GNU

Lab Seminar: 2022. 08. 02.

Machine Learning Algorithm : KNN & Naive Bayes

Data Science from Scratch 2nd – Chapter 12, 13

IDEALAB
Improving lives through learning

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

GNU 경상국립대학교 Gyeongsang National University

Contents

- Additional Explanation of Last Seminar
- KNN
- Naive Bayes Classification
- Conclusion & Realization

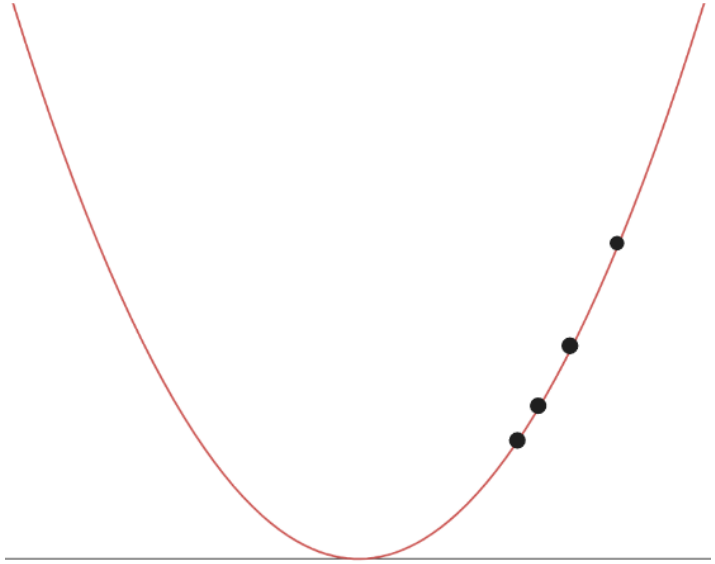
Additional Explanation of Last Seminar

- Further Explanation about Loss Function
  - 올바르게 갱신되는 가중치와 편향 값에 따라 감소
  - 여러 Loss Function의 형태에 따라 가중치와 편향이 갱신되는 형태 차이 존재
    - MSE vs RMSE vs MAE

$$W = W - \alpha \frac{\partial}{\partial W} J(W, b)$$

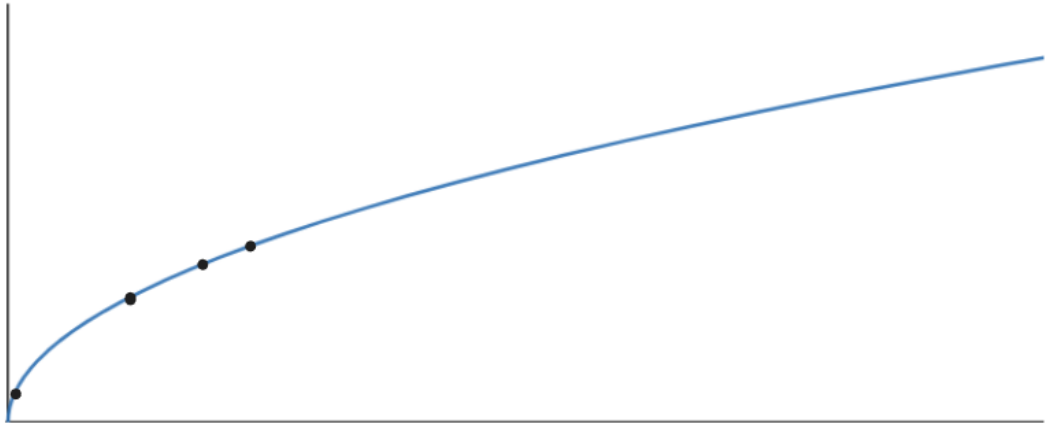
Additional Explanation of Last Seminar

- **Further Explanation about MSE**
  - 가중치가 갱신될 때마다 그 지점의 기울기가 완만해지기 때문에 가중치의 변화는 초반부보다는 더 세밀하다는 것을 그래프를 통해 확인 가능



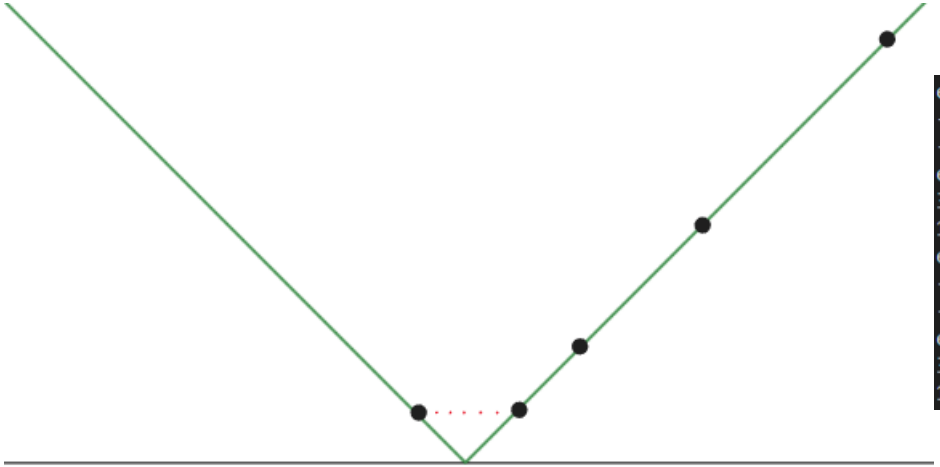
Additional Explanation of Last Seminar

- Further Explanation about RMSE
  - MSE와 마찬가지로 가중치가 갱신될 때마다 그 지점의 기울기는 완만, 그에 따른 가중치 변화는 초반보다 세밀



