서포트 벡터 머신 분류 (Support Vector Classifier, SVC)

- 분류(Classification)와 회귀(Regression)에 모두 사용될 수 있는 강력한 지도 학습 알고리즘
- 분류 문제에서는 SVC(Support Vector Classifier)라고 불리며, 각 클래스를 가장 잘 구분하는 **결정 경계(Decision Boundary)**를 찾는 것이 목표

- **최적의 결정 경계**: SVC는 단순히 두 클래스를 나누는 선(또는 면)을 찾는 것을 넘어, 각 클래스의 가장 가까운 데이터 포인트(서포트 벡터)로부터 **가장 멀리 떨어진(Margin이 최대가 되는)** 최적의 결정 경계를 찾으려고 합니다. 이 최대 마진(Maximum Margin) 덕분에 SVC는 일반화 성능이 뛰어나고 과적합에 강한 경향을 보입니다.
- **서포트 벡터 (Support Vectors)**: 결정 경계와 가장 가까이에 위치하여, 이 경계를 정의하는 데 직접적인 영향을 주는 데이터 포인트들을 의미합니다. 다른 데이터 포인트들은 결정 경계를 만드는 데 영향을 주지 않습니다.
- **커널 트릭 (Kernel Trick)**: 선형적으로 분리할 수 없는 복잡한 데이터의 경우, '커널 함수'를 사용하여 데이터를 더 높은 차원의 특성 공간으로 매핑합니다. 이 고차원 공간에서는 데이터가 선형적으로 분리될 수 있으며, 이를 통해 비선형 결정 경계를 효과적으로 만들 수 있습니다. 대표적인 커널로는 `rbf`(Radial Basis Function), `poly`(Polynomial), `linear` 등이 있습니다.

적용 가능한 상황

- 특성의 수가 많은 고차원 데이터셋(e.g., 이미지, 텍스트 데이터)에서 효과적입니다.
- 데이터의 양이 아주 많지는 않지만, 복잡한 결정 경계를 필요로 하는 분류 문제에 적합합니다.
- 명확한 마진을 통해 클래스를 구분할 수 있는 문제에서 뛰어난 성능을 보입니다.

구현 방법

`scikit-learn`의 `svm` 모듈에 있는 `SVC` 클래스를 사용합니다.

용도

- 데이터 포인트들을 가장 잘 구분하는 선형 또는 비선형 결정 경계를 찾아 새로운 데이터를 분류합니다.

주의사항

- **특성 스케일링 필수**
  - SVC는 거리 및 마진 기반 알고리즘
  - 특성들의 스케일이 다르면 큰 값을 가진 특성이 모델을 지배
  - **`StandardScaler` 등으로 스케일링 필수**
- **하이퍼파라미터 튜닝**: SVC는 성능에 큰 영향을 미치는 여러 하이퍼파라미터를 가지고 있어, `GridSearchCV` 등을 통한 튜닝이 매우 중요합니다.
  - `kernel`: 사용할 커널 함수. 'linear', 'poly', 'rbf' 등이 있으며, 'rbf'가 일반적으로 가장 많이 사용됩니다.
  - `C`: 규제 파라미터. 마진 위반(잘못 분류된 데이터)에 대한 페널티를 조절합니다. `C`가 클수록 마진 위반을 엄격하게 금지하여 모델이 훈련 데이터에 더 적합(과적합 위험)되고, 작을수록 마진을 더

허용(일반화)합니다.
    - `gamma`: 'rbf', 'poly' 커널에 사용되는 파라미터로, 하나의 데이터 샘플이 미치는 영향의 범위를 결정합니다. `gamma`가 크면 영향의 범위가 좁아져 결정 경계가 더 복잡해지고 과적합될 수 있습니다.
- **클래스 불균형**: `class_weight='balanced'` 옵션을 통해 클래스 불균형 문제에 대처할 수 있습니다.
- **확률 예측**: SVC는 기본적으로 확률 정보를 제공하지 않습니다. 확률 예측이 필요하면 `probability=True`로 설정해야 하지만, 이 경우 학습 시간이 더 오래 걸립니다.

```
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# 1. 데이터 준비 및 전처리
cancer = load_breast_cancer()
X, y = cancer.data, cancer.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)

# SVC는 스케일링이 매우 중요
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 2. SVC 모델 학습 (RBF 커널)
# 주요 하이퍼파라미터
# kernel: 커널 종류 ('linear', 'poly', 'rbf', 'sigmoid')
# C: 규제 파라미터
# gamma: 커널 계수
svc_rbf = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svc_rbf.fit(X_train_scaled, y_train)
svc_pred = svc_rbf.predict(X_test_scaled)
print(f"SVC (RBF Kernel) 정확도: {accuracy_score(y_test, svc_pred):.3f}") # 0.977

# 3. GridSearchCV를 이용한 최적 하이퍼파라미터 탐색
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.01, 0.1, 1]
}

grid_svc = GridSearchCV(SVC(kernel='rbf', random_state=42), param_grid, cv=5,
scoring='accuracy', n_jobs=-1)
grid_svc.fit(X_train_scaled, y_train)

print("\nGridSearchCV 최적 파라미터 (SVC):", grid_svc.best_params_) # {'C': 10,
'gamma': 0.01}
print(f"최적 파라미터 적용 시 정확도: {grid_svc.best_score_:.3f} (교차검증 평균)") #
0.975

# 최적 모델로 예측 및 평가
best_svc = grid_estimator_
```

```
best_pred = best_svc.predict(X_test_scaled)
print("\nBest SVC 분류 리포트:\n", classification_report(y_test, best_pred))
'''
               precision    recall  f1-score   support

           0       0.97      0.98      0.98        64
           1       0.99      0.98      0.99       107

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171
'''
```

결과 해석 방법

- SVC는 내부적으로 복잡한 연산을 수행하므로, 로지스틱 회귀나 결정 트리처럼 모델 자체를 직관적으로 해석하기는 어렵습니다.
- 모델의 성능은 정확도, 정밀도, 재현율, F1-점수 등 일반적인 분류 평가지표를 통해 평가합니다.
- `support_vectors_` 속성을 통해 어떤 데이터 포인트가 결정 경계를 만드는 데 사용된 서포트 벡터인지 확인할 수 있습니다.
- `n_support_` 속성은 각 클래스별 서포트 벡터의 개수를 보여줍니다.

장단점 및 대안

- **장점**:
  - 커널 트릭을 통해 복잡한 비선형 결정 경계를 만들 수 있어 높은 분류 성능을 보입니다.
  - 특성 수가 많은 고차원 데이터에서도 효과적입니다.
  - 최대 마진 개념 덕분에 일반화 성능이 우수하고 과적합에 강한 편입니다.
- **단점**:
  - 데이터의 양이 매우 클 경우 학습 속도가 현저히 느려집니다.
  - 어떤 커널과 하이퍼파라미터를 사용해야 할지 결정하기 위해 교차 검증을 통한 튜닝 과정이 거의 필수적입니다.
  - 모델의 내부 동작을 직관적으로 해석하기 어렵습니다.
- **대안**:
  - **로지스틱 회귀**: 모델의 해석이 중요하고, 데이터가 선형적으로 잘 분리될 때 좋은 대안입니다.
  - **랜덤 포레스트 / 그래디언트 부스팅**: 대용량 데이터셋에서 SVC보다 빠른 속도와 높은 성능을 보이는 경우가 많으며, 특성 중요도 등 해석을 위한 부가 정보를 제공합니다.
  - **K-최근접 이웃 (KNN)**: 더 간단한 비선형 분류 모델이 필요할 때 사용할 수 있습니다.