규제 선형 모델 (Regularized Linear Models)

- 기존 선형 회귀 모델의 비용 함수(Cost Function)에 **규제 항(Regularization Term)**을 추가하여 모델의 과적합(Overfitting)을 방지하고 일반화 성능을 높이는 기법
- 규제 항은 회귀 계수(가중치)의 크기를 제어하는 역할을 하며, 모델이 훈련 데이터에 너무 복잡하게 맞춰지는 것을 방지
- **기존 선형 회귀 비용 함수**: $Cost(W) = MSE(W)$
- **규제 모델 비용 함수**: $Cost(W) = MSE(W) + \alpha \cdot R(W)$
  - $MSE(W)$: 기존의 평균 제곱 오차
  - $R(W)$: 규제 항. 계수들의 크기를 측정하는 함수.
  - α: 규제 강도(Hyperparameter). α가 클수록 규제가 강해져 계수들이 0에 가까워지고, α가 0이면 일반 선형 회귀와 동일해집니다.

규제 항의 형태에 따른 모델 종류

1. **릿지 회귀 (Ridge Regression)**:
   - **규제 항**: L2 노름(Norm)을 사용. 모든 계수를 제곱한 값의 합. $R(W) = \sum_{i=1}^{n} w_i^2$
   - **특징**: 계수의 크기를 전반적으로 작게 만들어 과적합을 완화합니다. 계수를 0에 가깝게 만들지만 완전히 0으로 만들지는 않습니다. 다중공선성 문제에 특히 효과적입니다.
2. **라쏘 회귀 (Lasso Regression)**:
   - **규제 항**: L1 노름(Norm)을 사용. 모든 계수의 절대값의 합. $R(W) = \sum_{i=1}^{n} |w_i|$
   - **특징**: 중요하지 않은 특성의 회귀 계수를 완전히 0으로 만들어, **특성 선택(Feature Selection)** 효과를 가집니다. 모델을 단순화하고 해석력을 높일 수 있습니다.
3. **엘라스틱넷 회귀 (ElasticNet Regression)**:
   - **규제 항**: 릿지(L2)와 라쏘(L1) 규제를 결합한 형태. $R(W) = r \cdot (\sum |w_i|) + \frac{1-r}{2} \cdot (\sum w_i^2)$
   - **특징**: 릿지와 라쏘의 장점을 모두 가집니다. `l1_ratio` 파라미터(r)를 통해 L1 규제와 L2 규제의 비율을 조절할 수 있습니다. 상관관계가 높은 특성들이 많을 때 라쏘보다 더 안정적인 성능을 보입니다.

적용 가능한 상황

- 모델이 훈련 데이터에는 잘 맞지만 테스트 데이터에서 성능이 떨어지는 **과적합**이 의심될 때.
- 독립변수들 간의 상관관계가 높아 **다중공선성** 문제가 있을 때 (특히 릿지).
- 독립변수가 매우 많아 모델을 단순화하고 **중요한 변수만 선택**하고 싶을 때 (라쏘, 엘라스틱넷).

구현 방법

`scikit-learn`의 `linear_model` 모듈에 있는 `Ridge`, `Lasso`, `ElasticNet` 클래스를 사용합니다.

