Introduction to Machine Learning (ML)

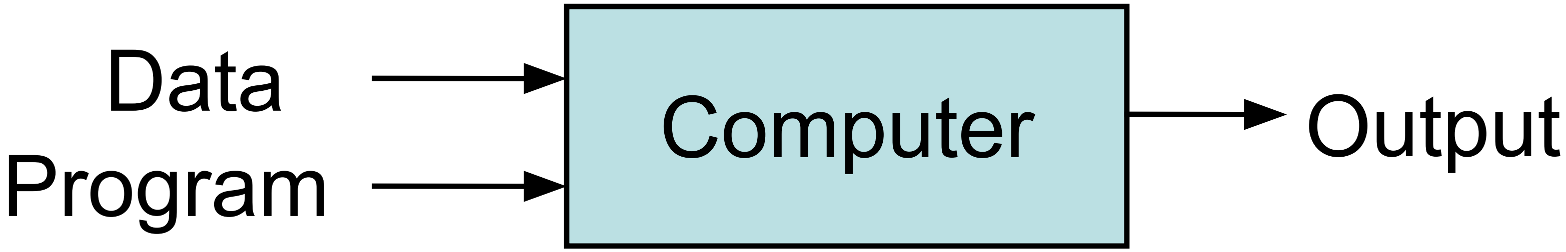
Focus: (Artificial) Neural Networks (ANNs)

What Is Machine Learning?

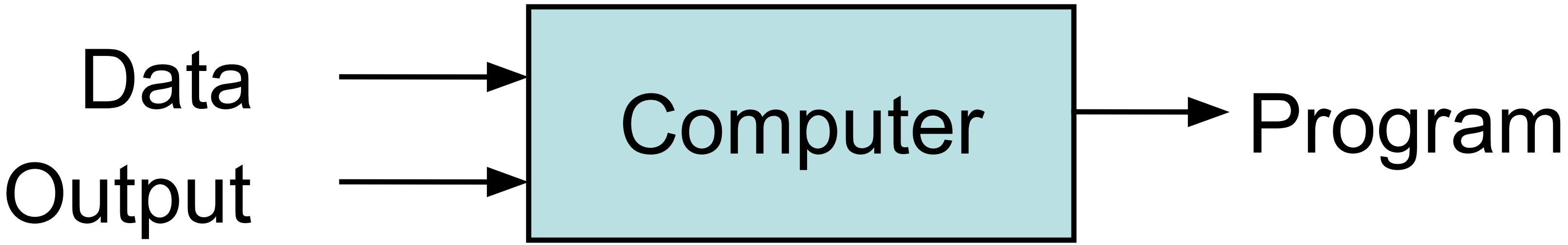
A technique that gives machines the ability to learn, without any explicit programming.

In simpler terms, a machine should be able to see some data, and learn to make decisions based on what it has seen.

Traditional Programming



Machine Learning



Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks/Fake News Detection
- Image/Speech/Text processing
- And more

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has these parts:
  - **Input**
  - **Representation (Objective Function)**
  - **Error computation (Cost Function)**
  - **Optimization (Optimization Function, includes Objective Function update)**
  - **Evaluation**

Input

- Tell me some examples...

Representation/Objective Function

- A function that we want to optimize (minimize/maximize)
- E.g. a linear function predicting house prices

Error Computation/Cost/Loss Function

- Training data has several instances
- Each instance is represented by a number of features
- Each instance comes also with its outcome, e.g. the price of a house
- The Objective Function will compute/predict the price of the house given its features
- There will be some difference between the actual and predicted price: Cost!

Error Computation/Cost Function

- There are several Cost Functions
- E.g. Sum Squared Error
- You write down:

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: (Batch / Stochastic) Gradient descent
- Constrained optimization
  - E.g.: Linear programming

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

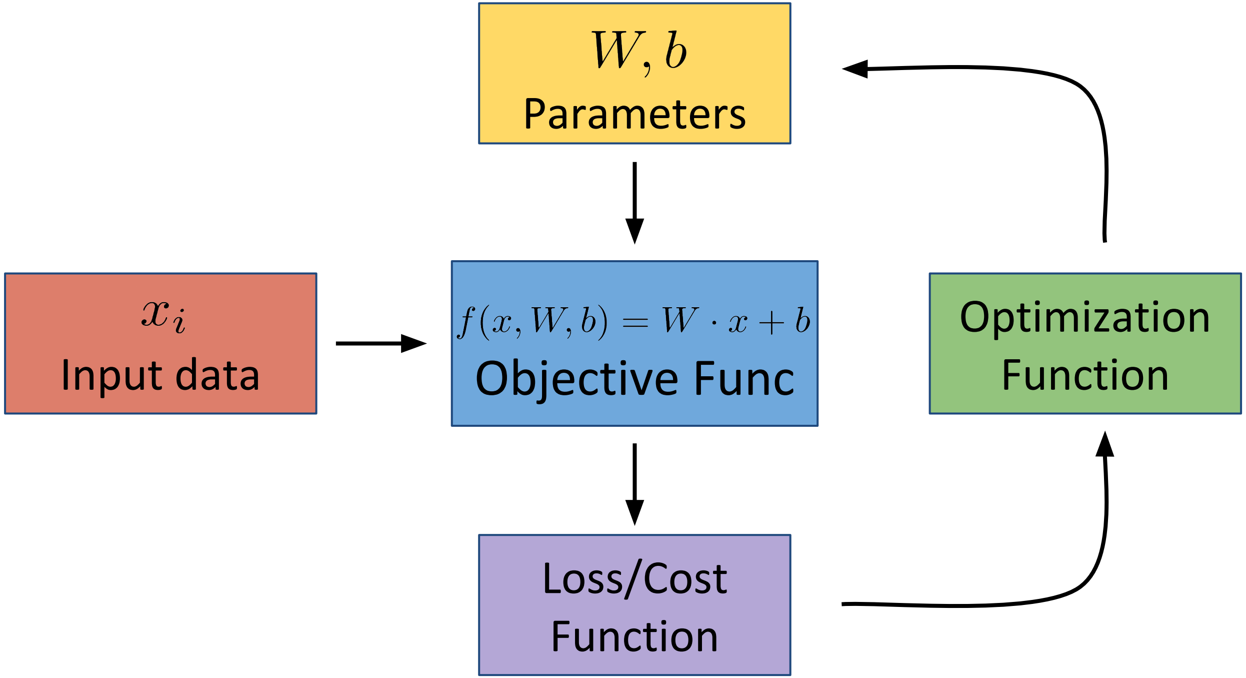
Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

