Deep Learning

2019 - 2020

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

Table of contents

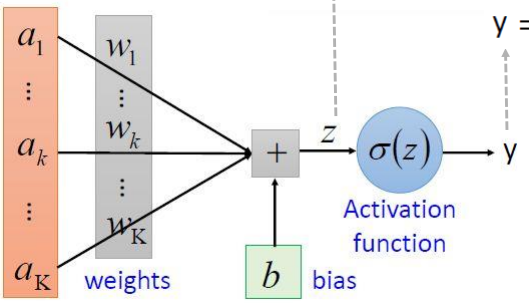
Artificial neuron: Perceptron

- Artificial Neuron: Perceptron
  - inputs
  - output
  - weights
  - bias
  - activation function

$$(a_1, a_2, \cdots, a_K) \times \begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ W_K \end{pmatrix} + b = z$$

$$z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b$$

$$y = \sigma(z)$$

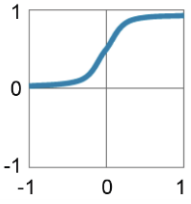


3

Artificial neuron: activation functions

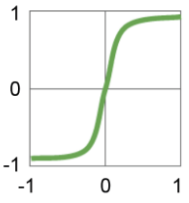
Logistic, soft step

Sigmoid



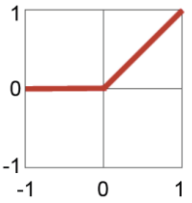
$$y = \frac{1}{1 + e^{-z}}$$

Hyperbolic Tangent



$$y = \frac{2}{1 + e^{-2z}} - 1$$

Rectified Linear Unit (ReLU)

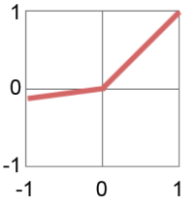


$y = 0 \; for \; z < 0$

$y = z \; for \; z \geq 0$

y=max(x,0)

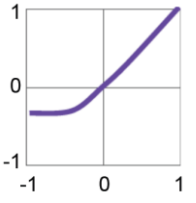
Leaky ReLU



$y = 0.01z \; for \; z < 0$

$y = z \; for \; z \geq 0$

Exponential LU



$y = \alpha(e^z - 1) \; for \; z < 0$

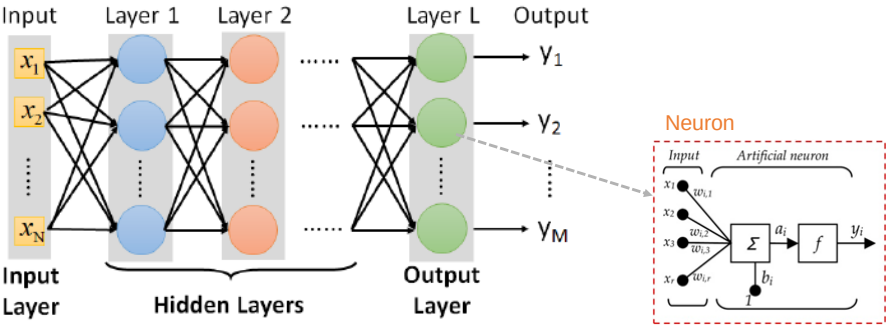
$y = z \; for \; z \geq 0$

$$\frac{dy(z)}{dz} = \frac{d}{dz}\left[\frac{1}{1 + e^{-z}}\right] = \frac{d}{dz}(1 + e^{-z})^{-1} = -(1 + e^{-z})^{-2}(-e^{-z}) = y(z) \cdot (1 - y(z))$$

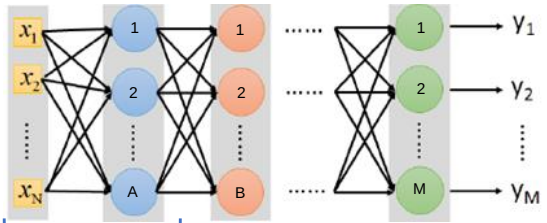
4

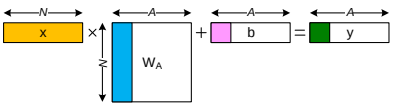
Artificial Neural Network: ANN

- Artificial Neural Network: ANN
  - Network structure by different connections
  - Each neuron can has different values of weights and bias
  - Weights and biases are network parameter θ .



