# 머신러닝 세션 5주차 과제

2023170832 고지원

1. **서포트 벡터 머신의 근본적인 아이디어는 무엇인가?**
   - 클래스를 가장 잘 구분하는 초평면(hyperplane)을 찾아서, 그 마진(margin)을 최대화함으로써 일반화 성능을 극대화하는 것
2. **서포트 벡터가 무엇인가?**
   - 결정 경계(Decision Boundary)에 가장 가까이 위치한 학습 데이터 포인트
3. **SVM 을 사용할 때 입력값의 스케일이 왜 중요한가?**
   - SVM 은 거리 기반 알고리즘이기 때문에, 입력 데이터의 스케일 차이가 크면 분류 경계가 왜곡되고 성능이 크게 저하될 수 있다.
4. **SVM 분류기가 샘플을 분류할 때 신뢰도 점수와 확률을 출력할 수 있는가?**
   - 신뢰도 점수에 해당하는 결정 함수 값을 출력할 수 있으며, 확률 값을 출력하려면 별도의 설정이 필요
5. **RBF 커널을 사용해 SVM 분류기를 훈련시켰더니 훈련 세트에 과소적합되었다. 이때, 어떤 값을 어떻게 수정해야 하는가?**
   - 모델의 복잡도를 높이기 위해 C 값과 gamma 값을 증가시켜야 한다.
6. **(심화) MNIST 데이터셋에 SVM 분류기를 훈련시켜라. (SVM 분류기는 이진 분류기이므로 OvR 전략을 사용하여 10 개의 숫자를 분류해야 한다.)**

```
[1]  import tensorflow as tf
     from sklearn import datasets
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.svm import SVC
     from sklearn.metrics import classification_report
```

```
[2]  # MNIST 데이터셋 불러오기
     mnist = tf.keras.datasets.mnist
     (X_train, y_train), (X_test, y_test) = mnist.load_data()

     # 데이터를 평탄화하고 정규화
     X_train = X_train.reshape(-1, 28*28) / 255.0
     X_test = X_test.reshape(-1, 28*28) / 255.0

     # SVM 분류기 훈련
     clf = SVC(kernel='rbf', C=10, gamma=0.01)
     clf.fit(X_train, y_train)

     # 예측 및 성능 평가
     y_pred = clf.predict(X_test)
     print(classification_report(y_test, y_pred))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      1.00      0.99      1135
           2       0.98      0.98      0.98      1032
           3       0.98      0.98      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.98      0.98      0.98       892
           6       0.99      0.99      0.99       958
           7       0.98      0.98      0.98      1028
           8       0.98      0.98      0.98       974
           9       0.97      0.97      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

7. **(심화) 캘리포니아 주택 가격 데이터셋에 SVM 회귀를 훈련시켜라.**

✓ 4초

```
[1] import numpy as np
    import pandas as pd
    from sklearn import datasets
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.svm import SVR
    from sklearn.metrics import mean_squared_error
    import matplotlib.pyplot as plt
```

✓ 2분

```
[2] # 캘리포니아 주택 가격 데이터셋 불러오기
    california_housing = datasets.fetch_california_housing()
    X = california_housing.data
    y = california_housing.target

    # 데이터 정규화 (특징 스케일링)
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # 훈련 데이터와 테스트 데이터 나누기
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

    # SVM 회귀 모델 훈련
    svr = SVR(kernel='rbf', C=100, epsilon=0.1, gamma='scale')
    svr.fit(X_train, y_train)

    # 테스트 데이터에 대해 예측
    y_pred = svr.predict(X_test)

    # 성능 평가 (평균 제곱 오차)
    mse = mean_squared_error(y_test, y_pred)
    print(f'Mean Squared Error (MSE): {mse}')

    # 예측 값과 실제 값을 비교하는 시각화
    plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linewidth=2)
    plt.xlabel('Actual Prices')
    plt.ylabel('Predicted Prices')
    plt.title('SVR: Actual vs Predicted Prices')
    plt.show()
```

```
Mean Squared Error (MSE): 0.31588765003267383
```