Machine Learning
Introduction to Neural Networks

Faculty:
Farshid Alizadeh-Shabdiz, PhD, MBA
Sept 2021

How Does Brain Work?

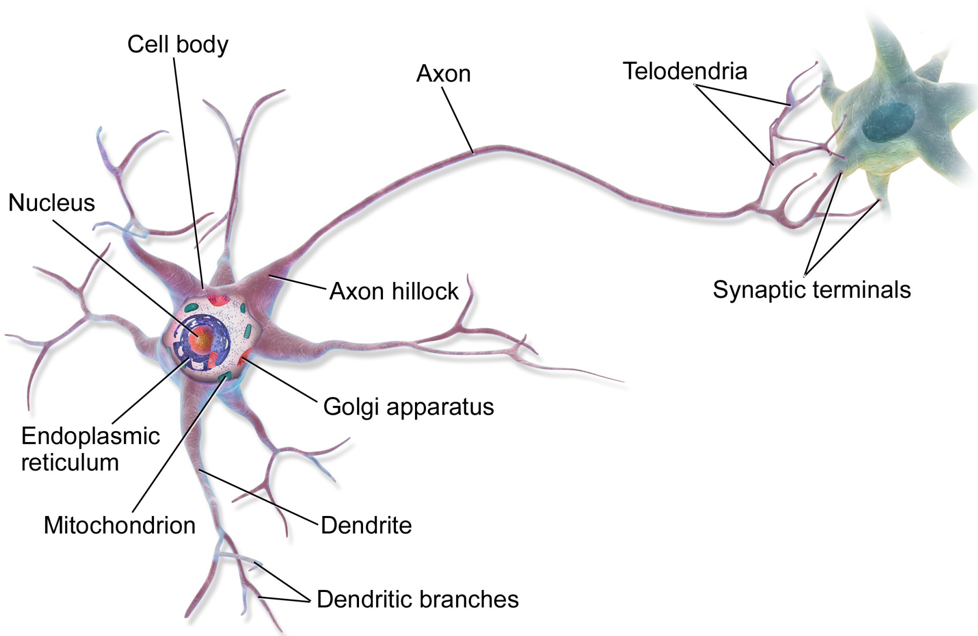
Why Study Brain?

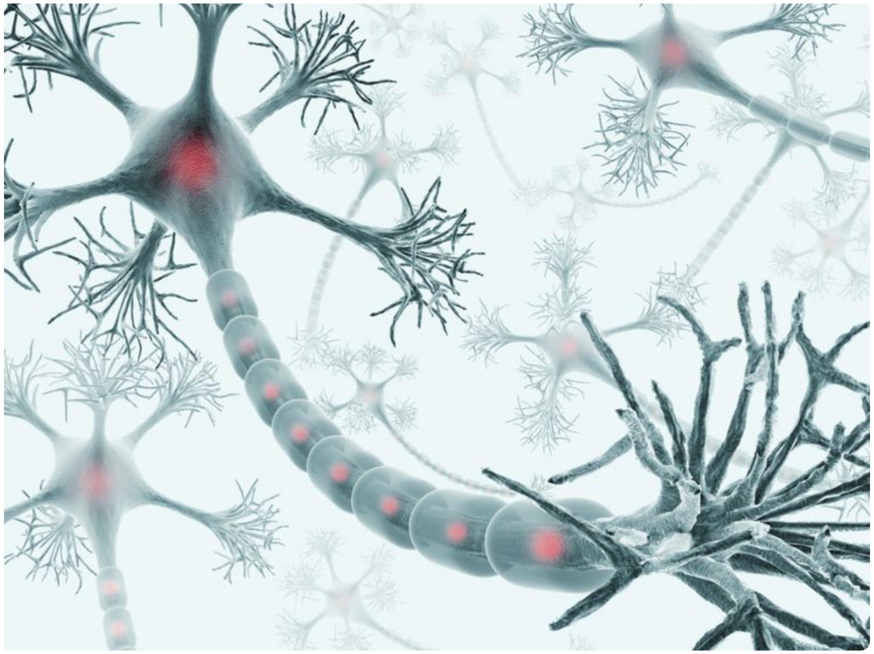
- Brain does many complex tasks extremely well, like
  - Vision and object detection
  - Speech recognition
- Human mind learns new things by observation and combining different data sets
- Brain inferences based on evidence and reasoning, which inspires new novel learning algorithms
- Brain parallel computation and adaptive learning would inspire new designs

- Note that human mind is not good in everything
  - Like multiplication of multi-digit numbers

Cortical Neuron Structure

- Body of the cell
- Dendrites: extension of the cell with many branches to receive signals
- Axon: very long extension of the neuron body, which carries electric signal spike to other cells
- Telodendria: branching extensions at the end of Axon to connect to Dandrites
- Synapses, where axon meet dendrites.
- Neurotransmitters: chemical signals released by signals at the end of an Axon