Artificial Neural Network: ANN





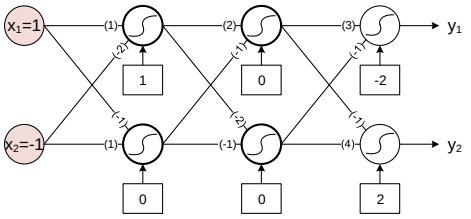
N: number of inputs
A: number of hidden layers

$$\begin{bmatrix} X_1 & X_2 & \cdots & X_N \end{bmatrix} \times \begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{A,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{A,N} \\ \vdots & & & \vdots \\ w_{1,N} & w_{2,N} & \cdots & w_{A,N} \end{bmatrix} + \begin{bmatrix} b_1 & b_2 & \cdots & b_A \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \cdots & y_A \end{bmatrix}$$

$$\begin{bmatrix} w_{1,1} & w_{2,1} & \cdots & w_{A,1} \\ w_{1,2} & w_{2,2} & \cdots & w_{A,N} \\ \vdots & & & \vdots \\ w_{1,N} & w_{2,N} & \cdots & w_{A,N} \end{bmatrix}^T \times \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_A \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_A \end{bmatrix}$$

Fully connected feed-forward network: FC-FFN

- Activation function: E.g., Sigmoid – S-shaped function



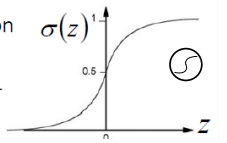
$$[1, -1] \times \begin{bmatrix} 1, -1 \\ -2, 1 \end{bmatrix} + [1, 0] = [4, -2] \xrightarrow{\text{sigmoid}} [0.98, 0.12]$$

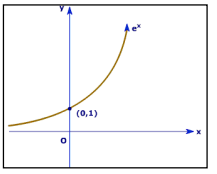
$$[0.98, 0.12] \times \begin{bmatrix} 2, -2 \\ -1, -1 \end{bmatrix} + [0, 0] = [1.84, -2.08] \xrightarrow{\text{sigmoid}} [0.86, 0.11]$$

$$[0.86, 0.11] \times \begin{bmatrix} 3, -1 \\ -1, 4 \end{bmatrix} + [-2, 2] = [\ ??,\ ??] \xrightarrow{\text{sigmoid}} [\ ??,\ ??]$$

Sigmoid Function

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





x_1 x_2 y_1 y_2

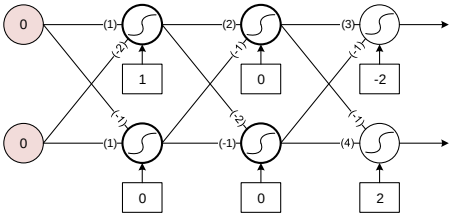
f ([1, -1]) = [0.62, -0.83]

f ([0, 0]) = [051, 0.85]

7

Do it yourself

- Calculate the output



8

Optional output layer: Softmax

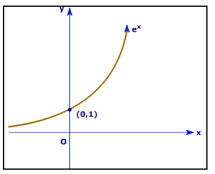
- Outputs of artificial neural network will be any values from very small to very large including negative.

f ([1, -1]) = [0.62, -0.83]
f ([0, 0]) = [051, 0.85]

The output can be any value.
→ Hard to interpret.

- Softmax for output layer
  - Softmax is a function to transform a number of values to a range of value to between 0 ~ 1.
    - Score (-inf, inf) ==> probabilities [0,1]
  - Multinomial logistic or normalized exponential function

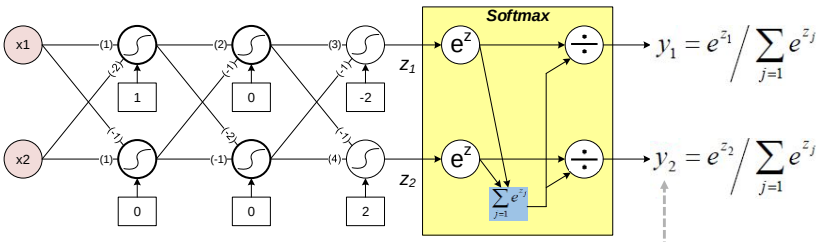
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

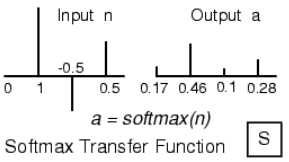


9

Optional output layer: Softmax

- Softmax converts score to probability: Score (-inf, inf) ==> probabilities [0,1]
  - un-normalized probabilities (summation will not give 1): e^{z_j} for result j.
  - normalized probabilities (summation will give 1): -- see below --



