Used in almost every network
- Need to pick a number of neurons and an activation function

Revisiting Our Initial Diagram

- This network has two layers.
- The first (hidden) is a 5 neuron layer.
- The second is the output (a single neuron).
- Note that the activation function is not mentioned here (and is almost never mentioned in diagrams).
- When building a NN, you must pick an activation function.
- The input is not technically a layer, but often to build a NN, it is required to designate the input dimensions.



Optimizers

- An optimizer is what updates the weights of the NN.
- Optimizers do the backwards pass.
- Stochastic gradient descent was one of the first optimizers (and often the default).
- Best practices are to use more “modern” optimizers like “adam” and “RMSprop.”
- We need to tell the model what optimizer to use **and** what loss function we are optimizing.

Steps to Build a Model

- State the input dimensions.
- Set up the layers and how they are connected.
- Specify an optimizer with a loss function.
- Set the batch size and number of epochs.
- Then, we can train the model.

DataScience@SMU