Neural Networks I

Mark A. Austin

austin@umd.edu

ENCE 688P, Spring Semester 2021
University of Maryland

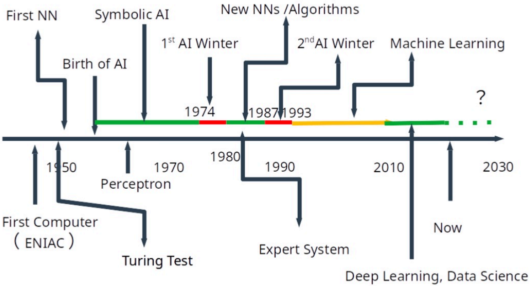
March 24, 2021

Overview

Part 01

Quick Review

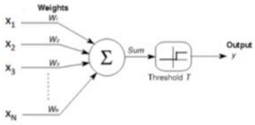
A Brief History



- 1943: First neural networks invented (McCulloch and Pitts)
- 1958-1969: Perceptrons (Rosenblatt, Minsky and Papert).
- 1980s-1990s: CNN, Back Propagation.
- 1990s-2010s: SVMs, decision trees and random forests.
- 2010s: Deep Neural Networks and deep learning.

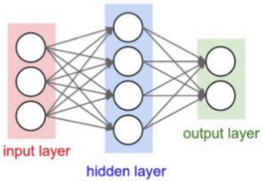
Machine Learning Capabilities (1980-1990)

Expressive Power of a Neural Network

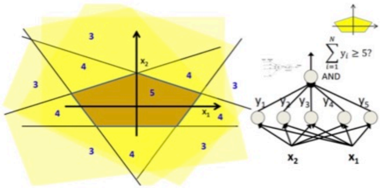


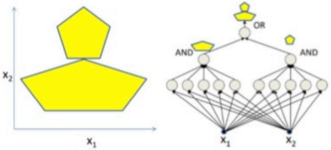
$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{d} w_i x_i \geq T \\ 0 & \text{else} \end{cases}$$

Neural Network with Single Hidden Layer



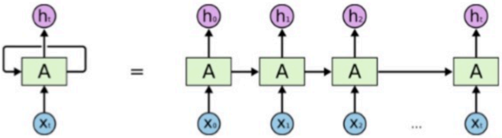
Approximation of Functions / Boolean Logic





Machine Learning Capabilities (1997-2014)

Recurrent Neural Networks (RNN): Learn sequences in data streams (text, speech)



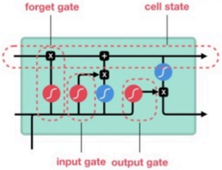
Hidden state "h" serves two purposes:

- Make an output prediction.
- Represent features in the previous steps ....

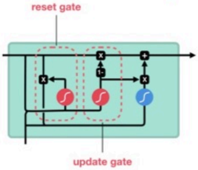
Key Features of LSTM:

- Standard RNN suffers from vanishing gradients for modeling of long-term dependencies.
- LSTM gives cells the ability to remember values for long periods of time.
- Gates regulate the flow of information in / out of the cell, and what should be remembered or discarded.

Long Short-Term Memory (1997)



Gated Recurrent Units (2014)

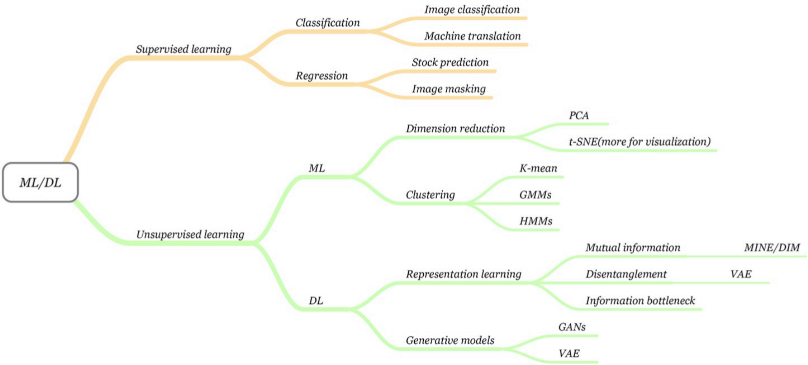


Applications:

- Time series prediction.
- Time-series anomaly detection.