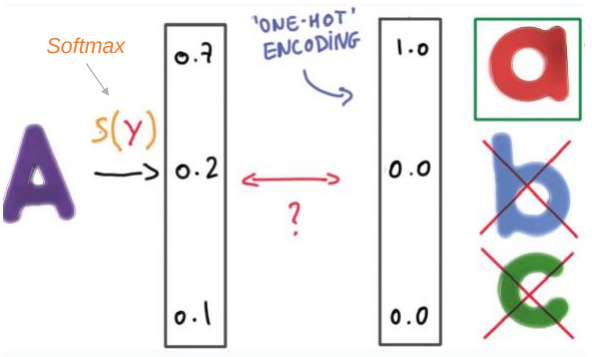


$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

10

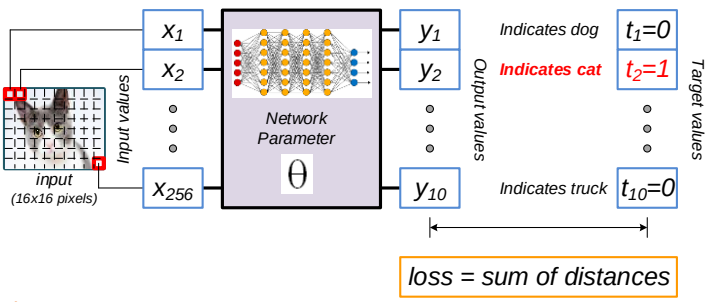
Optional output layer: one-hot encoding

- One-hot encoding by encoding class labels
- Select one only among many.



How to find a good or the best network: Loss/Cost

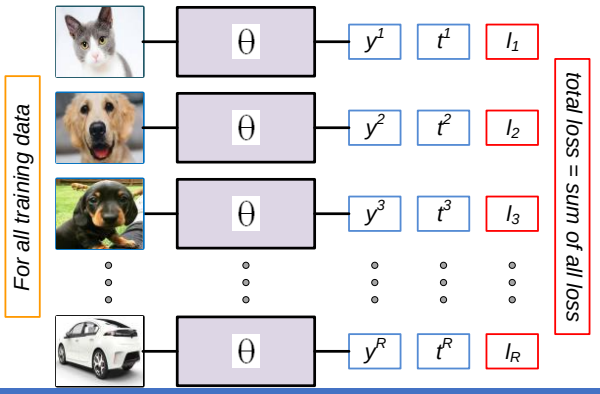
- **Loss function** is the distance between the network output and the target
  - cost function or error function
  - It indicates how good the result is.
  - There can be different loss functions.
    - The simplest one will be a summation of | t – y |.
      - Perfect match will give 0.



- Training error: error by training data set
- Generalization error (test error): error by test data set in order to evaluate the training model.

How to find a good or the best network: Total Loss

- Total loss (L) is a sum of losses (l_r)
  - Make it as small as possible
- Training means to find the network parameter θ that minimize total loss L.
  - This means we should modify the network parameter according to the total loss.



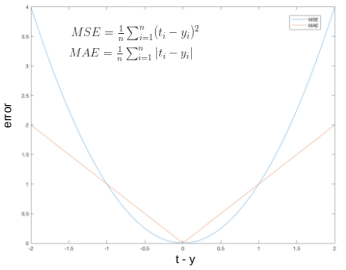
$$L = \sum_{r=1} l_r$$

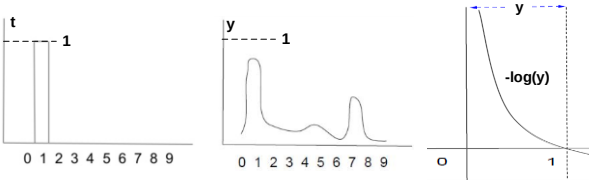
Sum of losses for R test images

Cost functions (error function)

- Absolute error
  - Sum of absolute errors
    - sum(|t – y|)
  - Mean absolute errors (MAE)
    - sum(|t-y|)/n
- Squared error loss
  - Sum of squared errors
    - sum((t - y)**2)
  - Mean squared errors (MSE)
    - sum((t-y)**2)/n
  - Root mean square errors (RMSE)
    - (MSE)**(1/2)
- Cross-entropy loss
  - For classification after Softmax
  - Sum of cross-entropy loss
    - -sum(t*log(y))
      - all except t=1 does not contribute
    - or - sum[t*log(y) + (1-t)*log(1-y)]
      - add cost when t is not 1.

