

Instructions

- Homework 3 is due December 14th at 16:00 Chicago Time.
 - We will not accept any submissions past 16:00:00, even if they are only one second late.
 - You **must** upload the following files to the class Canvas:
 - LASTNAME_FIRSTNAME.pdf
 - LASTNAME_FIRSTNAME.ipynb
 - Your code notebook **must** be runnable using my environment outlines in class 1 (Python 3.14, and the `requirements.txt`).
 - You **must** use this template file and fill out your solutions for the written portion.
 - Please note that your last name and first name should match what you appear on Canvas as.
 - Include code snippets where required, as well as math and equations.
 - Be *concise* where possible, all of the homework problems can be answered in a few lines of math, code, and words.
-

Problem 1: Ridge Regression and Stability

Generate a dataset with $n = 100$ observations based on the true model:

$$y = 1 + x_1 + x_2 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 1)$.

Additionally, make $x_1, x_2 \sim \mathcal{N}(0, 1)$ highly correlated, with $\rho = 0.99999$. You can do this via the following code snippet:

```

1 import numpy as np
2 np.random.seed(1)
3
4 mean = [0, 0]
5 cov = [[1, 0.99999], [0.99999, 1]]
6 x1, x2 = np.random.multivariate_normal(mean, cov, 100).T
7 epsilon = np.random.normal(0, 1, 100)

```

Answer:

Problem 1.1: OLS Inference

Fit an OLS model to your generated data using `statsmodels`.

- Report the p-values and confidence intervals for $\hat{\beta}_1$ and $\hat{\beta}_2$.
- Discuss the results. Are the coefficients close to the true values (1 and 1)? Are they statistically significant?

Answer:

Problem 1.2: Stability

Now, re-generate your data using `np.random.seed(42)` and fit the OLS model again.

- Report the new estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$.
- Compare these estimates to the previous ones. What do you notice?

Answer:

Problem 1.3 Variance via Simulation

To better understand this phenomenon, run a simulation (note, you will need to unset the random seed for this part):

- Run a simulation loop 1000 times. In each iteration, generate a new dataset (x_1, x_2, ϵ) and fit an OLS model.
- Store the estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$.

Plot 2 histograms showing the distribution of the estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$. Additionally, report the volatility (standard deviation) of your $\hat{\beta}_1$ and $\hat{\beta}_2$ estimates across the simulations.

Answer:

Problem 1.3: Ridge to the Rescue

Now, fit a Ridge Regression model using `statsmodels` (you can use `fit_regularized` with `L1_wt=0` for Ridge).

- Use three different values for alpha (lambda): 0.01, 0.1, and 1.
- For each alpha, report the coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$.
- Discuss how the coefficients change with different alpha values.

Answer:

Problem 1.4: Ridge Simulation

Next, you should repeat the simulation from subproblem 1.3, but this time use a Ridge model with a fixed alpha of 0.01.

- Run the simulation loop 1000 times, generating new datasets and fitting the Ridge model each time.
- Store the estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$.

Plot 2 histograms showing the distribution of the Ridge estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$. Additionally, report the volatility (standard deviation) of your Ridge $\hat{\beta}_1$ and $\hat{\beta}_2$ estimates across the simulations. Compare these volatilities to those obtained from the OLS simulation in subproblem 1.3.

Answer:

Problem 2: Lasso Regression and Sparsity

This problem explores Lasso regression for a sparse model.

Generate a dataset with $n = 100$ observations and $p = 50$ features. The true model depends on only the first 3 features, while the remaining 47 features are irrelevant.

$$y = 5x_1 - 2x_2 + 3x_3 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 1)$ and all $x_j \sim \mathcal{N}(0, 1)$.

Use the following code snippet to generate your data:

```
1 import numpy as np
2 np.random.seed(42)
3
4 n_samples = 100
5 n_features = 50
6
7 X = np.random.normal(0, 1, (n_samples, n_features))
8 true_beta = np.zeros(n_features)
9 true_beta[:3] = [5, -2, 3]
10
11 y = np.dot(X, true_beta) + np.random.normal(0, 1, n_samples)
```

Problem 2.1: Naive OLS

Fit an OLS model using all 50 features.

- Look at the coefficients for the 47 “noise” features (x_4 through x_{50}). Are they exactly zero (you don’t need to report them all)?
- How many of these noise features have p-values < 0.05 (i.e., appear statistically significant purely by chance)?

Answer:

Problem 2.2: Lasso for Feature Selection

Now, fit a Lasso model using `statsmodels`. You can do this using `fit_regularized` with `L1_wt=1` (which specifies pure Lasso). Use an alpha of 0.1, 1.0, and 100.

