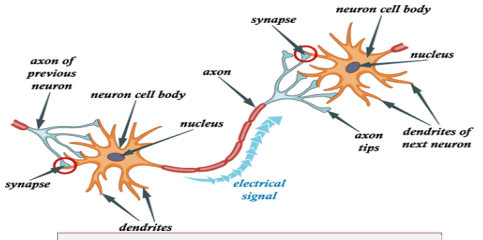
Introduction to Deep Learning

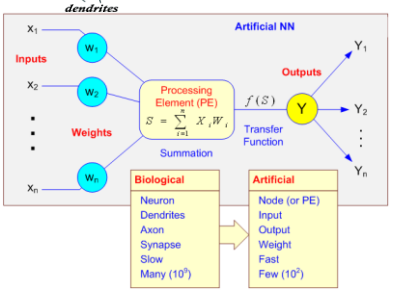
2019 - 2020

Ando Ki, Ph.D.
adki@future-ds.com

Table of contents

Modeling a neuron

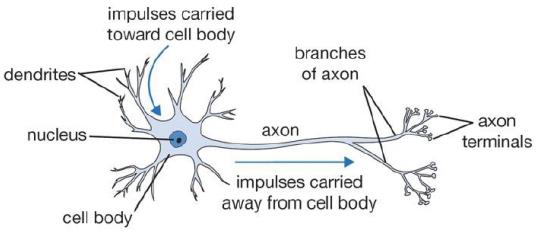


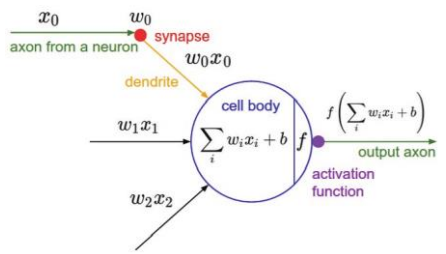


https://www.quora.com/What-is-deep-learning

- Neuron: 신경세포(神經細胞)
  - Dendrite: 수상돌기(樹狀突起)
    - input
  - Axon: 축삭돌기(軸索突起)
    - output
    - Branches of axon
    - Terminals of axon (axon tip)
      - synaptic knob
  - Synapse: 연접
    - junction between two nerve cells
- Human
  - whole brain
    - ~86 billion neurons (Giga, 10^9)
    - ~100 trillion synapses (Tera, 10^{12})
  - cerebral cortex: 대뇌피질
    - 19~23 billion neurons

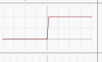
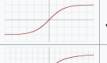
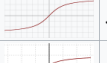
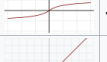
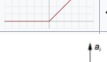
Modeling a neuron





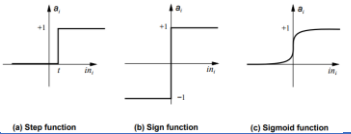
https://en.wikipedia.org/wiki/Activation_function

- Activation functions

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1+e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x\rvert}$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

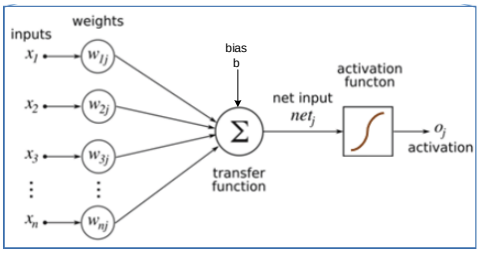
sigmoid

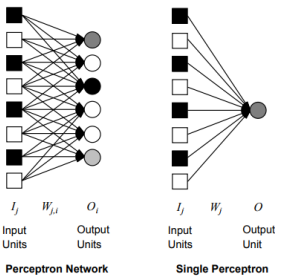
sigmoid



Perceptron: single layer neural network

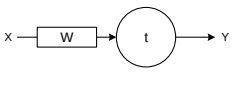
- Perceptron is a single artificial neuron that computes its weighted input and uses a threshold activation function.
  - It is also called a TLU (threshold logic unit).
  - It effectively separates the input space into two categories by the hyperplane: W*X+b = 0
  - Perceptron is a linear classifier.
    - Cannot deal with non-linear cases
  - Perceptron refers to a particular supervised learning model with backpropagation learning algorithm.
  - Perceptron is an algorithm for supervised learning of binary classifiers.

