베이지안 회귀 (Bayesian Regression)

- 베이즈 정리를 기반으로 회귀 계수를 추정하는 통계적 방법
- 베이지안 회귀에서는 회귀 계수를 **'확률 분포를 따르는 변수'**로 취급
    - 일반적인 빈도주의(Frequentist) 회귀 분석은 회귀 계수를 **'고정된 상수'**로 간주
- 데이터로부터 계수의 정확한 값 하나를 찾는 것이 아니라, 계수가 가질 수 있는 값의 범위와 그 확률(신뢰 구간과 유사한 '신용 구간')을 추정

베이지안 회귀 과정

1. **사전 분포 (Prior Distribution)**: 분석가가 회귀 계수에 대해 가지고 있는 사전 지식이나 신념을 확률 분포로 설정합니다. (e.g., "이 계수는 0에 가까울 것이다.")
2. **가능도 (Likelihood)**: 주어진 데이터가 특정 회귀 계수 값에서 나타날 확률을 계산합니다.
3. **사후 분포 (Posterior Distribution)**: 사전 분포와 가능도를 결합(베이즈 정리 적용)하여, 데이터를 관찰한 후 업데이트된 회귀 계수의 확률 분포를 구합니다. 이 사후 분포가 베이지안 회귀의 최종 결과물이 됩니다.

적용 가능한 상황

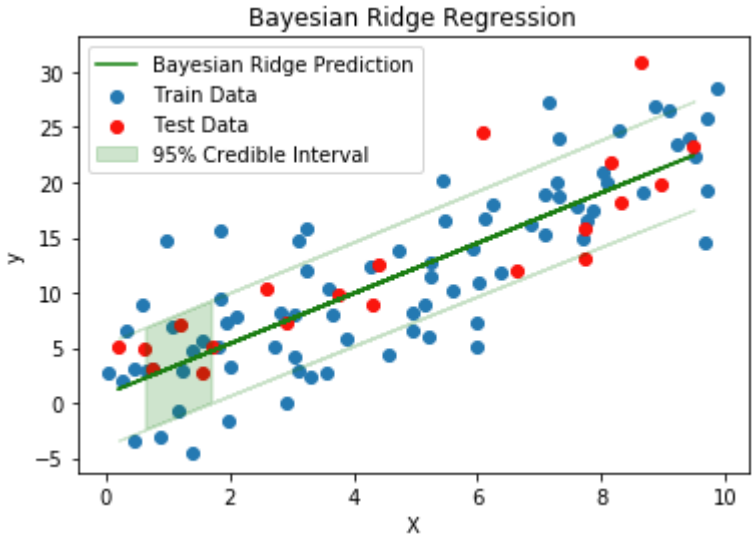
- **데이터의 양이 적을 때**: 사전 지식을 모델에 반영하여 부족한 데이터로부터 오는 불확실성을 보완하고 더 안정적인 추론을 할 수 있습니다.
- **과적합 방지**: 사전 분포를 통해 회귀 계수의 크기에 제약을 가하는 효과(규제와 유사)를 주어 과적합을 방지할 수 있습니다.
- **결과의 불확실성 측정**: 회귀 계수나 예측 결과가 단일 값이 아닌 확률 분포로 제공되므로, "계수가 0일 확률은 얼마인가?", "예측값의 95% 신용 구간은 어디인가?"와 같은 질문에 답하며 결과의 불확실성을 명확하게 표현하고 싶을 때 유용합니다.

