CS 240 - Lab 1

TAs: Malay & Sujan

Spring 2025

Welcome to the first graded lab of this course!! We expect it to be a relatively easy one :)

The exercises in this lab build on different aspects and applications of linear regression. They are arranged in increasing order of difficulty, and you are encouraged to attempt them sequentially. Throughout the lab, focus on understanding what is happening and why, and feel free to pause and ask questions rather than mechanically completing the tasks and moving on.

Instructions:

- This lab has 5 questions. Only the first 2 are graded. We have provided you with the following files:

```
lab1.pdf
q1/
  q1.py
  q1_outliers.csv
  q1_test.csv
  q1_train.csv
q2/
  q2.py
  q2_train.csv
q3/
  air_quality_data.csv
q4/
  gene_features.csv
  housing_features.csv
  sensor_features.csv
q5/
  q5_test.csv
  q5_train.csv
```

- Create a folder on your Desktop with the following directory structure. The folder must contain only the two specified files:

```
submission_<roll_number>/
|-- q1.py
|-- q2.py
```

- Once you have completed the work, open the terminal, type the following command, and then call a TA: `check-cs240`
- The TA will run `submit-cs240` to submit your work to the server. **It is your responsibility to ensure your work is submitted.** If you fail to inform us to submit your work, you will not be graded.

Q1. Simple Linear Regression

In this exercise, we study *linear regression* using the closed-form solution for Ordinary Least Squares (OLS) method. You are given a dataset consisting of feature vectors and real-valued targets:

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}, \qquad \mathbf{x}_i \in \mathbb{R}^d,\ y_i \in \mathbb{R}.$$

Let $X \in \mathbb{R}^{N \times d}$ denote the design matrix whose rows are the feature vectors $\mathbf{x}_i^\top$, and let $y \in \mathbb{R}^n$ denote the target vector. OLS fits a linear predictor of the form

$$\hat{y} = X\mathbf{w} + w_0\mathbf{1},$$

where $\beta \in \mathbb{R}^d$ is the weight vector, $\beta_0 \in \mathbb{R}$ is the intercept, by minimizing the squared error loss.

Throughout this lab, you must compute the OLS solution using the *closed-form expression*. The use of machine learning libraries (such as scikit-learn) for regression is not permitted.

Task 1 Standard OLS

Compute the OLS predictor using the training dataset.
Evaluate the following quantities on both the *training* and *test* datasets:

1. The mean squared error (MSE).
2. The correlation coefficient between y and $\hat{y}$, and its square.
3. The coefficient of determination (R^2).

Examine the values obtained before proceeding to the next section.

Task 2 Effect of Extreme Outliers

Append the provided CSV file containing extreme outlier samples to the train dataframe. Recompute the OLS solution and all evaluation metrics from Task 1.
Observe how the presence of outliers affects the learned parameters and the reported metrics.

Task 3 OLS without Intercept

Now consider linear regression *without* an intercept term, i.e.,

$$\hat{y} = X\mathbf{w}$$

Derive and implement the closed-form OLS estimator

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \|y - X\mathbf{w}\|_2^2.$$

Evaluate the same metrics as in Task 1 on the training and test datasets. Compare the results obtained with and without the intercept term, and notice the effect of forcing the regression model to pass through the origin.

Note: Once you are done with this question, you are recommended to take a 3-4 minute break and stare at your generated JSON file. Try to make sense of the metrics in the training and test sets. Try to compare the metrics across different tasks. Continue to the next section once you have understood these numbers.

Q2. Cross-Validation and Polynomial Features

A central challenge in machine learning is the **bias–variance trade-off**. As models become more expressive, they can better fit the training data, but this does not always lead to improved performance on unseen data. A model may achieve very low training error yet generalize poorly, a phenomenon known as overfitting. Since we do not have access to future data, we need a principled way to estimate generalization performance using only the given dataset. This motivates the use of **cross-validation**.

