

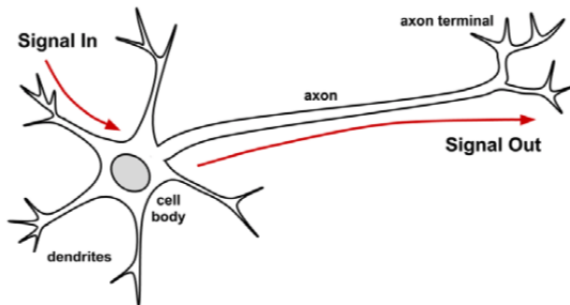
# Lecture 19: Neural Networks

## STAT 452, Spring 2021

# Introduction

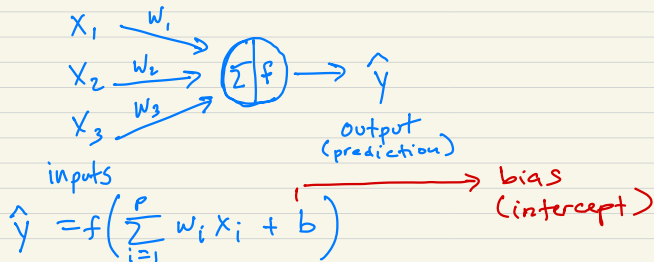
- ▶ An **Artificial Neural Network** (ANN) models the relationship between a set of input and output signals using a model inspired from our understanding of the human brain.
- ▶ The original idea behind neural networks was to use a computer-based model of the human brain to perform complex tasks. We can recognize people in fractions of a second, but this task is difficult for computers. So why not make software more like the human brain?
- ▶ ANNs are considered black box methods, since the network model is difficult to interpret, but can have great predictive performance.

Because ANNs were intentionally designed as conceptual models of human brain activity, it is helpful to first understand how biological neurons function. As illustrated in the following figure, incoming signals are received by the cell's **dendrites** through a biochemical process that allows the impulse to be weighted according to its relative importance or frequency. As the cell body begins to accumulate the incoming signals, a threshold is reached at which the cell fires and the output signal is then transmitted via an electrochemical process down the **axon**. At the axon's terminals, the electric signal is again processed as a chemical signal to be passed to the neighboring neurons across a tiny gap known as a **synapse**.



Passage from Brett Lantz. *Machine Learning with R*. Chapter 7.

- A perceptron is a model of a single neuron
- It's the basic building block of a neural network

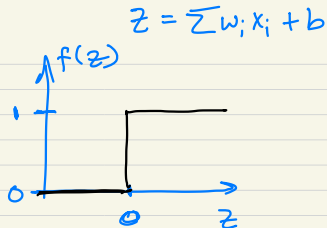


- The output, or prediction, is a weighted sum of inputs
- The function  $f()$  is called activation function

## Activation Functions

### - Threshold activation function

$$f(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$



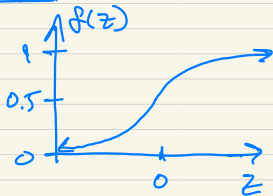
if  $\sum w_i x_i + b < 0 \Rightarrow$  outputs 0

if  $\sum w_i x_i + b \geq 0 \Rightarrow$  outputs 1

• Sigmoid activation function

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

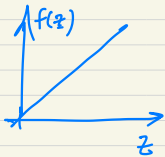
(equivalent to  
logistic regression)



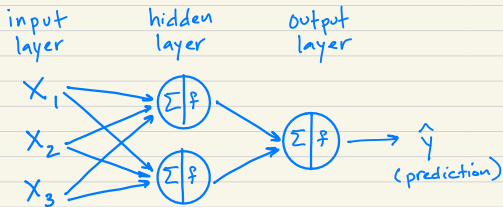
• Linear (identity) activation function

$$f(z) = z$$

(equivalent to multiple  
linear regression)



## Feedforward neural network with one hidden layer



- Each arrow represents a weight (strength of connection)
- New features are derived on each node of the hidden layer, which are each a weighted sum of inputs, transformed by activation function
- The output is then another weighted sum of features from the hidden layer, transformed by activation function

# Network Architecture

The **network architecture** (or **topology**) describes the number of artificial neurons (nodes) in the model as well as the number of layers and the manner in which they are connected.

