ECS 171: Machine Learning

Summer 2023
Edwin Solares
easolares@ucdavis.edu
Logistic Regression & Neural Network Introduction

Waymo: Autonomous Driving using NN Classification

Visualizing the Math

m x n * n x 1 matrix multiplication creates an m x 1 vector

$$w_0 + \quad X \quad w = \hat{y}$$

$$w_0 + \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n} \\ \cdots & \cdots & \cdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_m \end{bmatrix}$$

Visualizing the Math

$$
\begin{array}{ccc}
X & w & = \hat{y} \\
\begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,n} \\ 1 & x_{2,1} & \dots & x_{2,n} \\ \dots & \dots & \dots & \dots \\ 1 & x_{m,1} & \dots & x_{m,n} \end{bmatrix} & \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_n \end{bmatrix} & = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_m \end{bmatrix}
\end{array}
$$

$$
\begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,3} \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = w_0 + w_1 x_{1,1} + w_2 x_{1,2} + w_3 x_{1,3} = \hat{y}_i
$$

Simple Linear Regression Function

$$X \quad w = \hat{y}$$

$$\begin{bmatrix} 1 & x_{1,1} \\ 1 & x_{2,1} \\ \dots & \dots \\ 1 & x_{m,1} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \dots \\ \hat{y_m} \end{bmatrix}$$

1st Order Simple Polynomial Regression

2nd Order Polynomial Regression

$$\begin{array}{c} \begin{array}{ccc} & n = 1 & n = 2 \end{array} \\ \begin{bmatrix} 1 & x_{1,1} & (x_{1,1})^2 \\ 1 & x_{2,1} & (x_{2,1})^2 \\ \cdots & \cdots & \cdots \\ 1 & x_{m,1} & (x_{m,1})^2 \end{bmatrix} \end{array} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_m \end{bmatrix}$$

m^th Order Polynomial Regression

$$
\begin{array}{ccc}
X & w & = & \hat{y} \\
\begin{bmatrix}
1 & x_{1,1}^{1} & x_{1,2}^{2} & \cdots & x_{1,n}^{n} \\
1 & x_{2,1}^{1} & x_{2,2}^{2} & \cdots & x_{2,n}^{n} \\
\cdots & \cdots & \cdots & \cdots\cdot & \cdots \\
1 & x_{m,1}^{1} & x_{m,2}^{2} & \cdots & x_{m,n}^{n}
\end{bmatrix}
&
\begin{bmatrix}
w_0 \\ w_1 \\ w_2 \\ \cdots \\ w_n
\end{bmatrix}
& = &
\begin{bmatrix}
\hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_m
\end{bmatrix}
\end{array}
$$