Classification of Machine Learning Capabilities

Tree of Machine Learning and Deep Learning Capabilities

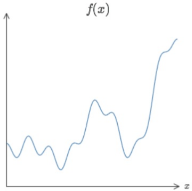


Introduction to Neural Networks

Why Neural Networks?

Reasons to use Neural Networks:

- Neural networks are universal function approximators, no matter how complex:



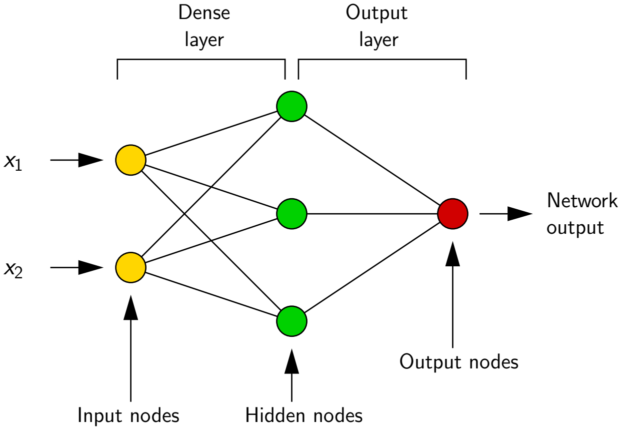
- Neural network architectures are highly scalable and flexible.

Caveat:

- Very large neural networks may be close to impossible to train and generalize correctly.

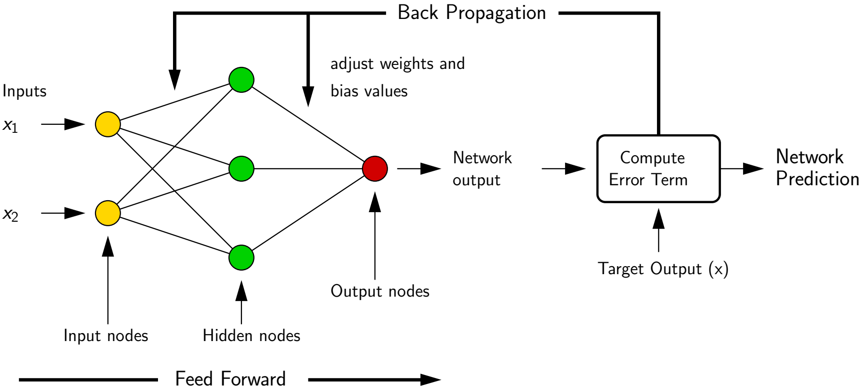
Basic Neural Network Architecture

Neural Network with One Hidden Layer:



Basic Neural Network Architecture

Training Procedure: Back Propogation



Modeling Expectations

Capabilities of a Perceptron Model: (From Lippman, 1987)

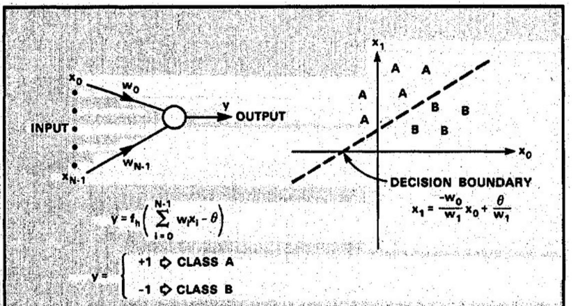


Figure 12. A single layer perceptron that classifies an analog input vector into two classes denoted A and B. This net divides the space spanned by the input into two regions separated by a hyperplane or a line in two dimensions as shown on the top right.

Modeling Expectations

Neural Networks with Hidden Layers: (From Lippman, 1987)

