

Python Tensorflow를 활용한 머신러닝

1. Machine Learning

1.1 통계학, 머신러닝, 인공지능

(1) 통계학, 인공지능, 머신러닝 - 정의

Statistics (S. M. Ross)

: Statistics is the art of learning from data.

Machine learning (A. Samuel)

: Machine Learning is a field of study that gives computers the ability to learn without being explicitly programmed.

: 명시적 프로그램 없이 데이터로부터 학습할 수 있는 능력을 컴퓨터에게 제공하는 기법

Artificial Intelligence (S. Russell)

: Artificial Intelligence is making computers intelligent.

(2) 확률분포 (probability distribution)

- 이항 확률분포 (Binomial distribution)

X: n번의 베르누이 시행 중 성공 횟수

$$X \sim B(n, p)$$

$$P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}, \quad x = 0, 1, 2, \dots, n$$

(Example: Binomial)

동전을 3번 던질 때 앞면이 2번 나올 확률

$$X \sim B(3, 0.5)$$

$$P(X = 2) = \binom{3}{2} 0.5^2 (1 - 0.5)^{3-2}$$

```
dbinom(2, 3, 0.5) # x, n, p  
[1] 0.375
```

- 포아송 확률분포 (Poisson distribution)

X: 사건의 빈도수

$$X \sim \text{Poisson}(m), \quad m > 0$$
$$P(X = x) = e^{-m} \frac{m^x}{x!}, \quad x = 0, 1, 2, \dots$$

(Example: Binomial)

지난 주까지 프로그래밍 강의에 지각한 학생수는 다음과 같다.

3, 5, 4, 3, 2

이 결과를 이용하여 오늘 프로그래밍 강의 시간에 3명이 지각할 확률을 구하시오.

```
m=mean(c(3, 5, 4, 3, 2))  
dpois(3, m)  
[1] 0.2186172
```

(note) 빅데이터 전처리 후 얻게 되는 문서 데이터 (text – Web, SNS, Patents, Papers, ...) 는 많은 경우가 count 데이터이다.

1.2 학습

(1) 지도 학습 (Supervised learning, with teacher)

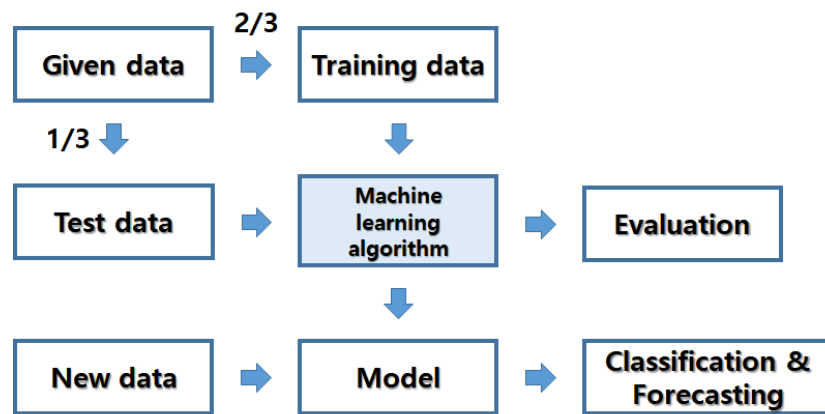
: Input variable (Explanatory variable)와 Output variable (Response variable)를 모두 알고 있음

분류, classification

: (나이, 연봉 / **신용상태**) → 신용상태 예측

회귀, regression

: (광고비, 온도 / **아이스크림판매량**) → 매출 예측



(2) 자율 학습 (Unsupervised learning, without teacher)

: Input variable (Explanatory variable)만 알고 있음

군집화, clustering

: (나이, 연봉) → 신용상태 예측

(3) Classification 기법

- 판별분석 (Discriminant analysis)
- 의사결정나무 (Decision tree)
- 랜덤 포레스트 (Random forest)
- Support vector machine (SVM)

(4) Regression 기법

- 선형 회귀분석 (Linear regression)
- 로지스틱 회귀분석 (Logistic regression)
- Lasso (least absolute shrinkage and selection operator) regression

(5) Clustering 기법

- K-means clustering
- Silhouette width (최적 군집수 결정)

- Fuzzy clustering
- Hierarchical methods
- K-medoids clustering

(6) UCI machine learning repository

<https://archive.ics.uci.edu/ml/index.php>

1.3 모형 평가

(1) Accuracy

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

A\P	C	$\neg C$	
C	TP	FN	P
$\neg C$	FP	TN	N
	P'	N'	All

- **Classifier Accuracy**
 - Accuracy = (TP + TN)/All
- **Misclassification rate:** 1 – accuracy
 - Misclassification rate = (FP + FN)/All

(출처: J. Han, et al., 2012)

→ accuracy가 클수록 우수한 모형

(2) MES (mean squared error)

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Y_i : 실제값

\hat{Y}_i : 예측값

n: test data의 크기

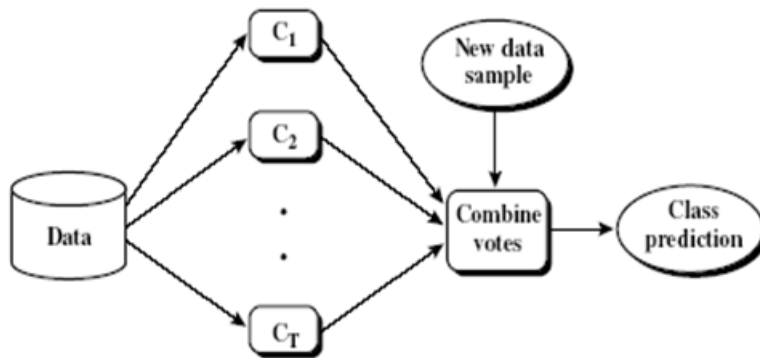
→ MSE의 크기가 작을수록 우수한 모형

(3) AIC (Akaike's Information Criterion)

: 좋은 예측을 하는 모델을 찾으려는 지표, AIC가 작을수록 좋은 모델

$$AIC = -2 \left(\text{최대로그우도} - \text{모수의수} \right)$$

1.4 Ensemble Learning



(출처: J. Han, et al., 2012)

(1) Ensemble Learning

- 다양한 기본 모형의 가중치 조합 학습

$$f(y|x, \theta) = \sum_{m \in M} w_m f_m(y|x)$$

- 정확도를 높이기 위하여 여러 모형들의 조합을 사용
- 한 모형의 성능향상을 위하여 여러 번의 학습 결과를 합침 (Bootstrap method)
- Committee learning → Voting

(2) 앙상블 방법들

- Bagging: 분류기 모음에 대한 예측의 평균 (averaging the prediction over a collection of classifiers)
- Boosting: 분류기 모음에 대한 가중 투표 (weighted vote with a collection of classifiers)
- Ensemble: 이질적인 분류자 집합의 결합 (combining a set of heterogeneous classifiers)

1.5 Data Scaling

(1) 정규화 (Normalization)

[0,1]

$$\frac{x - \min(x)}{\max(x) - \min(x)}$$

(2) 표준화 (standardization)

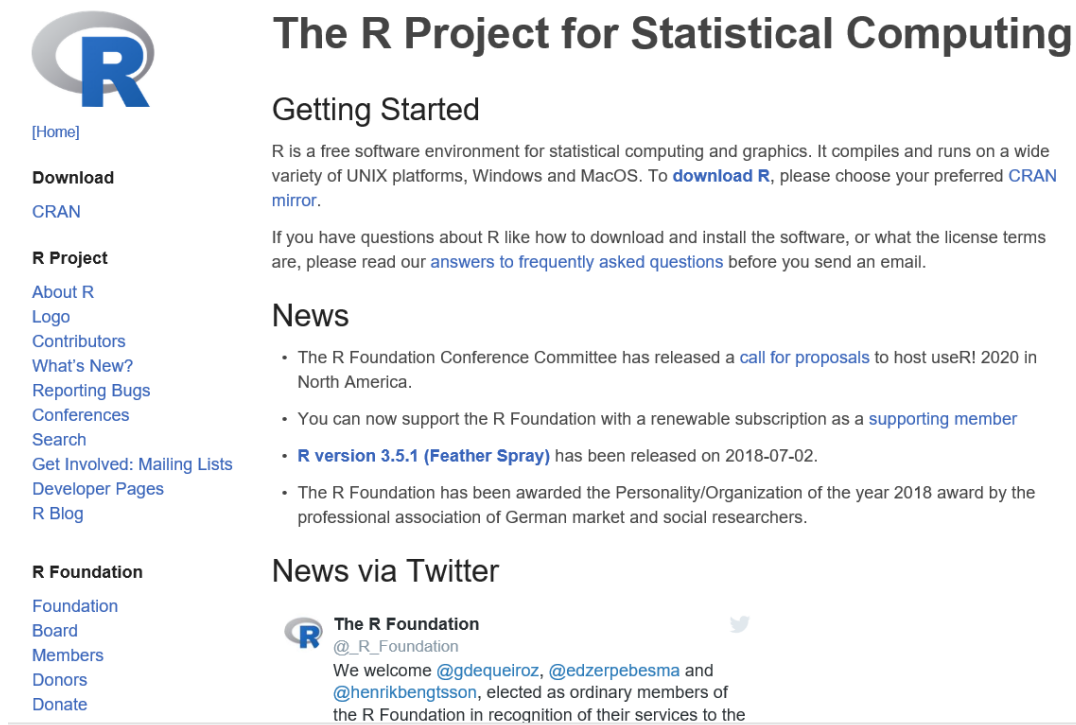
$(-\infty, \infty)$ $[-3, 3]$

$$\frac{x - \text{mean}(x)}{\text{sd}(x)}$$

1.6 R and Python for Machine Learning

(1) R data language

<https://www.r-project.org/>



The R Project for Statistical Computing

Getting Started


R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred [CRAN mirror](#).

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News

- The R Foundation Conference Committee has released a [call for proposals](#) to host useR! 2020 in North America.
- You can now support the R Foundation with a renewable subscription as a [supporting member](#)
- R version 3.5.1 (Feather Spray)** has been released on 2018-07-02.
- The R Foundation has been awarded the Personality/Organization of the year 2018 award by the professional association of German market and social researchers.

News via Twitter

 **The R Foundation**
@_R_Foundation
We welcome [@gdequeiroz](#), [@edzerpebesma](#) and [@henrikbengtsson](#), elected as ordinary members of the R Foundation in recognition of their services to the

- 오컬랜드 대학의 로버트 젠틀맨(Robert Gentleman)과 로스 이하카(Ross Ihaka)에 의해 개발한 객체지향 프로그래밍 언어
- 무료로 사용할 수 있는 오픈 소스
- 다양한 패키지를 통하여 최신 분석 기법을 제공
- 간편한 시각화 기능

- 방대한 데이터 분석 함수를 보유

(2) R과 Python의 차이

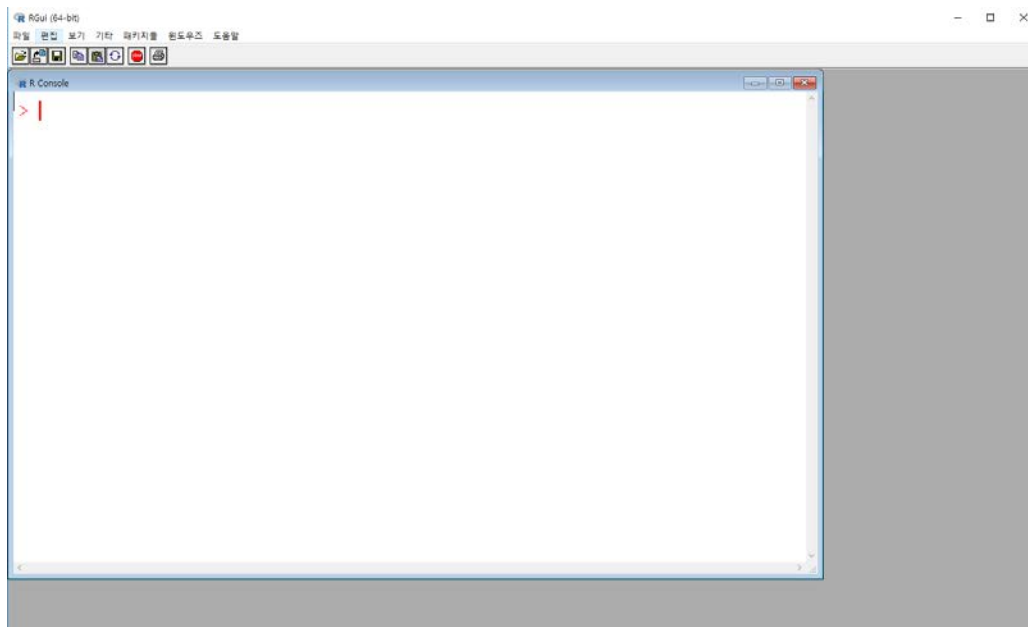
- R은 데이터분석에 강점, 파이썬은 소프트웨어(서비스) 개발에 강점
- 물론 R로도 소프트웨어 개발(웹서비스 등)이 가능하지만 python에 비해 효율이 떨어짐
- Python은 C/C++, Java와 같은 다른 프로그래밍 언어에 비해 데이터분석 기능이 잘 갖추어져 있음 (Tensorflow, Numpy, 등)