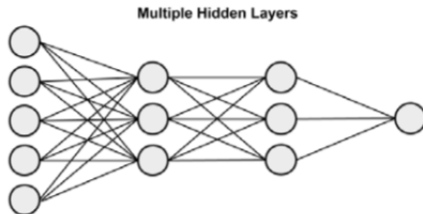
A NN has three types of layers:

- ▶ **input layer**, which are just the original predictor variables
- ▶ **hidden layer**
- ▶ **ouput layer**



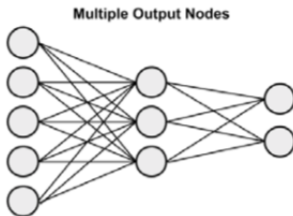
# Hidden Layer

- ▶ A neural network can have multiple hidden layers.
- ▶ Having more than one hidden layer can be useful when dealing with high dimensional data (e.g., images, video).
- ▶ The number of nodes in a hidden layer is a tuning parameter (left to the user to decide).
- ▶ Generally, the more nodes in a hidden layer, the greater the flexibility of the NN to model nonlinearities in the data. But too many nodes can potentially overfit the data.
- ▶ Often the number of nodes in a hidden layer is less than, or equal to, the number of features (predictor variables), but this is not a hard requirement.



# Output Layer

- ▶ The choice of output layer is driven by the modeling task.
- ▶ For regression problems, the output layer will contain one node that outputs the final predicted value.
- ▶ For binary (0/1) classification, output layer will still contain only one node and that will give the predicted probability,  $Pr(Y = 1|\mathbf{x})$ .
- ▶ For classification problems with more than two categories, the output layer will contain the same number of nodes as the number of classes being predicted.
  - ▶ For example, in the MNIST data, we are predicting 10 classes (0–9); therefore, the output layer will have 10 nodes and the output would provide the probability of each class.



# Activation Functions

Here are some of the most common activation functions for the nodes in the hidden and output layers of a NN:

- ▶ Linear (identity):

$$f(z) = z$$

- ▶ Rectified linear unit (ReLU):

$$f(z) = \max(0, z) = \begin{cases} 0, & \text{if } z \leq 0, \\ z, & \text{if } z > 0 \end{cases}$$

- ▶ Sigmoid:

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

- ▶ Softmax:

$$f_k(z) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

# Activation Functions

- ▶ The ReLU and sigmoid are commonly used in the hidden layers.
- ▶ For regression problems, we use the linear activation function in the output layer.
- ▶ For binary classification, we use the sigmoid activation function in the output layer.
- ▶ For multinomial classification problems (predicting  $k = 1, \dots, K$  categories), we use the softmax activation function in the output layer.


# Fitting Neural Networks

- ▶ Neural networks have many parameters, called **weights**, that need to be estimated. Recall, the weights represent the strengths of the connections between the artificial neurons (nodes).
- ▶ Neural networks were first proposed in the 1950's, but an efficient method to learn, or estimate, the weights called **backpropagation** was not developed until the 1980's.

# Fitting Neural Networks

- ▶ Backpropagation is an iterative method. At each iteration, or **epoch**, there are two phases:
  - ▶ A **forward phase** where predictions for the response are made using the current set of weights, and the performance is measured according to some **loss function** (e.g., MSE) that we are trying to minimize.
  - ▶ A **backwards phase** where the weights are updated, or adjusted, in a way that improves performance and reduces the error.
- ▶ The weights are adjusted during the backwards phase using a method called **gradient descent**, which involves evaluating the derivatives of the loss function with respect to the weights.
- ▶ Before running the algorithm, the weights are usually initialized at random values.
- ▶ The mathematical details of fitting a neural network using backpropagation are beyond the scope of the class.<sup>1</sup>

---

<sup>1</sup>See *Elements of Statistical Learning*, pp. 393–397, for details 

# Concluding Remarks

- ▶ NNs are best applied to problems where the input and output data are well-understood, yet the process that relates the input to the output is extremely complex (e.g., facial recognition, computer vision).
- ▶ Generally, NNs perform best on large data sets that have many of rows (large  $n$ ) and many features (large  $p$ ).
- ▶ NNs also involve lots of tuning, for instance, when trying to decide on the number of nodes and hidden layers, and types of activation functions.
- ▶ For more moderately sized data sets, other techniques that we have covered in this class (e.g., KNN, decision trees, random forests) are easier to implement and often perform as well, or perhaps better than a NN.
- ▶ NNs are black-box models, so if explanation and interpretation are the goal of a data analysis, NNs are not a good choice.

# References

- ▶ Brett Lantz. *Machine Learning with R*. Chapter 7. [PDF on Blackboard]
- ▶ Bradley Boehmke and Brandon Greenwell. *Hands-On Machine Learning with R*. Chapter 13.  
<https://bradleyboehmke.github.io/HOML/deep-learning.html>
- ▶ Trevor Hastie, Robert Tibshirani, Jerome Friedman. *Elements of Statistical Learning*. Second Edition. Chapter 11.
- ▶ Julian Faraway. *Extending the Linear Model*. Chapter 14.