주의사항

- **특성 스케일링 필수**
  - 규제는 계수의 크기에 직접적으로 영향을 받으므로, 모든 특성들이 동일한 스케일을 갖도록 **스케일링(e.g. `StandardScaler`)을 적용 필수**
  - 스케일링을 하지 않으면 특정 변수의 계수만 부당하게 커져 규제가 제대로 작동하지 않을 수 있습니다.
- **최적의 `alpha` 찾기**
  - 규제 강도 `alpha`는 모델 성능에 큰 영향을 미치는 중요한 하이퍼파라미터
  - `GridSearchCV`나 `RandomizedSearchCV`를 통한 교차 검증으로 최적의 `alpha` 값을 찾는 과정이 필요
  - `RidgeCV`, `LassoCV`, `ElasticNetCV`와 같이 교차 검증 기능이 내장된 클래스를 사용 편리

```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error

# 1. 데이터 준비 및 전처리
housing = fetch_california_housing()
X = housing.data
y = housing.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 2. 릿지 회귀 (Ridge Regression)
# alpha: 규제 강도.
ridge = Ridge(alpha=1.0)
ridge.fit(X_train_scaled, y_train)
ridge_pred = ridge.predict(X_test_scaled)
print(f"Ridge MSE: {mean_squared_error(y_test, ridge_pred):.3f}") # 0.556

# 3. 라쏘 회귀 (Lasso Regression)
# alpha: 규제 강도.
lasso = Lasso(alpha=0.01)
lasso.fit(X_train_scaled, y_train)
lasso_pred = lasso.predict(X_test_scaled)
print(f"Lasso MSE: {mean_squared_error(y_test, lasso_pred):.3f}") # 0.548

# 라쏘의 특성 선택 효과 확인: 계수가 0이 된 변수들
lasso_coef = pd.Series(lasso.coef_, index=housing.feature_names)
print("\nLasso 회귀 계수 (0이 아닌 것):\n", lasso_coef[lasso_coef !=
0].sort_values())
'''
Latitude    -0.790113
Longitude   -0.755674
```

```
AveRooms    -0.162759
AveOccup    -0.030602
HouseAge     0.127087
AveBedrms    0.206207
MedInc       0.800957
dtype: float64
'''

# 4. 엘라스틱넷 회귀 (ElasticNet Regression)
# alpha: 전체 규제 강도.
# l1_ratio: L1 규제의 비율 (0 <= l1_ratio <= 1). 0이면 L2(릿지), 1이면 L1(라쏘).
elastic = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic.fit(X_train_scaled, y_train)
elastic_pred = elastic.predict(X_test_scaled)
print(f"\nElasticNet MSE: {mean_squared_error(y_test, elastic_pred):.3f}") # 0.636

# 5. GridSearchCV를 이용한 최적 alpha 찾기 (Ridge 예시)
ridge_params = {'alpha': [0.01, 0.1, 1, 10, 100]}
grid_ridge = GridSearchCV(Ridge(), param_grid=ridge_params,
scoring='neg_mean_squared_error', cv=5)
grid_ridge.fit(X_train_scaled, y_train)

print(f"\nGridSearchCV 최적 alpha (Ridge): {grid_ridge.best_params_['alpha']}") #
0.1
print(f"최적 alpha 적용 시 MSE: {-grid_ridge.best_score_:.3f} (교차검증 평균)") #
0.519
```

결과 해석 방법

- `coef_`: 각 모델의 학습된 회귀 계수를 확인합니다.
  - **릿지**: 계수들이 전반적으로 작아진 것을 확인할 수 있습니다.
  - **라쏘**: 일부 계수들이 정확히 0이 된 것을 볼 수 있으며, 이는 해당 특성이 모델에서 제외되었음을 의미합니다. 0이 아닌 계수를 가진 특성들이 모델이 선택한 중요한 변수들입니다.
  - **엘라스틱넷**: 라쏘와 유사하게 일부 계수가 0이 될 수 있으며, 릿지와 라쏘의 절충된 결과를 보입니다.
- **최적의 `alpha`**
  - 교차 검증을 통해 찾은 최적의 `alpha` 값은 해당 데이터셋에서 **일반화 성능이 가장 좋은 규제 강도**를 의미합니다.
  - `alpha`가 너무 크면 과소적합(Underfitting), 너무 작으면 과적합(Overfitting)이 될 수 있습니다.

장단점 및 대안

- **장점**:
  - **릿지**: 과적합 방지에 효과적이며, 특히 다중공선성 문제에 강건합니다.
  - **라쏘**: 과적합 방지와 동시에 변수 선택 기능까지 제공하여 모델을 단순화하고 해석력을 높입니다.
  - **엘라스틱넷**: 릿지와 라쏘의 장점을 결합하여, 변수 간 상관관계가 높은 상황에서도 안정적인 변수 선택이 가능합니다.
- **단점**:
  - 최적의 규제 강도 `alpha`를 찾기 위한 하이퍼파라미터 튜닝 과정이 필요합니다.
  - 선형 관계를 가정하는 것은 일반 선형 회귀와 동일합니다.

- **대안**:
    - **트리 기반 모델 (RandomForest, GradientBoosting)**: 규제가 필요 없으며, 변수 중요도(Feature Importance)를 통해 중요한 변수를 파악할 수 있습니다.
    - **PCA (주성분 분석)**: 회귀 분석 전에 PCA를 적용하여 다중공선성 문제를 해결하고 변수의 수를 줄일 수 있습니다.