第10章

ブートストラップ法と信頼区間

データ解析入門

中村 知繁

T. Nakamura | Juntendo Univ.

2025/02/08

1. 線形回帰モデルについて

まず、基本となる線形回帰モデルから始めましょう。線形回帰は、一つの変数（**説明変数**、x）を使って、もう一つの変数（**目的変数**、y）を予測・説明するための最も基本的なモデルです。

モデルは以下のような数式で表されます。

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

- y_i: i番目のサンプルの目的変数（例：家の価格）
- x_i: i番目のサンプルの説明変数（例：家の広さ）
- β_0: **切片 (Intercept)**
- β_1: **回帰係数 (Regression Coefficient)** または **傾き (Slope)**
- ε_i: **誤差項 (Error Term)**

目的は、データ $(x_1, y_1), \ldots, (x_n, y_n)$ を使い、**最小二乗法 (OLS)** で最適な $\hat{\beta}_0, \hat{\beta}_1$ を見つけることです。

Pythonコード例：基本的な線形回帰

綺麗な線形関係を持つデータを生成し、`statsmodels` で回帰モデルを当てはめます。

```python
# 必要なライブラリをインポート
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns

# スタイルの設定
sns.set_style('whitegrid')
np.random.seed(42)

# 1. データの生成
X = np.linspace(0, 10, 100)
beta_0_true = 2.0
beta_1_true = 5.0
epsilon = np.random.normal(0, 2.0, 100)
y = beta_0_true + beta_1_true * X + epsilon

# 2. statsmodelsのためのデータ準備（切片項を追加）
X_const = sm.add_constant(X)

# 3. モデルの学習（最小二乗法）
model = sm.OLS(y, X_const)
results = model.fit()

# 4. 結果の表示と可視化
print(results.summary())
plt.figure(figsize=(10, 6))
plt.scatter(X, y, label='Data points')
plt.plot(X, results.predict(X_const), color='red', linewidth=2, label='OLS Regression Line')
plt.title('Basic Linear Regression'); plt.xlabel('X'); plt.ylabel('y')
```

2. 線形回帰モデルの回帰係数のばらつきについて

`results.summary()` には `std err` (標準誤差) や 95%信頼区間が表示されます。なぜなら、**私たちが得た回帰係数 $\hat{\beta}_1$ は、あくまでサンプルデータから計算された推定値に過ぎない**からです。

- **標準誤差**: もし別のサンプルだったら、係数はどれくらい変動するかの指標。
- **信頼区間**: そのばらつきを考慮して「真の β_1 が含まれるであろう区間」。

`statsmodels` の計算は、誤差項 ε_i が以下の**理論的な仮定**を満たすことを前提とします。

1. **正規性**: 誤差は正規分布に従う。
2. **等分散性**: 誤差の分散は、x の値によらず一定である。
3. **独立性**: 誤差は互いに独立である。

3. 外れ値に対する回帰係数の不安定性

最小二乗法（OLS）は、残差の「二乗」和を最小化するため、一つでも**非常に大きな残差（外れ値）があると、その点に強く影響を受けてしまいます。**

データに意図的に外れ値を一つ加えて、回帰係数がどう変わるか見てみましょう。

!(https://www.google.com/search?q=https://i.imgur.com/8E3sU2D.png)

Pythonコード例：外れ値の影響

```python
# 1. 外れ値の追加
X_outlier = np.append(X, 1)   # x=1 の位置に
y_outlier = np.append(y, 40)  # y=40 という大きな値を追加

# 2. statsmodelsのためのデータ準備
X_outlier_const = sm.add_constant(X_outlier)

# 3. モデルの再学習
model_outlier = sm.OLS(y_outlier, X_outlier_const)
results_outlier = model_outlier.fit()

# 4. 結果の表示と可視化
print(f"Original beta_1: {results.params[1]:.2f}")
print(f"Outlier beta_1: {results_outlier.params[1]:.2f}") # -> 5.04から3.66へ劇的に変化

plt.figure(figsize=(12, 7))
plt.scatter(X_outlier, y_outlier, label='Data with Outlier', alpha=0.6)
plt.plot(X, results.predict(X_const), color='blue', linestyle='--', label=f'Original Line (beta_1={results.params[1]:.2f})')
plt.plot(X_outlier, results_outlier.predict(X_outlier_const), color='red', linewidth=2, label=f'Outlier Line (beta_1={results_outlier.params[1]:.2f})')
plt.title('Impact of an Outlier'); plt.xlabel('X'); plt.ylabel('y')
plt.legend(); plt.show()
```