Science Photo Library – KTSDESIGN/Getty Images

https://en.wikipedia.org/wiki/Neuron

How Cortical Neurons Work

- Axon is an arm of a neural cell which transfers electricity from the cell to the next
  - A spike is transferred at 0.5 – 2.0 m/s (or 1.1 – 4.5 miles/hour)
  - Average length of cortical axon: 86.8 mm
- A spike of an axon injects charge into the post-synaptic neuron at synapse
- When enough charges get injected into the post synaptic neuron, it depolarizes the cell membrane and generates outgoing spike

https://en.wikipedia.org/wiki/Neuron

Synapsis Structure

- Injection of charges by axon, causes vesicles (a liquid enclosed by a lipid bilayer) of transmitter chemical to be released. There are positive and negative transmitter vesicles.
- The transmitter molecules diffuse across the synaptic cliff
- After moving to the other side, they bind to receptor molecules of the post synaptic neuron
- By binding with receptor molecules, it changes their shape and creates holes for positive and/or negative ions to cross to the post-synapsis.
- After accumulation of enough charges, the electric spike get generated
- Synapsis stats
  - 10^{11} cortical neurons
  - 10^{4} average synaptic connections

https://en.wikipedia.org/wiki/Neuron

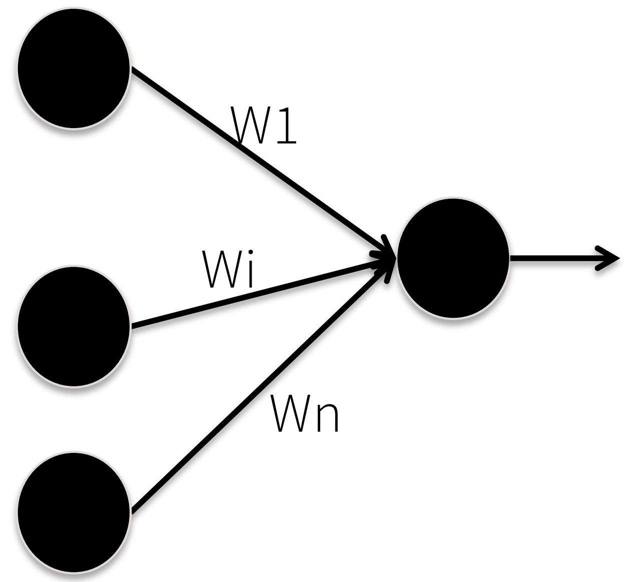
Synapsis Adapt

- Learning occurs by changing synapsis
- The synapsis learn by changing
  - Number of vesicles
  - Number of receptor molecules
  - Also long term number of neurons connections
- Synapsis are slow but they are also very low power compare to our technology, **but they adapt**
  - Synapsis adapt to locally available signals

https://en.wikipedia.org/wiki/Neuron

Brain as a System

- Neurons
  - Send messages with electric spikes
  - 10^4 weights that adapt
  - Many neurons provide input to Dendritic tree
  - Small number of neurons connect to receptors
- Brain learns to be modular. Each function is concentrated in a region
- Early age damage might relocate a function
- All the neurons are the same, but they become specialized in action