- Report the estimated coefficients.
- How many coefficients are estimated to be exactly zero?
- Did the Lasso model correctly identify the 3 relevant features (x_1, x_2, x_3) while suppressing the noise?
- Which alpha value would you recommend?

Answer:

Problem 2.3: The Regularization Path

The sparsity of the model depends heavily on the strength of α .

- Create a list of alphas: [0.001, 0.01, 0.1, 0.5, 1.0, 5.0, 10, 100, 1000].

- Loop through these alphas, fitting a Lasso model for each.
- For each alpha, store the values of all coefficients.

Plot a graph with $\log(\alpha)$ on the x-axis and the coefficients on the y-axis (you can plot each coefficient as a separate line). Discuss how the coefficients change as alpha increases, and what the bias looks like (how far are the first 3 coefficients from their true values?).

Hint: It might be useful to plot the irrelevant coefficients in gray, and the relevant ones in different colors.

Answer:

Problem 3: Time Series and Time-Varying Heteroskedasticity

Generate a dataset with $n = 1000$ observations. The data follows an AR(1) process:

$$y_t = 0.5y_{t-1} + \epsilon_t$$

The error term ϵ_t follows an ARCH(1) process plus a regime:

$$\begin{aligned}\epsilon_t &= \sigma_t z_t, \quad z_t \sim \mathcal{N}(0, 1) \\ \sigma_t^2 &= 1 + 0.5\epsilon_{t-1}^2 + 10 \cdot \mathbb{I}_{\text{HighRegime}}\end{aligned}$$

Use the following code snippet to generate your data:

```

1 import numpy as np
2 import pandas as pd
3
4 np.random.seed(100)
5 n = 1000
6
7 indicator = np.zeros(n)
8 for i in range(0, n, 200):
9     indicator[i:i+100] = 1
10
11 epsilon = np.zeros(n)
12 sigma2 = np.zeros(n)
13 epsilon[0] = np.random.normal(0, 1)
14
15 for t in range(1, n):
16     sigma2[t] = 1 + 0.5 * (epsilon[t-1]**2) + 10 * indicator[t]
17     epsilon[t] = np.random.normal(0, np.sqrt(sigma2[t]))
18
19 y = np.zeros(n)
20 phi = 0.5
21 for t in range(1, n):
22     y[t] = phi * y[t-1] + epsilon[t]
```

Problem 3.1: The AR(1) Model

First, fit an AR(1) model to the generated series y_t .

$$y_t = \phi y_{t-1} + \epsilon_t$$

Extract the residuals ϵ_t from this model.

Create two plots:

- Residuals vs. fitted values.
- Residuals over time.

What do you notice about the first graph vs. the second graph?

Answer:

Problem 3.2: ACF

Plot the Autocorrelation Function (ACF) of the squared residuals $\hat{\epsilon}_t^2$.

What does the ACF plot suggest about the independence of the squared residuals?

Answer:

Problem 3.3: Modeling Variance (ARCH + Regime)

Now, we will model the variance of these residuals explicitly. We hypothesize that the variance σ_t^2 depends on the previous squared residual (ARCH effect) and the regime indicator.

- Construct the squared residuals $\hat{\epsilon}_t^2$.
- Construct a lagged feature $\hat{\epsilon}_{t-1}^2$.
- Run a linear regression of $\hat{\epsilon}_t^2$ on:

$$\sigma_t^2 = \beta_0 + \beta_1 \hat{\epsilon}_{t-1}^2 + \beta_2 \mathbb{I}_{\text{HighRegime}}$$

- Report the coefficients, compare them to the true values ($\beta_0 = 1$, $\beta_1 = 0.5$, $\beta_2 = 10$), and discuss their statistical significance.

Answer:

Problem 3.4: Regime-Dependent Forecasting

Suppose we are at the end of the series ($T = 1000$) and we want to predict the range of movement for the next step ($T = 1001$). Assume the last observed residual was $\hat{e}_{1000} = 2$.

Calculate the predicted variance $\hat{\sigma}_{1001}^2$ and the resulting 95% Confidence Interval width ($\pm 1.96\hat{\sigma}_{1001}$) for the error term under two scenarios:

- **Scenario A (Low Regime):** Assume the indicator for $T = 1001$ is 0.
- **Scenario B (High Regime):** Assume the indicator for $T = 1001$ is 1.

(Use the coefficients you estimated in 3.3).

What do you observe about your confidence intervals in the two scenarios?

Answer: