분위수 회귀 (Quantile Regression)

- 일반적인 선형 회귀가 종속변수의 조건부 '평균'을 예측하는 것과 달리, 조건부 '분위수'(Quantile)를 예측하는 회귀 분석 기법
- 0.5 분위수(중앙값)를 예측할 수도 있고, 0.1, 0.9 분위수 등을 예측하여 종속변수의 전체적인 분포를 더 폭넓게 이해할 수 있다.

적용 가능한 상황

- **이상치에 강건한 모델링**: 중앙값(0.5 분위수)을 예측하도록 설정하면, 평균을 사용하는 OLS에 비해 이상치의 영향을 덜 받는 강건한 회귀 모델을 만들 수 있습니다.
- **이분산성(Heteroscedasticity) 존재**: 오차의 분산이 독립변수 값에 따라 변하는 이분산성 데이터에서, 분위수 회귀는 각기 다른 분위수에서 변수들의 영향력이 어떻게 달라지는지 파악하는 데 유용합니다. 예를 들어, 소득 하위 10% 그룹과 상위 10% 그룹에서 교육 수준이 소득에 미치는 영향이 다를 수 있음을 모델링할 수 있습니다.
- **특정 분위수에 대한 예측 필요**: 특정 분위수(e.g., 최악의 시나리오를 가정한 10% 분위수, 최상의 시나리오를 가정한 90% 분위수)에 대한 예측이 필요할 때 직접적으로 사용할 수 있습니다. (e.g., 영유아 성장 곡선, 재무 위험 관리)

구현 방법

분위수 회귀는 `statsmodels` 라이브러리를 통해 주로 구현됩니다. `scikit-learn`에서는 `QuantileRegressor`(v0.23 이상)를 제공하지만, 통계적 추론 기능은 `statsmodels`가 더 풍부합니다.

용도

- 종속변수의 조건부 평균이 아닌, 다양한 조건부 분위수(중앙값, 10% 분위수 등)를 모델링합니다.

주의사항

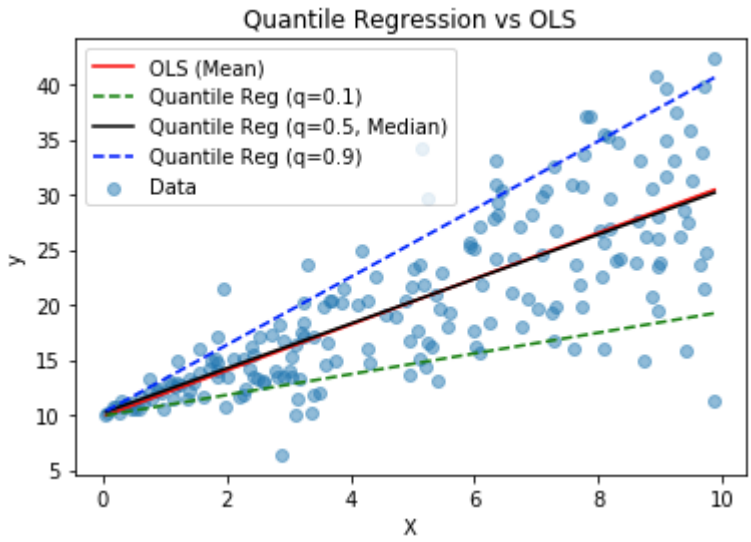
- 분위수 회귀는 OLS와 달리 잔차의 정규성이나 등분산성을 가정하지 않습니다.
- 계산 과정이 선형 계획법(Linear Programming)에 기반하므로 OLS보다 계산 비용이 더 높을 수 있습니다.
- 예측하려는 분위수 `q` 값(0 < q < 1)을 지정해야 합니다.

1. `statsmodels` 예시

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

# 1. 데이터 생성 (이분산성)
np.random.seed(42)
n_samples = 200
X = np.random.uniform(0, 10, n_samples)
# X가 클수록 오차의 분산이 커지도록 설정
error = np.random.normal(0, X, n_samples)
```

```
y = 10 + 2 * X + error

data = pd.DataFrame({'X': X, 'y': y})

# 2. OLS 모델과 분위수 회귀 모델(10%, 50%, 90%) 학습
# smf.quantreg('종속변수 ~ 독립변수', 데이터)
# q: 예측할 분위수 (0 < q < 1)
q_10 = smf.quantreg('y ~ X', data).fit(q=0.1)
q_50 = smf.quantreg('y ~ X', data).fit(q=0.5) # 중앙값 회귀
q_90 = smf.quantreg('y ~ X', data).fit(q=0.9)
ols = smf.ols('y ~ X', data).fit()

print("--- 10% 분위수 회귀 결과 ---")
print(q_10.summary())
'''
--- 10% 분위수 회귀 결과 ---
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                      y   Pseudo R-squared:               0.2291
Model:                       QuantReg   Bandwidth:                       4.128
Method:                 Least Squares   Sparsity:                        14.01
Date:                Fri, 10 Oct 2025   No. Observations:                  200
Time:                        15:28:07   Df Residuals:                      198
                                        Df Model:                            1
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      9.9227      0.565     17.557      0.000       8.808      11.037
X              0.9418      0.097      9.717      0.000       0.751       1.133
==============================================================================
'''
print("\n--- 50% 분위수(중앙값) 회귀 결과 ---")
print(q_50.summary())
'''
--- 50% 분위수(중앙값) 회귀 결과 ---
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                      y   Pseudo R-squared:               0.4474
Model:                       QuantReg   Bandwidth:                       3.330
Method:                 Least Squares   Sparsity:                        10.06
Date:                Fri, 10 Oct 2025   No. Observations:                  200
Time:                        15:28:07   Df Residuals:                      198
                                        Df Model:                            1
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     10.1705      0.685     14.855      0.000       8.820      11.521
X              2.0258      0.121     16.759      0.000       1.787       2.264
==============================================================================
'''
print("\n--- 90% 분위수 회귀 결과 ---")
print(q_90.summary())
'''
--- 90% 분위수 회귀 결과 ---
```