History of Neural Network

History of Neural Networks

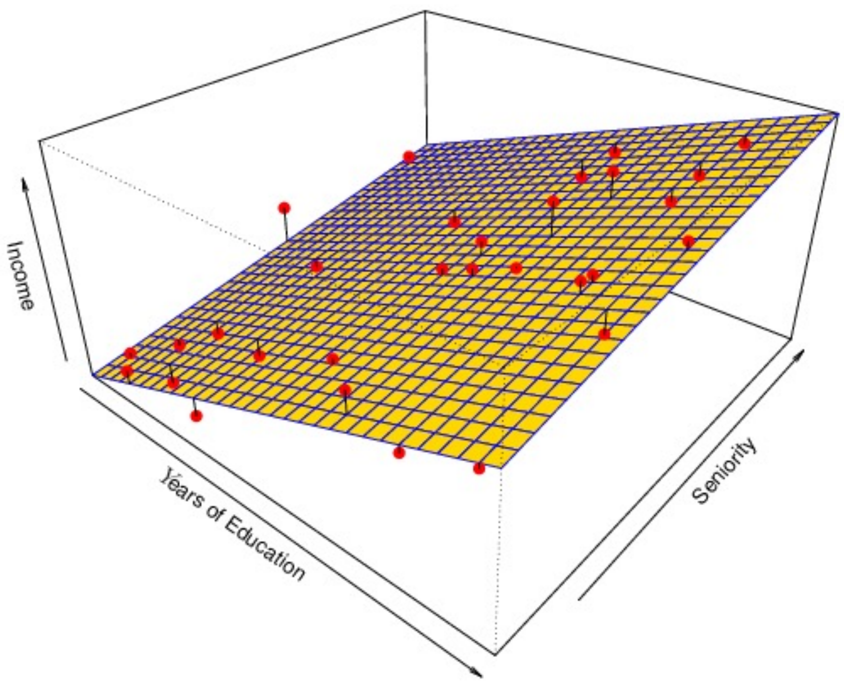
- 1943: McCulloch & Pitts created a computer model based on neural networks of brain
- 1949: Hebb's Rule – Hebb suggested how biological neurons work
- 1957: Perceptron invented by Frank Rosenblatt
- "The stuff promised in this video – still not really around" (1961). https://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/
- 1969: M Minsky & S. Papert showed limitation of perceptron
- 1970's: the first winter of AI
- 1986: D. Rumelhart, G. Hinton, and R. Williams that introduced backpropagation (introduced in 60's first) algorithm to train MLP.
- 1985-1990: the 2nd winter of AI
- 1996: IBM Deep Blue beat Kasparov, world chess champion
- 1997: LSTM (long-short term memory) for recurrent NN was developed
- 1999-2001: GPU and processing data were developed
- 2000 : Winter of AI
- 2006: Prof Hinton trained a network to read handwritten numbers
- 2012: AlexNet CNN architecture won the ImageNet challenge with a large margin (17% error vs next best 26%)
  - 2009: Fei-Fei Li, AI prof at Stanford launched ImageNet (14 million labeled images).

When to Use Neural Network

- Need lot of data
- Need training data
- It takes time
- No model needed
- Hard to verify
- Hard to interpret
- Learn complex problems

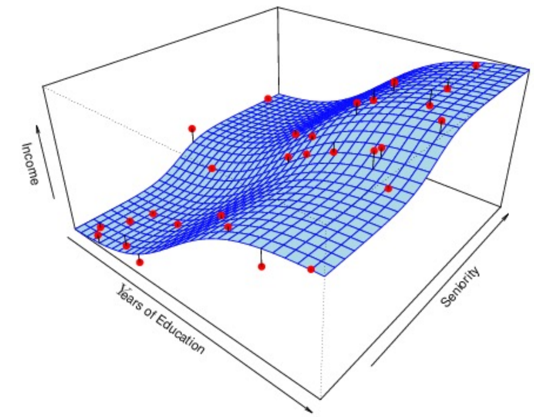
Simple Models of Neurons

Linear Regression - overview



- Linear Regression Model fit to Income vs Education & Seniority

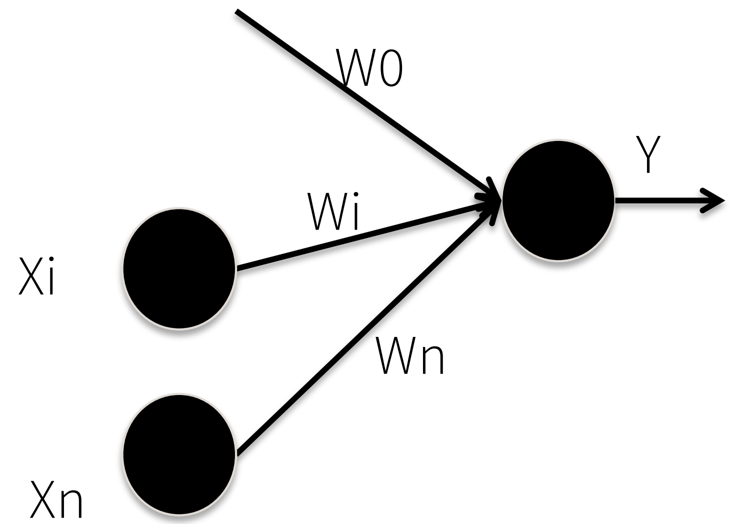
$$f = \beta_0 + \beta_1 \times Education + \beta_2 \times Seniority$$