Additional Explanation of Last Seminar

- Further Explanation about MAE
  - MAE의 그래프를 통해 알 수 있듯이 어떤 지점에서 미분을 해도 기울기의 절댓값이 같게 나오고 Learning Rate마저 고정 -> RMSE와 MSE보다 빠르게 수렴
    - 그러나, 기울기가 0으로 수렴하는 지점 존재 X

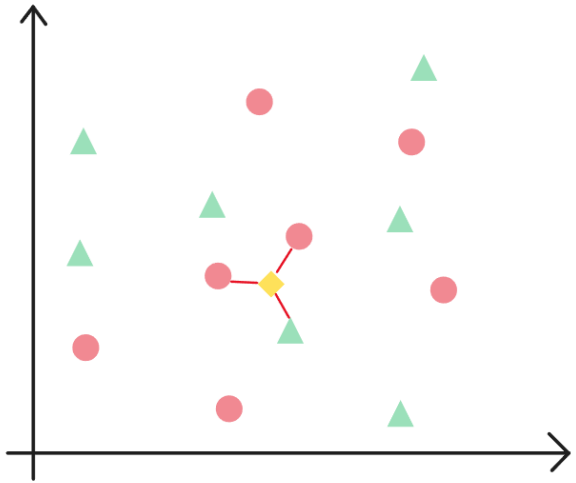


```
epoch = 221408, 기울기 = 23.1331,  절편 or 편차 = 0.6685, 에러= -148.6334
-33.10857142857142
-1
epoch = 221409, 기울기 = 23.1298,  절편 or 편차 = 0.6684, 에러= -148.7432
33.10857142857142
1
epoch = 221410, 기울기 = 23.1331,  절편 or 편차 = 0.6685, 에러= -148.6334
-33.10857142857142
-1
epoch = 221411, 기울기 = 23.1298,  절편 or 편차 = 0.6684, 에러= -148.7432
33.10857142857142
1
```

KNN

- What is KNN Algorithm?
  - 레이블 된 데이터들 간의 거리 개념을 이용한 알고리즘
  - 유사한 특성을 가진 데이터는 유사한 범주에 속한다는 가정
  - 가장 간단한 머신 러닝 알고리즘
    - KNN : K-Nearest Neighbor

- Learning Mechanism of KNN
  - 분류를 하기 위한 K, 즉 분류를 위한 이웃의 수 지정
  - 어떤 클래스인지 알고자 하는 원소에 가장 근접한 K개 중 가장 많이 속한 클래스로 분류
    - K = 3 일 때, 아래의 마름모 원소는 빨간색 원 클래스로 예측



KNN

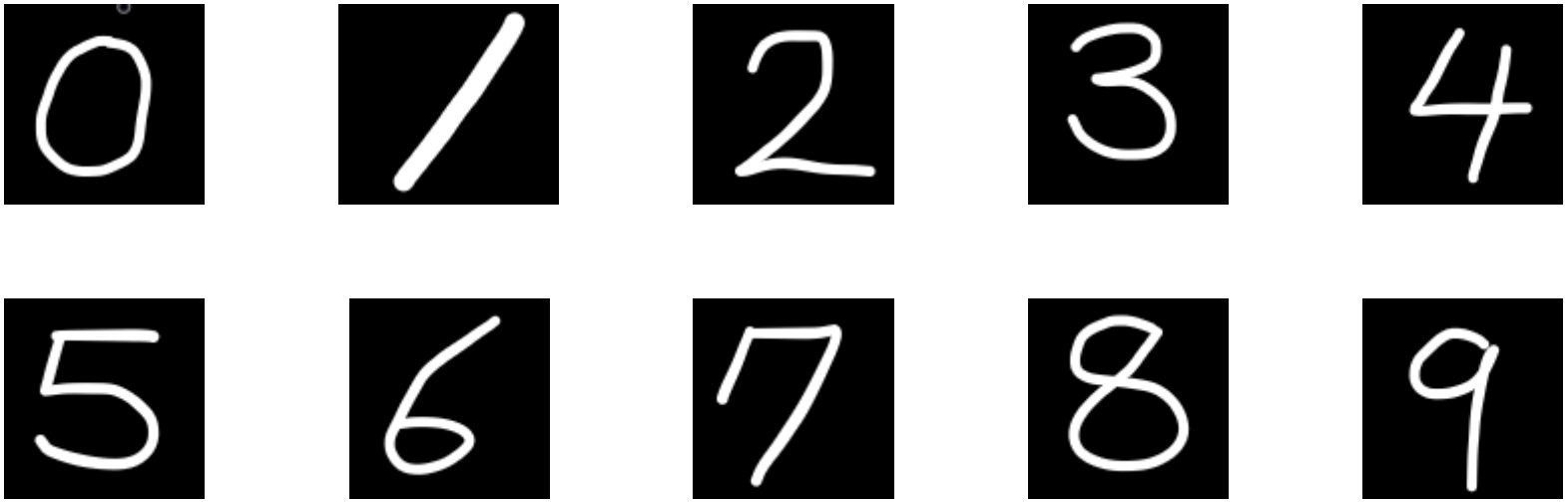
- **MNIST Dataset**
  - 손 글씨로 적은 숫자들의 데이터 집합
  - Neural Network, KNN 알고리즘을 통한 분류 가능

KNN

- Classification MNIST Train Set using KNN
  - Appearance of Data
    - 0~9 , Row : 100, Column : 50