Standard Logistic Growth Function

$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$

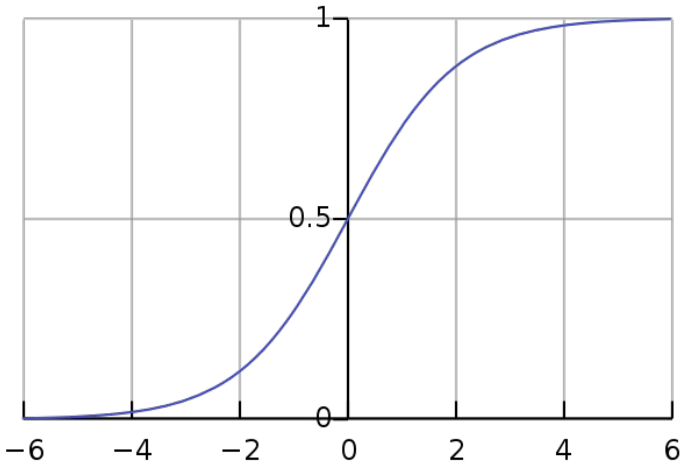
$f(x)$ = output of the function

L = the curve's maximum value

k = logistic growth rate or steepness of the curve

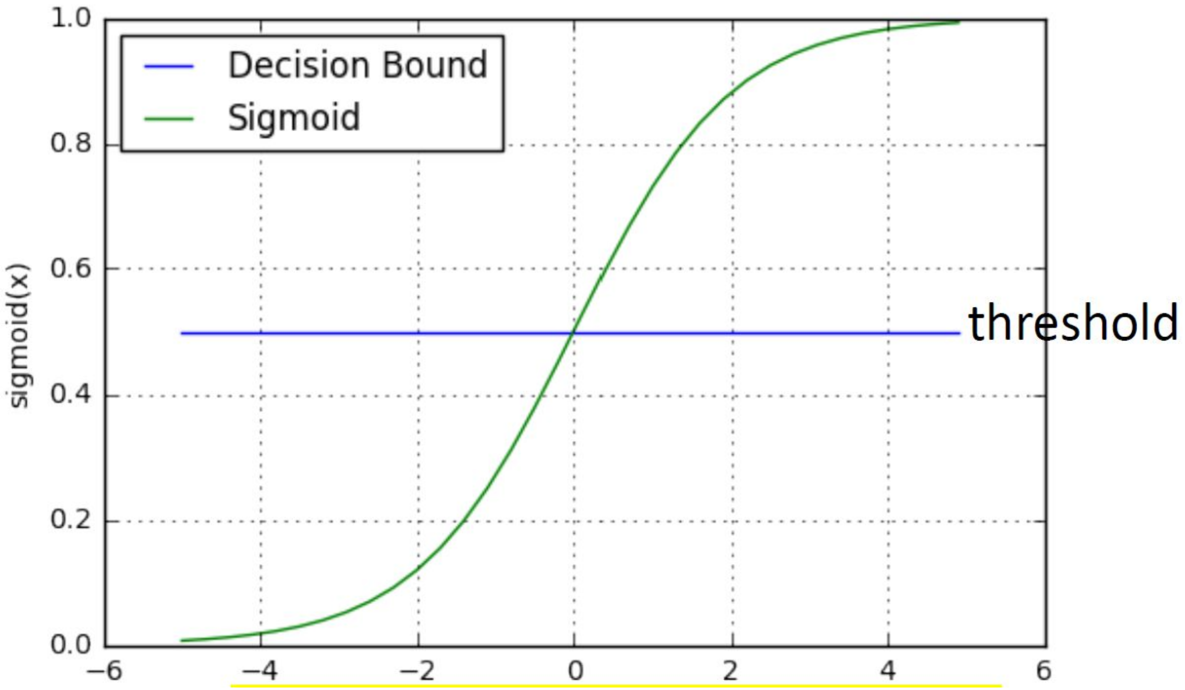
x_0 = the x value of the sigmoid midpoint

x = real number



$$L = 1, k = 1, x_0 = 0.5$$

Logistic Regression



$$p_k = \begin{cases} 0 & ; predicted\ value < thresold \\ 1 & ;\ predicted\ value\ \geq threshold \end{cases}$$

Logistic Regression

$$\hat{y} = \frac{1}{1 + e^{-\frac{1}{s}(x-\mu)}}$$

Substitute

$$\hat{y} = \frac{1}{1 + e^{-(w_0+w_1x)}}$$

where $w_0 = -\mu/s$ and $w_1 = 1/s$ $\therefore$ we can solve for μ and s

$$\mu = -w_0/w_1 \text{ and } s = 1/w_1$$

Logistic Regression

Where we have Bernoulli observations
And p_k is the probability of y_k=1 and
$1\text{-}p_k$ is the probability $y_k = 0$

The log loss for the *k*-th point is:

$$\begin{cases} -\ln p_k & \text{if } y_k = 1, \\ -\ln(1 - p_k) & \text{if } y_k = 0. \end{cases}$$

Cross Entropy

For the observed distribution of $(y_k, 1 - y_k)$ and the predicted distribution of $(p_k, 1 - p_k)$, we get a probability distribution:

$$-y_k \ln p_k - (1 - y_k) \ln(1 - p_k)$$

log-likelihood

MAXIMIZE!

$$\ell = \sum_{k=1}^{K} (y_k \ln(p_k) + (1 - y_k) \ln(1 - p_k))$$

Derivate Flashback

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

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{e^x \cdot (1+e^x) - e^x \cdot e^x}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2} = f(x)\big(1 - f(x)\big)$$