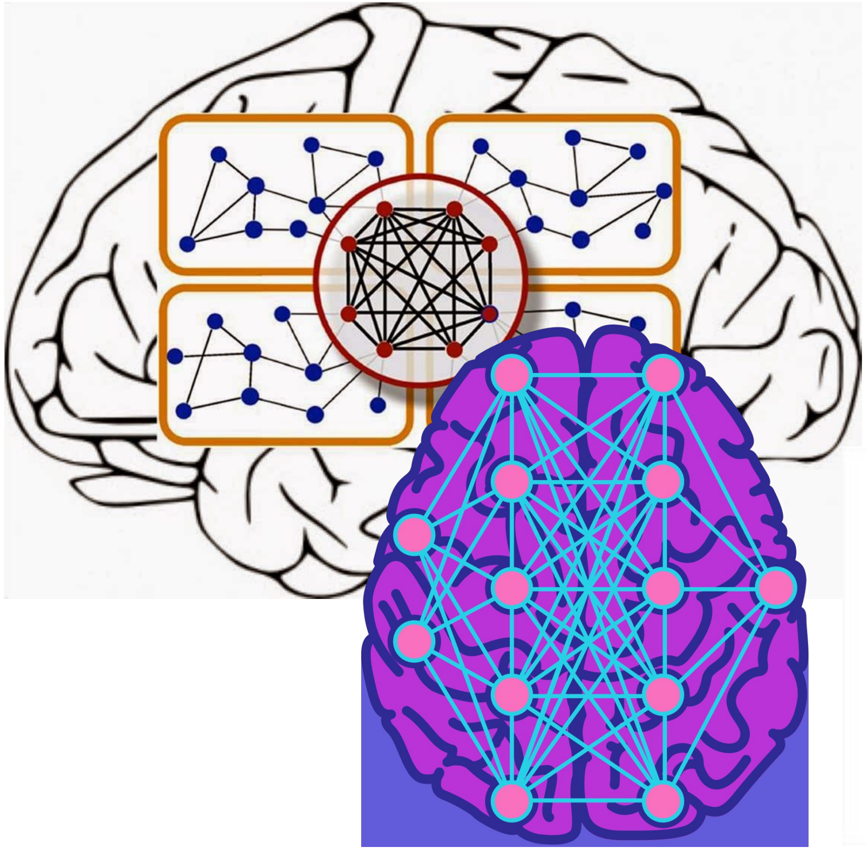
Supervised Learning

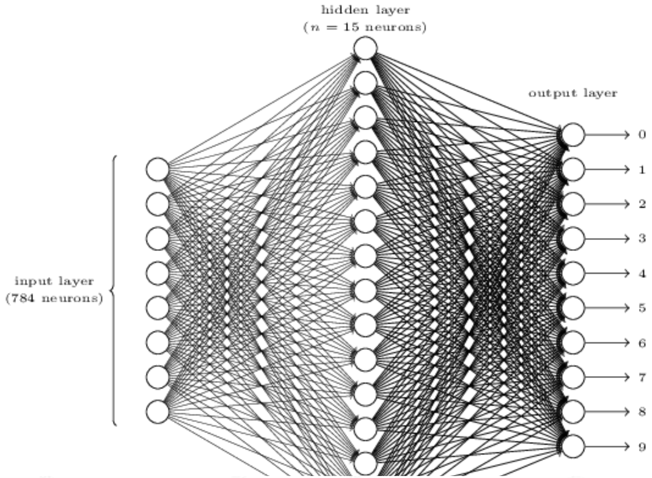
- **Given** examples of a function *(X, F(X))*
- **Predict** function *F(X)* for new examples *X*
  - Discrete *F(X)*: Classification
  - Continuous *F(X)*: Regression
  - *F(X)* = Probability(*X*): Probability estimation

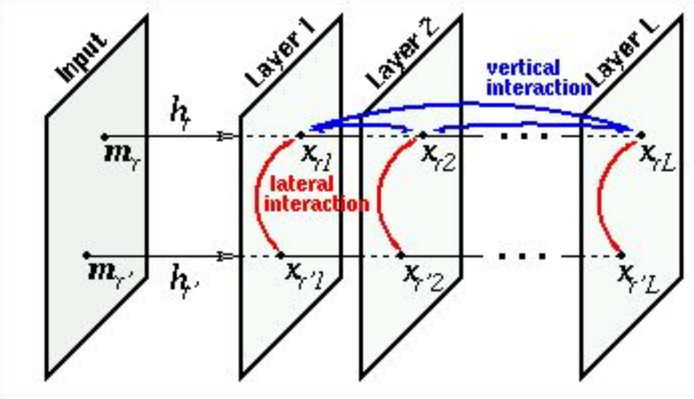
Cycle of Learning



Dive into Neural Networks





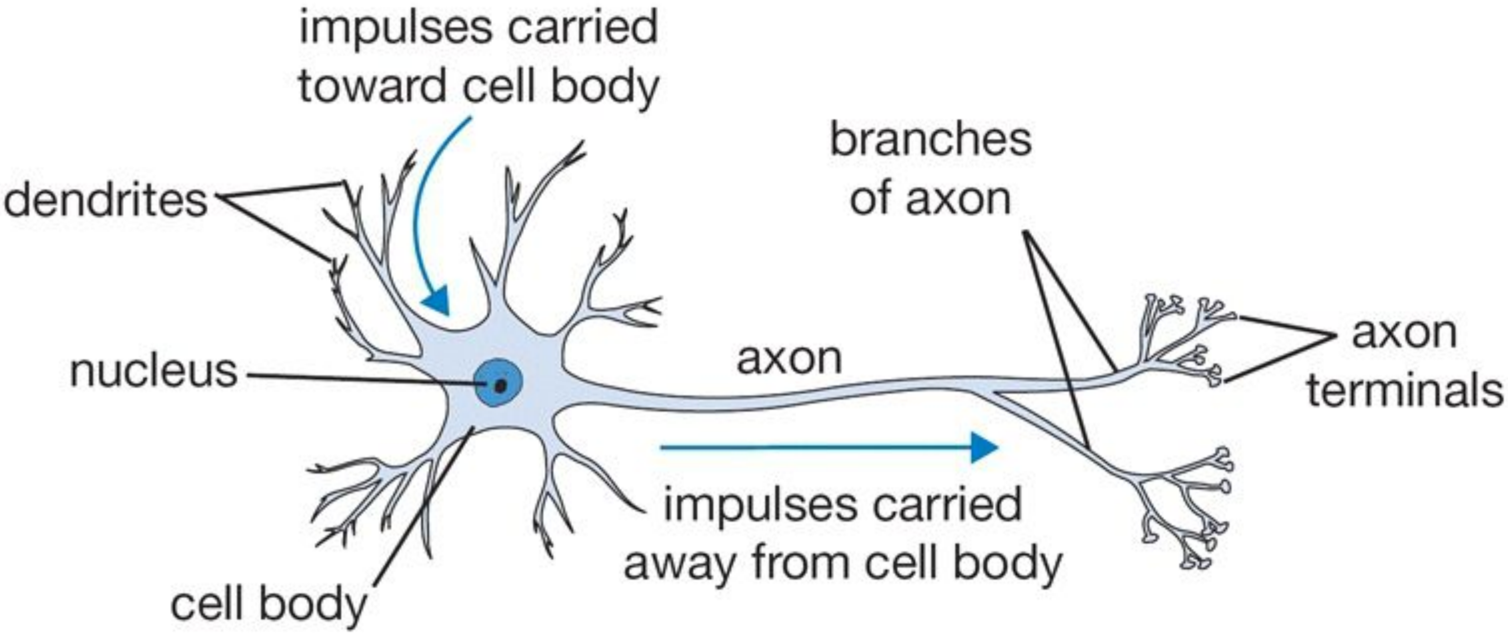


Neural Networks (NNs)

- **Neural network:** a complex structure that aims to process information and use the results to learn, like a human brain
- **Structure**:large number of highly interconnected processing elements (neurons) working together
- The more data (experience) they have the more they know (learn)

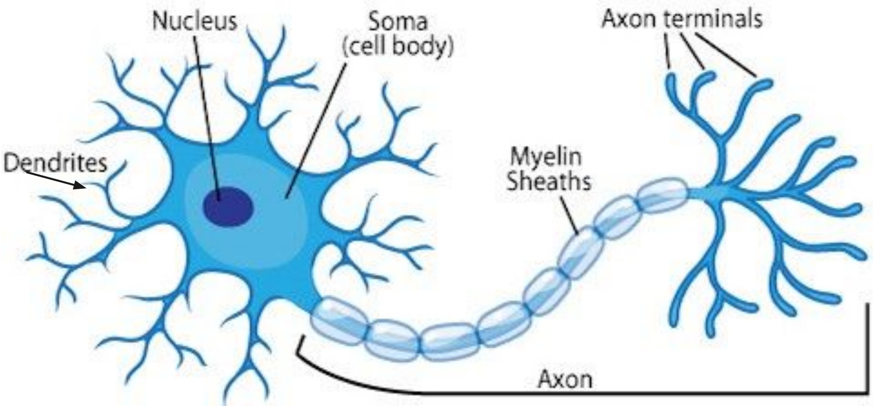
Biological Neural Network

Human Biological single Neuron



More on Biological Neuron

- **A biological neuron has three types of main components; <u>dendrites</u>, <u>soma</u> (or cell body) and <u>axon</u>.**
- **Dendrites receive signals from other neurons**
- **The soma, sums the incoming signals. When sufficient input is received, the cell fires; that is, it transmit a signal over its axon to other cells.**