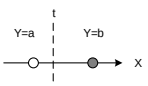


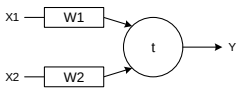


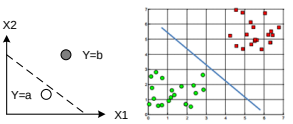
5

How perceptron classifies hyperplane

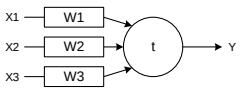


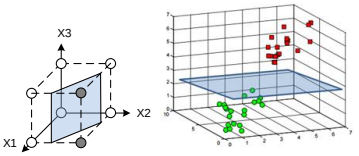




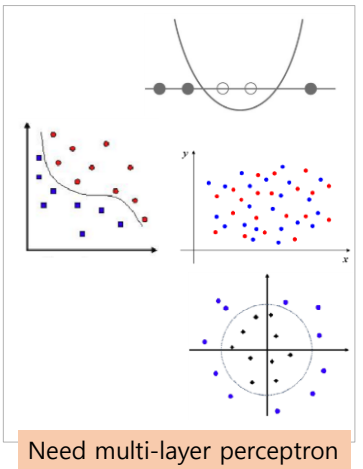


W1*x1+W2*x2 + W0 = 0
➔ y = ax + b





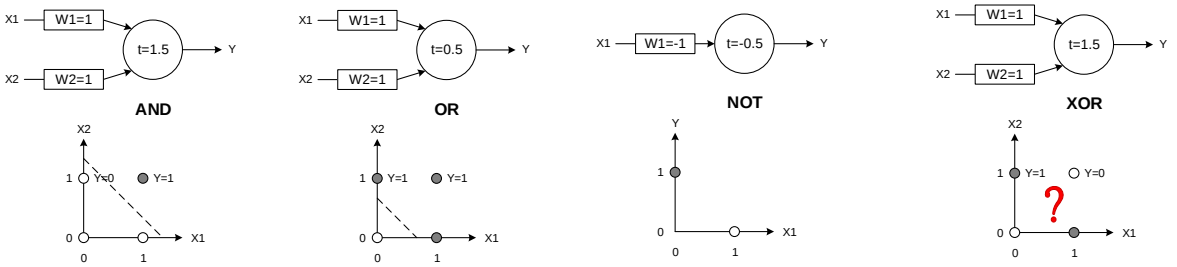
W1*x1+W2*x2 + W3*x3 + W0 = 0
➔ z = ax + by + c



Need multi-layer perceptron

6

Perceptron: Boolean



Perceptron: Boolean AND training

- Step 1: initialize the weight and the threshold.
  - Weights may be initialized to 0 or to a small random value.
- Step 2: repeat until error is less than a specific value
  - Calculate output (for j-th test set)

$$y_j(t) = f[\mathbf{w}(t) \cdot \mathbf{x}_j] = f[w_0(t)x_{j,0} + w_1(t)x_{j,1} + w_2(t)x_{j,2} + \cdots + w_n(t)x_{j,n}]$$

  - Update weights (for i-th path for j-th test set) (d_j is desired or expected value)

$$w_i(t+1) = w_i(t) + (d_j - y_j(t))x_{j,i}, \text{ for all features } 0 \le i \le n.$$

  - Calculate error

$$\frac{1}{s}\sum_{j=1}^{s} |d_j - y_j(t)|$$

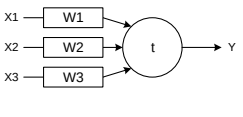
- Training set [{inputs: expected}]
  - T0={0,0:0}, T1={0,1:0}, T2={1,0:0}, T3={1,1:1}
- for T0 and T1 and T2 (assume all weights are 0)
  - y = 0x0+0x0 = 0
  - e = 0-0 = 0 (no error)
  - No update since no error
- for T3
  - y = 1x0+1x0=0
  - e = 1-0 = 1
  - w0 = 0 + (1-0) = 1
  - w1 = 0 + (1-0) = 1
- After updating
  - for T3, T2, and T1
    - y = 1x1+1x1=2 => apply threshold = 1.5
      - e = 1-1 = 0
    - y = 1x1+1x0=1 => apply threshold = 1.5
      - e = 0-0 = 0
    - y = 1x0+1x1=1 => apply threshold = 1.5
      - e = 0-0 = 0
    - y = 1x0+1x0=0 => apply threshold = 1.5
      - e = 0-0 = 0