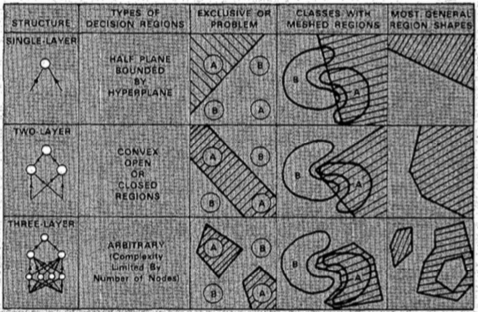


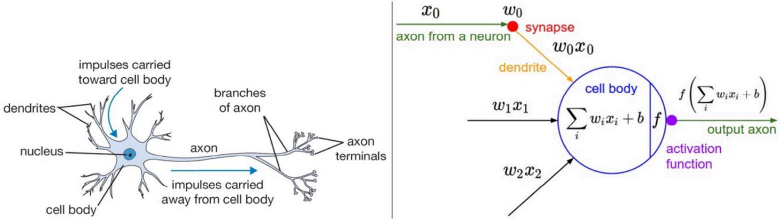
Figure 14. Types of decision regions that can be formed by single- and multi-layer perceptrons with one and two layers of hidden units and two inputs. Shading denotes decision regions for class A. Smooth closed contours bound input distributions for classes A and B. Nodes in all nets use hard limiting nonlinearities.

The Perceptron

Building Block of Machine Learning

A Little History / Biological Inspiration

Neural networks originally began as computational models of the brain (i.e., models of cognition).

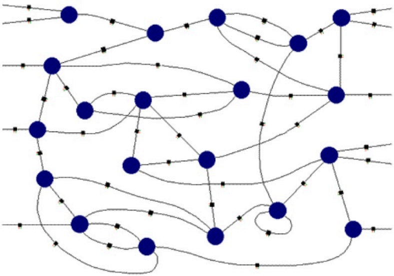


A cartoon drawing of a biological neuron (left) and its mathematical model (right).

- Early models were based on association relationship.
- More recent models of brain are connectionist – neurons connect to neurons.

Connectionist Models

Present-day neural network models are connectionist machines.

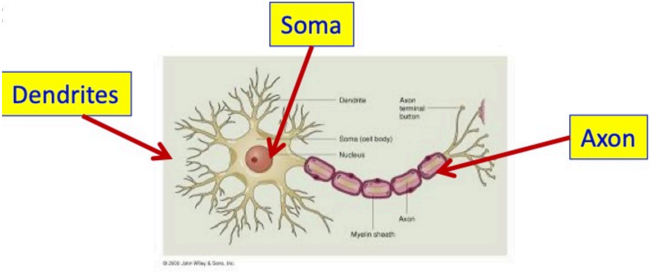


That is:

- Network of processing elements.
- Knowledge is stored in the connections between the elements.
- We need a model for these computational units.

Mathematical Model of a Single Neuron

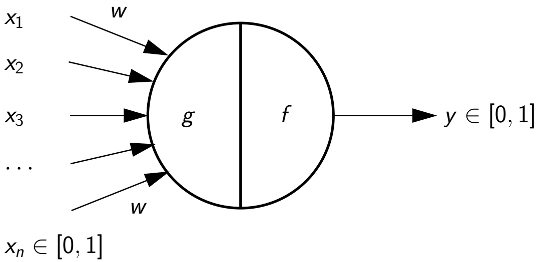
Modelling the Brain. Basic units are neurons:



- Signals come in through the dendrites into Soma.
- A signal exits via the axon to other neuron (only one axon per neuron).
- Neurons do not undergo cell division.

Mathematical Model of a Single Neuron

McCulloch and Pitts Model for a Single Neuron (1943):



First artificial neural network:

- Assumes boolean input (i.e., $x \in [0, 1]$).
- A neuron fires when its activation is 1, otherwise its activation is 0 (i.e, $y \in [0, 1]$).

Mathematical Model of a Single Neuron

Mathematical Model:

- All incoming connections have the same weight.
- Function $g()$ aggregates the inputs, i.e.,

$$g(x_1, x_2, \cdots, x_n) = g(x) = \sum_{i=1}^{n} x_i \quad (1)$$

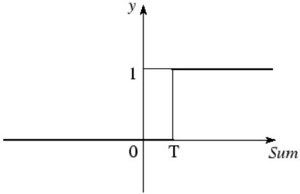
- Function $f()$ takes a decision based on this aggregation. y = 0 if any input x_i is inhibitory. Otherwise:

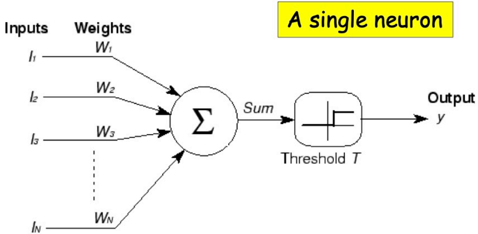
$$\begin{aligned} y = f(g(x)) &= 1 \text{ if } g(x) \geq \theta. \\ &= 0 \text{ if } g(x) < \theta. \end{aligned}$$

- θ is called the threshold parameter.