(3) 함수, 메서드 사용

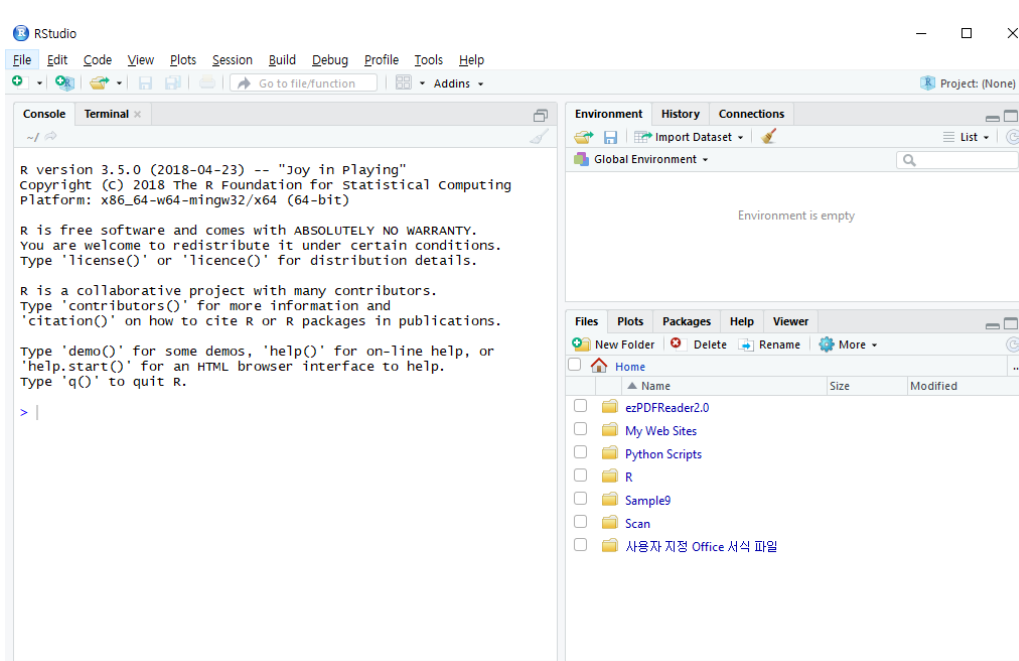
- R: 함수(객체)
- Python: 객체.함수

(4) R 기본과 RStudio

- R 기본



- RStudio



2. Python Tensorflow 설치 및 기본 사용법

2.1 Python 소개 및 설치

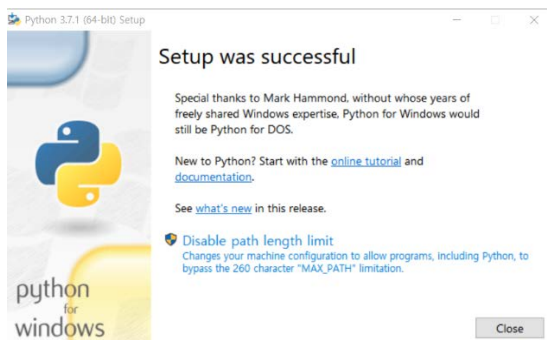
- (1) 1990년 귀도 반 로섬(Guido Van Rossum)이 만든 객체지향 프로그래밍 언어
- (2) 다양한 플랫폼(Window, Linux, Mac)에서 사용될 수 있는 인터프리터(Interpreter) 방식의 RAD(rapid application development) 언어
- (note) 인터프리터 언어: 한 줄씩 소스 코드를 해석해서 그때그때 실행해 결과를 바로 확인할 수 있는 언어
- (3) 방대한 라이브러리를 갖추고 있는 오픈소스 언어
- (4) 더 빠른 속도를 원하거나 일부 프로그램을 비공개로 해야할 경우 python 코드의 일부를 C/C++로 작성한 후, python에서 불러와서 사용할 수 있음
- (5) 윈도우 환경은 <https://www.python.org/downloads/windows/> 에서 설치가 가능하고 리눅스 환경은 대부분 이미 설치되어 있음 (32bit / 64bit 선택가능)

- [Python 3.7.1 - 2018-10-20](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86-64 web-based installer](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows help file](#)

: Download [Windows x86-64 web-based installer](#) 실행

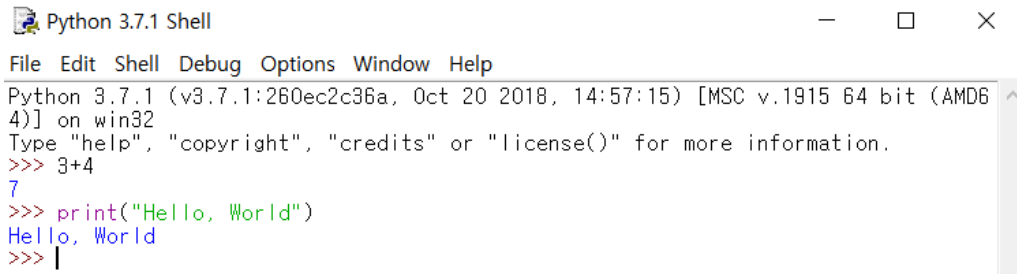


Add Python 3.7 to PATH를 선택하고 Install Now 클릭



(6) Python 실행

프로그램 → Python 3.7 → IDLE(Python 3.7 64-bit)



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 3+4
7
>>> print("Hello, World")
Hello, World
>>> |
```

2.2 Anaconda 소개 및 설치

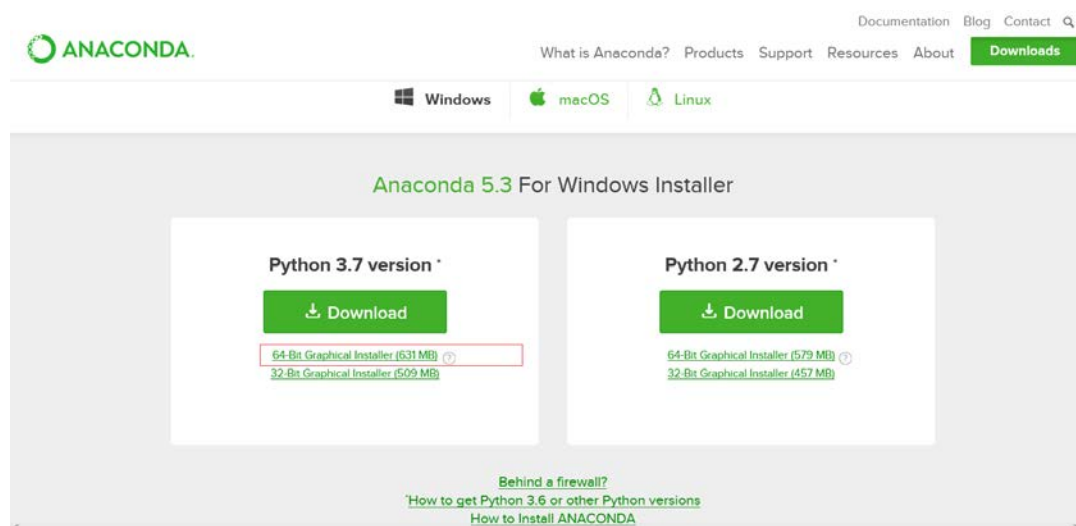
(1) 아나콘다 소개

- Python 기반의 프로그래밍을 위한 오픈소스를 포함하고 있는 개발 플랫폼 (환경)
- 아나콘다는 numpy, matplotlib 와 같이 데이터분석을 위한 다양한 패키지(라이브러리)가 내장되어 있음

(2) 아나콘다 설치

다음의 URL에서 파이썬 버전과 자신의 컴퓨터 비트수에 적합한 아나콘다 버전을 받아 설치

<https://www.anaconda.com/download/>

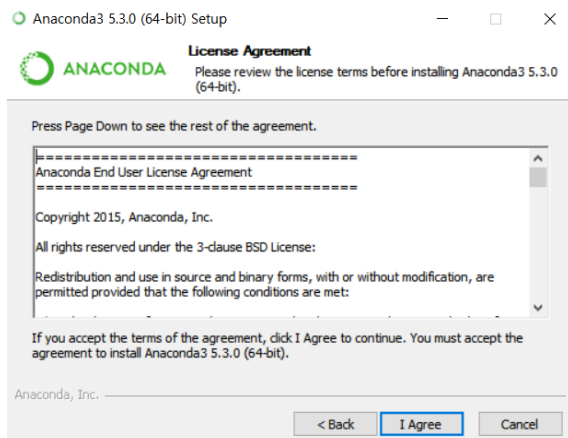


: 64-Bit Graphical Installer (631 MB) 클릭

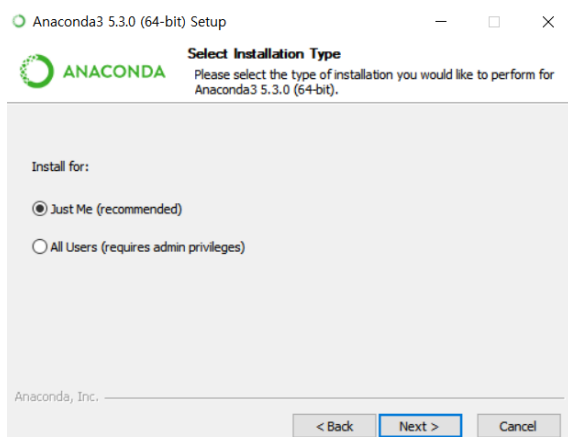
Python Tensorflow를 활용한 머신러닝



: Next 선택

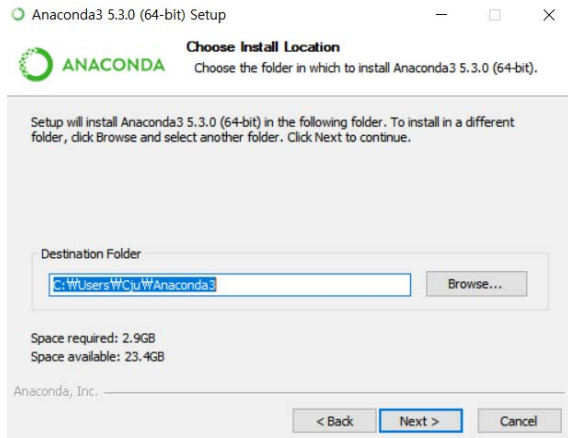


: I Agree 선택

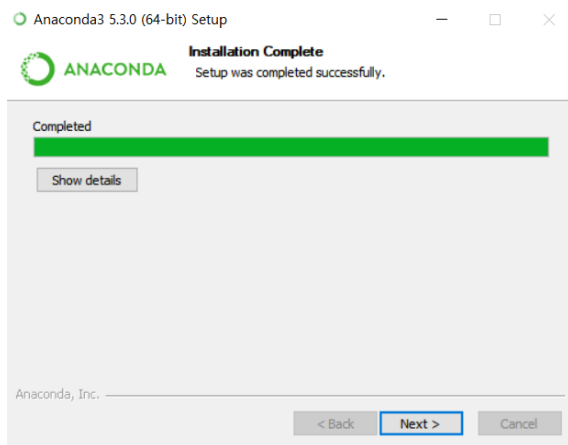


: Just Me 선택

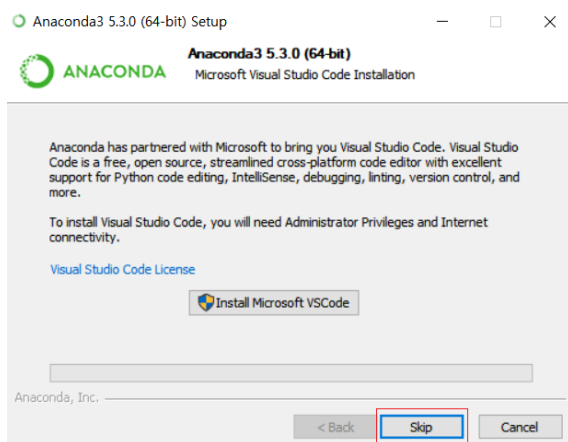
Python Tensorflow를 활용한 머신러닝



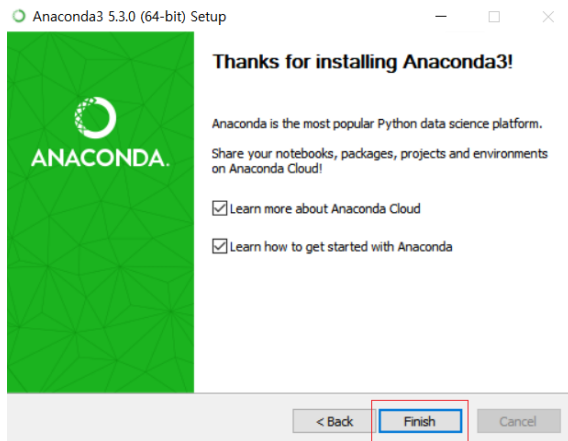
: Next 선택



: Next 선택



: Skip 선택



: Finish 선택

(3) 아나콘다에 있는 Spyder와 ipython 콘솔을 사용하면 프로그램의 실행 결과를 화면에서 볼 수 있어 편리함

- Spyder 편집기는 프로그래밍을 위한 usage 를 보여 주고 Syntax 체크도 제공

2.3 Tensorflow 소개 및 설치

(1) 설치 시작 – Anaconda Prompt 이용

: 윈도우 프로그램 → Anaconda3 (64-bit) → Anaconda Prompt

(2) conda 자체 업데이트

conda update -n base conda