Polynomial Regression as a Model Family

Throughout this exercise, we assume a single-feature case. We begin with a simple setting where model complexity can be easily controlled.
For a single input variable x, a polynomial model of degree p is:

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_p x^p$$

Although this represents a non-linear function of x, it is still a linear model in the parameters.

In this exercise, the polynomial degree p will play the role of a tunable design choice. Our goal is to decide which degree to choose.

Note that when the polynomial degree satisfies $p \geq n - 1$, the model has enough capacity to interpolate all n training points, and the training MSE becomes zero, resulting in a perfect fit to the training data.

Task 1 Polynomial Feature Expansion and Fitting

Given a dataset with input features $X \in \mathbb{R}^{n \times 1}$, construct a transformed feature matrix $\Phi_p(X)$ containing polynomial terms up to degree p.

For each degree d:

1. Compute $\Phi_d(X)$,
2. Fit linear regression using least squares,
3. Compute the Mean Squared Error (MSE) on the training dataset.

Notice what happens to the training error as the degree increases.

Task 2 k-Fold Cross-Validation

We can estimate performance on unseen data by splitting the training data into a **training set** (to fit the model) and a **validation set** (to evaluate it). However, the result can depend on the particular split chosen. To reduce this dependence, we use **k-fold cross-validation**.

In k-fold cross-validation:

1. The dataset is divided into k equal parts (folds).
2. The model is trained k times. Each time, one fold is held out for validation and the remaining $k - 1$ folds are used for training.
3. The validation errors are averaged.

This produces a more reliable estimate of generalization error than a single train–validation split.

For each polynomial degree p in the specified range, perform k-fold cross-validation to compute the average validation MSE.

Task 3 Selecting the Polynomial Degree

After evaluating the cross-validation error for degrees $p \in \{1, 2, \ldots, D_{\max}\}$, plot:

- training MSE versus polynomial degree,
- cross-validation MSE versus polynomial degree.

Use these plots to identify the optimal polynomial degree.

Train a model of this optimal degree on the entire training dataset.

Once you have completed these steps, you are encouraged to visualize the fitted models by plotting the data and overlaying predictions from a low-degree model, the cross-validation–optimal model, and a high-degree model.

Submission Format

You must submit your code implementation only for both the questions. Your solution must use only `numpy` and `matplotlib` libraries; the use of machine learning libraries (such as `scikit-learn`) is not permitted.
In question 1, your code should generate a file named `metrics.json` containing all required evaluation metrics in the specified format. This file will be used for automated grading across all tasks of the lab. In question 2, we shall be grading you based on the final model returned by your code.

Ensure that your submission is fully reproducible: running the submitted code on the provided datasets should regenerate the required outputs without any manual intervention.

Graded Lab Ends :)

Regularization helps prevent overfitting by adding a penalty term to the model's objective function. This penalty discourages overly complex models that fit noise in the training data. Two common approaches are:

Ridge Regression (ℓ_2 regularization): $\min_{\mathbf{w}} \left[\sum_{i=1}^{N} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{d} w_j^2 \right]$

Lasso Regression (ℓ_1 regularization): $\min_{\mathbf{w}} \left[\sum_{i=1}^{N} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{d} |w_j| \right]$

At first glance, the objective functions for ridge and lasso regression appear very similar, differing only in whether the regularization penalty is based on the ℓ_2 norm or the ℓ_1 norm of the parameter vector. However, this seemingly small change leads to fundamentally different geometric and statistical behavior. Throughout these exercises, you should observe that ridge regression primarily controls overfitting by smoothly shrinking coefficients and stabilizing the solution. In contrast, lasso regression acts as a soft-thresholding mechanism, often driving some coefficients exactly to zero, and is therefore commonly used for feature selection.

The goal of this part of the lab is to develop an intuition for how these regularization techniques differ in practice, despite their superficial similarity in formulation.