KNN

- Classification MNIST Test Set using KNN
  - Appearance of Data



KNN

- **Classification MNIST Dataset using KNN**
  - **Code Review**

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import glob

img = cv2.imread('mnist_knn.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cells = [np.hsplit(row, 100) for row in np.vsplit(gray, 50)]
x = np.array(cells)
print(x.shape)

train = x[:, :].reshape(-1, 400).astype(np.float32)
print(train.shape)

k = np.arange(10)
train_labels = np.repeat(k, 500)[:, np.newaxis]
print(train_labels.shape)

np.savez("train_sample.npz", train=train, train_labels=train_labels)
c = np.load("train_sample.npz")
print(c['train'][0])
```

KNN

- **Classification MNIST Dataset using KNN**
  - **Code Review**

```
(50, 100, 20, 20)
(5000, 400)
(5000, 1)
```

```
[[  0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   9.  33.   9.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
   41. 177. 249. 178.  29.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.  33. 198. 255. 240. 255. 107.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   1.  70. 199. 255. 255.
  197. 154. 253.  98.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.  45. 238. 255. 205. 224. 222.  83. 224. 128.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.  25. 202. 255. 193.  40.  99.  54.   0.
  190. 197.  16.   0.   0.   0.   0.   0.   0.   0.   0.  20. 163. 246.
  152.  72.   0.   0.   0.   0. 184. 252.  74.   0.   0.   0.   0.   0.
    0.   0.   0.  97. 255. 118.   0.   1.   0.   0.   0.   0. 184. 255.
   82.   0.   0.   0.   0.   0.   0.   0.  20. 218. 216.  17.   0.   0.
    0.   0.   0.   0. 183. 255.  78.   0.   0.   0.   0.   0.   0.   0.
   67. 255. 138.   0.   0.   0.   0.   0.   0.  24. 215. 188.  22.   0.
    0.   0.   0.   0.   0.   0.  69. 247.  87.   0.   0.   0.   0.   0.
   31. 185. 212.  54.   0.   0.   0.   0.   0.   0.   0.   0.  71. 237.
   68.   0.   0.   0.   0.  94. 211. 177.  17.   0.   0.   0.   0.   0.
    0.   0.   0.   0.  69. 255. 170.  39.  39. 115. 183. 238. 185.  37.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  57. 245. 255. 229.
  233. 255. 216. 117.  13.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   7. 114. 230. 255. 234. 137.  38.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  26.  38.  27.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
    0.   0.   0.   0.   0.   0.   0.   0.]
```

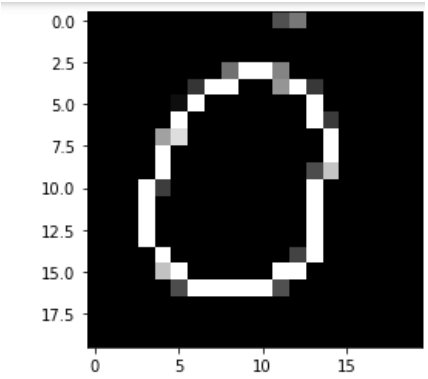
- **Classification MNIST Dataset using KNN**
  - **Code Review**

```
import cv2
import numpy as np


FILE_NAME = 'train_sample.npz'

def load_train_data(file_name):
  with np.load(file_name) as data:
    train = data['train']
    train_labels = data['train_labels']
  return train, train_labels

def flatten_process(image):
  img = cv2.imread(image)
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  gray_resize = cv2.resize(gray, (20, 20))
  plt.imshow(cv2.cvtColor(gray_resize, cv2.COLOR_GRAY2RGB))
  plt.show()
  return gray_resize.reshape(-1, 400).astype(np.float32)
```

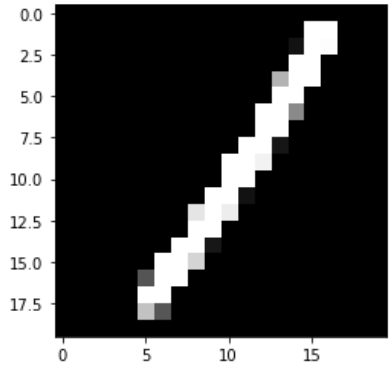
```
def check(test, train, train_labels):
  knn = cv2.ml.KNearest_create()
  knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
  ret, result, neighbours, dist = knn.findNearest(test, k=5)
  return result

train, train_labels = load_train_data(FILE_NAME)

