- **사전 지식 및 BDA Contents list**
- **R/Shiny**
- **AI, ML**
- **Web/Cloud**

Source :

빅데이터의 정보화 – 지능화 – 스마트화 흐름

산업/제조
기상
의료
금융
사회
Infrastructure
통신
안전/공공
석유/화학
건설
광산

데이터

Text
Image
영상
음성
DB
3D-Model



x1 x2 x3 ....
- 작업시간
- 공정
- 기계
- 날짜

y1 y2 ....
- 등급
- 합/불

융합 활용

3 스마트화

1 데이터 전 처리

데이터 정보화

CLOUD / Hadoop / DB

2 인공지능/Cognitive Computing

- Big Data Analytics
- Machine Learning
- 자연어 처리
- ML Vision

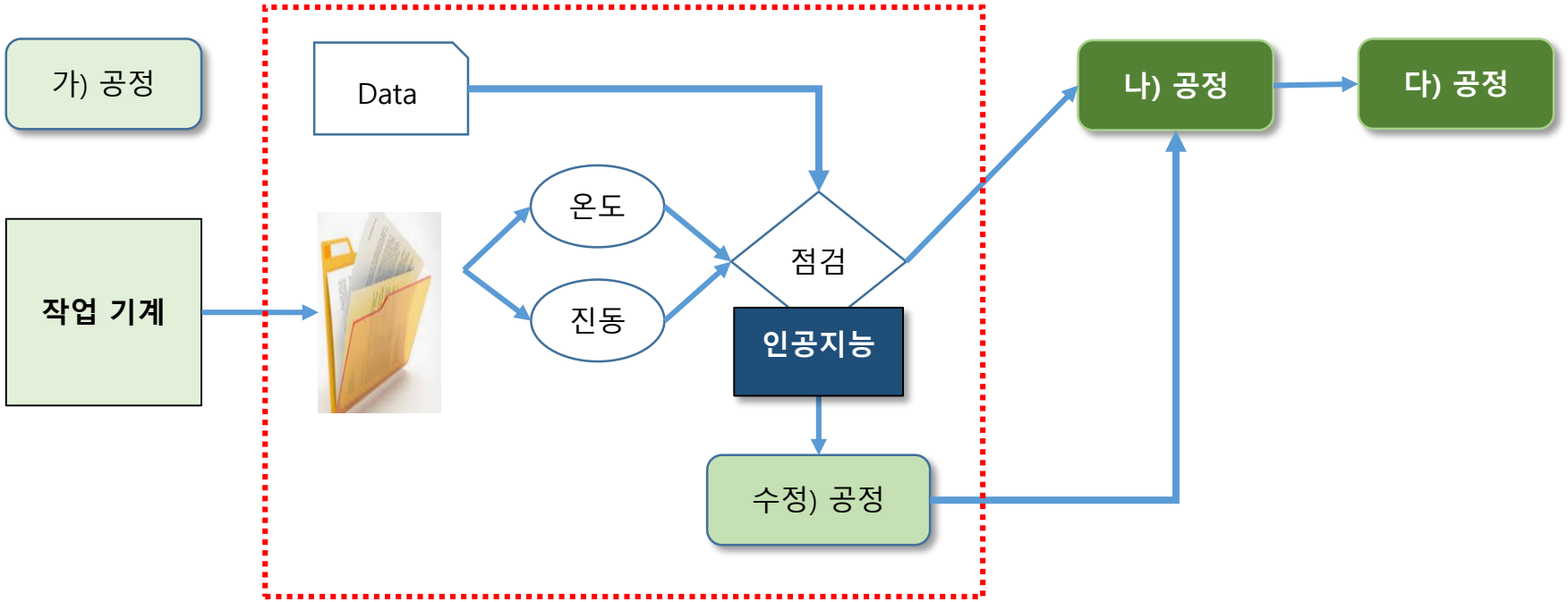
Source :

왜 스스로 분석해야 하는가 ?

작업 공정
가) 공정 – 나) 공정 – 다) 공정

해결 과제
다) 공정의 최우수 품질 등급이 많이 나오도록 조치

해결 아이디어
원인과 분석방법에 따라
다양한 아이디어가 존재

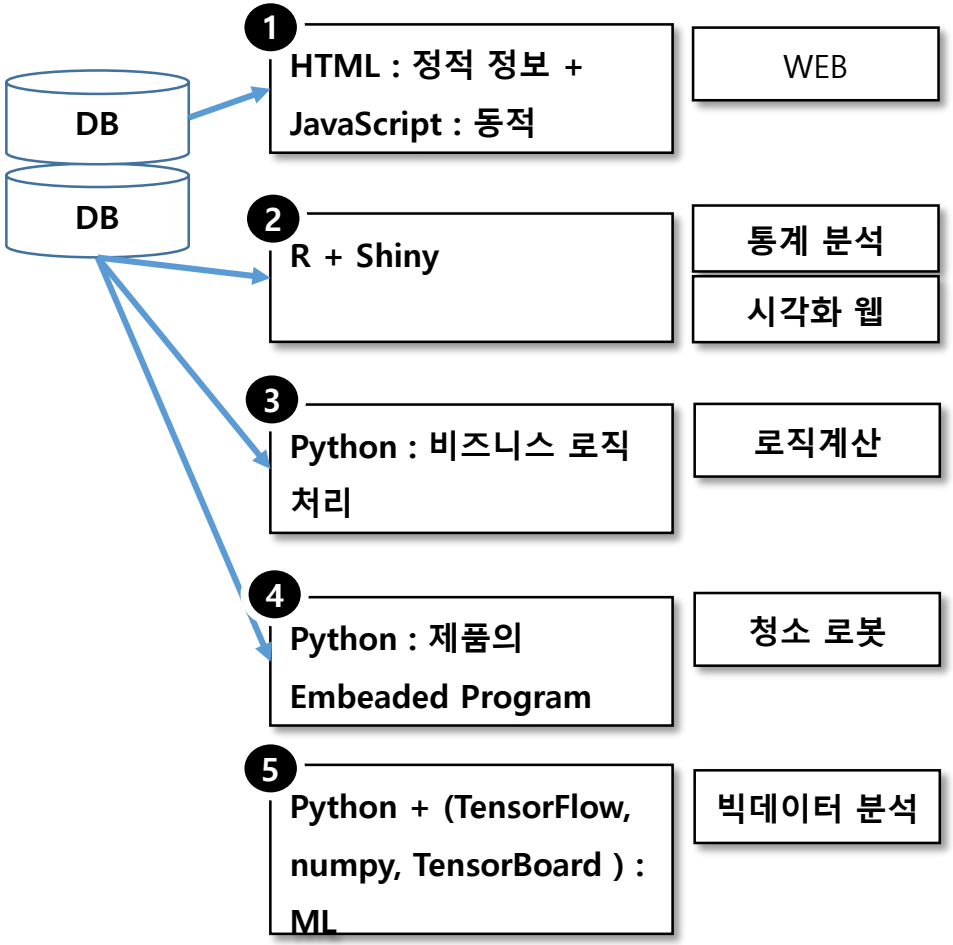


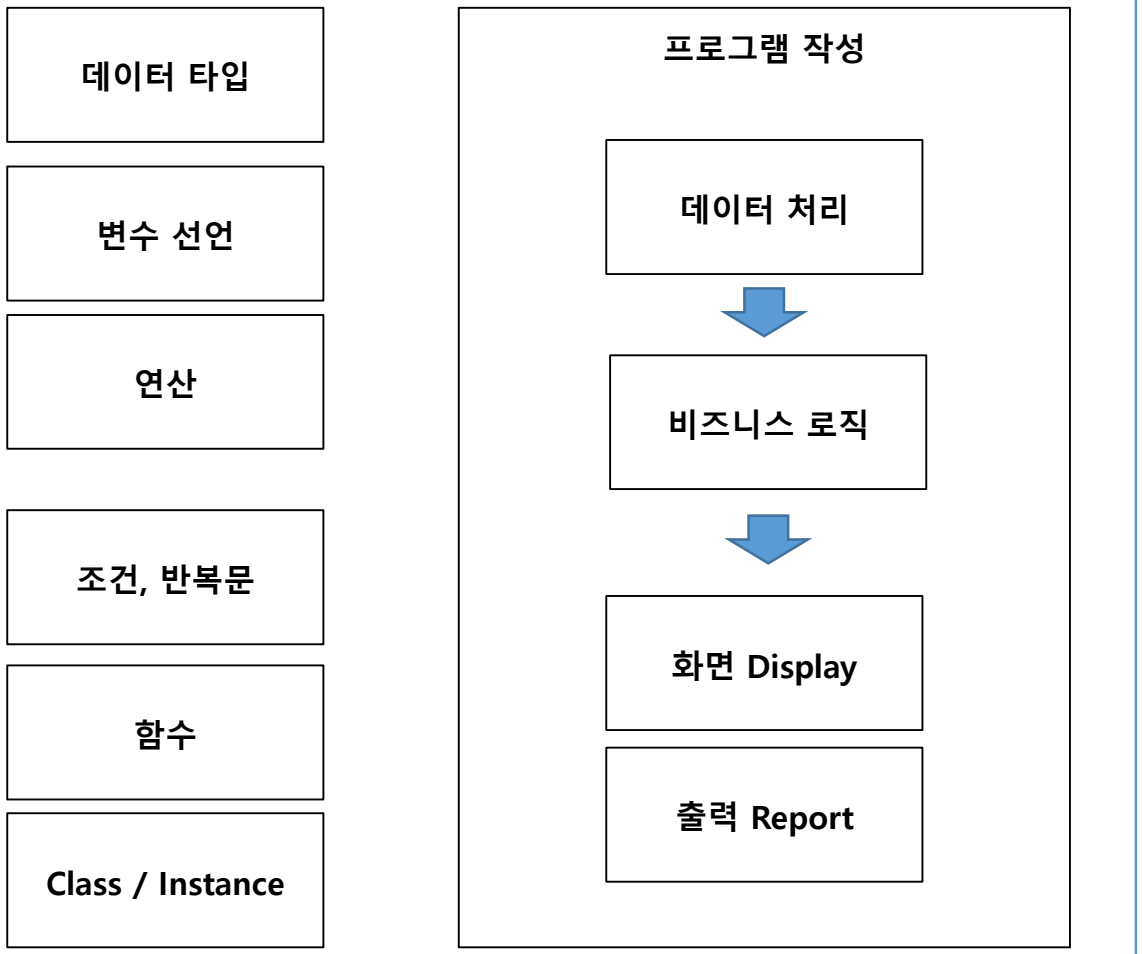
- **품질 등급 결정**

Source :

- BDA 프로그램 기본

데이터가 준비되어 있다면 오픈소스를 이용 web (정적, 동적), 통계분석, 비즈니스 로직 처리, embeaded 프로그램(Arduino로 확인), 빅데이터 분석을 직접 프로그램 해 볼 수 있다.





Source :

BDA 프로그램 학습 절차

구분	web	Java	Python	R	DB
기본 내용	• tag	데이터 타입 / 연산 / 함수 / Class / Instance 상속 / 변수 선언 / 조건, 반복문 / I/O			•
언어	HTML / CSS	Java JDK https://www.oracle.com/technetwork/java/javase/	python / tensorFlow/ numpy/pandas https://www.python.org/downloads/	R/Shiny https://www.r-project.org/	MySQL
IDE	Bitnami WAMP	eclipse https://www.eclipse.org/	anaconda https://www.anaconda.com/distribution/	Rstudio https://www.rstudio.com/	MySQL
	atom https://atom.io/		Spyder https://www.spyder-ide.org/		

Source :

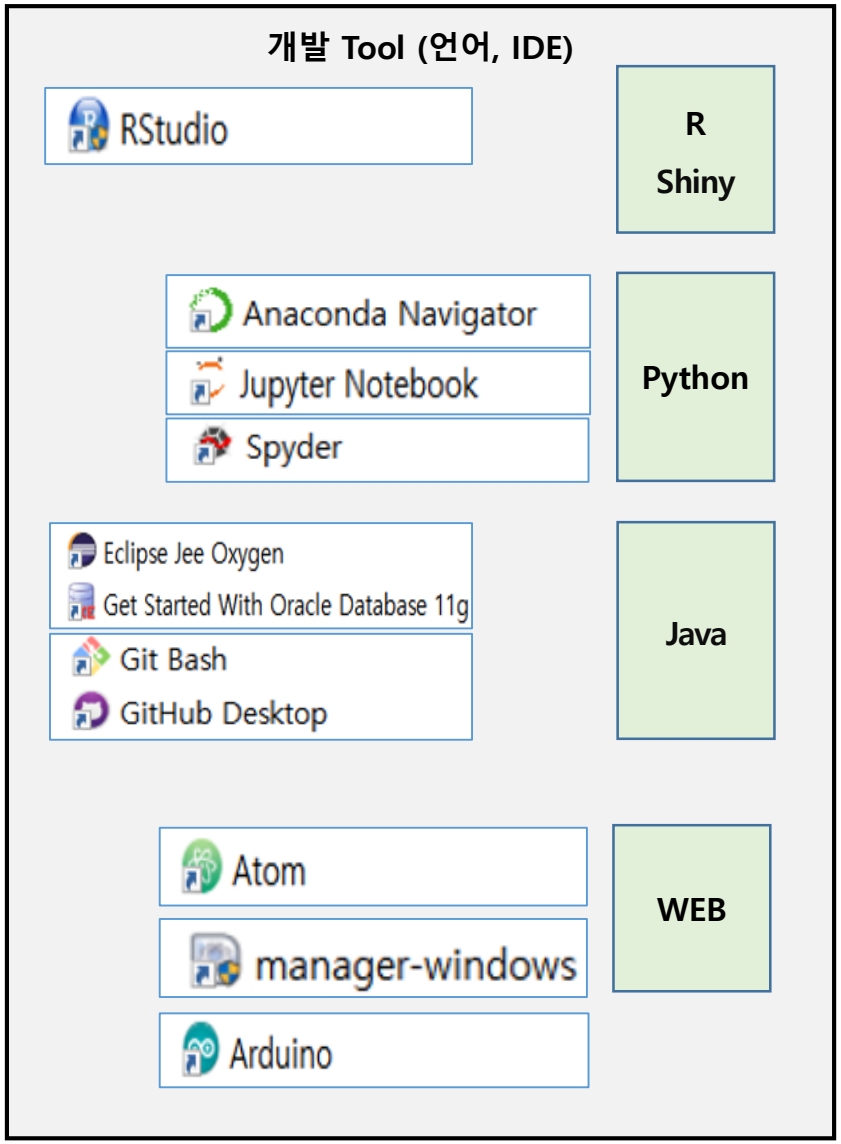
BDA 관련 오픈소스 프로그램 (언어, IDE)

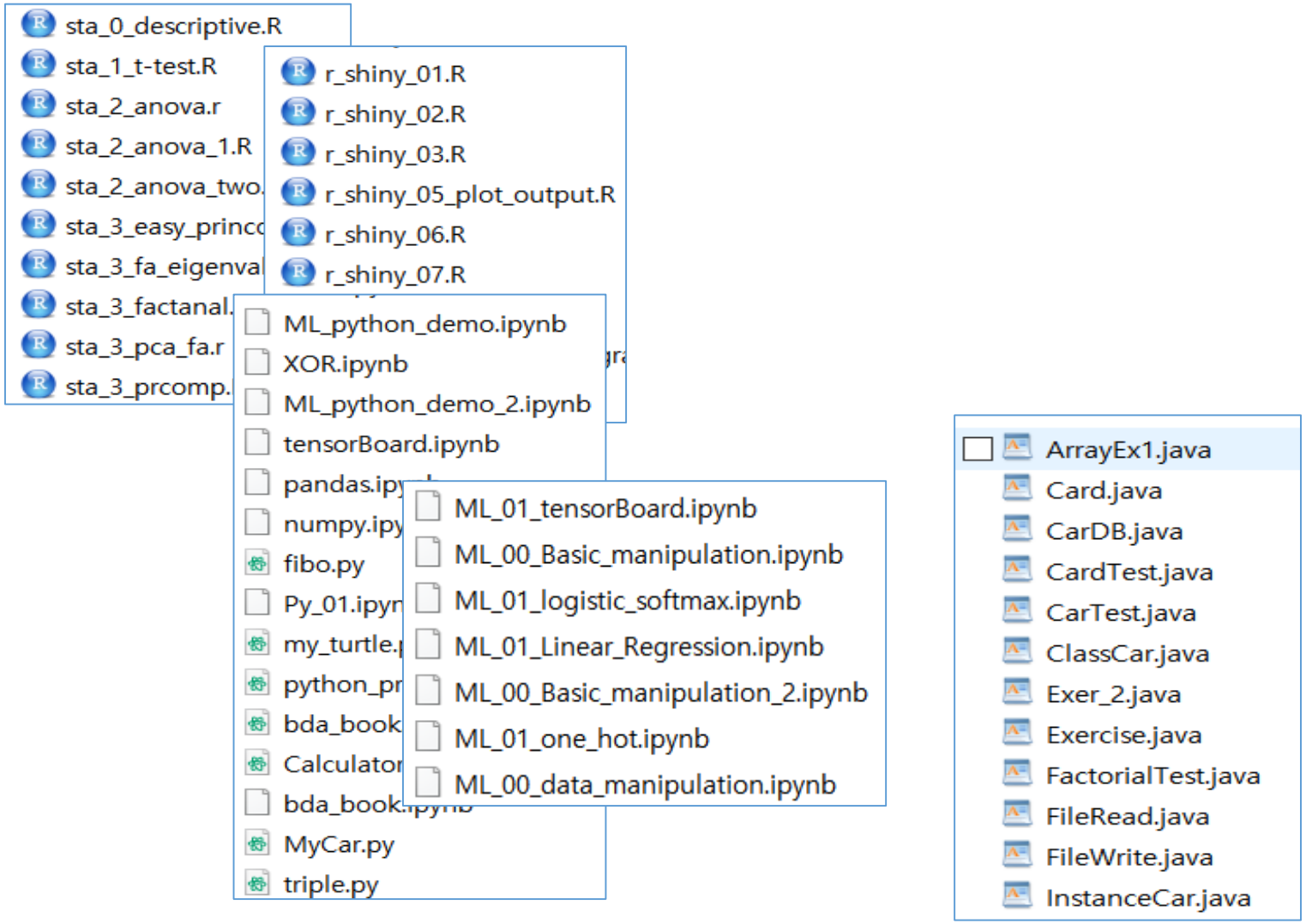
목적	web	통계분석	머신 러닝	서버 프로그램
마크업 언어 프로그램 언어	HTML / CSS	R/Shiny https://www.r-project.org/	python tensorFlow/numpy/pandas https://www.python.org/downloads/	Java JDK https://www.oracle.com/technetwork/java/javase/
통합 개발환경 (IDE)	atom https://atom.io/	Rstudio https://www.rstudio.com/	anaconda https://www.anaconda.com/distribution/ Spyder https://www.spyder-ide.org/	eclipse https://www.eclipse.org/

anaconda를 이용 python 설치 가능

Source : 411 Program_install.pptx

BDA 관련 오픈소스 프로그램 개발 환경 & Source code



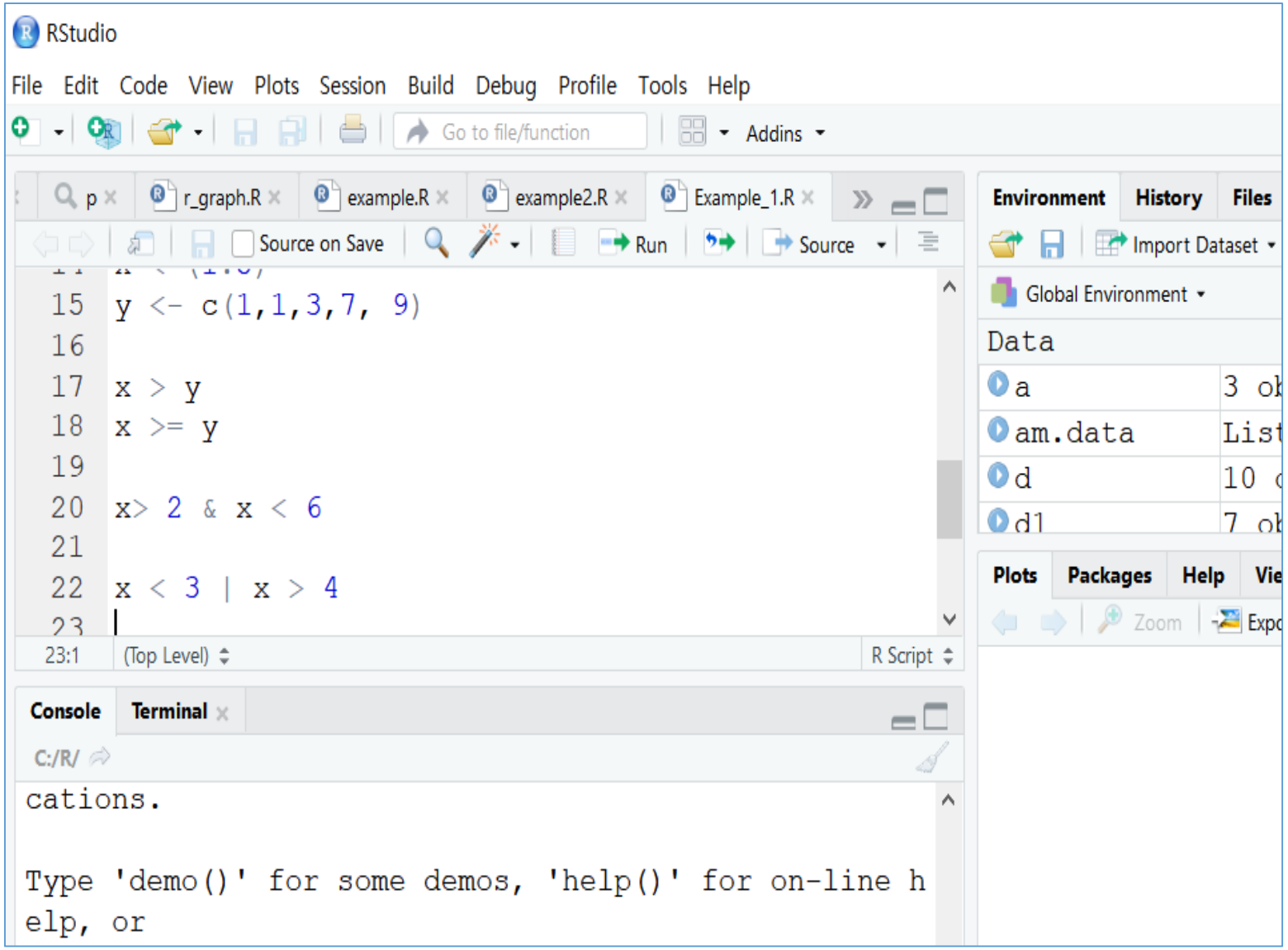


Source : 411 Program_install.pptx

Rstudio

RStudio

- C:₩RW₩work_r
- C:₩RW₩work_s



Source : 411 Program_install.pptx

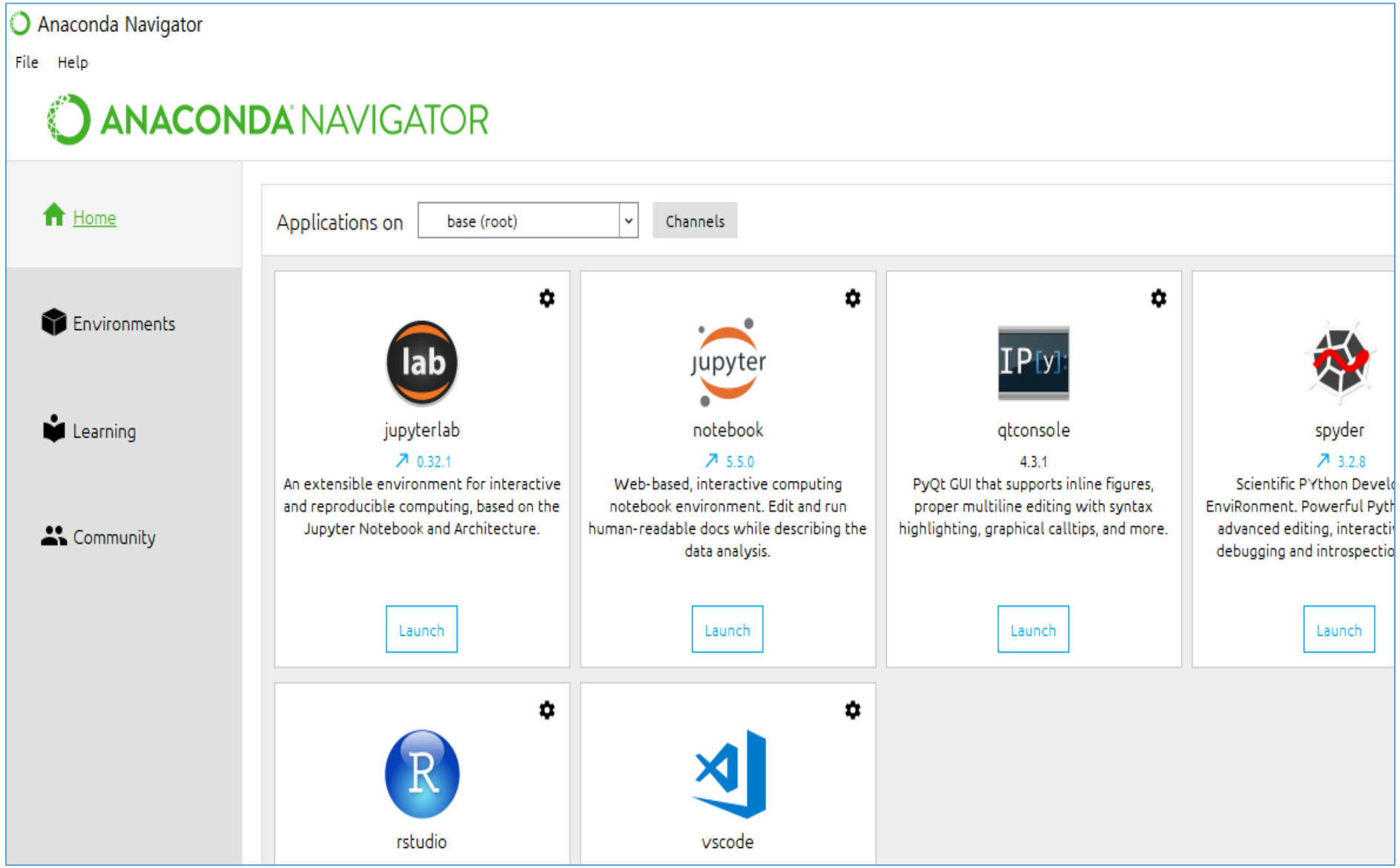
anaconda , Jupyter, Spyder

Anaconda Navigator

Jupyter Notebook

Spyder

- C:₩Users₩ ~~₩ML_01



Source : 411 Program_install.pptx

Jupyter Notebook

Jupyter Notebook

localhost:8888/tree/ML_01

jupyter

Files | Running | Clusters

Select items to perform actions on them.

0 / ML_01

- ..
- Dice.py
- exa_1.py
- ML_python_demo.ipynb
- XOR.ipynb
- ML_python_demo_2.ipynb
- tensorBoard.ipynb
- sample_1
- pandas.ipynb
- numpy.ipynb
- fibo.py
- Py_01.ipynb

jupyter ML_python_demo Last Checkpoint: 지난주 일요일 오후 6:37

File Edit View Insert Cell Kernel Widgets Help

Run Code

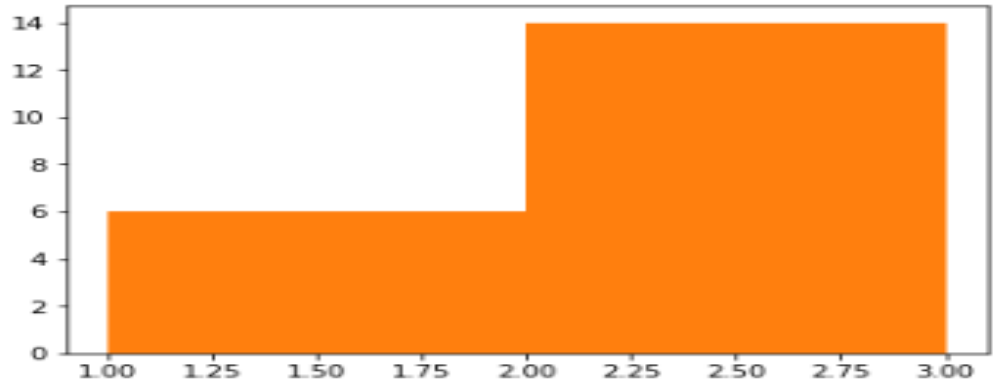
In [11]:
```
import numpy as np

x = np.random.randint(1, 4, 20)

np.mean(x)
# x
```

Out[11]: 2.05

In [13]:
```
x
```

Out[13]:
```
array([1, 3, 3, 3, 3, 1, 2, 1, 2, 3, 2, 1, 1, 2, 1, 2,
```

In [16]:
```
import random
import matplotlib.pyplot as plt

n, bins, patches = plt.hist(x, bins=20)
# a, b, c = plt.hist(x, bins=10)
plt.hist(x, bins=2)

plt.show()
```

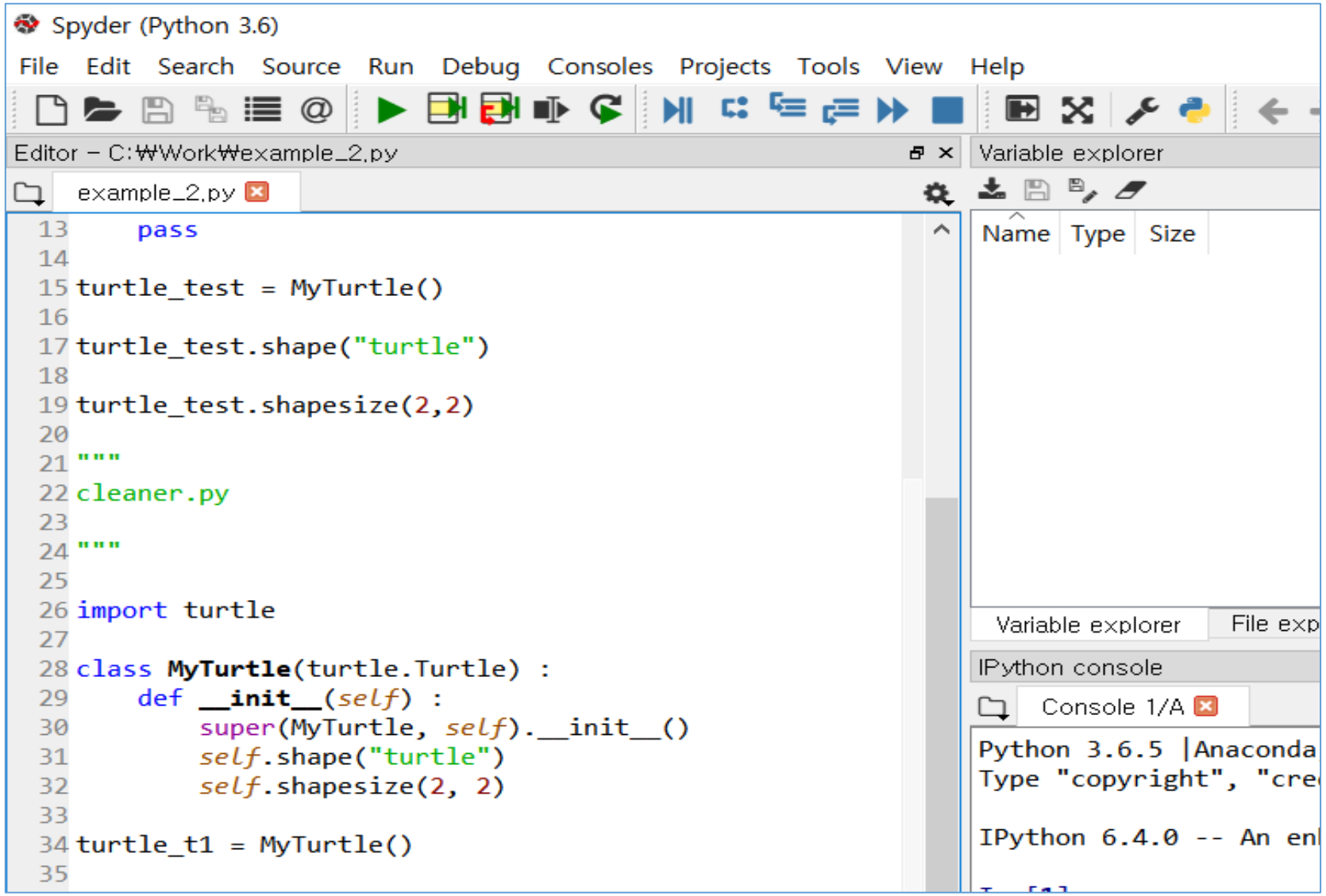


Source : 411 Program_install.pptx

Spyder

Spyder

- C:₩Work




```
    pass

turtle_test = MyTurtle()

turtle_test.shape("turtle")

turtle_test.shapesize(2,2)

"""
cleaner.py

"""

import turtle

class MyTurtle(turtle.Turtle) :
    def __init__(self) :
        super(MyTurtle, self).__init__()
        self.shape("turtle")
        self.shapesize(2, 2)

turtle_t1 = MyTurtle()
```

Source : 411 Program_install.pptx

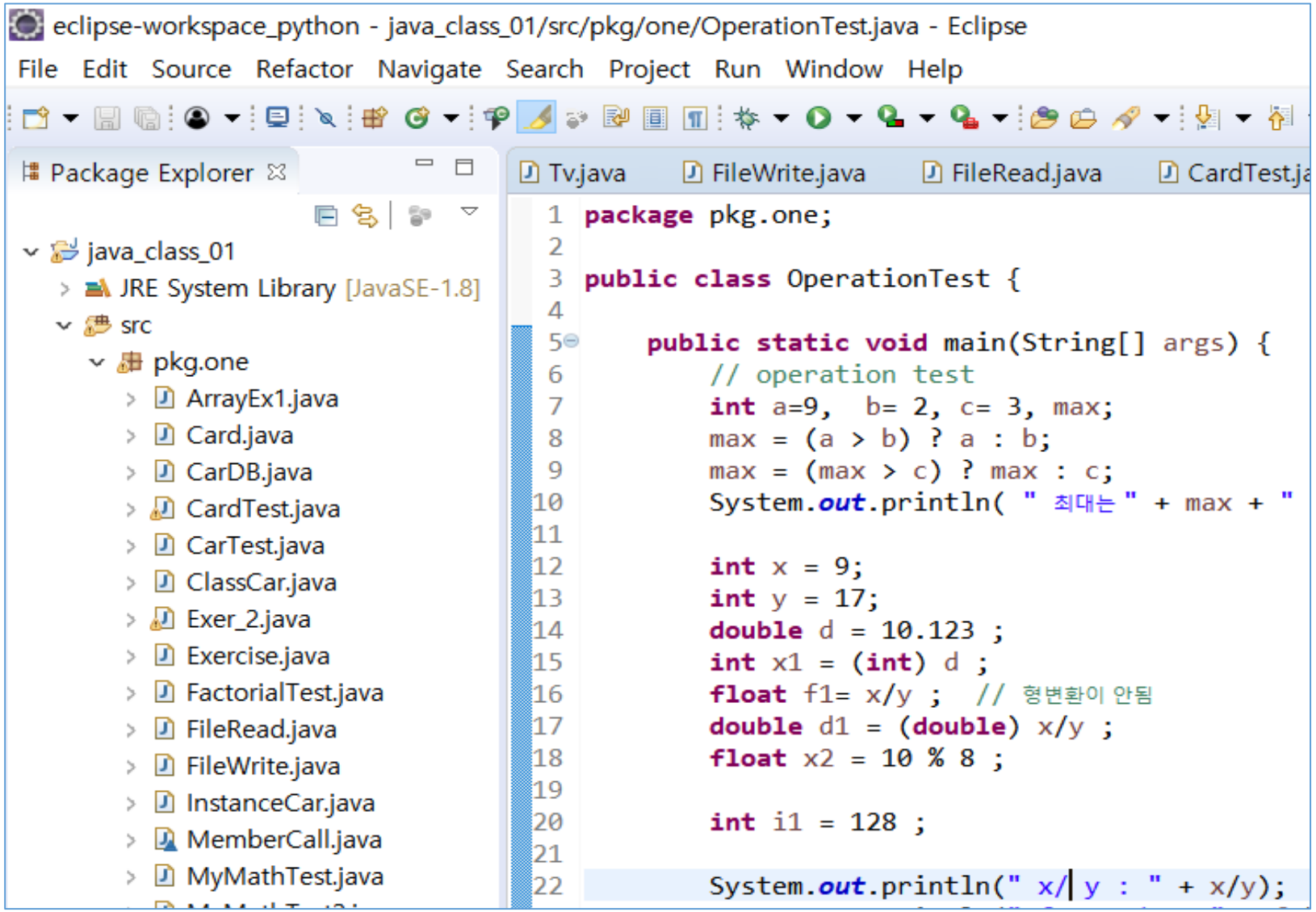
Eclipse Jee Oxygen

Get Started With Oracle Database 11g

- C:₩Users₩~~₩eclipse-workspace_python₩java_class_01₩src₩pkg₩one

Git Bash

GitHub Desktop



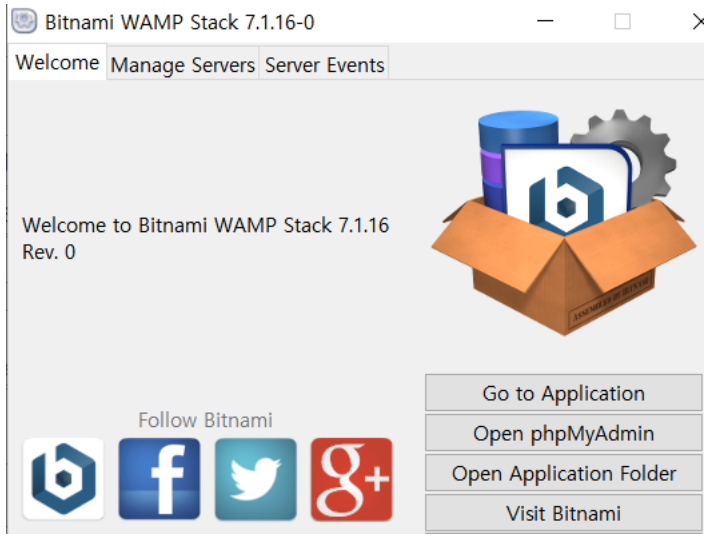
Source : 411 Program_install.pptx

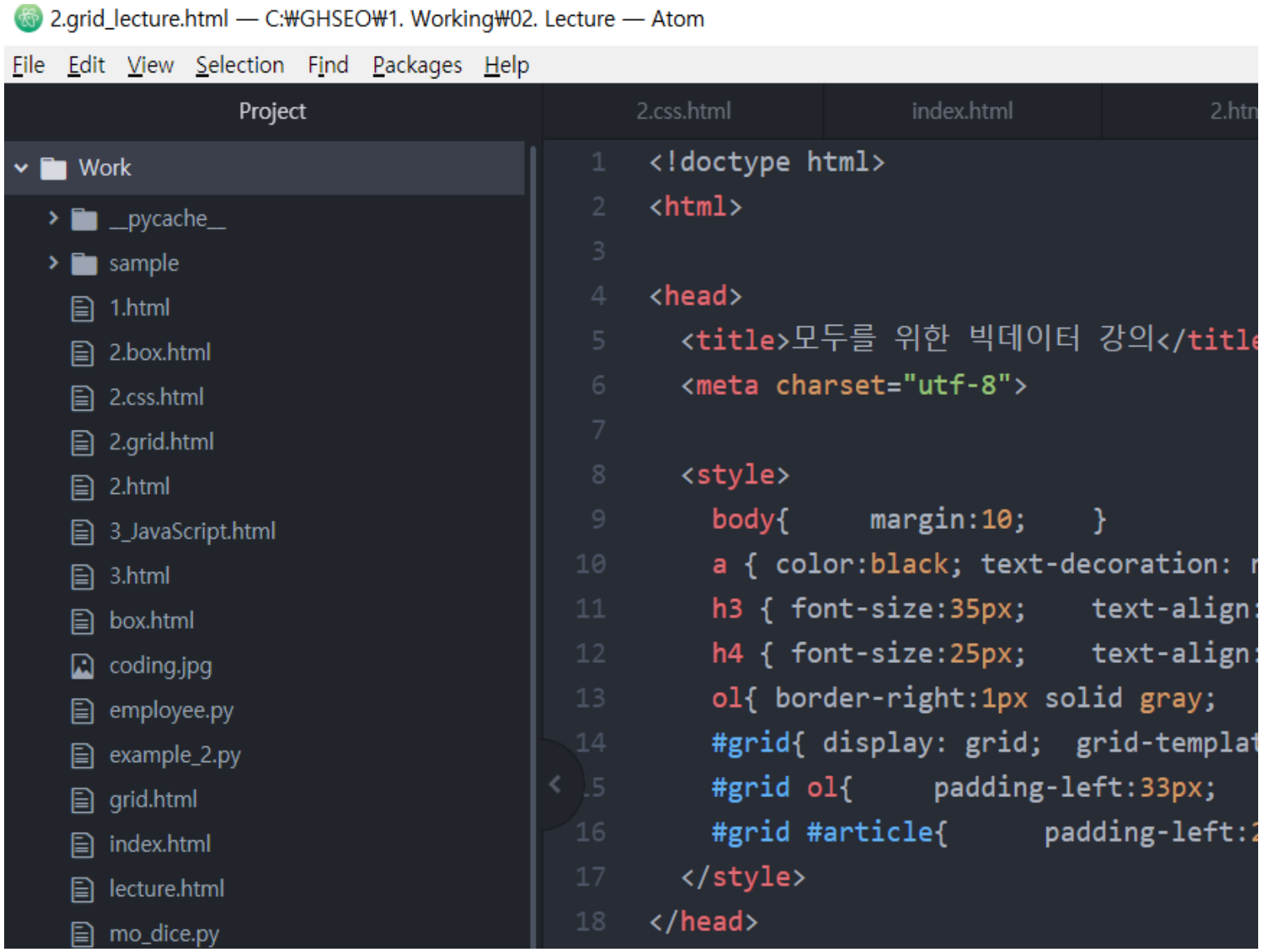
Bitnami manager, atom

Atom

manager-windows

- **C:₩Users₩ ~~₩ML_01**





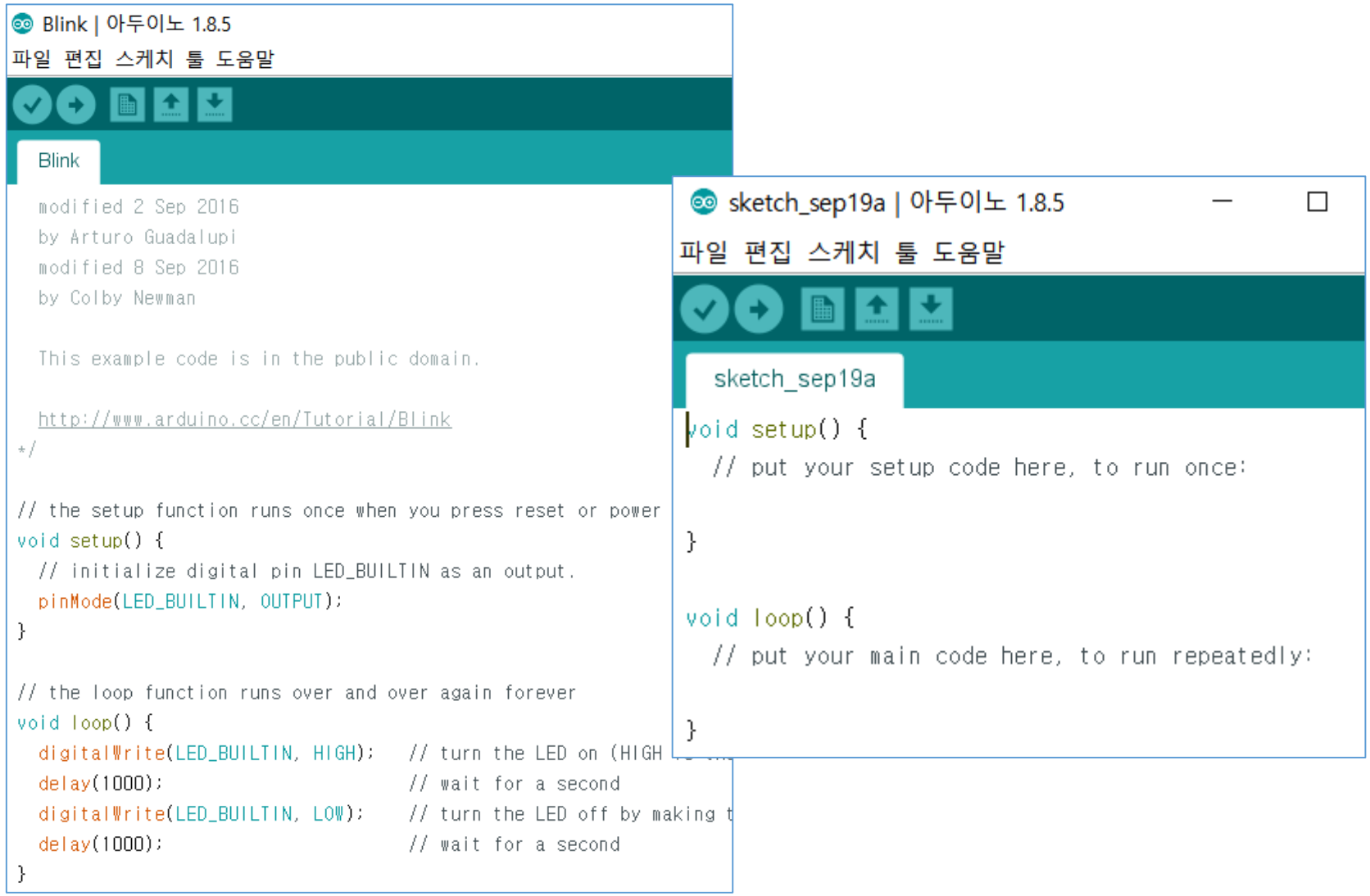
Source : 411 Program_install.pptx

arduino

Arduino

- **C:₩Users₩~~₩Documents₩Arduino**

test_arduino.ino



Source : 411 Program_install.pptx

4장. AI 머신러닝 프로그램을 직접 시도해 한 걸음 나아가 보자.

