

SVM1

1. 문제 요약

실험적으로 이를 확인하는 간접적인 방법은 query data point를 k=2, 4..일 때 전 영역으로 확장하여 색칠을 해보면 알게 될 것으로 보인다.

- 모든 실험에 같은 학습 데이터를 사용하게 만들어야 한다. -> seed() 함수
- 학습 데이터의 레이블의 분포(비율)를 통제할 수 있어야 한다. -> choice() 함수

2. 코드

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

N = 16
K = 5    # 1 ~ 5 범위

np.random.seed(370001)
data = np.random.randint(0, 100, (N, 2)).astype(np.float32)
...

# label 0 - 7 : label 1 - 3 비율
labels = np.zeros((N, 1), dtype=np.float32)
num_class_1 = int(N * 0.3)
labels[:num_class_1] = 1
rate = "7 : 3"
...
...

# label 0 - 3 : label 1 - 7 비율
labels = np.zeros((N, 1), dtype=np.float32)
num_class_1 = int(N * 0.7)
labels[:num_class_1] = 1
rate = "3 : 7"
...

# 5 : 5 비율
labels = np.zeros((N, 1), dtype=np.float32)
num_class_1 = N // 2
labels[:num_class_1] = 1
rate = "5 : 5"
```

```

# 섞어서 랜덤하게 배치
indices = np.random.choice(N, N, replace=False)
data = data[indices]
labels = labels[indices]

# 서브플롯 생성 (2행 K//2 + 1열)
fig, axs = plt.subplots(2, (K + 1) // 2, figsize=(13, 9))
fig.suptitle(f'KNN with different k values, label 0=red triangle, 1=blue rectangle, rate = {ra

# X, Y 범위 설정
x_min, x_max = 0, 100
y_min, y_max = 0, 100

# Grid 생성
xx, yy = np.meshgrid(np.arange(x_min, x_max, 1),
                     np.arange(y_min, y_max, 1))

grid_points = np.c_[xx.ravel(), yy.ravel()]

# 각 k에 대해 예측 및 시각화
for k in range(1, K + 1):
    # KNN 모델 생성 및 훈련
    knn = cv2.ml.KNearest_create()
    knn.train(data, cv2.ml.ROW_SAMPLE, labels)

    # 모든 grid_points에 대해 예측
    _, results, _, _ = knn.findNearest(grid_points.astype(np.float32), k)
    results = results.reshape(xx.shape)

    ax = axs[(k - 1) // ((K + 1) // 2), (k - 1) % ((K + 1) // 2)]
    ax.contourf(xx, yy, results, alpha=0.3, levels=[-1, 0, 1], colors=['red', 'blue'])

    # 데이터 포인트 시각화
    red_triangles = data[labels.ravel() == 0]
    blue_squares = data[labels.ravel() == 1]

    ax.scatter(red_triangles[:, 0], red_triangles[:, 1], 200, 'r', '^')
    ax.scatter(blue_squares[:, 0], blue_squares[:, 1], 200, 'b', 's')

    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)
    ax.set_title(f'k={k}')
    ax.grid(True)

# 숨겨진 서브플롯 제거
if K % 2 != 0:
    fig.delaxes(axs[1, (K) // 2])

plt.tight_layout()
plt.show()


```

3. 결과


- 실행 초기 상태(5:5 비율)

 Initial plot

- 3:7 비율

 3:7 plot

- 7:3 비율

 T = 4 plot

4. 결론

- (0, 0)부터 (100, 100) 까지의 좌표를 범위로 지정.
- 각 좌표에서 k값에 따라 knn을 수행시켜 해당 좌표가 어느 label을 갖게 되는지 실험.
- 7:3 비율과 3:7 비율의 결과를 비교해 볼 때 label=0 으로 나타나는 범위가 비교적 넓음.
- k가 짝수일 때 외부 요소가 knn모델의 결정 시에 영향을 끼칠 수 있겠지만, 개인적인 의견으로는 label의 순서가 앞인 경우 더 우선시 되는 것 같다고 여겨짐.
- k = 1~5 까지의 현상을 하나의 pyplot로 표현.
- 7:3, 3:7, 5:5 의 비율을 코드의 수정을 통해 시각화 가능.