Mathematical Model of a Single Neuron

Behavior of a Simple Neuron Unit:



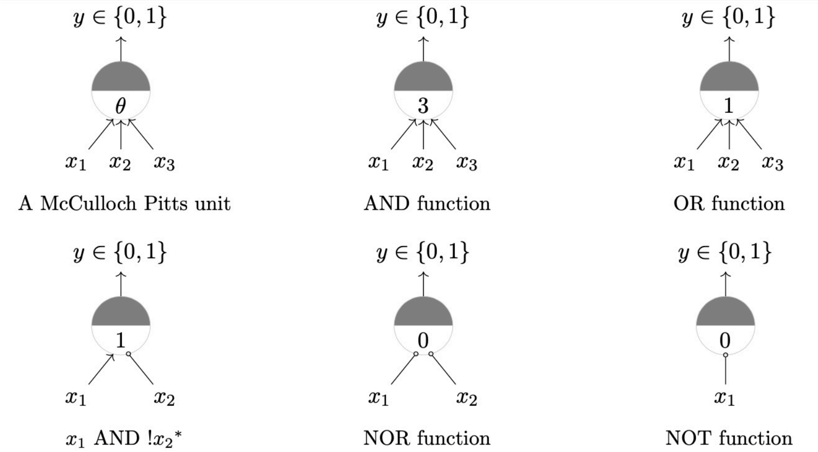


Criticisms:

- Claimed their machine could emulate a Turing machine.
- Did not provide a learning mechanism.

Mathematical Model of a Single Neuron

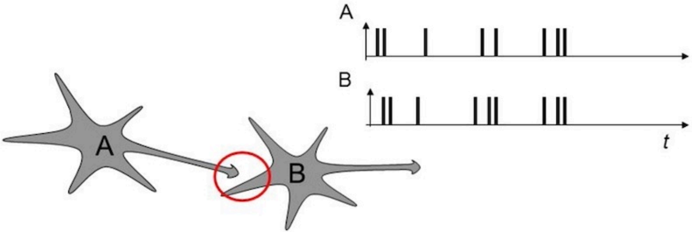
Simplified Modeling of Boolean Gates:



Mathematical Model of a Single Neuron

Hebbian Learning (Donald Hebb, 1949)

When an axon of cell A excites cell B and repeatedly or presistently takes part in firing it, some growth processes or metabolic change takes place in one or both cells so that A's efficiency is increased.



Observation: In other words, neurons that fire together wire together!

Mathematical Model of a Single Neuron

Principles of Hebbian Learning

- Neurons that fire together wire together!
- If neuron x_i repeatedly triggers neuron y, the synaptic knob connecting x_i to y gets larger.
- Mathematically, we can write:

$$w_i = w_i + \eta x_i y \tag{2}$$

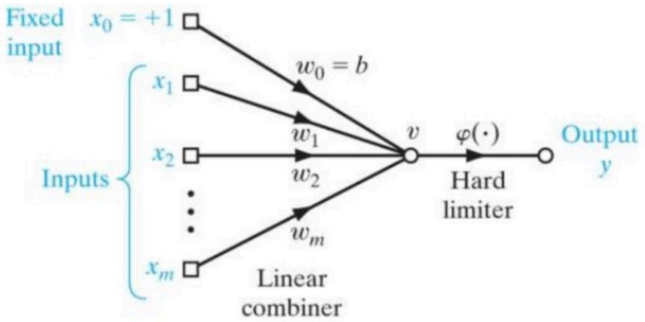
- Here, w_i is the weight of the i-th neuron's input to output neuron y.
- This simple formula is actually the basic of many learning algorithms in machie learning.

Mathematical Model of a Single Perceptron

Perceptron Model (Rosenblatt, 1958)

The simplest form of a neural network consists of a single neuron with adjustable synaptic weights and bias.