벡터, 행렬 & 통계	데이터 마이닝 & AI	머신러닝 (Machine Learning)	빅데이터 오픈소스 프로그램 이해
- 벡터, 통계, 프로그램 배우는 이유 • 벡터 • 행렬 • 차원 • 머신러닝에서 사용하는 관련 지식 - 통계, 및 알고리즘 • 통계 개념과 활용 방법 • 평균의 함정, 분산의 의미 • 기술 통계, 공분산/상관계수 • 요인분석 • 회귀분석, 로지스틱, 군집 • 가설검정, • 계절지수 • 예측값 y hat은 왜, 맞지 않을까	- 데이터 마이닝 • 데이터 마이닝 개념 • 상용 솔루션의 활용 • 빅데이터 분석 알고리즘 개념 • 모델의 성능평가 : ROC (TPR – FPR : 진짜 – 잘못 평가) - AI 개념 • AI 개념 • AI 계산주의, 연결주의 (간단한 기능을 복잡 연결) • XOR	- 머신러닝 • Deep Learning • LR (미분과 코스트/손실 함수, 경사하강) • Logistic Classification • SoftMax • CNN • Xavier • RNN • CNN+RNN) • 자연어 처리 • 자율주행차	- 프로그램 이해 • 컴퓨터 프로그램 개요 • Internet / DNS / URL / • OSI 7 layers • WEB & HTML • WEB System - 빅데이터 오픈소스 (설치 및 핵심 개념의 Code) • R • Shiny • Python / numpy / pandas • TensorFlow / TensorBoard • 학습 방법 소개 • 참조 Site : 구글 ANALYTICS.google.com • 컴퓨터 프로그램 학습 요령 (실제 활용 방안 : Frond-end, Back-end별 활용)

Source : 600. BDA_Book_4장_p123.pptx

4장. 빅데이터 지능화 아는 것 만큼 보인다.

벡터, 행렬 & 통계	데이터 마이닝 & AI	머신러닝 (Machine Learning)	빅데이터 오픈소스 프로그램 이해
- 벡터, 행렬 및 미적분 • 벡터 • 행렬 • 차원의 저주 - 통계, 및 알고리즘 • 통계, 왜 배워야 하는가 • 통계 개념과 활용 방법 • 평균의 함정, 분산의 의미 • 기술 통계, 공분산/상관계수 • 요인분석 • 회귀분석, 로지스틱, 군집 • 가설검정, • 계절지수 • 예측값 y hat은 왜, 맞지 않을까	- 데이터 마이닝 • 데이터 마이닝 개념 • 상용 솔루션의 활용 • 빅데이터 분석 알고리즘 개념 • 모델의 성능평가 : ROC (TPR – FPR : 진짜 – 잘못 평가) - AI 개념 • AI 개념 • AI 계산주의, 연결주의 (간단한 기능을 복잡 연결) • XOR	- 머신러닝 • Deep Learning • LR (미분과 코스트/손실 함수, 경사하강) • Logistic Classification • SoftMax • CNN • Xavier • RNN • CNN+RNN) • 자연어 처리 • 자율주행차	- 프로그램 이해 • 컴퓨터 프로그램 개요 • Internet • WEB & HTML • WEB System - 빅데이터 오픈소스 (설치 및 핵심 개념의 Code) • JAVA 프로그램 이해 • R • Shiny • Python / numpy / pandas • TensorFlow / TensorBoard • 학습 방법 소개 • 참조 Site : 구글 ANALYTICS.google.com • 컴퓨터 프로그램 학습 요령 (실제 활용 방안 : Frond-end, Back-end별 활용)

Source : 600. BDA_Book_4장_P4

일반	430. web_cloud L_Video_clip_UseCase_DT_ERP_PLM	00. KIR_BoK 122. DT_Methdology_main	
개념 설명 BDA Book	320. ML_algorithm_code_summary 510. BDA_Statistics_quick_review L_울산_BDA_CEA.pptx L_울산_ERP_SCM_PLM_MOM	0_Program_code_list (code & keyword) 0. Program_JPR 600. BDA_Book_3장 600. BDA_Book_4장_p123 / p4	L_Program_Code_total_summary.pptx (p 207)
R	312. DA_R2_code 420. PL_R_graph	430. System_demo_R_Python	• sta_1~ .r - C:₩R₩work_r • sh_~.r - C:₩R₩work_r₩NP_Shiny
Shiny	312. DA_Shiny_16_tool.pptx 420. PL_Shiny.pptx		
Python	ML_python_demo_2.ipynb (Jupyter) 321. ML_Concept_Basic 322. ML_python_demo 412. Python_detail / 413. Python_tf_tb	430. Class_Instance.pptx	• exe_1.py (Spyder) • C:₩Users₩gwihy₩ML_01
JAVA / web	430. Java_etc		

Source : 0_Program_Code_list.pptx

Program Code 구조

L_Program_Code_통계_R_Shiny_Python.pptx (p 252)

L_Program_Code_total_summary.pptx (p 207)

430. System_demo_R_Python (p 41)

(510) Statistics quick review for BDA

- t 검정, 분산분석, 타당성과 신뢰성
- 요인분석, 회귀분석
- 군집분석

(311) Statistics Basic

- Statistics basic
- 통계분석 활용

(321) ML concept detailed

- Neural network, xor, backpropagation, relu, weight : rbm, Xavier , overfitting : regularization, dropout, ensemble
- optimizer : adam

(322) ML Algorithm (p1~p3)

- Linear Regression
- Logistic Classification
- Softmax
- AI, ML & Deep Learning
- CNN, RNN

510. BDA_Statistics_quick_review (p 45)

311. DA_Statistics. theory (p120)
312. DA_R1_concept.pptx
312. DA_R2_code.pptx (p 68)
312. DA_Shiny_16_tool.pptx (p 66)

320. ML_algorithm_code_summary (p 75)

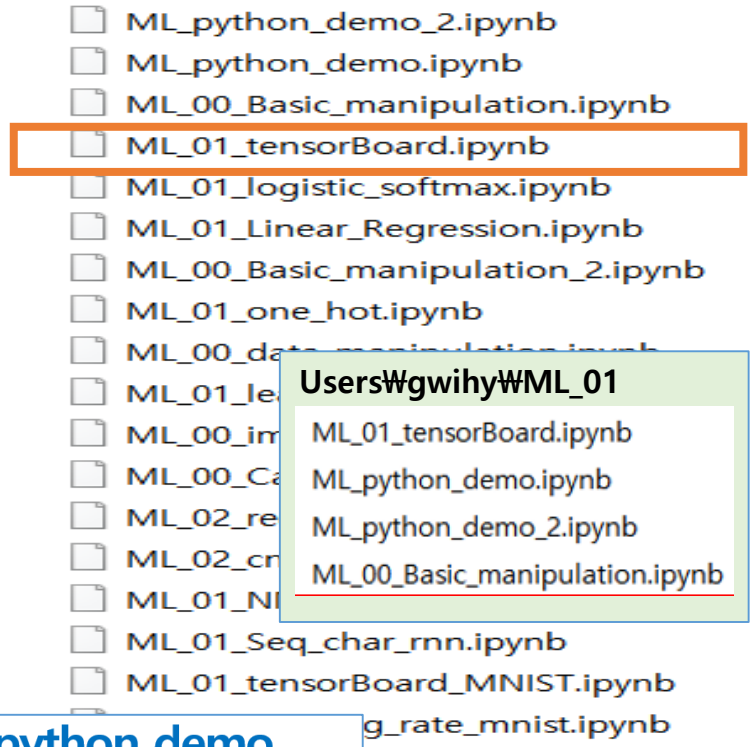
321. ML_concept_basic.pptx (p 74)
321. ML_concept_detailed.pptx
322. ML_Algorithm_p1_lm_softmax
322. ML_Algorithm_p2_cnn_xor ,
322. ML_Algorithm_p3_rnn

* 413. Python_tf_tb.pptx

322. ML_python_demo

(322) ML python demo

- python / matplotlib /tensorflow
- numpy / tensorboard
- ReLu/ Xavier / droupout / adam



Python Code (C:₩Users₩gwihy₩ML_01)

ML_01_tensorBoard.ipynb	2019-08-06 .
ML_python_demo.ipynb	2019-08-04 .
ML_python_demo_2.ipynb	2019-07-02 .
ML_00_Basic_manipulation.ipynb	2019-06-07 .
ML_01_logistic_softmax.ipynb	2019-06-03 .
ML_01_Linear_Regression.ipynb	2019-06-03 .
ML_00_Basic_manipulation_2.ipynb	2019-06-03 .
ML_01_one_hot.ipynb	2019-06-03 .
ML_00_data_manipulation.ipynb	2019-06-03 .
ML_01_learning_rate_x_data.ipynb	2019-06-03 .
ML_00_imshow_Image.ipynb	2019-06-03 .
ML_00_Cast.ipynb	2019-06-03 .
ML_02_regression_housing.ipynb	2019-06-03 .
ML_02_cnn_MNIST.ipynb	2019-06-03 .
ML_01_NN_MNIST.ipynb	2019-06-03 .
ML_01_Seq_char_rnn.ipynb	2019-06-03 .
ML_01_tensorBoard_MNIST.ipynb	2019-06-03 .
ML_01_learning_rate_mnist.ipynb	2019-06-03 .

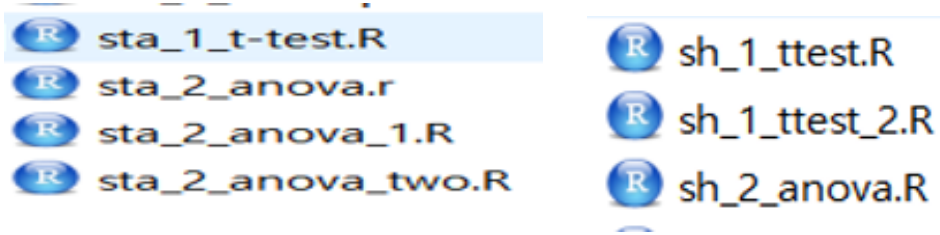
ML_12_4_rnn_long_char.ipynb	2019-04-29 .
ML_13_3_save_mnist_restore.ipynb	2019-04-29 .
ML_12_5_rnn_stock_prediction.ipynb	2019-04-29 .
ML_12_3_char_seq_softmax.ipynb	2019-04-29 .
ML_12_1_hello_rnn.ipynb	2019-04-29 .
ML_12_0_rnn_basic.ipynb	2019-04-29 .
ML_11_05_mnist_cnn_ensemble_layer....	2019-04-29 .
ML_11_04_mnist_cnn_ensemble.ipynb	2019-04-29 .
ML_11_03_mnist_class.ipynb	2019-04-29 .
ML_11_02_mnist_deep_cnn.ipynb	2019-04-29 .
ML_11_0_cnn_basic.ipynb	2019-04-29 .
10logistic2.csv	2019-04-29 .
ML_03_UseCase_scm.ipynb	2019-03-24 .
air_pax2.csv	2019-03-13 .
air_pax.csv	2019-03-13 .

프로그램 코드

R

- sta_1~ .r - C:₩R₩work_r
- sh_~.r - C:₩R₩work_r₩NP_Shiny

Shiny



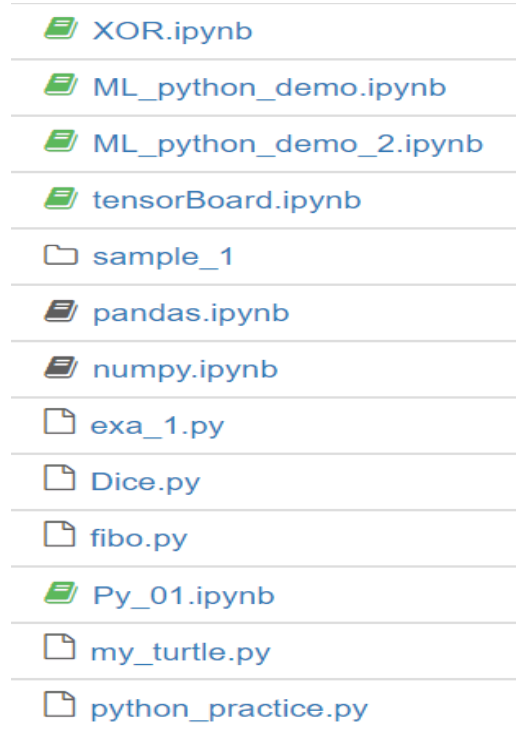
Python

- exe_1.py (Spyder)
- C:₩Users₩gwihy₩ML_01
- python – numpy – pandas / Jupyter- Spyder

XOR

random / matplotlib graph
Linear Regression
Multi-variable linear regression with air-pax
Logistic / SoftMax / CNN~RNN

NN for MNIST
CNN
RNN



JAVA / web

Source :

BDA 프로그램 학습 절차 / Java – Python – R 비교 / 프로그램 설치

- **BDA Contents list (통계, R/Shiny, Web/Cloud)**
- **Python Basic/ TensorFlow/ TensorBoard/ NumPy /pandas/ Matplotlib**
- **Reference**

Source :

별첨 자료

벡터, 행렬 & 통계	데이터 마이닝 & AI	머신러닝 (Machine Learning)	빅데이터 오픈소스 프로그램 이해

- **Program Code/System : R, Shiny , Python, Pandas , Tensorflow / TensorBoard**

312. DA_R2_code.pptx + 312. DA_Shiny_16_tool.pptx + ML_01

- L_Program_Code_executive_summary.pptx (p 207)
- L_Program_Code_통계_R_Shiny_Python.pptx (p 252)
- 430. System_demo_R_Python
  - ✓ R Concept / **R code / Shiny 시연**
  - ✓ ML Concept / **Python code 시연**
  - ✓ TensorBoard

ML_01_tensorBoard.ipynb
ML_python_demo.ipynb
ML_python_demo_2.ipynb
ML_00_Basic_manipulation.ipynb

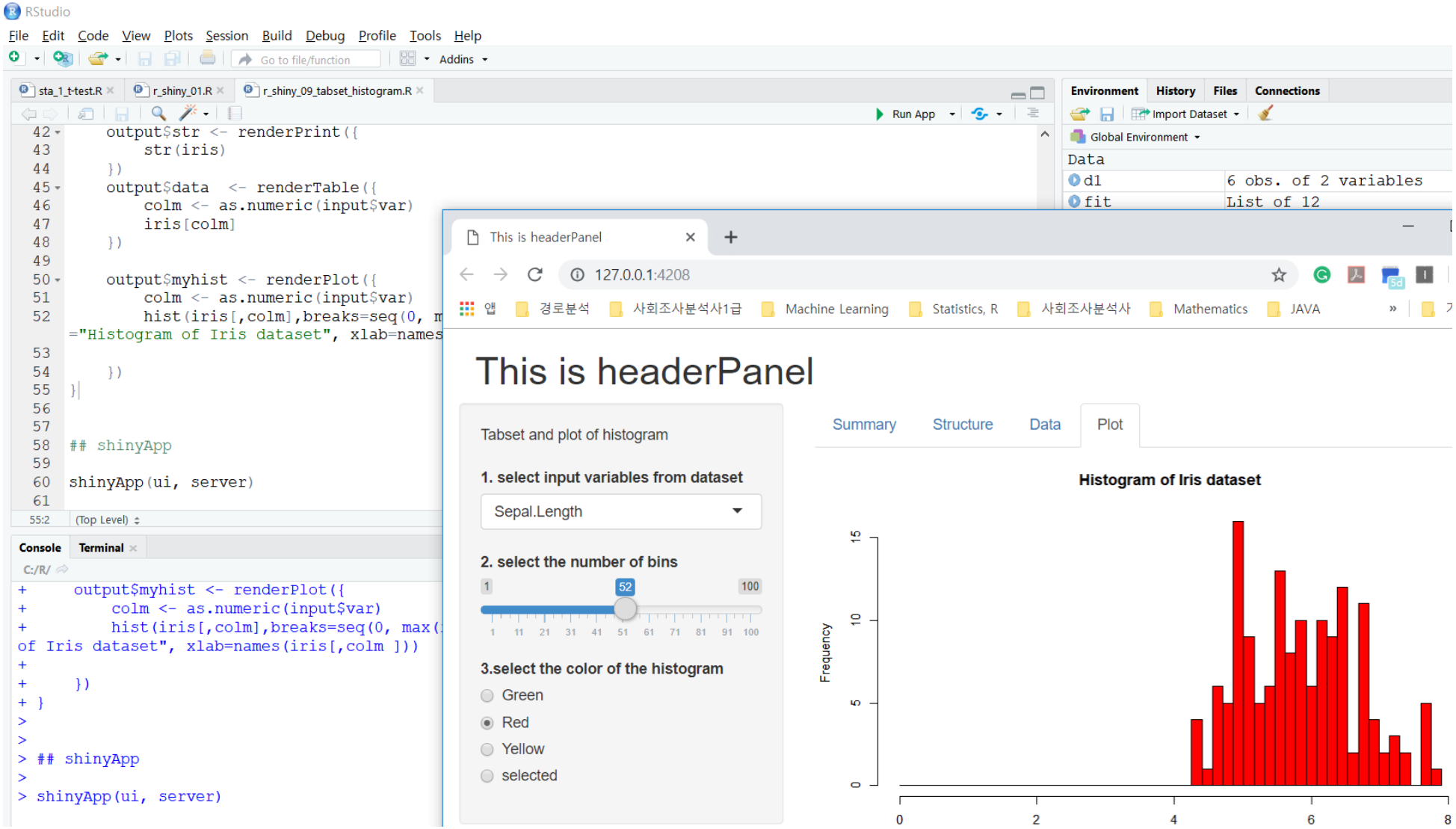
- **430. web_cloud.pptx**
- **IT**
- **430. Class_Instance.pptx**

Source :

- 사전 지식 및 BDA Contents list
- R/Shiny
- AI, ML
- Web/Cloud

Source :

R demo



Source :

R code

C:₩R₩work_r

1 기술 통계 Lecture.R

주요 통계분석 R 활용

- 공분산
- 독립 t-Test (일표본, 대응표본, 독립표본)
- ANOVA (one-way, two-way, MANOVA)
- 요인분석 (PCA/FA)
- 신뢰도 분석
- 상관분석
- 교차분석
- 회귀 / 다중 회귀분석
- 로지스틱
- 군
- ...

(312) R을 이용한 통계분석 code

- t 검정, 분산분석, 타당성과 신뢰성
- 요인분석, 회귀분석
- 군집분석

2 주요 통계분석

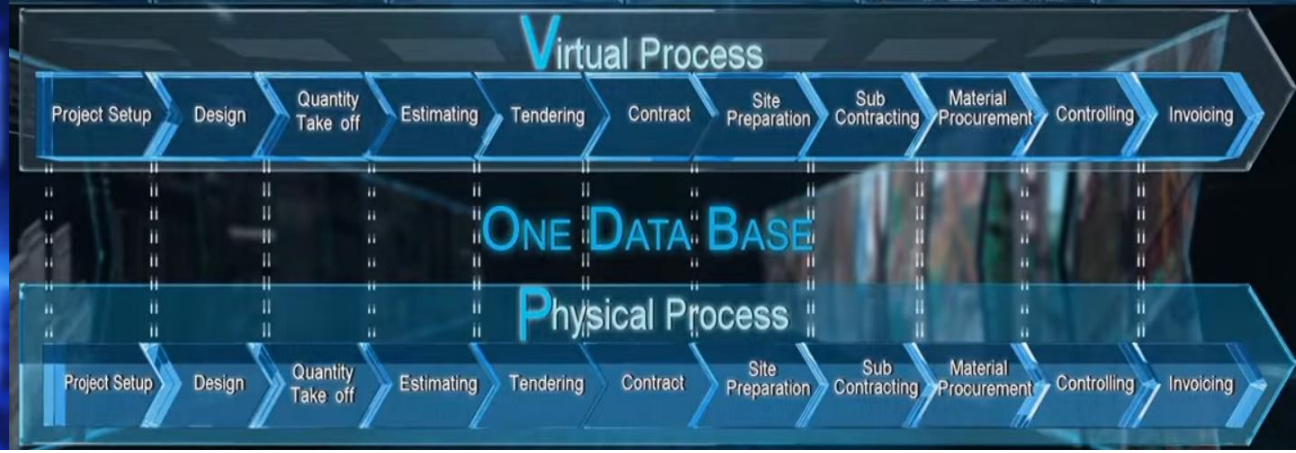
- sta_0_descriptive.R
- sta_1_t-test.R
- sta_2_anova.r
- sta_2_anova_1.R
- sta_2_anova_two.R
- sta_3_easy_princomp.R
- sta_3_fa_eigenvalue.R
- sta_3_factanal.R
- sta_3_pca_fa.r
- sta_3_prcomp.R
- sta_3_prcomp_lm.R
- sta_4_alpha.R
- sta_5_table.R
- sta_6_cross_table.R
- sta_7_dummy.R
- sta_7_simple_lr.R
- sta_8_logistic.R
- sta_9_hierarchical_clustering.R
- sta_9_hierarchical_clustering_2.R
- sta_9_kmeans.R
- sta_9_kmeans_2.R
- sta_10_non_parametric.R
- sta_10_wilcox_signed.R
- sta_11_discriminant.R
- sta_11_discriminant_iris.R
- sta_11_discriminant_wine_ldahist.R
- sta_11_lda_plot.R

3 활용 아이디어

- 탐색 : 기술통계로도 많은 것을 분석
- 아이디어의 중요성
- 3 why : 계절 지수

- 품질 분석
- 개선 효과 측정
- 원인 분석
- 수요 예측
- 융합

Source : 312. DA_R2_code



(312) 통계분석 Shiny web

- 통계 분석기법의 Shiny web 활용
- Shiny Source code

312. DA_Shiny_16_tool.pptx

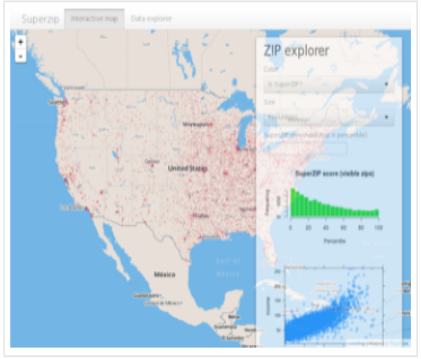
Shiny concept & Shiny User Showcase

기본형태	변수들이 입력된 상태

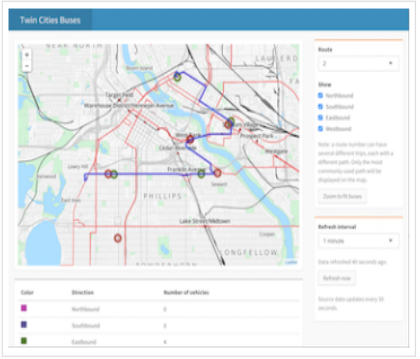
ui<-fluidPage()
server<-function(input, output) {}
shinyApp(ui=ui, server=server)
	ui<-fluidPage(
	sliderInput(inputId="num_1",
	label="숫자를 고르세요",
	value=25, min=1, max=80, step=1),
	plotOutput("hist")
	)
	server<-function(input, output) {
	output$hist <- renderPlot({
	title <- "80 random normal values"
	hist(rnorm(input$num_1),main=title)
	})
	}
	shinyApp(ui=ui, server=server)

Interactive visualizations

Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts.



SuperZip example

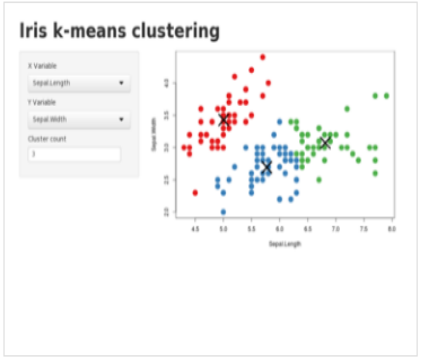


Bus dashboard

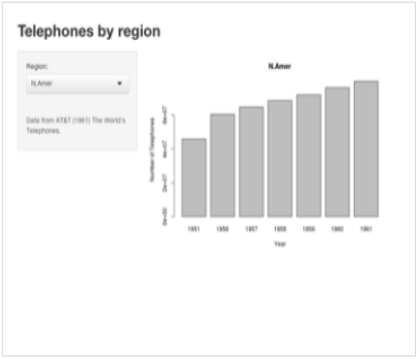
Movie explorer

Start simple

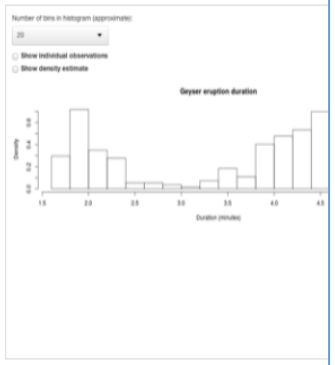
If you're new to Shiny, these simple but complete applications are designed for you to learn from.



Kmeans example



Telephones by region



Faithful

Source : https://jonnyomme.blogspot.com/2017/02/r-programming-shiny-shinyapp-1.html https://shiny.rstudio.com/gallery/

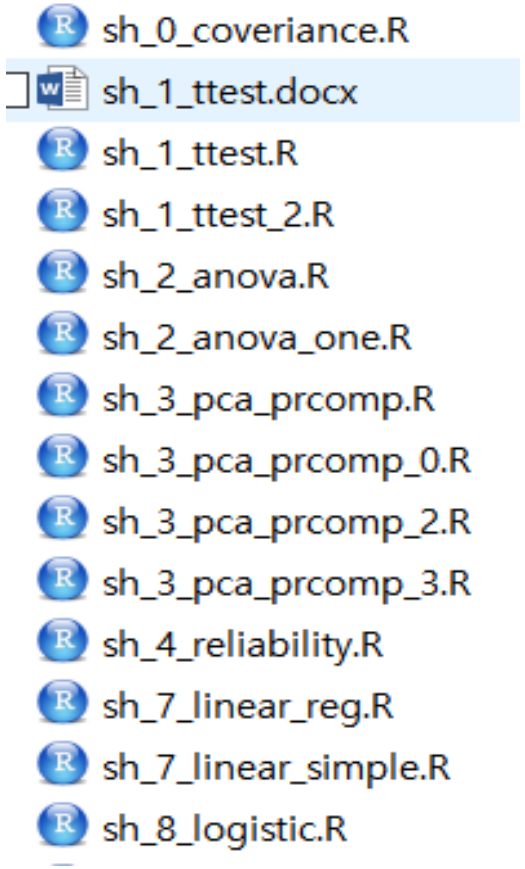
Shiny code

C:₩R₩work_r₩NP_Shiny

(420) Shiny

- Basic
- reactive
- dashboard

Shiny web



Shiny 확장 가능성

Source : 420. PL_Shiny.pptx sh_1_ttest_2.R ~

분석 기법

공분산	변수_X	품질 불량
	변수_Y	생산성
t test	검정 변수	작동 시간
	검정 값	500
대응표본	대응 변수_B	개선 전
	변수_A	개선 후
독립표본	검정 변수	작동 시간
	집단 변수	제조사(A,B)

anova	종속 변수	생산성
	요인	기계 제조사
two-way anova	모수 요인	제조사, 작업 시간
manova	종속 변수	생산량, 납기준수
	모수 요인	제조사, 작업 라인
pca	변수	var_1, ... var_n
	요인 추출	주성분
	요인 회전	베리맥스 쿼티맥스

신뢰도	항목	var_1 ~ var_n
상관분석	변수	요인_1, ...
교차분석	행	생산성 A 등급 건수
	열	제작 라인
회귀분석	종속 변수	생산량
	독립 변수	교육시간
로지스틱	종속 변수	품질등급 (A, B, C)
	공변량	제조사, 제작 라인

Source :

통계 분석 활용

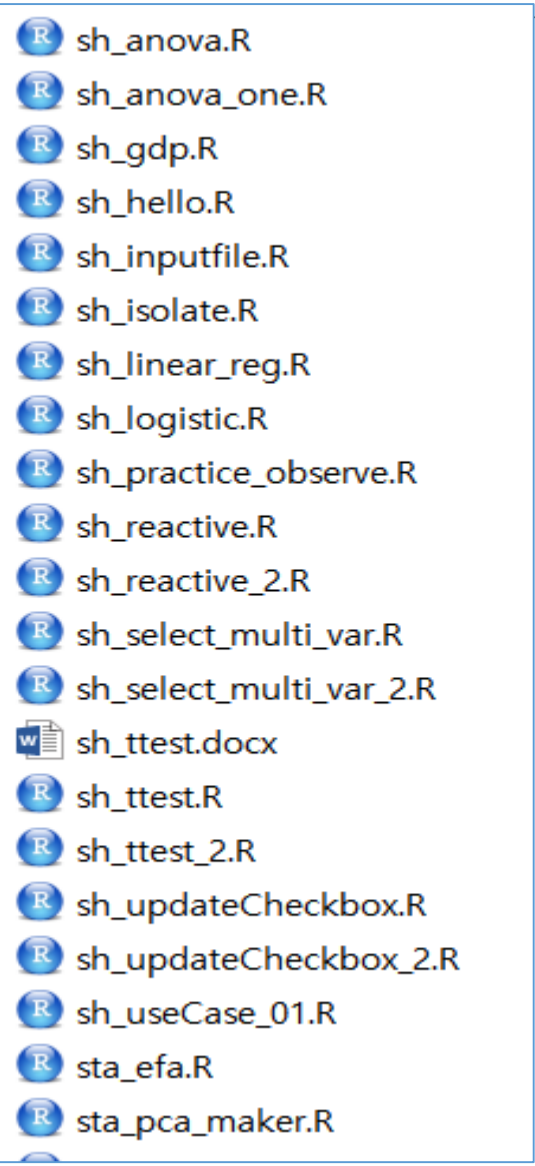
분석 기법

0. 공분산
1. 독립 t-Test (일표본, 대응표본, 독립표본)
2. ANOVA (one-way, two-way, MANOVA)
3. 요인분석 (PCA/FA)
4. 신뢰도 분석
5. 상관분석
6. 교차분석
7. 회귀 / 다중 회귀분석
8. 로지스틱
9. 군집분석
10. 비모수 통계
11. 판별분석

* 경로분석 / 구조분석

결과 해석 및 Plot

- 상관계수
- 가설 검정, p value
- 가설 검정, p value
- 차원 축소, 요인 도출
- 상관계수
- Cronbach alpha
- 회귀식
- 판정값
- plot : histogram / scree plot /



Source :

공분산

Choose CSV File

Browse... data-01-test-score_2.csv

Upload complete

Header

Separator

Comma

Semicolon

Tab

Quote

None

Double Quote

Single Quote

Data View | Analysis

Data Summary

```
       x1              x2              Min.
 Min.   :47.00   Min.   :46.00   Min.
 1st Qu.:73.00   1st Qu.:73.00   1st (
 Median :79.00   Median :82.00   Media
 Mean   :79.04   Mean   :79.48   Mean
 3rd Qu.:88.00   3rd Qu.:89.00   3rd (
 Max.   :96.00   Max.   :98.00   Max.
```

Data Structure

```
'data.frame':   25 obs. of  4 variable
 $ x1: int  73 93 89 96 73 53 69 47 87
 $ x2: int  80 88 91 98 66 46 74 56 79
 $ x3: int  75 93 90 100 70 55 77 60 9
 $ y : int  152 185 180 196 142 101 14
```

Data Table

x1	x2	x3	y
73	80	75	152
93	88	93	185

Numer of bins: 2 50

1 6 11 16 21 26 31 36 41 46 50

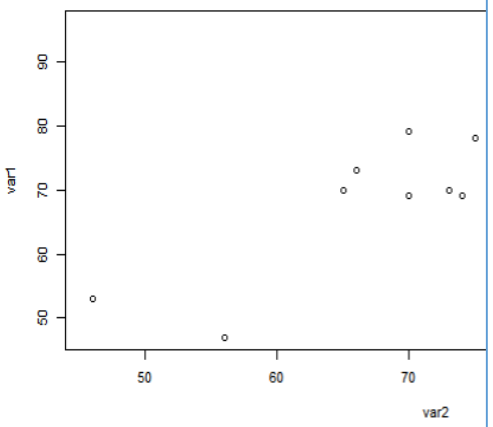
Please Select a Numerical Variable

x1

Please Select a Numerical Variable

x2

Data View | Analysis



Key summary statistics

The observed sample statistics were:

mean	standard_deviation	standard_error
79.04	12.28	2.46
79.48	12.50	2.50

Covariance & Correlation

The observed covariance:

[1] 138.3967

Correlation :

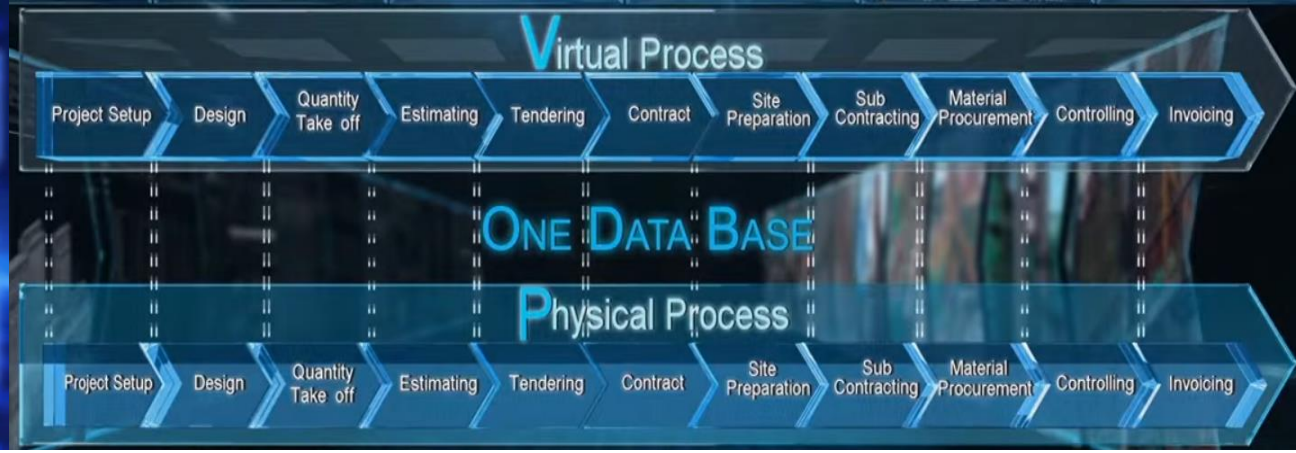
[1] 0.9013629

gwihyeonseo@naver.com

Source : [sh_coveriance.R]

- 사전 지식 및 BDA Contents list
- R/Shiny
- AI, ML
- Web/Cloud

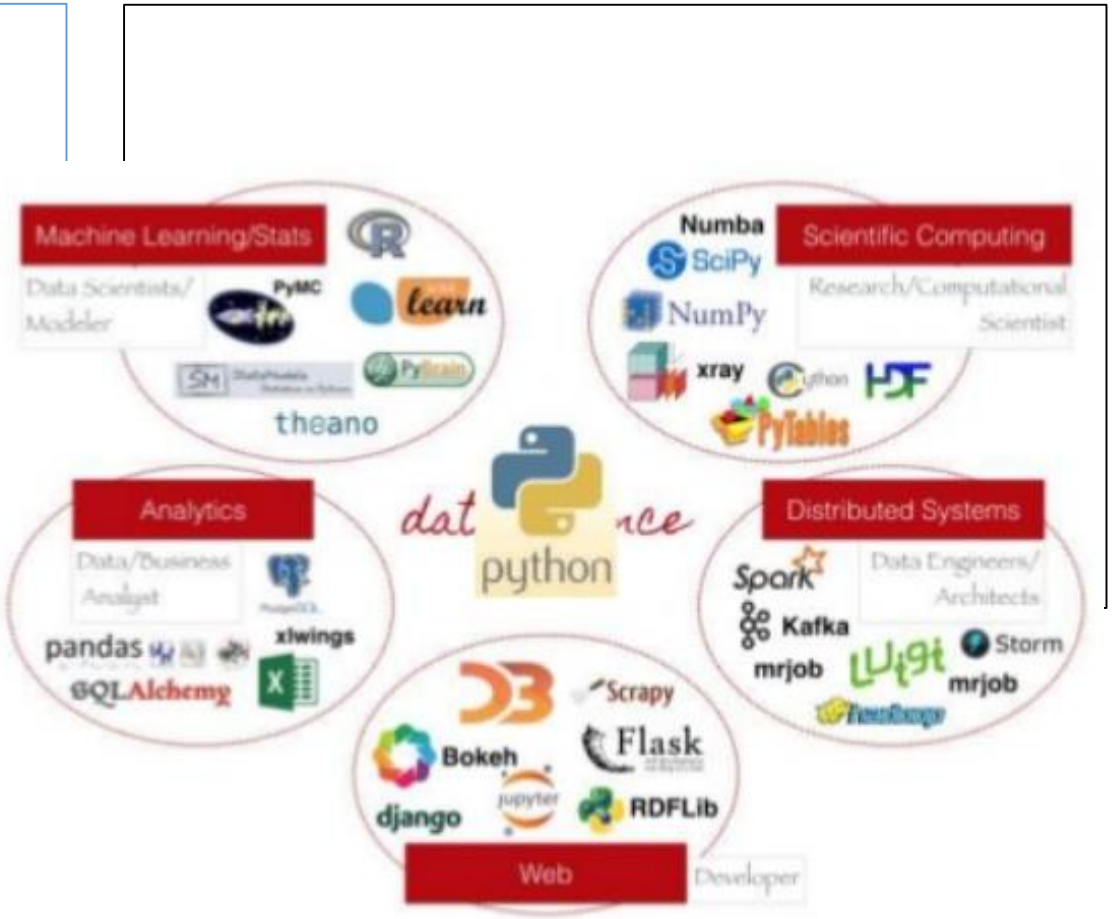
Source :



(321) ML Basic Concept

- Python, numpy, pandas, Matplotlib
- AI, xor, ML, Deep Learning
- CNN, TensorFlow, TensorBoard
- RNN
- 활용

Python은 함께 사용할 여러 Library 와 버전 문제가 되지 않도록 환경을 구분하여 운영하는 것에 필요한 배포판 Anacoda 를 이용 설치



Source : https://www.slideshare.net/teoliphant/python-for-data-science-with-anaconda

필요한 프로그램 설치

Source : error : https://rstudio-pubs-static.s3.amazonaws.com/242584_29bdf274c291408090962b3e860a299e.html

Python code

C:₩Users₩gwihy₩ML_01

Summary : 시스템이 되지 않을 경우 ppt 자료

322. ML_python_demo

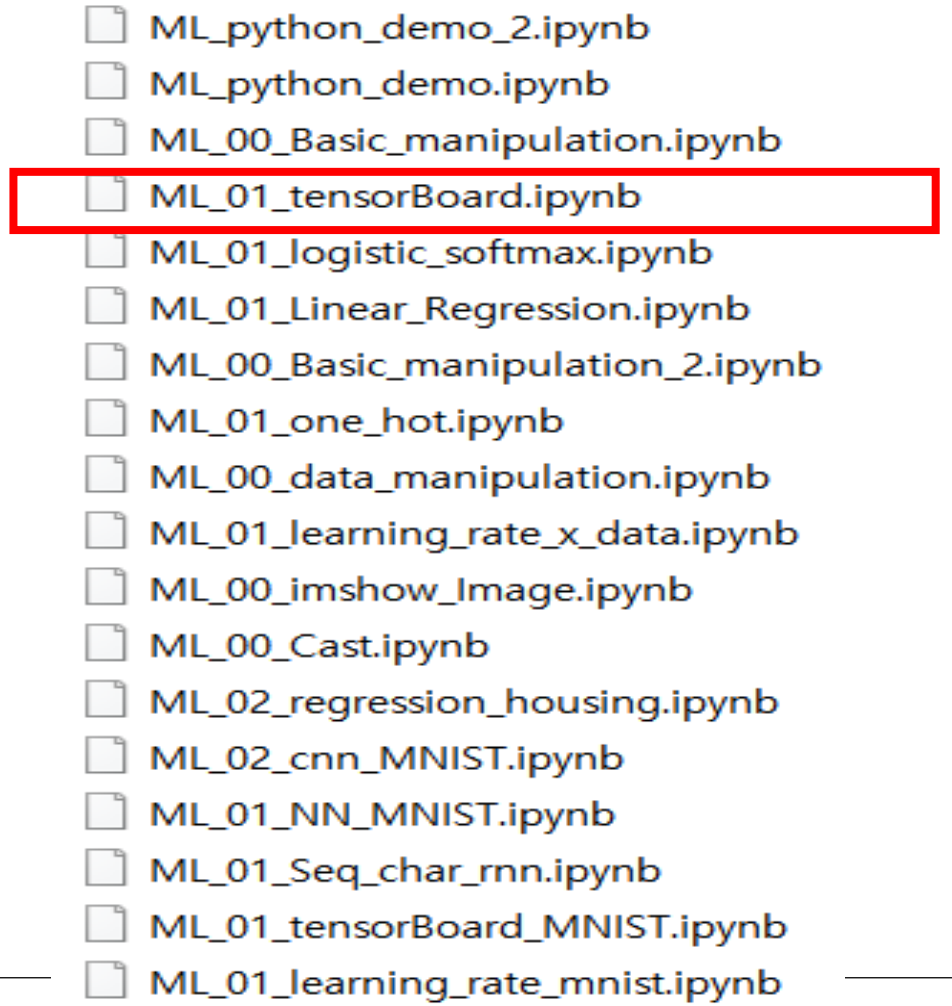
(322) ML python demo

- python / matplotlib /tensorflow
- numpy / tensorboard
- ReLu/ Xavier / droupout / adam

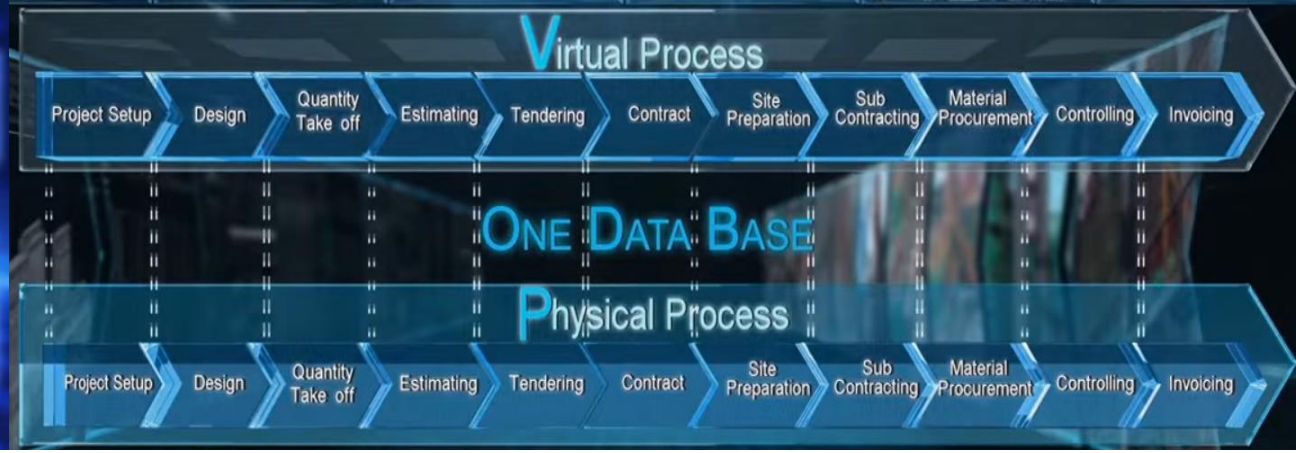
- Linear Regression
- Logistic / Softmax
- CNN
- RNN
- TensorBoard (anaconda/py2.7)
- 통계/수학적 기법(excel) vs ML

ML_excel_Hanll.xls

Jupyter Notebook



Source : 322. ML_python_demo.pptx



(322) ML python demo

- python / matplotlib /tensorflow
- numpy / tensorboard
- ReLu/ Xavier / droupout / adam

430. Class_Instance

- **JAVA**
- **Python**
- **Arduino**

Source :

- ROC 를 통해 모델의 성능평가 / 신뢰도 검증을 한다.