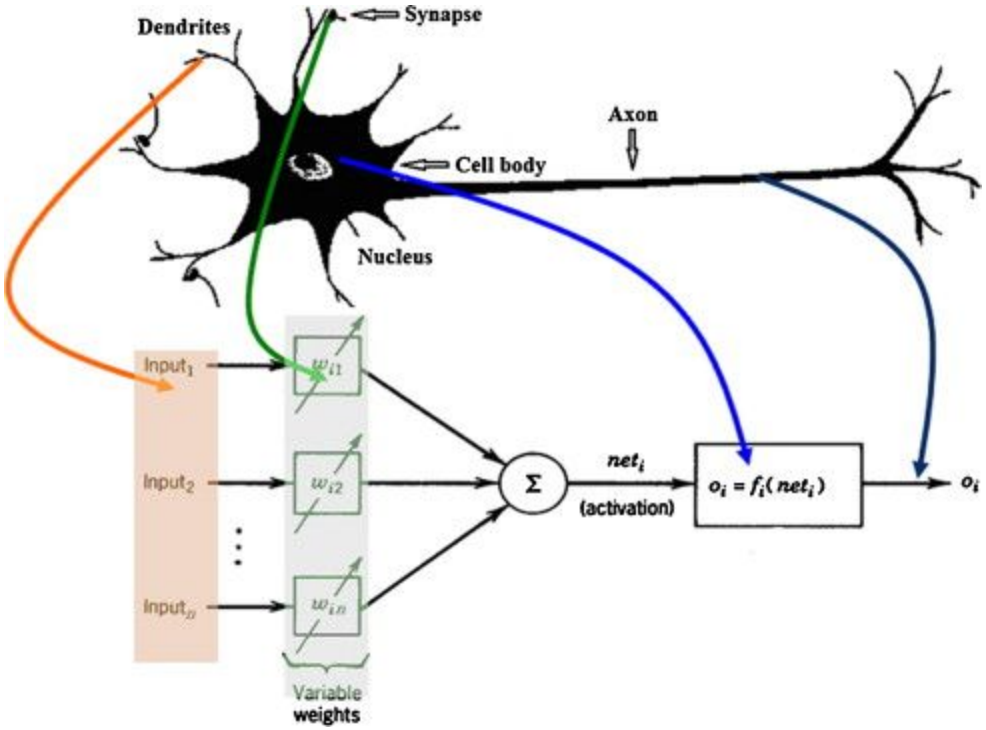
For Computers

Artificial Neural Network (ANNs)

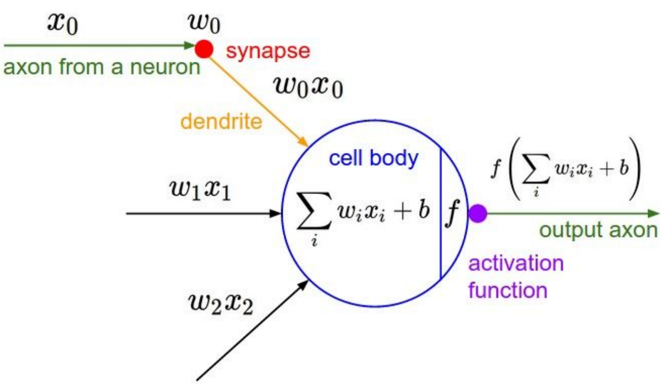
An artificial neural network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output.

Source: https://www.techopedia.com/definition/5967/artificial-neural-network-ann

From Biological to Artificial Neuron



More elaborate picture



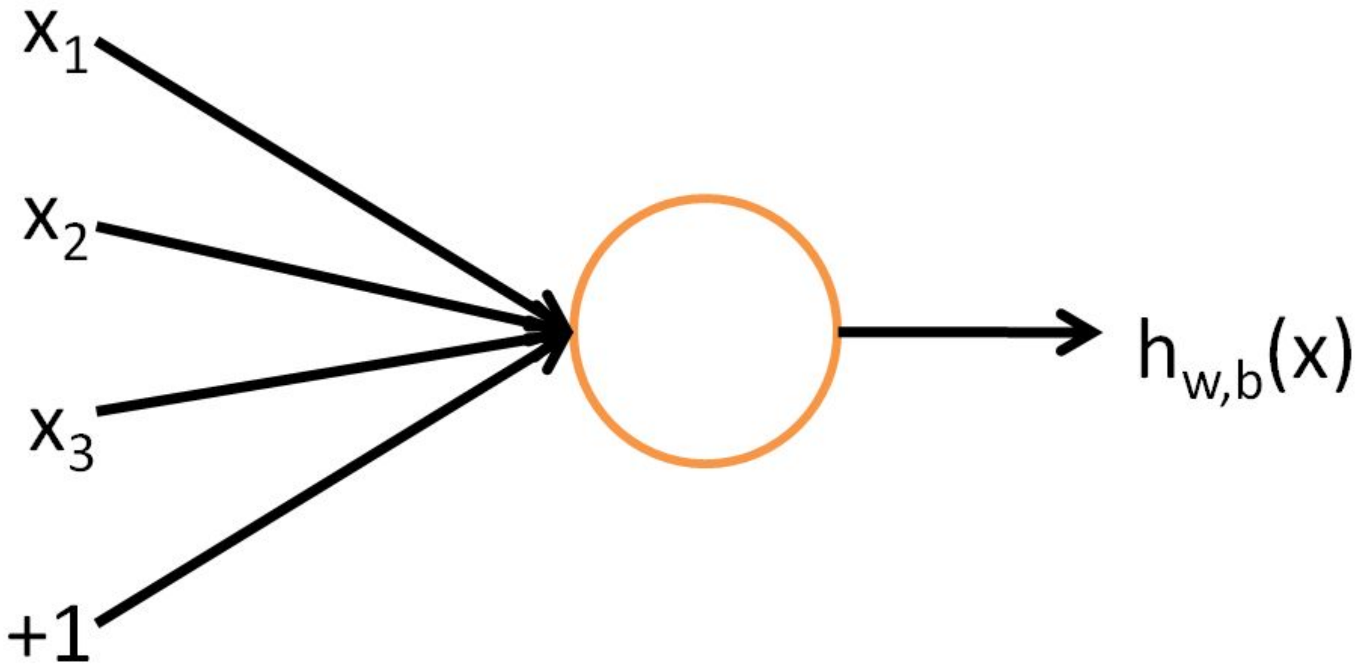
More on Artificial Neural Networks

ANNs have been developed as generalizations of mathematical models of neural biology, based on the assumptions that:

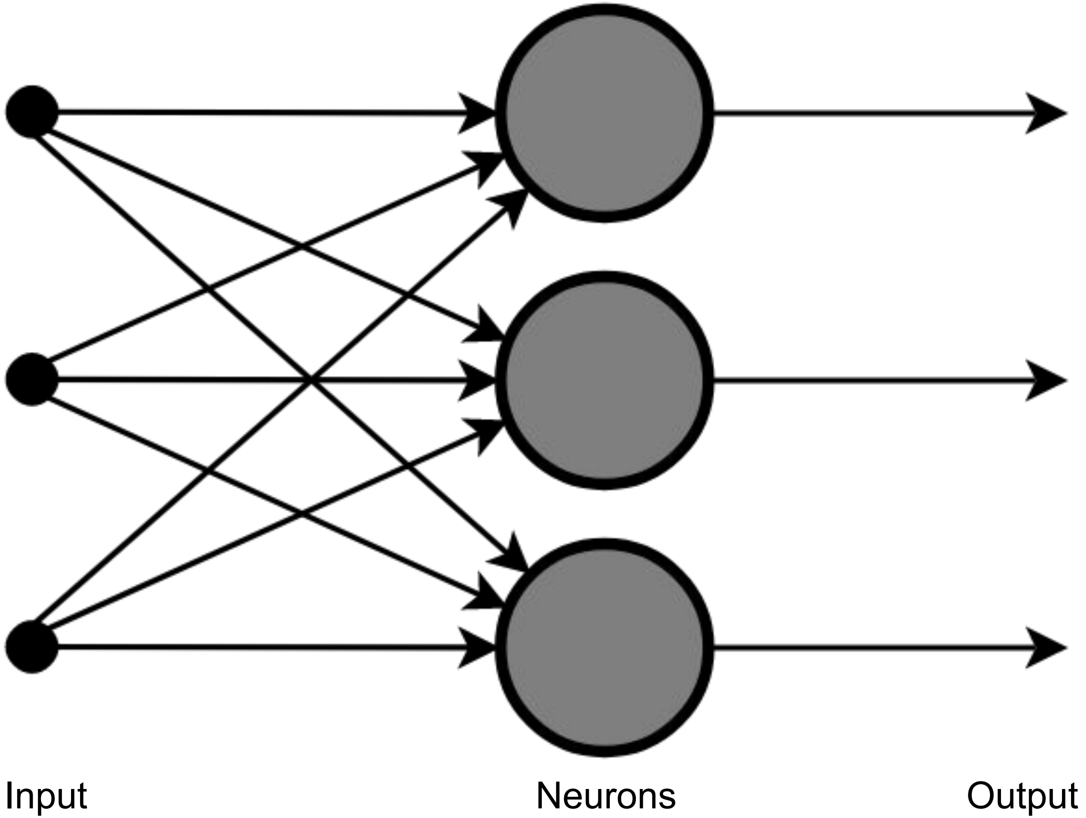
- **Information processing occurs at many simple elements called neurons.**
- **Signals are passed between neurons over connection links.**
- **Each connection link has an associated weight, which, in typical neural net, multiplies the signal transmitted.**
- **Each neuron applies an activation function to its net input to determine its output signal.**