Linear Neurons

- Simple and easy to analyze

$$y = w_0 + \sum_i w_i x_i$$

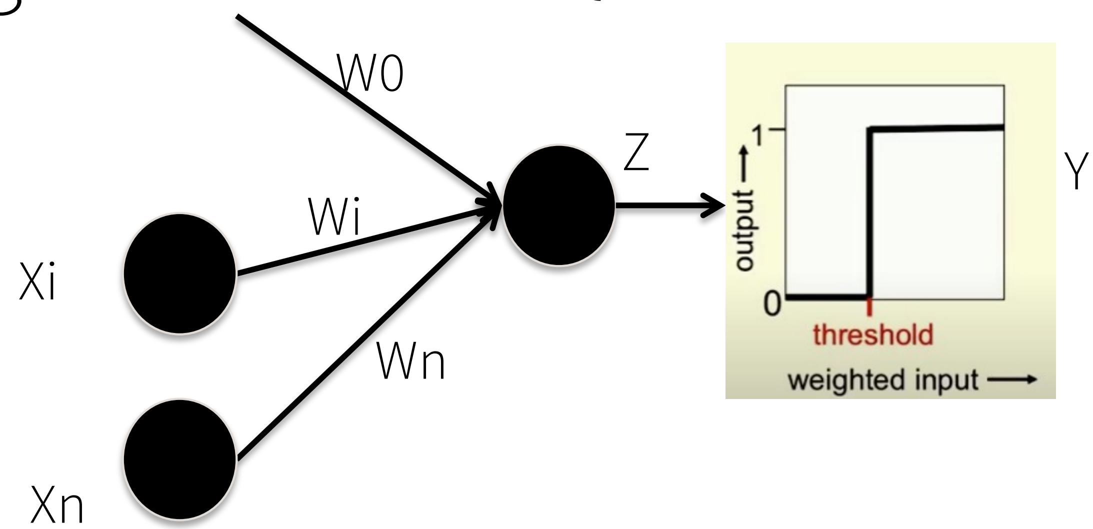


Binary Threshold Neurons

- Based McCulloch-Pitts (1943)
- This is based on the logical paradigm

$$z = w_0 + \sum_i w_i x_i$$

$$y = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ Otherwise} \end{cases}$$



Rectified Linear Neurons (ReLU)

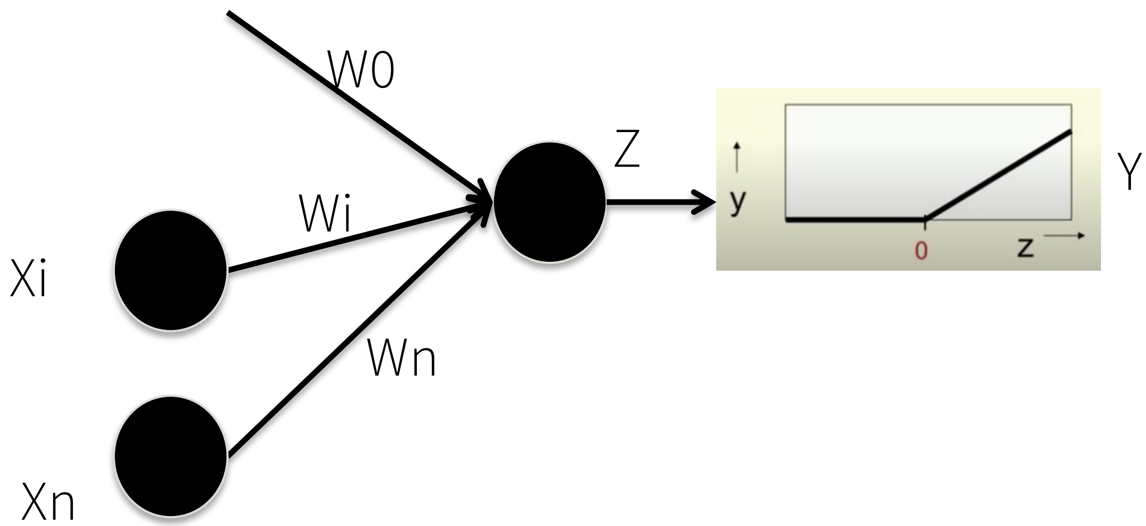
- Non-linear output
- Linear property above zero + being able to make a decision

$$z = w_0 + \sum_i w_i x_i$$

$$y = \begin{cases} z \text{ if } z \geq 0 \\ 0 \text{ Otherwise} \end{cases}$$

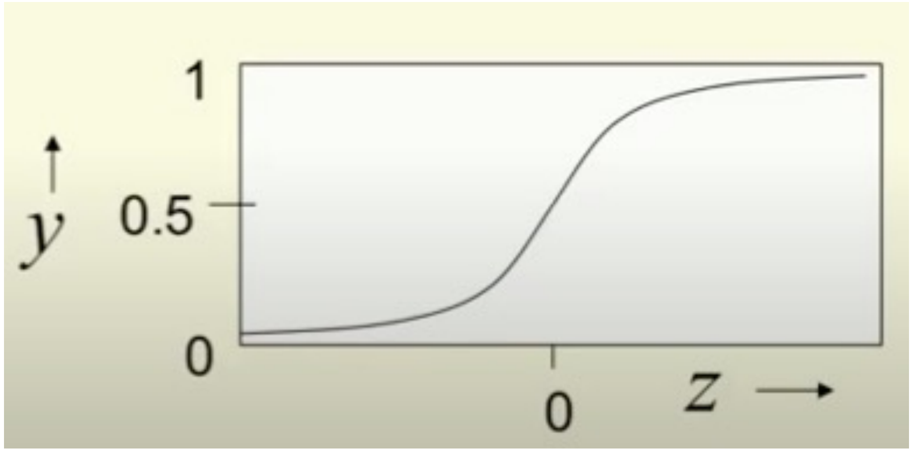
OR

$$y = \max(0, z)$$



Logistic Regression

- Weighted sum of inputs same as Linear Regression + logistic function (Sigmoid function)