Logistic Regression: Parameter Estimation

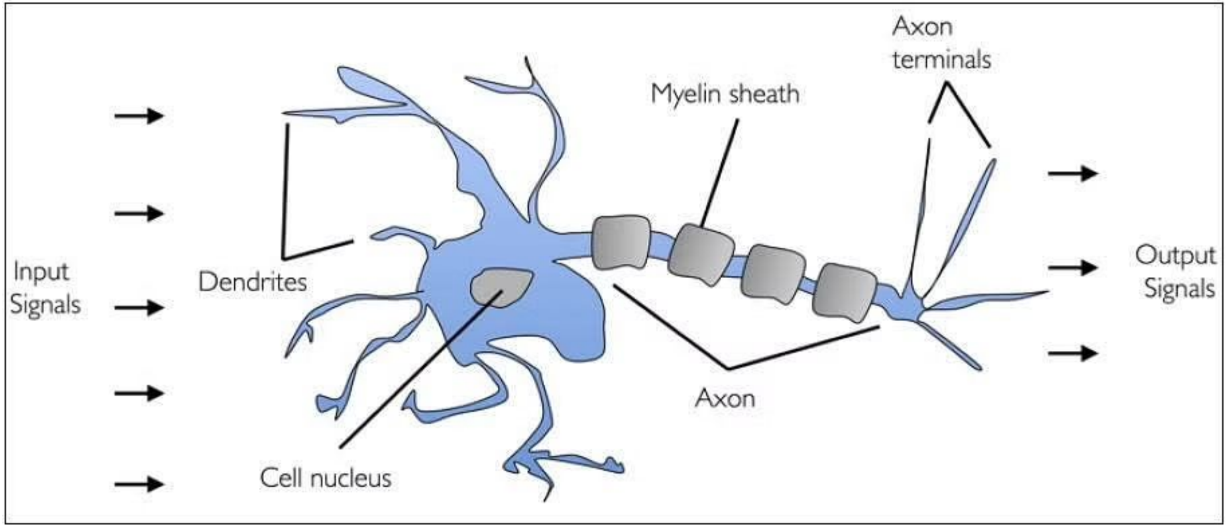
$$0 = \frac{\delta l}{\delta w_0} = \sum_{k=1}^{m} (y_k - p_k)$$

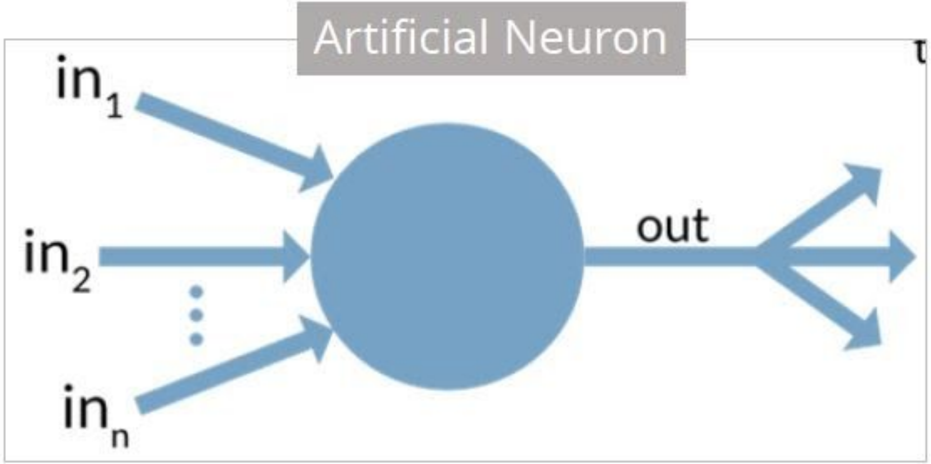
$$0 = \frac{\delta l}{\delta w_1} = \sum_{k=1}^{m} (y_k - p_k) x_k$$