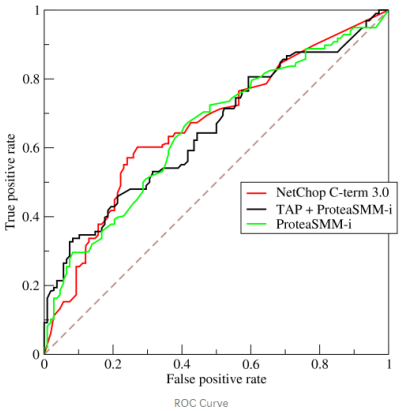
분석모형의 성능 평가란 우리의 예측과 실제 결과가 어떤 오차를 갖는지를 확인하고 그것이 유의미한 데이터인지 판별하는 것이다. 학습은 이 오차를 줄여가는 것 이다. 이를 위해 보통 ROC Curve를 그리고 AUC (the Area Under a ROC Curve)로 판단한다.

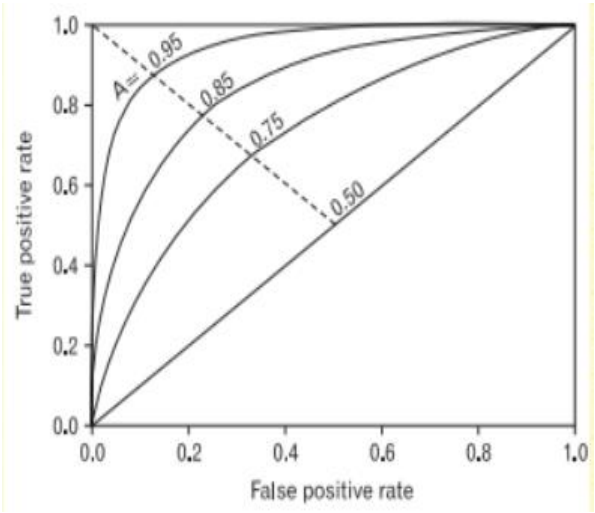
ROC curve :Receiver Operating Characteristic (ROC)는 Binary Classifier System (이진 분류 시스템)에 대한 성능 평가 기법이다. AUC = AUROC (the Area Under a ROC Curve) : ROC 커브의 밑면적을 구한 값이 바로 AUC. 이 값이 1에 가까울수록 성능이 좋다. 곡선이 굽어질수록 AUC가 넓어져 1로 수렴하므로 더 정확한 모델이다.

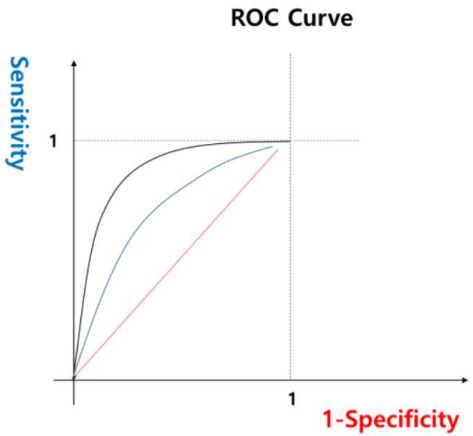
- **ROC curve의 X = 틀린 것을 맞다고 할 확률 = 정상인을 환자로 잘 못 평가 = 1- specificity = False Positive Rate = c / (c + d) = 얼마나 오류가 있는지 평가**
- **Y = 맞은 것을 맞다고 할 확률 = 환자를 환자로 잘 평가 = True Positive Rate = a / (a + b)**

예측(Predict) 은 분석 모형이나 우리가 판단하는 요인을 가지고 한다.

		Actual + (진짜환자)	Actual - (진짜정상인)
Predict	+	True Positive (a)	False Positive (c)
	-	False Negative (b)	True Negative (d)







Source : 통계 R 학습_roc.xlsx

- ROC 성능평가의 활용 사례

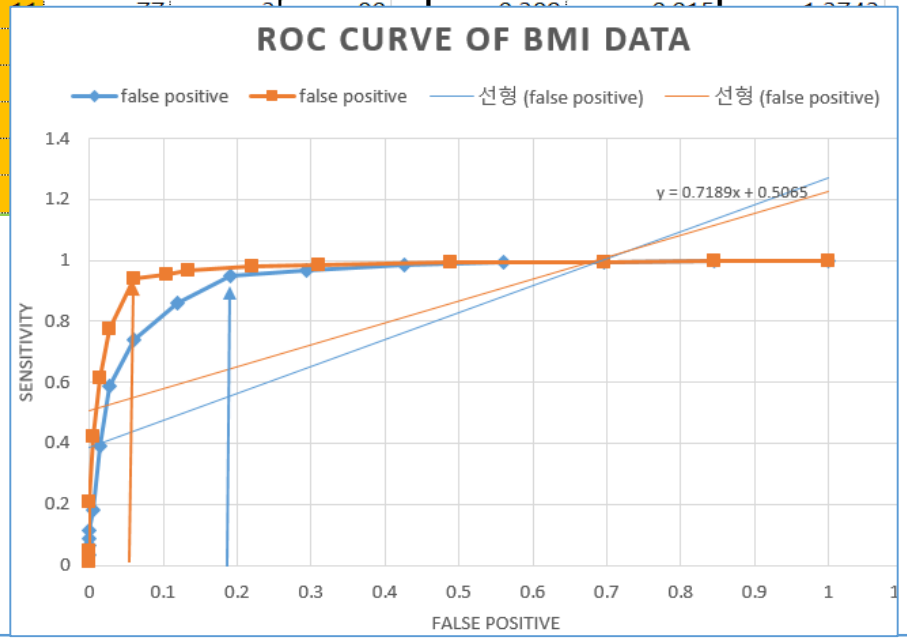
TPR과 FPR은 서로 반비례적인 관계에 있다. TPR과 FPR의 여러 가지 상황을 고려해서 성능을 판단해야 하는데, 이것을 한눈에 볼 수 있게 한 것이 바로 ROC 커브이다. 예를 들어 걸릴 확률은 매우 낮지만, 치사율이 극히 높은 병은 민감도가 높아야 하고, 걸릴 확률은 높지만, 위험성이 거의 없는 경우는 민감도가 낮아도 괜찮을 것이다.

- cut off point = Sensitivity + (1 - False Positive Rate) 가 가장 큰 지점이 가장 적절한 분석모형의 판단 기준 포인트 이다.

- ROC의 사용 목적은 분석모형 간의 성능을 비교하거나, 특정 분석 모형에서의 Cut off 을 찾기 위하여 사용한다.

- Sensitivity (민감도) : 진짜 환자를 찾는 수준 평가 (분석 모형의 예측 수준)
  - True Positive Rate (TPR) = a / (a + b)
- Specificity (특이도) : 정상을 정상으로 찾는 수준 평가
  - True Negative Rate(TNR) = d / (c + d)
- Accuracy (정확도) : (a+d)/(a+b+c+d)
- Precision (정밀도) : Positive로 예측한 것 중 실제로 Positive인 비율
- Precision: a / (a + c)

모델 bmi	분류 기준	성능 disease	성능 healthy	sum	sensitivity	false positive	cut off point
24	1	1	52	53	1	1	1.0000
25	2	1	50	51	0.997	0.845	1.1521
26	3	1	46	47	0.995	0.696	1.2982
27	4	3	45	48	0.992	0.560	1.4324
28	5	6	44	50	0.984	0.426	1.5582
29	6	7	35	42	0.968	0.295	1.6729
30	7	33	24	57	0.949	0.190	1.7582
31	8	45	20	65	0.859	0.119	1.7404
32	9	55	11	66	0.738	0.060	1.6783
33	10	74	4	78	0.589	0.027	1.5624
34							
35							
36							
37							
38							
39							



Source : https://chaegoner.tistory.com/111 https://newsight.tistory.com/53

Machine Learning 개요

활용 사례

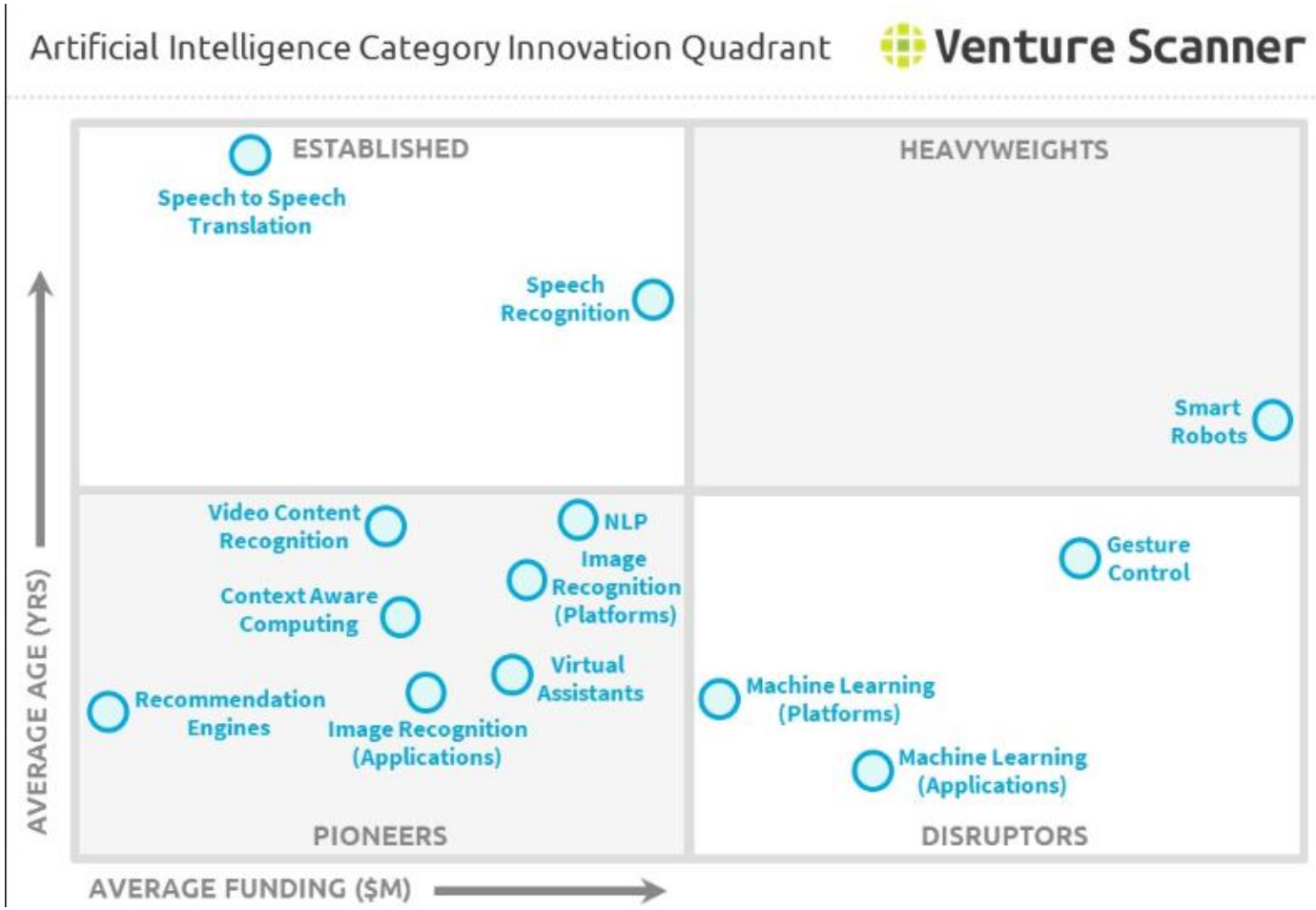
- 활용 범위

주요 개념

- Machine Learning 주요 개념
- 통계적 기법과 차이
- AI, ML, DL, Data Mining, KM

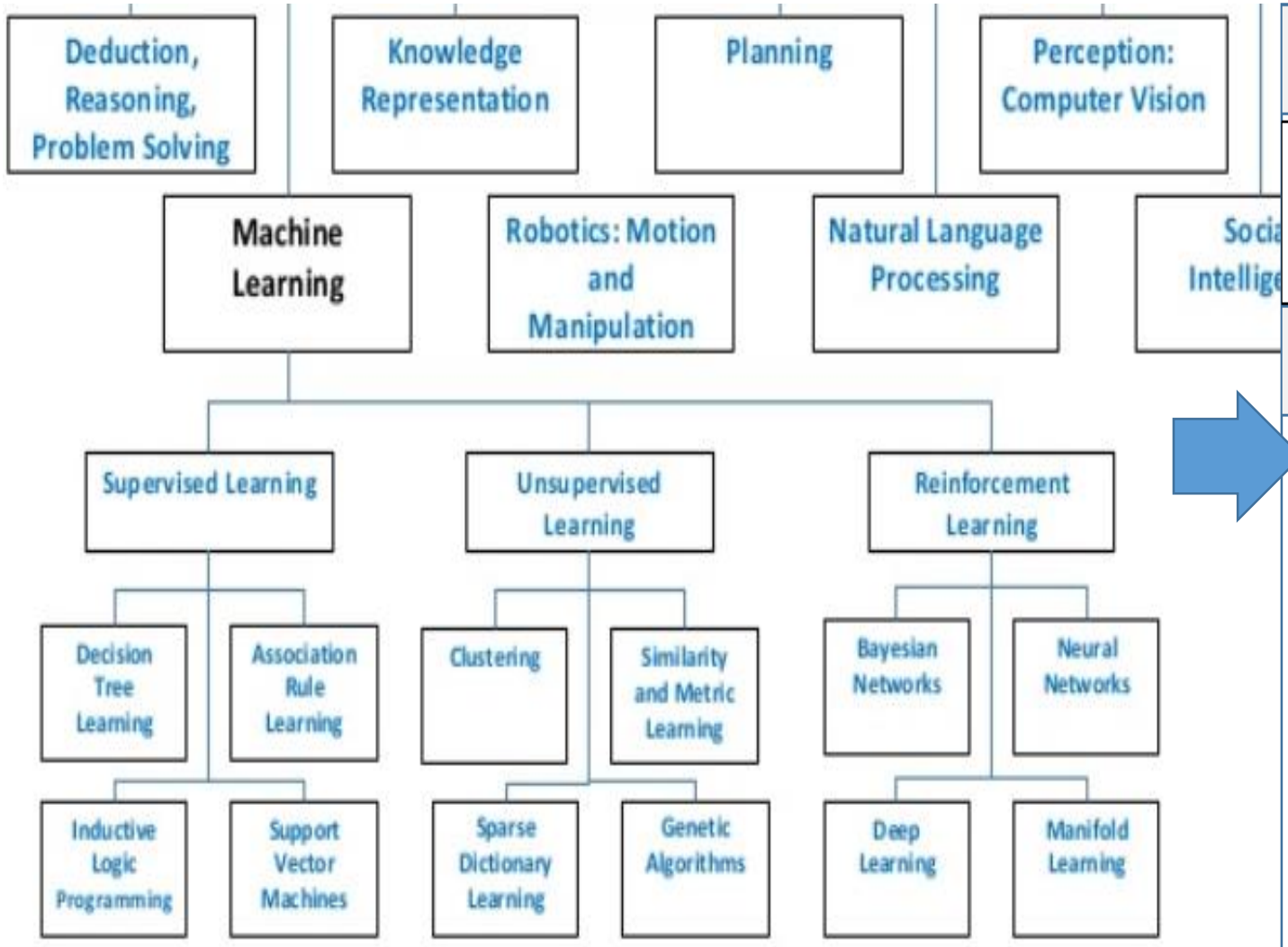
주요 분석

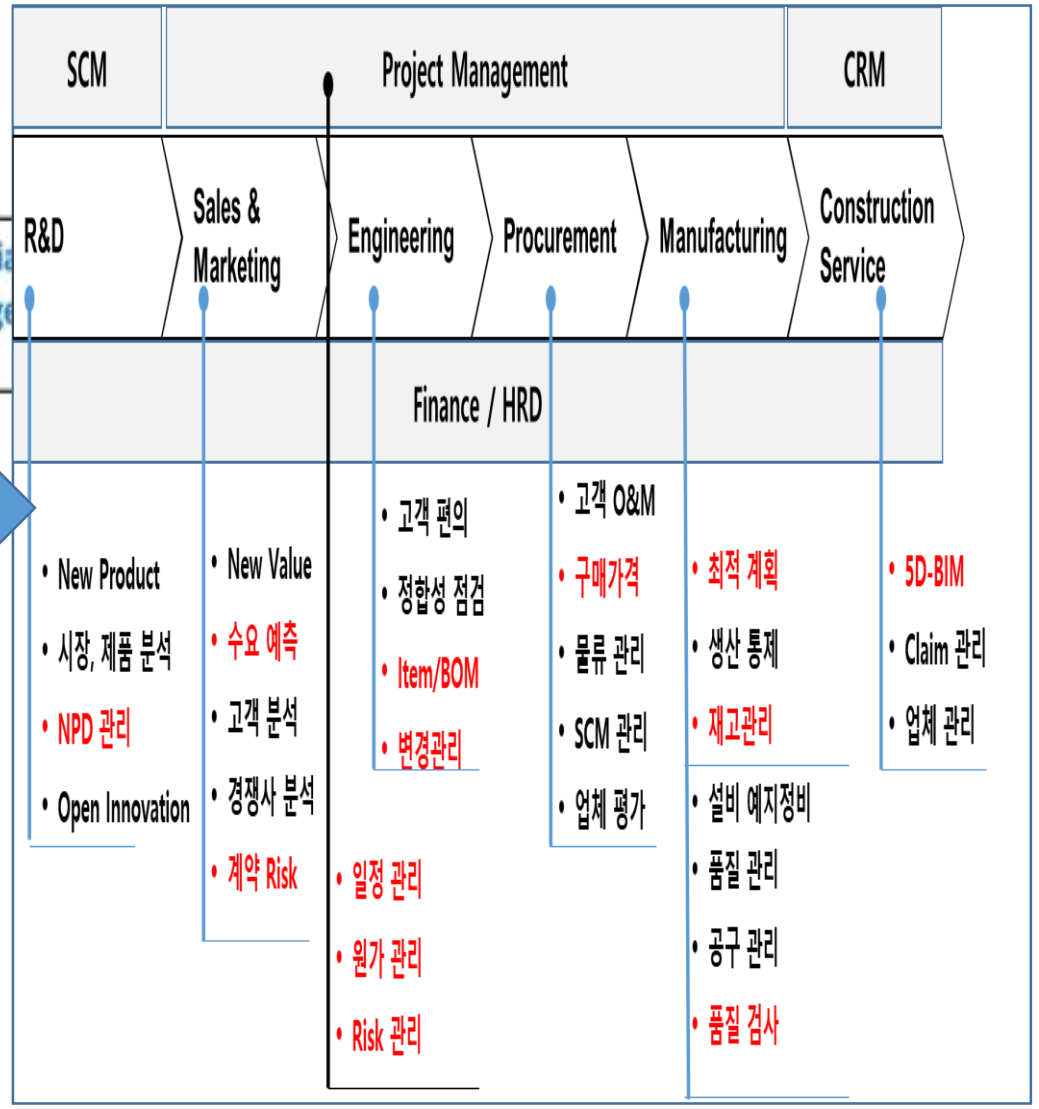
- 분류, 예측,
- 이미지 인식
- Text 인식
- …



Source : https://venturescannerinsights.wordpress.com/2016/11/18/artificial-intelligence-category-innovation-quadrant-q4/

솔루션을 기다리는 것이 아니라, AI / ML 을 이용한 가치 창출을 선점하자.





Source : Deepdive in AIML venture landscape By Ajit Nazre Rahul Garg https://www.slideshare.net/ajitnazre/deepdive-in-aiml-venture-landscape-by-ajit-nazre-rahul-garg

Code 작성 및 실행 : Spyder, Jupyter

Spyder (Python 2.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:₩Users₩gwihy

Editor – C:₩Users₩gwihy₩Pyworks₩python_lec_01_basic.py

python_practice_01.py | dice2.py | python_lec_01_basic.py*

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr  1 11:09:14 2019

@author: gwihy
"""

import sys

input_list = sys.argv[1:]
input_list.sort()
print(input_list)


# python words_sort.py orange apple


# Program to sort alphabetically the words form a string provided by the user

# change this value for a different result
my_str = "Hello this Is an Example With cased letters"

# uncomment to take input from the user
#my_str = input("Enter a string: ")

# breakdown the string into a list of words
words = my_str.split()

# sort the list
words.sort()

# display the sorted words

print("The sorted words are:")
for word in words:
   print(word)
```

Help

Source Editor Object

Variable explorer | File explorer | He

IPython console

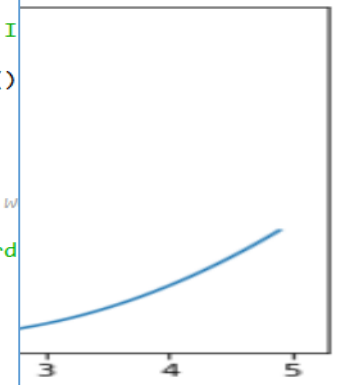
Console 1/A

```
In [4]: my_str = "Hello this I

In [5]: words = my_str.split()
   ...:
   ...: # sort the list
   ...: words.sort()
   ...:
   ...: # display the sorted w
   ...:
   ...: print("The sorted word
   ...: for word in words:
   ...:     print(word)
The sorted words are:
Example
Hello
Is
With
an
cased
letters
this
```

.03.24 (autosaved)

ts Help

```
W], feed_dict={W: feed_W})
```



Source :

데이터의 특성에 따라 다양한 tool 사용, 잊지 말 것은 활용과 아이디어 도출이다.

- Excel
- R
- dplyr, …
- Pandas
- Numpy

전처리

cleansing

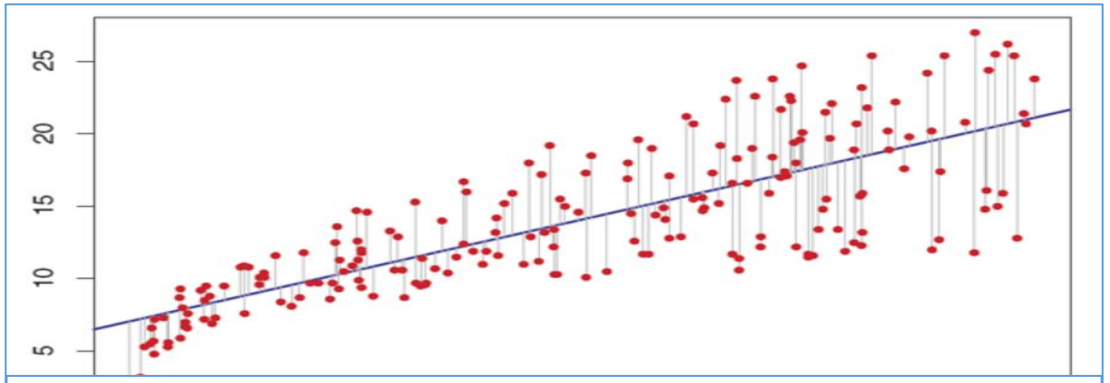
$$y = f(x)$$

H2O DataRobot R Studio python

Source :

Linear regression

```
import tensorflow as tf
tf.set_random_seed(777)  # for reproducibility

x_train = [1, 2, 3]
y_train = [2, 4, 6]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = x_train * W + b

cost = tf.reduce_mean(tf.square(hypothesis - y_train))

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)
```



Source : ** https://www.matlabsolutions.com/blog/tensorflow-linear-regression-understanding-the-concept.php

```
sess = tf.Session()

sess.run(tf.global_variables_initializer())

# Fit the line
# range 1001 : W 2.03 -> 2001 W  1.993

for step in range(2001):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

Source :

Summary – 6 steps

별첨

Data

- x_train = [1, 2, 3]
- y_train = [2, 4, 6]

Weight bias

- W = tf.Variable(tf.random_normal([1]), name='weight')
- b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis

- hypothesis = x_train * W + b

cost

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

optimizer

manual step :
(W- 기울기 * lr) ➔ update

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

```
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)
```

함수 사용
train= tf.train.GradientDescentOptimizer.minimize(cost)

build graph in a session

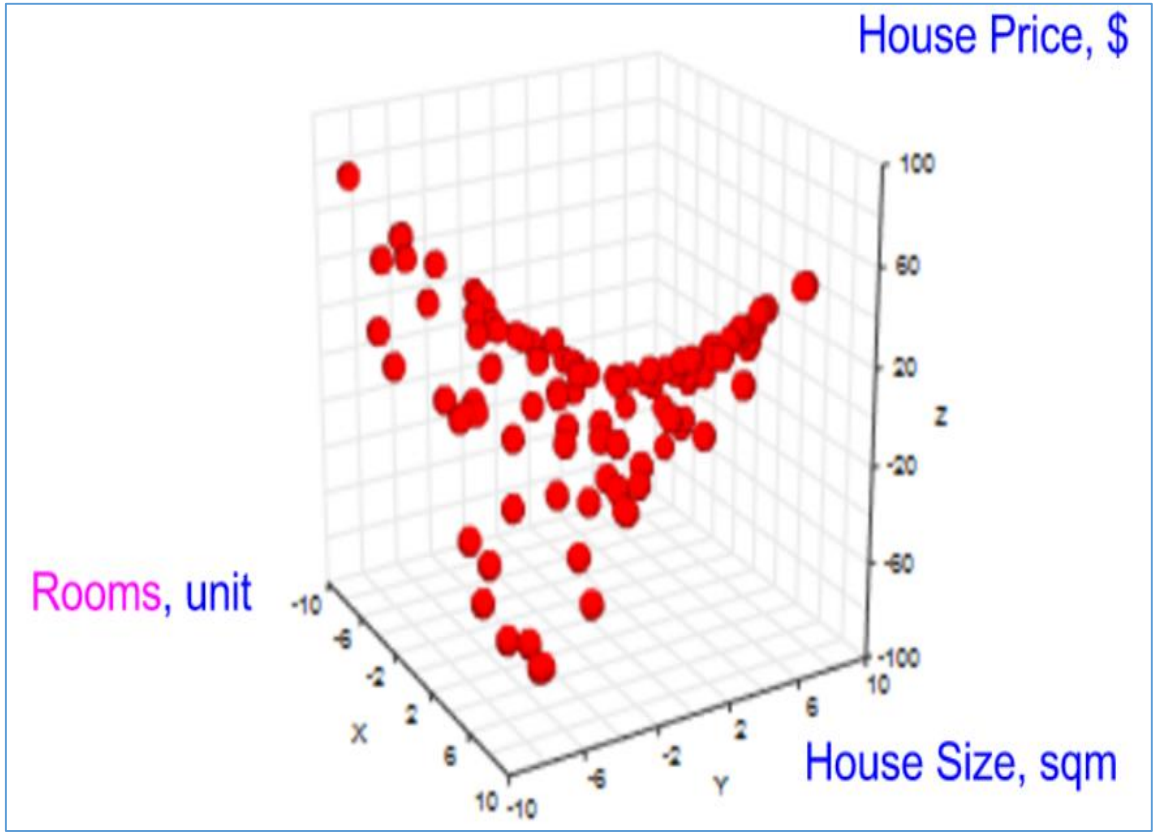
sess.run(update)

sess.run(train)

Source :

Multi-variable Linear regression

```
x_data = [[20., 10., 5.],
          [22, 11., 6.],
          [24., 12., 7.],
          [26., 13., 8.],
          [28., 14., 9.]]
y_data = [[65.],
          [72.],
          [79.],
          [86.],
          [93.]]

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis : h = 1 x1 + 2 x2 + 3 x3 + 10
hypothesis = tf.matmul(X, W) + b

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
```

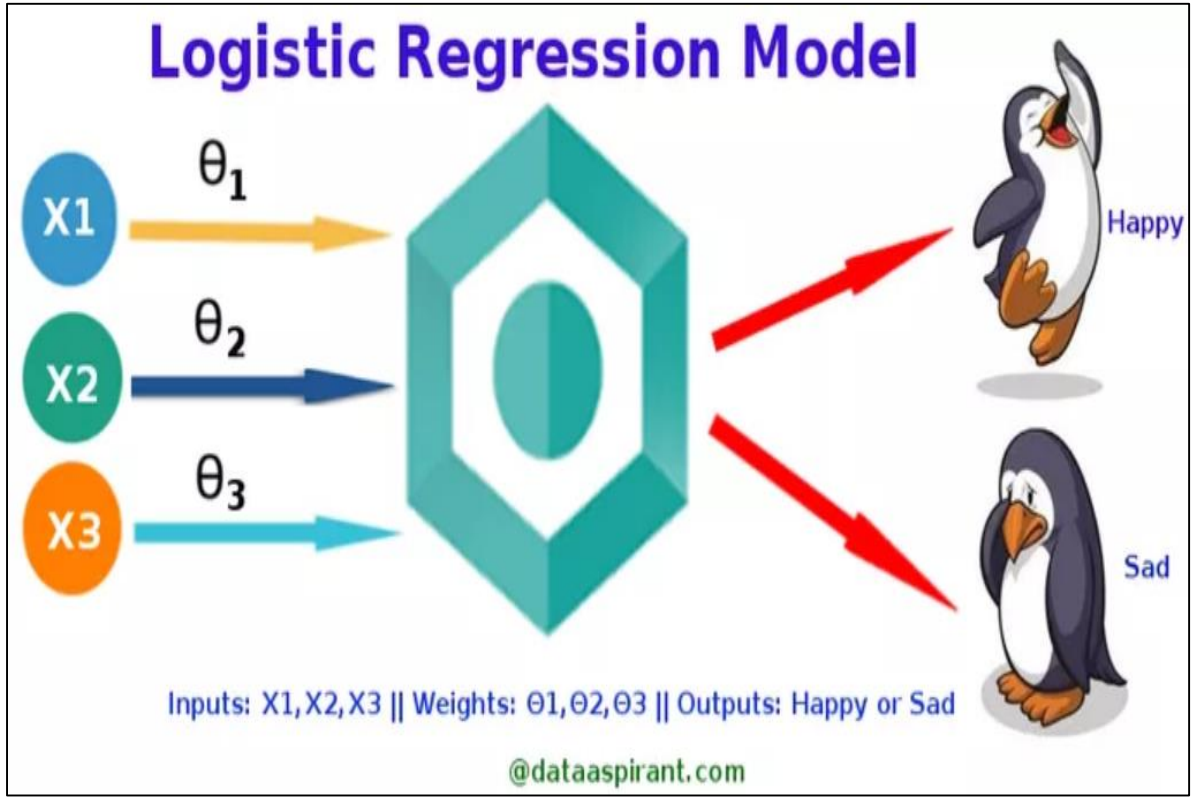


https://medium.com/all-of-us-are-belong-to-machines/gentlest-intro-to-tensorflow-part-3-matrices-multi-feature-linear-regression-30a81ebaaa6c

Source :

Logistic classification

```
xy = np.loadtxt('data-01-test-score.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

# Make sure the shape and data are OK
print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data)

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
```

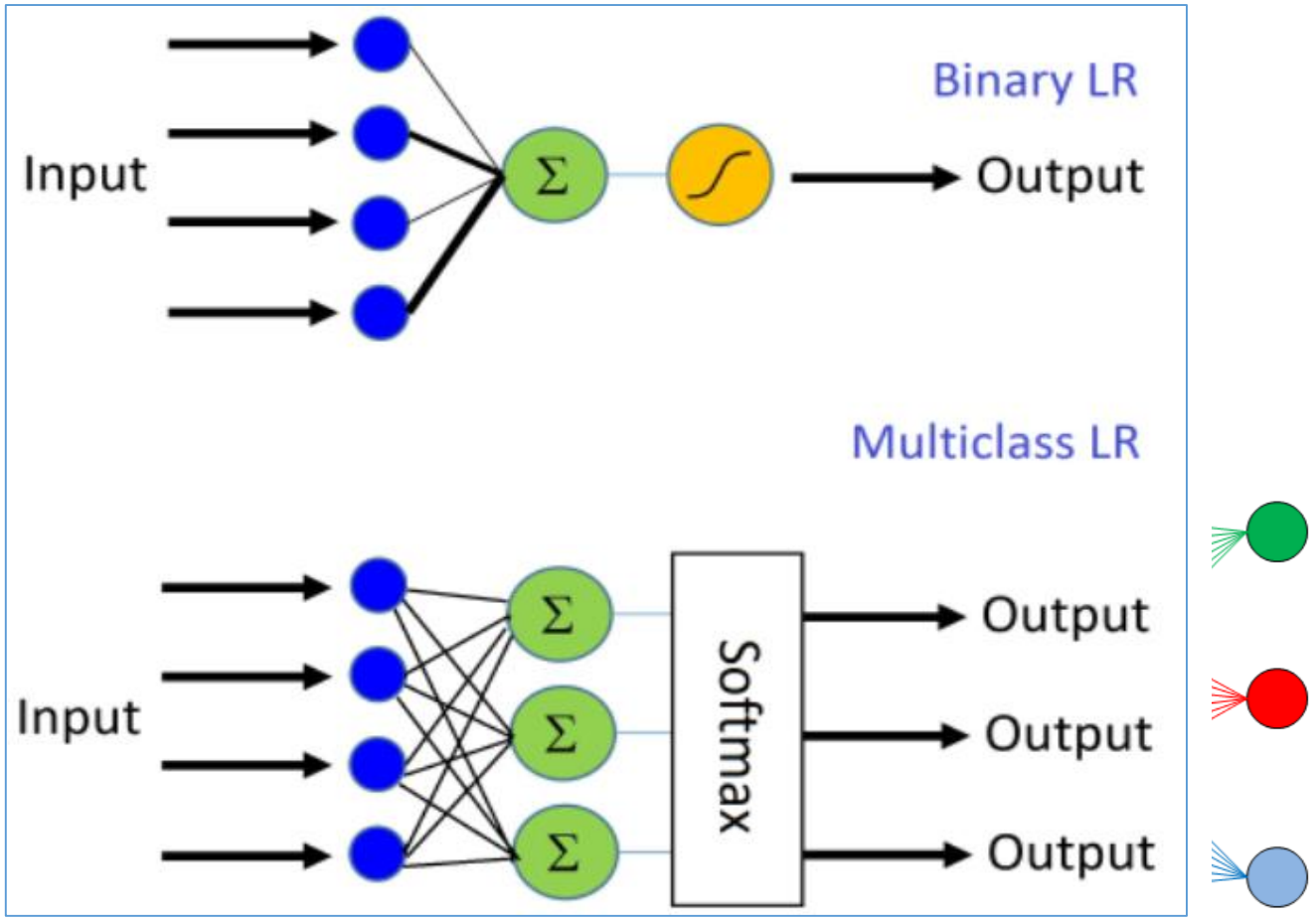


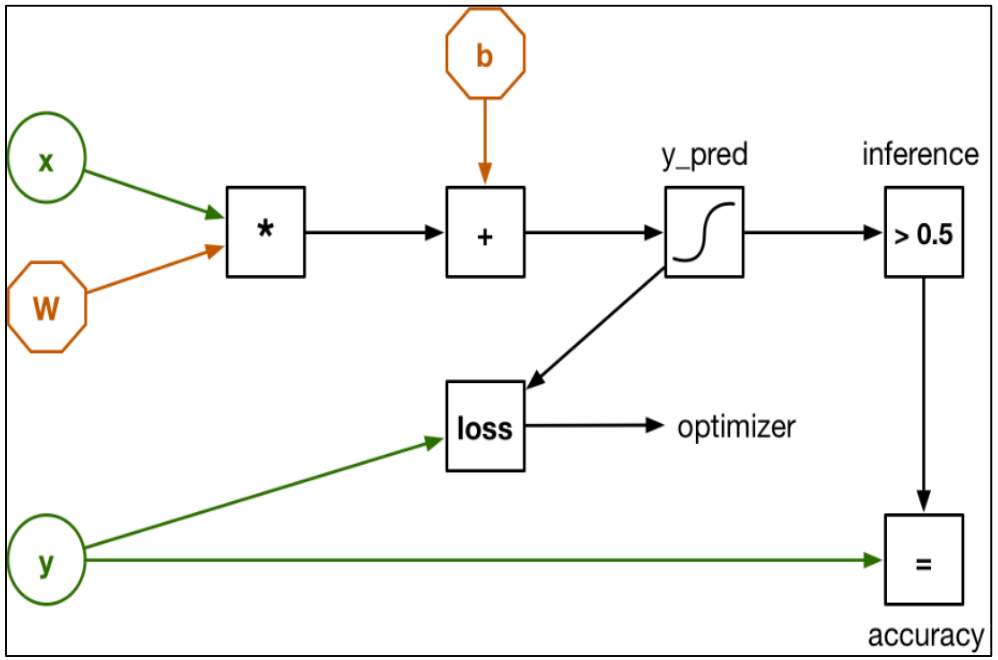
http://dataaspirant.com/2017/03/02/how-logistic-regression-model-works/

Source :

Logistic & Softmax Classification

여러 개 Labels 중 예측

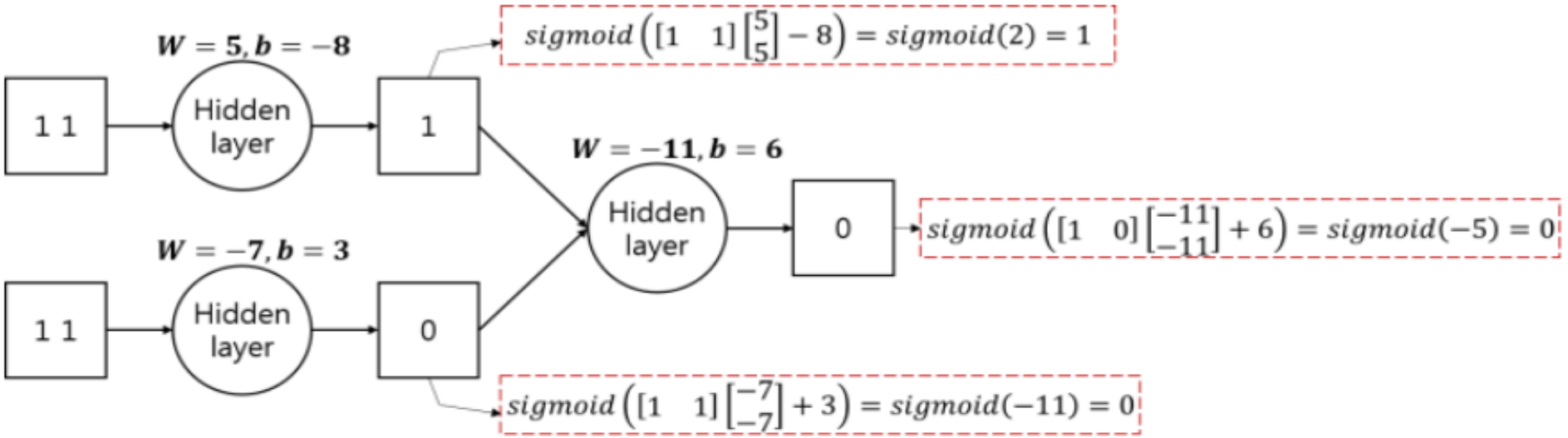




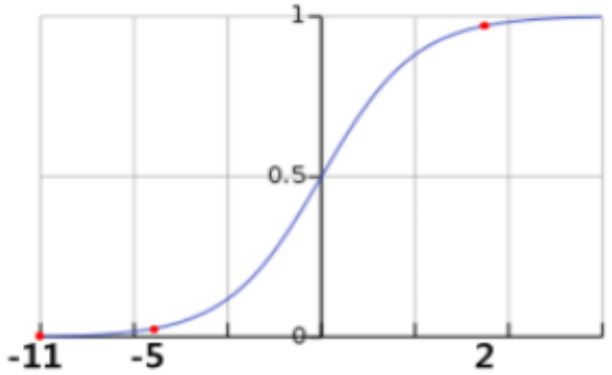
http://machinethink.net/blog/tensorflow-on-ios/

Source : https://cntk.ai/pythondocs/CNTK_103B_MNIST_LogisticRegression.html

sigmoid



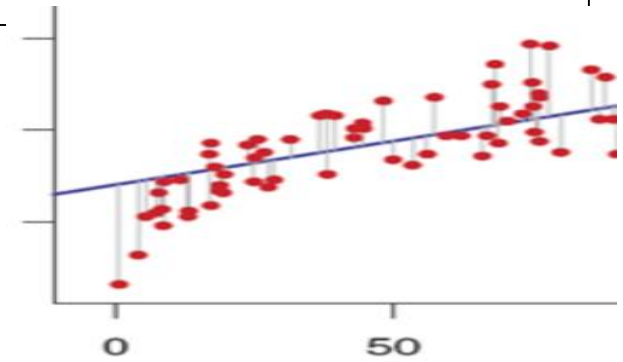
x_1	x_2	y_1	y_2	$\hat{y}$	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0



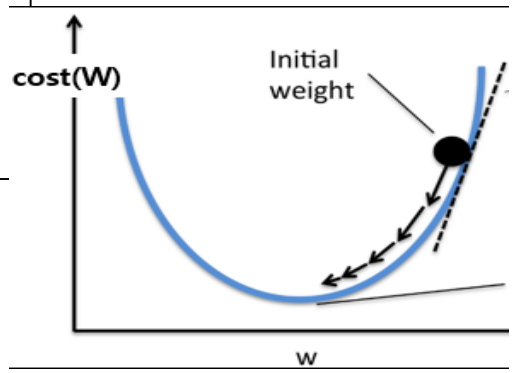
Source : https://www.youtube.com/watch?v=kNPGXgzxoHw

Linear regression vs Logistic classification

- **hypothesis = x_train * W + b**

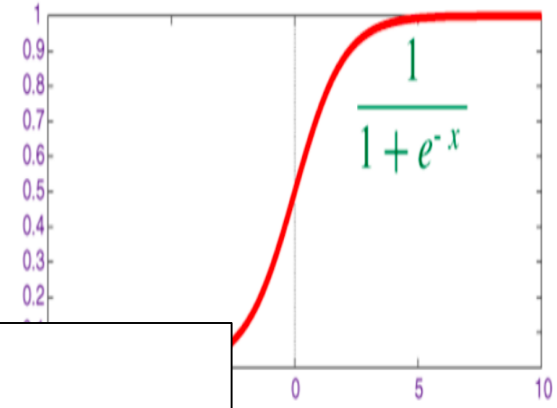


- **cost = tf.reduce_mean(tf.square(hypothesis - y_train))**

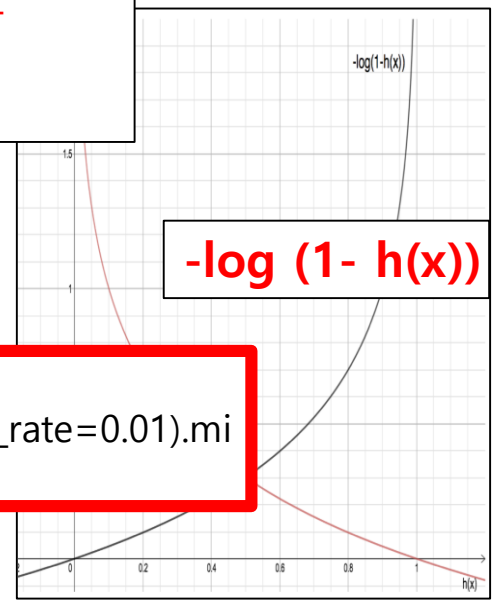


- **optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)**
- **xy_train = optimizer.minimize(cost)**

hypothesis = tf.sigmoid(tf.matmul(X, W) + b)



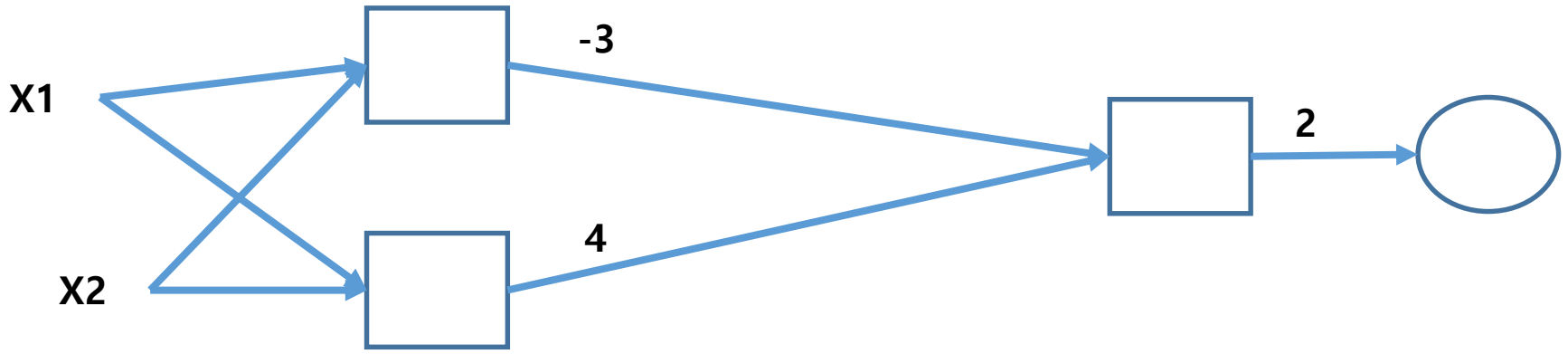

```
# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
```




```
train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)
```