OLSの理論的な信頼区間は、外れ値があると信頼性が低くなります。そこで登場するのが**ブートストラップ法**です。

4. ブートストラップ法の解説

手元のサンプルデータを「ミニチュアの母集団」と見なし、そこから何度もシミュレーション（リサンプリング）を行うことで、統計量のばらつきを経験的に調べる手法です。

アルゴリズム

1. **リサンプリング**: 元のデータセット（サイズ N）から、**重複を許して** N 個のデータをランダムに抽出し、「ブートストラップサンプル」を作成します。
2. **統計量の計算**: ブートストラップサンプルに対し、興味のある統計量（今回は $\hat{\beta}_1$）を計算します。
3. **繰り返し**: 上記のステップ1と2を、多数回（例：1000回）繰り返します。

最大の利点

正規分布などの強い仮定を必要としないため、外れ値などがあっても、より現実に即した頑健な（ロバストな）推定が可能になります。

5. ブートストラップ法による区間推定の実装

外れ値を含むデータセットに対してブートストラップ法を適用し、$\hat{\beta}_1$ の信頼区間を求めます。

```
# 1. 設定
n_iterations = 1000
n_size = len(X_outlier)
bootstrap_betas = []

# 2. ブートストラップ・ループ
for i in range(n_iterations):
    # a. リサンプリング
    indices = np.random.choice(range(n_size), size=n_size, replace=True)
    X_sample, y_sample = X_outlier[indices], y_outlier[indices]

    # b. モデル学習と係数保存
    X_sample_const = sm.add_constant(X_sample)
    model_boot = sm.OLS(y_sample, X_sample_const)
    results_boot = model_boot.fit()
    bootstrap_betas.append(results_boot.params[1])

# 3. 95%信頼区間の計算（パーセンタイル法）
alpha = 0.05
lower = np.percentile(bootstrap_betas, 100 * (alpha / 2.))
upper = np.percentile(bootstrap_betas, 100 * (1 - alpha / 2.))

print(f"Bootstrap 95% CI: [{lower:.4f}, {upper:.4f}]")
ci_outlier = results_outlier.conf_int().loc[1]
print(f"OLS Theoretical 95% CI: [{ci_outlier[0]:.4f}, {ci_outlier[1]:.4f}]")
```

6. ブートストラップによって得られた区間の解釈

コードを実行して得られた多数の係数（1000個）の分布が、「回帰係数のありえそうな値の範囲」をデータ自身が示したものです。

!(https://www.google.com/search?q=https://i.imgur.com/L4l3dE2.png)

この分布の下側2.5%点と上側2.5%点を結んだ区間が、**ブートストラップ95%信頼区間**です。

- **解釈**: 「もし実験を何度も繰り返せば、95%のケースで、この区間が『真の』係数を含むだろう」ということ。実用上は「真の係数がこの範囲にあると95%の信頼度で推定される」と考えてよい。
- **OLSとの比較**: 外れ値があるデータでは、ブートストラップ信頼区間はOLSの理論的信頼区間よりも**広く**なる傾向があります。これは、外れ値による係数の「ブレ」を正直に反映しているため、より現実的な評価を与えます。

まとめ

本講義では、以下の内容を学びました。

- 線形回帰の係数はサンプルデータに依存するため、常に**不確実性（ばらつき）**を伴います。
- OLS（最小二乗法）は**外れ値に非常に敏感**で、係数の推定値が大きく歪められる可能性があります。
- **ブートストラップ法**は、難しい理論的仮定に頼らず、データからの**リサンプリング**によって係数のばらつきを経験的に推定する強力な手法です。
- ブートストラップ信頼区間は、特にデータに外れ値があったり、誤差の正規性が疑わしい場合に、**より頑健な（ロバストな）推定**を与えてくれます。

データ分析では、係数を報告するだけでなく、その信頼性を評価することが極めて重要です。ブートストラップ法はそのための強力なツールの一つです。