Optimization of weights

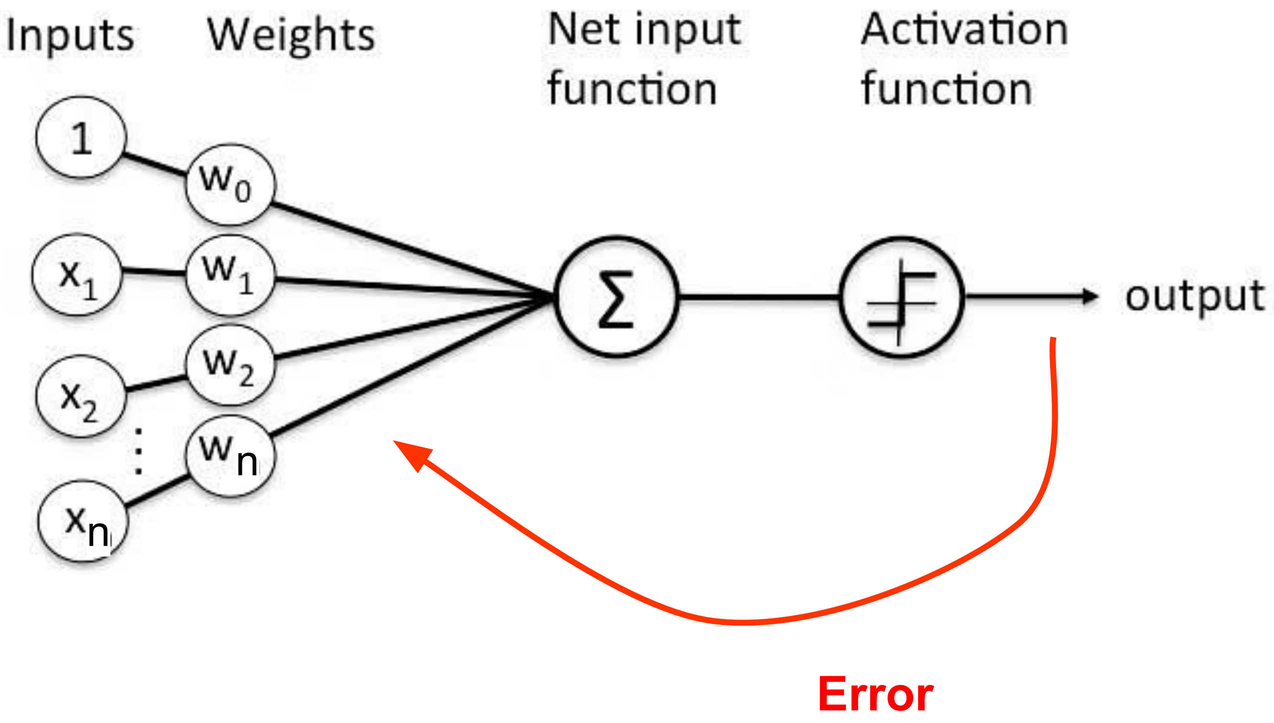
Gradient Descent!!!

Perceptron


Axon
terminals
Myelin sheath
Input
Signals
Dendrites
Output
Signals
Axon
Cell nucleus

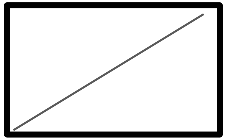

Artificial Neuron
in1
in2
inn
out

Perceptron

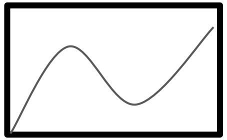

Inputs
Weights
Net input function
Activation function
1
w_0
x_1
w_1
x_2
w_2
w_n
x_n
Σ
output
Error

Activation Functions

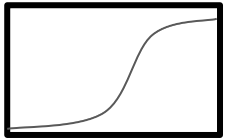
Linear



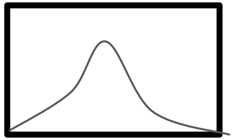
Polynomial



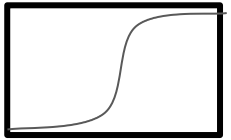
Logistic



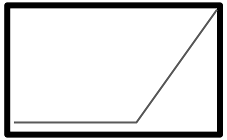
Gaussian



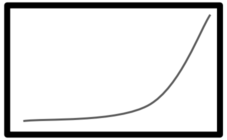
Sigmoid



ReLU (Rectified Linear Unit)



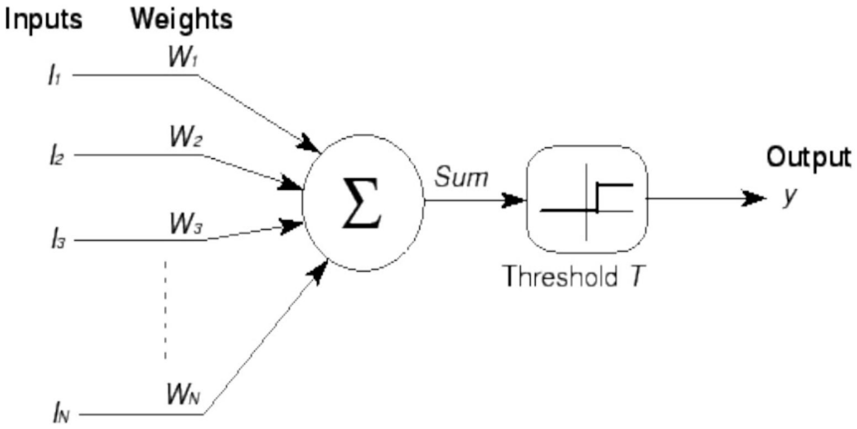
SoftMax



https://cs231n.github.io/neural-networks-1/
https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#elu