Source :

Neural net for XOR

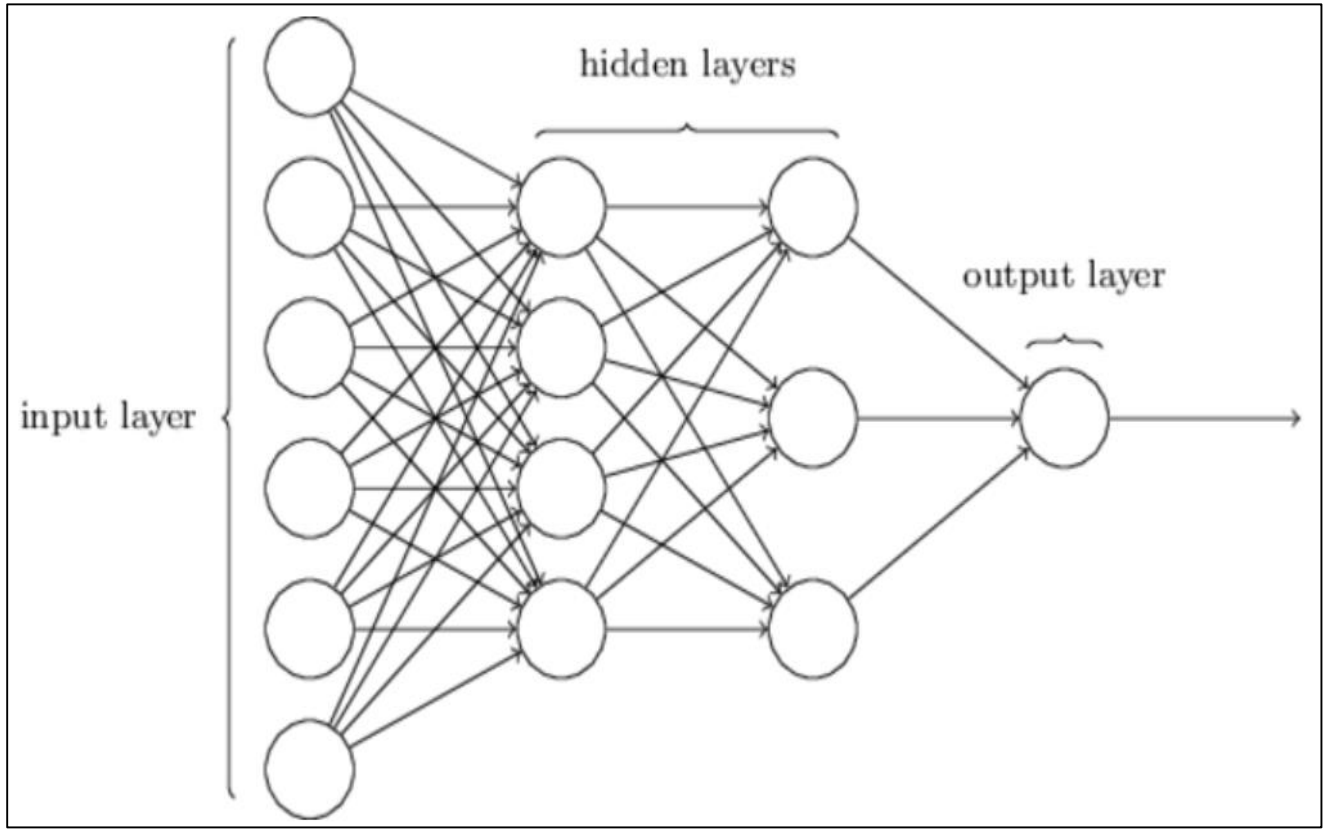


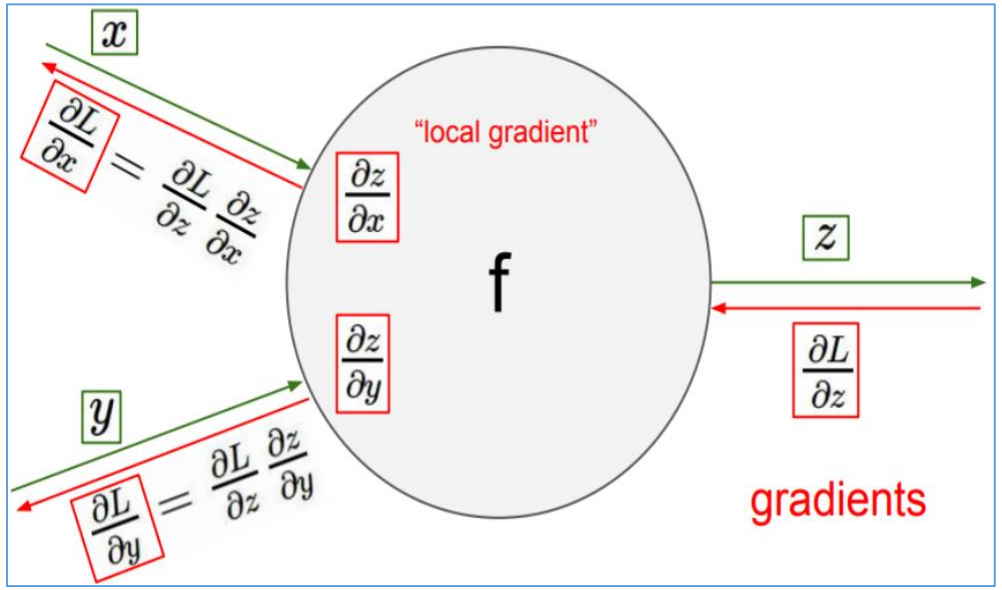
$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix} * \underset{\text{weight}}{\begin{pmatrix} 2 & -5 \\ 2 & -5 \end{pmatrix}} \underset{\text{bias}}{\begin{pmatrix} -3 & 4 \end{pmatrix}} \Rightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} * \underset{\text{weight}}{\begin{pmatrix} -5 \\ -5 \end{pmatrix}} \underset{\text{bias}}{\begin{pmatrix} 2 \end{pmatrix}} \Rightarrow \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Source :

MLP & Back propagation

다층구조, 초기값,

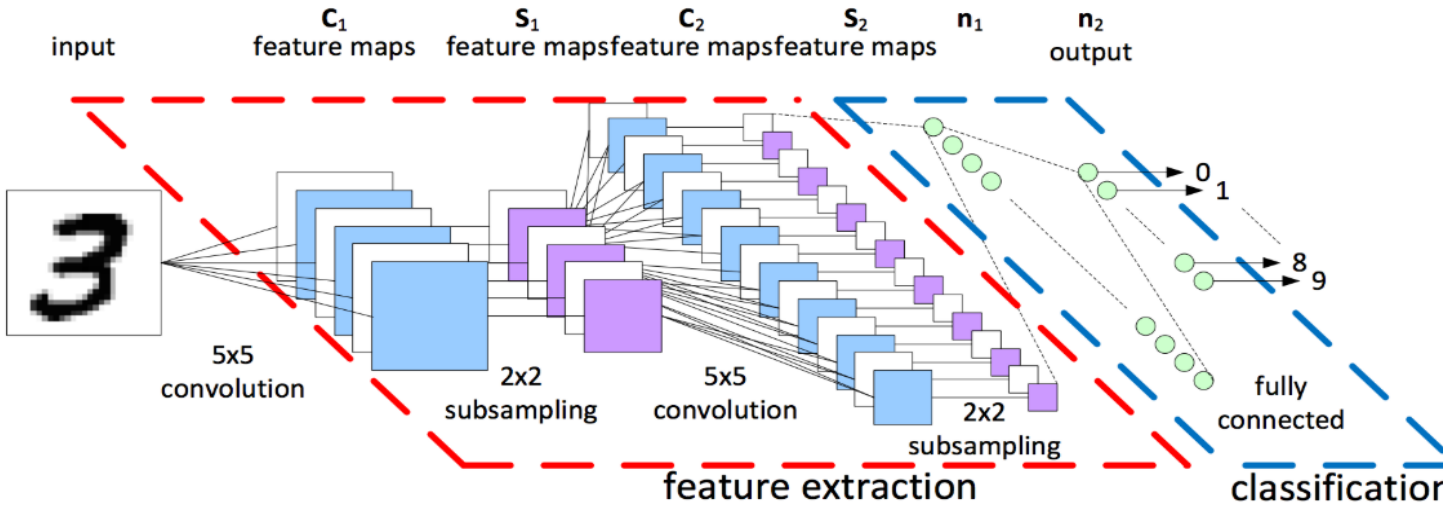


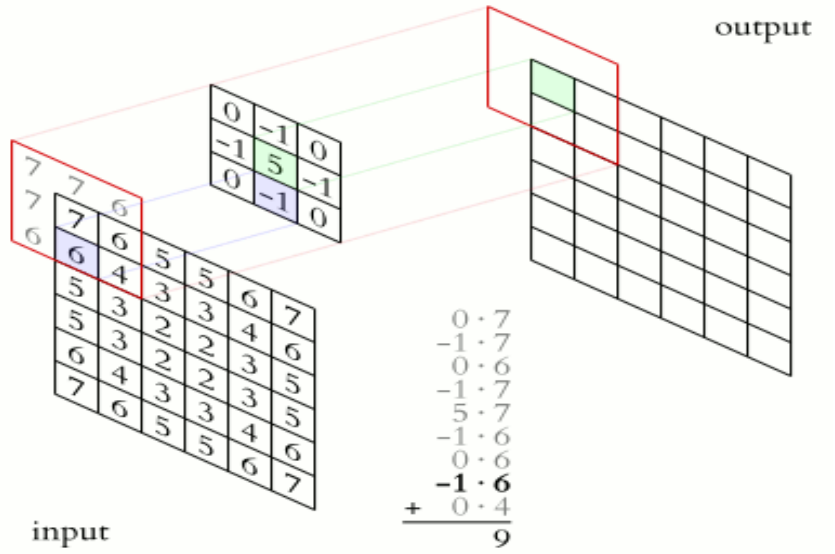


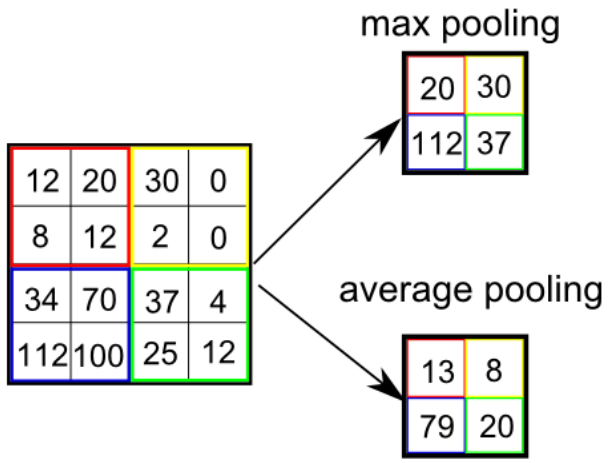
https://www.cl.cam.ac.uk/archive/mjcg/plans/Backpropagation.html

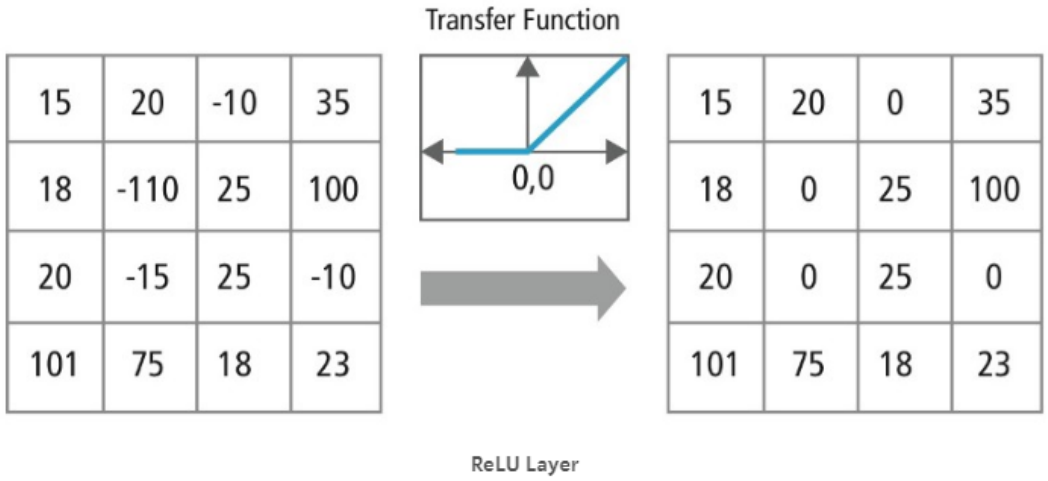
Source : https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

CNN Basics





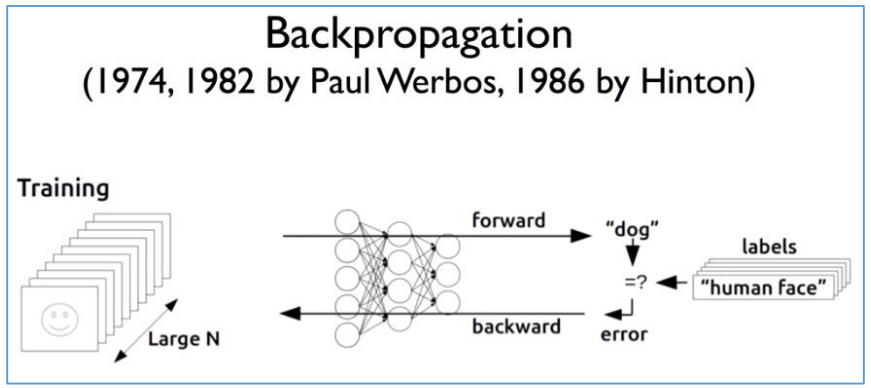


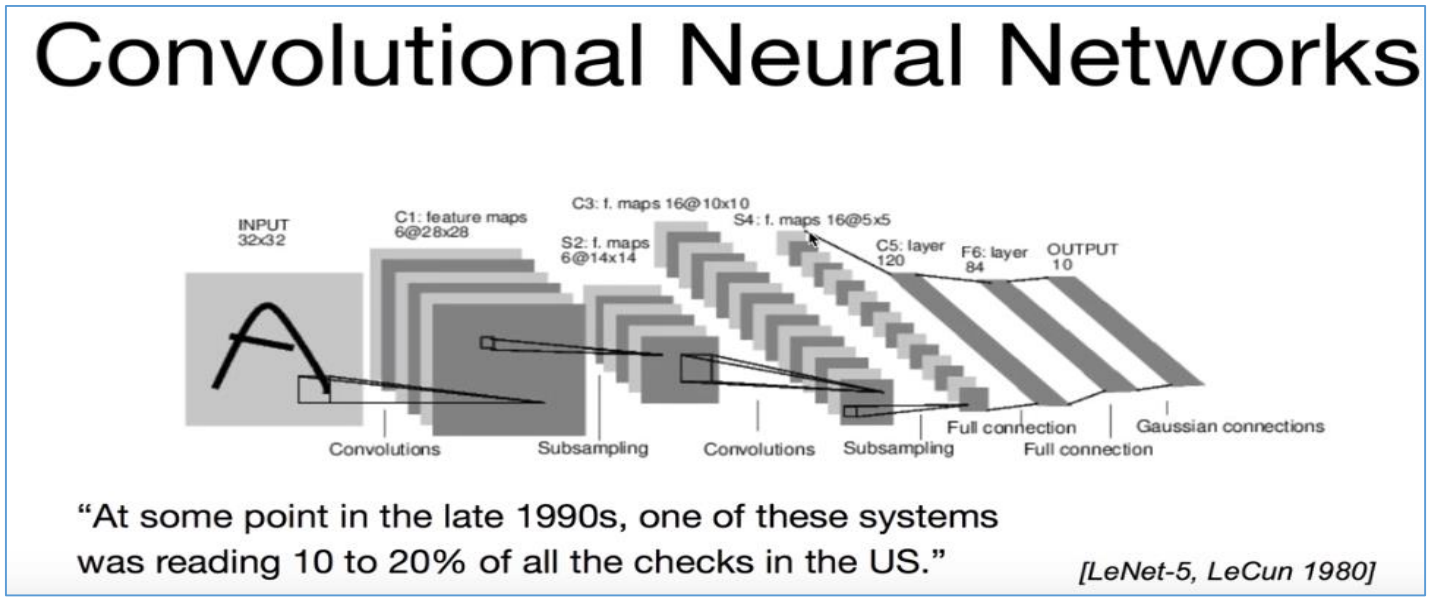


Source : https://medium.com/data-science-group-iitr/building-a-convolutional-neural-network-in-python-with-tensorflow-d251c3ca8117

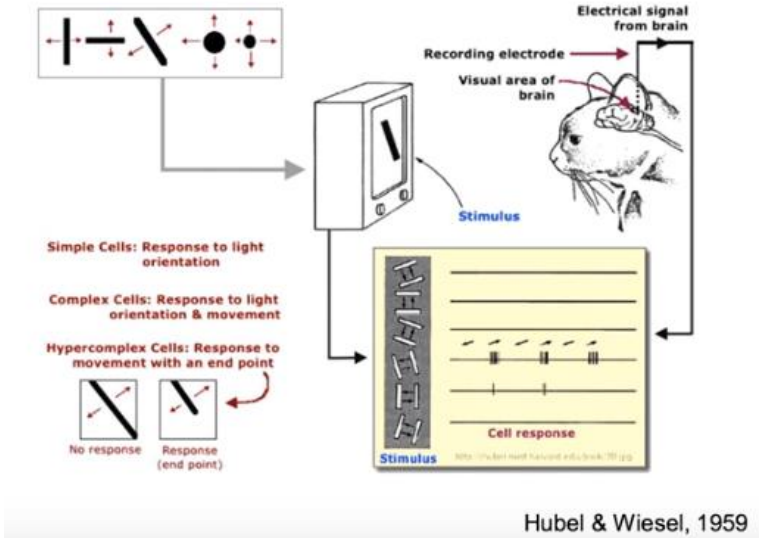
XOR 문제 해결 (Backpropagation, CNN)

Paul 74년 논문에 제시 : forward로 계산한 error를 backward 로 진행하면서 조정





- **Hinton 86년에 재발견 해결 제시**



- **고양이 그림의 형태에 따라 다른 뉴런이 동작하는 것을 발견**
- **LeCun : CNN 제시 : 부분 부분을 해석하여 나중에 합쳐서 해석 (잘 동작 : 문자, 숫자 인식)**
- **그림의 전체를 보는 것이 아니라 한번에 뉴런에 입력 시키는 것이 아니라, 부분 부분을 보냄**

Source : https://youtu.be/n7DNueHGkqE

Tensorboard (Neural Net for XOR)

5 steps of using TensorBoard

1. From TF graph, decide which tensors you want to log

```
w2_hist = tf.summary.histogram("weights2", W2)
cost_summ = tf.summary.scalar("cost", cost)
```

어떤 것을 logging할 것인지 정함

모든 요약을 합침

2. Merge all summaries

```
summary = tf.summary.merge_all()
```

3. Create writer and add graph

```
# Create summary writer
writer = tf.summary.FileWriter('./logs')
writer.add_graph(sess.graph)
```

- 기록할 file의 위치를 정함
- 그래프를 추가

4. Run summary merge and add_summary

```
s, _ = sess.run([summary, optimizer], feed_dict=feed_dict)
writer.add_summary(s, global_step=global_step)
```

- 그래프 summary도 tensor 이므로
- 실행을 위해 sess.run

5. Launch TensorBoard

```
tensorboard --logdir=./logs
```

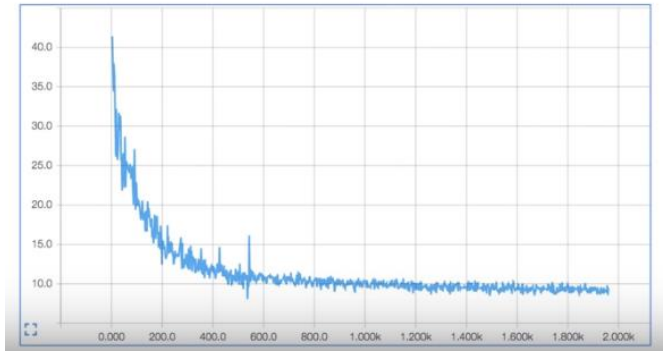
- terminal command

Source : https://youtu.be/lmrWZPFYjHM

Tensorboard (Neural Net for XOR)

Scalar tensors

```
cost_summ = tf.summary.scalar("cost", cost)
```



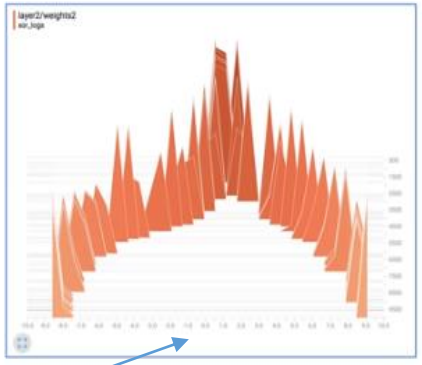
1. From TF graph, decide which tensors you want to log

```
w2_hist = tf.summary.histogram("weights2", W2)
cost_summ = tf.summary.scalar("cost", cost)
```

값이 하나의 값을 갖는 경우

Histogram (multi-dimensional tensors)

```
W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

w2_hist = tf.summary.histogram("weights2", W2)
b2_hist = tf.summary.histogram("biases2", b2)
hypothesis_hist = tf.summary.histogram("hypothesis", hypothesis)
```



값이 여러 개의 값을 갖는 경우
처음 weight가 모여 있다가, 점차 특정 값으로 나뉘어 분포

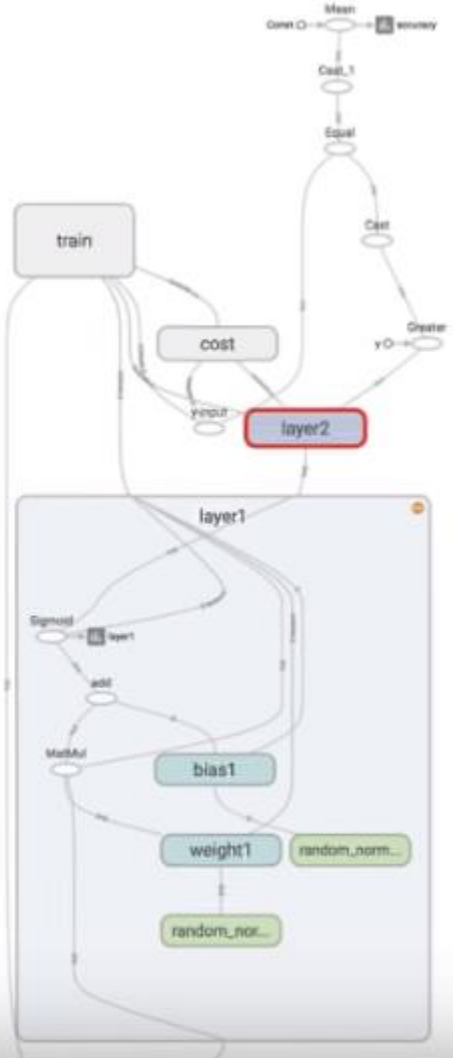
Source : https://youtu.be/lmrWZPFYjHM

Tensorboard (Neural Net for XOR)

name_scope 이용 구분

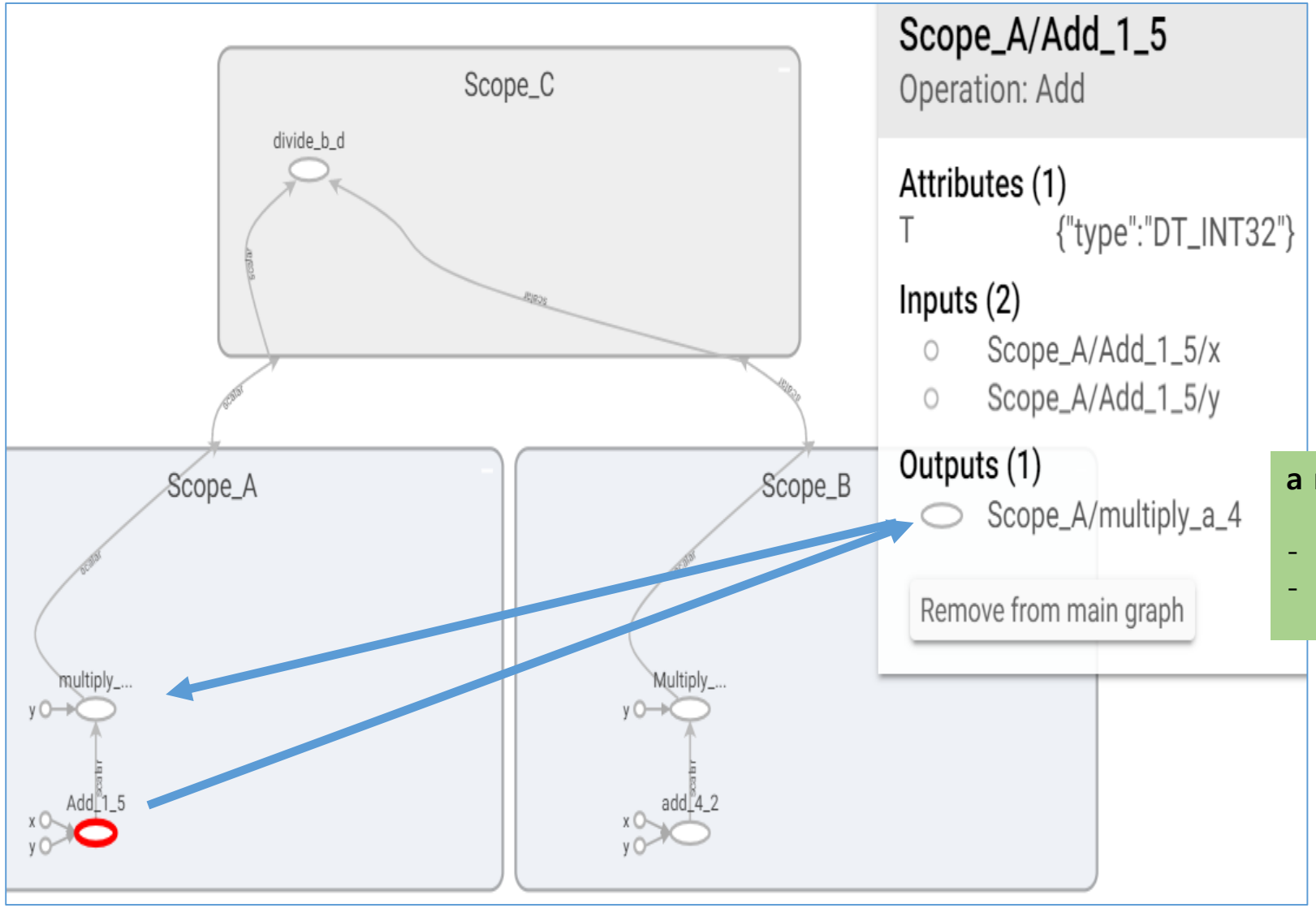
Add scope for better graph hierarchy

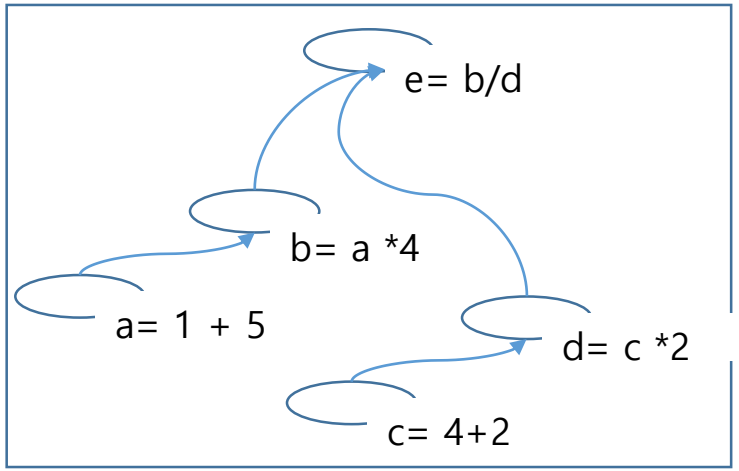
```
with tf.name_scope("layer1") as scope:
    W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
    b1 = tf.Variable(tf.random_normal([2]), name='bias1')
    layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

    w1_hist = tf.summary.histogram("weights1", W1)
    b1_hist = tf.summary.histogram("biases1", b1)
    layer1_hist = tf.summary.histogram("layer1", layer1)


with tf.name_scope("layer2") as scope:
    W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
    b2 = tf.Variable(tf.random_normal([1]), name='bias2')
    hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

    w2_hist = tf.summary.histogram("weights2", W2)
    b2_hist = tf.summary.histogram("biases2", b2)
    hypothesis_hist = tf.summary.histogram("hypothesis", hypothesis)
```



Source : https://youtu.be/lmrWZPFYjHM

TensorBoard



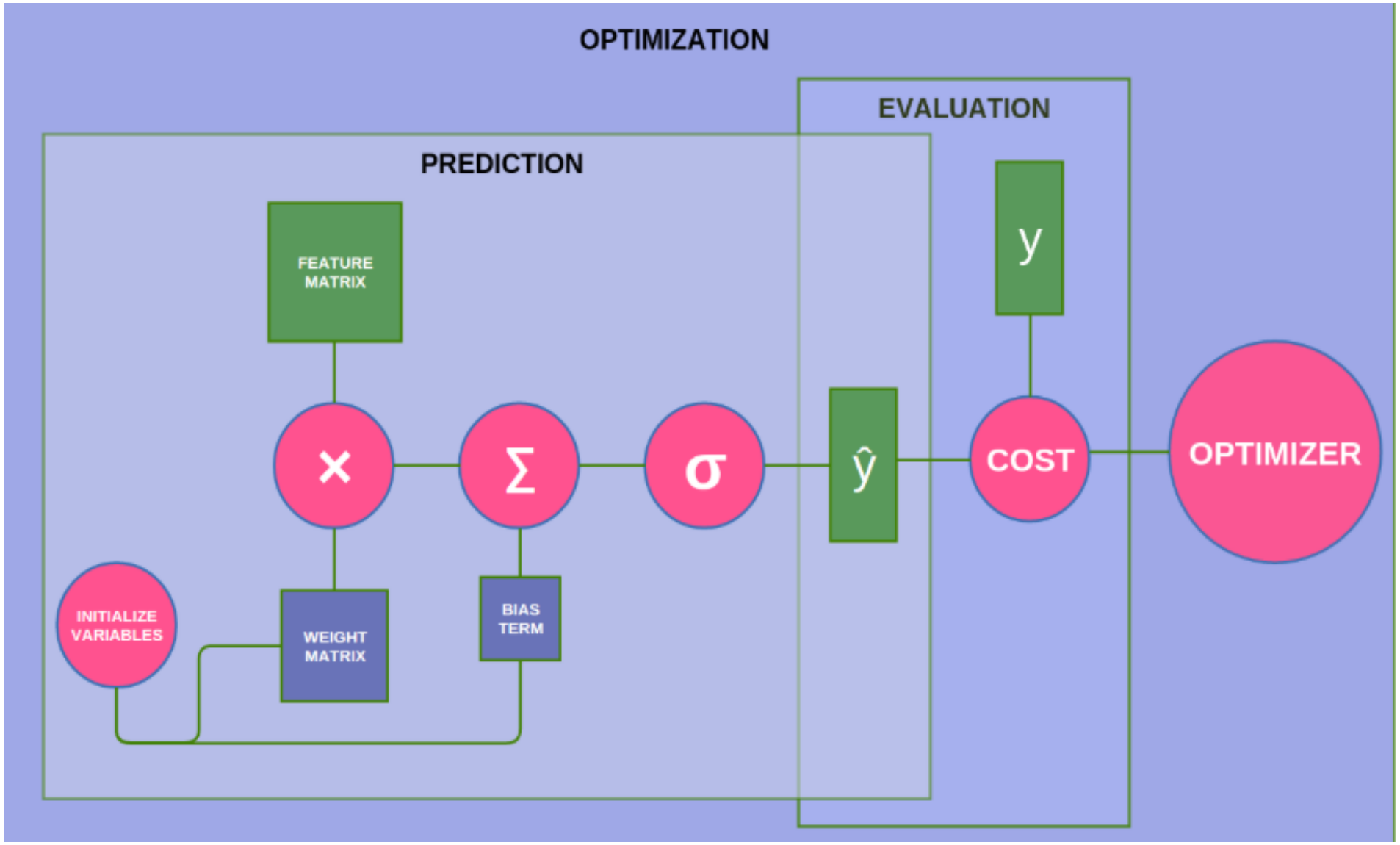


a node를 선택한 경우

- **input 2개 : 1, 5**
- **output : Scope_A multiply = a * 4**

Source :

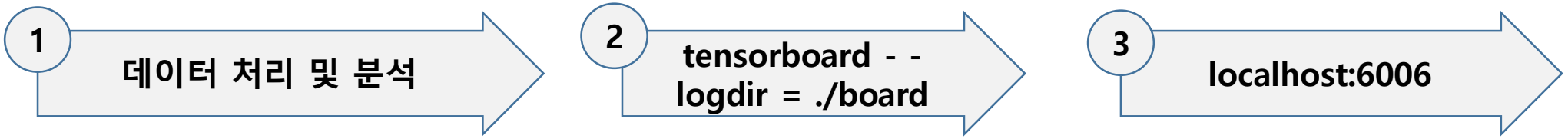
Analysis Flow



Source : http://jrmeyer.github.io/machinelearning/2016/02/01/TensorFlow-Tutorial.html

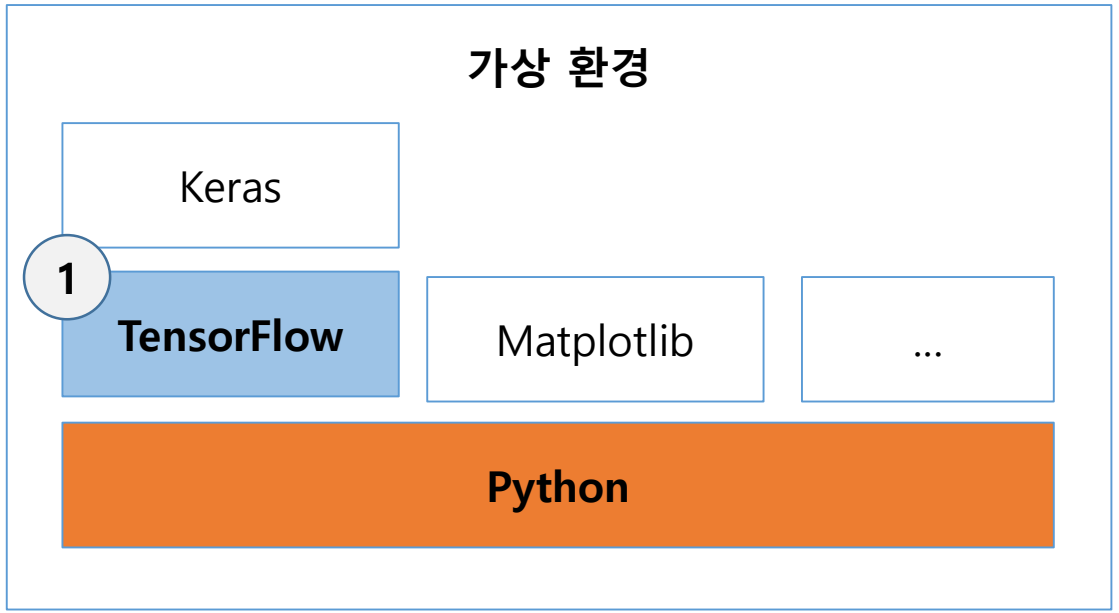
Program

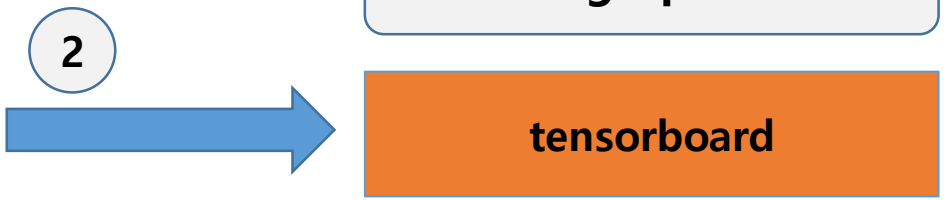
- **python / tensorflow / tensorboard.py**
- **source 에서 log 위치, 해당 디렉토리에 확장자 local 파일 생성, 콘솔에서 tensorboard 실행, 브라우저에서 실행**

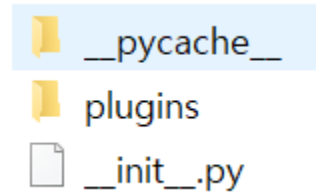


- C:₩Users₩beomc₩Anaconda3 /envs/py35 /Lib/sitepackages
  / tensorflow / include /tensorflow

- C:₩Users₩beomc₩Anaconda3 ~
  / tensorflow / contrib / tensorboard

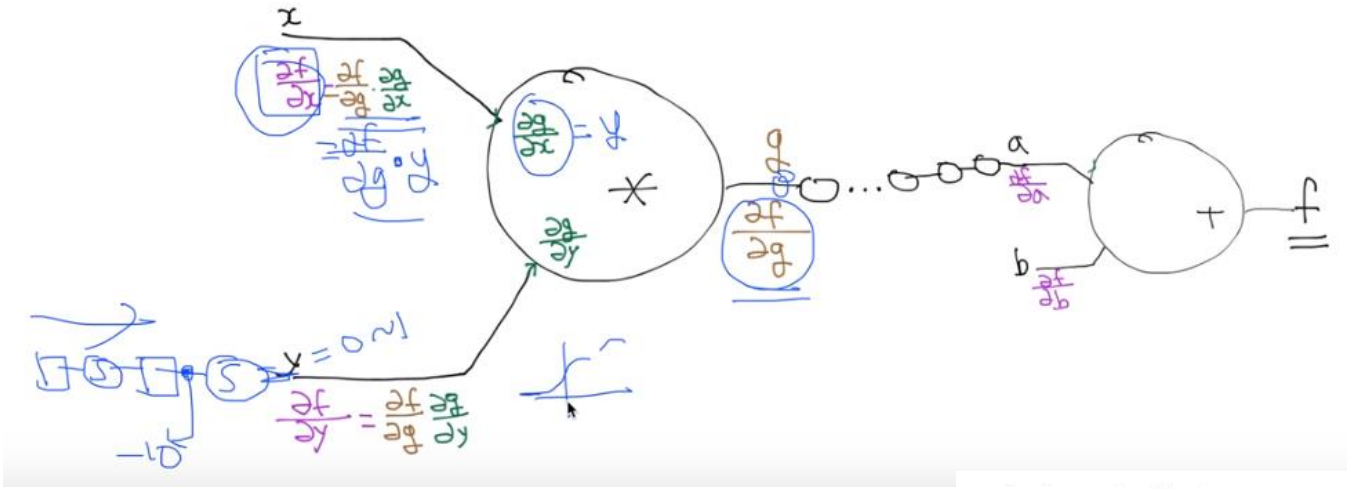






Source :

9 layers 임에도 결과가 나쁜 이유 : backpropagation (단이 깊어지면 문제 발생)



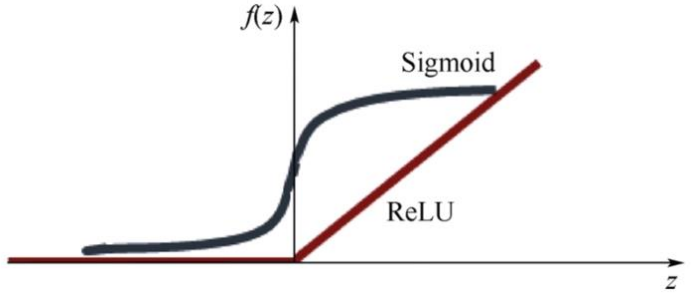
- x 가 f에 미치는 영향을 구함
- 연산이 * 인 경우
- local 미분 ➔ g를 x로 미분 = y

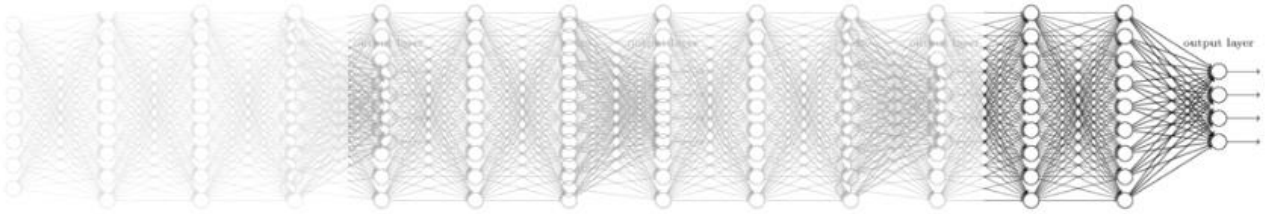
- sigmoid를 통과한 값 은 1보다 적은 값

shing gradient (NN winter2: 1986-2006)

- 단이 깊어질 수록 vanishing gradient (~ 2006)
- ReLU : 0 보다 적을 경우 꺼버리고, 0 보다 크면 그대로 반영

ReLU: Rectified Linear Unit



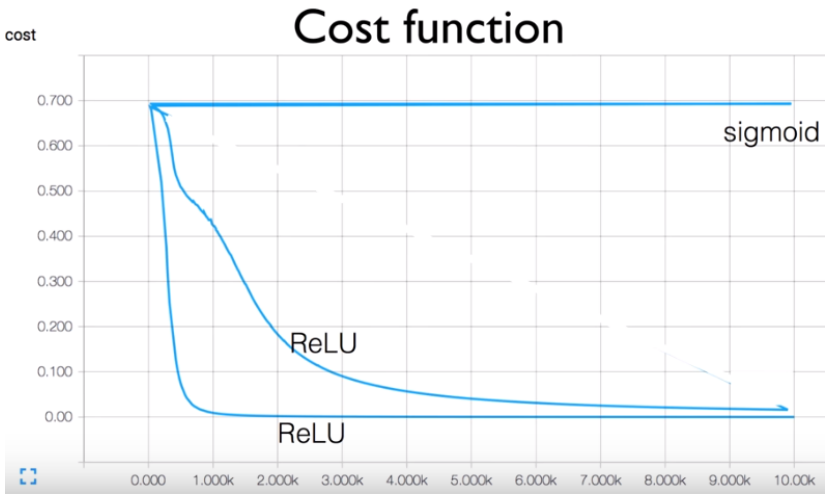


```
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
```

Source : https://youtu.be/cKtg_fpw88c

weight 초기화

- Our labeled datasets were thousands of times too small.
- Our computers were millions of times too slow.
- We initialized the weights in a stupid way.
- We used the wrong type of non-linearity.

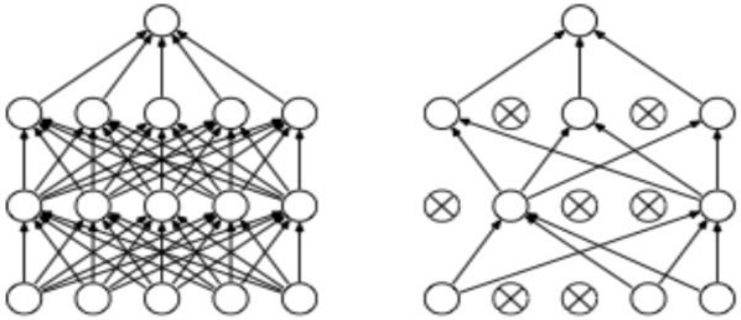


- Not all 0's
- Challenging issue
- Hinton et al. (2006) "A Fast Learning Algorithm for Deep Belief Nets"
  - Restricted Boatman Machine (RBM)

Source : https://youtu.be/4rC0sWrp3Uw

NN dropout & model ensemble

Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Srivastava et al. 2014]



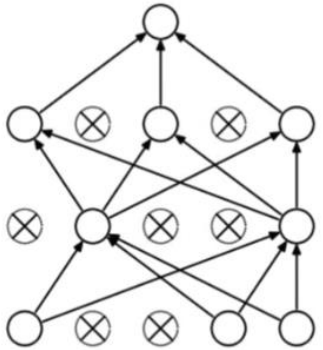
(a) Standard Neural Net (b) After applying dropout.