for file_name in glob.glob('./test_*.png'):
  test = flatten_process(file_name)
  result = check(test, train, train_labels)
  print(result)
```

KNN

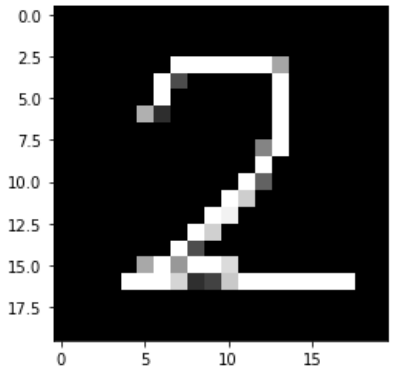
- **Classification MNIST Dataset using KNN**
  - **Code Review**



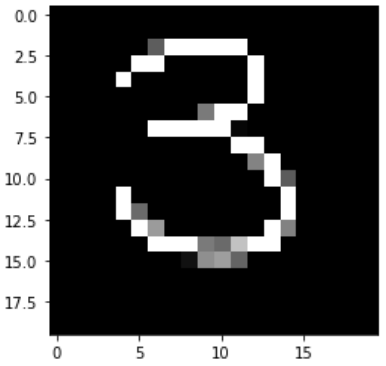
[[0.]]



[[1.]]



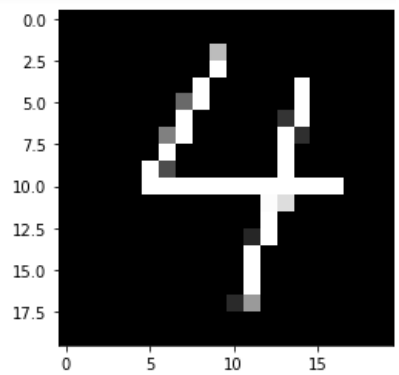
[[2.]]



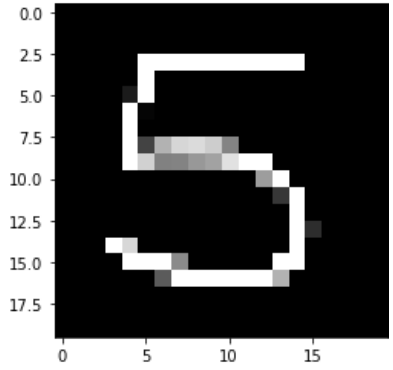
[[3.]]

KNN

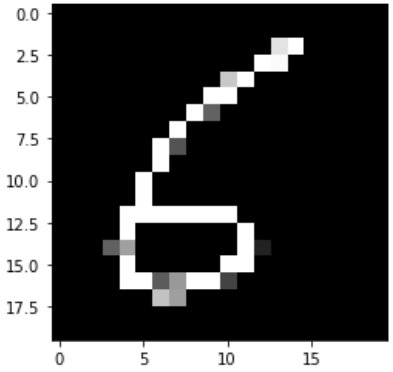
- **Classification MNIST Dataset using KNN**
  - **Code Review**



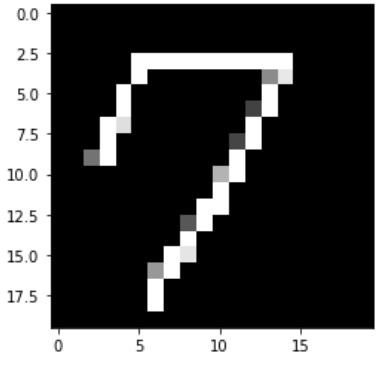
[[4.]]



[[5.]]



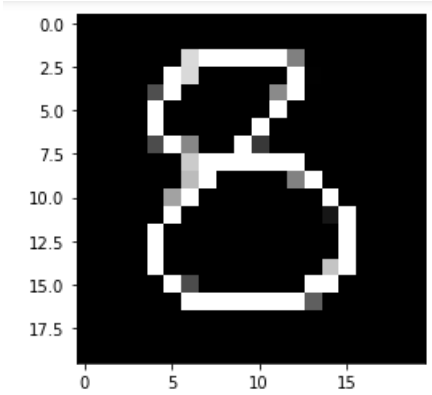
[[5.]]



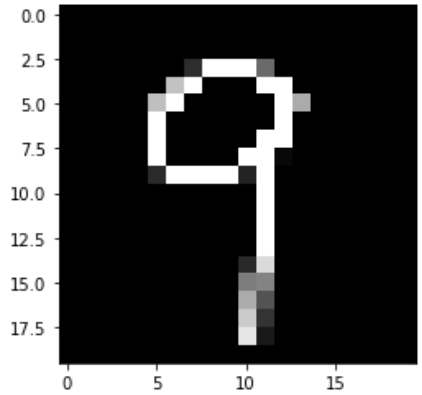
[[1.]]

KNN