A nonlinear neuron consists of a linear combiner followed by a hard limiter.



Mathematical Model of a Single Perceptron

Perceptron Model (Rosenblatt, 1958):

- Learning algorithm:

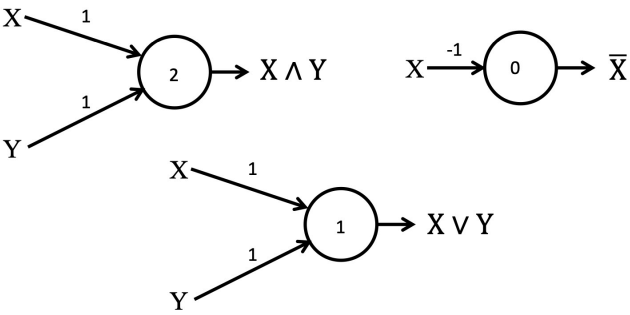
$$w(x) = w(x) + \eta\,(d(x) - y(x))\,x. \tag{3}$$

Here:

- η is the learning rate,
- $d(x)$ and $y(x)$ are the desired and actual outputs in response to x.
- Update weights whenever the perceptron output is wrong.
- Proved convergence.
- Solution for OR and AND Boolean Gates.

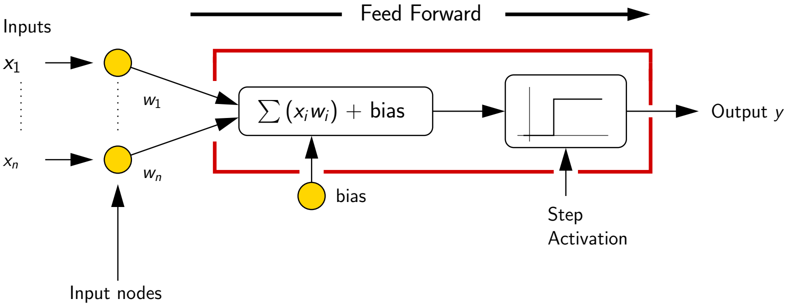
Mathematical Model of a Single Perceptron

Perceptron Model for OR and AND Boolean Gates



No solution for XOR Problem. Individual elements are weak. Networked elements are required.

The Perceptron Model: Forward Propagation



Here:

- Inputs $x_1, x_2, x_3, \cdots x_n$ are real valued.
- Weights $w_1, w_2, w_3, \cdots w_n$ are real valued.
- The output y can also be real valued.

The Perceptron Model: Forward Propagation

Step 1: Linear combiner:

$$z = g(x) = \sum_{i=1}^{n} w_i x_i + \text{bias}. \tag{4}$$

Step 2: Step activation:

$$y = f(z) = \begin{cases} 0, & z < \theta, \\ 1, & z \geq \theta. \end{cases} \tag{5}$$

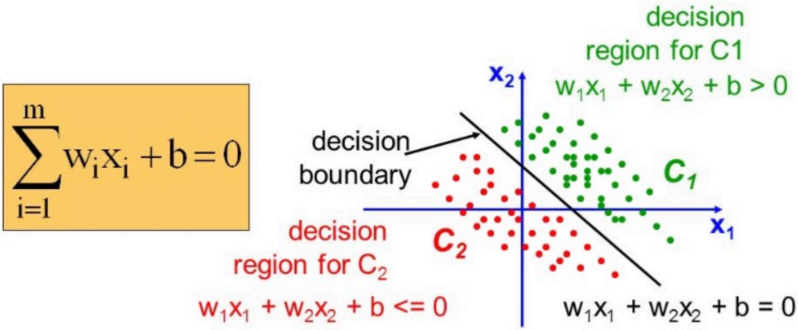
Here, θ is the threshold parameter.

Composition of steps 1 and 2:

$$y = f(g(x)) \tag{6}$$

Perceptron Model as a Linear Classifier

Perceptron operating on real-valued vectors is a linear classifier:



Addition of bias values expands modeling capability. No bias value $\rightarrow$ decision boundary constrained to pass through the origin.

References

- Lippmann R.P., An Introduction to Computing with Neural Nets, IEEE ASSP Magazine, April 1987.
- Bhiksha R., Introduction to Neural Networks, Lisbon Machine Learning School, June, 2018.