TensorFlow implementation

```
dropout_rate = tf.placeholder("float")
_L1 = tf.nn.relu(tf.add(tf.matmul(X, W1), B1))
L1 = tf.nn.dropout(_L1, dropout_rate)
```

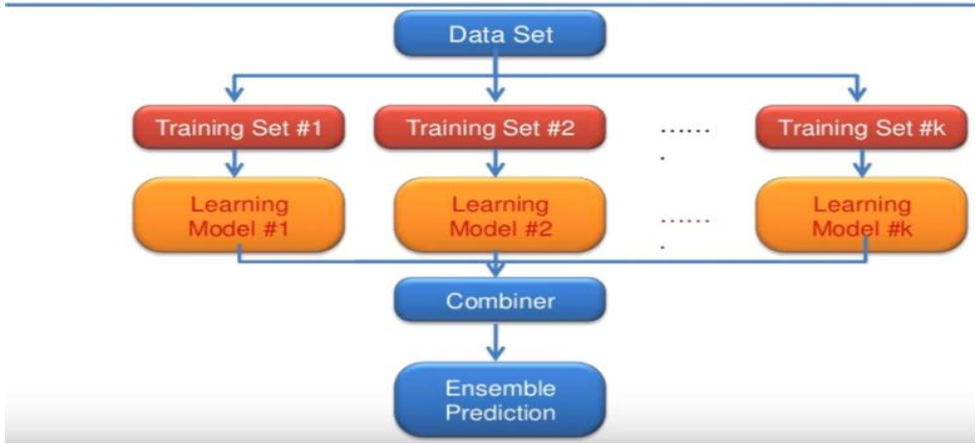
```
TRAIN:
  sess.run(optimizer, feed_dict={X: batch_xs, Y: batch_ys,
  dropout_rate: 0.7})


EVALUATION:
  print "Accuracy:", accuracy.eval({X: mnist.test.images, Y:
  mnist.test.labels, dropout_rate: 1})
```

Waaaait a second…
How could this possibly be a good idea?



Forces the network to have a redundant representation.

What is Ensemble?



Source : https://youtu.be/wTxMsp22llc

NN, ReLu, Xavier, Dropout, and Adam

softmax classifier for MNIST : 90 %

NN for MNIST : 94 %

```
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# weights & bias for nn layers
W1 = tf.Variable(tf.random_normal([784, 256]))
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([256, 256]))
b2 = tf.Variable(tf.random_normal([256]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)

W3 = tf.Variable(tf.random_normal([256, 10]))
b3 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L2, W3) + b3

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entro
   logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rat
```

Xavier for MNIST : 97.8 %

```
# weights & bias for nn layers
# http://stackoverflow.com/questions/33640581
W1 = tf.get_variable("W1", shape=[784, 256],
                     initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
```

softmax classifier for MNIST : 90 %

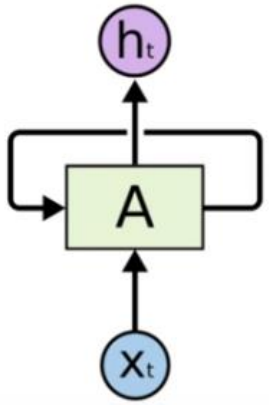
Deep NN for MNIST : 97 %
Overfitting problem

Dropout for MNIST : 98 %

```
W1 = tf.get_variable("W1", shape=[784, 512])
b1 = tf.Variable(tf.random_normal([512]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)

W2 = tf.get_variable("W2", shape=[512, 512])
b2 = tf.Variable(tf.random_normal([512]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)
L2 = tf.nn.dropout(L2, keep_prob=keep_prob)
...
# train my model
for epoch in range(training_epochs):
```

Source : https://youtu.be/6CCXyfvubvY

Basics




```
# One hot encoding
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

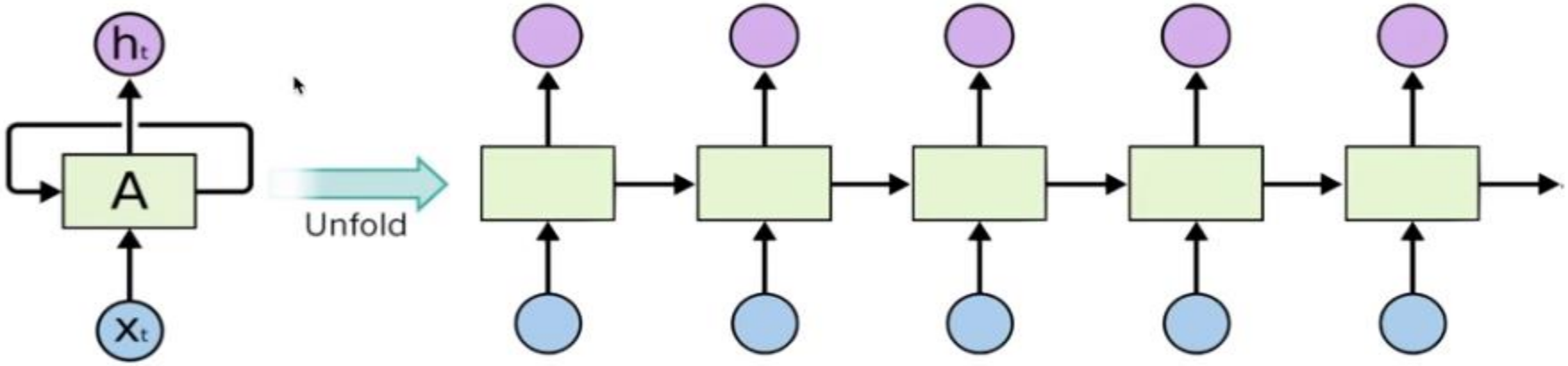
hidden size 2가 주어져 있기 때문에 출력 size 값들이 2

- **input dimension 4,**
- **sequence length = series data**

```
Hidden_size=2
sequence_length=5
```

```
[[[1,0,0,0]]]
shape=(1,1,4)
```

shape=(1,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]]]



shape=(1,5,4): [[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,1,0], [0,0,0,1]]]
h e l l o

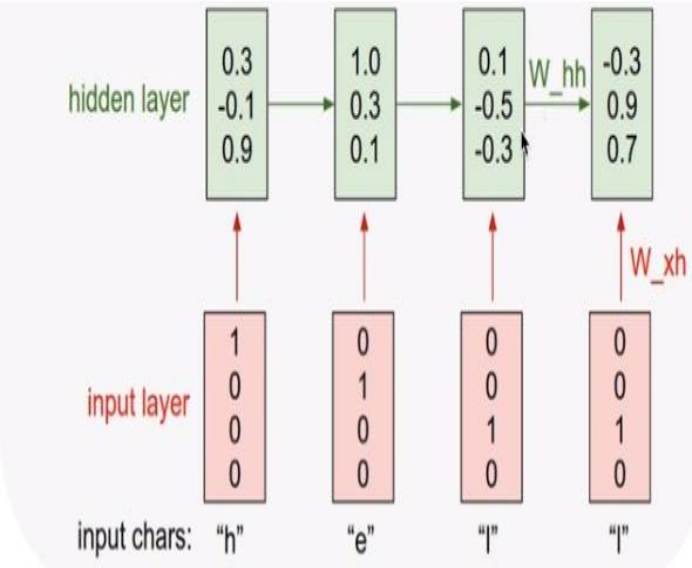
Source : https://youtu.be/B5GtZuUvujQ

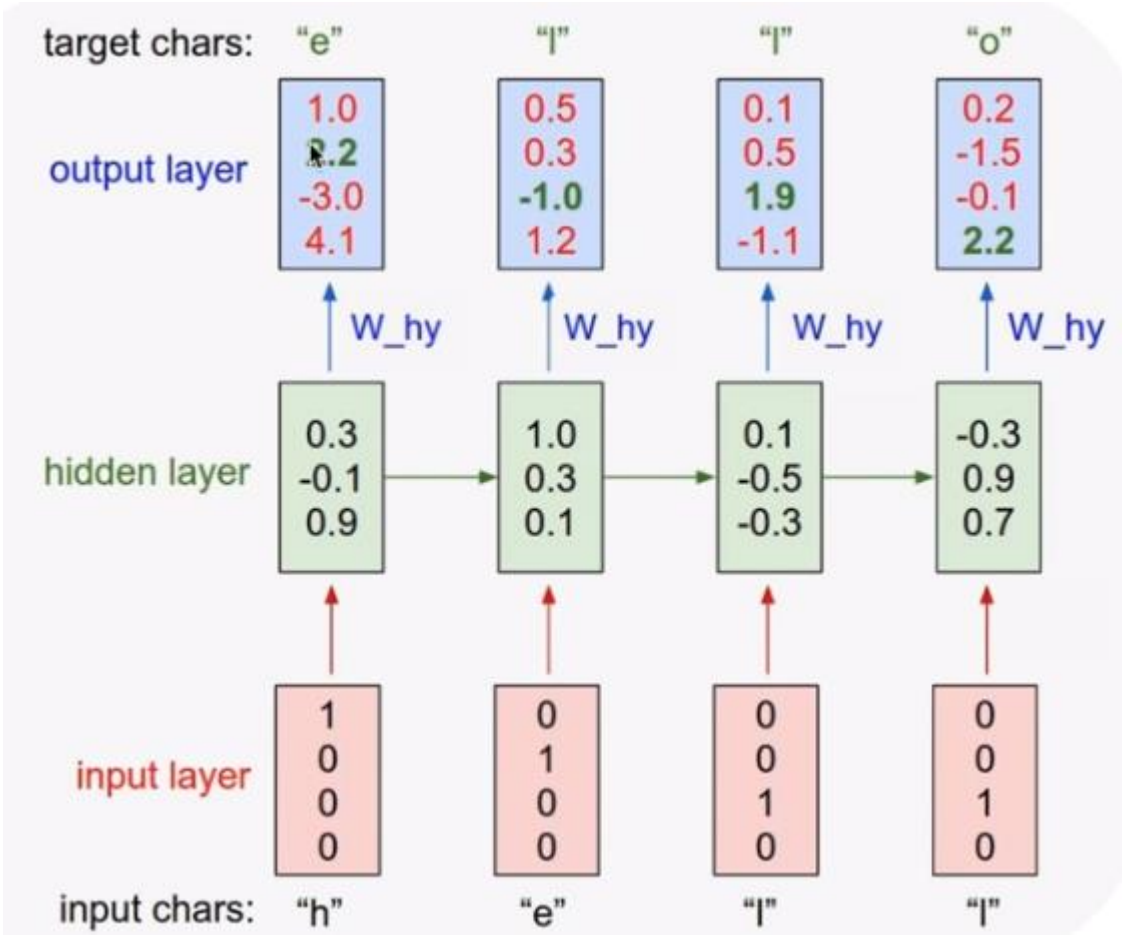
Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



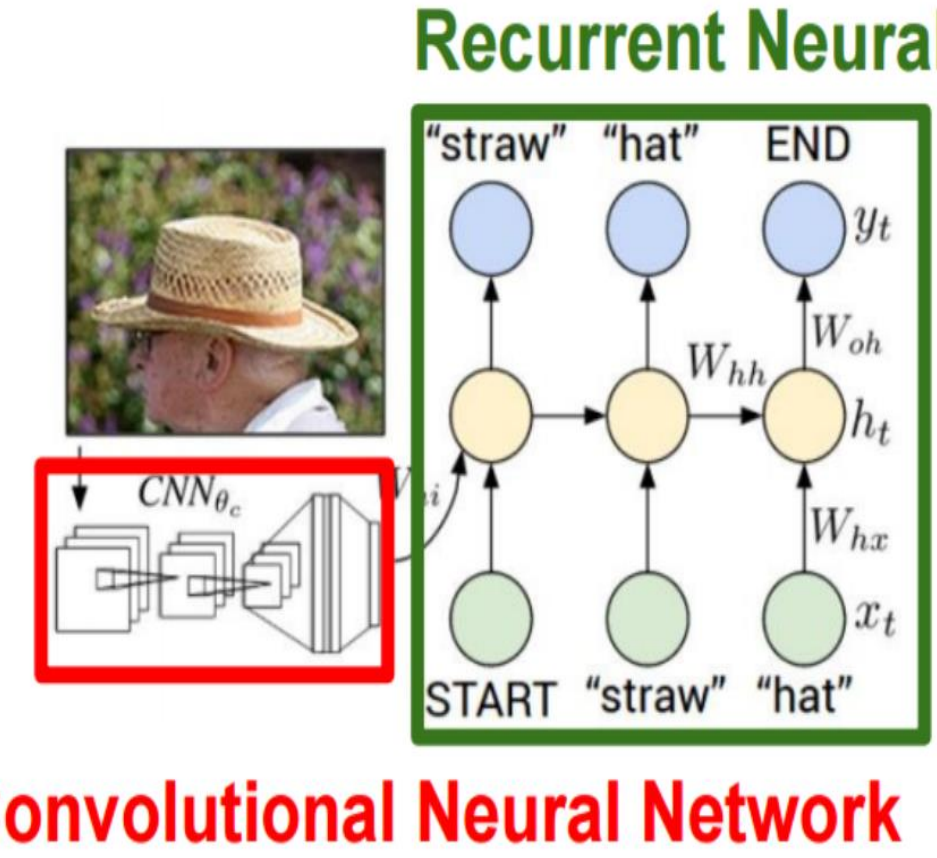


Source : https://youtu.be/-SHPG_KMUkQ

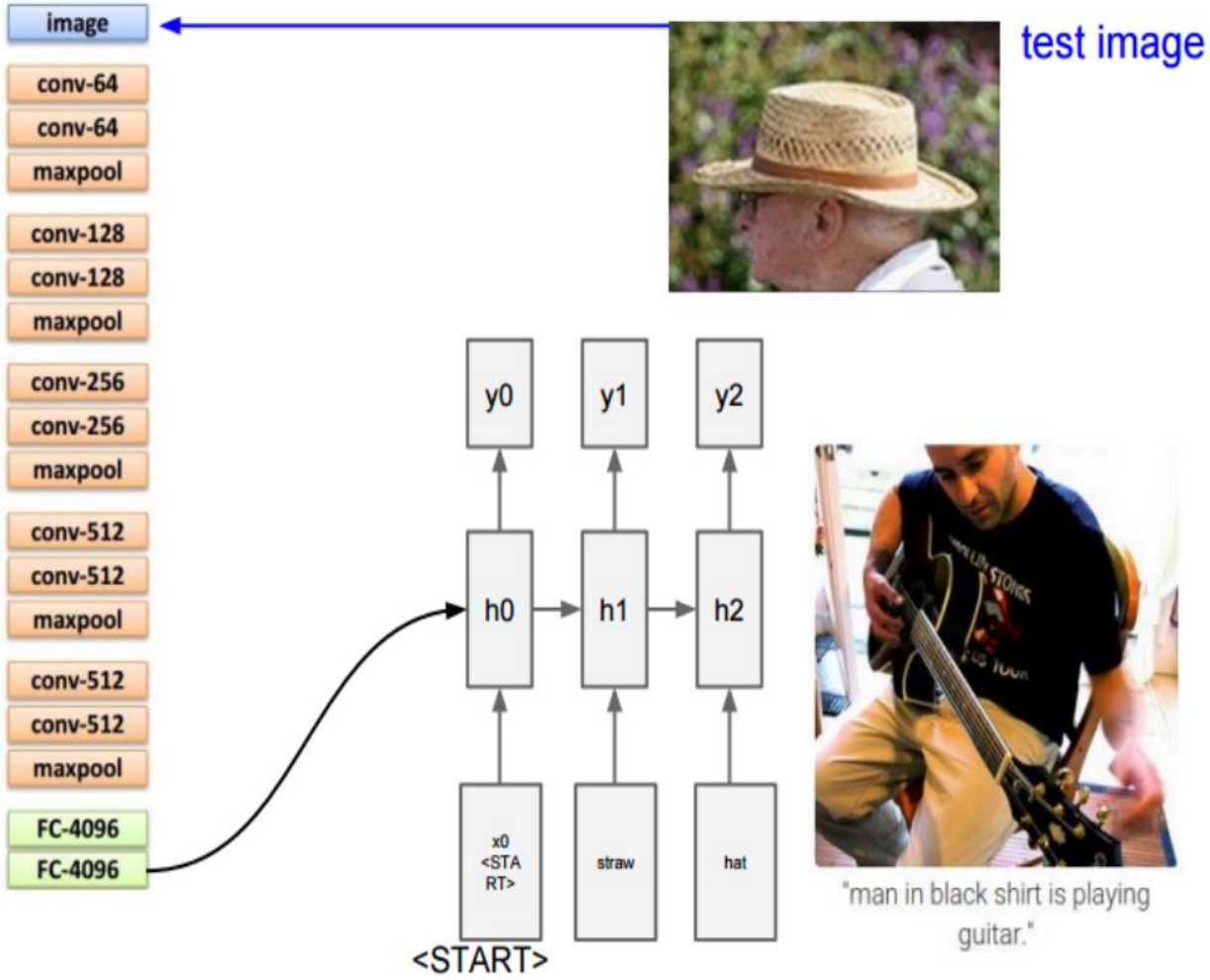
RNN

선형 회귀

If you connect a convolution neural network, with pre-trained RNN. The RNN will be a
it "see" on the image.



Basically we get a pre-trained CNN (ie: VGG) and connect the second-to-last FC layer and connect to a RNN. After this you train the whole thing end-to-end.



Source : https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/recurrent_neural_networks.html

linear regression – logistic – softmax classfication

	linear regression	logistic classification	CNN	RNN
hypothesis	= x_train * W + b	g(z) = 1 / (1+exp –z)		
cost	= tf.reduce_mean (tf.square(hypothesis – y_train))	= -y log(hx) – (1-y)log(1-hx)		
optimizer & training	= tf.train.GradientDescent Optimizer(learning_rate=0.1) train = optimizer.minimize (cost)	=		

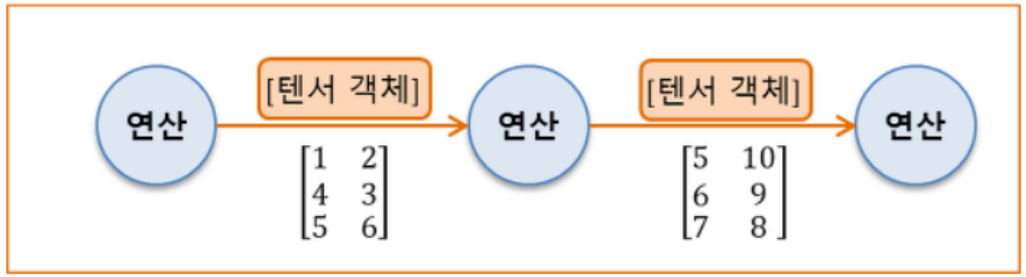
Source :

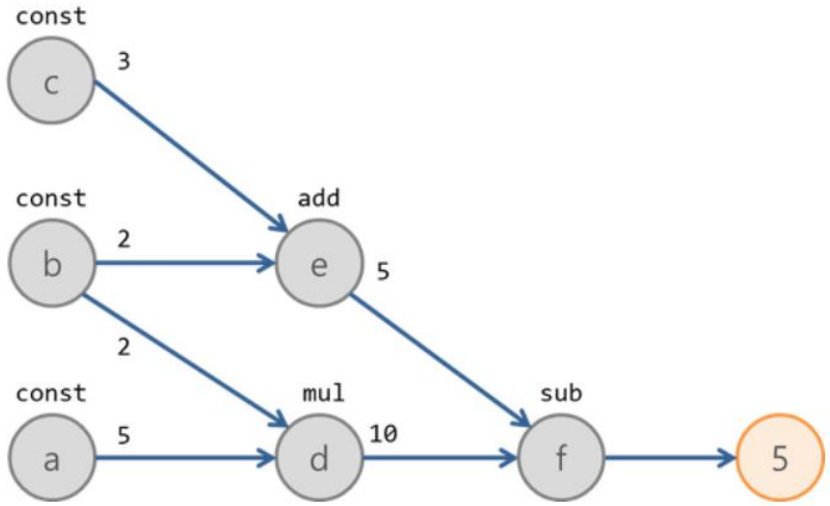
학습 절차

- 먼저, tf.Session()에서 그래프를 시작
- Session객체는 파이썬 객체와 데이터, 객체의 메모리가 할당되어 있는 실행 환경 사이를 연결하며, 중간 결과를 저장하고 최종 결과를 작업 환경으로 보내준다. 위의 코드에서는 Session 객체를 sess = tf.Session() 에 정의했다.
- 연산 그래프를 실행하려면 Session객체의 run() 메소드를 사용. 코드에서 sess.run(f)는 아래의 그림처럼 출력이 나와야 하는 f노드에서 시작해서 역방향으로 의존관계에 따라 노드의 연산을 수행
- 연산 수행이 완료되면 `sess.close()`를 통해 사용한 메모리를 해제

세션 실행되기 전

세션 실행된 후





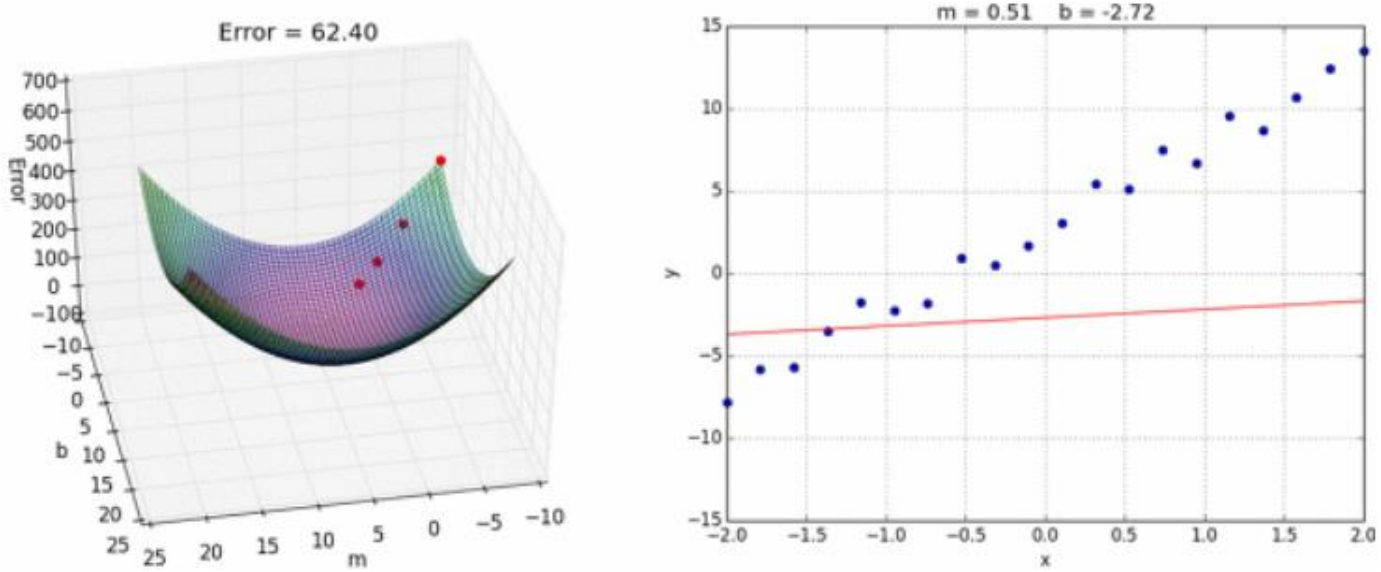
Source : https://excelsior-cjh.tistory.com/151

학습 절차

경사 하강법(Gradient Descent) 최적화 함수

딥러닝에서 손실함수를 최소화하는 다양한 최적화 알고리즘들이 있다. 가장 흔히 사용되는 알고리즘으로는 가중치의 집합에 대한 손실의 경사(기울기)를 이용한 **경사 하강법(gradient descent)**이 있다.

$$\mathrm{W} \leftarrow \mathrm{W} - \eta \frac{\partial L}{\partial \mathrm{W}}$$



(출처: Ghipy)

Source : https://excelsior-cjh.tistory.com/151

학습 절차

Linear Regression

Logistic Classification, sigmoid

Soft Max

Normalization, Xavier

Neural Network for MNIST

CNN

RNN

Excel 비교

Bias 큰 경우

TensorBoard

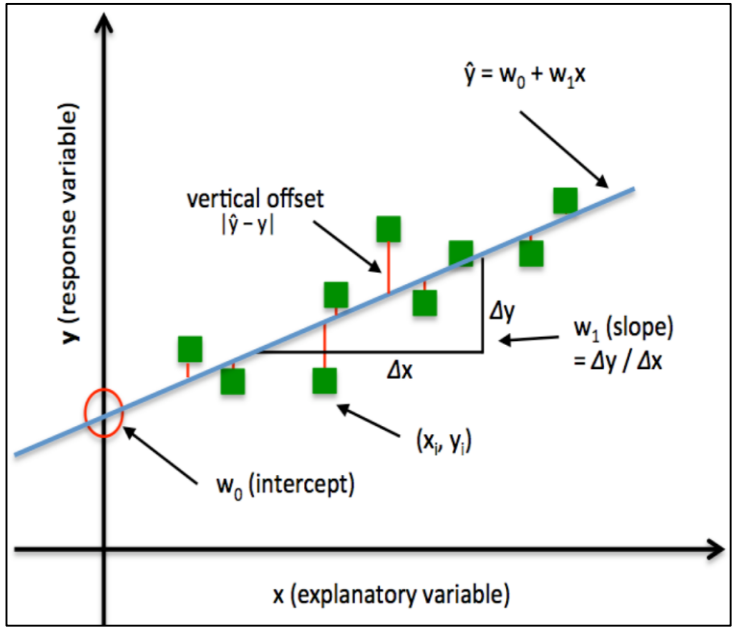
..

X 범위별 선형

Source :

Linear Regression

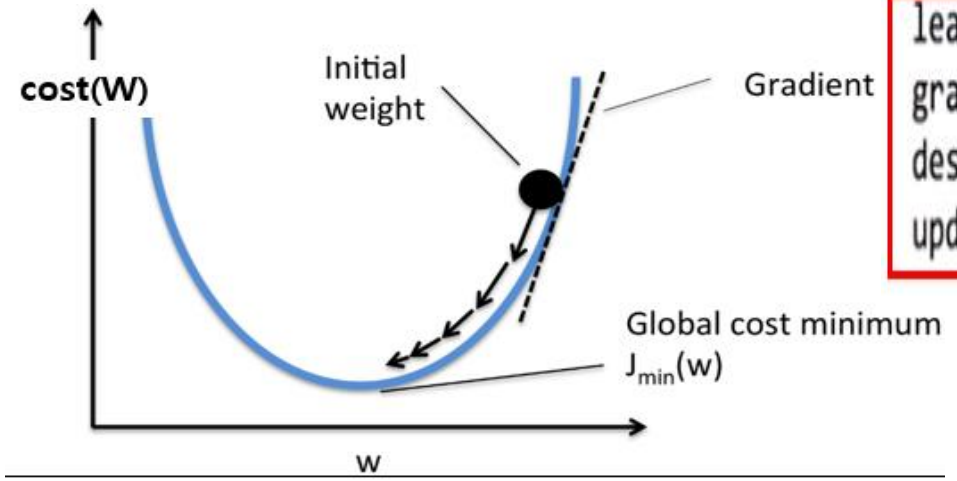
Hypothesis – Cost function



$$Cost(W) = \sum_{1}^{m} (Wx^i - y^i)^2$$

Optimization

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)}) x^{(i)}$$




```
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)
```

hypothesis = x_train * W + b

```
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize: Gradient Descent Magic
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
train = optimizer.minimize(cost)
```

Source :

linear regression

```
import tensorflow as tf

x_train = [1,2,3, 5]
y_train = [2, 4, 6, 10]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = x_train*W + b

cost = tf.reduce_mean(tf.square(hypothesis - y_train))


optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)

train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for step in range(201) :
    sess.run(train)
    if step % 20 ==0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

- **모델을 Test에 활용**
- **bias 가 큰 경우**

Source :

sigmoid

$$Odds(p) := \frac{p}{1-p}$$

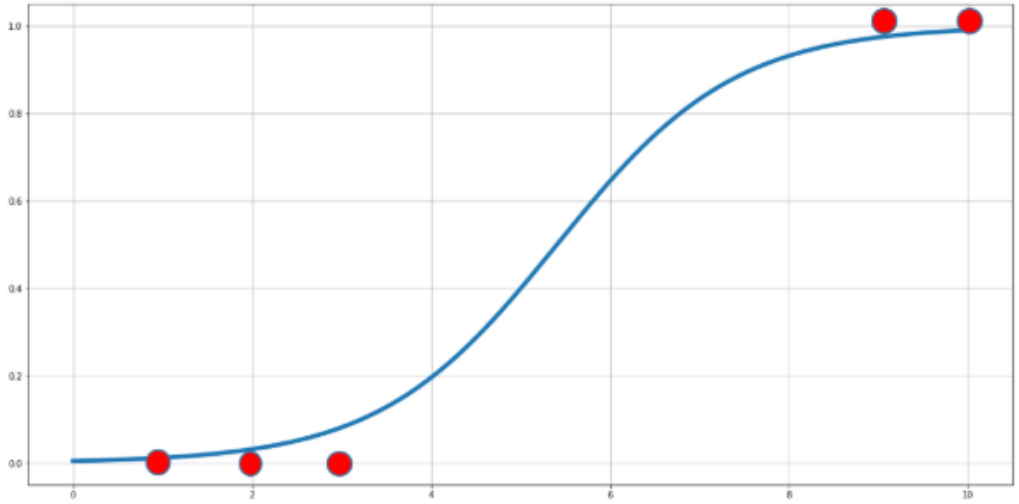
$lds(p)$의 범위는 (0,∞)이 된다. Od
이다. $\log(Odds(p))$의 범위가 실수

$$\log(Odds(p)) = wx + b$$

수 있다. 위 식을 p로 정리하면 다음과

$$p(x) = \frac{1}{1 + e^{-(wx+b)}}$$

x 데이터가 주어졌을 때 성공확률을 예측하는 로지스틱 회귀분석은 학습데이터를 잘 설명하는 시그모이드 함수의 w와 b를 찾는 문제다.



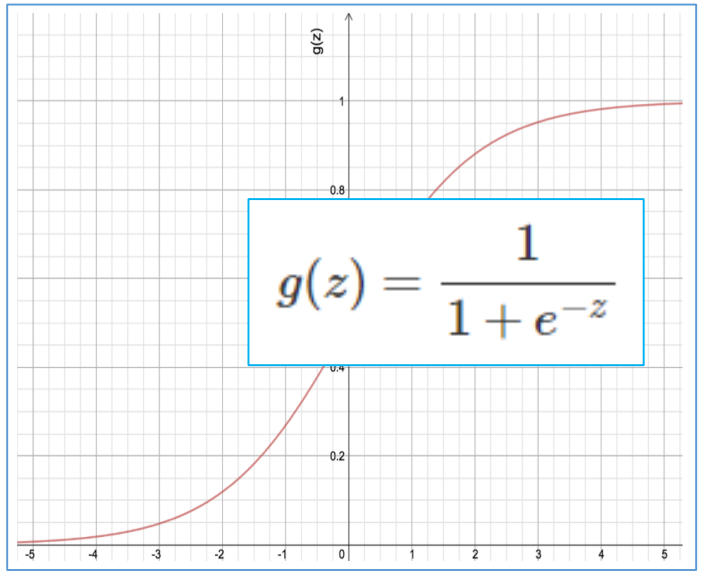
- $y = H(x) = \frac{1}{1+e^{-(wx+b)}}$
- w = 1.0072499
- b = − 5.4476051

Source : https://icim.nims.re.kr/post/easyMath/64

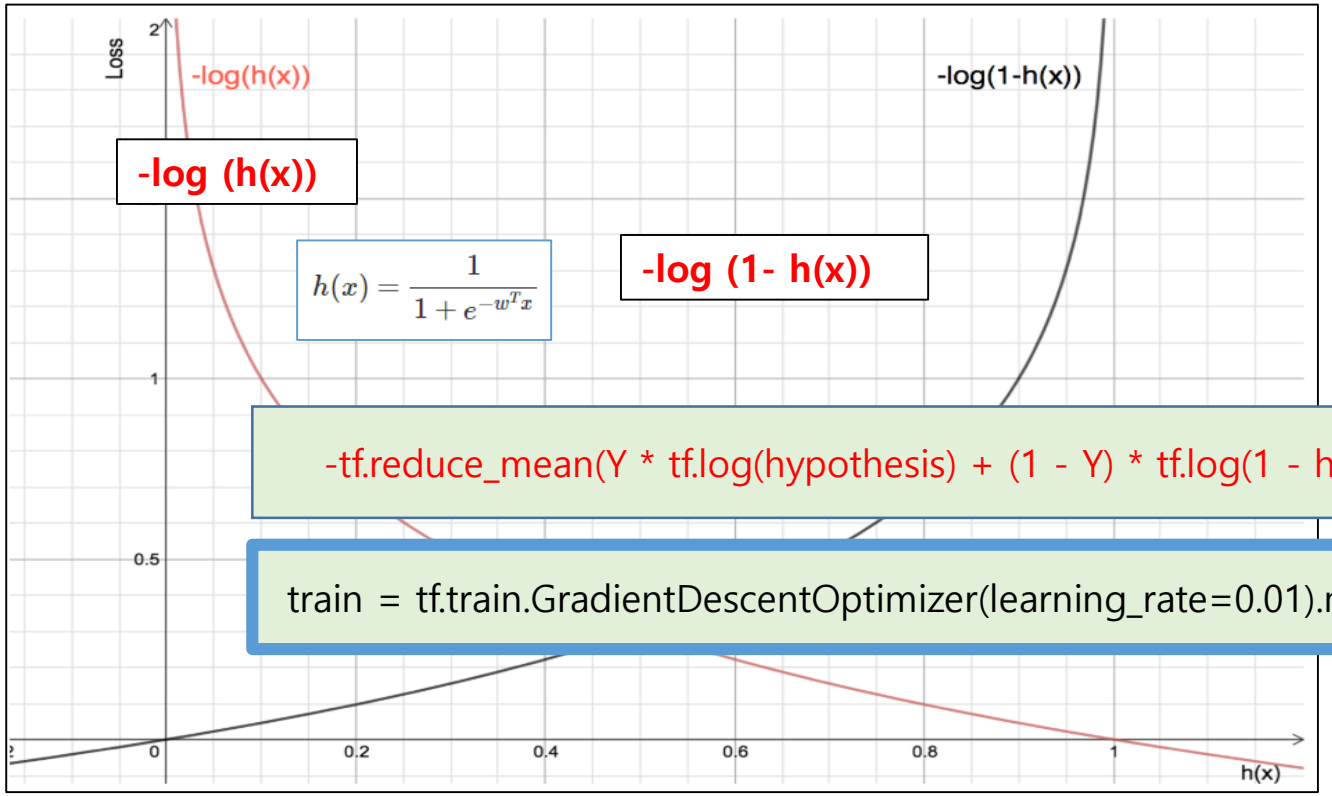
Logistic Classification

Hypothesis – Cost function

Optimization



$$L(h(x),y) = \begin{cases} -log(h(x)) & y=1 \\ -log(1-h(x)) & y=0 \end{cases} = L(h(x),y) = -ylog(h(x)) - (1-y)log(1-h(x))$$



tf.sigmoid(tf.matmul(X, W) + b)

-tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

Source :

Logistic Classification

z 계산

tf.sigmoid(tf.matmul(X, W) + b)

x data

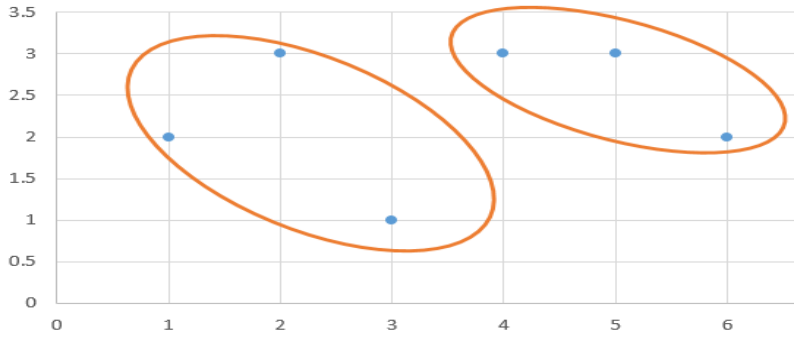
[[1, 2],
[2, 3],
[3, 1],
[4, 3],
[5, 3],
[6, 2]]

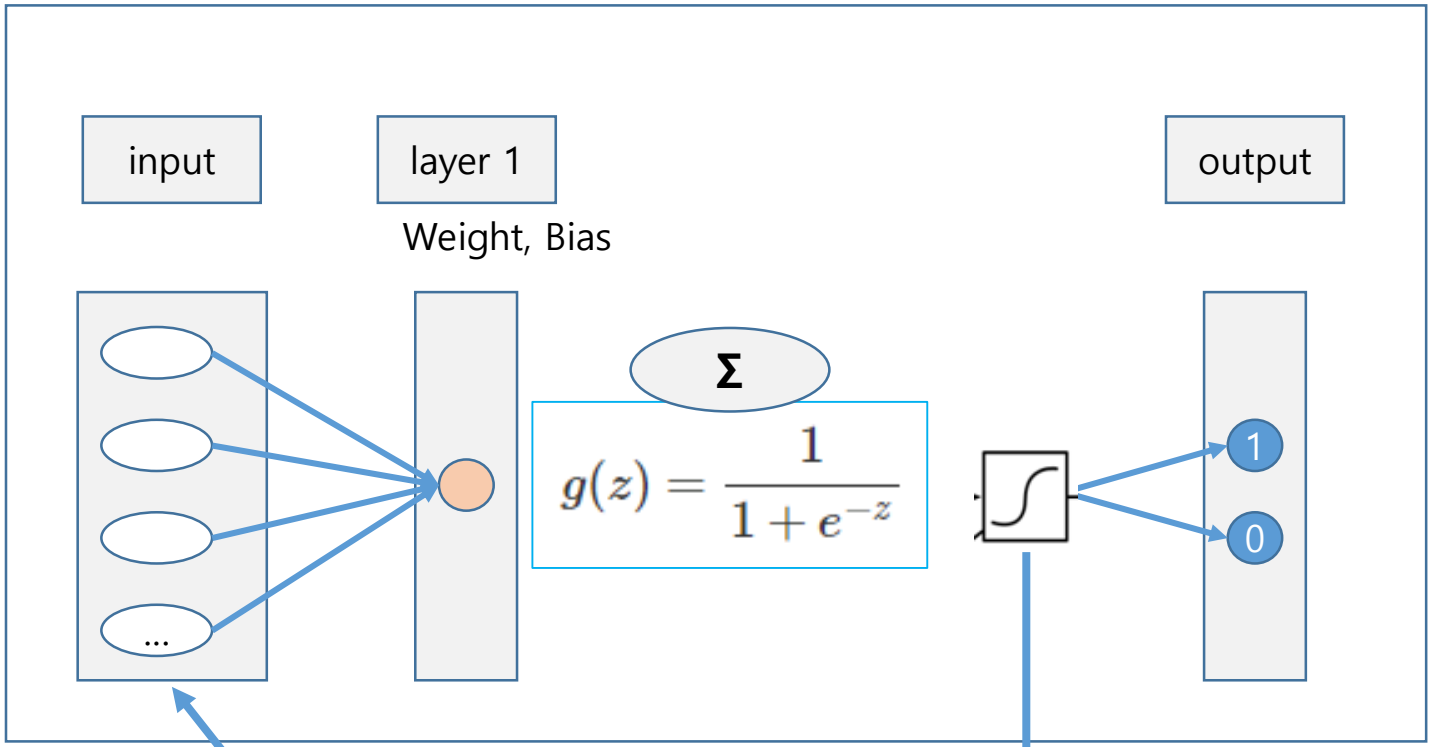
(6, 2)

W

[2]
[3]

(2,1)





Cost Function or Loss

Optimizer

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

Source :

logistic classification

```
import tensorflow as tf
tf.set_random_seed(777)  # for reproducibility

x_data = [[1, 2],  [2, 3],  [3, 1],  [4, 3],  [5, 3],  [6, 2]]
y_data = [[0],  [0],  [0],  [1],  [1],  [1]]

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 2])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([2, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)

cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))

train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

# Accuracy computation  /  # True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
```

```
# Launch graph

with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(1001):
        cost_val, _ = sess.run([cost, train], feed_dict={X: x_data, Y: y_data})
        if step % 200 == 0:
   print(step, cost_val)

    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy],
     feed_dict={X: x_data, Y: y_data})

    print("₩nHypothesis: ", h, "₩nCorrect (Y): ", c, "₩nAccuracy: ", a)
```

•

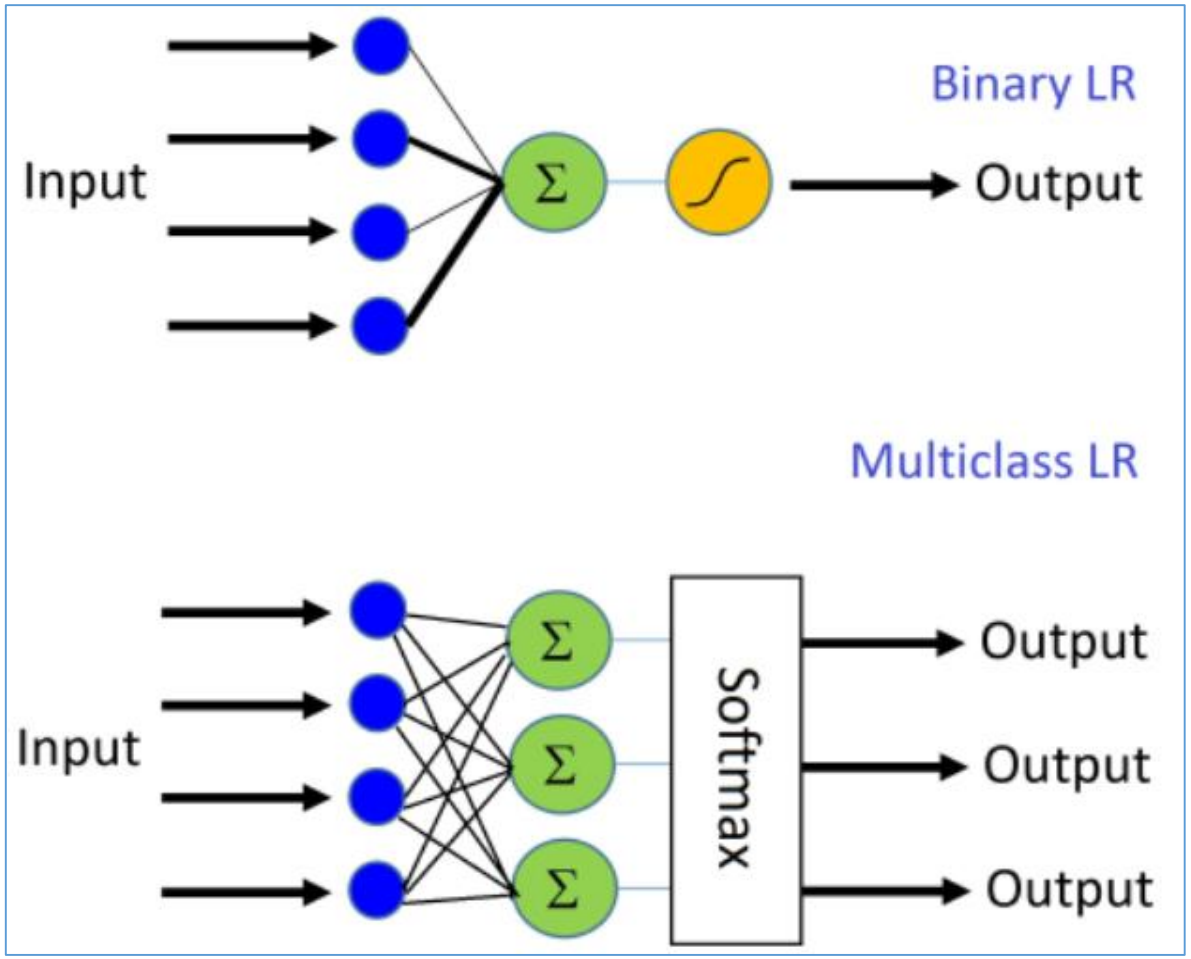
Q : cost function을 아래로 변경하여도 같은 결과인가 ?