```
Anaconda Prompt
(base) C:\Users\WCju>conda update -n base conda
Solving environment: done

# All requested packages already installed.

(base) C:\Users\WCju>_
```

(3) 설치된 파이썬 패키지 업데이트

conda update --all

```
(base) C:\Users\WCju>conda update --all
```

Proceed ([y]/n)? 에서 y 입력

(4) 텐서플로 설치

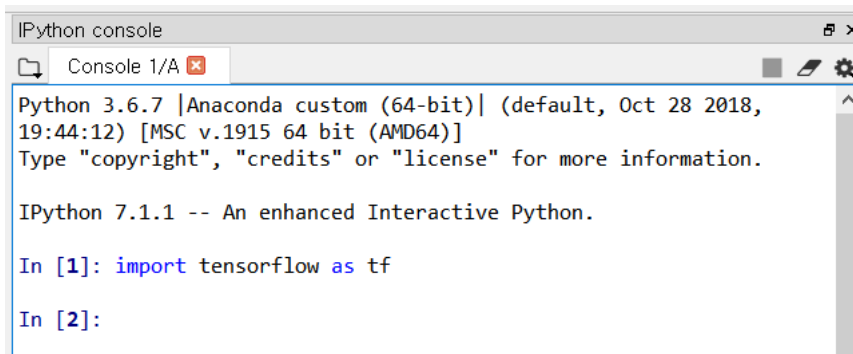
conda install tensorflow

```
(base) C:\Users\WCju>conda install tensorflow
```

Proceed ([y]/n)? 에서 y 입력

(5) 설치 확인

- 프로그램 → Anaconda3 (64-bit) → Spyder 실행 후 IPython Console 창에서 import tensorflow as tf 을 실행 후 오류(error) 메시지가 나타나지 않으면 설치가 잘 된 것임



```
IPython console
Console 1/A
Python 3.6.7 |Anaconda custom (64-bit)| (default, Oct 28 2018, 19:44:12) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.1.1 -- An enhanced Interactive Python.

In [1]: import tensorflow as tf

In [2]:
```

3. Python 프로그래밍 기초

- 인덱싱(Indexing): 파이썬의 인덱싱은 0부터 시작

(note) R의 인덱싱은 1부터 시작

- 대문자와 소문자를 구별
- 들여쓰기가 중요 (블록 구조)

3.1 Python data type

(1) 숫자형

```
>>> a = 3
```

```
>>> b = 4
```

```
>>> a + b
```

```
7
```

```
>>> a * b
```

```
12
```

```
>>> a / b
```

```
0.75
```

(2) 문자열

```
>>> x="Python"
```

```
>>> x
```

```
'Python'
```

```
>>> a = "Life is too short, You need Python"
```

```
>>> a[0]
```

```
'L'
```

```
>>> a[12]
```

```
's'
```

```
>>> a[-1]
```

```
'n'
```

```
>>> a[0:4]
```

```
'Life'
```

```
# Spyder: 문자열 포매팅
```

```
num = 5
```

```
st = "two"
```

```
print("I ate %d apples. so I was sick for %s days." % (num, st)).
```

```
# 문자열 개수 세기(count)
```

```
>>> a = "hobby"
```

```
>>> a.count('b')
```

```
2
```

(3) List

```
# 리스트의 인덱싱
```

```
>>> a = [1, 2, 3]
```

```
>>> a[0]
```

```
1
```

```
>>> a[0] + a[2]
```

```
4
```

```
>>> a[-1]
```

```
3
```

```
# 리스트 정렬(sort)
```

```
>>> a = [1, 4, 3, 2]
```

```
>>> a.sort()
```

```
>>> a
```

```
[1, 2, 3, 4]
```

```
# 리스트에 요소 삽입(insert)
```

```
>>> a = [1, 2, 3]
```

```
>>> a.insert(0, 4)
```

```
[4, 1, 2, 3]
```


리스트 요소 고집어내기(pop)

```
>>> a = [1,2,3]
```

```
>>> a.pop()
```

```
3
```

```
>>> a
```

```
[1, 2]
```

(4) Tuple

인덱싱

```
>>> t1 = (1, 2, 'a', 'b')
```

```
>>> t1[0]
```

```
1
```

```
>>> t1[3]
```

```
'b'
```

리스트는 수정이 가능하지만 튜플은 안 됨

(5) Dictionary

딕셔너리, key: value

Spyder

```
dic1 = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

```
a=dic1['name']
```

```
print(a)
```

```
dic2={1:23,2:14,6:89,'x':78}
```

```
b=dic2[2]
```

```
c=dic2['x']
```

```
print(b)
```

```
print(c)
```

pey

14

78

딕셔너리에 데이터 추가

```
>>> a = {1: 'a'}  
>>> a[2] = 'b'  
>>> a  
{2: 'b', 1: 'a'}
```

3.2 Python 제어 문

(1) 조건: if, elif

```
x = 10  
if x>10:  
    print("x is large")  
else:  
    print("x is small")
```

- if문의 기본구조

```
if 조건문:  
    수행할 문장들  
    ...  
else:  
    수행할 문장들  
    ...
```

다중 조건 판단 elif

```
score=88  
if score>=90:  
    print("High")  
elif score>=80:
```

```
    print("Middle")
else:
    print("Low")
```

(2) 반복: while, for

while 문을 이용한 1에서 10까지의 정수의 합

```
sum=0
i=1
while i<=10:
    sum=sum+i
    i=i+1
print(sum)
```

for 문을 이용한 1에서 10까지의 정수의 합

```
sum=0
for i in range(1,11):
    sum=sum+i
print(sum)
```

3.3 외부 데이터 불러오기

(1) 예제 데이터: cars – Speed and Stopping Distances of Cars (M. Ezekiel), 변수 – speed, dist

외부 데이터 불러오기 – 파일

```
import numpy as np
data_file_name='e:/data/python/cars.txt'
dat=np.genfromtxt(data_file_name,dtype='float32',skip_header=True)
print(np.shape(dat))
speed=dat[:,1]
dist=dat[:,2]
```

```
print(speed)
print(dist)
```

(note1) skip_header=True 또는 skip_header=1 → 데이터의 첫번째 행이 변수명일 경우 지정

(note2) np.shape(dat) → 데이터 객체의 행과 열을 나타냄

외부 데이터 불러오기 - 웹 상의 데이터

```
import pandas as pd
target_url = ("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")
# iris data를 pandas data frame 형식으로 불러옴
iris_data = pd.read_csv(target_url,header=None, prefix="X")
print(iris_data)
print(iris_data.X4)
summary = iris_data.describe()
print(summary)
Sepal_Length = list(iris_data.X0)
print(Sepal_Length)
```

EXAMPLE

Japan credit 데이터를 불러와서 각 열의 값들을 출력 하시오.

3.4 pandas와 numpy 다루기

(1) pandas/numpy

- 고급 데이터 분석과 수치 계산 등의 기능을 제공하는 확장 모듈
- C 언어로 작성되어 있어서 속도가 빠름
- numpy: 다차원 배열과 고수준의 수학 함수 제공

- pandas: 데이터분석을 제공하는 라이브러리, csv 파일 등을 데이터로 읽고 원하는 데이터 형식으로 변환

(2) 데이터프레임

- 데이터프레임(DataFrame): pandas 에서 사용되는 기본 데이터
- 데이터프레임을 정의할 때는 2 차원 리스트를 매개변수로 전달

```
import pandas as pd
```

```
a = pd.DataFrame([
```

```
    [10,20,30],
```

```
    [40,50,60],
```

```
    [70,80,90]
```

```
])
```

```
print(a)
```

```
In [24]: runfile('C:/python/game/sound/echo.py', wdir='C:/python/game/sound')
```

```
   0   1   2
0  10  20  30
1  40  50  60
2  70  80  90
```

1 차원 데이터는 Series 를 사용

```
import pandas as pd
```

```
import numpy as np
```

```
s = pd.Series([1.0, 3.0, 5.0, 7.0, 9.0])
```

```
print(s) # 자료형도 함께 출력됨
```

```
m = np.mean(s)
```

```
print(m)
```

```
In [3]: runfile('C:/Users/Cju/.spyder-py3/temp.py', wdir='C:/Users/Cju/.spyder-py3')
```

```
0    1.0
1    3.0
2    5.0
3    7.0
4    9.0
dtype: float64
5.0
```

(3) 원하는 데이터 추출

1 차원 리스트의 딕셔너리 자료형으로부터 키를 이용하여 원하는 열의 데이터 출력

```
import pandas as pd
```

키, 몸무게, 유형 데이터프레임 생성하기

```
tbl = pd.DataFrame({
    "weight": [80.0, 70.4, 65.5, 45.9, 51.2],
    "height": [170, 180, 155, 143, 154],
    "type": [ "f", "n", "n", "t", "t"]
})
```

몸무게 목록 추출하기

```
print("몸무게 목록")
```

```
print(tbl["weight"])
```

몸무게와 키 목록 추출하기

```
print("몸무게와 키 목록")
```

```
print(tbl[["weight", "height"]])
```

```
In [27]: runfile('C:/python/game/sound/echo.py', wdir='C:/
python/game/sound')
```

몸무게 목록

```
0    80.0
1    70.4
2    65.5
3    45.9
4    51.2
```

Name: weight, dtype: float64

몸무게와 키 목록

```
   weight  height
0    80.0    170
1    70.4    180
2    65.5    155
3    45.9    143
4    51.2    154
```

원하는 위치의 값을 추출할 때는 파이썬 리스트처럼 슬라이스를 사용

```
import pandas as pd
```

```
tbl = pd.DataFrame({
    "weight": [80.0, 70.4, 65.5, 45.9, 51.2],
    "height": [170, 180, 155, 143, 154],
    "type": [ "f", "n", "n", "t", "t"]
})
```

```

}))

print(tbl[2:4]\n", tbl[2:4])

print(tbl[3:]\n", tbl[3:])

In [31]: runfile('F:/python/mymodules/mod2.py', wdir='F:/
python/mymodules')
tbl[2:4]
   height type  weight
2     155    n   65.5
3     143    t   45.9
tbl[3:]
   height type  weight
3     143    t   45.9
4     154    t   51.2