Different Networks

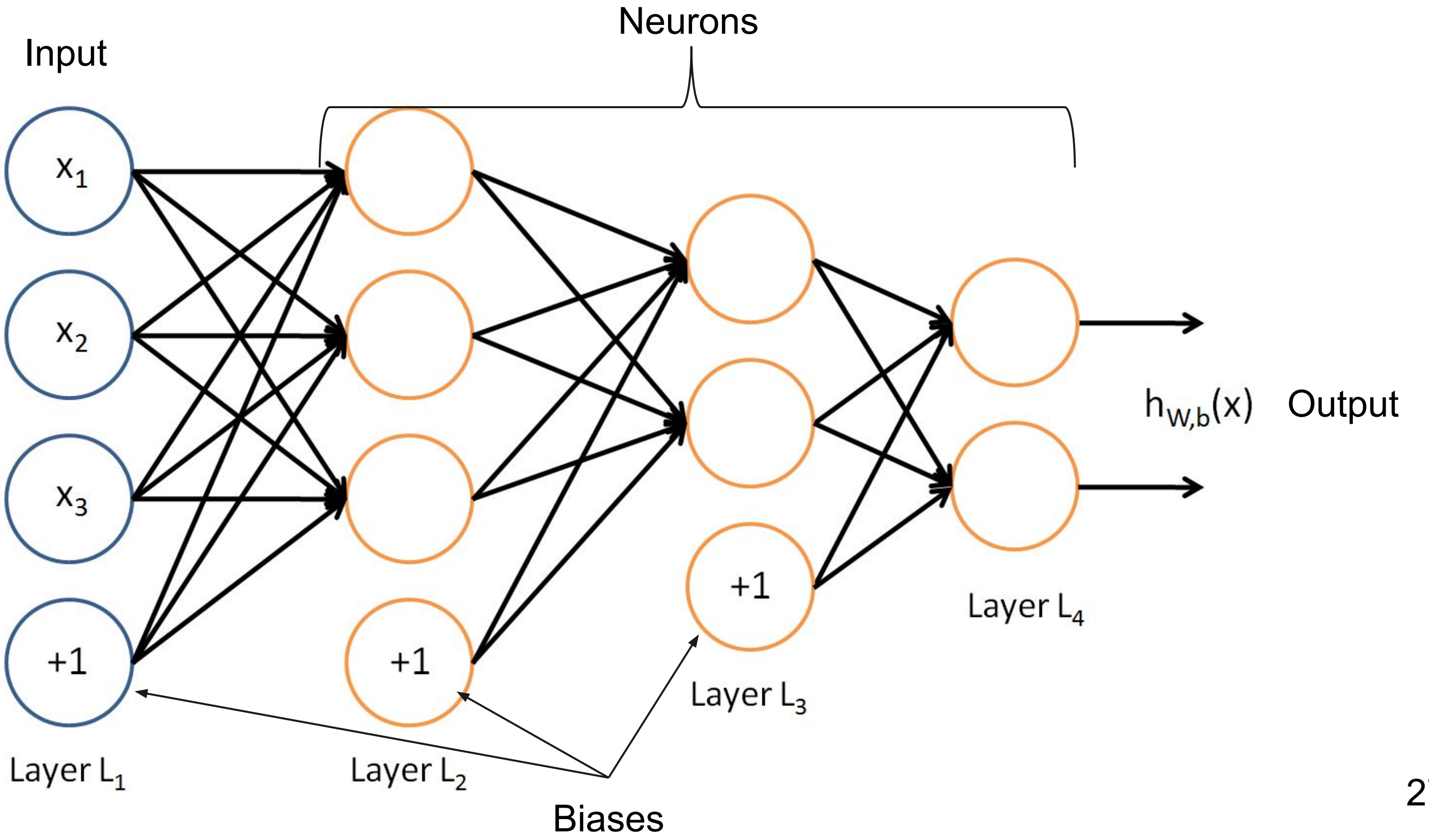
Single Neuron Network



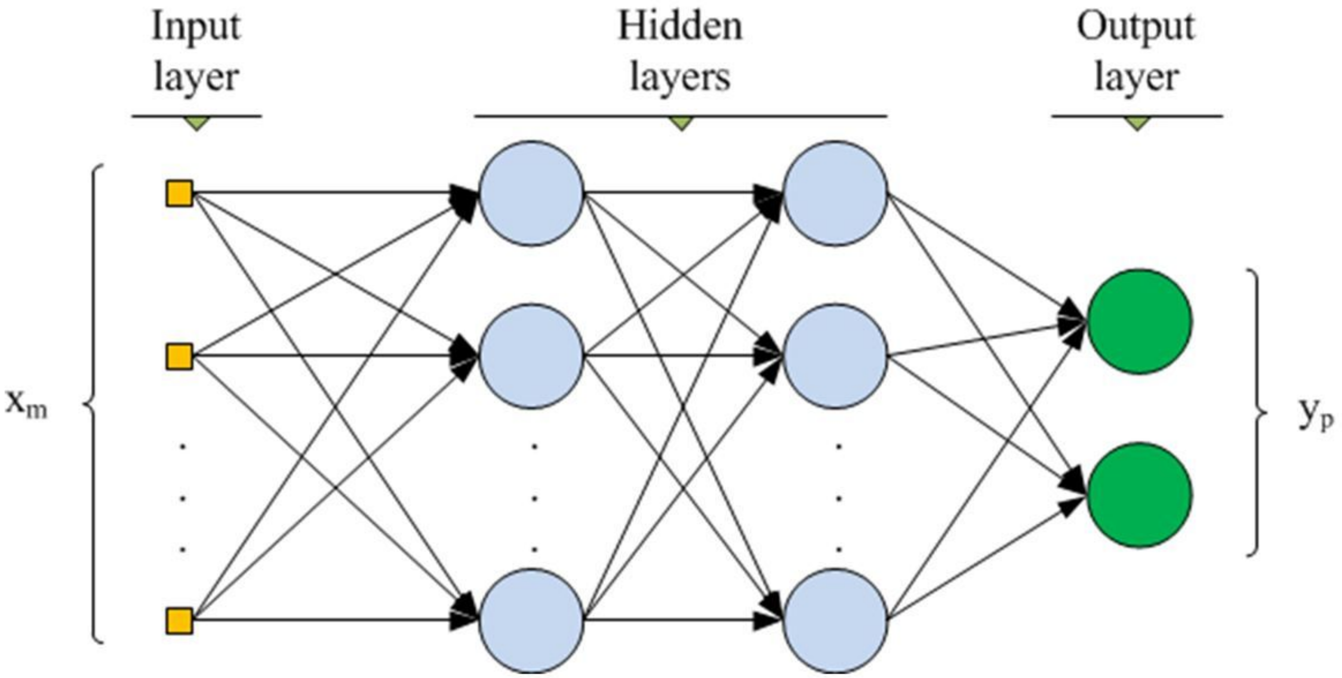
Single Layer Neural Network



Multi Layer Neural Network

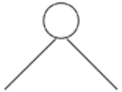
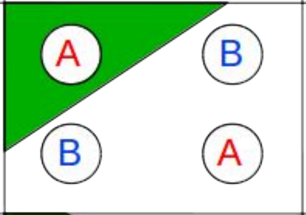
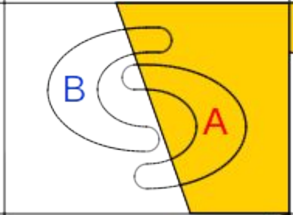
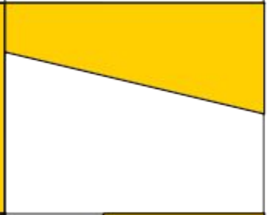
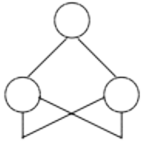
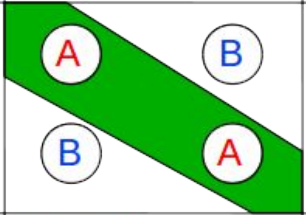
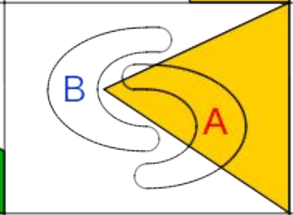
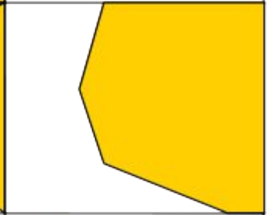
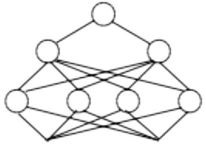
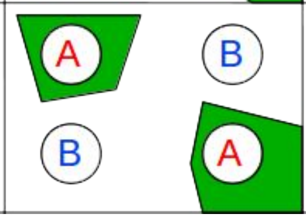
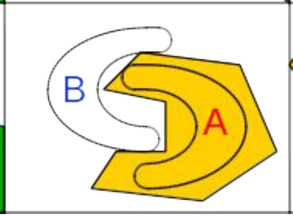
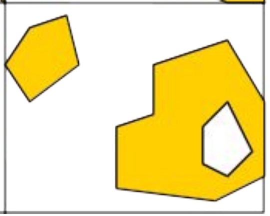


Terminology: Multi Layer Neural Network



Importance of Layers

The more Layer the more deeper Learning

Structure	Types of Decision Regions	Exclusive-OR Problem	Classes with Meshed regions	Most General Region Shapes
Single-Layer 	Half Plane Bounded By Hyperplane			
Two-Layer 	Convex Open Or Closed Regions			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