- y: inference value or calculated value
- t: target value

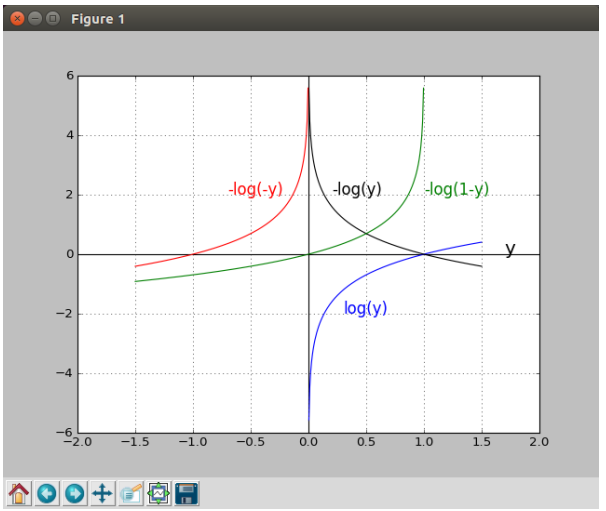




-log(y) emphasizes error (y) when softmax result (y) is small.
y<1 means error, y=1 means correct.

Log plots

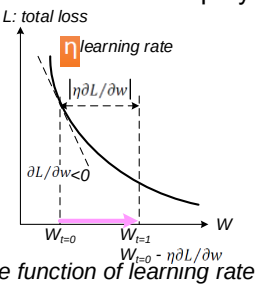
```
mport numpy as np
from matplotlib import pyplot as plt

y = np.linspace(-1.5, 1.5, 400)

plt.plot(y, np.log(y), color='blue')
plt.text(0.3, -2, 'log(y)', fontsize=15, color='blue')

plt.plot(y, -np.log(y), color='black')
plt.text(0.2, 2, '-log(y)', fontsize=15, color='black')

plt.plot(y, -np.log(-y), color='red')
plt.text(-0.7, 2, "-log(-y)", fontsize=15, color='red')

plt.plot(y, -np.log(1-y), color='green')
plt.text(1.0, 2, "-log(1-y)", fontsize=15, color='green')

plt.grid()
plt.show()
```

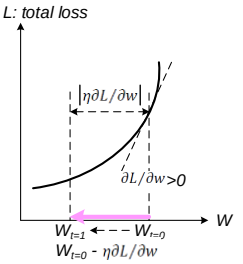


15

How to minimize total loss by changing [W] and [b]

- If we can find how the network parameters affect the total loss, it may be possible to figure out how to minimize the total loss.
- However, the number of parameters is too larger to figure out.
  - AlexNet: 650K neurons, 8 layers, 60 Million parameters
- So we apply gradual iterative progress method in step by step called '**Gradient descent**'. It is called *optimization algorithm.*



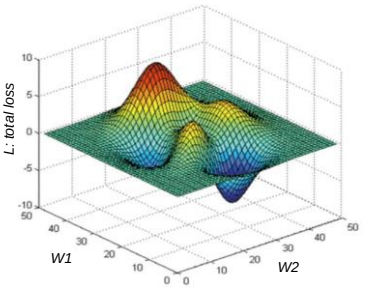


- *Negative slope ➔ increase W by some function of learning rate*
- *Positive slope ➔ decrease W*
- *Steep slope ➔ large change of W for the next time*
- *go on until the slope is small enough, i.e., inflection point*

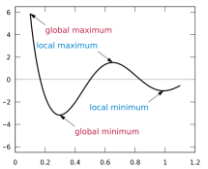
16

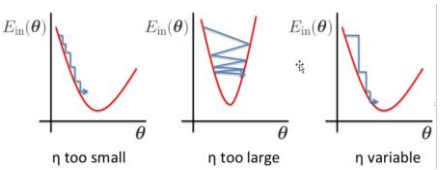
Optimization algorithm: gradient descent

- *Initial value problem*
  - *different initial point leads to different minima*
- *Local minimum problem (get stuck in local minima)*
  - *never guarantee global minima*
- *Learning rate problem*
  - *large learning rate could cause oscillation*
  - *small learning rate results in slow learning*



- *Vanishing gradient problem*
  - If a change in the parameter's value causes very small change in the network's output - the network just can't learn the parameter effectively, which is a problem.
- Gradient Exploding





17

How to compute gradient

- Backpropagation
  - 1. Feed-forward computation
  - 2. Back-propagation to the output layer
  - 3. Back-propagation to the hidden layers
  - 4. Weight updates

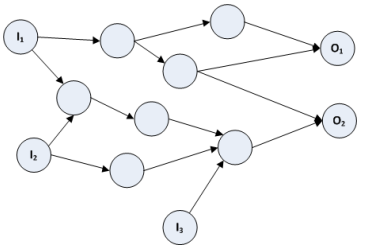
$\partial L / \partial w$

Do not panic. You do not need to worry about it because the program will do it.