- **Classification MNIST Dataset using KNN**
  - **Code Review**



[[5.]]



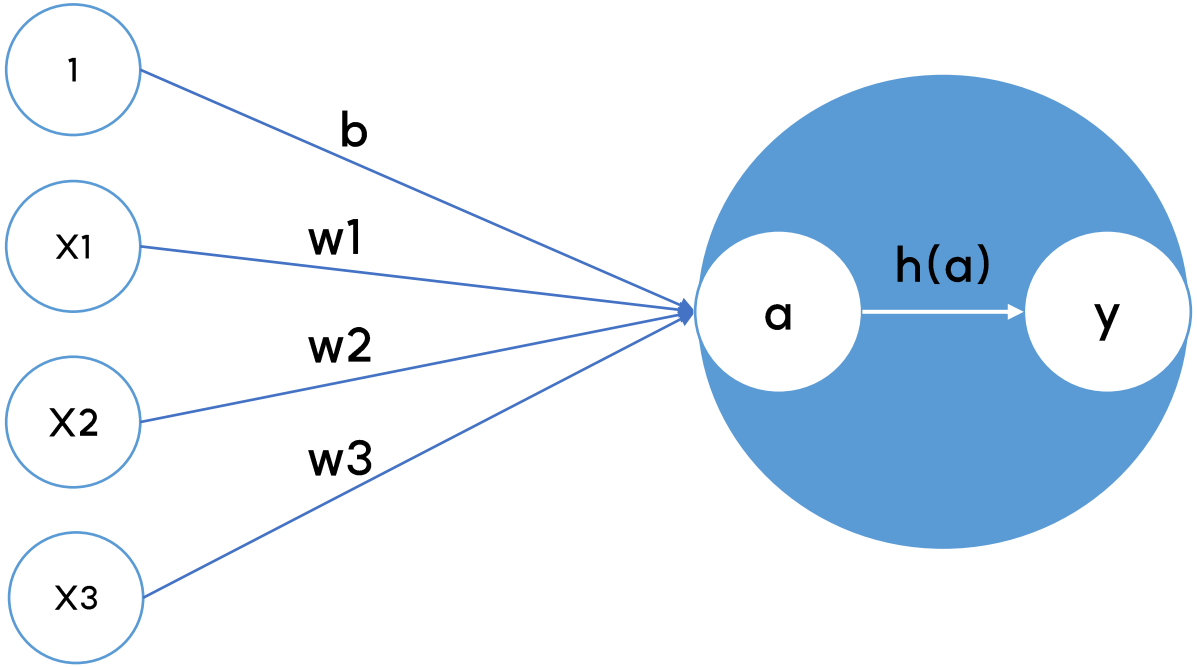
[[4.]]

KNN

- Conclusion of Classification MNIST Dataset using KNN
  - 0~9까지의 손 글씨 Test 데이터 중 0~5까지는 분류 성공, 6~9까지 분류 실패
  - 즉, 정확도는 약 60%로 추정 가능

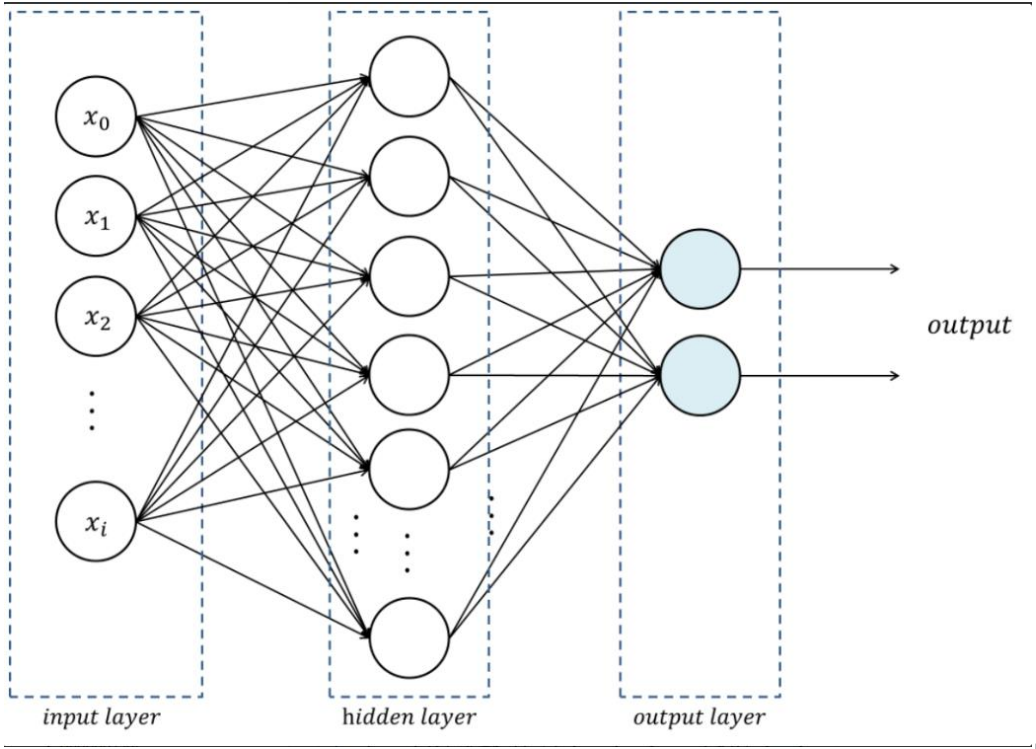
KNN

- Neural Network



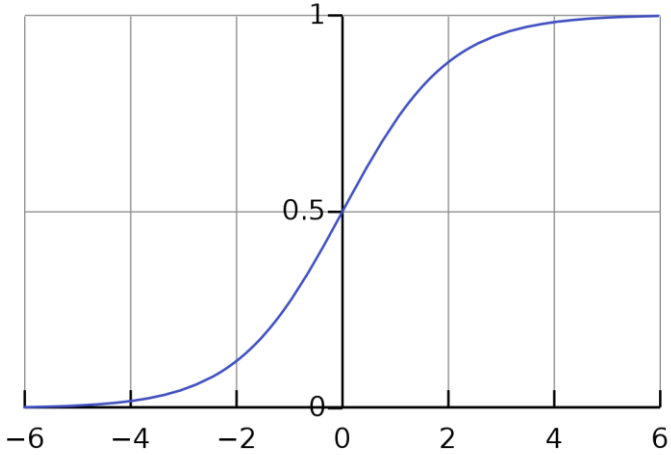
KNN

- Neural Network



KNN

- Hypothesis Function
  - 입력 신호의 총합을 출력 신호로 변환하는 함수
  - 모델의 복잡도를 올리기 위해 필수적
    - Ex) Step Function, Sigmoid Function



$$
\begin{aligned}
h(x) &= qx \\
y(x) &= h(h(h(x))) = q \times q \times q \times x \\
&= cx
\end{aligned}
$$

- **Classification MNIST Dataset using Neural Network**
  - **이미 학습된 가중치와 편향을 이용한 성능 테스트**