```
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                      y   Pseudo R-squared:               0.5366
Model:                       QuantReg   Bandwidth:                       4.242
Method:                 Least Squares   Sparsity:                        13.99
Date:                Fri, 10 Oct 2025   No. Observations:                  200
Time:                        15:28:07   Df Residuals:                      198
                                        Df Model:                            1
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     10.1672      0.611     16.628      0.000       8.961      11.373
X              3.0842      0.108     28.576      0.000       2.871       3.297
==============================================================================
'''

# 3. 결과 시각화
x_plot = np.linspace(data['X'].min(), data['X'].max(), 100)
plt.scatter(data['X'], data['y'], alpha=0.5, label='Data')
plt.plot(x_plot, ols.predict({'X': x_plot}), color='red', label='OLS (Mean)')
plt.plot(x_plot, q_10.predict({'X': x_plot}), color='green', linestyle='--',
label='Quantile Reg (q=0.1)')
plt.plot(x_plot, q_50.predict({'X': x_plot}), color='black', label='Quantile Reg
(q=0.5, Median)')
plt.plot(x_plot, q_90.predict({'X': x_plot}), color='blue', linestyle='--',
label='Quantile Reg (q=0.9)')

plt.title('Quantile Regression vs OLS')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```



2. scikit-learn 예시

- sklearn 1.0. 버전 이상에서 가능

```
from sklearn.linear_model import QuantileRegressor

# scikit-learn 입력을 위해 2D 배열로 변환
X_sk = X.reshape(-1, 1)

# QuantileRegressor 하이퍼파라미터
# quantile: 예측할 분위수 (0 < quantile < 1). (기본값=0.5)
# alpha: 규제 강도 (L1 규제). (기본값=1.0)
# solver: 최적화에 사용할 솔버. 데이터 크기에 따라 'highs', 'highs-ds', 'highs-ipm'
등을 선택. (기본값='highs')
q_reg_50 = QuantileRegressor(quantile=0.5, alpha=0)
q_reg_50.fit(X_sk, y)

print(f"Scikit-learn 50% 분위수 회귀 계수: {q_reg_50.coef_}, 절편:
{q_reg_50.intercept_}")
# Scikit-learn 50% 분위수 회귀 계수: [2.02581967], 절편: 10.170481620593494
```

결과 해석 방법

- **회귀 계수**: `statsmodels`의 `summary()` 결과에서 각 독립변수의 계수(`coef`)와 p-value(`P>|t|`)를 확인합니다. p-value가 유의수준(e.g., 0.05)보다 작으면 해당 변수가 특정 분위수에서 종속변수에 유의미한 영향을 미친다고 해석할 수 있습니다.
- **분위수별 계수 비교**: 서로 다른 분위수(e.g., 0.1, 0.9)에서 동일한 변수의 회귀 계수가 어떻게 달라지는지 확인합니다. 예를 들어, 90% 분위수에서의 계수가 10% 분위수에서의 계수보다 크다면, 해당 독립변수는 종속변수의 상위 값에 더 큰 영향을 미친다고 해석할 수 있습니다(이분산성 존재 암시).
- **시각화**: OLS 회귀선은 하나지만, 분위수 회귀선은 여러 개를 그릴 수 있습니다. 각 분위수 회귀선이 데이터의 분포를 어떻게 나누고 있는지, 특히 데이터가 퍼져있는 구간에서 회귀선들 간의 간격이 넓어지는지(이분산성) 등을 직관적으로 파악할 수 있습니다.

장단점 및 대안

- **장점**:
  - 이상치에 강건한 모델을 만들 수 있습니다 (특히 중앙값 회귀).
  - 종속변수의 전체 분포에 대한 포괄적인 정보를 제공합니다.
  - 이분산성 데이터 구조를 효과적으로 모델링하고 설명할 수 있습니다.
- **단점**:
  - OLS에 비해 계산이 복잡하고 시간이 더 걸립니다.
  - 결과(회귀 계수)가 OLS처럼 평균적인 영향력이 아니라 특정 분위수에서의 영향력이므로 해석에 주의가 필요합니다.
- **대안**:
  - **강건 회귀 (Robust Regression)**: 이상치 문제에 초점을 맞춘다면 Huber, RANSAC 등 다른 강건 회귀 기법을 사용할 수 있습니다.
  - **일반화 최소 제곱법 (GLS)**: 이분산성이나 자기상관이 있는 오차 구조를 명시적으로 모델링할 때 사용될 수 있습니다.
  - **로그 변환 등 변수 변환**: 종속변수에 로그 변환을 적용하여 이분산성을 완화한 후 OLS를 적용하는 방법도 있습니다.