18

Neural Network (NN)

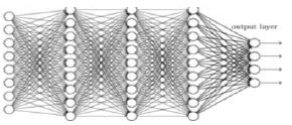
- NN has three elements
  - Architecture: the graph , weights/biases, activation functions
  - Activity Rule: weights/biases, activation functions
  - Learning Rule: a typical one is backpropagation algorithm
- The architecture basically determines the capability of a specific NN
  - Different architectures are suitable for different applications.
  - The most general architecture of an ANN is a DAG (directed acyclic graph).



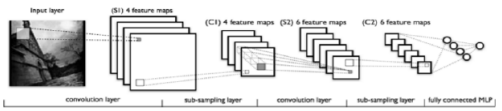
19

Popular types of Neural Network (NN)

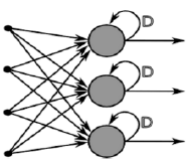
- DNN: Deep NN
  - More general model
  - fully connected
  - feed-forward (i.e., MLP: multilayer perceptron)
  - speech, image processing, natural language processing (NLP)



- CNN: Convolutional NN
  - Common image optimization
  - connected locally (i.e., sparsely-connected)
  - feed-forward
  - object/facial recognition



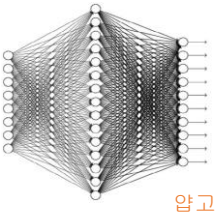
- RNN: Recurrent NN
  - context driven, time-series optimization
  - variable connectivity
  - feed-back in addition to feed-forward
  - NLP and speech recognition
  - Long Short-Term Memory (LSTM)
    - feed-back + storage



20

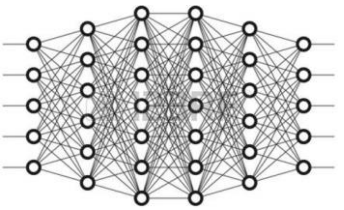
Deep neural net

- Any continuous function can be realized by a network with one hidden layer with sufficient neurons. (Universality theorem, universal approximation theorem)
  - A hidden layer network can represent any continuous function
  - A shallow fat neural net.



얇고 굵다(두껍다)

- Deep thin neural net (deep NN) is better than shallow fat net.
  - Using multiple layers of neurons to represent some functions are much simpler.
    - Less parameters ➔ less computation



깊고 가늘다(얇다)

21

NN categories by applications

Category	Theme	Technology (NN)	Application
Pattern classification	Image classification	CNN	Number, character, image
	Text classification	RNN, NLP	Text
Pattern recognition	Image segmentation	CNN	CT cancer
	Face recognition	CNN	Door lock
	Voice recognition	RNN	AI secretary
Synthesis	Voice synthesis	RNN, NLP	Voice style transform
	Image synthesis	CAN, GAN	Photo transform
Forecasting	Time-serial	RNN, DNN	Traffic estimation
	Weather forecasting	RNN, DNN	Weather
	Anomaly detection	DNN	Network security; Credit card
Control	Robot control	Reinforcement learning	Intelligent IoT control, DRONE
	Game	Reinforcement learning	Intelligent game agent
Optimization	Decision making	DNN, Tree search	Financial decision making
	Mathematical analysis	DNN	Product planning

22

Popular DNNs and Frameworks

- Popular DNNs
  - AlexNet
    - First CNN Winner of ILSVRC
    - Uses LRN (deprecated after this)
  - VGG-16
    - Goes Deeper (16+ layers)
    - Uses only 3x3 filters (stack for larger filters)
  - GoogLeNet (v1)
    - Reduces weights with Inception and only one FC layer
    - Inception: 1x1 and DAG (parallel connections)
    - Batch Normalization
  - ResNet
    - Goes Deeper (24+ layers)
    - Shortcut connections
- Popular Frameworks
  - TensorFlow
  - Caffe

㈜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