Artificial Neuron History

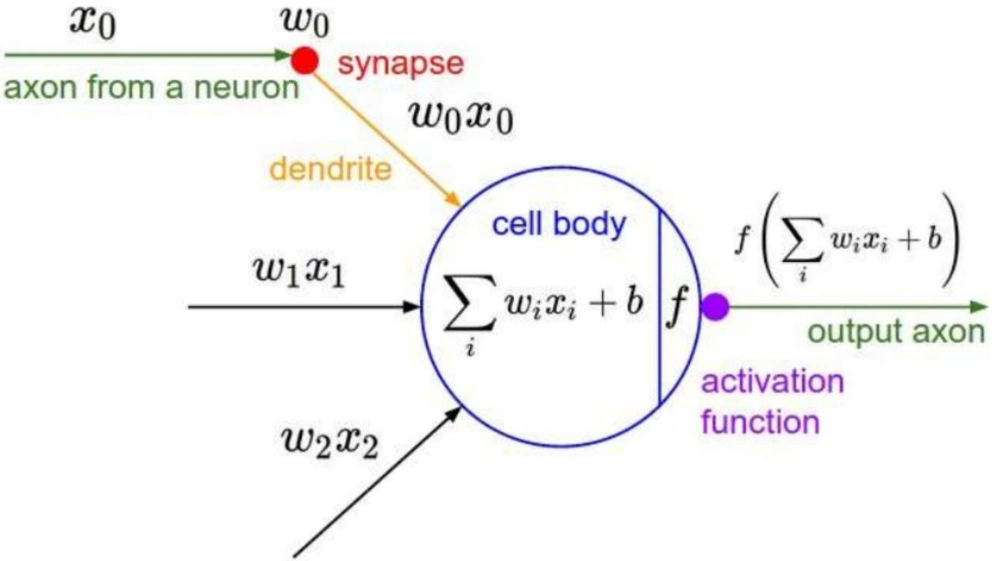
1st Generation Neuron (McCulloch-Pitts)



https://www.slideshare.net/hitechpro/introduction-to-spiking-neural-networksfrom-a-computational-neuroscience-perspective/30

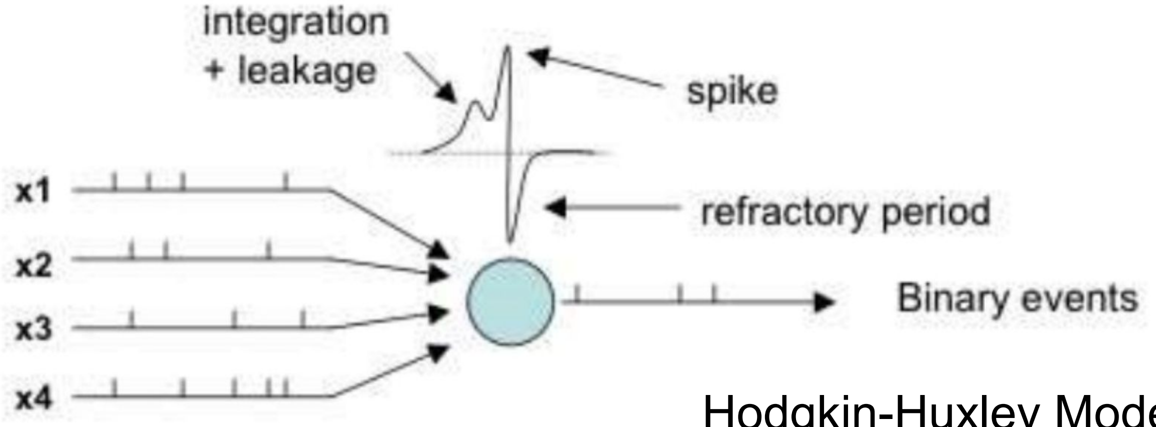
Artificial Neuron History

2nd Generation Neuron



https://www.slideshare.net/hitechpro/introduction-to-spiking-neural-networksfrom-a-computational-neuroscience-perspective/30