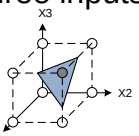
Perceptron: Boolean OR training

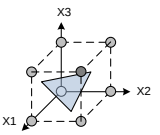
- Training set [{inputs: expected}]
  - T0={0,0:0}, T1={0,1:1}, T2={1,0:1}, T3={1,1:1}
- for T0 (assume all weights are 0)
  - y = 0x0+0x0 = 0
  - e = 0-0 = 0 (no error)
  - No update since no error
- for T1
  - y = 0x0+0x1=0
  - e = 1-1 = 1
  - w0 = 0 + (1-1) = 1
  - w1 = 0 + (1-1) = 1
  - Update w0 and w1
- After updating
  - for T2
    - y = 1x1+1x0=1 => apply threshold = 1
    - e = 1-1 = 0
  - No update since no error
- for T3
  - y = 1x1+1x1=2 ==> apply threshold = 1
  - e = 1-1 = 0
  - No update since no error

Your project

- Find W and threshold (t) for three inputs Boolean.





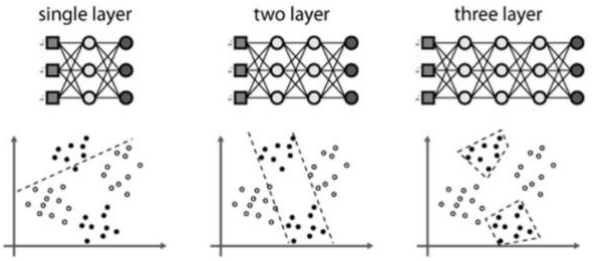


X1	X2	X3	Y=AND(X1,X2,X3)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

X1	X2	X3	Y=OR(X1,X2,X3)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

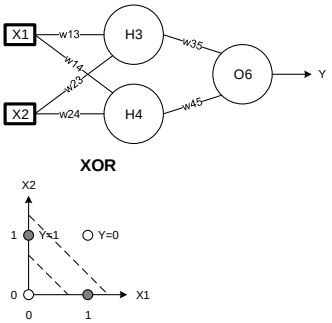
MLP: Multi-layered perceptron (다층 퍼셉트론)

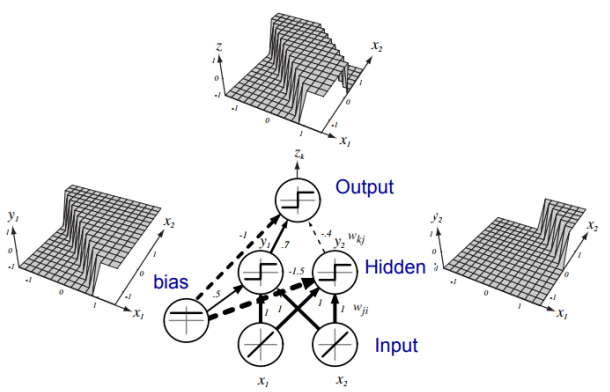
Structure	Types of Decision Regions	Exclusive-OR Problem
Single-Layer	Half Plane Bounded By Hyperplane	A B / B A
Two-Layer	Convex Open Or Closed Regions	A B / B A
Three-Layer	Arbitrary (Complexity Limited by No. of Nodes)	A B / B A



Multi-layered perceptron

- Two-unit network (two layers)

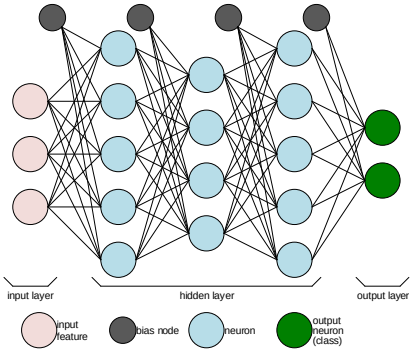




(from Pascal Vincent's slides)