```

원하는 조건 추출

```

import pandas as pd

tbl = pd.DataFrame({
    "weight": [80.0, 70.4, 65.5, 45.9, 51.2, 72.5],
    "height": [170, 180, 155, 143, 154, 160],
    "gender": [ "f", "m", "m", "f", "f", "m"]
})

print("몸무게와 키 목록")

print(tbl[["weight", "height"]])

print("--- height 가 160 이상인 것")

print(tbl[tbl.height >= 160])

print("--- gender 가 m 인 것")

print(tbl[tbl.gender == "m"])

In [35]: runfile('F:/python/mymodules/mod2.py', wdir='F:/
python/mymodules')
몸무게와 키 목록
   weight  height
0    80.0     170
1    70.4     180
2    65.5     155
3    45.9     143
4    51.2     154
5    72.5     160
--- height가 160 이상인 것
   gender  height  weight
0      f     170    80.0
1      m     180    70.4
5      m     160    72.5
--- gender가 m 인 것
   gender  height  weight
1      m     180    70.4
2      m     155    65.5
5      m     160    72.5

```

정렬

```
import pandas as pd

tbl = pd.DataFrame({
    "weight": [80.0, 70.4, 65.5, 45.9, 51.2, 72.5],
    "height": [170, 180, 155, 143, 154, 160],
    "gender": ["f", "m", "m", "f", "f", "m"]
})

print("--- 키로 정렬")

print(tbl.sort_values(by="height"))

print("--- 몸무게로 정렬")

print(tbl.sort_values(by="weight", ascending=False))
```

```
In [36]: runfile('F:/python/mymodules/mod2.py', wdir='F:/
python/mymodules')
--- 키로 정렬
   gender  height  weight
3      f     143    45.9
4      f     154    51.2
2      m     155    65.5
5      m     160    72.5
0      f     170    80.0
1      m     180    70.4
--- 몸무게로 정렬
   gender  height  weight
0      f     170    80.0
5      m     160    72.5
1      m     180    70.4
2      m     155    65.5
4      f     154    51.2
3      f     143    45.9
```

전치

```
import pandas as pd

tbl = pd.DataFrame([
    ["A", "B", "C"],
    ["D", "E", "F"],
    ["G", "H", "I"]
])
```



```
print(tbl)
print("-----")
print(tbl.T)
In [37]: runfile('F:/python/mymodules/mod2.py', wdir='F:/python/mymodules')
      0  1  2
0  A  B  C
1  D  E  F
2  G  H  I
-----
      0  1  2
0  A  D  G
1  B  E  H
2  C  F  I
```

(4) 데이터 조작

```
import numpy as np

# 10 개의 float32 자료형 데이터 생성
v = np.zeros(10, dtype=np.float32)
print(v)

# 연속된 10 개의 uint64 자료형 데이터 생성
v = np.arange(10, dtype=np.uint64)
print(v)

# v 값을 3 배하기
v *= 3
print(v)

# v 의 평균 구하기
print(v.mean())
In [38]: runfile('F:/python/mymodules/mod2.py', wdir='F:/python/mymodules')
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0 1 2 3 4 5 6 7 8 9]
[ 0  3  6  9 12 15 18 21 24 27]
13.5

# 데이터 정규화
import pandas as pd
```

키, 체중, 유형 데이터프레임 생성하기

```
tbl = pd.DataFrame({
    "weight": [80.0, 70.4, 65.5, 45.9, 51.2, 72.5],
    "height": [170, 180, 155, 143, 154, 160],
    "gender": ["f", "m", "m", "f", "f", "m"]
})
```

키와 몸무게 정규화하기

최댓값과 최솟값 구하기

```
def norm(tbl, key):
    c = tbl[key]
    v_max = c.max()
    v_min = c.min()
    print(key, "=", v_min, "-", v_max)
    tbl[key] = (c - v_min) / (v_max - v_min)

norm(tbl, "weight")
norm(tbl, "height")
print(tbl)
```

```
In [39]: runfile('F:/python/mymodules/mod2.py', wdir='F:/
python/mymodules')
weight = 45.9 - 80.0
height = 143 - 180
  gender  height  weight
0      f   0.729730  1.000000
1      m   1.000000  0.718475
2      m   0.324324  0.574780
3      f   0.000000  0.000000
4      f   0.297297  0.155425
5      m   0.459459  0.780059
```

(5) numpy 로 변환

머신러닝 라이브러리 중에서 pandas 의 데이터프레임을 지원하지 않는 경우 numpy 형식으로 변환하여 사용하면 됨

```
In [41]: n=tbl.as_matrix()

In [42]: print(n)
[['f' 0.7297297297297297 1.0]
 ['m' 1.0 0.7184750733137831]
 ['m' 0.32432432432432434 0.5747800586510264]
 ['f' 0.0 0.0]
 ['f' 0.2972972972972973 0.15542521994134909]
 ['m' 0.4594594594594595 0.7800586510263929]]

In [43]: print(tbl)
   gender  height  weight
0      f    0.729730  1.000000
1      m    1.000000  0.718475
2      m    0.324324  0.574780
3      f    0.000000  0.000000
4      f    0.297297  0.155425
5      m    0.459459  0.780059
```

EXAMPLE

[1] Japan credit 데이터를 pandas 데이터 프레임으로 불러와서 각 열(변수)에 대한 평균과 표준편차를 구하시오.

[2] Japan credit 데이터를 pandas 데이터 프레임으로 불러와서 각 열(변수)에 대한 정규화 및 표준화를 수행 하시오. (마지막 열은 제외)

3.5 추가적인 패키지들

(1) sklearn

- scikit-learn
- 다양한 데이터 셋 포함
- 데이터 전처리, 지도/자율 학습 알고리즘 및 평가 기법 포함

(2) scipy

//사이파이//

- 과학기술계산 지원
- 학습알고리즘 및 최적화 기법 제공

(3) statsmodels

- 추정, 검정을 포함한 통계분석 (regression, time-series analysis, ...) 제공

4. Tensorflow 소개

4.1 기본적인 사용

(1) Tensorflow

- 2015년 구글이 공개한 머신러닝을 위한 라이브러리

- Python으로 tensorflow를 구동

- 노드(node, 원)가 '함수/연산'을 의미하고 에지(edge, 화살표)는 텐서(tensor, 숫자, 매트릭스, 배열)를 의미하는 방향성 그래프

- Tensorflow는 텐서(tensor)와 플로우(flow)를 사용하여 프로그램을 구성하고 Session의 생성과 run을 통하여 결과를 얻음

- Tensorflow 구성

: 기본적인 연산 정의 → 정의한 데이터 플로우 그래프를 세션으로 실행

(2) Tensorflow 버전 확인

```
In [44]: import tensorflow as tf
In [45]: print(tf.__version__)
1.12.0
```

(3) 간단한 tensorflow 사용

- 덧셈 1

```
import tensorflow as tf
```

```
# 상수 정의
```

```
a = tf.constant(3)
```

```
b = tf.constant(5)
```

```
# 계산 정의 : tensorflow는 덧셈을 하는 것이 아니라 덧셈이라는 계산을 정의할 뿐임
```

```
c = a + b
```

```
# add_op 객체에 저장되는 것은 덧셈결과(숫자)가 아니라 데이터 플로우 그래프 (객체) 임
```

```
# 세션 수행: 세션을 실행하려면 데이터 플로우 그래프를 run() 메서드의 매개변수로 전달
```

```
sess = tf.Session()
```

```
ret = sess.run(c)
```

```
print(ret)
```

8

- 덧셈 2

```
import tensorflow as tf
```

```
# 상수 정의
```

```
a = tf.constant(2)
```

```
b = tf.constant(3)
```

```
c = tf.constant(4)
```

```
# 연산 정의
```

```
calc1_op = a + b * c
```

```
calc2_op = (a + b) * c
```

```
# 세션 시작
```

```
sess = tf.Session()
```

```
res1 = sess.run(calc1_op)
```

```
print(res1)
```

```
res2 = sess.run(calc2_op)
```

```
print(res2)
```

```
In [8]: runfile('C:/python/game/sound/echo.py', wdir='C:/python/game/sound')
```

14

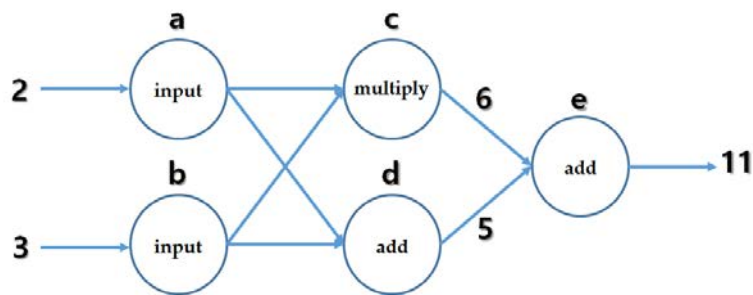
20

(4) Computation graph

- 연산 그래프: 서로 상호작용하는 연산을 만들고 실행하면서 머신러닝 작업을 수행
- 텐서플로의 연산은 데이터 플로우 그래프로 구성
- 노드(node): 산술연산자
- 에지(edge): tensor, 다중 다차원 데이터, 피연산자
- 세션

: session.run – 그래프로부터 출력값을 얻어 냄

(5) Tensorflow 그래프와 코드



```
import tensorflow as tf

a=tf.constant(2, name="input_a")
b=tf.constant(3, name="input_b")
c=tf.multiply(a,b, name="mul_c")
d=tf.add(a,b, name="add_d")
e=tf.add(c,d, name="add_e")

sess=tf.Session()

ret_e=sess.run(e)
print("e=",ret_e)

ret_c=sess.run(c)
print("c=",ret_c)
```

(6) Tensorflow에서 변수 표현

```
import tensorflow as tf

# 상수 정의

a = tf.constant(120, name="a")
b = tf.constant(130, name="b")
c = tf.constant(140, name="c")

# 변수 정의하기

v = tf.Variable(0, name="v")

# 데이터 플로우 그래프 정의

calc_op = a + b + c

assign_op = tf.assign(v, calc_op) # calc_op를 v에 대입

# 세션 실행
```

```
sess = tf.Session()
sess.run(assign_op)
# v의 내용 출력
print( sess.run(v) )
In [9]: runfile('C:/python/game/sound/echo.py', wdir='C:/
python/game/sound')
390
```

(7) Tensorflow의 placeholder

- 값을 넣을 공간을 만들어 두는 기능
- 선언과 동시에 초기화 하는 것이 아니라 일단 선언 후 그 다음 값을 전달
- 실행 시 반드시 데이터가 제공되어야 함 → 데이터를 상수 전달과 같이 할당하는 것이 아니라 다른 텐서를 placeholder에 맵핑 시키는 것임
- placeholder의 parameters

placeholder(dtype, shape=None, name=None)

dtype : 데이터 타입

shape : 입력 데이터의 형태 (상수, 다차원 배열, ...), (default는 None)

name : 해당 placeholder의 이름을 부여 (생략 가능), (default는 None)

```
import tensorflow as tf
# 플레이스홀더 정의
a = tf.placeholder(tf.int32, [3]) # 정수 자료형 3개를 가진 배열
# 배열을 모든 값을 2배하는 연산 정의
b = tf.constant(2)
x2_op = a * b
# 세션 시작
sess = tf.Session()
# 플레이스홀더에 값을 넣고 실행 (feed-dict 이용)
r1 = sess.run(x2_op, feed_dict={ a:[1, 2, 3] })
print(r1)
```

```
r2 = sess.run(x2_op, feed_dict={ a:[10, 20, 10] })  
print(r2)
```

```
In [10]: runfile('C:/python/game/sound/echo.py', wdir='C:/python/game/sound')  
[2 4 6]  
[20 40 20]
```

```
import tensorflow as tf
```

```
a = tf.placeholder(tf.int32, [None]) # None: 고정되지 않은 원하는 크기의 배열 사용
```

```
# 배열의 모든 값을 10배하는 연산 정의하기
```

```
b = tf.constant(10)
```

```
x10_op = a * b
```

```
# 세션 시작
```

```
sess = tf.Session()
```

```
# 플레이스홀더에 값을 넣어 실행
```

```
r1 = sess.run(x10_op, feed_dict={a: [1,2,3,4,5]})
```

```
print(r1)
```

```
r2 = sess.run(x10_op, feed_dict={a: [10,20]})
```

```
print(r2)
```

```
In [11]: runfile('C:/python/game/sound/echo.py', wdir='C:/python/game/sound')  
[10 20 30 40 50]  
[100 200]
```

4.2 기본적인 tensorflow 프로그램

(1) Python 기본과 Tensorflow

```
# Hi, Python! – general python
```

```
x1 = "Hi,"
```

```
x2 = " Python"
```

```
Y = x1 + x2
```

```
print(Y)
```