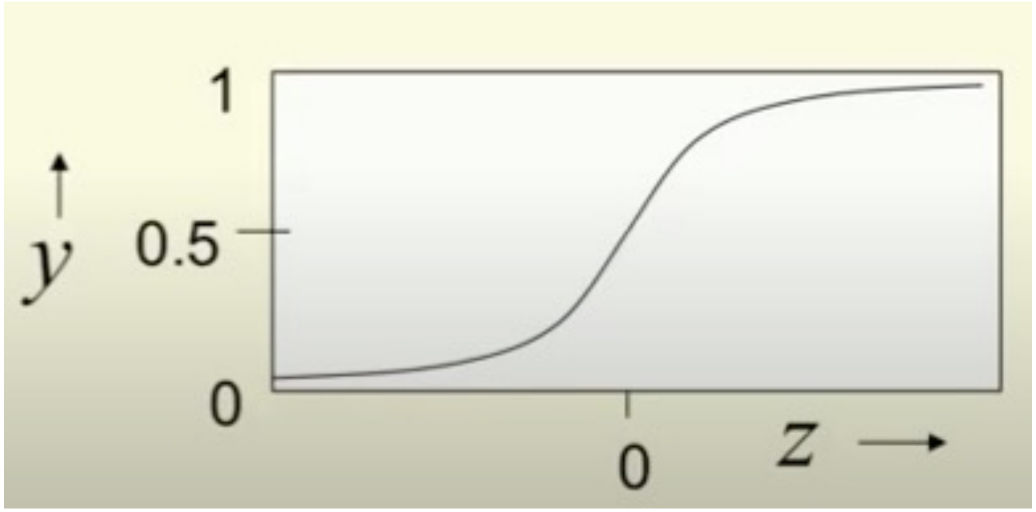
Sigmoid Neurons

- The most common neuron
- Nice derivative, which makes learning easy

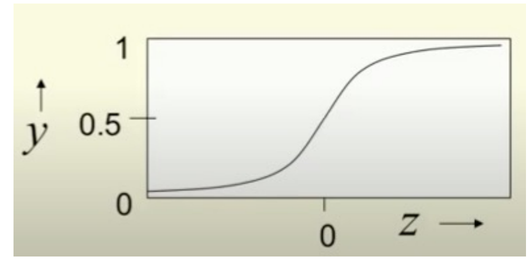
$$z = w_0 + \sum_i w_i x_i$$

$$y = \frac{1}{1 + e^{-z}}$$

Note: also common in logistic regression



Sigmoid Derivative



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

$$Derivative = \frac{df(z)}{d(z)} = \frac{d}{d(z)}[(1+e^{-z})^{-1}]$$

$$\frac{df(z)}{d(z)} = \frac{e^{-Z}}{(1+e^{-Z})^2} = \frac{1+e^{-Z}-1}{(1+e^{-Z})^2}$$

$$\frac{df(z)}{d(z)} = \frac{1}{1+e^{-Z}} - \frac{1}{(1+e^{-Z})^2} = f(z)[1-f(z)]$$

Logistic Neuron Learning

To apply a learning algorithm, derivative of output with respect to weights has to get calculated : $\frac{\partial y}{\partial w_i}$

- Using the chain rule, we can find

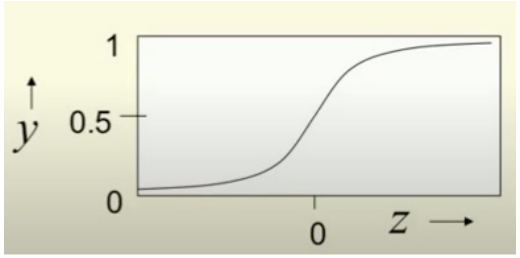
$$\frac{\partial y}{\partial w_i} = \frac{\partial y}{\partial z} \times \frac{\partial z}{\partial w_i} = xiy(1-y)$$