In the following parts of this lab, we will use `scikit-learn` to implement these linear regression models. Unlike earlier sections, where you derived and implemented closed-form solutions, we will now rely on standard machine learning tools to fit these models.

Note that both ridge and lasso regression can be naturally formulated as optimization problems. In practice, these problems are typically solved using iterative, gradient-based optimization methods. The implementations provided by `scikit-learn` use such numerical optimization techniques internally. For the purposes of this lab, you may treat these procedures as *black-box optimizers*, and focus on understanding their behavior and outcomes rather than their internal implementation details.

scikit-learn components to explore. You are encouraged to familiarize yourself with the following classes and utilities from `scikit-learn`, and to consult their documentation while working on this part of the lab:

- `sklearn.linear_model.LinearRegression`
- `sklearn.linear_model.Ridge`
- `sklearn.linear_model.Lasso`
- `sklearn.linear_model.RidgeCV`
- `sklearn.linear_model.LassoCV`
- `sklearn.model_selection.train_test_split`
- `sklearn.model_selection.KFold`
- `sklearn.metrics.mean_squared_error`
- `sklearn.metrics.r2_score`
- `sklearn.preprocessing.StandardScaler`
- `sklearn.preprocessing.RobustScaler`
- `sklearn.pipeline.Pipeline`

Q3. The Effect of Normalisation

Environmental scientists often estimate air quality using a combination of pollutant measurements collected by different monitoring stations with distinct instruments and reporting conventions. These measurements have fundamentally different units and scales.

A research team is building a regression model to predict the Air Quality Index (AQI) based on multiple pollutant measurements. The model must be interpretable, since regulatory decisions depend on understanding which pollutants contribute most to poor air quality.
A disagreement arises within the team:

> **Researcher A:** "We are using regularized regression, so differences in measurement scale should not matter; the regularization will handle it."
>
> **Researcher B:** "Regularization only works fairly if all features are on comparable scales first."

You are tasked with resolving this disagreement using data-driven evidence.

You are given a dataset of environmental quality. Each data point corresponds to air quality measurements from a monitoring station at a specific time. Multiple pollutant measurements like `PM25`, `CO`, `NO2`, `O3` and `SO2` along with `Temperature` , `Humidity` and `Wind Speed` have been measured. The target variable is the `Air Quality Index (AQI)`, a continuous measure from 0–500 derived from health guidelines.

Task 1 Baseline Modeling (No Regularization)

1. Fit ordinary linear regression (OLS) on:
   - Raw features
   - Standardized features (z-score: $(x - \mu)/\sigma$ for each feature)
2. Compare the coefficients:
   (a) Which features have the largest magnitude coefficients in the raw model?
   (b) How do coefficient magnitudes change after standardization?
   (c) What happens to coefficient *signs* and *relative rankings*?

Task 2 Ridge Regression Without Normalization

1. Apply Ridge regression to the unnormalized training data. Sweep λ over $[0.01, 0.1, 1, 10, 100, 1000]$.
2. Plot **regularization paths**: Coefficient values vs. $\log_{10}(\lambda)$ for all features.
3. Observation questions:
   (a) As λ increases, which feature's coefficient shrinks to zero first?
   (b) Which feature's coefficient is most resistant to shrinkage?
   (c) Is there a relationship between a feature's scale and how quickly its coefficient shrinks? Based on your plot, what seems to determine shrinkage order?

Task 3 Ridge Regression on Normalised Data

1. Standardize all features using z-score normalization on the training set, then apply the same transformation to the test set. (Hint: You can use `StandardScalar` to do so)
2. Repeat Ridge regression using the same λ values as Task 2. Plot new regularization paths.
3. Observation questions:
   (a) Visually compare with Task 2's plot: Do shrinkage patterns change? Do coefficients now shrink at more similar rates?
   (b) At $\lambda = 10$, which features still have substantial coefficients? Has this changed?

