**KNN과 K-Means 알고리즘을 활용한 분류 및 클러스터링 실습**

30501 김도현  
**30501 김도현**

# 1. 서론

이 보고서는 Jupyter Notebook을 활용한 실습을 기반으로 KNN, KMeans, 강화학습 알고리즘의 수학적 원리와 실제 구현을 정리한 결과물입니다. 각 알고리즘은 머신러닝 및 인공지능 분야에서 핵심적인 역할을 하며, 본 보고서에서는 해당 알고리즘들의 구현 코드, 시각화, 실험 결과 등을 종합적으로 분석합니다.

# 2. KNN (K-최근접 이웃 알고리즘)

KNN 알고리즘은 분류 문제에 주로 사용되며, 새로운 데이터를 주변의 K개의 가장 가까운 이웃을 참조해 다수결로 분류합니다.

실습에서는 sklearn의 `KNeighborsClassifier`를 사용하여 iris 데이터를 기반으로 다양한 K값에 대한 정확도를 비교하였으며, 시각화를 통해 결정 경계를 확인했습니다.

## 실습 코드 예시

```python  
import numpy as np  
#붓꽃데이터 쓸 건데   
#raw 데이터를 두 가지로 나눈다  
#랜덤하게 100개의 raw데이터가 있을 때  
#train ,test를 8:2로 나누어서  
#80개로 학습한거로 20개 정답 맞춰봐, 다맞으면 100%정확도  
#18개 맞으면 90%정확도  
from sklearn.model\_selection import train\_test\_split  
X=[[0,1],[2,3],[4,5],[6,7],[8,9]]  
Y=[0,1,2,3,4]#X의 입력에 따른 정답  
  
#train,test  
#0.2=>80%는 train 20%는 test  
X\_train, X\_test=train\_test\_split(X,test\_size=0.2,random\_state=123)  
print(X\_train)  
print(X\_test)  
```  
  
```python  
from sklearn import datasets  
  
iris=datasets.load\_iris()  
print(iris)  
```  
  
```python  
from sklearn import datasets  
iris=datasets.load\_iris()  
from sklearn.model\_selection import train\_test\_split  
X=iris.data  
Y=iris.target  
  
X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.2,random\_state=1)  
# print(X\_train)  
# print(Y\_train)  
# print(X\_test)  
# print(Y\_test)  
  
print(X\_train.shape)  
print(X\_test.shape)  
```  
  
```python  
import numpy as np  
from sklearn import datasets  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
from matplotlib.

# 3. KMeans (K-평균 군집화)

KMeans는 비지도 학습 기반의 군집화 알고리즘으로, 데이터를 K개의 그룹으로 나누는 방식입니다. 초기 중심점을 설정한 후 반복적으로 중심을 조정하며 군집을 최적화합니다.

실습에서는 `KMeans(n\_clusters=3)`를 적용하여 iris 데이터를 군집화하였고, 실제 레이블과의 시각적 비교를 통해 정확도를 분석하였습니다.

## 실습 코드 예시

# 지도학습 KNN  
  
```python  
import math  
from collections import Counter  
data=[(7,1,'기린'),  
 (4,9,'하마'),  
 (9,7,'기린'),  
 (1,8,'하마'),  
 (7,6,'기린'),  
 (4,7,'하마'),  
 (6,4,'기린'),  
 (2,9,'하마'),  
 (3,6,'하마')]  
  
x=int(input("동물의 키는"))  
y=int(input("동물의 몸무게는"))  
k=int(input('K의 수는? 홀수로'))  
def distance\_cul(x,y,a,b):  
 d=math.sqrt((x-a)\*\*2+(y-b)\*\*2)  
 return d  
def KNN(x,y,k):  
 distance=[]  
 for i in data:  
 d=distance\_cul(x,y,i[0],i[1])  
 distance.append((d,i[2]))  
 # 오름차순으로   
 distance.sort()  
 #최근접 k개  
 distance=distance[:k]  
 animals=[]  
 for i in distance:  
 animals.append(i[1])  
 x=Counter(animals).most\_common(1)  
 #most\_common은 리스트 안에 담아주니까  
 return x[0][0]  
  
print('당신이 입력한 데이터의 동물은 KNN알고리즘에 의해:',KNN(x,y,k))  
  
```  
  
# 라이브러리 KNN  
  
```python  
#아 나는 KNN 세부적인거 관심 없고 빠르게 빠르게 라이브러리로 하고 싶어! 상태일 때  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
  
a=load\_iris()  
x=a.data  
y=a.target  
  
x\_train,x\_test,y

# 4. 강화학습 (Reinforcement Learning)

강화학습은 환경과 상호작용하며 보상을 최대화하는 방향으로 학습하는 알고리즘입니다. 본 실습에서는 구글의 mediapipe를 사용하여 학습했습니다.

실습 코드 예시

# OPENCV,mediapipe를 활용한 손가락 관절 인식  
  
```python  
import cv2  
import mediapipe as mp  
  
cap=cv2.VideoCapture(0)  
mpHands=mp.solutions.hands  
  
hands=mpHands.Hands()  
print(hands)  
  
mpDraw=mp.solutions.drawing\_utils  
  
while True:  
 success,img=cap.read()  
 #openCV는 RGB를 BGR로 저장함  
 imgRGB=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)  
 results=hands.process(imgRGB)  
 print(results.multi\_hand\_landmarks)  
 #손가락을 인식했다면  
 if results.multi\_hand\_landmarks:  
 for i in results.multi\_hand\_landmarks:  
 mpDraw.draw\_landmarks(img,i)  
 cv2.imshow('text',img)  
 #1밀리초마다 0xFF비트를 검사해서 q 눌렀으면 종료  
 if cv2.waitKey(1) & 0xFF==ord('q'):  
 break  
```  
  
# 랜드마크 간 선 그리기  
  
```python  
import cv2  
import mediapipe as mp  
  
cap=cv2.VideoCapture(0)  
mpHands=mp.solutions.hands  
hands=mpHands.Hands()  
mpDraw=mp.solutions.drawing\_utils  
  
while True:  
 success,img=cap.read()  
 #이미지 반전하는 함수임, 1=좌우반전, -1=상하반전  
 img=cv2.flip(img,1)  
 imgRGB=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)  
 results=hands.process(imgRGB)

# 5. 결론

본 보고서를 통해 지도학습(KNN), 비지도학습(KMeans), 강화학습의 이론적 원리와 실습 적용을 직접 체험할 수 있었습니다. 각 알고리즘의 장단점을 비교하며, 머신러닝 모델의 구성 및 최적화 과정에 대한 이해를 심화시킬 수 있었습니다.