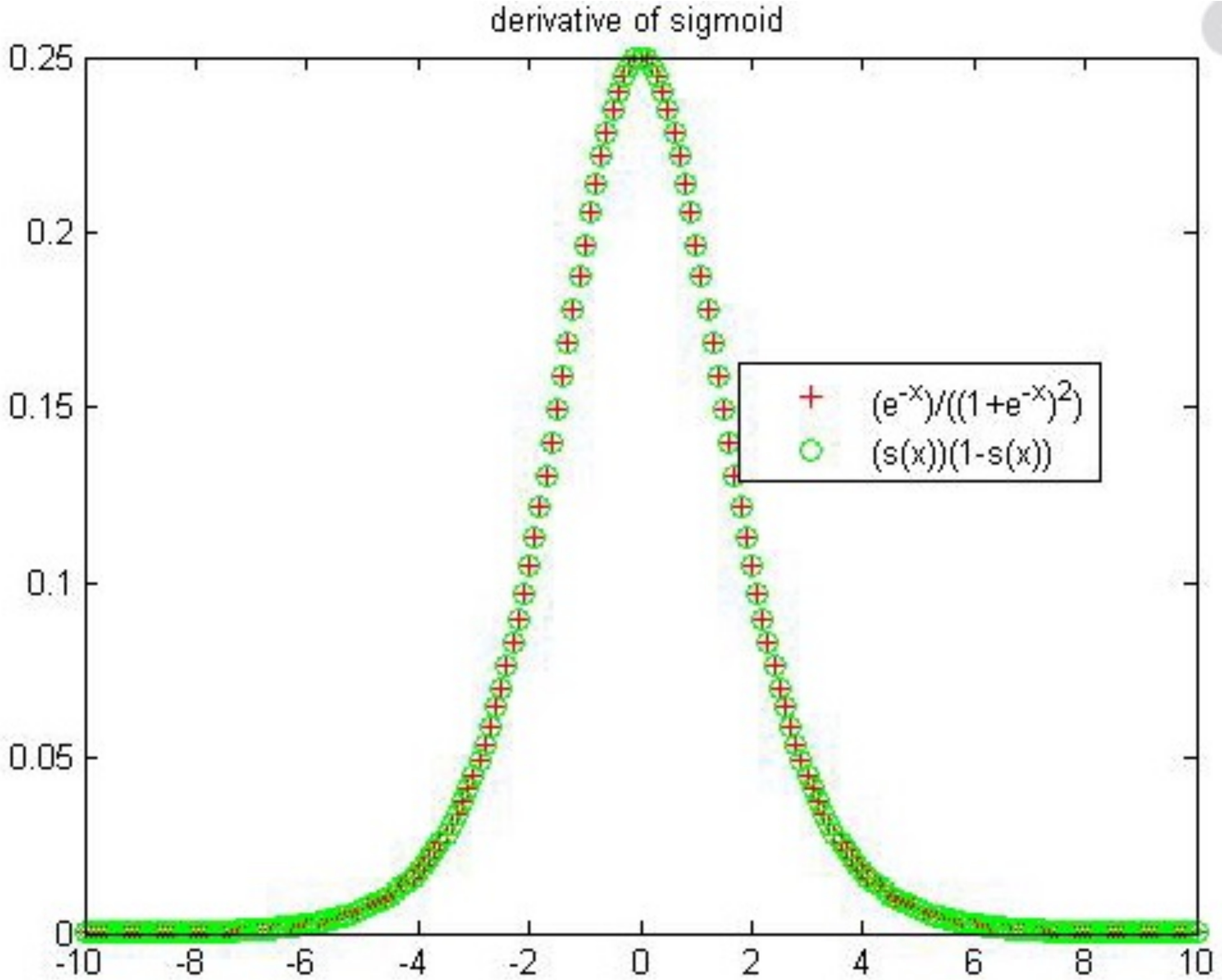
$$\frac{\partial E}{\partial w_i} = \sum_{j=1}^{n} \frac{\partial yj}{\partial wi}\frac{\partial E}{\partial yj} = \sum_{j=1}^{n} x_{ij}y_j(1-yj)(tj-yj)$$

In which, parameters index is *i* and training index is *j*.

- w_i is the weight of parameter *i*
- x_{ij} is the input parameter *i* associated to training point *j*.
- y_j is the output associated to training point *j*
- t_j is the expected output associated to training *j*

