

Lab Seminar: 2022. 07. 26.

# How to deal with Data and Machine Learning

Data Science from Scratch 2<sup>nd</sup> – Chapter 9, 10, 11

**IDEALAB**

Improving  
lives  
through  
learning

**ChanKi Kim**

School of Computer Science/Department of AI Convergence Engineering  
Gyeongsang National University (GNU)

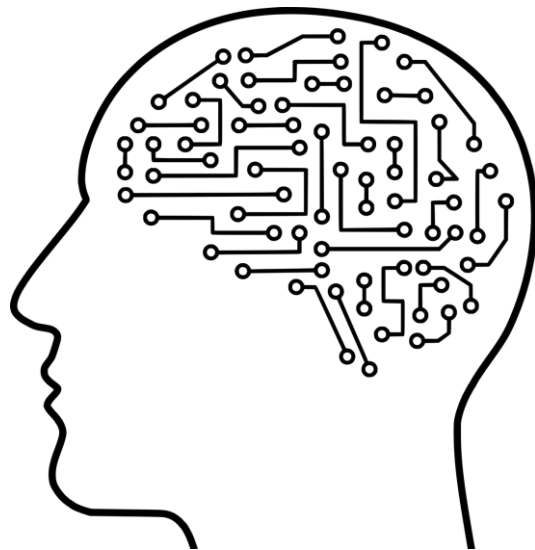
# Contents

2

- Introduction
- Dealing with Data
- About Machine Learning
- Additional Explanation of Last Seminar
- Conclusion & Realization

## ■ Flow of Presentation

- 데이터 다루는 여러 가지 방법 탐색
- 머신 러닝이란 무엇인가? – 각종 개념 이해
- 선형 회귀에서 비용 함수 MSE, RMSE, MAE 적용



# Dealing with Data

4


- Piping using with Stdin & Stdout
  - Pipeline 구축은 데이터의 변환을 순차적으로 적용한 후 학습 가능
  - Pipeline 구축은 표준 입출력인 Stdin(), Stdout()을 이용하여 가능



# Dealing with Data

5

- Piping using with Stdin & Stdout – Sample\_Data

 Sample\_Data - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

Hello I'm CK

This is sample data!

HI HI HI HI HI

CK CK CK

# Dealing with Data

6

- Piping using with Stdin & Stdout – Matching Regex & Count Line Codes

```
import sys, re

regex = sys.argv[1] #sys.argv[0] -> Output = Script Name ex) ".#Sample_Data.txt"

for line in sys.stdin:
    if re.search(regex, line):
        sys.stdout.write(line)

print(regex)
```

```
import sys

cnt = 0

for line in sys.stdin:
    cnt += 1

print(cnt)
```

# Dealing with Data

7

- Piping using with Stdin & Stdout – Piping

```
type "Sample_Data.txt"|python Matching_Regex.py "[A-Z]"|python Count_line.py
```

4

## ■ Piping using with Stdin & Stdout – Word Frequency Codes

```
import sys
from collections import Counter

try:
    num_words = int(sys.argv[1]) #sys.argv[0] -> Output = Script Name ex) ".#Sample_Data.txt"
except:
    print("Usage: Count_Words.py num_words")
    sys.exit(0)

counter = Counter(word.lower()
                  for line in sys.stdin
                  for word in line.strip().split()
                  if word)

for word, cnt in counter.most_common(num_words):
    sys.stdout.write(str(cnt))
    sys.stdout.write("\t")
    sys.stdout.write(word)
    sys.stdout.write("\n")
```



# Dealing with Data

9

- Piping using with Stdin & Stdout – Piping

```
type "Sample_Data.txt" | python Count_Words.py 3
```

```
5      hi  
4      ck  
1      hello
```

# Dealing with Data

10

- Piping using with Stdin & Stdout – Conclusion

```
type "Sample_Data.txt"|python Count_Words.py 3
```

```
type "Sample_Data.txt"|python Matching_Regex.py "[A-Z]"|python Count_line.py
```



## ■ Using the Github API

**ChanKi Kim**  
CKtrace

[Edit profile](#)

1 follower · 1 following

Seoul, Korea  
<https://cktrace.tistory.com>

**Achievements**

Overview Repositories 6 Projects Packages Stars

CKtrace / README.md

- Hi, I'm ChanKi Kim
- I'm interested in AI
- I'm currently learning Deep Learning
- I'm also running a blog (<https://cktrace.tistory.com>) for knowledge sharing!
- I'll do my best to be a better programmer than yesterday!