Layer-wise organization

- 3 types of layers
  - Input layer
  - hidden layer
  - output layer



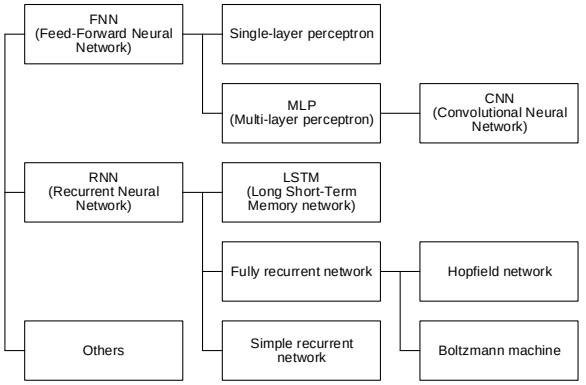
fully-connected multi-layered neural network

- input layer: not counted for the number of layers
- hidden layer
- output layer

- For the picture on the left
  - assume fully connected
  - 4-layered including 3-hidden layers
  - 17 neurons: 5+4+5+2
  - 65 weights: 3x5+5x4+4x5+5x2
    - ➲ not including bias
  - 16 biases: 5+4+5+2
  - 82 learnable parameters: 65+16

- Modern neural network
  - 10~20 layers, ~100 million parameters
  - How about 125 layers?

Categories of ANN (Artificial Neural network)



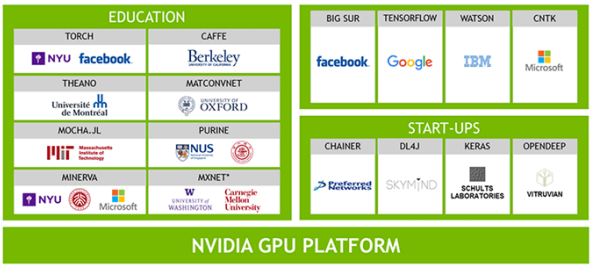
- Fully-Connected NN
  - feed forward
  - Multi-Layer Perceptron (MLP)
- Convolutional NN (CNN)
  - feed forward, sparsely-connected
  - Image recognition
  - AlphaGo
- Recurrent NN (RNN)
  - feedback
- Long Short-Term Memory (LSTM)
  - feedback + storage
  - Microsoft speech recognition
  - Google neural machine translation (GNMT)

See neural network topology: http://www.asimovinstitute.org/neural-network-zoo/

Popular Frameworks

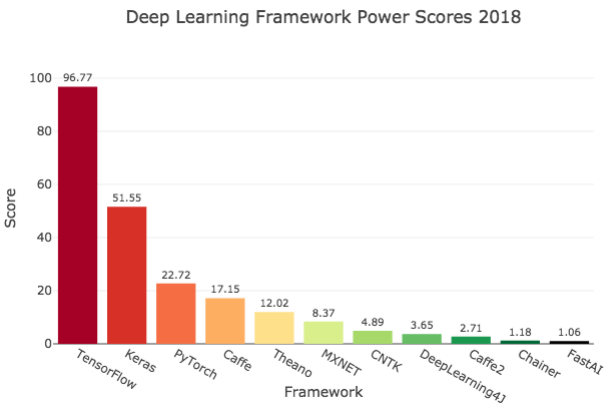
- Popular Frameworks with supported interfaces
  - Caffe
    - Berkeley / BVLC (Berkeley Artificial Intelligence Research)
    - C, C++, Python, Matlab
  - TensorFlow
    - Google Brain
    - C++, Python
  - PyTorch
  - theano
    - U. Montreal
    - Python
  - torch
    - Facebook / NUU
    - C, C++, Lua
  - CNTK
    - Microsoft
  - MXNet
    - Carnegie Mellon University / DMLC (Distributed Machine Learning Community)

https://developer.nvidia.com/deep-learning-frameworks



https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/

Popularity



Deep Learning Framework Deep Learning Framework Power Scores (by Jeff Hale) http://bit.ly/2GBa3tU

https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a

1. TensorFlow
2. Keras
3. PyTorch
4. Caffe
5. theano
6. mxnet
7. CNTK
8. DL4J
9. Caffe2
10. Chainer
11. fast.ai

㈜퓨쳐디자인시스템
34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호
(042) 864-0211~0212 / contact@future-ds.com / www.future-ds.com

Future Design Systems, Inc.
Faculty Wing F723, KAIST Munji Campus, 193 Munji-ro, Yuseong-gu, Daejeon 34051, Korea
+82-042-864-0211~0212 / contact@future-ds.com / www.future-ds.com

FUTURE
Design Systems