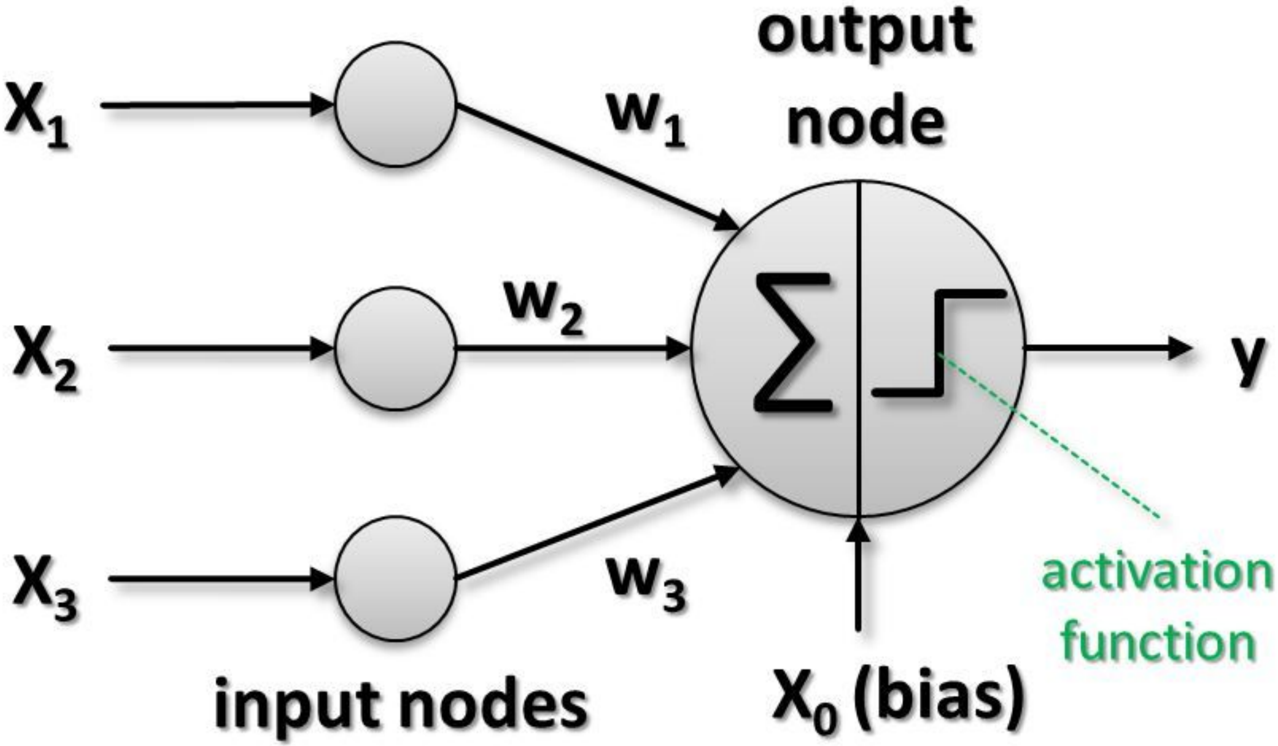
The more Layer the more deeper Learning

https://ai.google/education#?modal_active=none

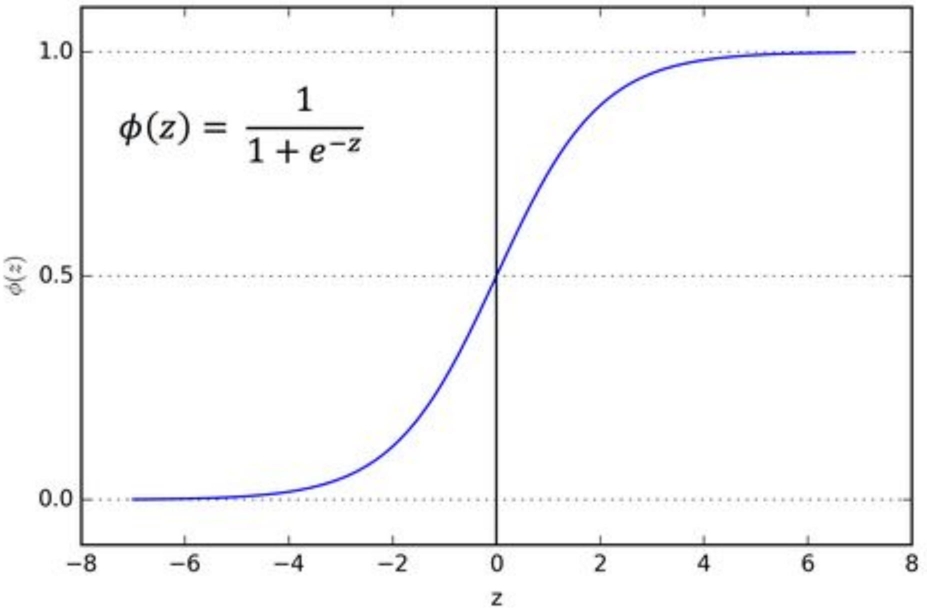
http://playground.tensorflow.org

Details

Back to single Neuron



Activation Function: Sigmoid

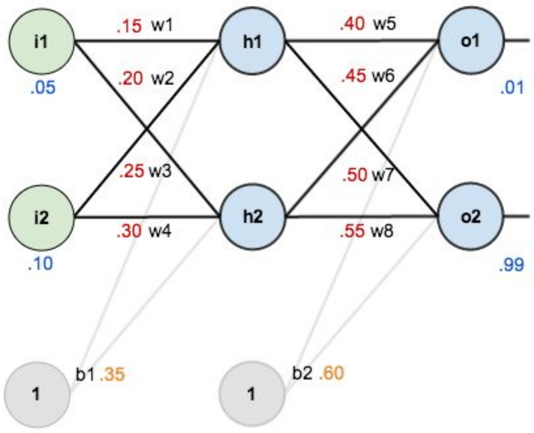


Activation Function: Sigmoid (example)

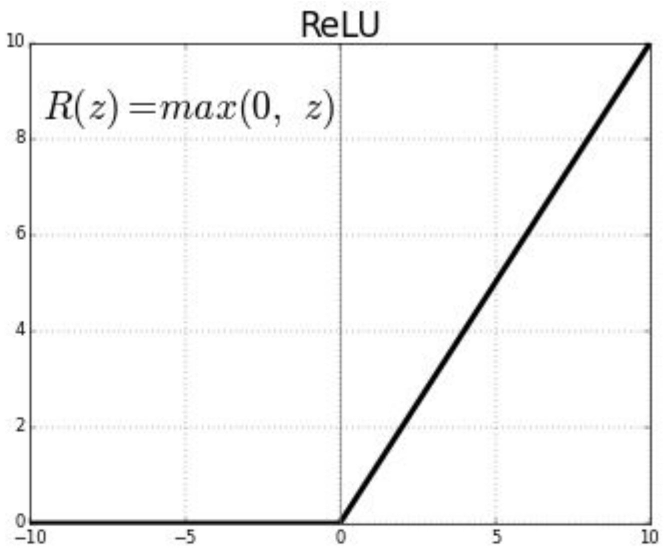
$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

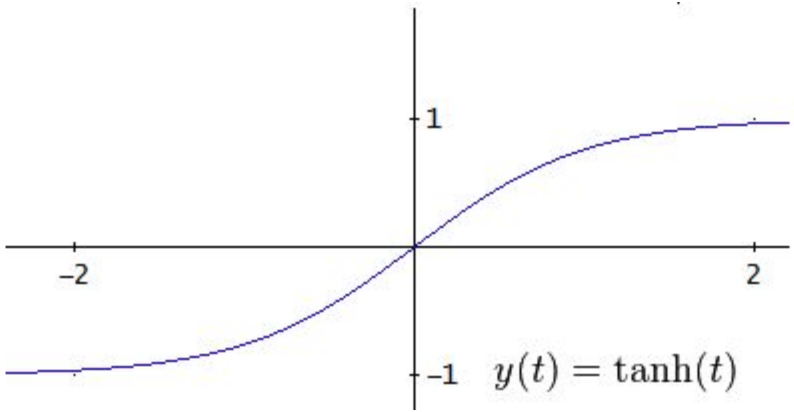
$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

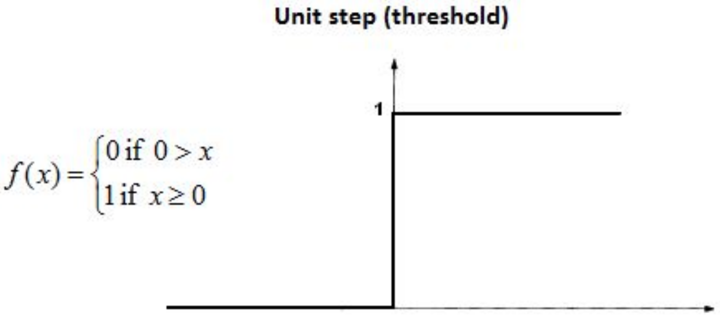
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$