**Pinned**

[2022\\_IDEALab\\_Summer\\_Seminar](#) Public  
This Repository is for the 2022 Summer Vacation IDEALab Seminar Materials.  
Jupyter Notebook

[Deep\\_Learning\\_from\\_Scratch](#) Public  
Jupyter Notebook

[Programmers](#) Public  
Programmers 코딩 테스트 문제 해결 코드  
Python

[Data\\_Structure](#) Public  
Python

173 contributions in the last year

Contribution settings

Aug Sep Oct Nov Dec Jan Feb Mar Apr May Jun Jul

## ■ Using the Github API

- Github 저장소 정보들이 담긴 API에서 저장소들이 만들어지고 업데이트된 달마다의 빈도수를 알아보고자 한다.

- 이때, 저장소가 만들어진 날짜와 업데이트된 날짜는 유니코드

-> `dateutil.parser.parse()`로 해결

```
"created_at": "2022-07-06T11:13:24Z",  
"updated_at": "2022-07-11T07:30:54Z",
```

```
import json  
import requests  
  
user = "CKtrace"  
endpoint = f"https://api.github.com/users/{user}/repos"  
  
repos = json.loads(requests.get(endpoint).text)  
print(repos)  
  
# {  
#   "created_at": "2022-07-06T11:13:24Z",  
#   "updated_at": "2022-07-11T07:30:54Z",  
#   "name": "IDEALab Summer Seminar Materials",  
#   "description": "IDEALab Summer Seminar Materials",  
#   "fork": false,  
#   "private": false,  
#   "owner": {  
#     "login": "CKtrace",  
#     "id": 123456789,  
#     "node_id": "MDc6MTIzNDU2Nzg5",  
#     "avatar_url": "https://avatars.githubusercontent.com/u/123456789?v=4",  
#     "html_url": "https://github.com/CKtrace",  
#     "followers_url": "https://github.com/CKtrace/followers",  
#     "following_url": "https://github.com/CKtrace/following",  
#     "subscriptions_url": "https://github.com/CKtrace/subscriptions",  
#     "orgs_url": "https://github.com/CKtrace/orgs",  
#     "repos_url": "https://github.com/CKtrace/repos",  
#     "events_url": "https://github.com/CKtrace/events",  
#     "received_events_url": "https://github.com/CKtrace/received_events",  
#     "type": "User",  
#     "site_admin": false  
#   },  
#   "teams_url": "https://github.com/CKtrace/teams",  
#   "collaborators_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/collaborators",  
#   "teams_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/teams",  
#   "issues_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/issues",  
#   "events_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/events",  
#   "pulls_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/pulls",  
#   "branches_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/branches",  
#   "tags_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/tags",  
#   "commits_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/commits",  
#   "statuses_url": "https://github.com/CKtrace/2022-IDEALab-Summer-Seminar/keys/{key_id}/statuses/{sha}",  
#   "language": "Python",  
#   "has_watcher": true,  
#   "watchers_count": 1,  
#   "forks_count": 0,  
#   "stargazers_count": 0,  
#   "subscribers_count": 0,  
#   "permissions": {  
#     "admin": false,  
#     "push": false,  
#     "pull": true,  
#     "triage": false,  
#     "push_restricted": false,  
#     "maintain": false,  
#     "package_deployment": false  
#   },  
#   "default_branch": "main"  
# }
```

## ■ Using the Github API

- 저장소가 만들어진 날과 업데이트 날이 유니코드로 나오는 문제 해결
- 저장소가 만들어진 횟수와 업데이트 된 횟수 월 별로 카운트하여 결론 도출

```
import json
import requests
from collections import Counter
from dateutil.parser import parse

user = "CKtrace"
endpoint = f"https://api.github.com/users/{user}/repos"

repos = json.loads(requests.get(endpoint).text)

mk_dates = [parse(repo["created_at"]) for repo in repos]
mk_month_counts = Counter(date.month for date in mk_dates)
up_dates = [parse(repo["updated_at"]) for repo in repos]
up_month_counts = Counter(date.month for date in up_dates)
print(mk_dates, "\n", mk_month_counts)
print(up_dates, "\n", up_month_counts)
```

## ■ Using the Github API – Conclusion

- 3개의 가장 많은 저장소를 만든 달을 7월 달이며, 1월, 3월, 4월,에는 각각 하나의 저장소를 생성하였다는 결과 파악

```
[datetime.datetime(2022, 7, 6, 11, 13, 24, tzinfo=tzutc()), datetime.datetime(2022, 1, 17, 17, 9, 45, tzinfo=tzutc()), datetime.datetime(2022, 4, 5, 6, 55, 51, tzinfo=tzutc()), datetime.datetime(2022, 3, 18, 9, 4, 45, tzinfo=tzutc()), datetime.datetime(2022, 7, 21, 6, 46, 7, tzinfo=tzutc()), datetime.datetime(2022, 7, 15, 8, 4, 12, tzinfo=tzutc())]  
Counter({7: 3, 1: 1, 4: 1, 3: 1})
```