Sigmoid Derivative





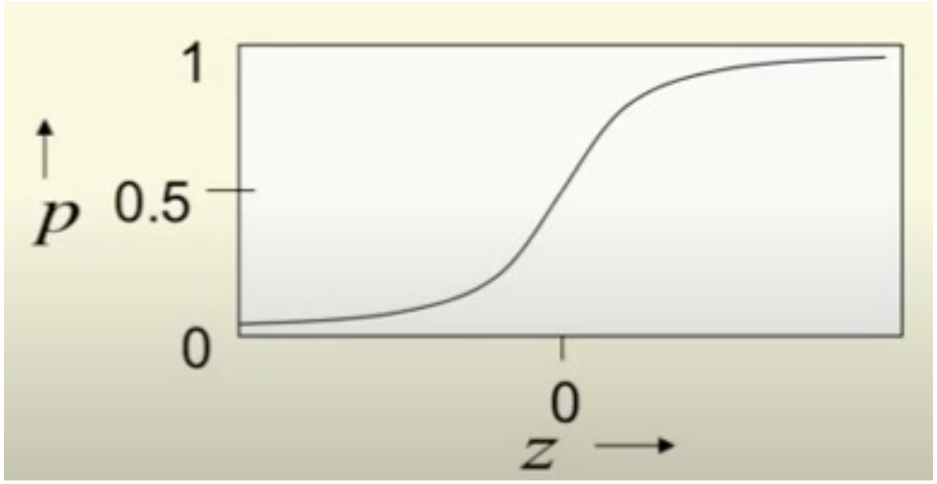
Reference: Kawahara.ca

Stochastic Binary Neurons

- Use sigmoid function as a probability function to generate 0/1 output

$$z = w_0 + \sum_i w_i x_i$$

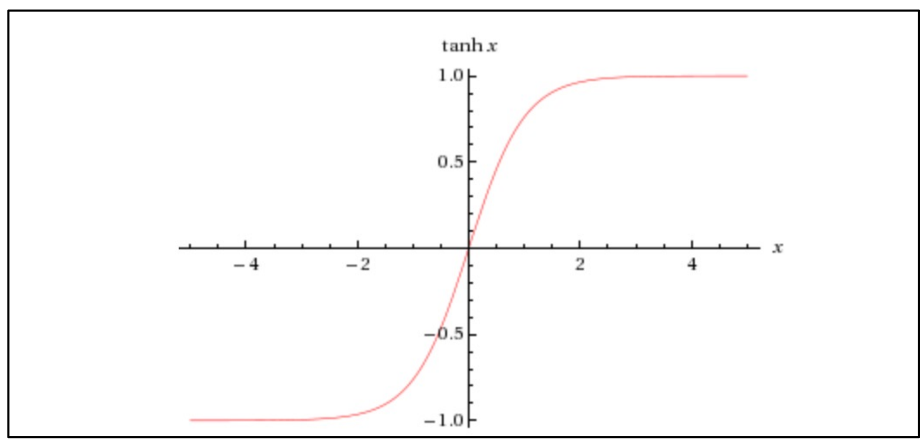
$$p(y=1) = \frac{1}{1+e^{-z}}$$



Tanh - Activation Function

$$Tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

- Makes the output vary around zero at the beginning



ReLU Stochastic Neurons

- The ReLU output is used as the rate of generating spikes, but the spikes are generated randomly. So ReLU output is the average rate of spikes.
  - Therefore the spikes follow Poisson distribution with average of ReLU output
  - Or time difference between spikes follow exponential distribution

Leaky ReLU

- Why ReLU
  - It doesn't saturate
  - It is easy to compute
  - But it suffers from "dying" issue
- To solve dying issue use variations of ReLU, like "Leaky-ReLU"
  - The hyper-parameter *a* can be set to [0.005 , 0.2]