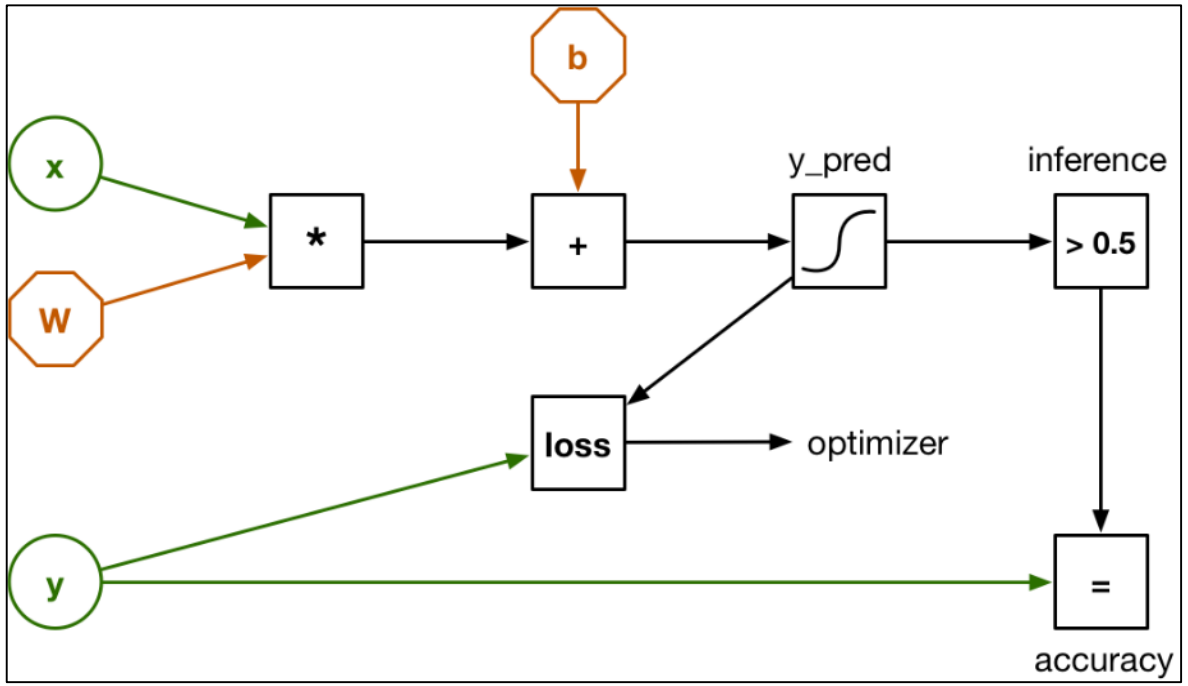
```
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

Source :

Softmax

- y 예측 값이 0~1 사이 여러 개이며,
- 합하면 1이 되는 함수 softmax



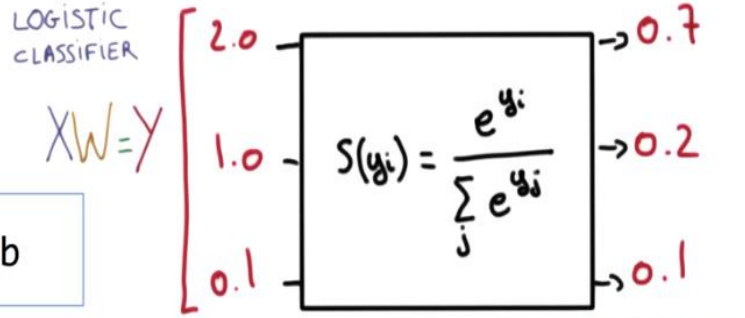


Source :

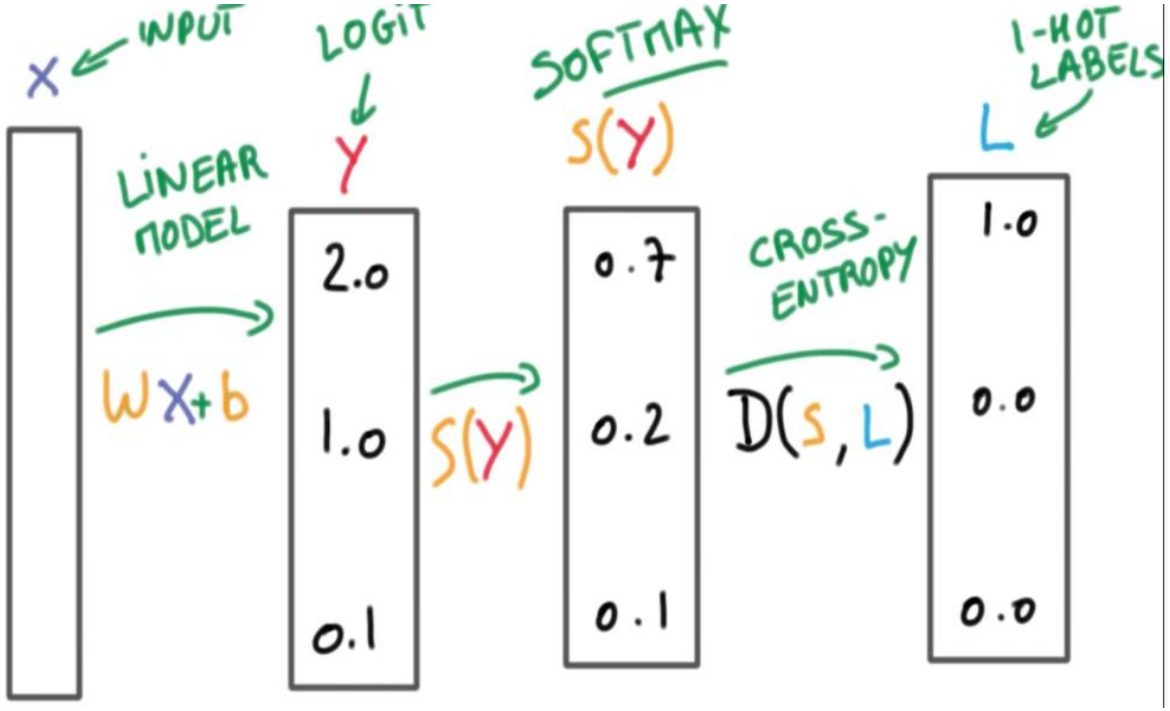
Softmax : logit을 받아서 sigmoid 한 후에 다시 전체 합 기준 각각의 비율 계산

z = tf.matmul (X, W) + B	$g(z) = \frac{1}{1+e^{-z}}$	Hypothesis	softmax
Input 값과 weight	tf.sigmoid(tf.matmul(X, W) + b)		tf.nn.softmax(tf.matmul(X,W)+b)

```
hypothesis = tf.nn.softmax(tf.matmul(X,W)+b)
```




```
tf.matmul(X,W)+b
```



Source :

Softmax cost function 비교

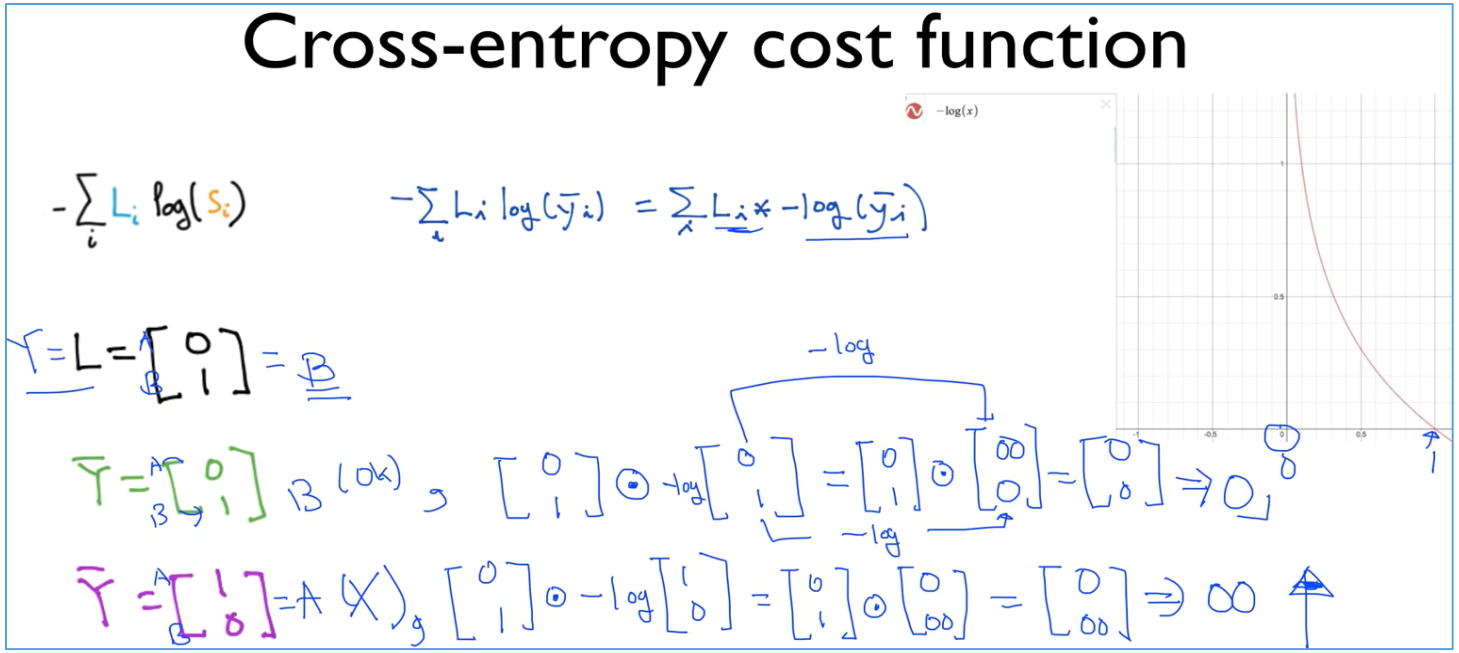
```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)
```

-tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

one hot

같은 이유 ?



Source :

Softmax

```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)
```

-tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))

1

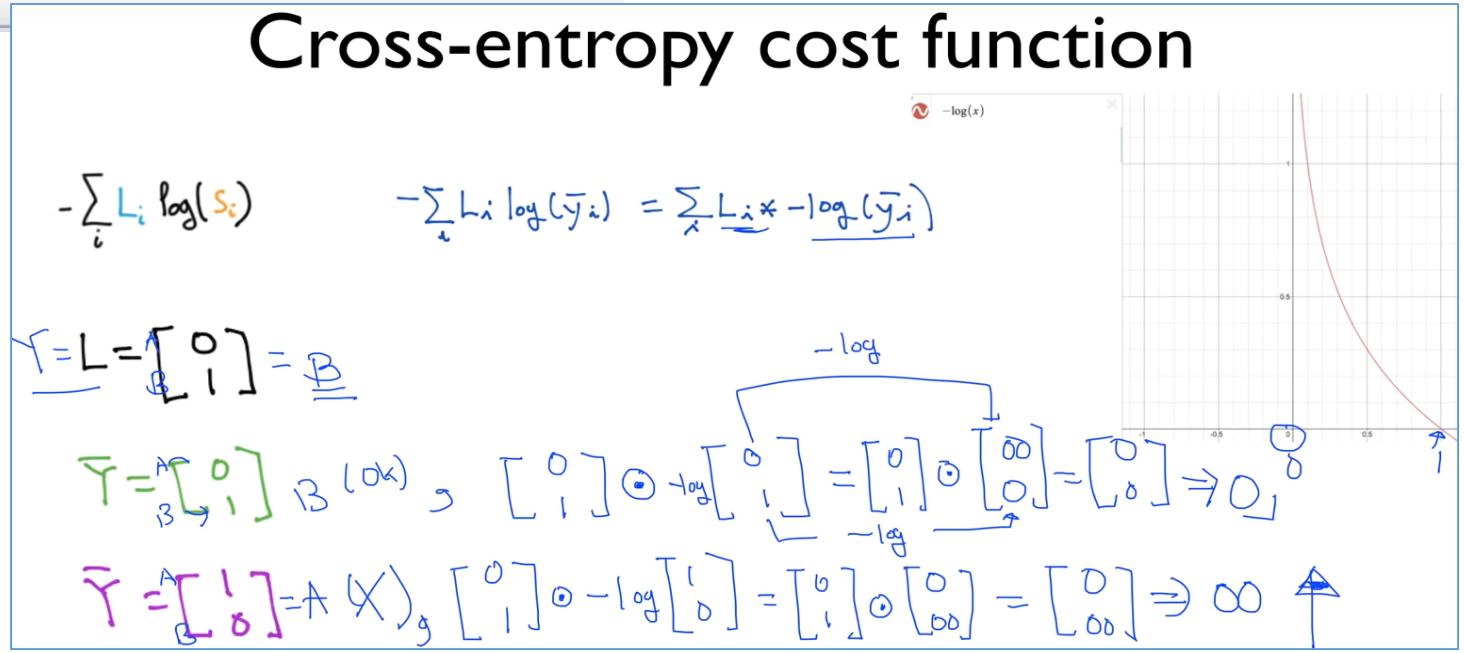
```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

one hot

2

```
# Cross entropy cost/loss
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                 labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
```

Cross-entropy cost function



Source :

Softmax

Hypothesis – Cost function

Optimization

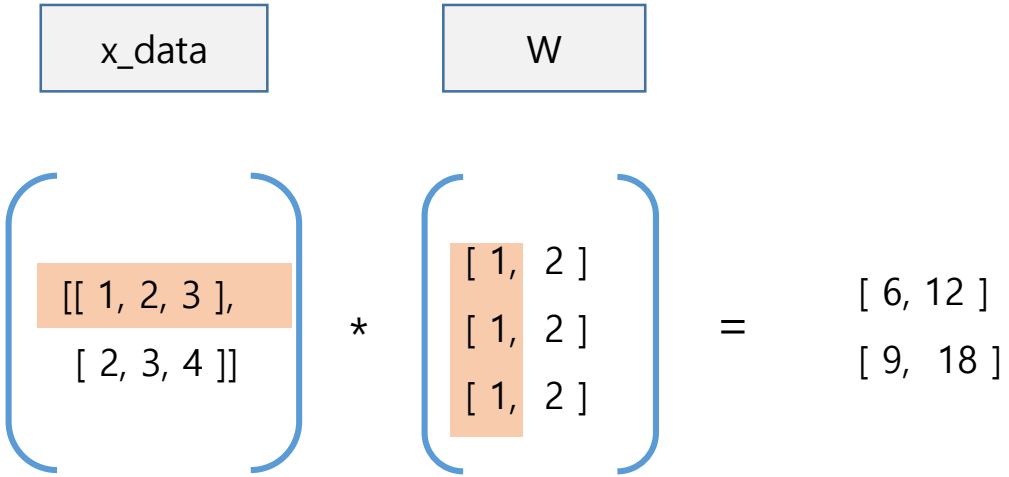
```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)
```

```
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                 labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
```

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

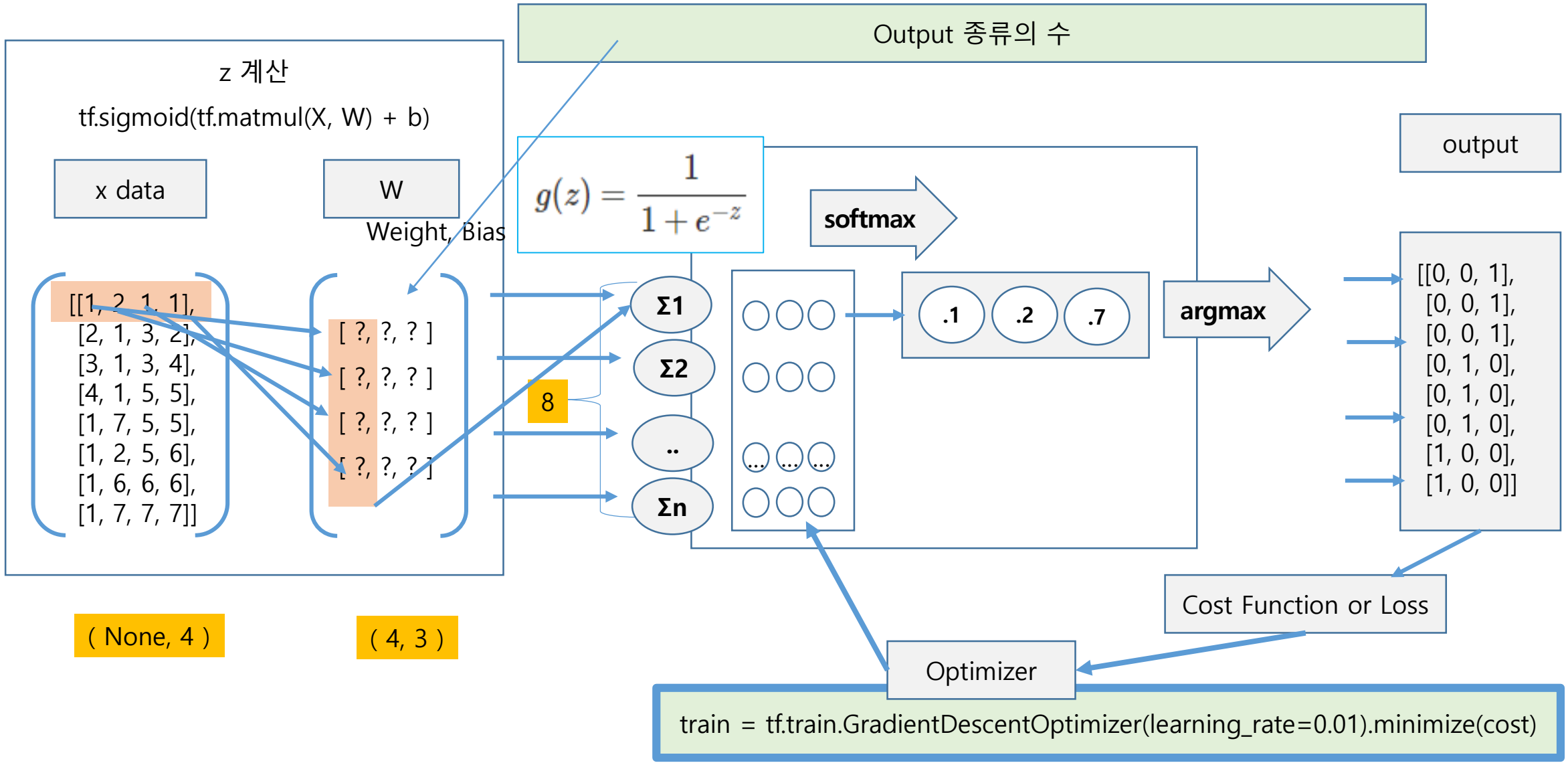
Source :

softmax & argmax



Source :

softmax & argmax



Source :

softmax & argmax

```
import tensorflow as tf
tf.set_random_seed(123)  # for reproducibility

x_data = [[1, 2, 1, 1],  [2, 1, 3, 2],  [3, 1, 3, 4],  [4, 1, 5, 5], [1, 7, 5, 5], [1, 2, 5, 6],  [1, 6, 6, 6],  [1, 7, 7, 7]]
y_data = [[0, 0, 1],  [0, 0, 1],  [0, 0, 1],  [0, 1, 0],  [0, 1, 0],  [0, 1, 0],  [1, 0, 0],  [1, 0, 0]]

X = tf.placeholder("float", [None, 4])
Y = tf.placeholder("float", [None, 3])
nb_classes = 3

W = tf.Variable(tf.random_normal([4, nb_classes]), name='weight')
b = tf.Variable(tf.random_normal([nb_classes]), name='bias')

# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)

hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)

# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

optimizer =
tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

```
# Launch graph
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for step in range(2001):
   _, cost_val = sess.run([optimizer, cost], feed_dict={X: x_data, Y: y_data})

   if step % 200 == 0:
       print(step, cost_val)

    print('--------------')
    # Testing & One-hot encoding
    a = sess.run(hypothesis, feed_dict={X: [[1, 11, 7, 9]]})
    print(a, sess.run(tf.argmax(a, 1)))

    print('--------------')
    b = sess.run(hypothesis, feed_dict={X: [[1, 3, 4, 3]]})

    print(b, sess.run(tf.argmax(b, 1)))

    print('--------------')
    c = sess.run(hypothesis, feed_dict={X: [[1, 1, 0, 1]]})
    print(c, sess.run(tf.argmax(c, 1)))

    print('--------------')
    all = sess.run(hypothesis, feed_dict={X: [[1, 11, 7, 9], [1, 3, 4, 3], [1, 1, 0, 1]]})
    print(all, sess.run(tf.argmax(all, 1)))
```

Source :

NN for MNIST

모델

```
import tensorflow as tf
import matplotlib.pyplot as plt
import random

from tensorflow.examples.tutorials.mnist import input_data

tf.set_random_seed(123)  # reproducibility

# Check out https://www.tensorflow.org/get_started/mnist/beginners
for
# more information about the mnist dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# weights & bias for nn layers
W = tf.Variable(tf.random_normal([784, 10]))
b = tf.Variable(tf.random_normal([10]))

# parameters
learning_rate = 0.001
batch_size = 100
num_epochs = 10
num_iterations = int(mnist.train.num_examples / batch_size)
```

```
hypothesis = tf.matmul(X, W) + b

# define cost/loss & optimizer
cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits_v2(
        logits=hypothesis, labels=tf.stop_gradient(Y)
    )
)
train = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

correct_prediction = tf.equal(tf.argmax(hypothesis, axis=1), tf.argmax(Y, axis=1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# train my model
with tf.Session() as sess:
    # initialize
    sess.run(tf.global_variables_initializer())

    for epoch in range(num_epochs):
        avg_cost = 0

        for iteration in range(num_iterations):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            _, cost_val = sess.run([train, cost], feed_dict={X: batch_xs, Y: batch_ys})
            avg_cost += cost_val / num_iterations
```

Source :

linear regression – logistic – softmax classfication

	linear regression	logistic classification	softmax
hypothesis	= x_train * W + b	g(z) = 1 / (1+exp –z) hypothesis = tf.sigmoid(tf.matmul(X, W) + b)	logits = tf.matmul(X, W) + b hypothesis = tf.nn.softmax(logits)
cost	= tf.reduce_mean (tf.square(hypothesis – y_train))	= -y log(hx) – (1-y)log(1-hx)	cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
			cost_i = tf.nn.softmax_cross_entropy _with_logits_v2(logits=logits, labels=tf.stop_gradient ([Y_one_hot])) cost = tf.reduce_mean(cost_i)
optimizer & training	optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1) train = optimizer.minimize(cost)	train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)	

Source : 320. ML_algorithm_code_summary

NN, ReLu, Xavier, Dropout, and Adam

softmax classifier for MNIST : 90 %

softmax classifier for MNIST : 90 %

NN for MNIST : 94 %

```
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# weights & bias for nn layers
W1 = tf.Variable(tf.random_normal([784, 256]))
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([256, 256]))
b2 = tf.Variable(tf.random_normal([256]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)

W3 = tf.Variable(tf.random_normal([256, 10]))
b3 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L2, W3) + b3

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entro
    logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rat
```

hypothesis = tf.matmul(L2, W3) + b3

Xavier for MNIST : 97.8 %

```
# weights & bias for nn layers
# http://stackoverflow.com/questions/33640581
W1 = tf.get_variable("W1", shape=[784, 256],
                     initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
```

Deep NN for MNIST : 97 %
Overfitting problem

Dropout for MNIST : 98 %

```
W1 = tf.get_variable("W1", shape=[784, 512])
b1 = tf.Variable(tf.random_normal([512]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)

W2 = tf.get_variable("W2", shape=[512, 512])
b2 = tf.Variable(tf.random_normal([512]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)
L2 = tf.nn.dropout(L2, keep_prob=keep_prob)
...
# train my model
for epoch in range(training_epochs):
```

Source : https://youtu.be/6CCXyfvubvY

LR w/ Xavier, Normalized

softmax classifier for MNIST : 90 %

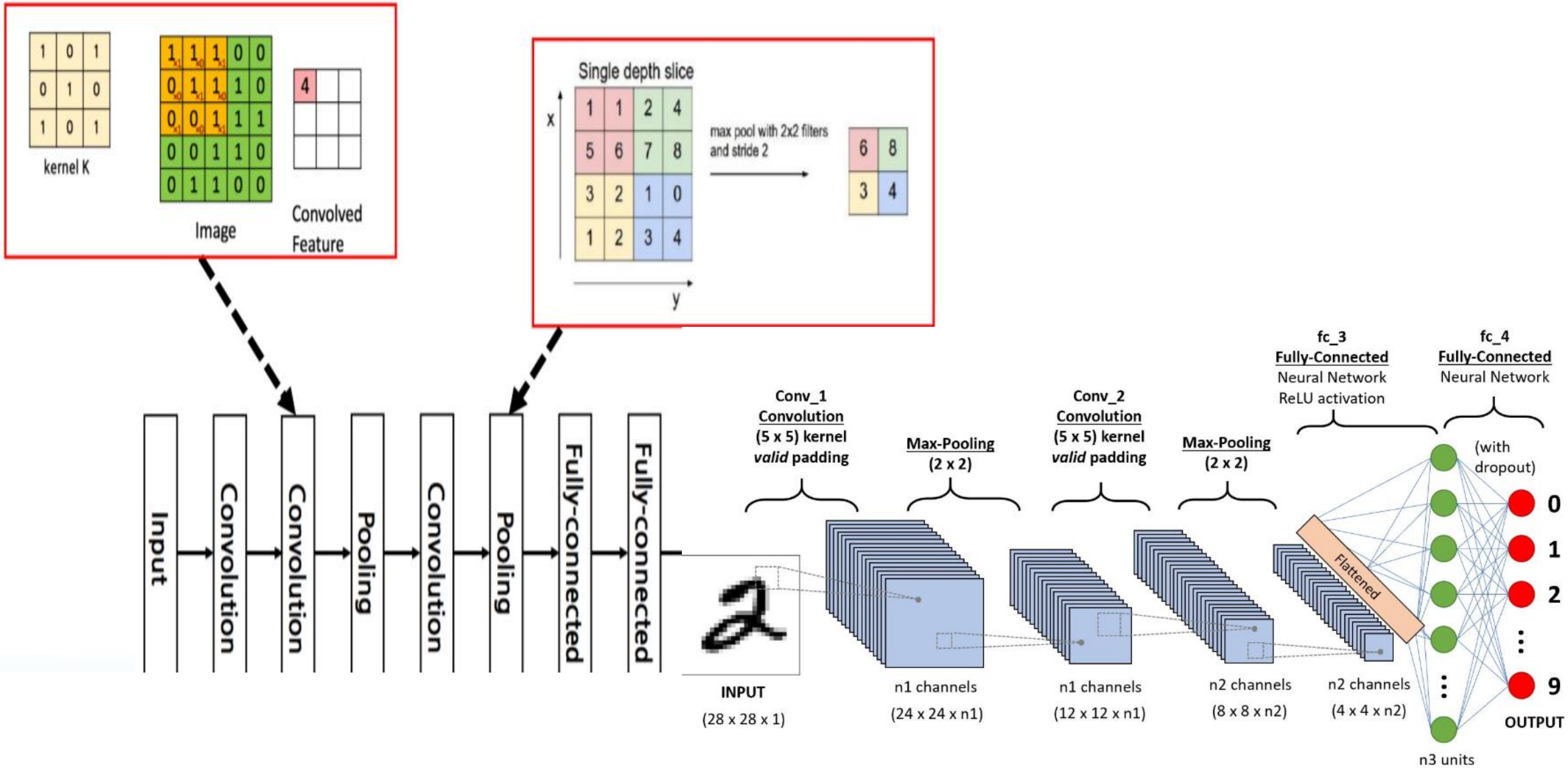
Xavier for

```
# weights & bias for nn layers
# http://stackoverflow.com/questions/33640581
W1 = tf.get_variable("W1", shape=[784, 256],
                     initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.Variable(tf.random_normal([256]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)
```

```
hypothesis = tf.matmul(L2, W3) + b3
```

Source :

CNN

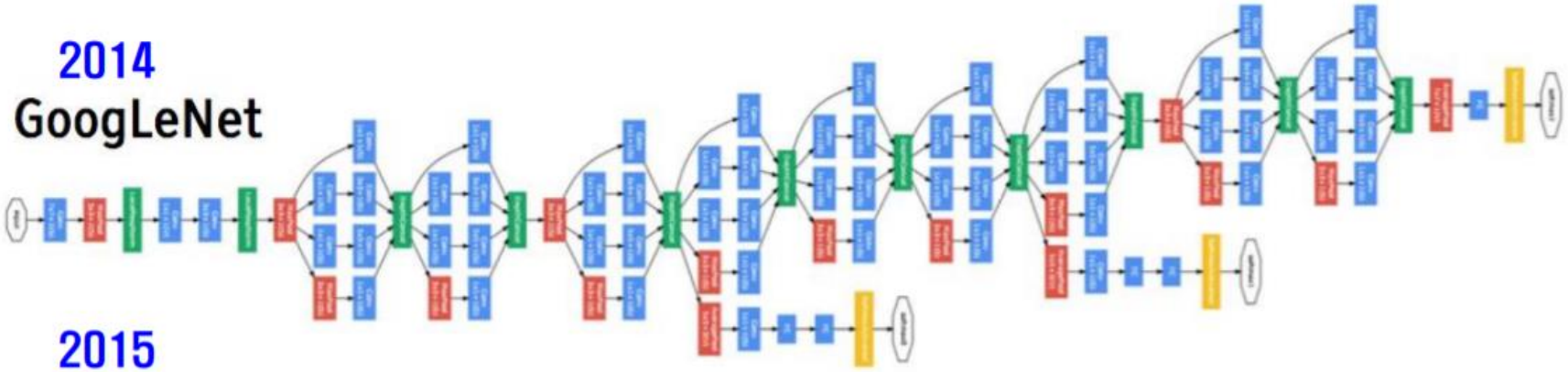

kernel K
Image
Convolved Feature
Single depth slice
max pool with 2x2 filters and stride 2
Input
Convolution
Convolution
Pooling
Convolution
Pooling
Fully-connected
Fully-connected
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

Source :

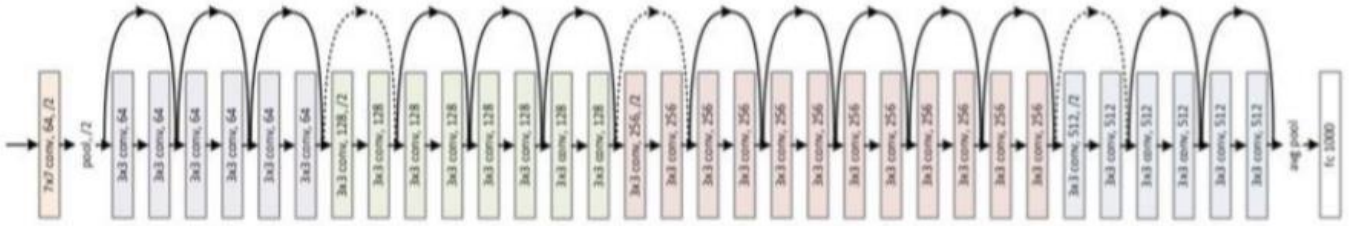
Popular CNN Architectures

2014
VGG

2014
GoogLeNet



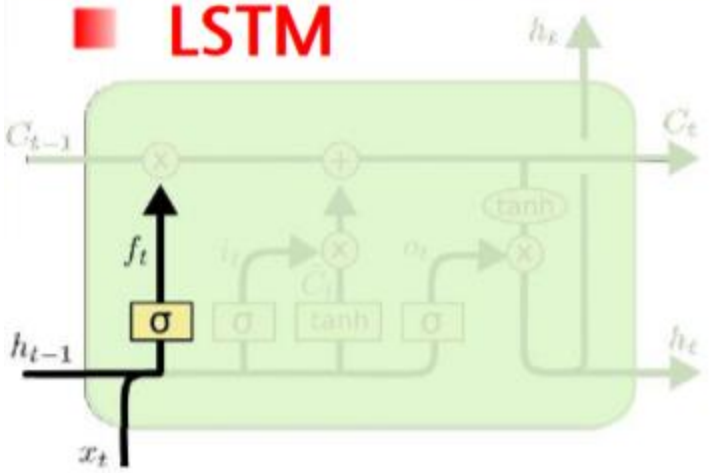
2015
ResNet



Source :

2-3 LSTM(Long Short-Term Memory)

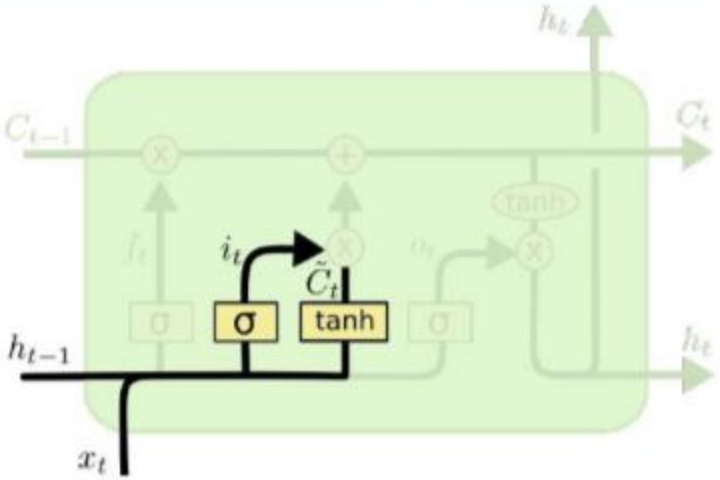
LSTM



<Forget gate>

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Concatenate



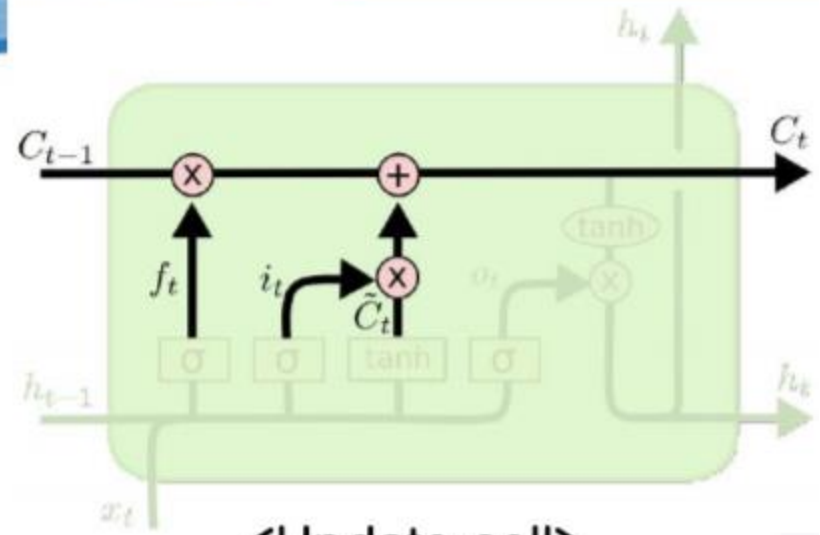
<Input gate>

Input Gate Layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

New contribution to cell state

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$ 기존 neuron과 동일함



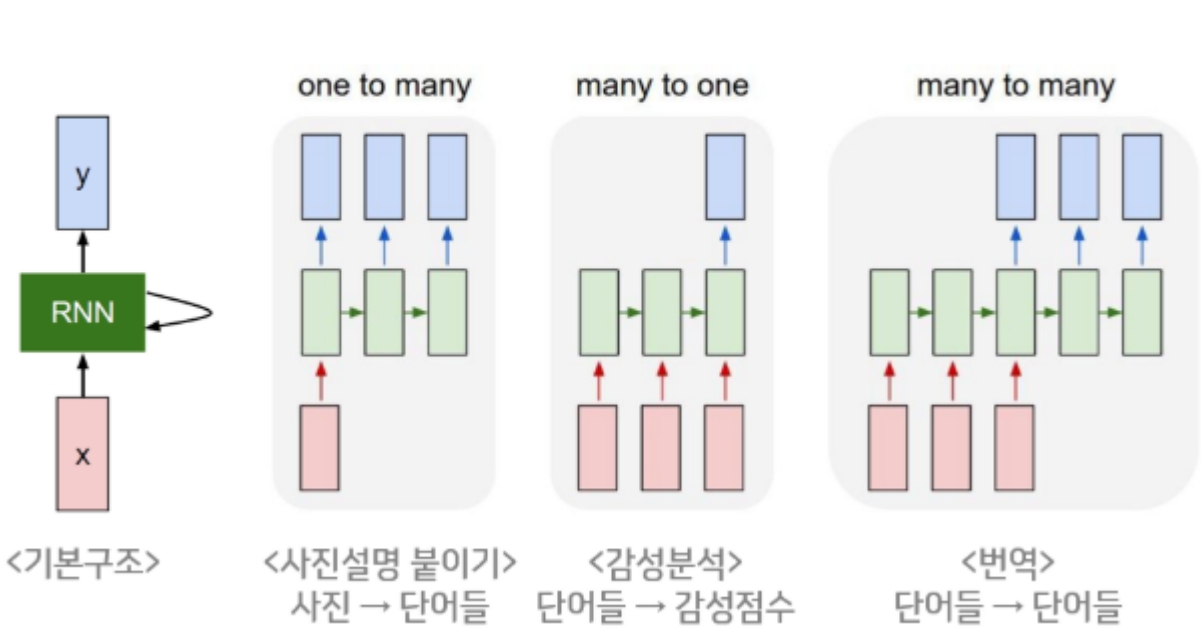
<Update cell>

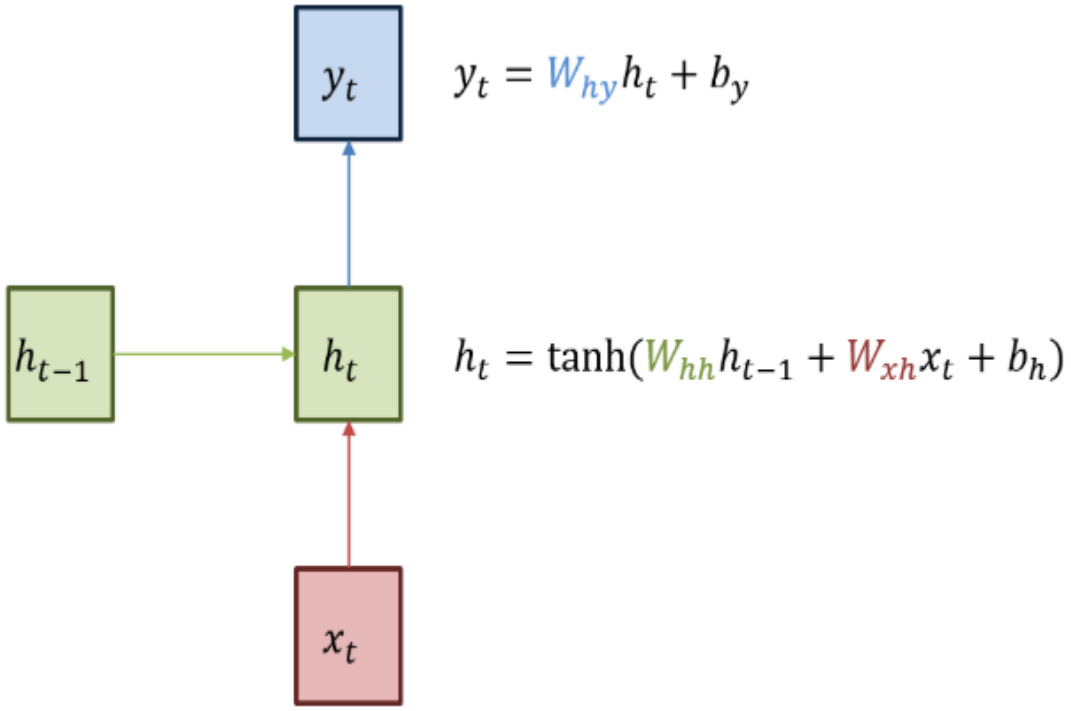
Update Cell State (memory):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source : http://infosec.pusan.ac.kr/wp-content/uploads/2017/11/CNN-and-RNN-%EC%9D%B4%EB%A1%A0.pdf

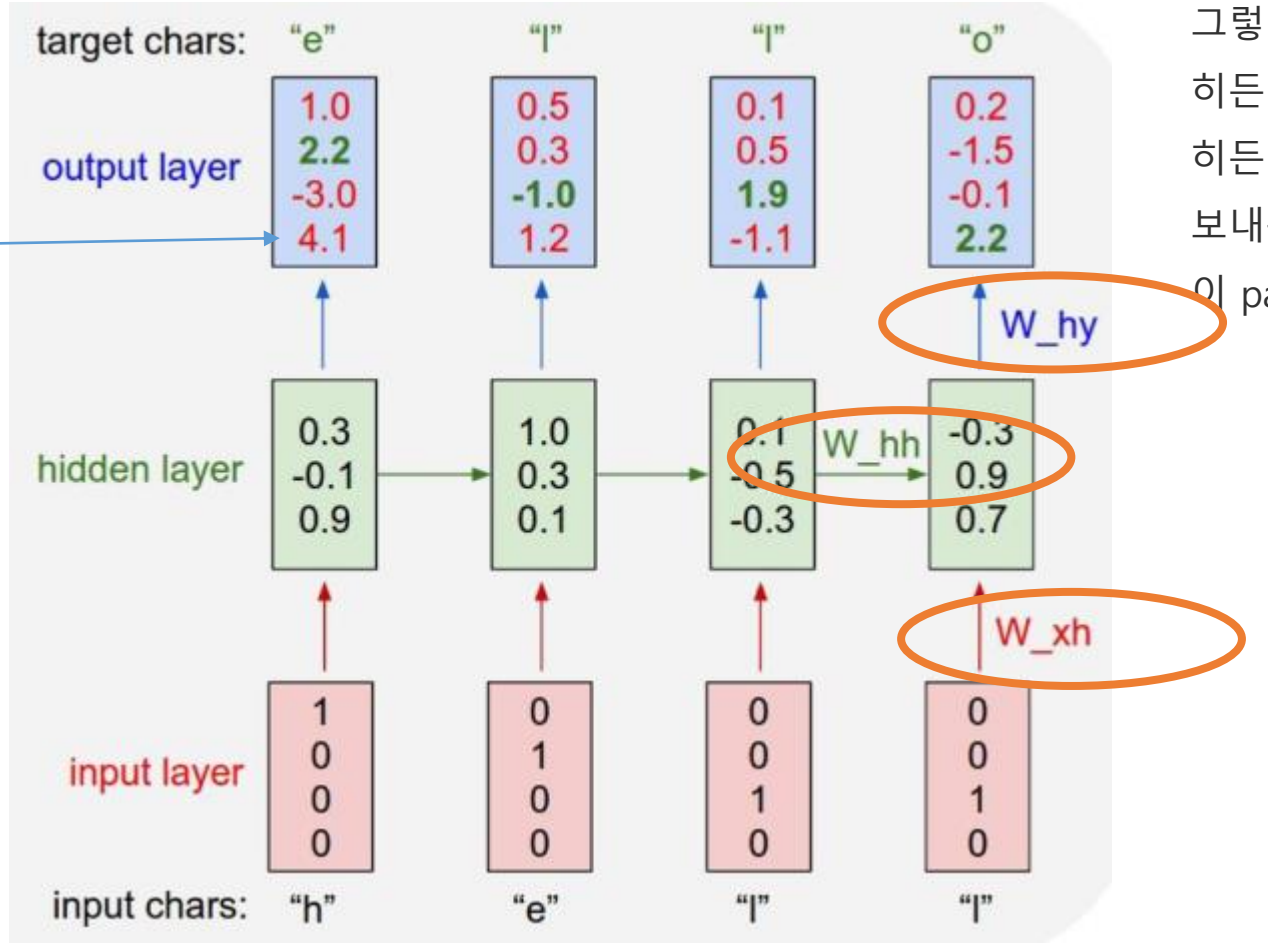
RNN의 기본 구조





Source : https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

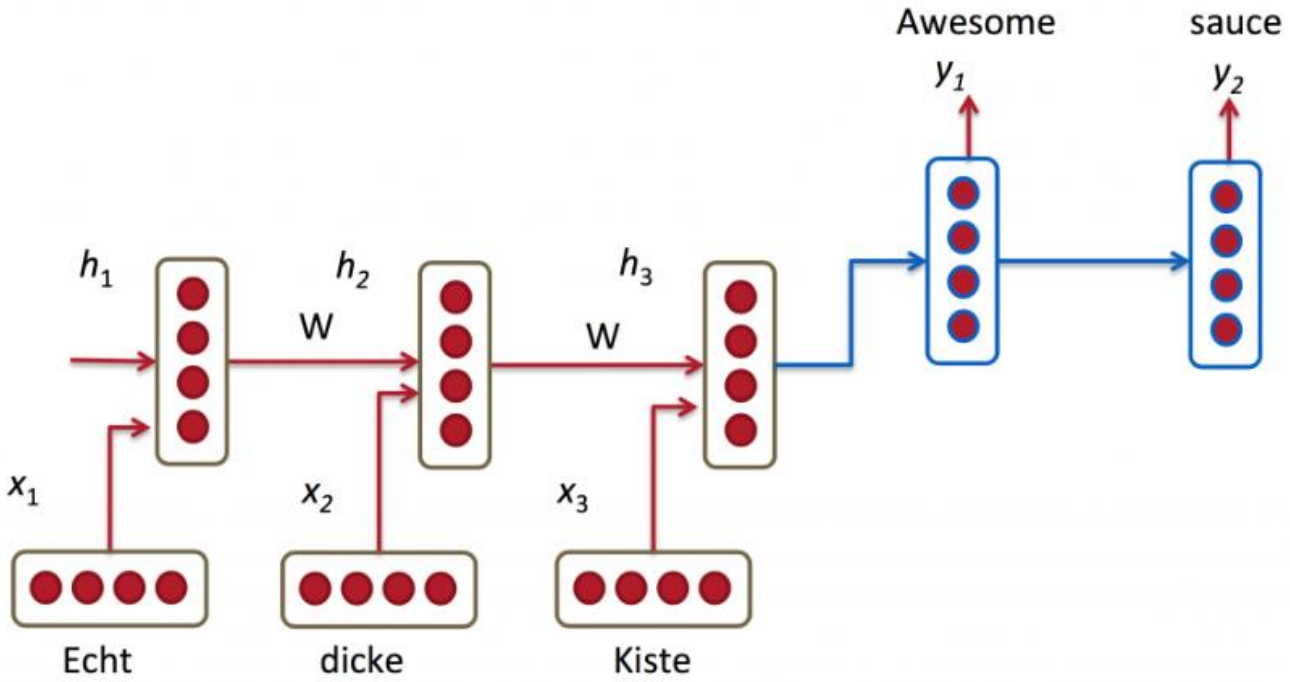
RNN



그렇다면 RNN이 학습하는 parameter는 무엇일까요? 인풋 xx를 히든레이어 hh로 보내는 WxhWxh, 이전 히든레이어 hh에서 다음 히든레이어 hh로 보내는 WhhWhh, 히든레이어 hh에서 아웃풋 yy로 보내는 WhyWhy가 바로 parameter입니다. 그리고 모든 시점의 state에서 이 parameter는 동일하게 적용됩니다(**shared weights**).

Source : https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

자동 번역 (기계 번역)

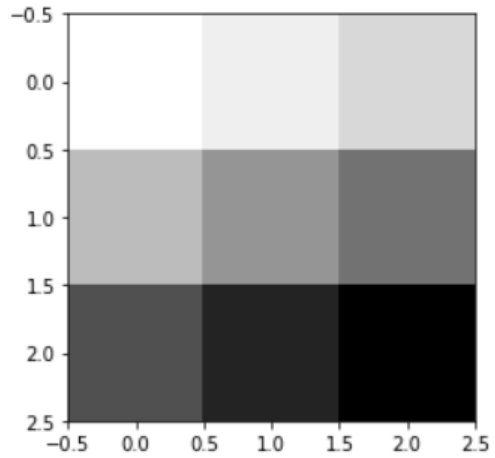


Source https://aikorea.org/blog/rnn-tutorial-1/

CNN basic

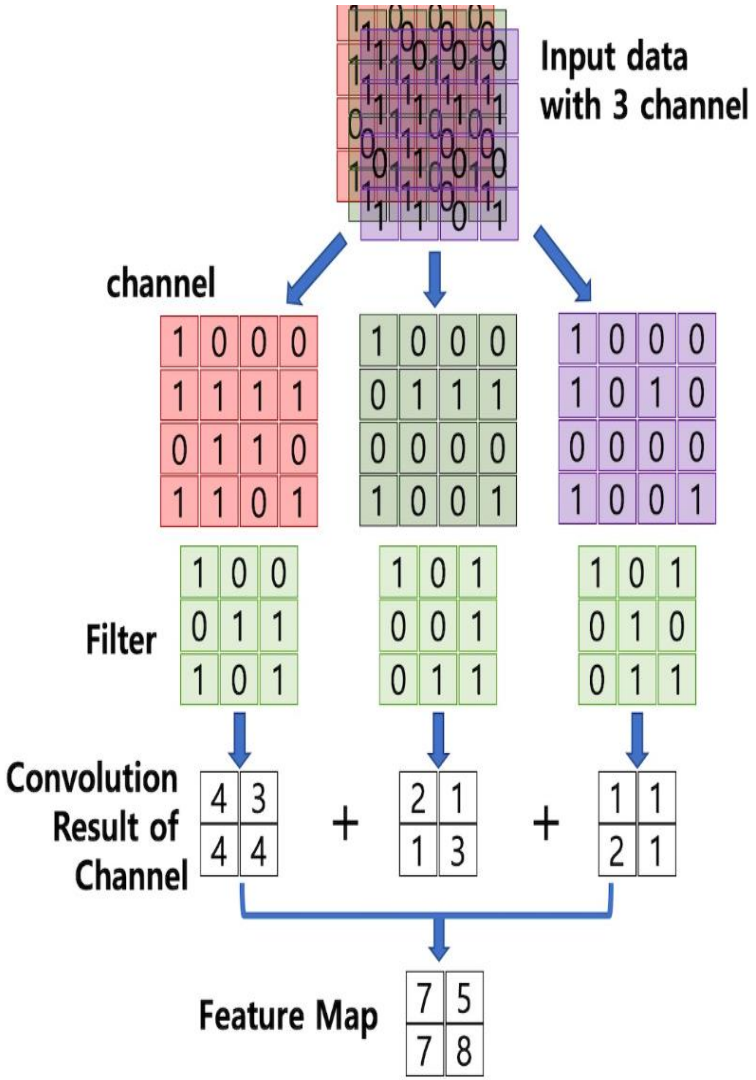
```
%matplotlib inline
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

sess = tf.InteractiveSession()
image = np.array([[[[1],[2],[3]],
                   [[4],[5],[6]],
                   [[7],[8],[9]]]], dtype=np.float32)
print(image.shape)
plt.imshow(image.reshape(3,3), cmap='Greys')
```