3rd Generation Neuron (Spiking Neurons)

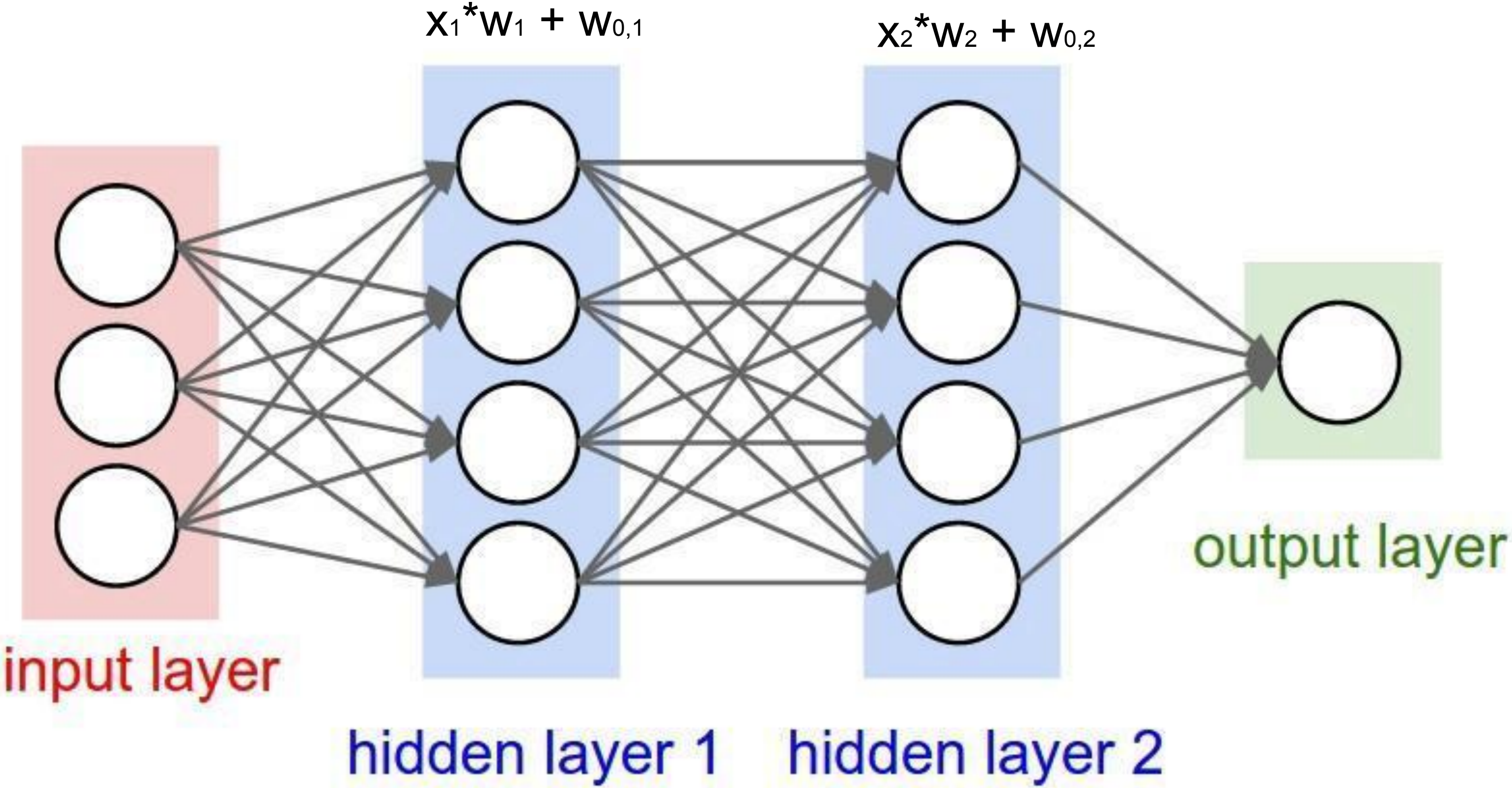


Hodgkin-Huxley Model
Izhikevish Model
Leakage Integrate-and-Fire Model

https://www.slideshare.net/hitechpro/introduction-to-spiking-neural-networksfrom-a-computational-neuroscience-perspective/30