`scikit-learn`의 `BayesianRidge`

- **용도**
    - 릿지(Ridge) 회귀에 베이지안 접근법을 적용한 모델
    - 규제 강도 파라미터(alpha, lambda)를 스스로 추정하여 최적화하므로, 교차 검증을 통한 튜닝 과정이 간소화
- **주의사항**: 사전 분포가 특정 형태(정규분포)로 고정되어 있어 유연성이 떨어집니다.

```
import numpy as np
from sklearn.linear_model import BayesianRidge, LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# 1. 데이터 생성
np.random.seed(42)
X = np.random.rand(100, 1) * 10
y = 2.5 * X.ravel() + np.random.randn(100) * 5

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# 2. 모델 학습
# BayesianRidge 하이퍼파라미터
# n_iter: 최대 반복 횟수. (기본값=300)
# tol: 수렴 조건. (기본값=1e-3)
# alpha_1, alpha_2: alpha(정밀도) 사전분포(감마분포)의 하이퍼파라미터. (기본값=1e-6)
# lambda_1, lambda_2: lambda(정밀도) 사전분포(감마분포)의 하이퍼파라미터. (기본값=1e-6)
#   - 이 값들은 보통 기본값을 사용하며, 모델이 데이터로부터 alpha와 lambda를 추정하게 됩니다.
br = BayesianRidge()
br.fit(X_train, y_train)

lr = LinearRegression()
lr.fit(X_train, y_train)

# 3. 예측 및 불확실성 확인
# return_std=True로 설정하면 예측값의 표준편차를 함께 반환합니다.
y_pred_br, y_std_br = br.predict(X_test, return_std=True)
y_pred_lr = lr.predict(X_test)

# 4. 결과 시각화
plt.scatter(X_train, y_train, label='Train Data')
plt.scatter(X_test, y_test, color='red', label='Test Data')

# 베이지안 회귀 예측 및 신용 구간
plt.plot(X_test, y_pred_br, color='green', label='Bayesian Ridge Prediction')
plt.fill_between(X_test.ravel(), y_pred_br - y_std_br, y_pred_br + y_std_br,
color='green', alpha=0.2, label='95% Credible Interval')

plt.title('Bayesian Ridge Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()

# 5. 추정된 파라미터 확인
print(f"추정된 회귀 계수 (평균): {br.coef_}")               # [2.28556861]
print(f"추정된 절편 (평균): {br.intercept_}")               # 0.7789537607408423
print(f"추정된 alpha (오차의 정밀도): {br.alpha_:.4f}")     # 0.0466
print(f"추정된 lambda (가중치의 정밀도): {br.lambda_:.4f}") # 0.1903
```



결과 해석 방법

- **회귀 계수 (`br.coef_`, `br.intercept_`)**: 사후 분포의 평균값으로, 점 추정치에 해당합니다.
- `br.sigma_`: 추정된 회귀 계수들의 공분산 행렬입니다. 이를 통해 계수 추정치의 불확실성을 알 수 있습니다.
- `br.alpha_`, `br.lambda_`: 데이터로부터 추정된 정밀도 파라미터입니다. `alpha_`는 오차(noise)의 정밀도(분산의 역수)이고, `lambda_`는 회귀 계수(가중치)의 정밀도입니다. `lambda_`가 클수록 계수들이 0에 가깝게 수축되어 규제 효과가 커집니다.
- **예측의 불확실성**: `predict` 메소드에서 `return_std=True`로 설정하여 얻은 표준편차를 이용해 예측값의 신용 구간(Credible Interval)을 계산할 수 있습니다. 이는 예측값이 특정 범위 내에 존재할 확률을 나타내며, 모델의 불확실성을 시각적으로 보여줍니다.

장단점 및 대안

- **장점**:
  - 모델 파라미터와 예측 결과의 불확실성을 확률적으로 명확하게 표현할 수 있습니다.
  - 사전 지식을 모델에 통합할 수 있어 적은 데이터에서도 안정적인 추론이 가능합니다.
  - 규제 파라미터를 데이터로부터 자동 추정하여 교차 검증의 필요성을 줄여줍니다.
- **단점**:
  - 사전 분포 설정이 주관적일 수 있으며, 결과에 영향을 미칩니다.
  - 계산 과정이 복잡하고(특히 MCMC 사용 시) 빈도주의 방법에 비해 시간이 오래 걸립니다.
  - `scikit-learn`의 구현은 기능이 제한적이어서, 복잡한 모델링을 위해서는 `PyMC` 등 별도의 라이브러리 학습이 필요합니다.
- **대안**:
  - **릿지(Ridge)/라쏘(Lasso) 회귀**: 과적합 방지가 주 목적이라면, 해석과 사용이 더 간단한 규제 선형 회귀 모델을 사용할 수 있습니다. 이들은 베이지안 회귀의 특정 사전 분포(정규분포, 라플라스분포)와 수학적으로 연결됩니다.
  - **부트스트래핑 (Bootstrapping)**: 데이터에서 반복적으로 샘플을 추출하여 여러 모델을 만든 후, 계수나 예측값의 분포를 확인하여 불확실성을 추정하는 빈도주의적 접근법입니다.