More Activation Functions





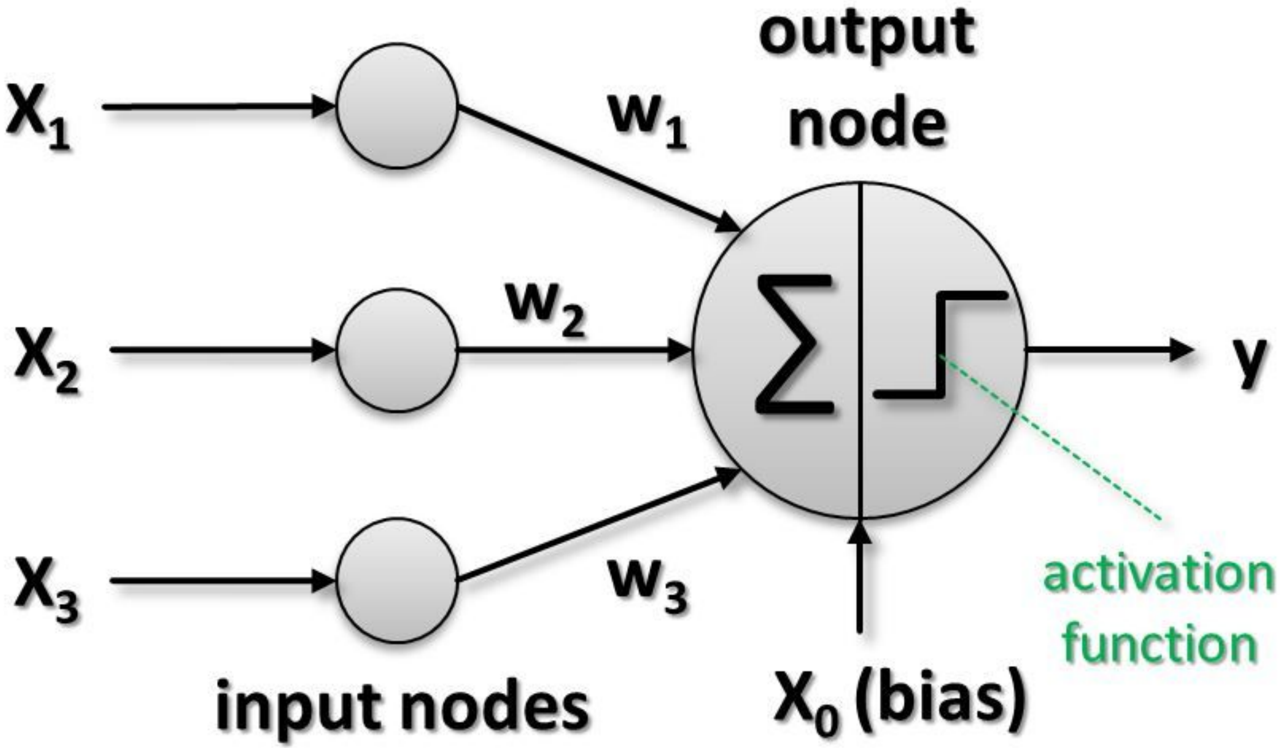


Input, Objective Function, Cost Function, Optimization Function, Evaluation

Input

- Similar what we discussed, but let us repeat

Objective Function



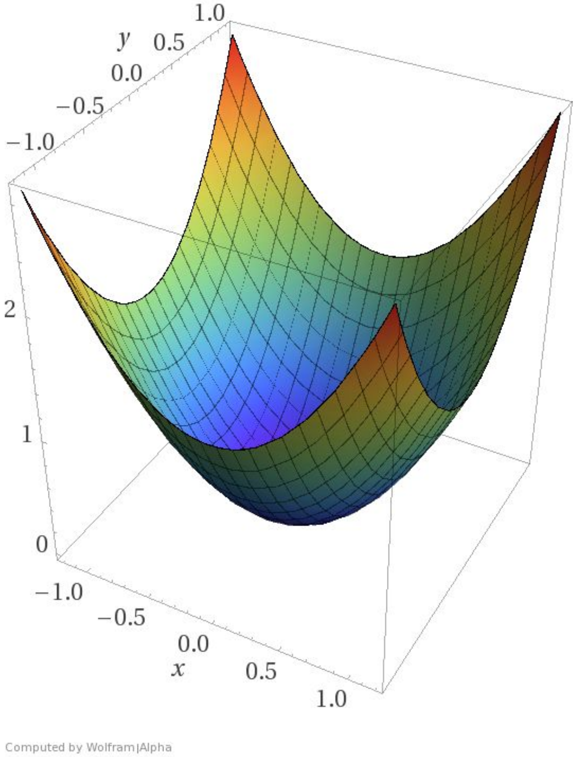
Cost Function

- E.g. Sum of squared difference between the actual and predicted outputs -- convex function
- Cross Entropy (ideal for classification)

Optimization Function

- E.g. Gradient Descent, assume a function is convex

Convex Function



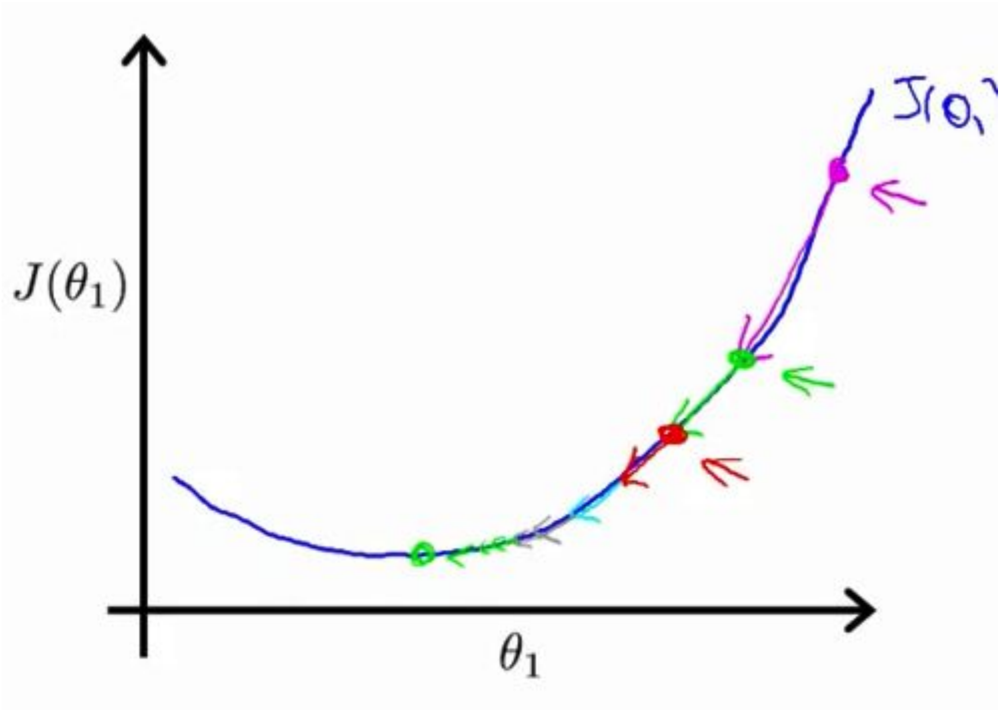
Using Gradient

- When a function f(x) is differentiable and convex, a necessary and sufficient condition for a point x^* to be optimal is $f'(x^*) = 0$

- Then minimizing f(x) is equivalent to finding solution for $f'(x^*) = 0$

- Because $\mathbf{-f'(x^*)}$ is the descent direction

Using Gradient



Gradient Descent

We have k parameters $w_1,\ldots,w_k$ we would like to train for a model (representation, objective function) -- with respect to an error/loss $J(w_1,\ldots,w_k)$ to be minimized

Gradient Descent:

1. Initialize $w_1,\ldots,w_k$ randomly
2. Update $w_1,\ldots,w_k$ to reduce $J(w_1,\ldots,w_k)$
3. $J'(w_1,\ldots,w_k)$ tells us in which direction $J(w_1,\ldots,w_k)$ increases the most
4. Use opposite direction of $J'(w_1,\ldots,w_k)$ to minimize $J(w_1,\ldots,w_k)$

Pseudo Code: Gradient Descent

Gradient Descent:

1. *Initialize* $w_1, \ldots, w_k$ *randomly*
2. *While (not converged) {*

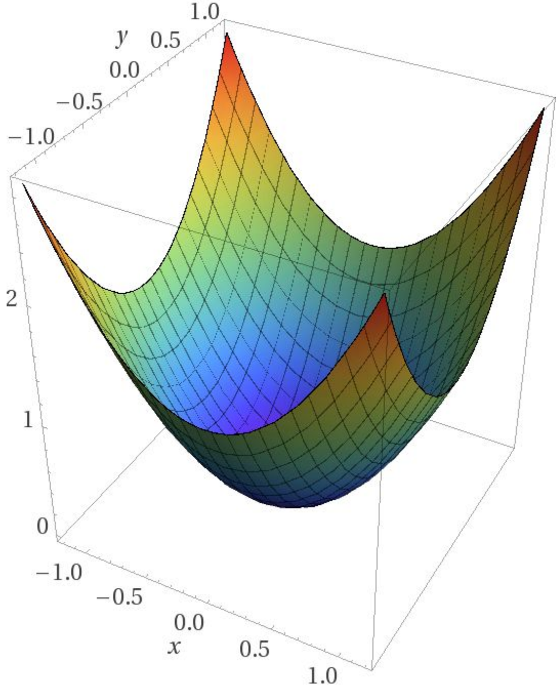
   $w_1^* = w_1 - \alpha * \Delta J / \Delta w_1$