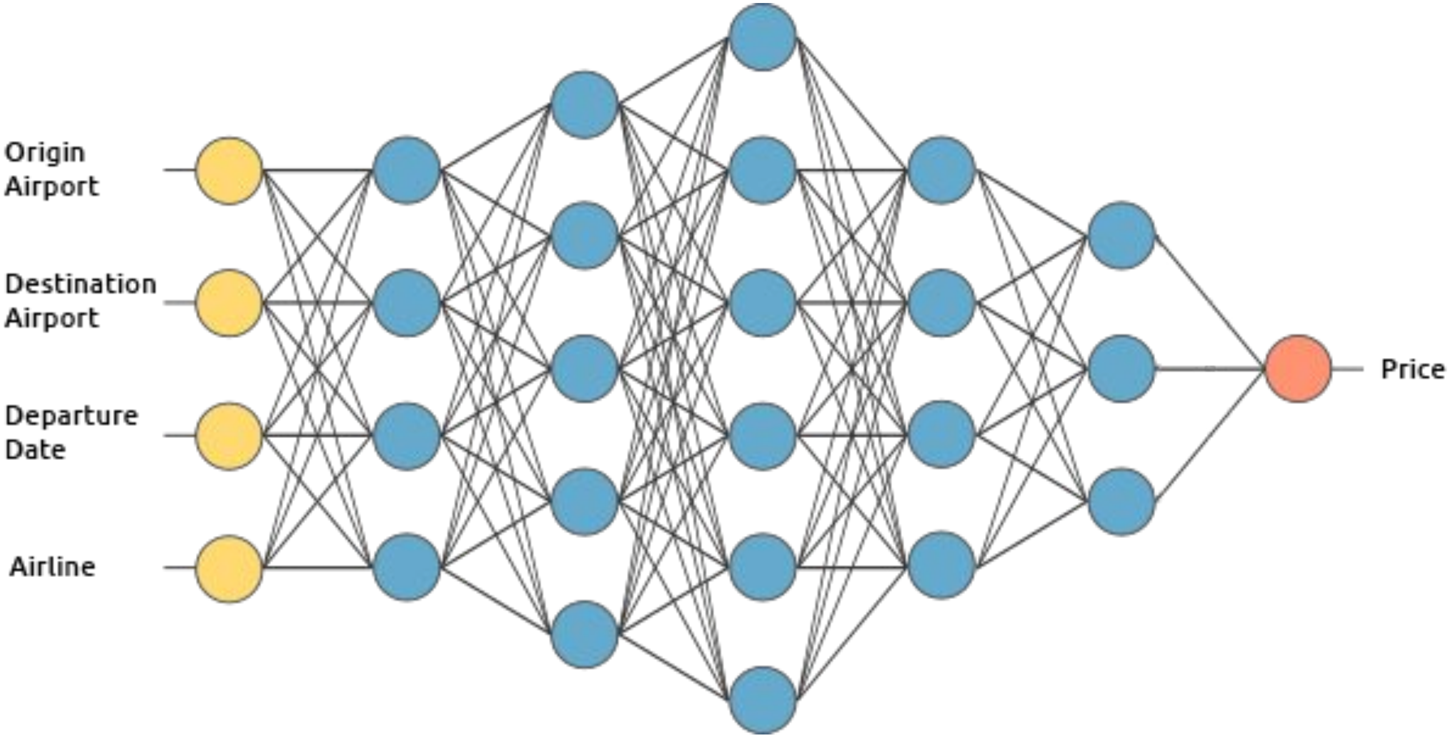
Courtesy of Nvidia Website

Simple Neural Net: 2 Hidden Layers



Courtesy of Nvidia Website

Deep Neural Net: Several Hidden Layers



In Depth Relatable NN Example

https://www.youtube.com/watch?v=CqOfi41LfDw

Try running NN's on sample data:
BCC Data:
https://www.youtube.com/watch?v=_VTtrSDHPwU
California Housing Data:
https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/intro_to_neural_nets.ipynb

Jupyter Notebooks Time!

http://playground.tensorflow.org/