```
(1, 3, 3, 1)
<matplotlib.image.AxesImage at 0x1e6e4dc7f
```



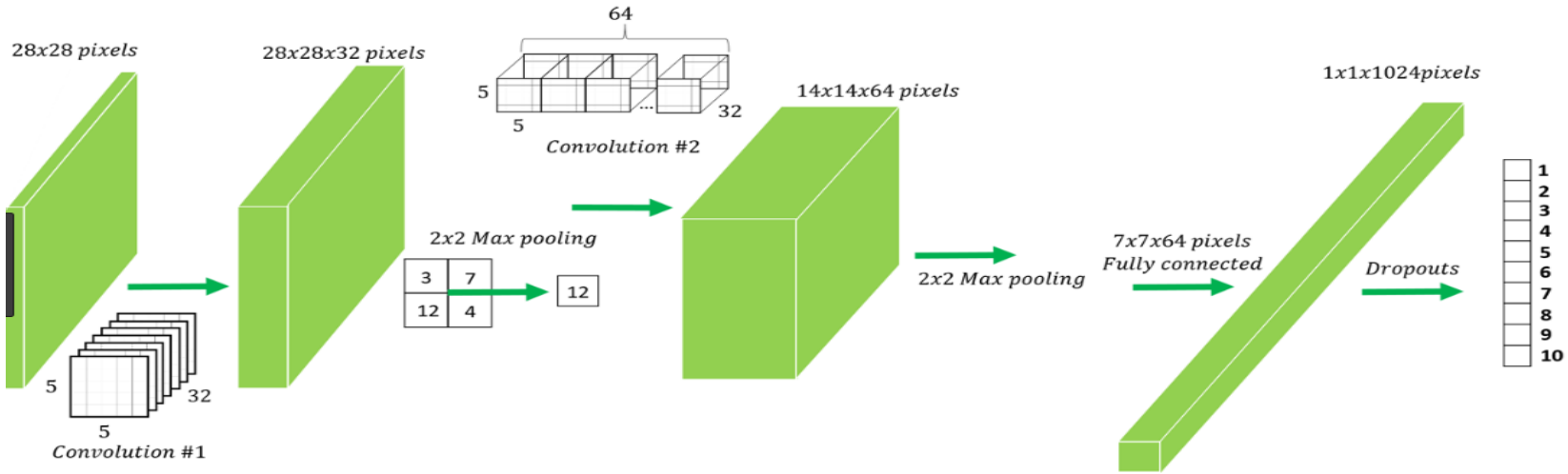
Source :

CNN Basics



컬러 사진은 천연색을 표현 위해, 각 픽셀을 RGB 3개의 실수로 표현한 3차원 데이터 = 컬러 이미지는 3개의 채널

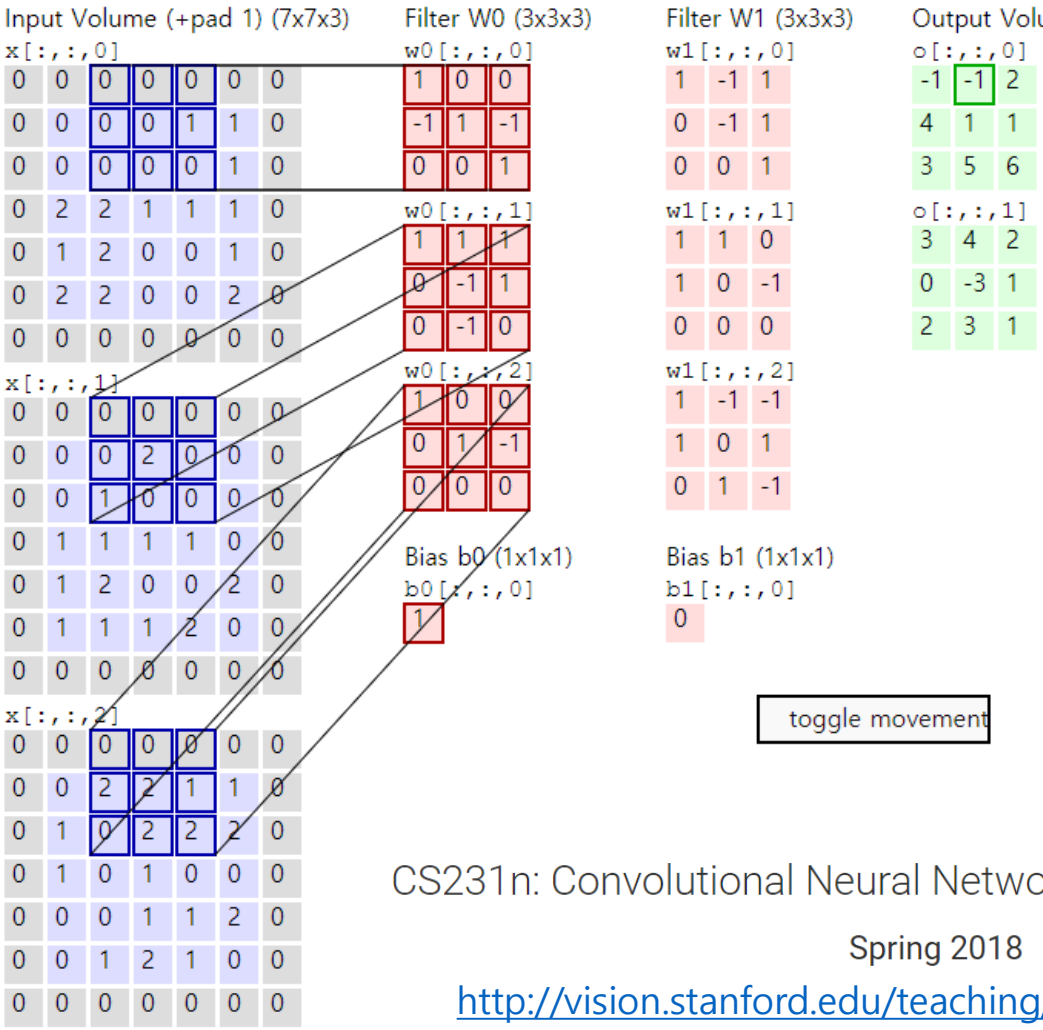
- Convolution(합성곱)
- 채널(Channel)
- 필터(Filter) = 커널
- 커널(Kernel)
- 스트라이드(Strid)
- 패딩(Padding)
- 피처 맵(Feature Map)
- 액티베이션 맵(Activation Map)
- 풀링(Pooling) 레이어



Source : http://taewan.kim/post/cnn/ ** https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html

CNN Basics

stride 2, conv = weight filter 계산 값의 합산, pooling = 가장 큰 값의 추출



-1 + (-2) + (2 -1) + 1 = -1

Table of Contents:

- Architecture Overview
- ConvNet Layers
  - Convolutional Layer
  - Pooling Layer
  - Normalization Layer
  - Fully-Connected Layer
  - Converting Fully-Connected Layers to Convolutional Layers
- ConvNet Architectures
  - Layer Patterns
  - Layer Sizing Patterns
  - Case Studies (LeNet / AlexNet / ZFNet / GoogLeNet / VGGNet)
  - Computational Considerations
- Additional References

Convolutional Neural Networks (CNNs / ConvNets)

CS231n: Convolutional Neural Networks for Visual Recognition

Spring 2018

http://vision.stanford.edu/teaching/cs231n/

Source : ** http://cs231n.github.io/convolutional-networks/

CS231n Convolutional Neural Networks for Visual Recognition

CNN : it takes too much time

```
import tensorflow as tf
import numpy as np
import random
# import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input_data

tf.set_random_seed(777)  # reproducibility

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
# Check out https://www.tensorflow.org/get_started/mnist/beginners for
# more information about the mnist dataset

# hyper parameters
learning_rate = 0.001
training_epochs = 1  # 15
batch_size = 100

# input place holders
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1])   # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 10])

# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv      -> (?, 28, 28, 32)
#    Pool      -> (?, 14, 14, 32)
```

```
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='SAME')

'''
Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
'''

# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv      ->(?, 14, 14, 64)
#    Pool      ->(?, 7, 7, 64)
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L2_flat = tf.reshape(L2, [-1, 7 * 7 * 64])
'''
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
'''
```

Source :

CNN : it takes too much time

```
# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv      ->(?, 14, 14, 64)
#    Pool      ->(?, 7, 7, 64)
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L2_flat = tf.reshape(L2, [-1, 7 * 7 * 64])
'''
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
'''
# Final FC 7x7x64 inputs -> 10 outputs
W3 = tf.get_variable("W3", shape=[7 * 7 * 64, 10],
                     initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.random_normal([10]))
logits = tf.matmul(L2_flat, W3) + b

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

```
# train my model
print('Learning started. It takes sometime.')
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')

# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy:', sess.run(accuracy, feed_dict={
      X: mnist.test.images, Y: mnist.test.labels}))

# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label: ", sess.run(tf.argmax(mnist.test.labels[r:r + 1], 1)))
print("Prediction: ", sess.run(
    tf.argmax(logits, 1), feed_dict={X: mnist.test.images[r:r + 1]}))

# plt.imshow(mnist.test.images[r:r + 1].
#            reshape(28, 28), cmap='Greys', interpolation='nearest')
# plt.show()
```

Source :

```
Learning started. It takes sometime.
Epoch: 0001 cost = 0.381484106
Epoch: 0002 cost = 0.092666151
Epoch: 0003 cost = 0.066978173
Epoch: 0004 cost = 0.055014879
Epoch: 0005 cost = 0.046633794
Epoch: 0006 cost = 0.039136048
Epoch: 0007 cost = 0.033794922
Epoch: 0008 cost = 0.029938363
Epoch: 0009 cost = 0.026493469
Epoch: 0010 cost = 0.022644098
Learning Finished!
Accuracy: 0.9878
Label:  [6]
Prediction:  [6]
```

```
Learning started. It takes sometime.
Epoch: 0001 cost = 0.360349698
Epoch: 0002 cost = 0.091666671
Epoch: 0003 cost = 0.069225828
Epoch: 0004 cost = 0.057063460
Epoch: 0005 cost = 0.048663615
Learning Finished!
Accuracy: 0.9855
Label:  [0]
Prediction:  [0]
```

4분 소요

Source :

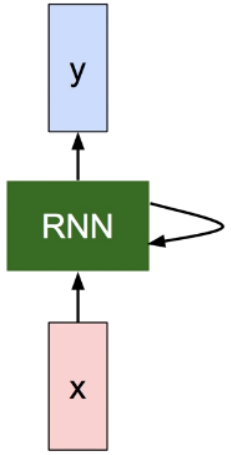
Basics

$$h_t = f_W(h_{t-1}, x_t)$$

new state / old state input

(Vanilla) Recurrent Neural Network

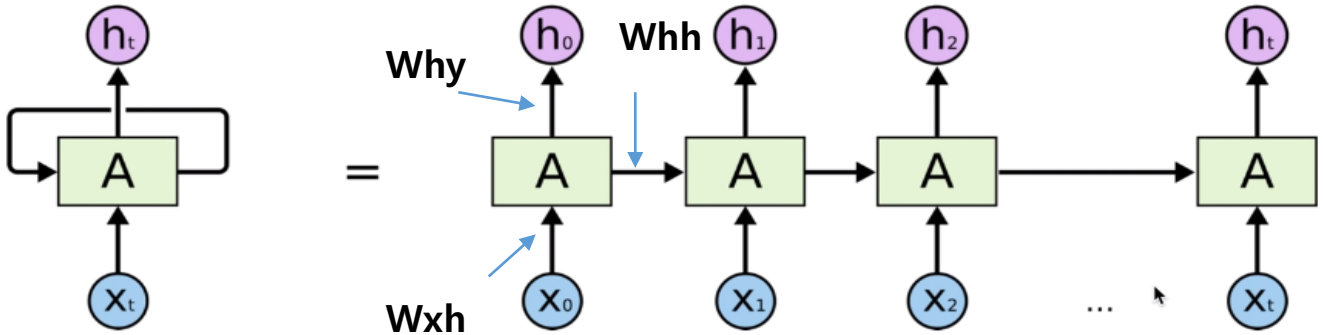
The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



- RNN Basic으로 Vanilla 사용
- 함수는 W * x
- sigmoid 대산 tanh
- ht = 다음 state 계산에 사용
- y = 계산된 ht에 다른 W 이용
- 3개 weight Whh, Wxh, Why 가 전체 cell에서 동일

Notice: the same function and the same set of parameters are used at every time step.

Source : https://youtu.be/-SHPG_KMUkQ

RNN

```
import tensorflow as tf
import numpy as np

tf.set_random_seed(123)  # reproducibility

tf.reset_default_graph() # correct an error : Variable rnn/basic_lstm_cell/kernel already exists

sample = " if you want you"
idx2char = list(set(sample))  # index -> char
char2idx = {c: i for i, c in enumerate(idx2char)}  # char -> idex

# hyper parameters
dic_size = len(char2idx)  # RNN input size (one hot size)
hidden_size = len(char2idx)  # RNN output size
num_classes = len(char2idx)  # final output size (RNN or softmax, etc.)
batch_size = 1  # one sample data, one batch
sequence_length = len(sample) - 1  # number of lstm rollings (unit #)

sample_idx = [char2idx[c] for c in sample]  # char to index
x_data = [sample_idx[:-1]]  # X data sample (0 ~ n-1) hello: hell
y_data = [sample_idx[1:]]   # Y label sample (1 ~ n) hello: ello

X = tf.placeholder(tf.int32, [None, sequence_length])  # X data
Y = tf.placeholder(tf.int32, [None, sequence_length])  # Y label
x_one_hot = tf.one_hot(X, num_classes)  # one hot: 1 -> 0 1 0 0 0 0 0 0 0 0
cell = tf.contrib.rnn.BasicLSTMCell(
    num_units=hidden_size, state_is_tuple=True)
initial_state = cell.zero_state(batch_size, tf.float32)
```

```
outputs, _states = tf.nn.dynamic_rnn(
    cell, x_one_hot, initial_state=initial_state, dtype=tf.float32)

# FC layer
X_for_fc = tf.reshape(outputs, [-1, hidden_size])
outputs = tf.contrib.layers.fully_connected(outputs, num_classes,
activation_fn=None)

# reshape out for sequence_loss
outputs = tf.reshape(outputs, [batch_size, sequence_length, num_classes])

weights = tf.ones([batch_size, sequence_length])
sequence_loss = tf.contrib.seq2seq.sequence_loss(
    logits=outputs, targets=Y, weights=weights)
loss = tf.reduce_mean(sequence_loss)
train = tf.train.AdamOptimizer(learning_rate=0.1).minimize(loss)

prediction = tf.argmax(outputs, axis=2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(50):
        l, _ = sess.run([loss, train], feed_dict={X: x_data, Y: y_data})
        result = sess.run(prediction, feed_dict={X: x_data})

        # print char using dic
        result_str = [idx2char[c] for c in np.squeeze(result)]
        print(i, "loss:", l, "Prediction:", ''.join(result_str))
```

Source :

Summary

	linear regression	logistic classification	CNN	RNN
hypothesis	= x_train * W + b	g(z) = 1 / (1+exp –z) = tf.sigmoid(tf.matmul(X, W) + b)		
cost	= tf.reduce_mean (tf.square(hypothesis – y_train))	-y log(hx) – (1-y)log(1-hx) = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))		
optimizer & training	optimizer = = tf.train.GradientDescent Optimizer(learning_rate=0.1) xy_train = optimizer.minimize (cost) for step in range(2001): sess.run(xy_train)			

```
# Ask my score
print("Your score will be ", sess.run(
    hypothesis, feed_dict={X: [[100, 70, 101]]}))
```

Source :

linear regression – logistic

	linear regression	logistic classification
hypothesis	= x_train * W + b	g(z) = 1 / (1+exp –z) = tf.sigmoid(tf.matmul(X, W) + b)
cost	= tf.reduce_mean (tf.square(hypothesis – y_train))	-y log(hx) – (1-y)log(1-hx) = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
optimizer & training	optimizer = tf.train.GradientDescent Optimizer(learning_rate=0.1) xy_train = optimizer.minimize (cost) for step in range(2001): sess.run(xy_train)	

Source :

Multi-variable Linear regression : ask my score

```
# Multi-variable linear regression
import tensorflow as tf
import numpy as np
tf.set_random_seed(123)  # for reproducibility

xy = np.loadtxt('data-01-test-score.csv', delimiter=',',
dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

# Make sure the shape and data are OK
print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data)
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b
# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
```

```
# Minimize : learning_rate=1e-5, 2000 Cost = 64.67786   /  lr = 0.001  : 2000
cost = nan  / .00001 : 3000 cost = 6.54
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.00001)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in range(3001):
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train], feed_dict={X: x_data, Y: y_data})
    if step % 50 == 0:
        print(step, "Cost: ", cost_val, "₩nPrediction:₩n", hy_val)

# Ask my score
print("Your score will be ", sess.run(
    hypothesis, feed_dict={X: [[100, 70, 101]]}))

print("Other scores will be ", sess.run(hypothesis,
                                        feed_dict={X: [[60, 70, 110], [90, 100, 80]]}))
```

Source :

logistic softmax classification (여러 개 Labels 중 예측)

```
x_data = [[1, 2, 1, 1], [2, 1, 3, 2], [3, 1, 3, 4], [4, 1, 5, 5], [1, 7, 5, 5],
                                                   [1, 2, 5, 6], [1, 6, 6, 6], [1, 7, 7, 7]]
y_data = [[0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 1, 0], [0, 1, 0], [0, 1, 0], [1, 0, 0], [1, 0, 0]]
```

hypothesis

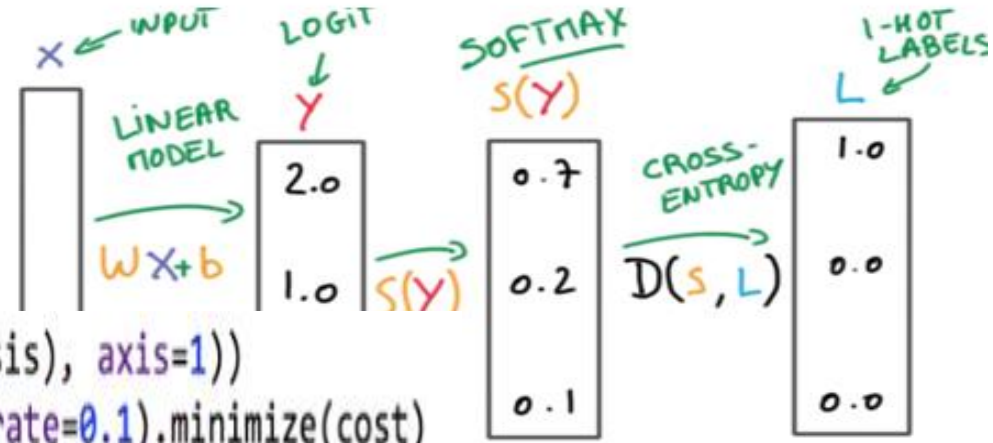
= tf. nn. softmax (tf.matmul(X, W) + b)

cost

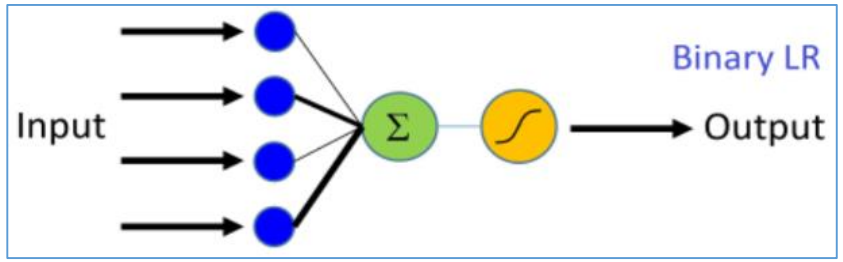
optimizer & training

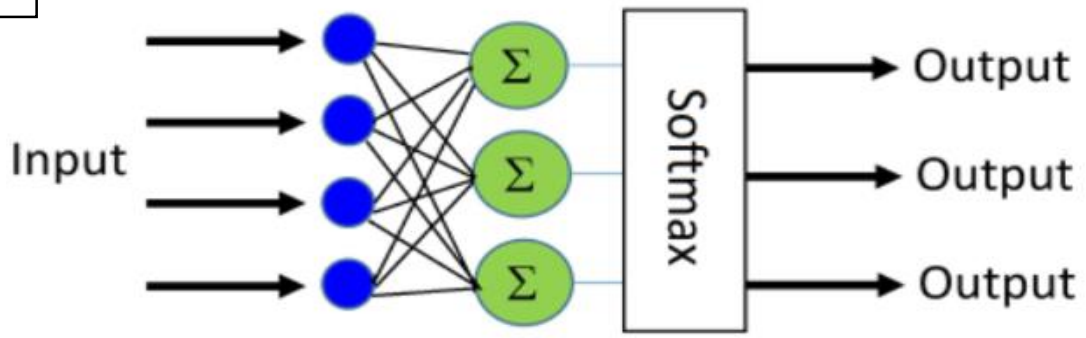
```
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

apply



Multiclass LR





Source :

CNN & RNN

	CNN	RNN
hypothesis	= x_train * W + b	g(z) = 1 / (1+exp –z) = tf.sigmoid(tf.matmul(X, W) + b)
cost	= tf.reduce_mean (tf.square(hypothesis – y_train))	-y log(hx) – (1-y)log(1-hx) = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
optimizer =		
optimizer & training	= tf.train.GradientDescent Optimizer(learning_rate=0.1) xy_train = optimizer.minimize (cost)	=
	for step in range(2001): sess.run(xy_train)	=

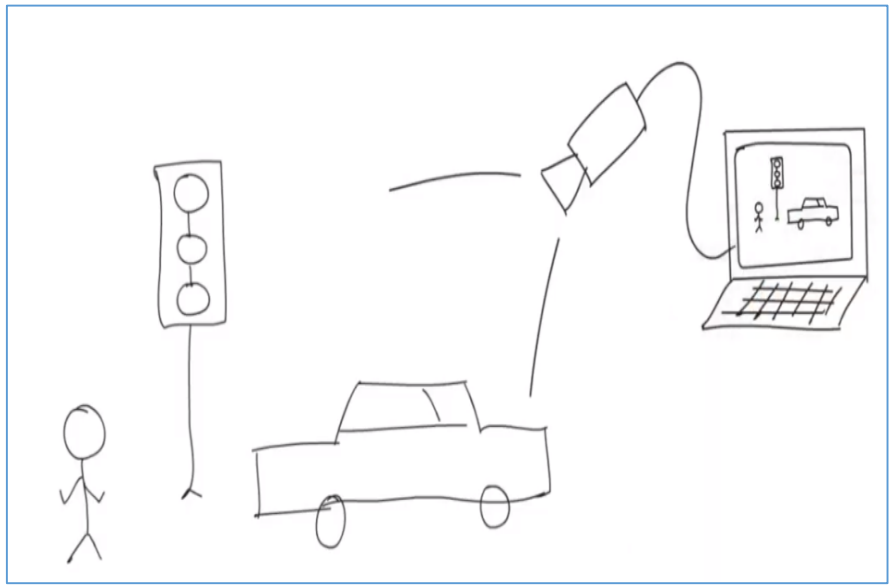
Source :

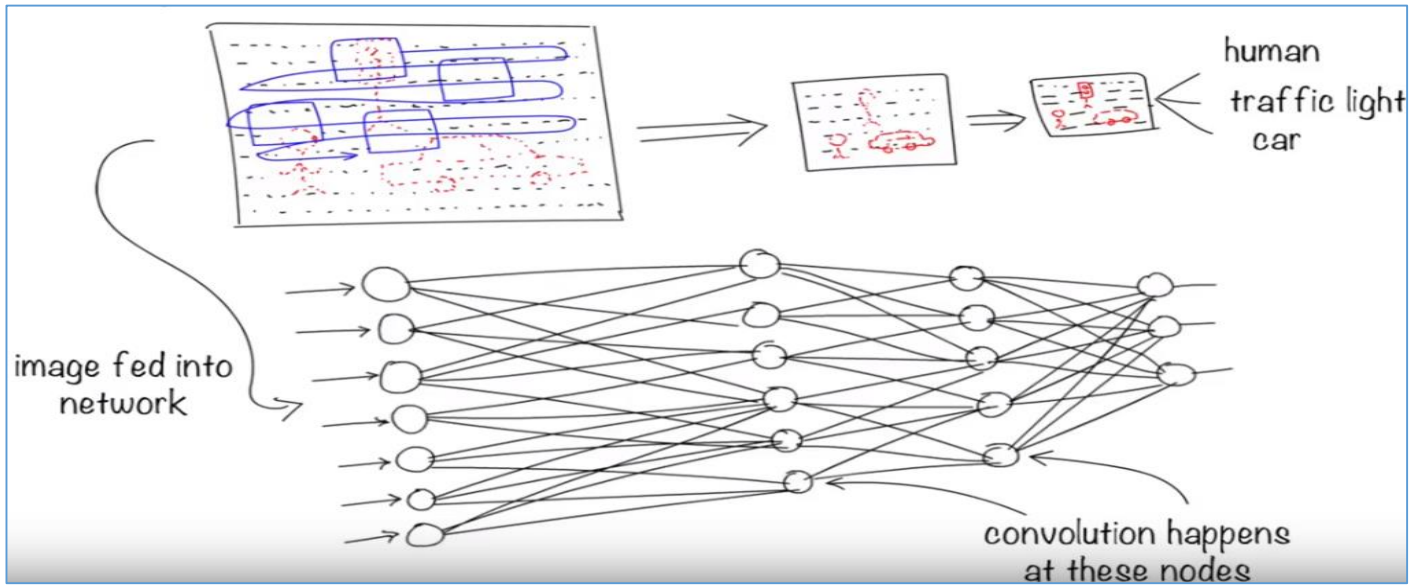
tensorBoard

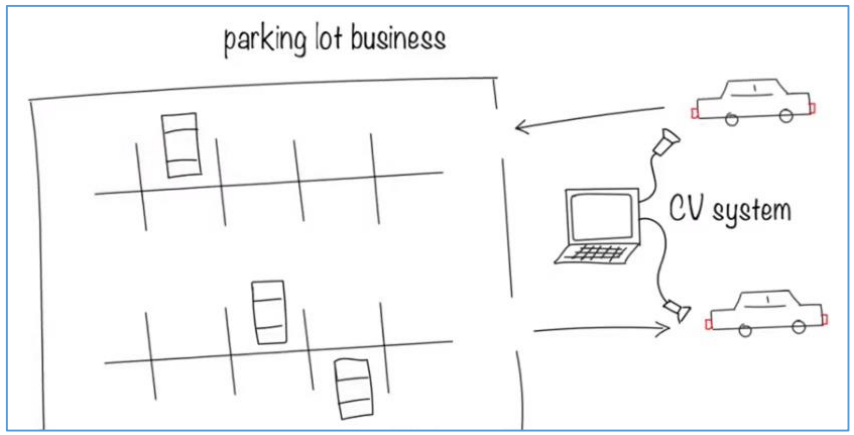
```
import tensorflow as tf

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

add = tf.add(X, Y)
mul = tf.multiply(X, Y)

# step 1: node 선택
add_hist = tf.summary.scalar('add_scalar', add)
# add_hist = tf.summary.histogram('weight', add)
mul_hist = tf.summary.scalar('mul_scalar', mul)

# step 2: summary 통합. 두 개의 코드 모두 동작.
merged = tf.summary.merge_all()
# merged = tf.summary.merge([add_hist, mul_hist])

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    # step 3: writer 생성
    writer = tf.summary.FileWriter('./board/sample_2', sess.graph)

    for step in range(100):
        # step 4: 노드 추가
        summary = sess.run(merged, feed_dict={X: step * 1.0, Y: 2.0})
        writer.add_summary(summary, step)