   $\ldots$

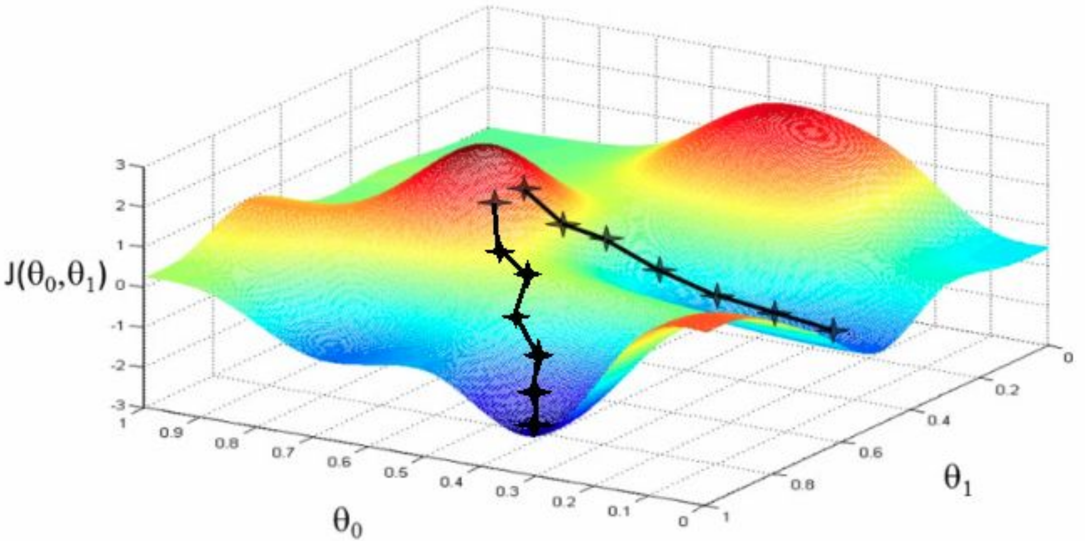
   $w_k^* = w_k - \alpha * \Delta J / \Delta w_k$
3. *}*

The character α is the learning rate or step size.

Issues



Computed by Wolfram|Alpha



Issues (con)

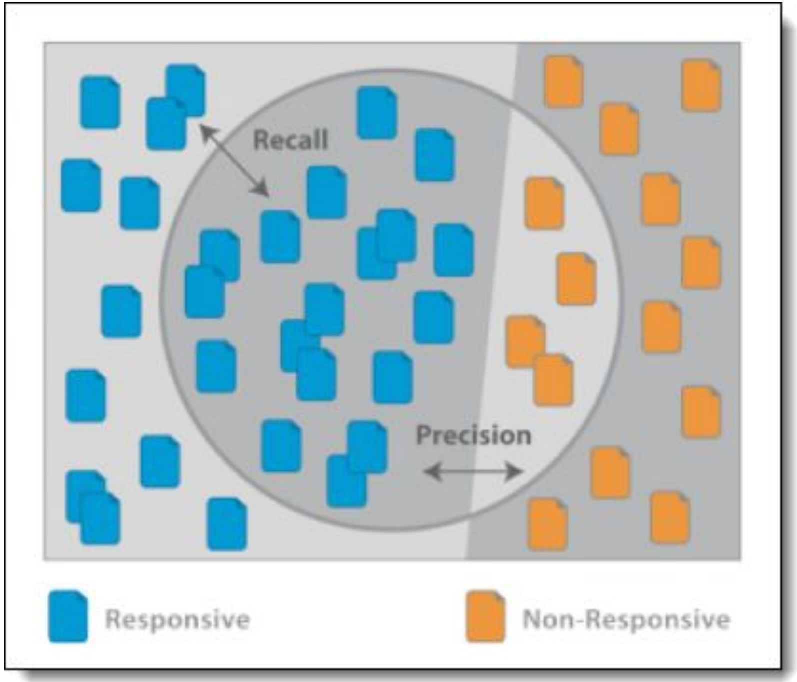
Convergence can be slow

- Larger learning rates/step size (α) can speed up -- but with too large α the optimum can be skipped or jumped over
- To small α will slow down the process
- To combine the gradient descent with line search to optimize the process (on every iteration the α value is determined using the line search)

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Precision & Recall



Precision & Recall + Accuracy

	Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

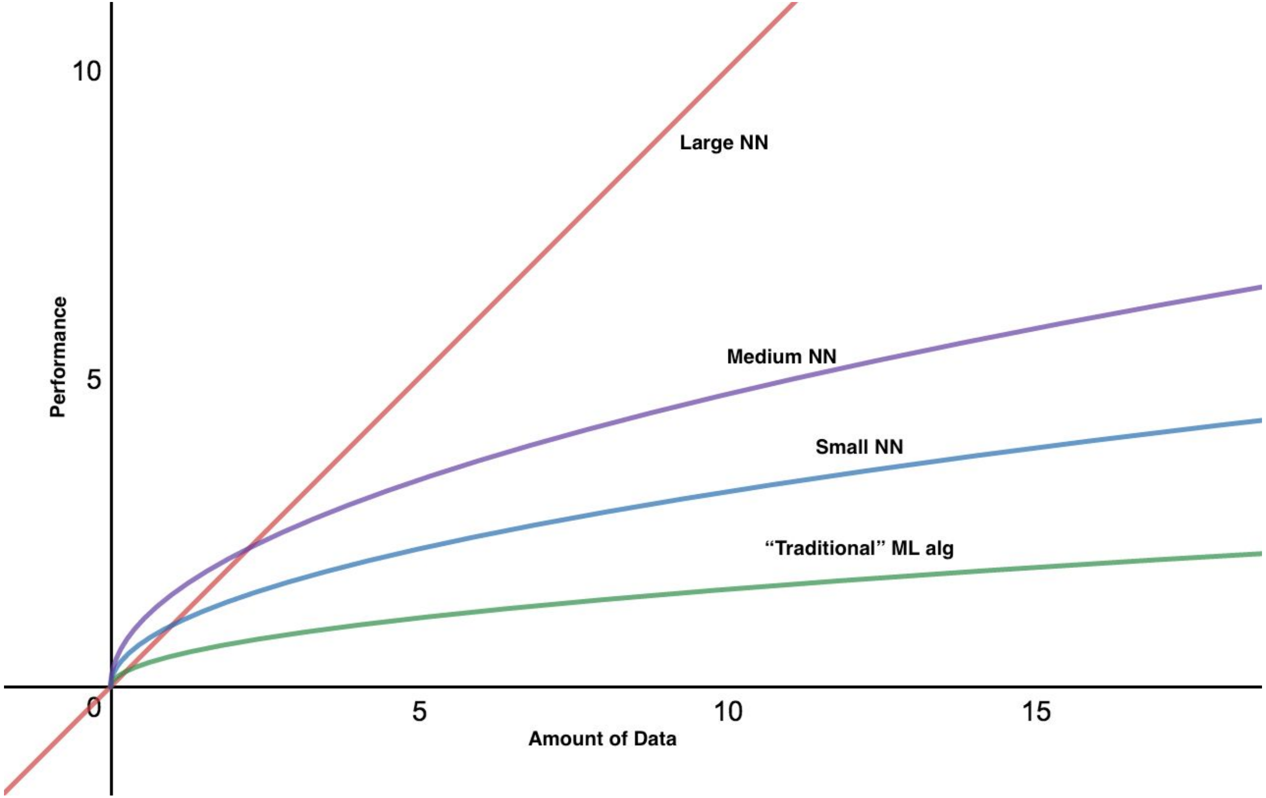
TP:= True Positive
FN:= False Negative
FP:= False Positive
TN:= True Negative

Is it possible to have high accurate results but very low F1?

P:= Precision = TP / (TP + FP)
R:= Recall = TP (TP + FN)
F1 = 2 * P*R / P + R
Accuracy = TP + TN / TP + TN + FP + FN

When to use ANNs?

When do Neural Networks shine?



Some Notes

Neural Networks are not new

- Have been there since 1950s
- But training multi-layer networks was an issue
- With (re)discovery of the backpropagation and efficient CPU/GPU the training is not an issue
- See for some history: https://en.wikipedia.org/wiki/AI_winter

Major companies with Deep Networks

- Facebook: (DeepFace for tagging images)
- Baidu: DeepSpeech for handeling voice queries in Mandarin
- Google: language translation service
- And many others...