Hi, Python! - tensorflow

```
import tensorflow as tf
x1 = tf.constant("Hi,")
x2 = tf.constant(" Python")
Y = x1 + x2
with tf.Session() as sess:
    ret = sess.run(Y)
print(ret)
```

다음 코드는 10이 출력

```
x = 1
y = x + 9
print(y)
```

Tensorflow를 이용하여 동일한 결과 출력

```
import tensorflow as tf
x = tf.constant(1)
y = tf.Variable(x+9)
model = tf.global_variables_initializer() # 변수 초기화 함수 호출
# 앞에서 생성한 model을 사용하여 변수 y의 값을 연산한 후 결과 출력
with tf.Session() as session:
    session.run(model) # y 값은 session 이 실행되기 전까지 연산 되지 않음
    print(session.run(y))
```

(2) Tensorflow 프로그래밍 - 정수 a와 b의 곱하기

```
import tensorflow as tf
a = tf.placeholder("int32") # placeholder 로 명명된 기본 자료구조 정의
b = tf.placeholder("int32")
y = tf.multiply(a,b) # 정수 a와 b의 곱셈을 리턴
```

```
sess = tf.Session() # 세션을 생성해 실행 흐름을 관리
print(sess.run(y, feed_dict={a:2,b:5})) # 연산 결과 출력
```

(3) 텐서 자료 구조

- tensor

: tensorflow의 기본 자료구조, 데이터 플로우 그래프에서 에지 연결

: 다차원 배열이나 리스트로 구성된 구조

- tensor는 rank, shape, type의 3가지 매개변수로 구성

rank: tensor의 차원, 1=벡터, 2=행렬, ..., N=N차원 배열

shape: tensor의 행과 열의 개수

type: tensor의 데이터 형식

(4) 1차원 tensor

Numpy의 array를 이용한 1차원 tensor 생성

```
import numpy as np
tensor_1d = np.array([1.3,1,4.0,23.99])
print(tensor_1d)
print(tensor_1d[0])
print(tensor_1d[2])
print(tensor_1d.ndim) # rank 조회
print(tensor_1d.shape) # shape 조회
print(tensor_1d.dtype) # type 조회
```

tensorflow의 텐서로 변환

```
import tensorflow as tf
import numpy as np
tensor_1d = np.array([1.3,1,4.0,23.99])
tf_tensor = tf.convert_to_tensor(tensor_1d, dtype=tf.float64)
with tf.Session() as sess:
```

```
print(sess.run(tf_tensor))
print(sess.run(tf_tensor[0]))
print(sess.run(tf_tensor[2]))
```

`convert_to_tensor` 함수: Numpy의 배열, 파이썬리스트, 파이썬스칼라 등 다양한 파이썬 객체를
tensor 형식으로 변환

```
[ 1.3  1.   4.  23.99]
1.3
4.0
```

(5) 2차원 tensor

행렬 이용하기

```
import tensorflow as tf
import numpy as np
tensor_2d=np.array([(1,2,3,4),(4,5,6,7),(8,9,10,11),(12,13,14,15)])
print(tensor_2d)
print(tensor_2d[3][3])
print(tensor_2d[0:2,0:2])
tf_tensor = tf.convert_to_tensor(tensor_2d, dtype=tf.float64)
with tf.Session() as sess:
    print(sess.run(tf_tensor))
```

tensor 다루기

```
import tensorflow as tf
import numpy as np
matrix1=np.array([(2,2,2),(2,2,2),(2,2,2)],dtype='int32')
matrix2=np.array([(1,1,1),(1,1,1),(1,1,1)],dtype='int32')
print("matrix1 =")
print(matrix1)
print("matrix2 =")
print(matrix2)
# matrix1=tf.constant(matrix1)
# matrix2=tf.constant(matrix2)
matrix_product=tf.matmul(matrix1,matrix2)
```

```
matrix_sum=tf.add(matrix1,matrix2)
```

```
with tf.Session() as sess:
```

```
    result1=sess.run(matrix_product)
```

```
    result2=sess.run(matrix_sum)
```

```
print("matrix1*matrix2 =")
```

```
print(result1)
```

```
print("matrix1+matrix2 =")
```

```
print(result2)
```

EXAMPLE

다음 행렬의 연산 결과를 출력하는 프로그램을 작성하시오.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} * \begin{pmatrix} 5 & 6 \\ 7 & 9 \end{pmatrix} + \begin{pmatrix} 2 & 1 \\ 2 & 4 \end{pmatrix}$$

(solution)

```
import tensorflow as tf
```

```
import numpy as np
```

```
matrix1=np.array([(1,2),(3,4)],dtype='int32')
```

```
matrix2=np.array([(5,6),(7,9)],dtype='int32')
```

```
matrix3=np.array([(2,1),(2,4)],dtype='int32')
```

```
matrix_product=tf.matmul(matrix1,matrix2)
```

```
matrix_sum=tf.add(matrix_product,matrix3)
```

```
with tf.Session() as sess:
```

```
    result1=sess.run(matrix_product)
```

```
    result2=sess.run(matrix_sum)
```

```
print("matrix1*matrix2 =")
```

```
print(result1)
```

```
print("matrix1+matrix2 =")
```

```
print(result2)
```

(6) 난수

균일 분포 (Uniform distribution)

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

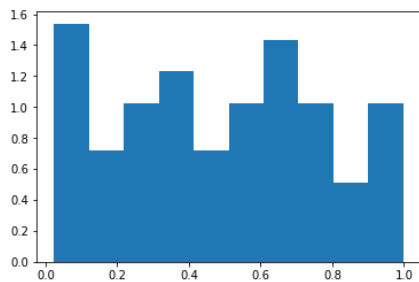
uniform = tf.random_uniform([100],minval=0,maxval=1,dtype=tf.float32)

with tf.Session() as session:
    print(uniform.eval())

    plt.hist(uniform.eval(),normed=True) # 상대빈도로 출력

    plt.show()

In [45]: runfile('F:/python/mymodules/mod2.py', wdir='F:/
python/mymodules')
[0.5597875  0.47928822 0.87417614 ... 0.39429343 0.01808977
0.30018544]
```



정규분포 (Normal distribution, Gaussian distribution)

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

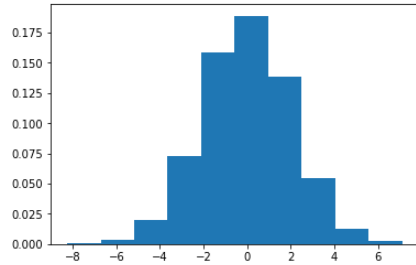
norm = tf.random_normal([10000], mean=0, stddev=2)

with tf.Session() as session:
    print(norm.eval())

    plt.hist(norm.eval(),normed=True)

    plt.show()
```

```
In [48]: runfile('F:/python/mymodules/mod2.py', wdir='F:/python/mymodules')  
[-1.1597803 -3.3429701 -2.6335025 ... 3.2924457  
-0.47597122  
0.15731491]
```



EXAMPLE

- [1] $n=10$, $p=0.5$ 인 이항분포(binomial distribution)를 따르는 난수 1000개를 생성하고 이 값들의 히스토그램을 작성하시오.
- [2] $\lambda=3$ 인 포아송분포(Poisson distribution)를 따르는 난수 1000개를 생성하고 이 값들의 히스토그램을 작성하시오.

5. Linear Regression Analysis

5.1 회귀분석 – 통계학

(1) Multiple linear regression

여러 변수들 사이의 관계를 결정하는 문제

x_1, \dots, x_r : 독립변수(independent variable), 입력변수(input), 설명변수(explanatory)

Y : 종속변수(dependent variable), 출력변수(output), 반응변수(response)

$\beta_0, \beta_1, \dots, \beta_r$: 회귀계수(regression parameters)

$$Y = \beta_0 + \beta_1 x_1 + \dots + \beta_r x_r + e$$

e : 평균이 0인 확률변수로 가정

위 식의 또 다른 표현:

$$E[Y|x] = \beta_0 + \beta_1 x_1 + \dots + \beta_r x_r$$

$E[Y|x]$: 입력변수들인 x 가 주어졌을 때 반응치(Y)의 기댓값

상수 $\beta_0, \beta_1, \dots, \beta_r$: 회귀계수(regression coefficients), **데이터로부터 추정**

- 단순회귀(simple regression): 독립변수가 1개
- 다중회귀(multiple regression): 독립변수가 여러 개

최소자승추정 (least squared estimation)

$$Y = \alpha + \beta x + e$$

단순선형 회귀모형(simple linear regression)

A: α 에 대한 추정량

B: β 에 대한 추정량

SS (sum of squared): 실제값(actual response values)과 예측값(estimated responses values)의 차이

$$SS = \sum_{i=1}^n (Y_i - A - Bx_i)^2$$

$$\frac{\partial SS}{\partial A} = -2 \sum_{i=1}^n (Y_i - A - Bx_i)$$

$$\frac{\partial SS}{\partial B} = -2 \sum_{i=1}^n x_i (Y_i - A - Bx_i)$$

위의 편미분 결과를 0으로 두면 SS를 최소로 하는 A와 B의 값을 구할 수 있다.

$$\begin{aligned} \sum_{i=1}^n Y_i &= nA + B \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i Y_i &= A \sum_{i=1}^n x_i + B \sum_{i=1}^n x_i^2 \end{aligned}$$

$$\begin{aligned} B &= \frac{\sum_i (x_i - \bar{x})(Y_i - \bar{Y})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i x_i Y_i - n\bar{x}\bar{Y}}{\sum_i x_i^2 - n\bar{x}^2} = \frac{S_{xy}}{S_{xx}} \\ A &= \bar{Y} - B\bar{x} \end{aligned}$$

(2) 회귀분석의 성능평가

- 결정계수 (Coefficient of determination)

$$R^2 = \frac{SSR}{SST}$$

$$0 \leq R^2 \leq 1$$

SST (total sum of squared deviation)

SSR (sum of squares due to regression)

SSE (sum of squared errors)