Task 4 Lasso on Raw Data

1. Apply Lasso regression to **raw training data** with λ sweep, and plot Lasso paths (coefficients vs. λ).
2. Create a "feature entry table" showing at what λ each feature first gets a non-zero coefficient.
3. Observation Questions:
   (a) Which features enter the model first (at largest λ)?
   (b) Which features are never selected (always zero)?

Task 5 Lasso on Normalized Data

1. Apply Lasso to **standardized data** with same λ range and plot normalized Lasso paths.
2. Observation Questions:
   (a) Does the feature selection order change dramatically?
   (b) Which features are now consistently selected? Which are consistently dropped?

Task 6 Impact of Instrument Spikes on Regularization

1. Characterize the outliers in the data:
   (a) Identify the samples where PM2.5 $> 200\mu g/m^3$ and the samples where CO > 4 ppm.
   (b) Check overlap; Are these same sensors?
2. Create two modified datasets:
   - *Dataset A:* Remove outliers and normalize.
   - *Dataset B:* Keep all data but use robust scaling (median/IQR instead of mean/std)
3. Repeat Ridge regression on Dataset A and B.
4. **Observation questions**:
   (a) How much do coefficients change between these three approaches?
   (b) Which approach seems most stable?

Task 7 Questions for Thought

- Ridge regression penalizes coefficient magnitude uniformly. Why does this penalty become implicitly *non-uniform* when features are on different scales?
- You found very different optimal λ values for raw vs. normalized data. Could you theoretically find some λ_{raw} that gives the same solution as some λ_{norm} on normalized data?
- Ridge regression can be derived from a Bayesian perspective by placing a Gaussian prior $\beta_j \sim \mathcal{N}(0, \tau^2)$ on each coefficient.
  (a) In this framework, what does the τ^2 parameter represent, and how does it relate to λ?
  (b) When features have different scales (like PM2.5 vs CO), what implicit prior assumption are we making if we use the same τ^2 for all coefficients without normalization?
  (c) After normalization, what does a shared τ^2 (or λ) for all coefficients imply about our prior beliefs?
- In a policy-facing, interpretable model, which coefficient interpretation is more meaningful: raw-scale coefficients or normalized coefficients? Why?
- In your Lasso experiments on raw data, which type of features (large-scale like PM2.5 or small-scale like CO) tended to be selected first? How does this create misleading conclusions about which pollutants are "most important" for air quality?
- Instrument failures introduce extreme but rare values.
  1. How do such outliers interact with regularization penalties?
  2. Why might robust scaling outperform simple outlier removal in some cases?
- Although normalization is generally recommended before regularization, are there circumstances under which skipping normalization is a principled modeling choice? Identify such scenarios and justify them using domain knowledge or statistical reasoning.

Q4. Deconstructing Linear Models

In this section, you are provided with three datasets (`housing_features.csv`, `gene_features.csv`, and `sensor_features.csv`). Each represents a distinct data modeling challenge. Your task is to diagnose model behavior by examining learned coefficients and error metrics across different regularization techniques.

Task 1 Experimental Protocol

For each dataset, complete the following:

1. Generate a correlation heatmap to identify potential collinearity or structural patterns.
2. Train OLS, Ridge, and Lasso models across a logarithmic grid of λ values (e.g., $\lambda \in \{0, 0.003, 0.01, \ldots, 30, 100\}$
3. Record the Test MSE, Test R^2, and the full coefficient vector β for each configuration.

Task 2 Case Studies

Answer the following questions using specific evidence (MSE values, coefficient magnitudes) from your experiments.

Case A: Back to the ordinary (`housing_features.csv`)

Context: Modeling housing prices with a data consisting of 5 features.