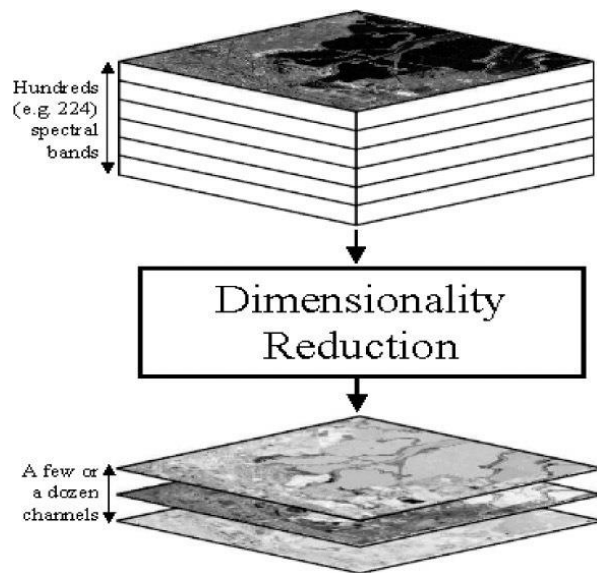
## ■ Using the Github API – Conclusion

- 6번의 가장 많은 업데이트를 한 달은 7월이고, 3개의 가장 많은 저장소를 만든 달 또한 7월이므로 7월 달에 활발한 활동을 하였다는 결론 도출

```
[datetime.datetime(2022, 7, 11, 7, 30, 54, tzinfo=tzutc()), datetime.datetime(2022, 7, 11, 7, 30, 55, tzinfo=tzutc()), datetime.datetime(2022, 7, 11, 7, 30, 57, tzinfo=tzutc()), datetime.datetime(2022, 7, 21, 3, 29, 17, tzinfo=tzutc()), datetime.datetime(2022, 7, 21, 11, 9, 32, tzinfo=tzutc()), datetime.datetime(2022, 7, 15, 9, 14, 13, tzinfo=tzutc())]  
Counter({'7': 6})
```

## ■ Dimension Reduction

- 높은 차원의 데이터로부터 낮은 차원의 데이터로의 변환 방법
  - Ex) Feature Extraction, Feature Selection



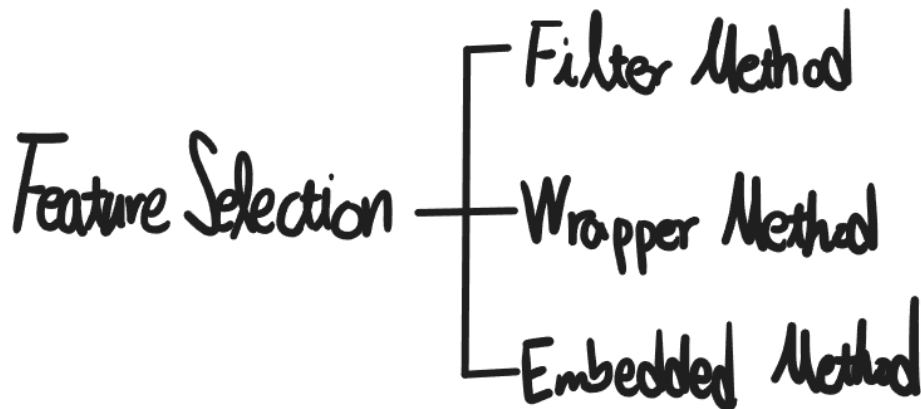


## ■ Purpose of Dimension Reduction

- 특징이 많다고 해서 모든 특징을 사용하여 모델을 만드는 것이 아니라 설득력이 높은 특징들만 정제하여 사용하는 것이 효과적
- 그러나, 선형 모델에서는 고차원 데이터를 넣었을 때 모델 성능이 높아진다는 점 또한 존재

- Feature Selection

- “Feature Selection을 한다. = Feature Subset을 생성한다.”
  - Filter Method, Wrapper Method, Embedded Method



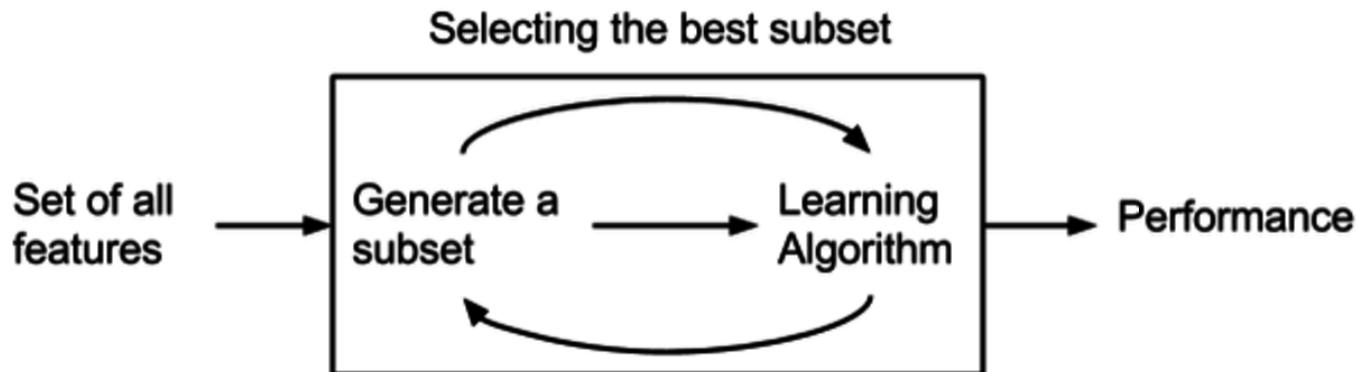
- **Filter Method**

- Feature간의 높은 상관 계수를 가지는 Feature를 선택하는 방법
  - 이때 , 높은 상관 계수란 큰 영향력을 의미



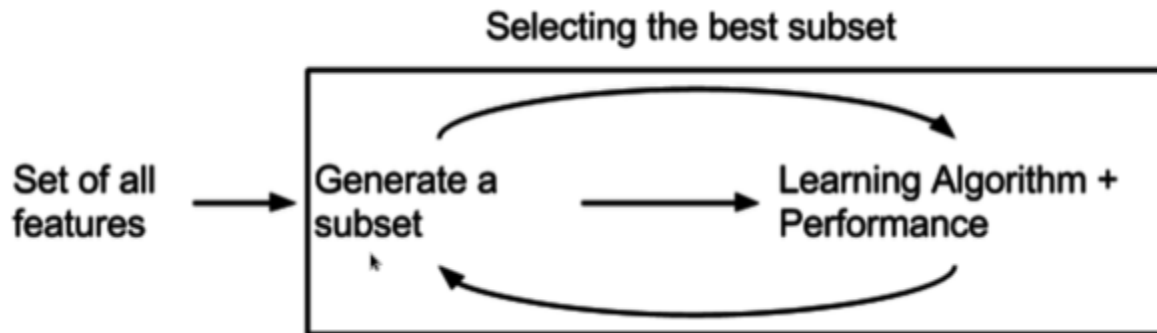
## ■ Wrapper Method

- 예측 정확도 측면에서 가장 좋은 성능을 보이는 Feature Subset 추출하는 방법
  - 단, 학습이 여러 번 진행하기 때문에 시간과 비용이 높게 든다는 단점이 존재하지만, 최종적으로 최적의 Feature Subset이 도출되기 때문에 바람직함



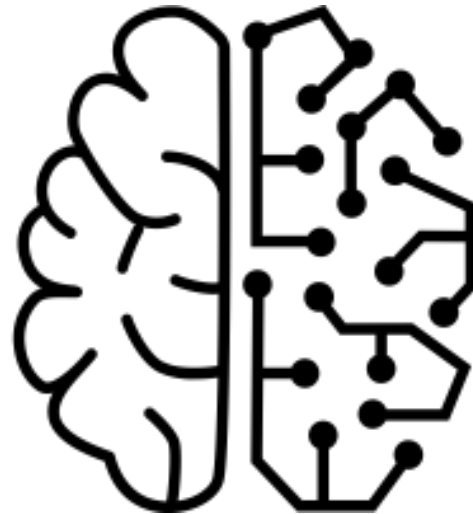
## ■ Embedded Method

- Wrapper Method와 달리 각각의 Feature를 직접 학습하고 모델의 정확도에 기여하는 Feature 선발



- What is Machine Learning?

- 주어진 데이터에서 적용한 알고리즘(Ex. Regression)을 통해 패턴 탐색
- 탐색한 패턴을 통해 의사 결정과 예측에 결정적 도움



- **Supervised Learning**

- 학습에 사용될 데이터와 각 데이터의 답이 포함되어 있는 학습 방법
  - Ex) Classification, Regression

- **Unsupervised Learning**

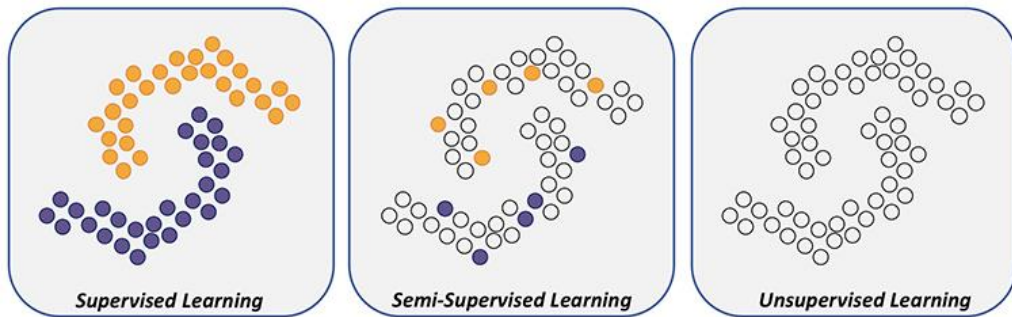
- 학습에 사용될 데이터만 포함되어 있는 학습 방법
  - Ex) Clustering, Dimensionality reduction

# About Machine Learning

24

- **Semi-supervised Learning**

- 학습에 사용될 데이터와 데이터 중 일부의 답만 포함되어 있는 학습 방법
  - Ex) Google Photo Hosting





- Classification vs Regression

- Classification

- 데이터에 독립변수와 종속변수, 두 개의 변수가 있고 종속변수가 이름일 때 '분류'
- 주어진 데이터를 클래스 별로 분류

- Regression

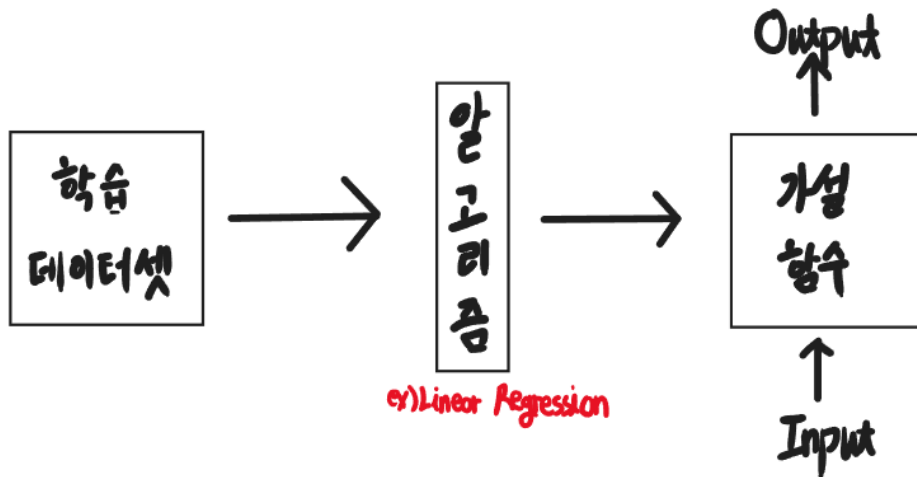
- 데이터에 독립변수와 종속변수, 두 개의 변수가 있고 종속변수가 숫자일 때 '회귀'
- 주어진 두 변수의 관계를 분석

# About Machine Learning

26

## ■ Linear Regression

- 주어진 두 변수의 관계를 나타내는 최적의 직선을 도출해내는 알고리즘



- Hypothesis Function

- 두 변수의 관계를 나타내는 함수

- 가설 함수에서  $a$ 는 기울기이자 가중치를 의미하고,  $b$ 는 절편이자 편향을 의미

$$h(x) = ax + b$$

- Cost Function(= Loss Function)

- 예측하는 값과 실제의 값 사이의 오차를 최소화하기 위한 함수
  - MSE, RMSE, MAE

- Mean Squared Error

- 예측하는 값과 실제의 값 차이를 제공한 후 평균화

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$$

- Root Mean Square Error

- 예측하는 값과 실제의 값 차이를 제곱한 값에 루트

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}$$

- Mean Absolute Error

- 예측하는 값과 실제의 값 차이의 절댓값을 평균화

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}|$$

## ■ Experiment Purpose

- Kaggle의 데이터 셋에서 도미의 무게와 길이 변수의 관계를 선형 회귀를 통한 도출
- 손실 함수인 MSE, RMSE, MAE를 각각 적용했을 때, 나타나는 추이와 이를 통한 각 손실 함수의 특징 도출

### Fish market

Database of common fish species for fish market



[Data](#) [Code \(198\)](#) [Discussion \(7\)](#) [Metadata](#)

#### About Dataset

##### Content

This dataset is a record of 7 common different fish species in fish market sales. With this dataset, a predictive model can be performed using machine friendly data and estimate the weight of fish can be predicted.

##### Acknowledgements

##### Usability ⓘ

10.00

##### License

[GPL 2](#)

##### Expected update frequency

Never

# Additional Explanation of Last Seminar

31

## ■ Bream Dataset

```
bream_length = np.array([25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0,  
                        31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0,  
                        35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0])  
bream_weight = np.array([242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0,  
                        500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,  
                        700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0])
```

## ■ Linear Regression with Gradient Descent

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

bream_length = np.array([25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0,
                          31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0,
                          35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0])
bream_weight = np.array([242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0,
                          500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,
                          700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0])

# y = ax + b
a = 1
b = 0

lr = 0.0005

epochs = 1000000

for i in range(epochs):

    y_pred = a * bream_length + b
    error = bream_weight - y_pred
    """
    MSE, RMSE, MAE 미분 공식 대입
    """
    a = a - lr * a_diff
    b = b - lr * b_diff

    if i % 1000 == 0:
        print("epoch = %.f, 기울기 = %.04f, 절편 = %.04f, 오차=%.04f" % (i, a, b, error.mean()))
        print(a_diff)
        print(b_diff)

plt.title("MSE")
plt.scatter(bream_length, bream_weight)
plt.plot([min(bream_length), max(bream_length)], [min(y_pred), max(y_pred)], 'r')
plt.xlabel('Length')
plt.ylabel('Weight')
plt.show()
```

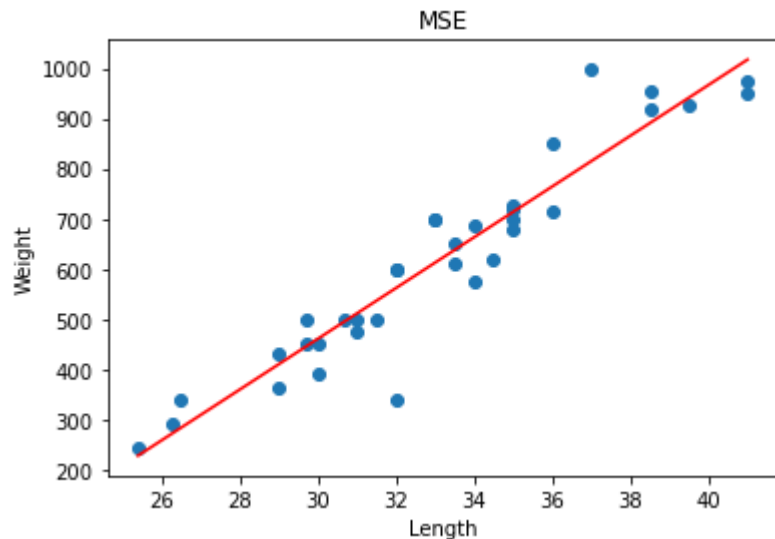


# Additional Explanation of Last Seminar

33

## ■ Linear Regression with Gradient Descent – MSE

```
a_diff = -(2/len(bream_length)) * sum((bream_length*(error)))  
b_diff = -(2/len(bream_weight)) * sum(error)
```



$lr = 0.0005$

## ■ Linear Regression with Gradient Descent – MSE

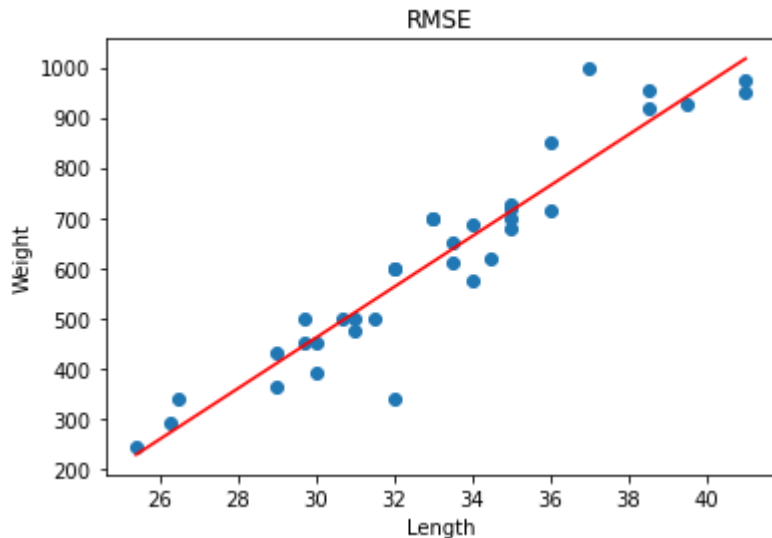
```
epoch = 939000, 기울기 = 50.6047, 절편 = -1057.6202, 오차=-0.0001  
-2.9800182736445484e-06  
0.00010000094342623404  
epoch = 940000, 기울기 = 50.6047, 절편 = -1057.6203, 오차=-0.0000  
-2.940448757726699e-06  
9.867299008224758e-05  
epoch = 941000, 기울기 = 50.6047, 절편 = -1057.6203, 오차=-0.0000  
-2.9014003043162767e-06  
9.73626712688461e-05  
epoch = 942000, 기울기 = 50.6047, 절편 = -1057.6203, 오차=-0.0000  
-2.862861843563483e-06  
9.606975298603564e-05  
epoch = 943000, 기울기 = 50.6047, 절편 = -1057.6204, 오차=-0.0000  
-2.8248481872391755e-06  
9.479400352055173e-05  
epoch = 944000, 기울기 = 50.6047, 절편 = -1057.6204, 오차=-0.0000  
-2.7873403658824306e-06  
9.353519519988497e-05
```

# Additional Explanation of Last Seminar

35

## ■ Linear Regression with Gradient Descent – RMSE

```
a_diff = -((len(bream_length)*sum(error**2))**(-1/2)) * sum((bream_length*(error)))  
b_diff = -((len(bream_length)*sum(error**2))**(-1/2)) * sum(error)
```



$$lr = 0.07$$

## ■ Linear Regression with Gradient Descent – RMSE

epoch = 939000,	기울기 = 50.6047,	절편 = -1057.6201,	에러=-0.0001
epoch = 940000,	기울기 = 50.6047,	절편 = -1057.6201,	에러=-0.0001
epoch = 941000,	기울기 = 50.6047,	절편 = -1057.6202,	에러=-0.0001
epoch = 942000,	기울기 = 50.6047,	절편 = -1057.6202,	에러=-0.0000
epoch = 943000,	기울기 = 50.6047,	절편 = -1057.6203,	에러=-0.0000
epoch = 944000,	기울기 = 50.6047,	절편 = -1057.6203,	에러=-0.0000
epoch = 945000,	기울기 = 50.6047,	절편 = -1057.6204,	에러=-0.0000
epoch = 946000,	기울기 = 50.6047,	절편 = -1057.6204,	에러=-0.0000
epoch = 947000,	기울기 = 50.6047,	절편 = -1057.6205,	에러=-0.0000
epoch = 948000,	기울기 = 50.6047,	절편 = -1057.6205,	에러=-0.0000
epoch = 949000,	기울기 = 50.6047,	절편 = -1057.6206,	에러=-0.0000
epoch = 950000,	기울기 = 50.6047,	절편 = -1057.6206,	에러=-0.0000
epoch = 951000,	기울기 = 50.6047,	절편 = -1057.6207,	에러=-0.0000
epoch = 952000,	기울기 = 50.6047,	절편 = -1057.6207,	에러=-0.0000
epoch = 953000,	기울기 = 50.6047,	절편 = -1057.6208,	에러=-0.0000
epoch = 954000,	기울기 = 50.6047,	절편 = -1057.6208,	에러=-0.0000
epoch = 955000,	기울기 = 50.6047,	절편 = -1057.6209,	에러=-0.0000
epoch = 956000,	기울기 = 50.6047,	절편 = -1057.6209,	에러=-0.0000
epoch = 957000,	기울기 = 50.6047,	절편 = -1057.6209,	에러=-0.0000
epoch = 958000,	기울기 = 50.6047,	절편 = -1057.6210,	에러=-0.0000

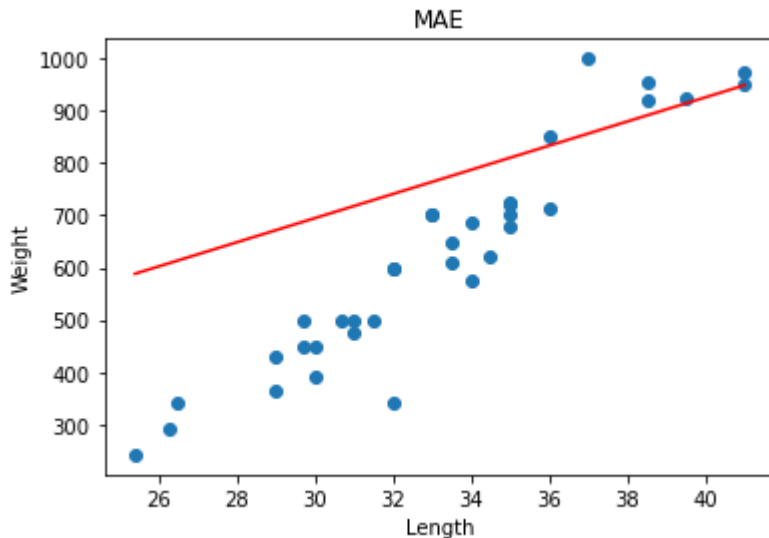
# Additional Explanation of Last Seminar

37

## ■ Linear Regression with Gradient Descent – MAE

`lr = 0.0001`

```
for z in range(len(error)):
    if error[z] >= 1 :
        a_diff = -(1/len(bream_weight))*sum(bream_length)
        b_diff = - 1
    else:
        a_diff = (1/len(bream_weight))*sum(bream_length)
        b_diff = 1
```



## ■ Linear Regression with Gradient Descent – MAE

```
epoch = 6000, 기울기 = 20.8685, 절편 or 편차 = 0.6001, 에러= -73.5865  
-33.10857142857142  
-1  
epoch = 7000, 기울기 = 23.1331, 절편 or 편차 = 0.6685, 에러= -148.6334  
-33.10857142857142  
-1  
epoch = 8000, 기울기 = 23.1331, 절편 or 편차 = 0.6685, 에러= -148.6334  
-33.10857142857142  
-1  
epoch = 9000, 기울기 = 23.1331, 절편 or 편차 = 0.6685, 에러= -148.6334  
-33.10857142857142  
-1  
epoch = 10000, 기울기 = 23.1331, 절편 or 편차 = 0.6685, 에러= -148.6334  
-33.10857142857142  
-1  
epoch = 11000, 기울기 = 23.1331, 절편 or 편차 = 0.6685, 에러= -148.6334  
-33.10857142857142  
-1
```

## ■ Experiment Conclusion

- MAE를 사용하면 RMSE와 MSE와 달리 적은 Epoch 값으로 가중치와 편향이 수렴한다는 장점을 갖고 있지만 회귀에서는 올바른 값을 내지 못함.
- RMSE와 MSE는 식 또한 루트를 제외하고는 동일하지만, MSE의 제곱을 하는 과정 때문에 이상치에 민감한 단점을 갖지만 RMSE에서의 루트로 이를 해소
- 하지만 RMSE와 MSE의 식에서 제공하는 것은 동일하기 때문에, 1 미만의 오차는 더욱 더 작아지고, 그 이상의 에러는 더 커진다는 단점 보유
- 또한, RMSE와 MSE의 루트의 여부로 인해 RMSE의 학습률은 0.07, MSE의 학습률은 0.0005
- MAE는 예측하는 학습 과정에서 효과적 ex) 집 값 예측

## ■ TQDM Application

- TQDM이란, 작업 진행률을 시각화 해주는 라이브러리
- 상당한 시간이 소요되는 학습에서 진행률을 알기 위해 사용
- -> 앞서 알아본 경사 하강법을 이용한 선형 회귀는 상대적으로 빠르게 학습
- -> TQDM을 쓰기엔 적합 X
- -> 그러나, time 라이브러리의 sleep을 적용하여 학습 진행 속도 저하한 후 적용



# Additional Explanation of Last Seminar

41

## ■ TQDM Application

```
for i in range(1001):
    y_pred = a * bearm_length + b
    error = bearm_weight - y_pred

    a_diff = -(2/len(bearm_length)) * sum((bearm_length*(error)))
    b_diff = -(2/len(bearm_weight)) * sum(error)

    a = a - lr * a_diff
    b = b - lr * b_diff
    for z in tqdm.tqdm(range(epochs//1000), desc = "Process", mininterval=0.01):
        y_pred = a * bearm_length + b
        error = bearm_weight - y_pred

        #MSE
        a_diff = -(2/len(bearm_length)) * sum((bearm_length*(error)))
        b_diff = -(2/len(bearm_weight)) * sum(error)

        a = a - lr * a_diff
        b = b - lr * b_diff
        time.sleep(0.0001)

    print("epoch = %.f, 기울기 = %.04f, 절편 = %.04f, 오차 = %.04f" % ((i*epochs//1000)+1000, a, b, error.mean()))
    print(a_diff)
    print(b_diff)
```

## ■ TQDM Application

- 개선 사항 : tqdm 라이브러리의 익숙하지 않아, 상당히 비효율적인 코드로 작성
- 개선할 부분을 개선하여 추후에 상대적으로 시간이 많이 걸리는 학습들의 진행 상황을 tqdm 라이브러리를 이용해 파악



- 데이터를 직접 다루고 학습에 용이하게 만드는 과정을 학습하며, 앞으로 학습 모델을 돌릴 때 더욱 더 좋은 결과가 나올 수 있게 하는 역량을 갖게 됨
- 함수들을 직접 작성해보고, 코드로 구현하는 과정에서 개념을 학습하는 데에 있어 이해가 어려웠던 부분들을 효과적으로 이해 가능
- 추후에도 여러 머신 러닝 알고리즘 코드를 짜보고 학습 시켜 보며 이해도를 높여 나갈 예정



경상국립대학교

Gyeongsang National University

Improving lives through learning

**IDEALAB**