$$SST = \sum (y_i - \bar{y})^2$$

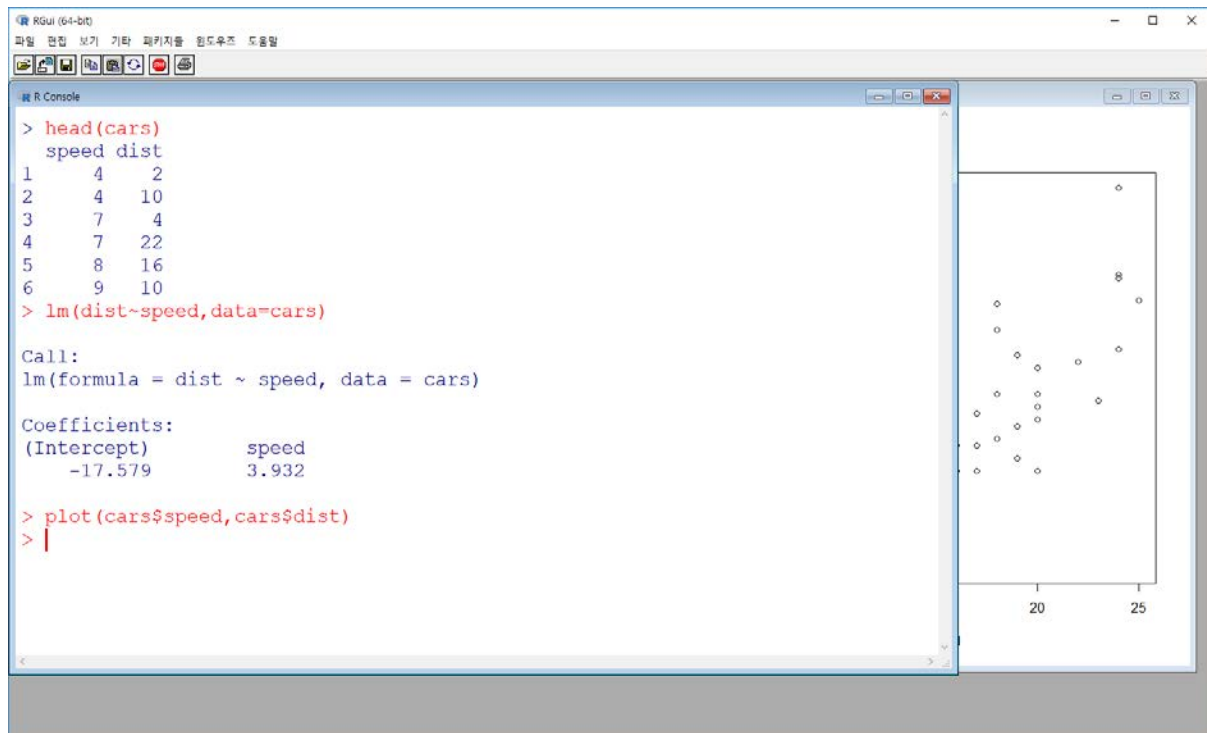
$$SSR = \sum (\hat{y}_i - \bar{y})^2$$

$$SSE = \sum (y_i - \hat{y}_i)^2$$

$$SST = SSR + SSE$$

: 결정계수가 클수록 모형의 설명력이 큼

5.2 R을 이용한 회귀분석



5.3 Tensorflow를 이용한 회귀분석

회귀계수 학습 (추정)

```
import numpy as np
```

```
import tensorflow as tf
```

```
import matplotlib.pyplot as plt
```

```
data_file_name='L:/data/python/cars.txt'
```

```
dat=np.genfromtxt(data_file_name,dtype='float32',skip_header=True)
```

```
speed=dat[:,1]
```

```
dist=dat[:,2]
```

```
X=tf.placeholder("float32")
```

```
Y=tf.placeholder("float32")
```

```
init_b0=0.5
```

```
init_b1=0.5
```

```
b0=tf.Variable(init_b0)
```

```
b1=tf.Variable(init_b1)
```

```
y=b0+b1*X
```

```
cost=tf.reduce_mean(tf.square(y-Y))
```

```
opti=tf.train.GradientDescentOptimizer(0.001)
```

```
training=opti.minimize(cost)
```

```
init=tf.global_variables_initializer()
```

```
with tf.Session() as sess:
```

```
    sess.run(init)
```

```
    for i in range(0,5000):
```

```
        sess.run(training, feed_dict={X:speed, Y:dist})
```

```
        if(i%100==0):
```

```
            cost_out=sess.run(cost,feed_dict={X:speed, Y:dist})
```

```
            b0_out=sess.run(b0,feed_dict={X:speed, Y:dist})
```

```
            b1_out=sess.run(b1,feed_dict={X:speed, Y:dist})
```

```
            print(i, "session is performed.. cost is ",cost_out," b1=", b1_out, "b0=", b0_out)
```

```
plt.plot(speed, dist, 'o')
```

```
plt.show()
```

(note1) placeholder(dtype, shape=None, name=None)

(note2) 초기값(init_b0=1.0, init_b1=1.0)에 따라 추정된 회귀계수 값이 달라짐

(note3) 반복수(for i in range(0,5000):)에 따라 추정된 회귀계수 값이 달라짐

추정된 회귀계수 비교

회귀계수	통계학 (최소자승법)	머신러닝 (Cost 최적화)				
	반복 없음	1,000 반복	5,000 반복	10,000 반복	20,000 반복	30,000 반복
b0	-17.5790	-2.4541	-10.9563	-15.2201	-17.2798	-17.5409
b1	3.9320	3.0516	3.5467	3.7950	3.9150	3.9302

EXAMPLE

[1] 다음 데이터를 이용하여 추정된 회귀식을 구하시오.

$$Y = b_0 + b_1x$$

광고료(X)	매출액(Y)
4	9
8	20
9	22
8	15
8	17
12	30
6	18
10	25
6	10
9	20

[2] 광고료와 매출액에 대한 산점도를 그리시오.

5.4 회귀분석 – Simulation data

$$Y = Ax + b$$

데이터 모델

```
import numpy as np
number_of_points = 200
x_point = []
y_point = []
a = 0.22
b = 0.78
for i in range(number_of_points):
    x = np.random.normal(0.0, 0.5)
    y = a * x + b + np.random.normal(0.0, 0.1)
    x_point.append([x])
    y_point.append([y])
```

```
import matplotlib.pyplot as plt
```

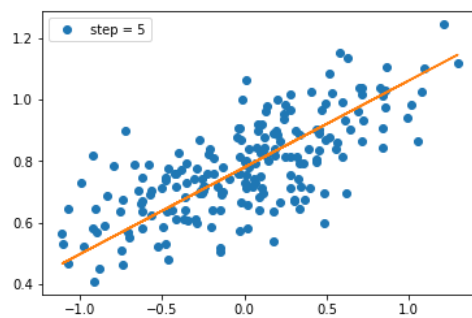
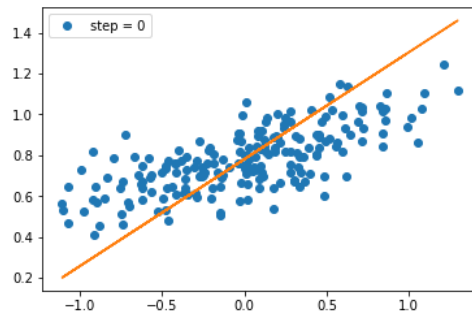
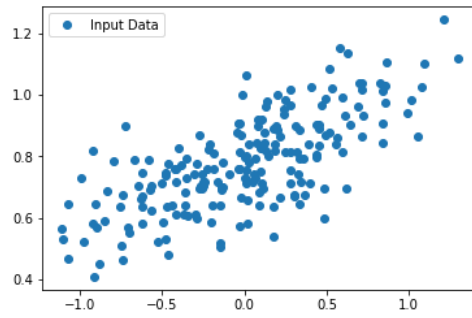
```
plt.plot(x_point,y_point, 'o', label='Input Data')
plt.legend()
plt.show()
```

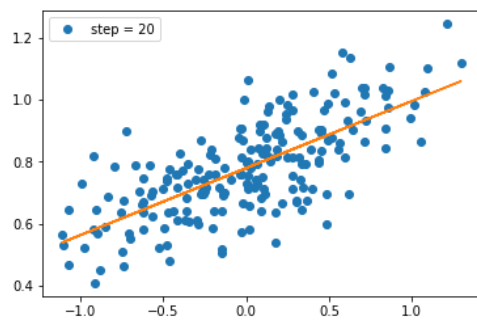
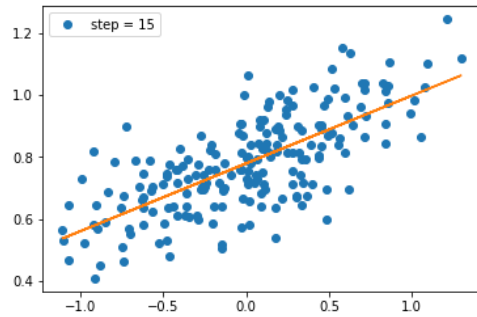
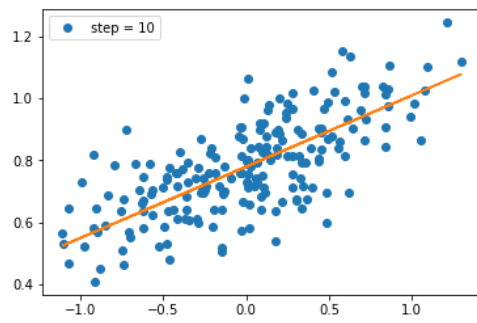
비용함수와 경사하강법

```
import tensorflow as tf

# A와 b를 tf.Variable로 정의하고 임의의 값을 할당
A = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
# A는 -1에서 1사이의 임의의 값으로, b는 0으로 초기화
B = tf.Variable(tf.zeros([1]))
# y와 x의 선형 관계식 정의
y = A * x_point + B
# 비용함수(cost function) 정의: 예측값과 실제값의 차이 -> mean squared error (MSE)
cost_function = tf.reduce_mean(tf.square(y - y_point))
# tensorflow에서 경사하강법(gradient descent)을 이용하여 cost_function을 최소화
optimizer = tf.train.GradientDescentOptimizer(0.5) # 0.5는 학습률(learning rate)
train = optimizer.minimize(cost_function)
# 변수 초기화
model = tf.global_variables_initializer()
# A와 b의 값을 도출할 수 있게 세션을 통해 모델 학습을 20회 반복하도록 설정
with tf.Session() as session: # 모델 시뮬레이션을 수행
    session.run(model)
    for step in range(0,21):
        session.run(train) # 각 스텝마다 학습을 수행
        if (step % 5) == 0: # 매 5번째 스텝마다 점이 어떤 패턴인지 출력
            plt.plot(x_point, y_point, 'o',label='step = {}'.format(step))
# 학습된 A와 b를 이용한 회귀직선  $y=Ax+b$  출력
plt.plot(x_point, session.run(A) * x_point + session.run(B))
plt.legend()
plt.show()
```

Python Tensorflow를 활용한 머신러닝





EXAMPLE

Iris 데이터를 이용하여 다음의 회귀식을 추정하시오.

$$\text{Sepal. Width} = b_0 + b_1 \text{Sepal. Length} + b_2 \text{Patal. Width}$$

6. Logistic Regression Analysis

6.1 로지스틱 회귀분석

(1) Logistic regression

- 이항분포(binomial distribution)를 사용한 일반화선형모형 (generalized linear model, GLM)
- GLM은 확률분포, 링크함수, 선형예측식의 지정이 필요한 통계 모형

(2) 로지스틱회귀 GLM

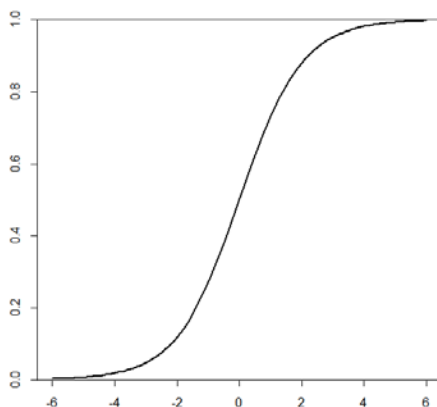
- 확률분포 → 이항분포
- 링크함수 → 로짓링크함수 (logit link function)
- 선형예측식 → $b_0 + b_1x_1 + \dots$

(3) 로지스틱함수 (logistic function)

- 제약조건, $0 \leq q_i \leq 1$ (q_i 는 확률)

$$q_i = \text{logistic}(z_i) = \frac{1}{1 + \exp(-z_i)}$$

- 선형예측식 $z_i = \beta_1 + \beta_2x_i + \dots$
- q_i 가 z_i 의 로지스틱함수로 표현된다고 가정하면 선형예측식 z_i 가 어떠한 값을 가져도 $0 \leq q_i \leq 1$ 의 조건은 만족됨 (Probability, score, ...)



(4) 로지스틱 함수의 변형

$$q_i = \frac{1}{1 + e^{-z_i}}$$

$$\log\left(\frac{q_i}{1-q_i}\right) = z_i$$

좌변의 식이 로짓 함수 (logit function)

$$\text{logit}(q_i) = \log\left(\frac{q_i}{1-q_i}\right)$$

- 로짓 함수와 로지스틱 함수는 서로 역함수 관계
- 따라서 다음과 같은 로지스틱 회귀식을 구함

$$\text{logit}(q_i) = b_0 + b_1x_1 + \dots$$

6.2 Python을 이용한 로지스틱 회귀

```
import numpy as np
np.random.seed(456)

import tensorflow as tf
tf.set_random_seed(456)

from sklearn.linear_model import LogisticRegression

import statsmodels.api as sm

N = 100

# np.identity(2), np.eye(2) : 단위행렬
x_zeros = np.random.multivariate_normal(mean=np.array((-1, -1)), cov=1*np.eye(2), size=(N//2,))
y_zeros = np.zeros((N//2,))
x_ones = np.random.multivariate_normal(mean=np.array((1, 1)), cov=1*np.eye(2), size=(N//2,))
y_ones = np.ones((N//2,))

x_np = np.vstack([x_zeros, x_ones])
y_np = np.concatenate([y_zeros, y_ones])

y=y_np
X=x_np
```



```
X_with_constant=sm.add_constant(X,prepend=True)
model = LogisticRegression()
model = model.fit(X_with_constant,y)
print(model.coef_)
print(model.intercept_)
```

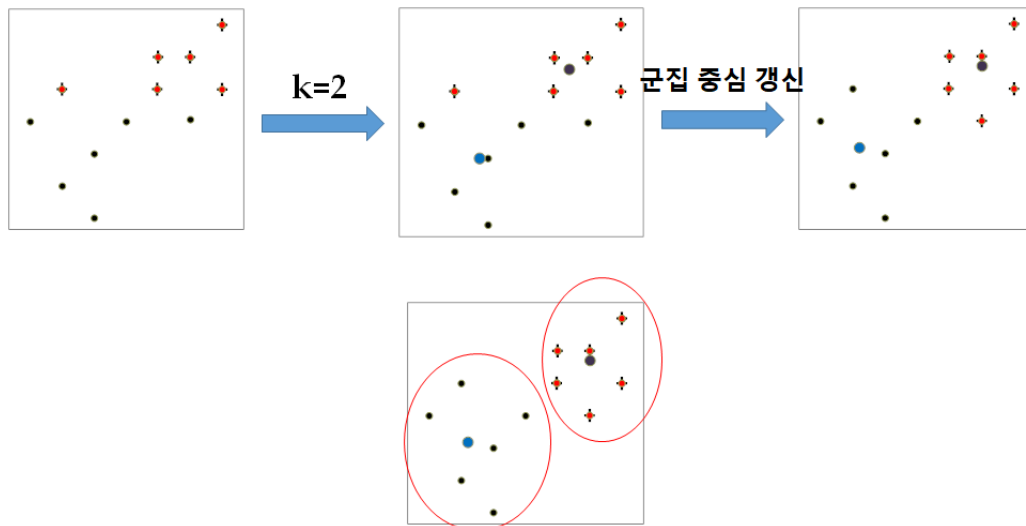
EXAMPLE

JAPAN Credit 데이터를 이용하여 로지스틱 회귀식을 추정하시오.