# step 5: 콘솔에서 명령 실행
# tensorboard --logdir=./board/sample_1
# 콘솔 폴더가 현재 폴더(.)가 아니라면, 절대 경로 지정
# tensorboard --logdir=~/PycharmProjects/test/board/sample_2
```

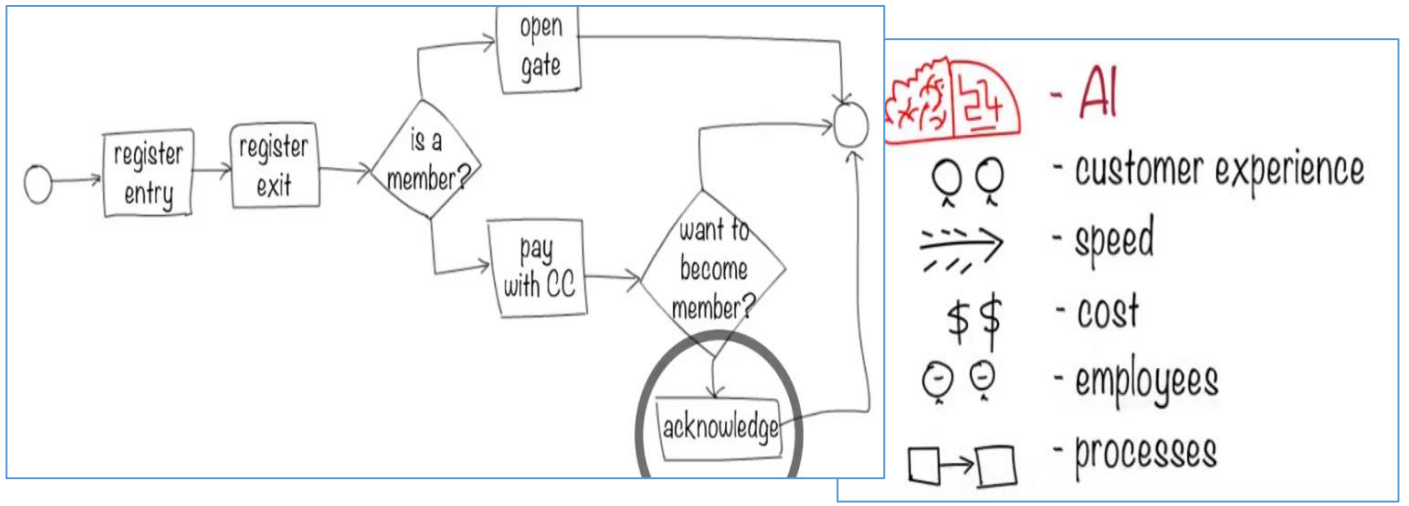
Source :

How to Apply AI in Business







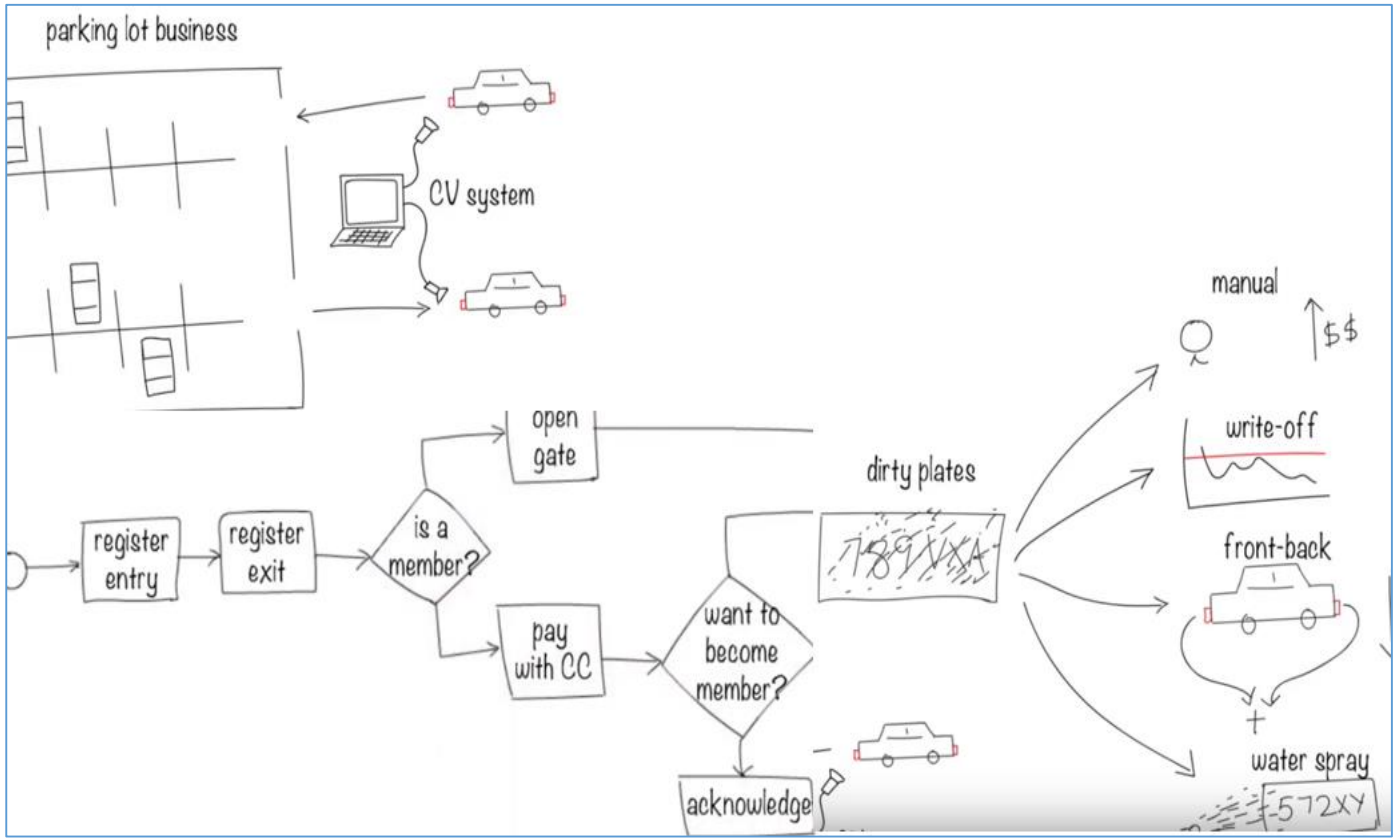


Source : https://youtu.be/N_eHmaRf9T4

문제 해결을 위한 시도

- 문제 정의
- 문제해결을 위한 각종 방법론 (7 Steps, TRIZ) +
- 다양한 디지털 기술
- Pilot으로 신속한 검증

사례 참조



Source : how to apply ai in business, https://youtu.be/N_eHmaRf9T4

ROC

- **정분류율(Accuracy)** $= \frac{a+d}{a+b+c+d}$
- **오분류율(Error rate)** $= \frac{b+c}{a+b+c+d}$
- **민감도(Sensitivity) 또는 재현율(recall)** $= \frac{a}{a+b}$
- **특이도(Specificity)** $= \frac{d}{c+d}$
- **정밀도(Precision)** $= \frac{a}{a+c}$

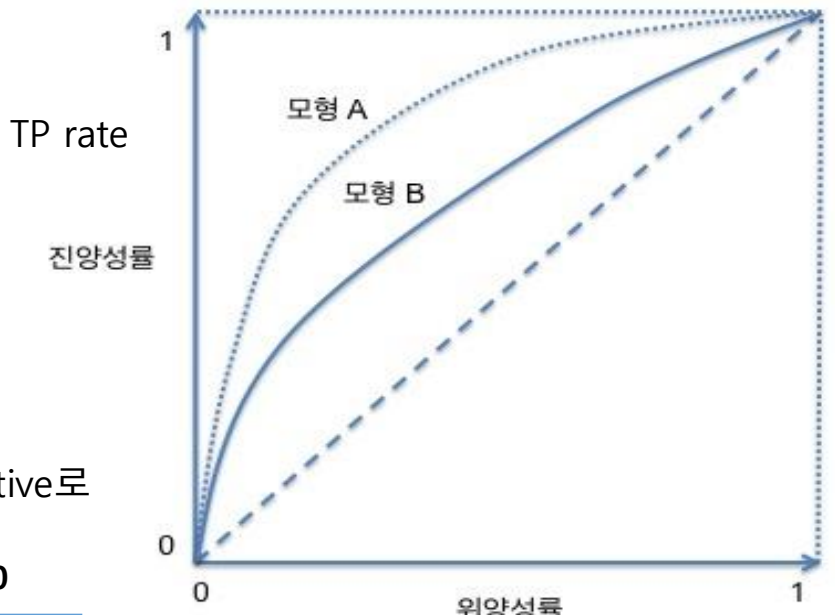
		예측값	
		Positive	Negative
실제값	Positive	a	b
	Negative	c	d

- **ROC 그래프**
  - **ROC(Reciever Operating Characteristic)**
  - **X축은 FP 비율(1-specificity)**
  - **Y축은 TP 비율(Sensitivity)**
- **AUC**
  - **Area Under the ROC Curve**
  - **1에 가까울수록 좋은 모형**

모두 Positive로 분류

100

100 + 0



모두 Negative로 분류

0

0 + 100

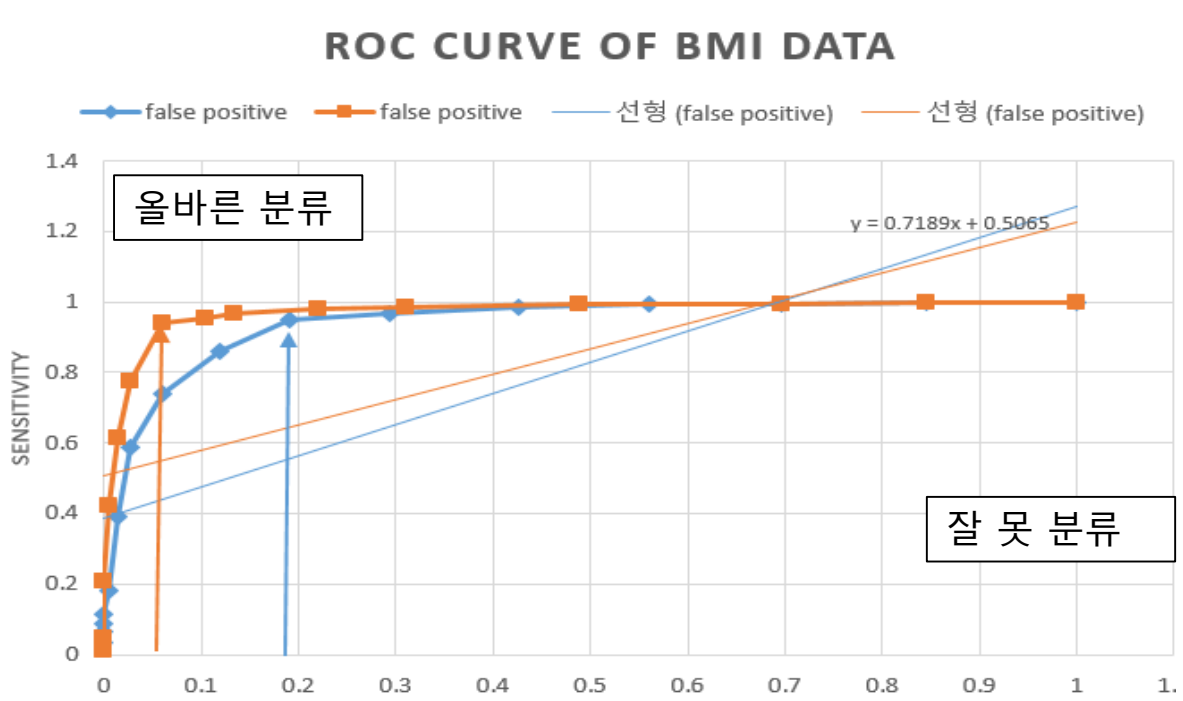
c

c + d

Source :

ROC

모델	분류	성능			sensitivity	false positive	cut off point
bmi	기준	disease	healthy	sum			
24	1	1	52	53	1	1	1.0000
25	2	1	50	51	0.997	0.845	1.1521
26	3	1	46	47	0.995	0.696	1.2982
27	4	3	45	48	0.992	0.560	1.4324
28	5	6	44	50	0.984	0.426	1.5582
29	6	7	35	42	0.968	0.295	1.6729
30	7	33	24	57	0.949	0.190	1.7582
31	8	45	20	65	0.859	0.119	1.7404
32	9	55	11	66	0.738	0.060	1.6783
33	10	74	4	78	0.589	0.027	1.5624
34	11	77	3	80	0.389	0.015	1.3743
35	12	25	2	27	0.181	0.006	1.1751
36	13	10	0	10	0.114	0.000	1.1135
37	14	8	0	8	0.086	0.000	1.0865
38	15	12	0	12	0.065	0.000	1.0649
39	16	12	0	12	0.032	0.000	1.0324
total		370	336				



		예측	
		p	n
		40	60
실제	30	30	10
	70	10	60
TP	0.75		
FP	0.14		

		예측	
		p	n
		50	50
실제	30	30	0
	70	20	50
TP	1		
FP	0.29		

Source : 통계 R 학습_roc.xlsx / https://youtu.be/EQASA00NQgM

Python, Tensorflow code

- ML_00_Basic_manipulation.ipynb
- ML_00_Basic_manipulation_2.ipynb
- ML_00_Cast.ipynb
- ML_00_data_manipulation.ipynb
- ML_00_data_manipulation_2.ipynb
- ML_00_imshow_Image.ipynb
- ML_01_learning_rate_mnist.ipynb
- ML_01_learning_rate_x_data.ipynb
- ML_01_Linear_Regression.ipynb
- ML_01_lr_softmax.ipynb
- ML_01_NN_MNIST.ipynb
- ML_01_one_hot.ipynb
- ML_01_Seq_char_rnn.ipynb
- ML_01_Summary.ipynb
- ML_01_tensorBoard.ipynb
- ML_01_tensorBoard_MNIST.ipynb
- ML_02_cnn_MNIST.ipynb
- ML_02_regression_housing.ipynb
- ML_03_UseCase_scm.ipynb

Linear regression

- Basic, linear regression, softmax
- MNIST : matplotlib.pyplot (2번 실행 : 7)
- ln(15) : mul-reg.csv

Logistic classification

RNN

- Character Sequence RNN (time ok)

CNN

- cnn : takes too much time

tensorBoard

- tensorBoard 설명 : 413. Python_tf_tb
- TB_SUMMARY_DIR = './tb/mnist' : start learning .... (10 minutes)

- ML_12_1_hello_rnn.ipynb
- ML_12_0_rnn_basic.ipynb
- ML_11_05_mnist_cnn_ensemble_layer.ipynb
- ML_11_04_mnist_cnn_ensemble.ipynb
- ML_11_03_mnist_class.ipynb
- ML_11_02_mnist_deep_cnn.ipynb
- ML_11_0_cnn_basic.ipynb

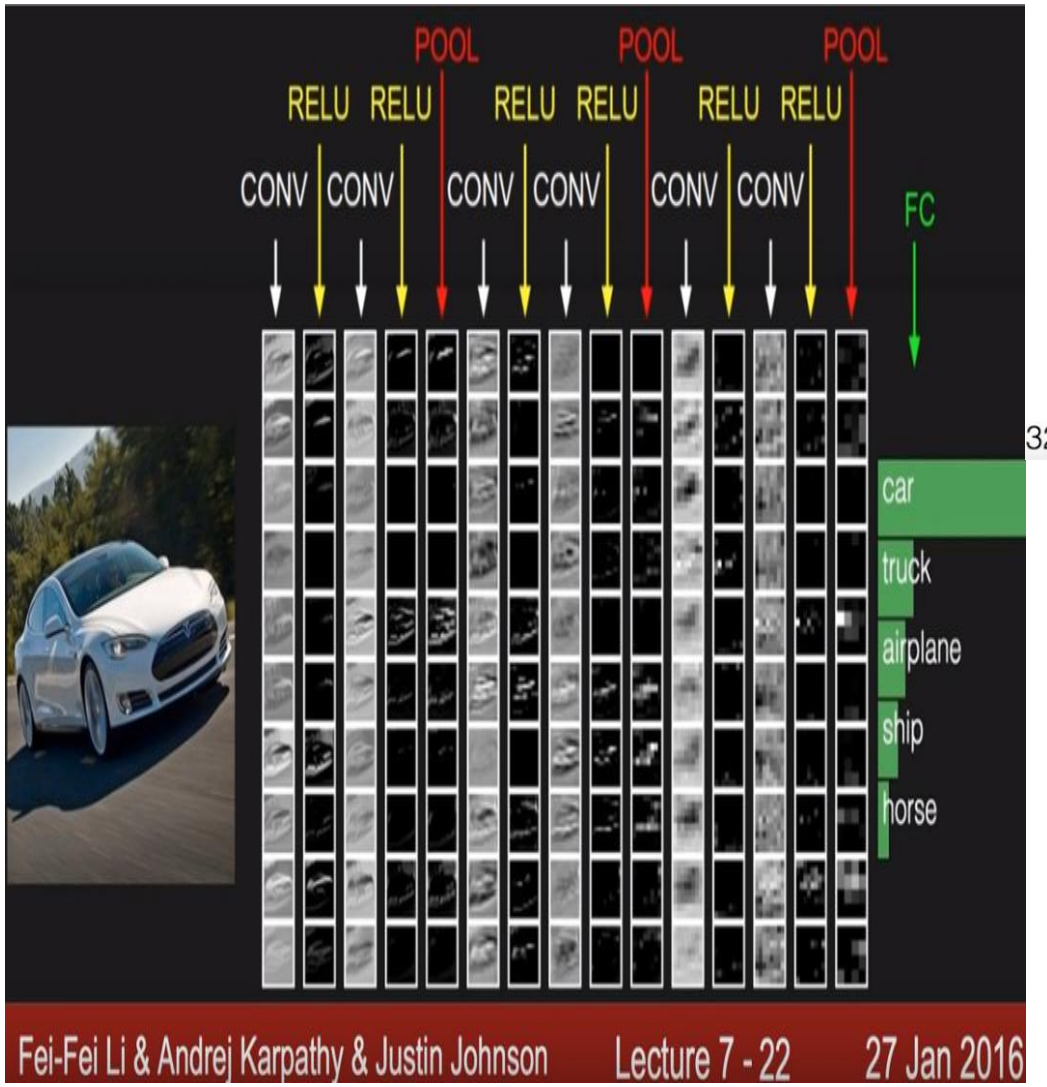
Source : ML_10 ~ 13

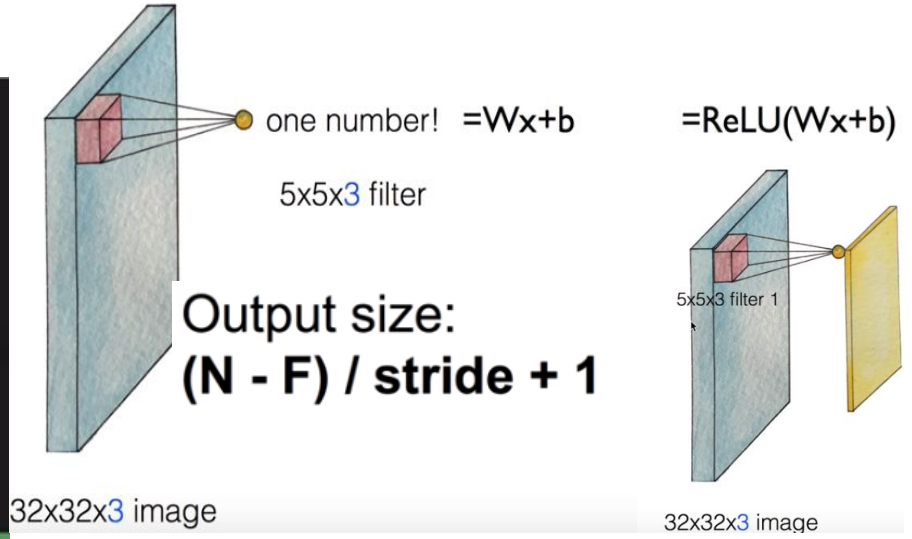
linear regression – logistic – softmax classfication

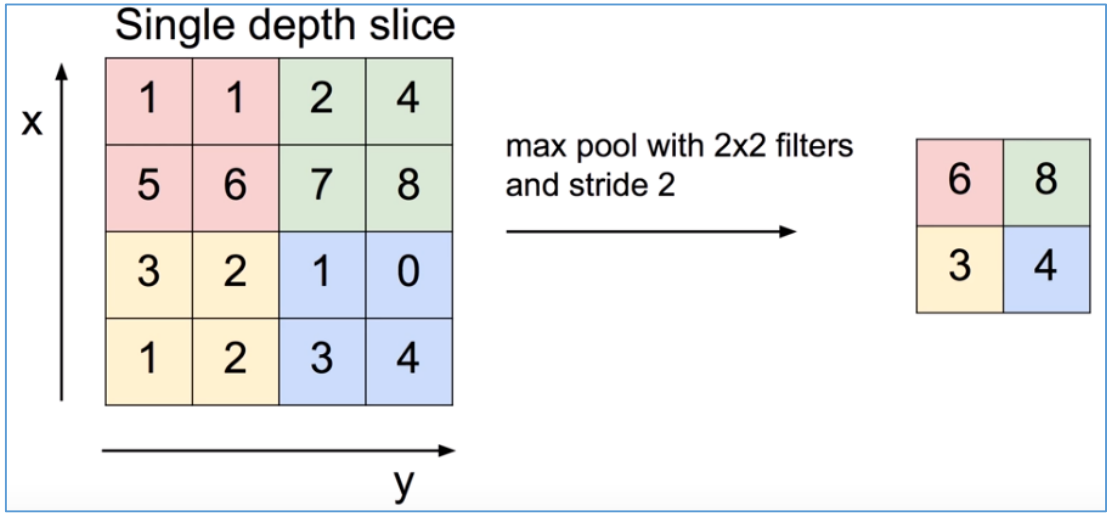
	linear regression	logistic classification	softmax
hypothesis	= x_train * W + b	g(z) = 1 / (1+exp –z) hypothesis = tf.sigmoid(tf.matmul(X, W) + b)	logits = tf.matmul(X, W) + b hypothesis = tf.nn.softmax(logits)
cost	= tf.reduce_mean (tf.square(hypothesis – y_train))	= -y log(hx) – (1-y)log(1-hx)	cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
			cost_i = tf.nn.softmax_cross_entropy _with_logits_v2(logits=logits, labels=tf.stop_gradient ([Y_one_hot])) cost = tf.reduce_mean(cost_i)
optimizer & training	optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1) train = optimizer.minimize(cost)	train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)	

Source : 320. ML_algorithm_code_summary

ConvNet의 Conv layer, pooling & Fully Connected layer + softmax

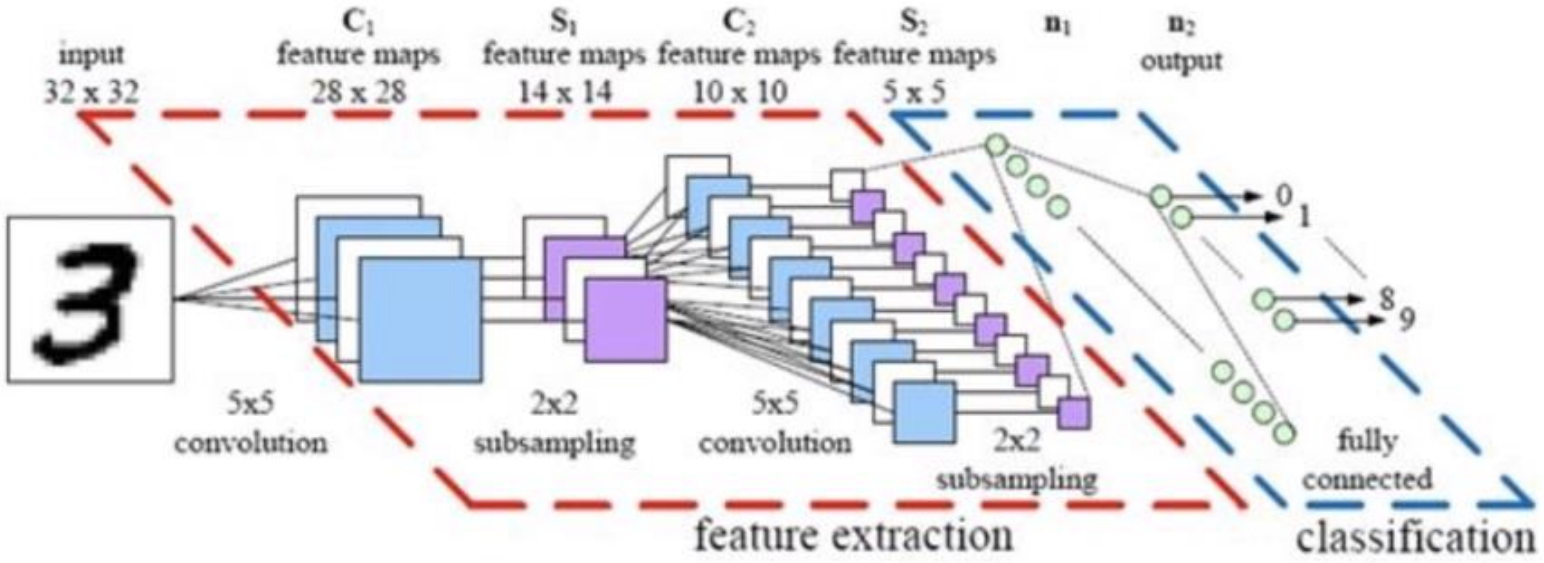






Source : Sung kim, https://youtu.be/Em63mknbtWo

TensorFlow CNN Basics

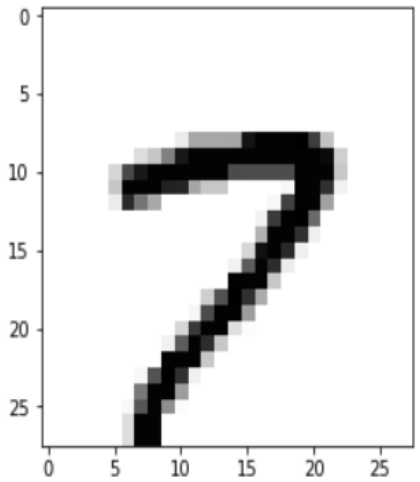


Source : https://youtu.be/E9Xh_fc9KnQ

TensorFlow CNN Basics

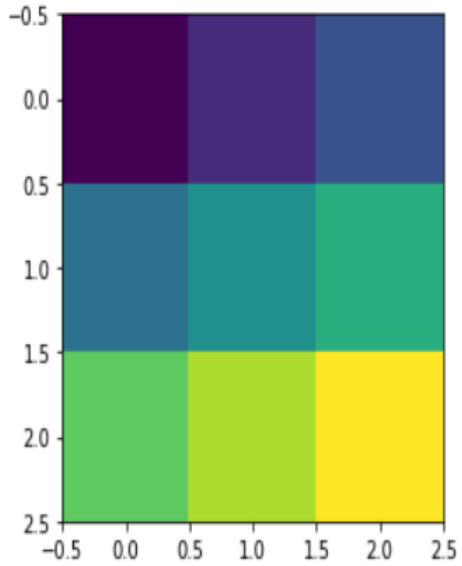
```
import matplotlib.pyplot as plt
import tensorflow as tf

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load

# %matplotlib inline # Only use this if using iPython

image_index = 123 # You may select anything up to 60,000

print(y_train[image_index]) # The label is 8
plt.imshow(x_train[image_index], cmap='Greys')
```

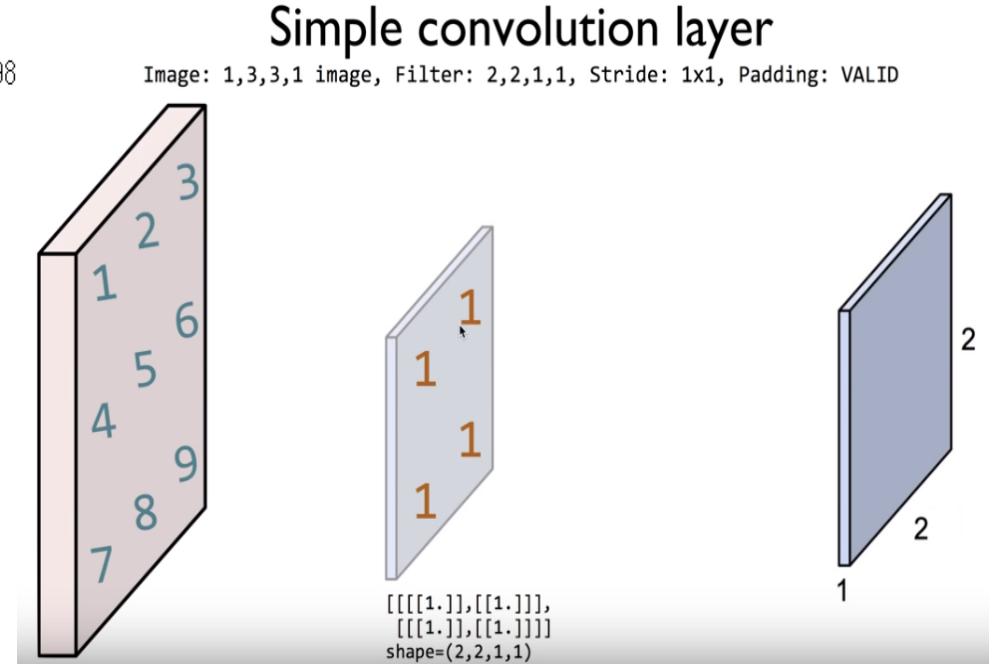
```
7

<matplotlib.image.AxesImage at 0x2a659a50d30>
```



```
import numpy as np
# sess = tf.InteractiveSession()

image = np.array([[[[1], [2], [3]], [[4], [5], [6]], [[7], [8], [9]]]], dtype = np.float32)

print(image.shape)
# plt.imshow(image.reshape(3,3), cmap='Greys') # error Greys

plt.imshow(image.reshape(3,3))
```

```
(1, 3, 3, 1)

<matplotlib.image.AxesImage at 0x2a656369198
```



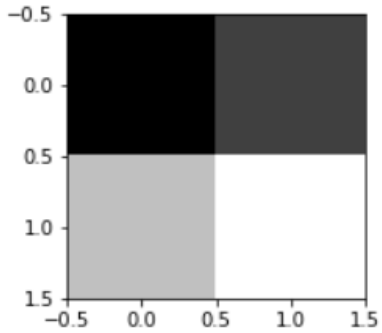


Source : ML_10_all.ipynb

TensorFlow CNN Basics

```
import numpy as np

# image = np.array([[[[1], [2], [3]], [[4], [5], [6]], [[7], [8], [9]]]], dtype = np.

print('image.shape', image.shape)

weight = tf.constant([[[[1.]], [[1.]]], [[[1.]], [[1.]]]])

print('weight.shape', weight.shape)
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1], padding='VALID')
conv2d_img = conv2d.eval()
print('conv2d_img', conv2d_img.shape)

con2d_img = np.swapaxes(conv2d_img, 0, 3)
for i, one_img in enumerate(conv2d_img) :
                        print(one_img.reshape(2, 2))
                        plt.subplot(1, 2, i+1), plt.imshow(one_img.reshape(2, 2), cmap
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d_img (1, 2, 2, 1)
[[12. 16.]
 [24. 28.]]
```

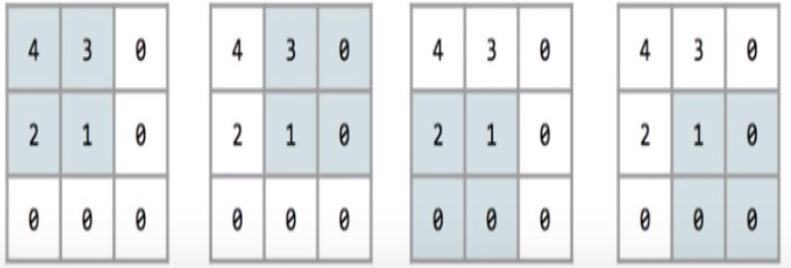


Max Pooling

```
In [19]: image = np.array([[[[4],[3]],
                           [[2],[1]]]], dtype=np.float32)
pool = tf.nn.max_pool(image, ksize=[1, 2, 2, 1],
                      strides=[1, 1, 1, 1], padding='SAME')
print(pool.shape)
print(pool.eval())
```

```
(1, 2, 2, 1)
[[[[ 4.]
   [ 3.]]

  [[ 2.]
   [ 1.]]]]
```

SAME: Zero paddings

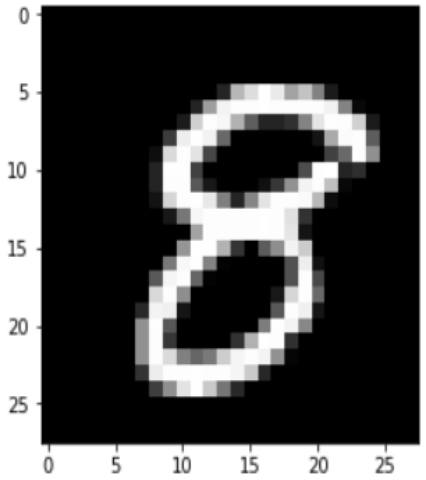


Source : ML_10_all.ipynb / https://youtu.be/E9Xh_fc9KnQ

TensorFlow CNN Basics

```
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('MNIST_data/', one_hot=True)


img = mnist.train.images[55].reshape(28,28)
plt.imshow(img, cmap='gray')
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz

<matplotlib.image.AxesImage at 0x2a6000a8160>
```

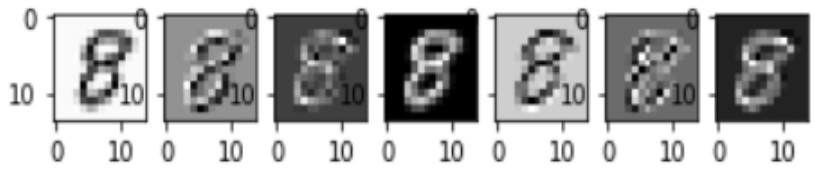


```
img = img.reshape( -1, 28, 28, 1)

W1 = tf.Variable(tf.random_normal([ 3, 3, 1, 7], stddev=0.01))

conv2d = tf.nn.conv2d(img, W1, strides=[1, 2, 2, 1], padding='SAME')

print(conv2d )

sess.run(tf.global_variables_initializer())

conv2d_img = conv2d.eval()
conv2d_img = np.swapaxes(conv2d_img, 0, 3)

for i, one_img in enumerate(conv2d_img) :
                    plt.subplot(1, 7, i+1), plt.imshow(one_img.reshape(14, 14), cmap='gray')
```

1 color

7 filter

```
Tensor("Conv2D_10:0", shape=(1, 14, 14, 7), dtype=float32)
```



Source : ML_10_all.ipynb

MNIST 98% with NN

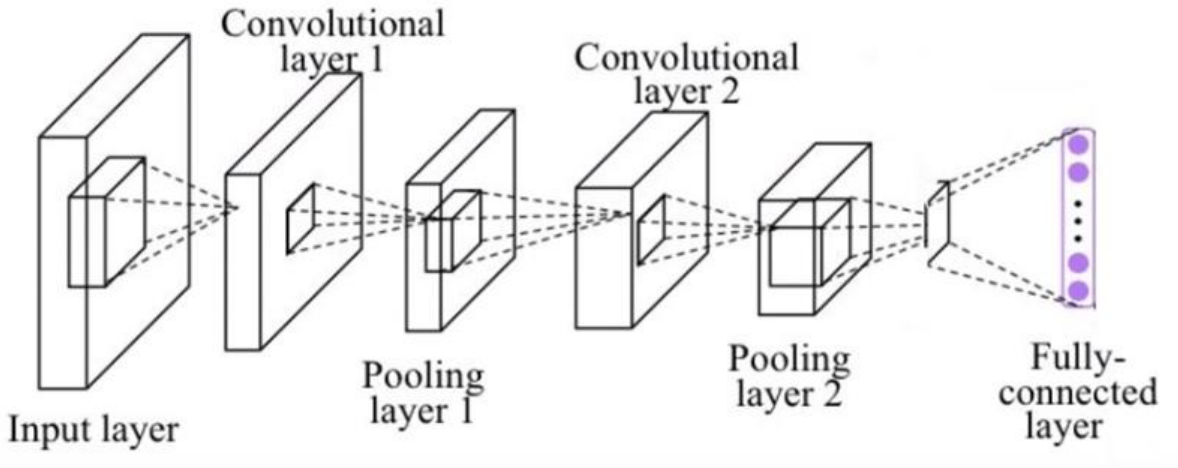
```
W1 = tf.get_variable("W1", shape=[784, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.Variable(tf.random_normal([512]))
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

W2 = tf.get_variable("W2", shape=[512, 512],
                     initializer=tf.contrib.layers.xavier_initializer())
b2 = tf.Variable(tf.random_normal([512]))
L2 = tf.nn.relu(tf.matmul(L1, W2) + b2)
```

```
# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
```

```
Epoch: 0001 cost = 0.291563575
Epoch: 0002 cost = 0.101755542
Epoch: 0003 cost = 0.071341329
Epoch: 0004 cost = 0.052176375
Epoch: 0005 cost = 0.041552496
Epoch: 0006 cost = 0.033437219
Epoch: 0007 cost = 0.027995250
Epoch: 0008 cost = 0.027004075
Epoch: 0009 cost = 0.023694695
Epoch: 0010 cost = 0.019384865
Epoch: 0011 cost = 0.018703812
Epoch: 0012 cost = 0.017348291
Epoch: 0013 cost = 0.016754357
Epoch: 0014 cost = 0.013416878
Epoch: 0015 cost = 0.014078607
Learning Finished!
Accuracy: 0.98
Label:  [1]
Prediction:  [1]
```

Source : ML_10_all.ipynb

MNIST 99% with CNN



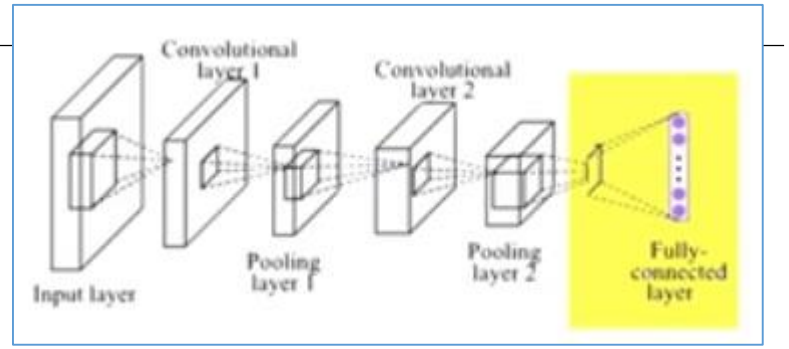

```
# input place holders
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1])   # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 10])

# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
```

```
# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv      ->(?, 14, 14, 64)
#    Pool      ->(?, 7, 7, 64)
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L2_flat = tf.reshape(L2, [-1, 7 * 7 * 64])
```

xavier를 통해 가중치 초기시 오류가 발생하면
import tensorflow as tf
tf.reset_default_graph() <<<<<<< 이부분을 추가해주면 오류가 사라집니다
import random

Source : ML_11_01_mnist_cnn.ipynb

MNIST 99% with CNN



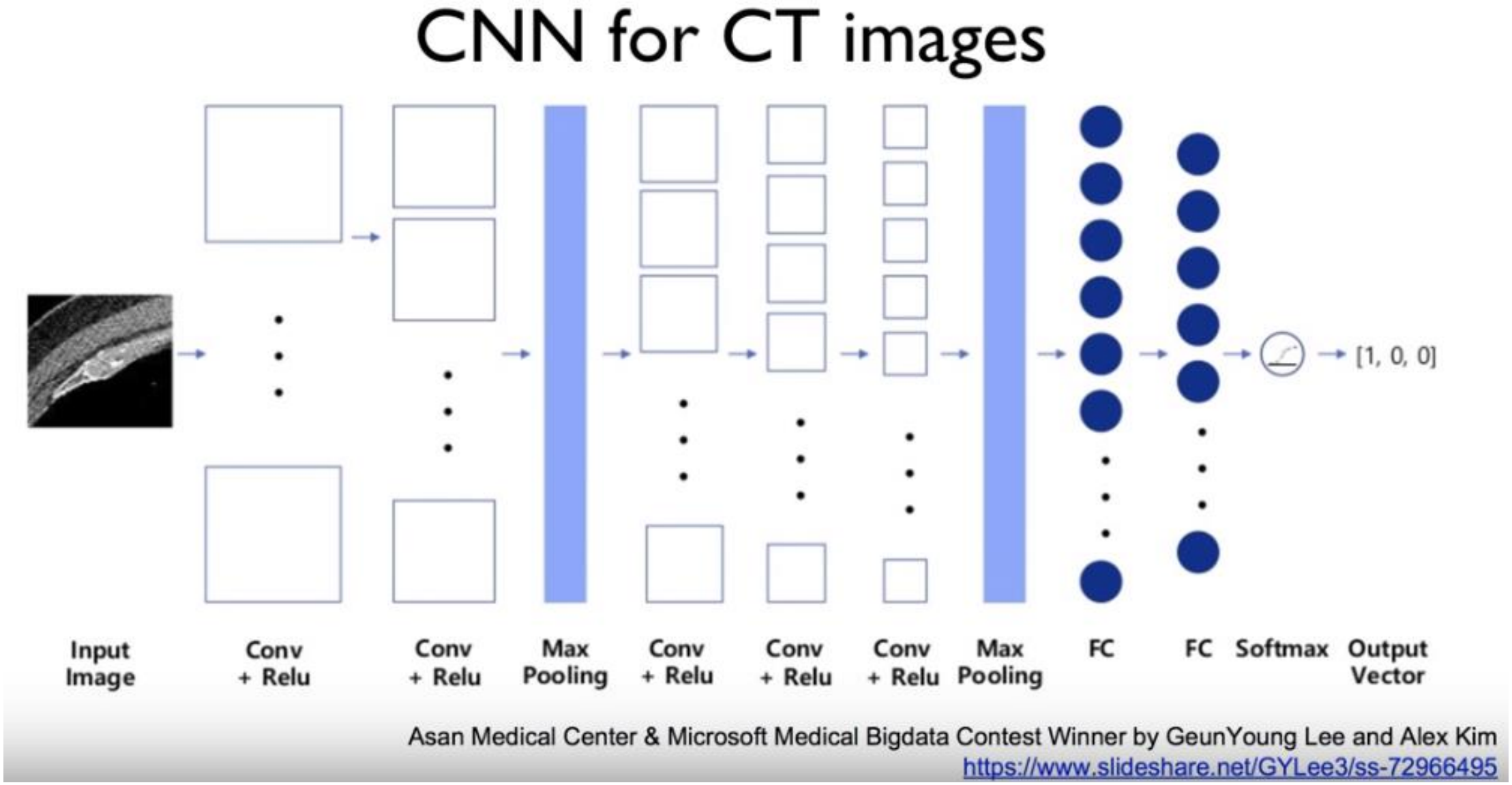

```
# Final FC 7x7x64 inputs -> 10 outputs
W3 = tf.get_variable("W2", shape=[7 * 7 * 64, 10],initializer=tf.contrib.layers
b = tf.Variable(tf.random_normal([10]))
logits = tf.matmul(L2_flat, W3) + b

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# initialize
sess = tf.Session()

sess.run(tf.global_variables_initializer())

# train my model
print('Learning started. It takes sometime.')
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)
```

```
Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See @{tf.nn.softmax_cross_entropy_with_logits_v2}.

Learning started. It takes sometime.
Epoch: 0001 cost = 0.380359673
Epoch: 0002 cost = 0.107616738
Epoch: 0003 cost = 0.078412918
Learning Finished!
Accuracy: 0.9792
Label:  [0]
Prediction:  [0]
```

시간 소요 ~ 1 epoch 21분

Source : ML_11_01_mnist_cnn.ipynb

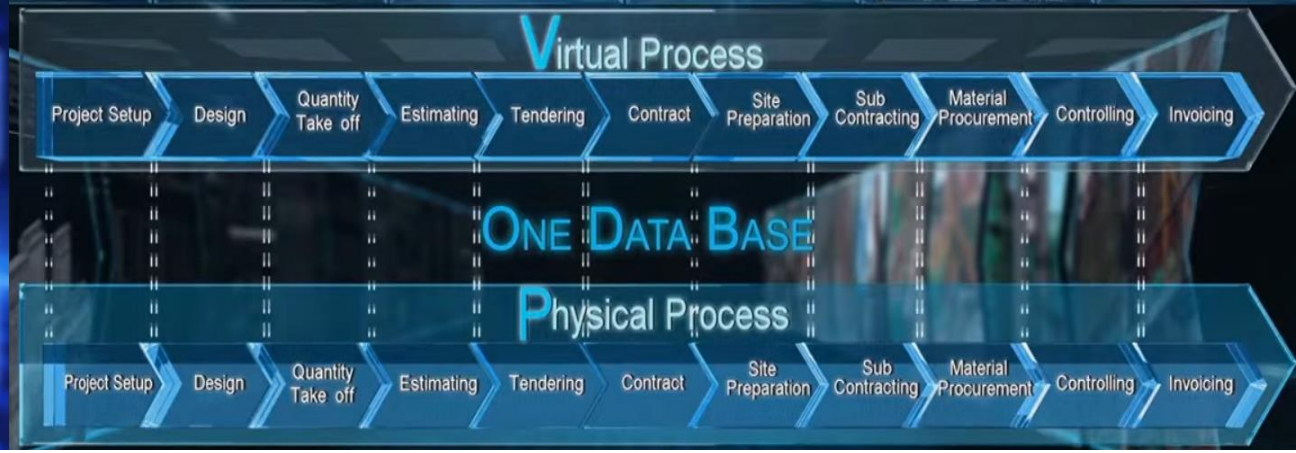
CNN 사례



Source : https://youtu.be/E9Xh_fc9KnQ

- 사전 지식 및 BDA Contents list
- R/Shiny
- AI, ML
- Web/Cloud

Source :



(430) web & CLOUD

- web, CLOUD 프로그램
- API economy

HTML(Hyper Text Markup Language) : 문서와 문서간의 이동이 가능한(Hyper Text = link) 특정 의미와 문법적인 형식/구조의 (Markup) 웹 페이지를 구성하는 언어(Language)

```
<html>
 <!-- head는 웹 페이지에 관한 정보를 갖음 -->
 <head>
 </head>

    <!-- body는 실제 눈으로 볼 수 있는 내용 -<html>

    <body>


</body>
</html>
```

중요 Tag :

- **사용자로 부터 입력을 받는 : input , 속성 type : text, password , submit , 속성 name :**
- **form : action="http~login.php" >**

Source : https://opentutorials.org/course/1688/9340

웹사이트 완성

HTML(Hyper Text Markup Language) : 문서와 문서간의 이동이 가능한(Hyper Text = link) 특정 의미와 문법적인 형식/구조의 (Markup) 웹 페이지를 구성하는 언어(Language)

```
<html>
 <!-- head는 웹 페이지에 관한 정보를 갖음 -->
 <head>
 </head>

    <!-- body는 실제 눈으로 볼 수 있는 내용 -<html>

    <body>


</body>
</html>
```

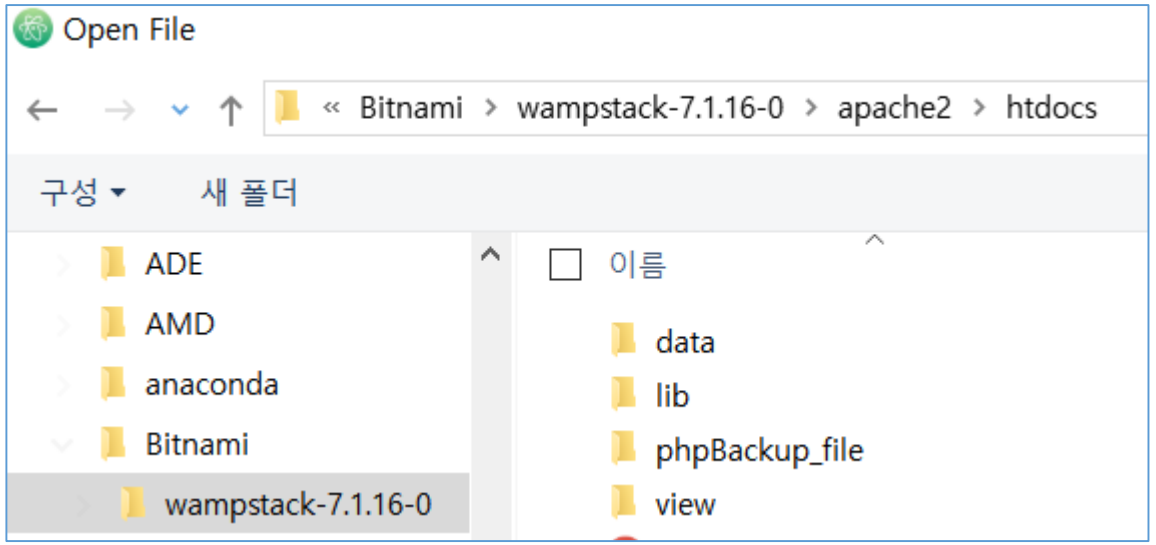
중요 Tag :

- **사용자로 부터 입력을 받는 : input , 속성 type : text, password , submit , 속성 name :**
- **form : action="http~login.php" >**

1.html 파일을 복제해서
index.html, 2.html, 3.html을 만듭니다.
그리고 각각의 파일에 맞게 내용을 수정합니다.
전체소스코드 다운로드

Source : https://opentutorials.org/course/3084/18431

bitnami



Source :

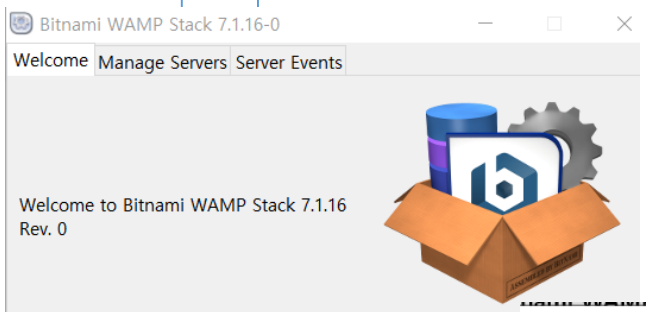
bitnami

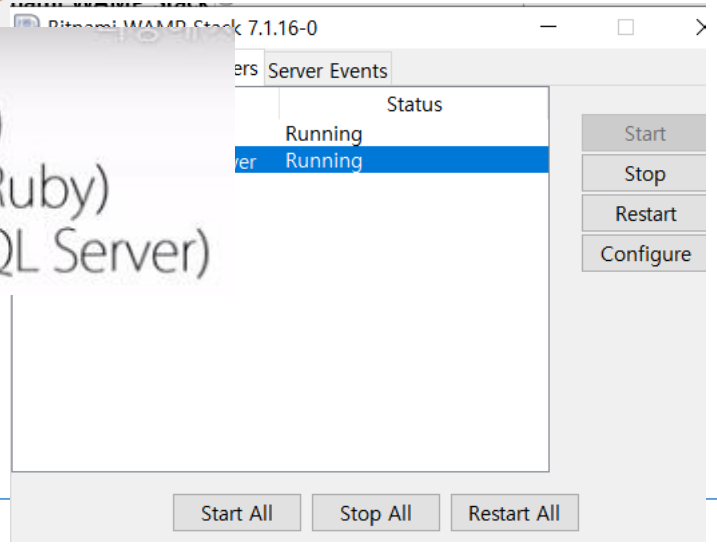
managers windows : Bitnami WAMP Stack Manager Tool

여러 소프트웨어를 제어

MySQL

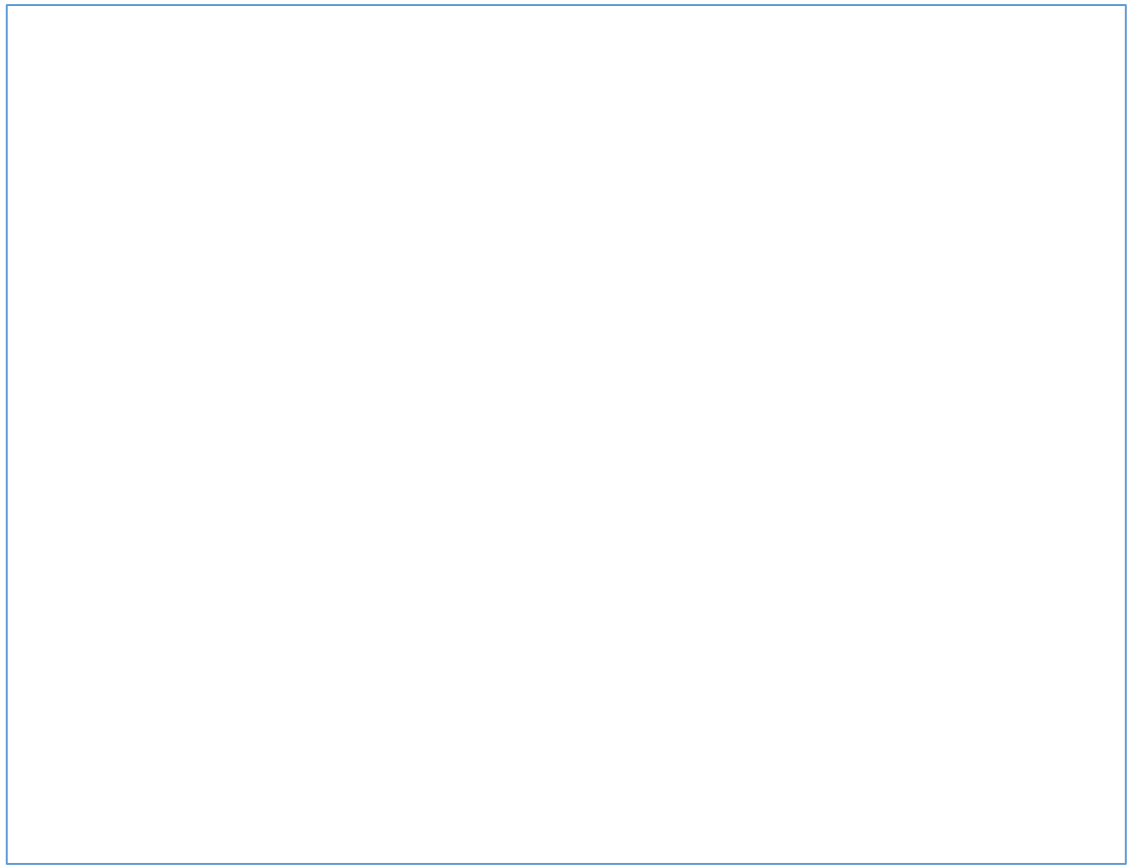
Apache Web Server 실행 여부 점검

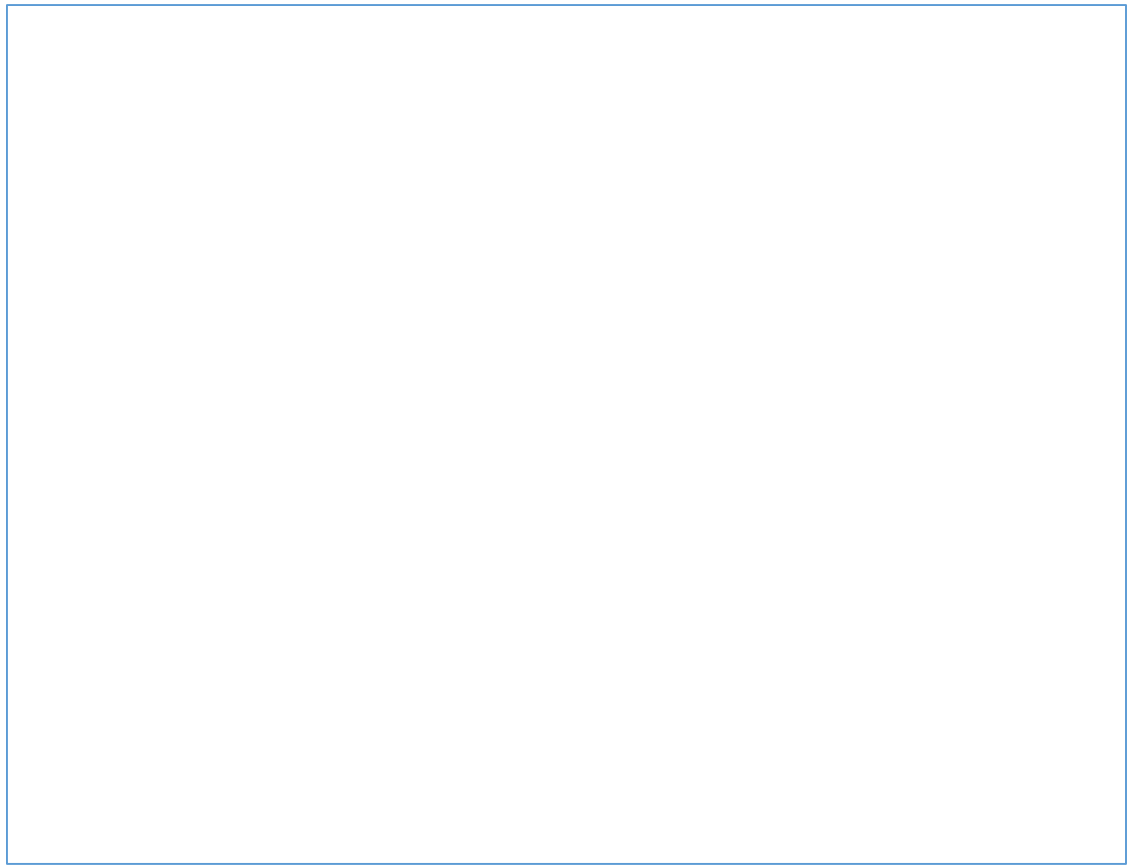




Source : https://opentutorials.org/course/1688/9340

pencil / balsamiq /





Source : https://opentutorials.org/course/1688/9340