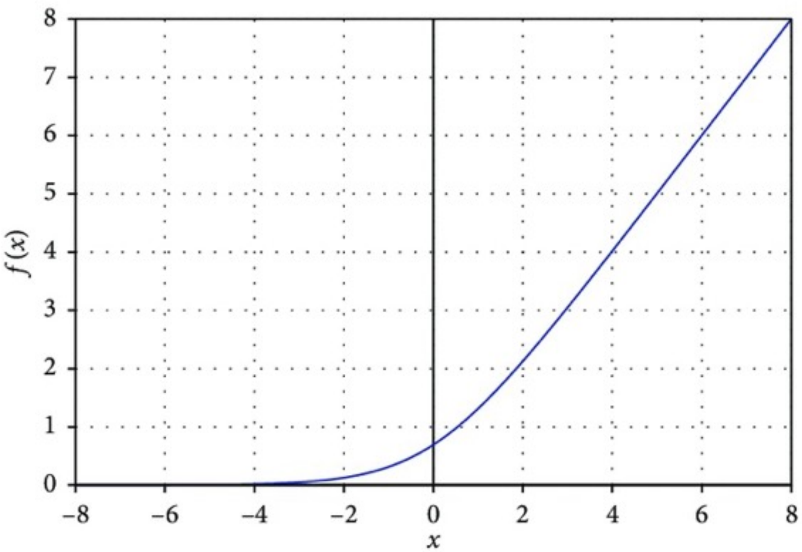
$$y = \max(az, z)$$

Leaky ReLU Options

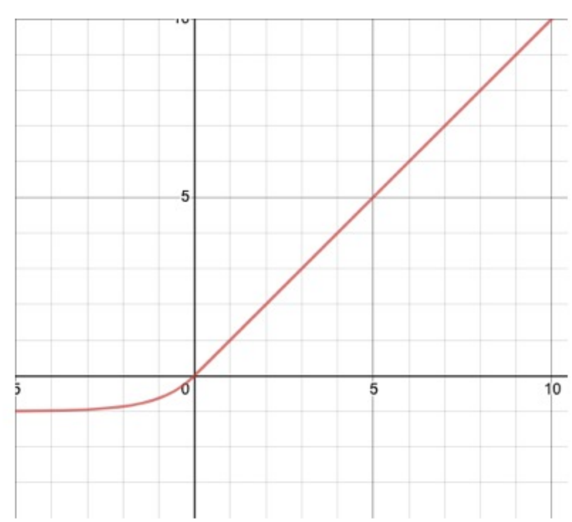
- Option: Randomized Leaky-ReLU
- Option: Parametric Leaky-ReLU
- Softplus: $\ln(1 + e^{x})$
- Option: Exponential LU (ELU)

$$y = \begin{cases} z, \text{ if } z \geq 0 \\ a(e^{z} - 1), \text{ if } z < 0 \end{cases}$$

Softplus



ELU

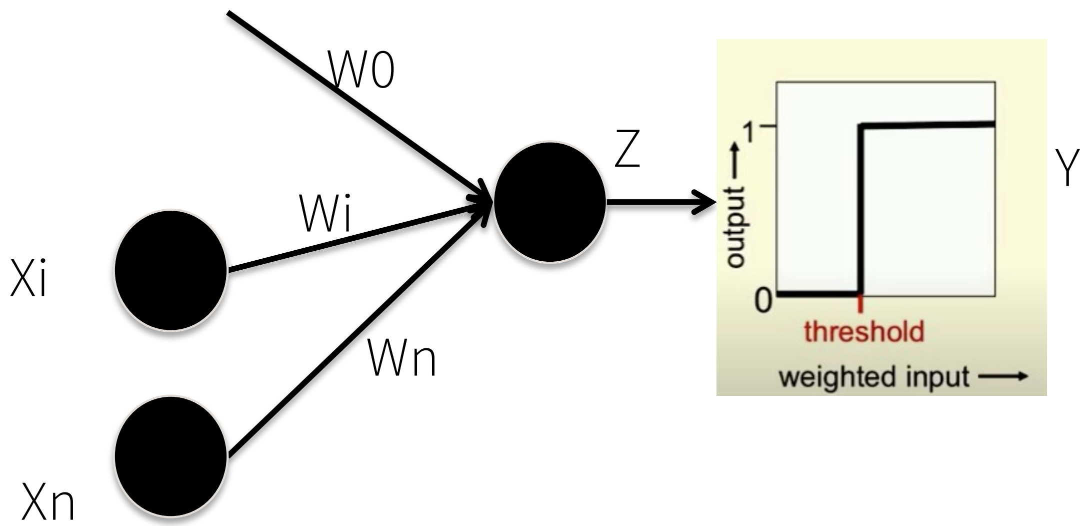


Perceptron or Binary Threshold Neurons

- Linear model
- Identical to Logistic Regression without class probability

$$z = w_0 + \sum_i w_i x_i$$

$$y = \begin{cases} 1 \text{ if } z \geq 0 \\ 0 \text{ Otherwise} \end{cases}$$



NN Architecture

- Dense layer or fully connected layer
- Bias Neuron
- Linear Algebra and NN

  Perceptron output using linear algebra

  $$Y = f(XW + b), f: activation\ function$$

  Perceptron learning

  $$W = W + \varepsilon\left(\hat{Y} - Y\right)X \ \ or$$

  $$w_i(t+1) = w_i(t) + \sum_{j \in Training\ Set} \varepsilon_i(\widehat{y_{ij}} - yij)x_{ij}$$

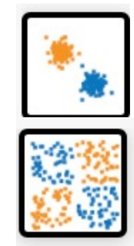
- Multi-layer perceptron
  - Pros: non-linear model
  - Cons: didn't know how to train until 1986 with backpropagation
  - Changing step function with sigmoid or Tanh functions
  - Initialize weights randomly. Otherwise, no variation in the network

TensorFlow Playground

- https://playground.tensorflow.org/

Playground

1. Choose activation function "linear"
    1. Try to learn classifying two simple separate classes
    2. Try to learn classifying X-OR model – change anything that feel appropriate
    3. Try to learn other input models



2. Set activation function to ReLu
    1. Set to one hidden layer with two neurons and try to learn X-OR model
    2. Change to Sigmoid activation and Tanh and try again
3. Set activation to ReLU with one hidden layer and 6 neurons

    Try to trains the model with following learning rates

    1. Set to 10
    2. Set to 3
    3. Set to 1
    4. Set to 0.001
4. Try item 3 with different activation function and learning rate 0.03