7. K – Means Clustering

7.1 K-평균 군집화

(1) K-means clustering 개요



(2) K-means clustering 절차

- 군집수 K 결정 (Silhouette Width)
- 초기 K 개의 군집 중심 결정 (random 또는 분석가가 결정)
- 군집 중심에 가장 가까운 객체들끼리 묶여 감
- 최종적으로 더 이상의 군집 변동이 없으면 학습 종료

7.2 K-means clustering 실습

```
from sklearn.cluster import KMeans  
from sklearn import datasets  
import numpy as np  
import matplotlib.pyplot as plt
```

```
# np.random.seed(5)
# centers = [[1, 1], [-1, -1], [1, -1]]

iris = datasets.load_iris()
X = iris.data
y = iris.target
Sepal_Length=X[:,0]
Sepal_Width=X[:,1]
Patal_Length=X[:,2]
Patal_Width=X[:,3]
Species=y
print(Species)
clustering = KMeans(n_clusters=3)
clustering.fit(X)
y_predict = clustering.predict(X)
plt.scatter(Sepal_Length,Sepal_Width)
```

EXAMPLE

Japan credit 데이터를 이용한 군집화

8. Deep Learning 입문

8.1 딥러닝이란?

(1) Deep learning

- Deep learning: 심층 학습이 가능한 신경망 모형 기반의 머신러닝
- 입력데이터에 대한 특징 추출과 문제해결을 위한 복잡한 (비선형) 함수를 학습하기 위하여 다수의 층(layer)을 갖는 신경망 구조
- 많은 데이터와 컴퓨팅 자원을 필요로 함
- 통계학 뿐만 아니라 기존의 머신러닝 기법에 비해 월등한 성능향상을 보임 (음성인식, 이미지인식, ...)

(2) 딥러닝의 문제 해결

- 학습을 통하여 입력 데이터로부터 적합한 특징을 추출하면서 문제해결을 위한 모형을 구축
 - 입력층에 가까운 층: 낮은 수준의 특징이 학습, 출력층에 가까운 층: 더 추상적인 특징이 학습
- 계층적 특징 (hierarchical feature) 학습

8.2 Convolutional Neural Network (CNN)

(1) CNN

- 동물의 시각 피질 (visual cortex) 구조에 영향을 받은 신경망 구조
 - 시각 피질의 각 신경세포는 시야 내의 특정 영역의 자극만 수용 (해당 영역의 특정 특징에 대해서만 반응)
 - 시각 인식: 시각 자극이 1차 시각 피질을 통해 처리, 2차 시각 피질, 3차 시각 피질, ...
- 계층적인 정보처리 (정보가 계층적으로 처리되어 가면서 점차 추상적인 특징이 추출되어 시각 인식이 이루어짐)

(2) CNN 구조



8.3 신경망 모형

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

data_file_name='e:/data/python/cars.txt'

dat=np.genfromtxt(data_file_name,dtype='float32',skip_header=True)

speed_data=dat[:,1]
dist_data=dat[:,2]

speed=np.reshape(speed_data, [1,-1])
dist=np.reshape(dist_data, [1,-1])

x=tf.placeholder(dtype=tf.float32, shape=[1,None])
y=tf.placeholder(dtype=tf.float32, shape=[1,None])

hidden_number=10

b1_hidden=tf.Variable(tf.random_normal([hidden_number,1]))
b0_hidden=tf.Variable(tf.random_normal([hidden_number,1]))
layer1_out=tf.nn.sigmoid(tf.matmul(b1_hidden,x)+b0_hidden)

b1_out=tf.Variable(tf.random_normal([1,hidden_number]))
b0_out=tf.Variable(tf.random_normal([1,1]))
y_out=tf.matmul(b1_out,layer1_out)+b0_out

cost=tf.nn.l2_loss(y_out-y)

optimizer=tf.train.AdamOptimizer(0.1)
training=optimizer.minimize(cost)

init=tf.global_variables_initializer()
```

```
with tf.Session() as sess:

    sess.run(init)

    for i in range(500):

        sess.run(training,feed_dict={x:speed, y:dist})

    speed_data=np.linspace(0,20,50)

    x_test=[speed_data]

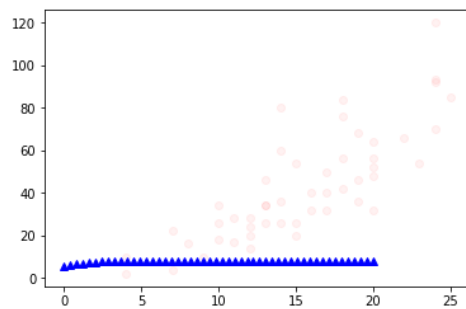
    y_test=sess.run(y_out, feed_dict={x: x_test})
```

```
plt.plot(speed, dist, 'ro', alpha=0.05)

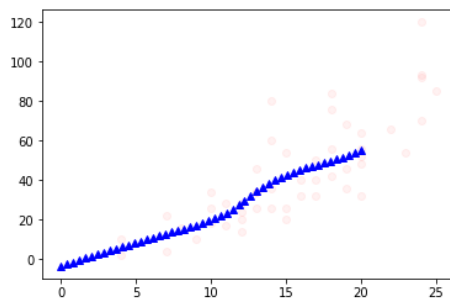
plt.plot(x_test,y_test, 'b^', alpha=1)

plt.show()
```

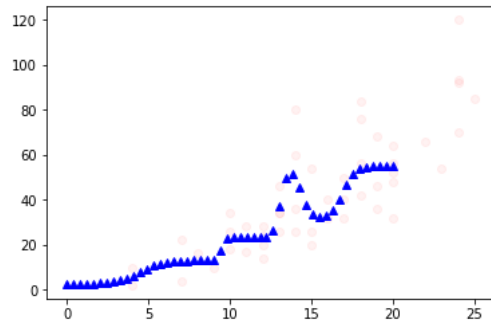
10회 반복



500회 반복

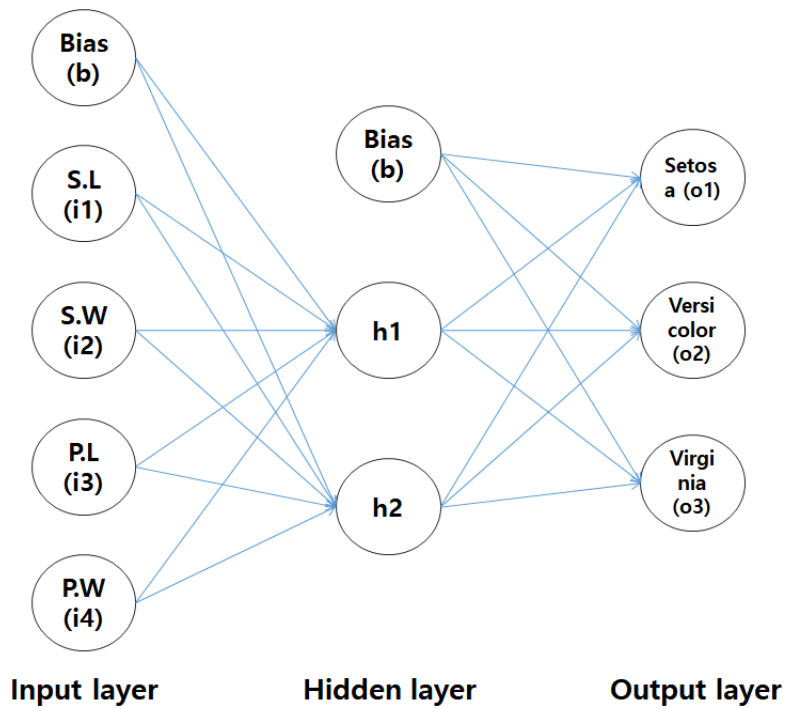


5000회 반복



8.4 신경망 모형 실습

(1) IRIS 데이터 – classification (Sigmoid 함수)



(2) IRIS 데이터 – regression (Linear 함수)

EXAMPLE

[1] JAPAN Credit 데이터를 이용하여 신경망 모형을 수행하시오.

9. 실습 예제

9.1 연관 규칙 마이닝, Association Rule Mining – ARM

(1) 아이템(items)과 거래(transactions) 데이터를 이용하여 아이템 간의 연관성을 분석 (아이템=사건, 거래=실험결과)

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

(2) 아이템과 트랜잭션 데이터 집합

$I=\{i_1, i_2, \dots, i_n\}$: n개의 아이템 집합

$T=\{t_1, t_2, \dots, t_m\}$: m개의 트랜잭션 집합

(ex) Wal mart data

$I=\{\text{Beer, Nuts, Diaper, Coffee, Eggs, Milk}\}$

$T=\{10,20,30,40,50\}$

(3) 개별 트랜잭션은 번호(unique identical number)와 이에 포함된 아이템들로 구성

$$t_j = (i_{j1}, i_{j2}, \dots, i_{jp})$$

(ex) Wal mart data

$t_{10}=(\text{Beer, Nuts, Diaper})$

(4) 연관규칙의 표현 - X 아이템이 거래되고 나서 Y 아이템이 거래된 것을 의미

$$X \rightarrow Y$$

X와 Y는 아이템집합에 포함된 아이템

X: 선행사건(antecedent), lhs(left hand side)

Y: 후행사건(consequent), rhs(right hand side)

(5) ARM의 3가지 평가 측도(evaluation measures)

- 지지도 (support): 두 사건(event) A와 B에 대하여 A와 B가 동시에 발생할 확률

$$P(A \cap B)$$

- 신뢰도 (confidence): A가 발생했다는 조건 하에서 B가 발생할 확률

$$P(B|A)$$

- 향상도 (lift)

$$\frac{P(B|A)}{P(B)}$$

(6) 지지도와 신뢰도 최소 확률값을 정하여 이 값보다 큰 규칙들에 대하여 의미를 부여

: ARM에서는 최소 임계값(minimum threshold)

(7) Support

$$\text{support}(X \rightarrow Y) = P(X \cap Y) = \frac{\text{X와 Y를 함께 포함하고 있는 트랜잭션 수}}{\text{전체 트랜잭션 수}}$$

$$0 \leq \text{support}(X \rightarrow Y) \leq 1$$

$(X \rightarrow Y)$ 와 $(Y \rightarrow X)$ 의 지지도 값은 같기 때문에 두 규칙 간의 차이를 알 수 없음.

$$\text{support}(X \rightarrow Y) = P(X \cap Y) = P(Y \cap X) = \text{support}(Y \rightarrow X)$$

(8) Confidence

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{P(X \cap Y)}{P(X)} = \frac{\text{X와 Y를 함께 포함하고 있는 트랜잭션 수}}{\text{X를 포함한 트랜잭션 수}}$$

$$0 \leq \text{confidence}(X \rightarrow Y) \leq 1$$

X가 발생하였다는 조건 하에서 Y가 발생할 확률로 정의되는 신뢰도는 다음과 같이 X와 Y의 지지도($P(X \cap Y)$)를 X의 지지도($P(X)$)로 나눈 값임

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{P(X \cap Y)}{P(X)} = \frac{\text{support}(X \rightarrow Y)}{\text{support}(X)}$$

(9) Lift

$$\text{lift}(X \rightarrow Y) = \frac{\text{confidence}(X \rightarrow Y)}{\text{support}(Y)} = \frac{P(Y | X)}{P(Y)} = \frac{P(X \cap Y)}{P(X)P(Y)} = \frac{\text{support}(X \rightarrow Y)}{\text{support}(X)\text{support}(Y)}$$

$$0 \leq \text{lift}(X \rightarrow Y) < \infty$$

- 향상도 값은 확률이 아니고 이론적으로 0에서 무한대(∞) 사이의 값을 갖음
- 향상도 값이 1이 되면 X와 Y는 서로 독립(independent)이 됨

$$\text{lift}(X \rightarrow Y) = \frac{P(X \cap Y)}{P(X)P(Y)} = 1, \quad P(X \cap Y) = P(X)P(Y)$$

- 향상도 값에 따른 X와 Y의 관계

$$\text{lift}(X \rightarrow Y) = \begin{cases} > 1 & X \text{ and } Y \text{ are complementary (상호보완)} \\ 1 & X \text{ and } Y \text{ are independent (독립)} \\ < 1 & X \text{ and } Y \text{ are substitutive (상호 대체)} \end{cases}$$

(10) Example

[Wal Mart Case]

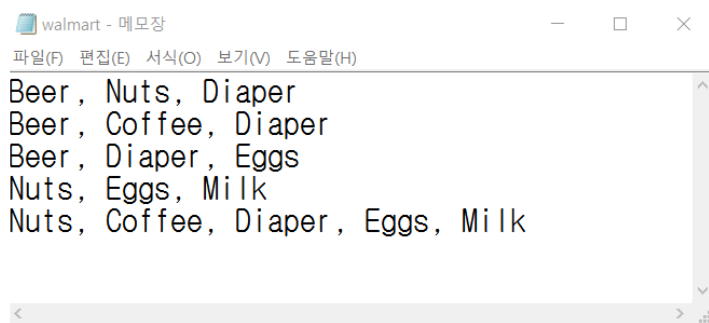
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

$$P(\text{beer}) = , P(\text{diaper}) = , P(\text{beer} \cap \text{diaper}) =$$

$$P(\text{beer} | \text{diaper}) = , P(\text{diaper} | \text{beer}) =$$

$$\frac{P(\text{diaper} | \text{beer})}{P(\text{diaper})} = , \frac{P(\text{beer} | \text{diaper})}{P(\text{beer})} =$$

(11) 실습 코드