Analysis Questions:

1. **Baseline Comparison:** Compare the Test MSE of unregularized Linear Regression ($\lambda = 0$) against the optimal Ridge and Lasso models.
   - Do the results differ statistically?
   - Explain why adding model complexity (regularization) fails to improve performance in this specific regime. (Hint: Consider the dataset size and the dimension size).
2. **Coefficient Stability:** Inspect the Lasso coefficients at a high penalty (e.g., $\lambda = 10$).
   - Which feature retains a non-zero coefficient while others are eliminated?
   - Compare the OLS coefficients of `feature_3` and `feature_5`. If both are significant in the baseline, why does Lasso penalize `feature_3` to zero faster than `feature_5`?
3. **Stress Testing:** Perturb the conditions that favor OLS to observe when regularization becomes necessary:
   - Artificially increase the noise in the target variable y. How does the performance gap between OLS and Ridge/Lasso change?
   - Re-train models using a subset of the training data (e.g., $N = 30$) but evaluate on the full test set.
   - Describe the degradation of OLS Test MSE in the small-sample limit. Why do regularized models remain more stable?

Case B: The hidden gene ? (`gene_features.csv`)

Context: Analyzing gene expression data ($p = 24$) for a rare condition ($N = 80$). Most genes are irrelevant or part of metabolic pathways (highly correlated but non-causal), with only a few true "driver" genes.

Analysis Questions:

1. **Collinearity Detection:** Your correlation analysis should reveal a block of features (Indices 4–10) that are highly correlated with each other.
2. **Regularization Behavior:** Compare the coefficients for this block in the best Ridge model versus the best Lasso model.
   - Does Ridge set them to zero, or distribute small weights across the group?
   - Does Lasso successfully identify them as irrelevant (zero magnitude)?
3. **Optimal Model Selection:** Assume the Lasso model achieves the lowest Test MSE at $\lambda = \lambda_0$.
4. **Coefficient Inspection:** Examine the coefficients at $\lambda = \lambda_0$.
   - Note the weights of the "True Genes" versus noise genes.
   - *Analysis:* Why does the model retain some noise rather than increasing λ (e.g., to 1.0) to remove it? (Hint: Compare the MSE and coefficient magnitude of the true genes at $\lambda = 0.3$ vs $\lambda = 1.0$. What is the cost of over-regularization?)

5. **Post-Selection Inference:** A domain expert notes that while Lasso selected the right features, the coefficients for the main genes appear "shrunk" (biased downwards).
   (a) How does Lasso's bias affect the interpretation of the "impact" of the top genes?
   (b) Propose a two-stage method using the *output* of the Lasso model (non-zero indices) as the *input* for an unregularized OLS model.
   (c) Execute this strategy. Report the Test MSE and R^2.
      - Did R^2 improve compared to the pure Lasso model?
      - Did this process filter out residual noise variables?
6. **Model Interpretability:** If you used Ridge instead of Lasso, would it be possible to confidently select 3 target genes? Compare the sparsity of the best Ridge vector vs. the best Lasso vector.

Case C: Imperfect Clones ? (`sensor_features.csv`)

Context: Monitoring an industrial reactor using 10 sensors.

- **Cluster A (Sensors 1–4):** Redundant sensors at intake valve Alpha.
- **Cluster B (Sensors 5–8):** Redundant sensors at exhaust valve Beta.
- **External (Sensors 9–10):** Sensors on the outer casing .

The "True" flux is a stable function of the valve sensors, but individual sensors are noisy.

Analysis Questions:

1. **Performance Assessment:** Identify the model and λ achieving the lowest Test MSE. Quantify the improvement over the OLS baseline.
2. **Cluster Analysis:** Focus on **Cluster A** (Sensors 1–4). Theoretically, these measure the same input and should have equal importance.
   - **OLS ($\lambda = 0$):** Inspect the coefficients. Calculate their standard deviation. Are they stable or volatile?
   - **Ridge ($\lambda = 30$):** Calculate the standard deviation of the coefficients for Sensors 1–4. Does Ridge enforce a grouping effect?
   - **Lasso ($\lambda = 1$):** Does Lasso select all 4 sensors or a subset? Does it average them as effectively as Ridge?
3. **Noise Rejection:** Inspect coefficients for **Sensors 9 and 10** in the best Lasso and Ridge models.
   - Which model better identified these sensors as noise (zero coefficients)?
   - *Analysis:* If Lasso removed the noise but Ridge achieved a lower overall MSE, explain the trade-off. (Hint: Weigh the benefit of noise removal against the benefit of averaging redundant signals in Cluster A).
4. **Hybrid Pipeline:** Lasso excels at selection but fails at averaging. Ridge excels at averaging but fails at selection.
   (a) Design a two-step pipeline:
      - **Step 1 (Selection):** Use Lasso to identify noise sensors (coefficient ≈ 0).

- **Step 2 (Estimation):** Remove noise sensors and train a **Ridge** model on the remaining features.

(b) Implement this strategy and report the Test MSE.

(c) Does this hybrid approach outperform the previous best Ridge model?

Q5. Regularization Analysis

Context: You are analyzing a signal from a legacy sensor array known to be malfunctioning. The "True Control Signal" is latent (unobservable), and the provided features are a mixture of broken sensors, echoes, and corrupted proxies.
Objective: Separate signal from noise by deducing feature identity through their reaction to regularization.

Task 1 Data Collection

1. Train Lasso and Ridge models across a logarithmic range of λ (e.g., 10^{-4} to 10^{3}).
2. Plot the "Coefficient Path" for both models.
3. Generate a correlation heatmap and scatter plots of features against the target y.

Task 2 Forensic Analysis

Answer the following questions based on the observed coefficient paths.

1. **Feature Elimination**
   Inspect the Lasso coefficient path. Identify the feature driven to zero immediately, even at low penalties ($\lambda \approx 0.1$).
   - What does this suggest about the information content of this feature?
   - Confirm by inspecting the scatter plot of this feature against y. Is it signal or noise?
2. **Correlated Features**
   Focus on the Ridge coefficient path. Identify a pair of features that exhibit "grouping" behavior: starting with different magnitudes at $\lambda = 0$ (OLS) but converging as λ increases.
   - Which two features exhibit this behavior?
   - Why does OLS assign them different weights while regularization treats them similarly? What does this imply about redundancy?
   - Calculate the correlation coefficient between these two features.
3. **Reliability vs. Availability**
   Two features measure the same phenomenon but have different error profiles.
   - **Sensor A** is always active but noisy (High Variance). **Sensor B** is precise when active but has frequent dropouts (Intermittent).
   - Scatter plot both features against each other.
   - Relying solely on Sensor B fails due to gaps; relying solely on Sensor A introduces noise.
4. **Independent Signals**
   Examine the behavior of a ceratain feature across the Ridge coefficient path. Let this feature be z.

- While other coefficients shrink or converge, `z` maintains its magnitude. Does this persistence hold in the Lasso path?
- What does this stability suggest about the relationship between `z` and other features? (Hint: Consider how regularization treats features with no substitutes).
- Check the correlation matrix. Does `z` correlate significantly with any other feature?
- In the Lasso path, how long does `z` survive compared to other features?

5. **Structural Corruption vs. Noise**
   Two remaining features correlate with y but show distinct structural differences.
   - Plot both features against y. Note the difference in scatter patterns (e.g., uniform noise vs. outliers).
   - Use LASSO to adjudicate. Include both features and observe which one is penalized to zero first.
   - What does LASSO's preference reveal about the stability of these features? (Hint: Consider the sensitivity of squared error to extreme outliers).

6. **Final Model Construction**
   Synthesize your findings to build a robust predictive model.
   - Select features and regularization methods based on the insights above.
   - Apply the model to the `q5_test.csv` dataset.
   - Upload predictions to the competition leaderboard:
     `https://www.kaggle.com/t/c6f6a79536024ef0b9f1075ba68aa280`
     Leaderboard standing serves as verification.