```python
import numpy as np
from mnist_helper import load_mnist
import pickle

def get_data():
    (x_train, t_train), (x_test, t_test) = ₩
    load_mnist(flatten=True, normalize=False)
    return x_test, t_test

def init_network():
    with open("sample_weight.pkl", 'rb') as f:
        network = pickle.load(f)
    return network

# sigmoid overflow encountered ex) x = -200000000
def sigmoid(x):
    return 1/(1+np.exp(-x))

# solution of sigmoid overflow
# def sigmoid(x):
#     if x < 0 :
#         return np.exp(x)/(1+np.exp(x))
#     else:
#         return 1/(1+np.exp(-x))
```

```python
def softmax(a):
    c = np.max(a)
    exp_a = np.exp(a - c)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y

def predict(network, x):
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']


    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3) + b3
    y = softmax(a3)

    return y
```

- **Classification MNIST Dataset using Neural Network**
  - **MNIST 전체 데이터셋에서 6만 개의 train dataset, 1만 개의 test dataset을 이용**

```
x, t = get_data()

network = init_network()

batch_size = 100
accuracy_cnt = 0

# Batch Size == 100
for i in range(0, len(x), batch_size):
    x_batch = x[i:i+batch_size]
    y_batch = predict(network, x_batch)
    p = np.argmax(y_batch, axis=1)
    accuracy_cnt += np.sum(p==t[i:i+batch_size])

print("Accuracy:" + str(float(accuracy_cnt)/len(x)))
```

KNN

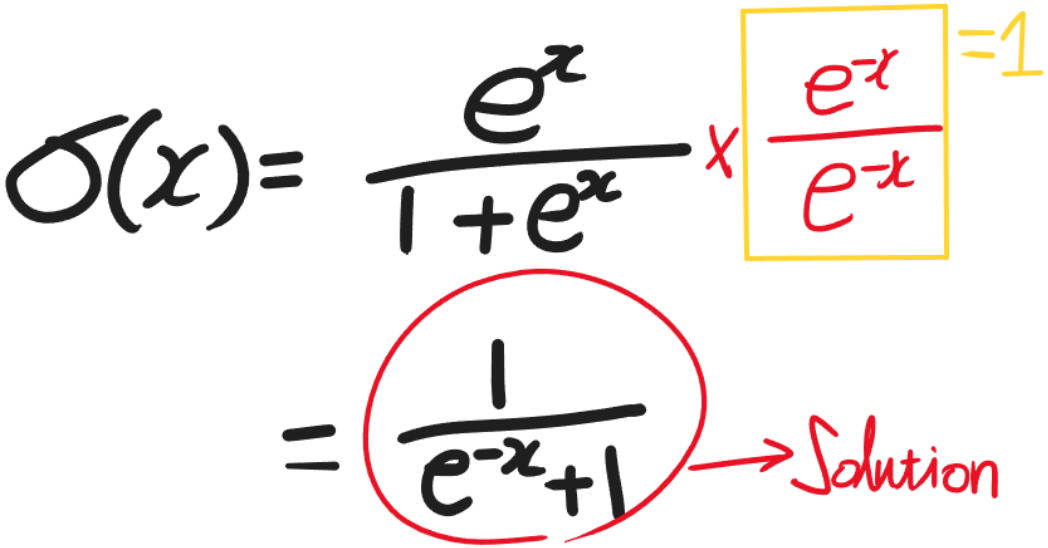
- **Conclusion of Classification MNIST Dataset using KNN**
  - 정확도는 약 92%가 나왔으며, 10개 중 9개 꼴로 정확하게 분류

```
Accuracy:0.9207
```

```
C:₩Users₩brand₩AppData₩Local₩Temp₩ipykernel_8532₩3323820423.py:18: RuntimeWarning: o
verflow encountered in exp
  return 1/(1+np.exp(-x))
```

KNN

- Runtime Warning : Overflow encountered in exp
  - 만약, x = −200000000인 경우, sigmoid function에서 overflow 발생



KNN

- **Runtime Warning : Overflow encountered in exp**
  - **Implementation**

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

```
def sigmoid(x):
    if x < 0 :
        return np.exp(x)/(1+np.exp(x))
    else:
        return 1/(1+np.exp(-x))
```

KNN

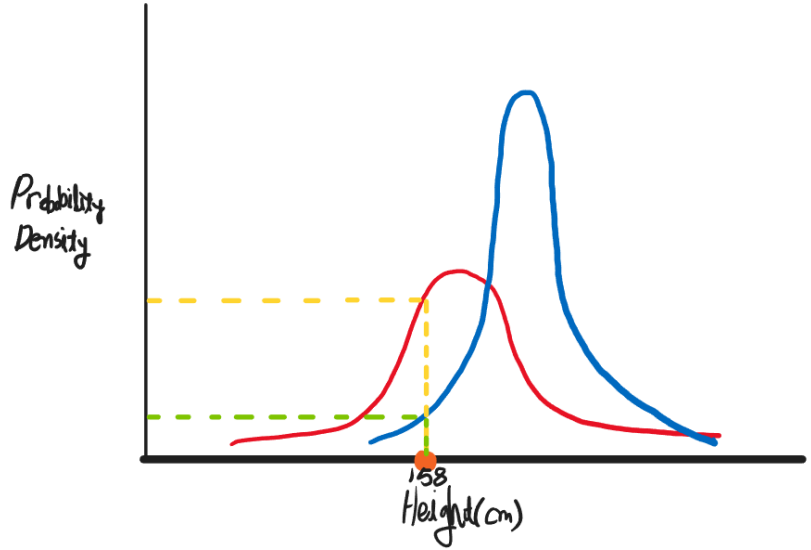
- Improvement
  - 향후 KNN과 Neural Network을 이용한 MNIST Dataset 분류에서 동일한 train set과 test set을 사용하여 동일 기준에서의 Accuracy를 측정할 수 있는 능력 함양

Naive Bayes Classification

- Prior Probability
  - Data Sample에 대한 Feature 없이도 배경 지식만을 가지고 Data Sample 판별 가능
    - Ex) “전체 인구 중 남자나 여자인 확률은 각각 대략 50%로 확률일 것이다”
      - P(성별=남자) or P(성별=여자)

Naive Bayes Classification

- Likelihood
  - 사전 지식 이외의 어떤 지식(정보)이 추가되는 경우
    - Ex) 사람의 키(빨간색 분포 : 여성, 파란색 분포 : 남성)가 아래와 같은 분포를 따를 때, 158cm의 사람이 여성에 속할 가능도가 남성에 속할 가능도보다 높다는 결론 도출 가능

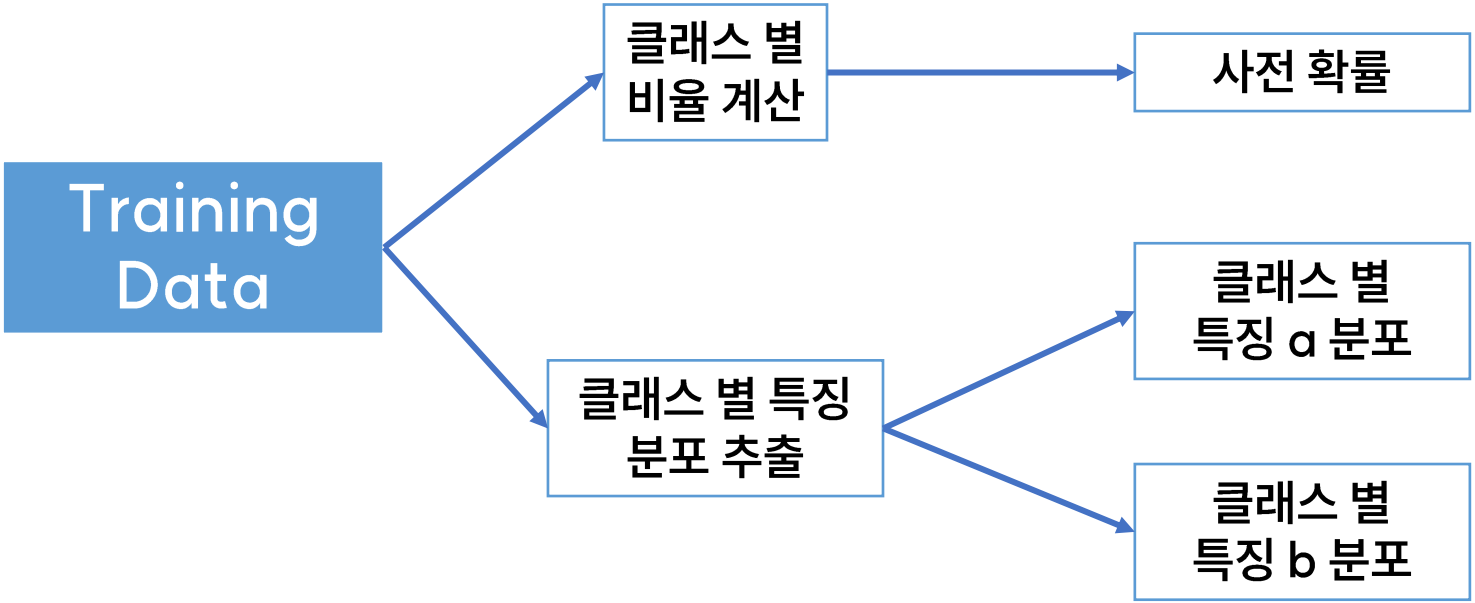


Naive Bayes Classification

- Judgement
  - 판단 근거 = 사전 확률 x 가능도
  - Process = 사전 확률(Base)에 얻게 되는 추가 지식들을 쌓아 나가며 갱신
    - Ex) P(성별 = 여성) x P(키 158 | 성별 = 여성)
    - Ex) P(성별 = 남성) x P(키 158 | 성별 = 남성)

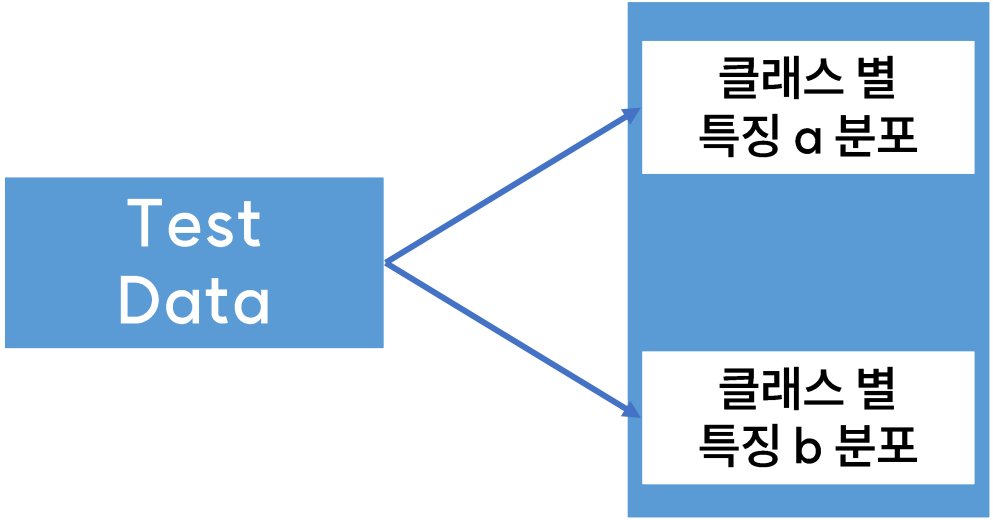
Naive Bayes Classification

- Mechanism of Naive Bayes Classification



Naive Bayes Classification

- **Mechanism of Naive Bayes Classification**
  - 값을 비교 한 후, 가장 값이 큰 클래스로 분류



Naive Bayes Classification

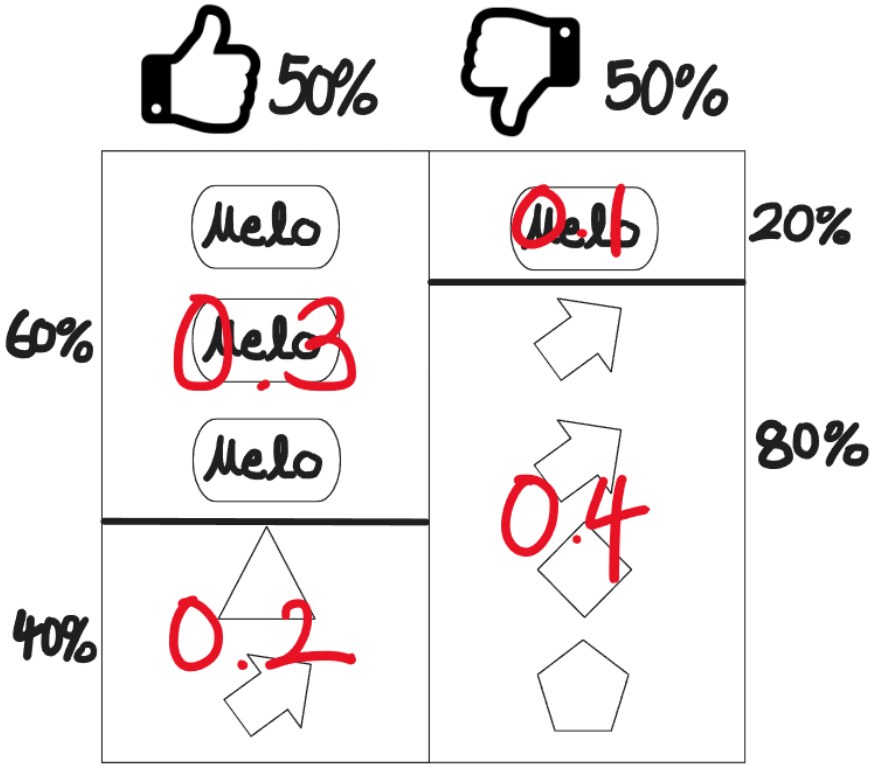
- Recommender Systems in Netflix
  - 신규 가입자 A씨의 관심 있는 장르 및 배우에 대한 정보 X -> Cold Start
    - 즉, 하나의 영화를 무작위로 추천하였을 때, 좋아할 확률 50%/좋아하지 않을 확률 50%
      - 이 확률은 Prior Probability

Naive Bayes Classification

- Recommender Systems in Netflix
  - 신규 가입자 A씨는 가입 후 여러 작품 시청 후 '좋아요', '싫어요'를 각각 5개씩 평가
    - 멜로 장르에 좋아요 3개, 싫어요 1개와 사전확률을 이용
      - 이 확률은 Posteriori Probability

Naive Bayes Classification

- Recommender Systems in Netflix



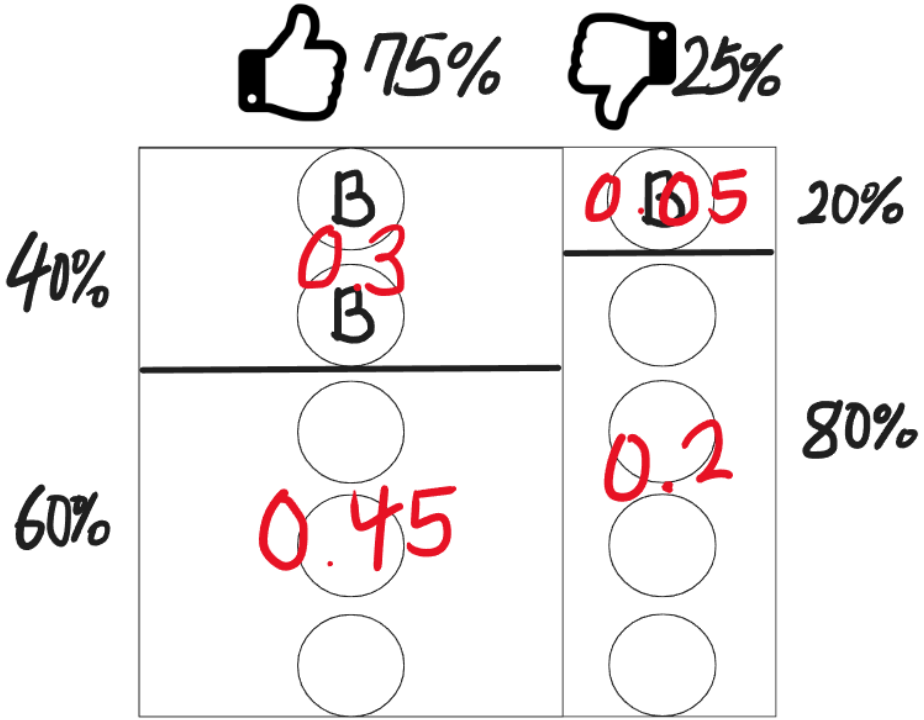
$$\frac{0.3}{0.3+0.1} = 75\%$$

Naive Bayes Classification

- Recommender Systems in Netflix
  - 주연 배우를 기준으로 '좋아요'와 '싫어요'를 평가
    - 이때, 주연 배우 B씨가 출연한 멜로 장르의 영화를 좋아할 확률은?
      - 사후 확률인 멜로 장르의 영화를 좋아할 확률 75%를 이용

Naive Bayes Classification

- Recommender Systems in Netflix



$$\frac{0.3}{0.3+0.05} \fallingdotseq 0.86$$

Naive Bayes Classification

- Recommender Systems in Netflix
  - 새로운 정보가 들어올 때마다 학습을 진행하여 추천 알고리즘의 정확도를 개선
  - 이에 맞게 사용자들에게 작품들을 노출
    - 즉, Naive Bayes Classification은 새로운 정보를 통해 정확도를 개선해 나가는 것

Conclusion & Realization

- 최근 머신 러닝 공부 뿐만 아니라 딥러닝 공부를 하면서 수학적 능력 및 이를 코드로 구현하는 능력의 중요도 파악
- 학습한 내용이 많아질수록 제대로 정리하며 쌓아가지 않는다면 추후에 학습한 내용들이 정확하게 기억 X -> 학습 내용 정리의 중요성 파악

경상국립대학교

Gyeongsang National University

Improving lives through learning

IDEALAB