```
> library(arules)
```

```
> library(arulesViz)
```

```
> tr = read.transactions("c:/data/walmart.txt", format = "basket", sep = ",")
```

```
> tr
```

transactions in sparse format with

5 transactions (rows) and

6 items (columns)

```
> rules = apriori(tr, parameter = list(support = 0.1, confidence = 0.8))
```

```
> rules
```

set of 46 rules

```
> inspect(rules)
```

```
> inspect(rules)
```

	lhs	rhs	support	confidence	lift
1	{}	=> {Diaper}	0.8	0.8	1.000000
2	{Coffee}	=> {Diaper}	0.4	1.0	1.250000
3	{Milk}	=> {Nuts}	0.4	1.0	1.666667
4	{Milk}	=> {Eggs}	0.4	1.0	1.666667
5	{Beer}	=> {Diaper}	0.6	1.0	1.250000
6	{Coffee,Milk}	=> {Nuts}	0.2	1.0	1.666667
7	{Coffee,Nuts}	=> {Milk}	0.2	1.0	2.500000
8	{Coffee,Milk}	=> {Eggs}	0.2	1.0	1.666667
9	{Coffee,Eggs}	=> {Milk}	0.2	1.0	2.500000
10	{Coffee,Milk}	=> {Diaper}	0.2	1.0	1.250000
11	{Diaper,Milk}	=> {Coffee}	0.2	1.0	2.500000
12	{Beer,Coffee}	=> {Diaper}	0.2	1.0	1.250000
13	{Coffee,Nuts}	=> {Eggs}	0.2	1.0	1.666667
14	{Coffee,Eggs}	=> {Nuts}	0.2	1.0	1.666667
15	{Coffee,Nuts}	=> {Diaper}	0.2	1.0	1.250000
16	{Coffee,Eggs}	=> {Diaper}	0.2	1.0	1.250000
17	{Milk,Nuts}	=> {Eggs}	0.4	1.0	1.666667
18	{Eggs,Milk}	=> {Nuts}	0.4	1.0	1.666667
19	{Eggs,Nuts}	=> {Milk}	0.4	1.0	2.500000
20	{Diaper,Milk}	=> {Nuts}	0.2	1.0	1.666667
21	{Diaper,Milk}	=> {Eggs}	0.2	1.0	1.666667
22	{Beer,Nuts}	=> {Diaper}	0.2	1.0	1.250000
23	{Beer,Eggs}	=> {Diaper}	0.2	1.0	1.250000
24	{Coffee,Milk,Nuts}	=> {Eggs}	0.2	1.0	1.666667

25	{Coffee,Eggs,Milk}	=>	{Nuts}	0.2	1.0	1.666667
26	{Coffee,Eggs,Nuts}	=>	{Milk}	0.2	1.0	2.500000
27	{Coffee,Milk,Nuts}	=>	{Diaper}	0.2	1.0	1.250000
28	{Coffee,Diaper,Milk}	=>	{Nuts}	0.2	1.0	1.666667
29	{Coffee,Diaper,Nuts}	=>	{Milk}	0.2	1.0	2.500000
30	{Diaper,Milk,Nuts}	=>	{Coffee}	0.2	1.0	2.500000
31	{Coffee,Eggs,Milk}	=>	{Diaper}	0.2	1.0	1.250000
32	{Coffee,Diaper,Milk}	=>	{Eggs}	0.2	1.0	1.666667
33	{Coffee,Diaper,Eggs}	=>	{Milk}	0.2	1.0	2.500000
34	{Diaper,Eggs,Milk}	=>	{Coffee}	0.2	1.0	2.500000
35	{Coffee,Eggs,Nuts}	=>	{Diaper}	0.2	1.0	1.250000
36	{Coffee,Diaper,Nuts}	=>	{Eggs}	0.2	1.0	1.666667
37	{Coffee,Diaper,Eggs}	=>	{Nuts}	0.2	1.0	1.666667
38	{Diaper,Eggs,Nuts}	=>	{Coffee}	0.2	1.0	2.500000
39	{Diaper,Milk,Nuts}	=>	{Eggs}	0.2	1.0	1.666667
40	{Diaper,Eggs,Milk}	=>	{Nuts}	0.2	1.0	1.666667
41	{Diaper,Eggs,Nuts}	=>	{Milk}	0.2	1.0	2.500000
42	{Coffee,Eggs,Milk,Nuts}	=>	{Diaper}	0.2	1.0	1.250000
43	{Coffee,Diaper,Milk,Nuts}	=>	{Eggs}	0.2	1.0	1.666667
44	{Coffee,Diaper,Eggs,Milk}	=>	{Nuts}	0.2	1.0	1.666667
45	{Coffee,Diaper,Eggs,Nuts}	=>	{Milk}	0.2	1.0	2.500000
46	{Diaper,Eggs,Milk,Nuts}	=>	{Coffee}	0.2	1.0	2.500000

9.2 Decision Tree

(1) 의사결정나무

- Breiman 등이 의사결정나무 모형은 소개하였고, Loh 등에 의해 많은 발전되었음 [Breiman, 1984],[Loh, 1997]
- 모형의 구축과정을 나무 형태로 표현하여 대상이 되는 집단을 몇 개의 소집단으로 구분하는 분류 및 예측 기법

(2) 실습

```
library(tree)
```

```
iris.tr=tree(Species~., iris)
```

```
iris.tr
```

```
> iris.tr=tree(Species~., iris)
> iris.tr
node), split, n, deviance, yval, (yprob)
  * denotes terminal node

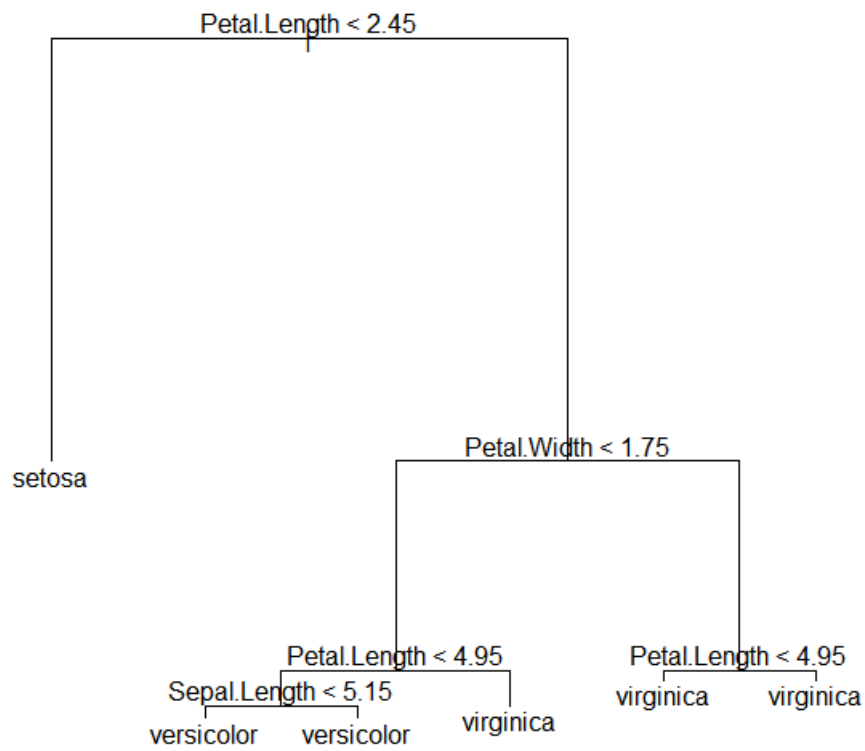
1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
2) Petal.Length < 2.45 50 0.000 setosa ( 1.00000 0.00000 0.00000 ) *
3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
6) Petal.Width < 1.75 54 33.320 versicolor ( 0.00000 0.90741 0.09259 )
12) Petal.Length < 4.95 48 9.721 versicolor ( 0.00000 0.97917 0.02083 )
24) Sepal.Length < 5.15 5 5.004 versicolor ( 0.00000 0.80000 0.20000 ) *
25) Sepal.Length > 5.15 43 0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
13) Petal.Length > 4.95 6 7.638 virginica ( 0.00000 0.33333 0.66667 ) *
7) Petal.Width > 1.75 46 9.635 virginica ( 0.00000 0.02174 0.97826 )
14) Petal.Length < 4.95 6 5.407 virginica ( 0.00000 0.16667 0.83333 ) *
15) Petal.Length > 4.95 40 0.000 virginica ( 0.00000 0.00000 1.00000 ) *
```

```
summary(adult.tr)
```

```
> summary(iris.tr)
```

```
Classification tree:
tree(formula = Species ~ ., data = iris)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width" "Sepal.Length"
Number of terminal nodes: 6
Residual mean deviance: 0.1253 = 18.05 / 144
Misclassification error rate: 0.02667 = 4 / 150
```

```
plot(iris.tr); text(iris.tr)
```



References

- Abrahams, S. et al. (2016) Tensorflow for Machine Intelligence, Bleeding Edge Press.
- Brownley, C. (2017) Foundation for Analytics with Python, O'Reilly.
- Chatterjee, S. et al. (2012) Regression analysis by example, 5th edition, Wiley.
- Efron, B., and Hastie, T. (2016), Computer Age Statistical Inference, Cambridge University Press.
- Goodfellow, I. et al. (2016) Deep learning, MIT Press.
- Han, J. et al. (2012) Data Mining Concepts and Techniques, Morgan Kaufmann.
- McClure, N. (2017) Tensorflow Machine Learning Cookbook, Packt Publishing.
- Murphy, K. P. (2012) Machine Learning: a probabilistic perspective, MIT Press.
- Ramsundar, B. et al. (2018) TensorFlow for Deep Learning, O'Reilly.
- Zaccone, G. (2016) Getting Started with Tensorflow, Packt Publishing.
- 김영우 (2017) 쉽게 배우는 R 데이터분석, 이지스퍼블리싱.
- 나카이 에츠지 (2016) 텐서플로로 시작하는 딥러닝, 제이펍.
- 박응용 (2017) 점프 투 파이썬, 이지스퍼블리싱.
- 이건명 (2018) 인공지능, 생능출판.
- 최병관 외 (2018) Tensorflow 프로그래밍 기초, 청구문화사.