CSC 6022 Assignment 1

Homework Due Date: **Oct 10, 2025**

Instructions (Please read carefully)

- Submit your answers as an electronic copy only in **pdf** format on Blackboard.
- No late submissions will be accepted. Zero credit will be assigned for late submissions. Email requests for late submission will not be replied.
- Please type in Latex or provide handwritten submissions scanned to **pdf**.
- Explicitly mention your collaborators if any. Collaborations should be limited to discussing and learning from each other but please do your own work and write your own codes. We will actively monitor any attempt to copy solutions from each other or from the internet.
- The full score of this homework is 125 pts.

1 Multi-output Linear Regression [10 pts]

Consider a linear regression model with a 2-dimensional response vector $\boldsymbol{y}_i \in \mathbb{R}^2$ and a binary input $x_i \in \{0, 1\}$. The training data is:

x	y
0	$(-1, -1)^\top$
0	$(-1, -2)^\top$
0	$(-2, -1)^\top$
1	$(1, 1)^\top$
1	$(1, 2)^\top$
1	$(2, 1)^\top$

Model setup:

- Embed each input into 2-dimensions using
$$\phi(0) = (1, 0)^\top, \quad \phi(1) = (0, 1)^\top.$$
- Linear model:
$$\hat{\boldsymbol{y}} = \mathbf{W}^\top \phi(x), \quad \mathbf{W} \in \mathbb{R}^{2\times 2}.$$
- Probabilistic assumption (needed for MLE):
$$\boldsymbol{y}_i \mid x_i \sim \mathcal{N}\big(\mathbf{W}^\top \phi(x_i), \sigma^2 I_2\big).$$

Please compute the MLE for W from the above data and show that the MLE here coincides with the ordinary least squares estimator.

2 Simple Linear Regression — MLE, MAP, and Bayesian Inference [15 pts]

Consider real-valued variables X and Y. The Y variable is generated, conditional on X, from the following process:

$$\epsilon \sim \mathcal{N}(0, \sigma^2), \qquad Y = aX + \epsilon$$

where every ϵ is an independent noise term, drawn from a Gaussian distribution with mean 0 and variance σ^2. This is a one-feature linear regression model, where a is the only weight parameter. The conditional distribution of Y is

$$p(Y \mid X, a) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(Y - aX)^2\right).$$

The following questions are about this model.

1. **MLE.** Derive the maximum likelihood estimate of the parameter a in terms of the training examples $\{X_i, Y_i\}_{i=1}^n$. (*Hint: take the derivative of the log-likelihood with respect to a.*)
2. **MAP (Closed Form).** Assume $\sigma = 1$, and a Gaussian prior $a \sim \mathcal{N}(0, \lambda^2)$. Solve for

   $$\hat{a}^{\text{MAP}} = \arg\max_a \Big[\ln p(Y_1, \ldots, Y_n \mid X_1, \ldots, X_n, a) + \ln p(a \mid \lambda)\Big].$$

   (*Hint: this would become the MLE with an additional ridge regularization term.*)
3. **Qualitative Comparison.** Under the following conditions, how do the prior and likelihood curves change? Do a^{MLE} and a^{MAP} become closer or further apart? (*Hint: Recall that the variance of a Gaussian controls its width. Think about which term dominates the posterior in each case.*)

	$p(a \mid \lambda)$ prior probability: wider, narrower, or same?	$p(Y_1, \ldots, Y_n \mid X_1, \ldots, X_n, a)$ conditional likelihood: wider, narrower, or same?	$\lvert a^{\text{MLE}} - a^{\text{MAP}} \rvert$ increase or decrease?
As $\lambda \to \infty$			
As $\lambda \to 0$			
More data: as $n \to \infty$ (fixed λ)			

3 Solving LASSO [10 pts]

Consider the following model where X is a matrix of features, and Y is a vector of observed responses. The relationship between Y and X is assumed to be linear, and the noise term follows a Gaussian distribution:

$$Y = X\beta + \epsilon, \quad \epsilon \sim N\left(0, \sigma^2 I\right)$$

Now, let's impose a **Laplace prior** on the weights β in the Bayesian framework:

$$p(\beta \mid \lambda) = \prod_{j=1}^{p} \frac{\lambda}{2} \exp\left(-\lambda \lvert\beta_j\rvert\right)$$

where λ is a positive regularization parameter controlling the strength of the prior (and thus the sparsity of the solution). The likelihood function is given by:

$$p(Y \mid X, \beta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(Y_i - X_i\beta)^2}{2\sigma^2}\right)$$

1. Please show that the MAP estimation of β is equivalent to solving the LASSO optimization problem. Please provide step-by-step derivations.

2. Consider LASSO regression, where we optimize:

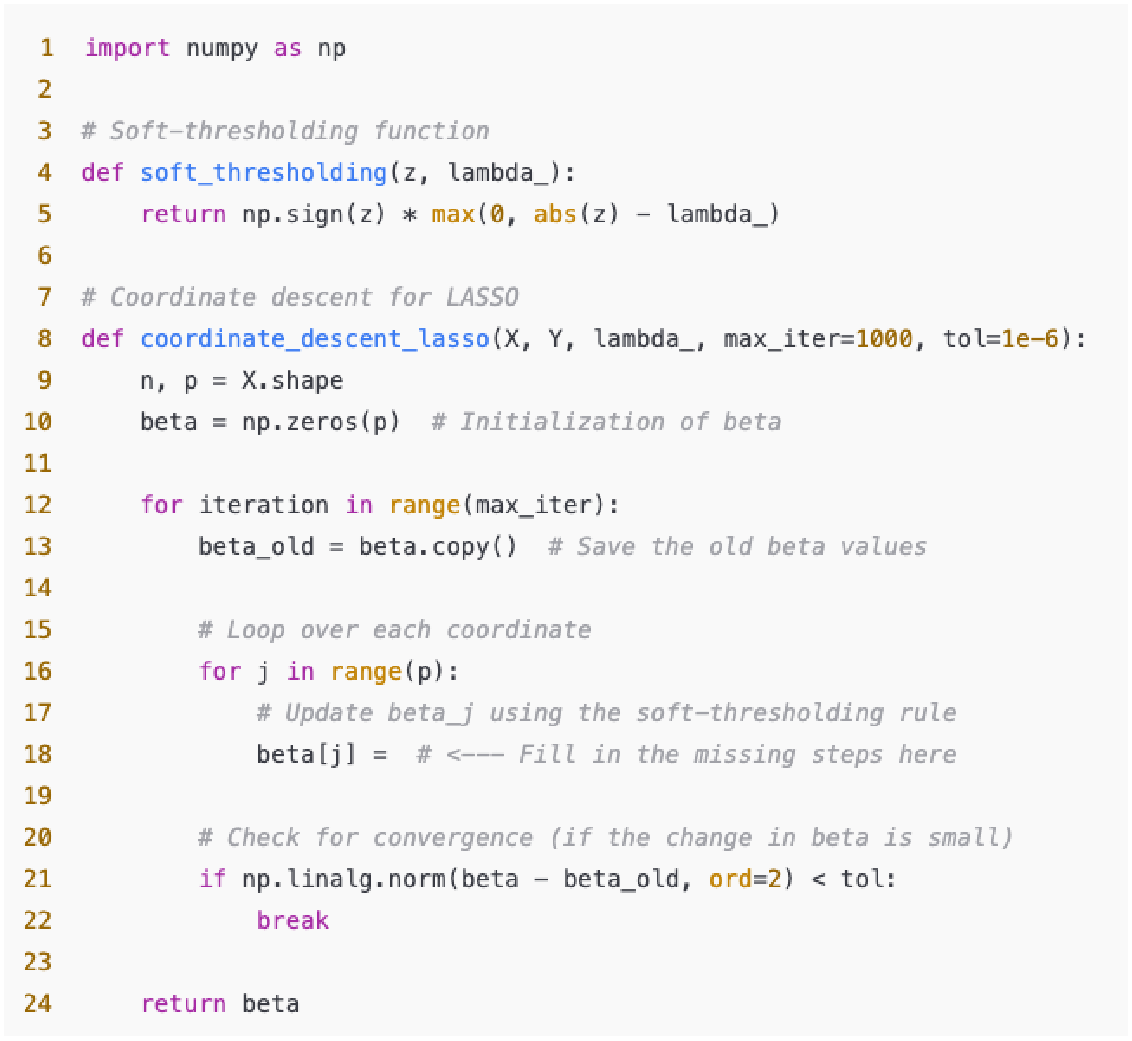
$$\min_{\beta}\left(\frac{1}{2n}\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1\right)$$

Please derive the subgradient update rule for LASSO. Specifically, derive and present the update rule in the following form:

$$\beta^{(k+1)} = \beta^{(k)} - \alpha_k \cdot g$$

In your derivation, specify the expression for the subgradient g.

3. Using the above subgradient descent method may lead to slow convergence due to non-smoothness. More often, coordinate descent is used to solve LASSO as it efficiently exploits sparsity. Please review the following code to understand how coordinate descent works. You will need to:
(1). Specify the condition under which β_j, for $j = 1, \ldots, p$, will be zero.
(2). Exploit this condition to fill in the missing part of the code at line 18. Note that you do not need to use Python code; instead, provide the mathematical formula for updating β_j. Specifically, express the update rule for β_j using the soft-thresholding rule.
(3). Does a larger or smaller λ lead to sparse solutions? Briefly explain why.



```
1   import numpy as np
2
3   # Soft-thresholding function
4   def soft_thresholding(z, lambda_):
5       return np.sign(z) * max(0, abs(z) - lambda_)
6
7   # Coordinate descent for LASSO
8   def coordinate_descent_lasso(X, Y, lambda_, max_iter=1000, tol=1e-6):
9       n, p = X.shape
10      beta = np.zeros(p)  # Initialization of beta
11
12      for iteration in range(max_iter):
13          beta_old = beta.copy()  # Save the old beta values
14
15          # Loop over each coordinate
16          for j in range(p):
17              # Update beta_j using the soft-thresholding rule
18              beta[j] =  # <--- Fill in the missing steps here
19
20          # Check for convergence (if the change in beta is small)
21          if np.linalg.norm(beta - beta_old, ord=2) < tol:
22              break
23
24      return beta
```

Figure 1: Python Codes: Coordinate descent for LASSO

4 Shrinkage in Linear Regression [10 pts]

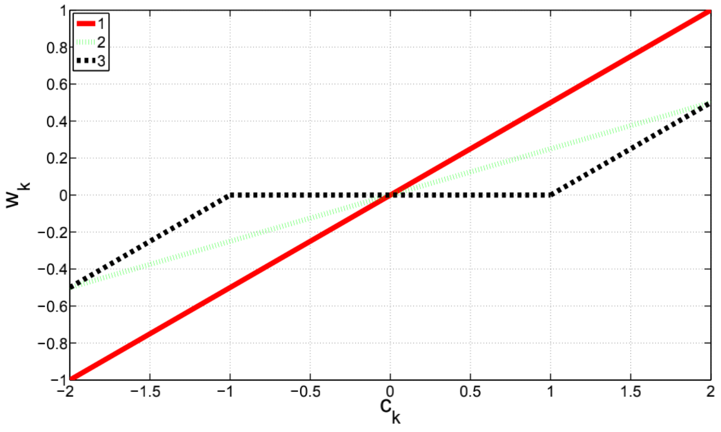


Figure 2: The plots of $\hat{w}_k$ vs c_k

Consider linear regression with an **orthonormal design matrix**, i.e.,

$$\|\boldsymbol{x}_{:,k}\|_2^2 = 1 \quad \text{for each feature } k, \quad \boldsymbol{x}_{:,k}^T \boldsymbol{x}_{:,j} = 0 \;\; \text{for } j \neq k.$$

In this setting, each regression coefficient w_k can be estimated independently.

Figure 2 plots the estimator $\hat{w}_k$ against

$$c_k = 2\,\boldsymbol{y}^T \boldsymbol{x}_{:,k},$$

the correlation of feature k with the response, under three different estimation methods:

- Ordinary Least Squares (OLS),
- Ridge Regression with penalty/regularization parameter λ_2,
- LASSO with penalty/regularization parameter λ_1.

1. The figure contains three curves: solid (1), dotted (2), and dashed (3). Identify which curve corresponds to OLS, ridge regression, and LASSO. Justify your reasoning based on the functional form of the estimators.
2. From the plot, determine the value of the LASSO regularization parameter λ_1.
3. From the plot, determine the value of the ridge regularization parameter λ_2.

5 Naive Bayes Classifier and MLE [10 pts]

You are given a multi-class classification problem with a dataset of N i.i.d. samples:

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

We use a Naive Bayes model with Gaussian class-conditionals and conditional independence:

$$p(\mathbf{x} \mid y = k) = \prod_{j=1}^{d} \mathcal{N}(x_j \mid \mu_{k,j}, \sigma_{k,j}^2).$$

1. **Model Assumptions:** Why can the density factorize as above?

2. **MLE Derivation:** Define the model parameters as:
   - Class priors: $\pi_k = P(y = k)$
   - Class-conditional means and variances: $\mu_{k,j}$ and $\sigma^2_{k,j}$ for $k = 1, \dots, K$ and $j = 1, \dots, d$

   (i) Write the likelihood $\mathcal{L}(\Theta)$ for all N samples. (*Hint: Separate terms for* π_{y_i} *and Gaussian densities.*)

   (ii) Take logs to get $\log \mathcal{L}(\Theta)$. (*Hint: Summation over* i*, then over features* j*.*)

   (iii) Derive π_k. (*Hint: Count how often each class* k *appears in the data.*)

   (iv) Derive $\mu_{k,j}$ and $\sigma^2_{k,j}$. (*Hint: Think about sample mean and variance, restricted to class-*k *samples.*)

3. **Modified Assumption — Shared Variances:** Now suppose all classes share the same variance for feature j:
$$p(\mathbf{x} \mid y = k) = \prod_{j=1}^{d} \mathcal{N}(x_j \mid \mu_{k,j}, \sigma_j^2).$$

   (i) How does this change the likelihood?

   (ii) What are the new MLEs for $\pi_k, \mu_{k,j}, \sigma_j^2$? (*Hint:* σ_j^2 *now uses* all *samples for feature* j*, not class-specific.*)

4. **Decision Boundary:** Compare the boundaries with and without shared variances. (*Hint: Look at the log-likelihood ratio — what happens to the quadratic terms if variances are shared?*)

6 Logistic Regression [10 pts]

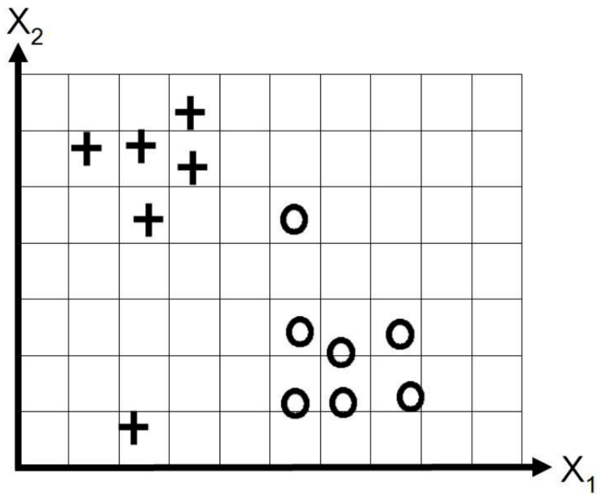


Figure 3: Data for the logistic regression question.

1. Consider the data in Figure 3 (a), where we fit the model
$$p(y = 1 \mid \boldsymbol{x}, \boldsymbol{w}) = \sigma\left(w_0 + w_1 x_1 + w_2 x_2\right).$$
Suppose we fit the model by maximum likelihood, i.e., we minimize
$$J(\boldsymbol{w}) = -\ell\left(\boldsymbol{w}, \mathcal{D}_{\text{train}}\right)$$
where $\ell\left(\boldsymbol{w}, \mathcal{D}_{\text{train}}\right)$ is the log likelihood on the training set. Sketch a possible decision boundary corresponding to $\hat{\boldsymbol{w}}$. (Copy the figure first (a rough sketch is enough), and then superimpose your

answer on your copy, since you will need multiple versions of this figure). Is your answer (decision boundary) unique? How many classification errors does your method make on the training set?

2. Now suppose we regularize only the w_0 parameter, i.e., we minimize

$$J_0(\boldsymbol{w}) = -\ell\left(\boldsymbol{w}, \mathcal{D}_{\text{train}}\right) + \lambda w_0^2$$

Suppose λ is a very large number, so we regularize w_0 all the way to 0, but all other parameters are unregularized. Sketch a possible decision boundary. How many classification errors does your method make on the training set? Hint: consider the behavior of simple linear regression, $w_0 + w_1 x_1 + w_2 x_2$ when $x_1 = x_2 = 0$.

3. Now suppose we heavily regularize only the w_1 parameter, i.e., we minimize

$$J_1(\boldsymbol{w}) = -\ell\left(\boldsymbol{w}, \mathcal{D}_{\text{train}}\right) + \lambda w_1^2$$

Sketch a possible decision boundary. How many classification errors does your method make on the training set?

4. Now suppose we heavily regularize only the w_2 parameter. Sketch a possible decision boundary. How many classification errors does your method make on the training set?

7 Logistic Regression vs. LDA/QDA [10 pts]

We train the following binary classifiers by *maximum likelihood* on a training set $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $y_i \in \{0, 1\}$.

1. **GaussI (shared spherical covariances):** A generative classifier with class-conditional densities

$$p(\boldsymbol{x} \mid y = c) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_c, \mathbf{I}),$$

and a uniform class prior $p(y)$.

2. **GaussX (class-specific covariances):** A generative classifier with

$$p(\boldsymbol{x} \mid y = c) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c),$$

with unconstrained $\boldsymbol{\Sigma}_c \succ 0$.

3. **LinLog (discriminative, linear):** Logistic regression with linear features,

$$p(y = 1 \mid \boldsymbol{x}) = \sigma(\beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x}),$$

where $\sigma(t) = 1/(1 + e^{-t})$.

4. **QuadLog (discriminative, quadratic):** Logistic regression with linear and quadratic features (all monomials up to degree 2).

After training each model M (obtaining its MLE parameters $\hat{\boldsymbol{\theta}}_M$), define the *average conditional log-likelihood* on the training set:

$$L(M) \;=\; \frac{1}{n}\sum_{i=1}^{n} \log p\Big(y_i \mid \boldsymbol{x}_i,\ \hat{\boldsymbol{\theta}}_M,\ M\Big).$$

(*Note:* This is the conditional log-likelihood $\log p(y \mid \boldsymbol{x})$, not the joint $\log p(y, \boldsymbol{x})$.)

We will write $L(M) \le L(M')$ if, for *every* possible training set, model M attains a (training) conditional log-likelihood no larger than that of M'.

For each pair below, state whether $L(M) \le L(M')$, $L(M) \ge L(M')$, or *no universal ordering* can be made (i.e., M can be better or worse depending on the dataset). Give a brief (1–2 sentence) justification.

1. **GaussI vs. LinLog.** *Hint: Derive the posterior $p(y \mid \boldsymbol{x})$ for GaussI; what functional form (in $\boldsymbol{x}$) do you get? Which model family directly* maximizes *the conditional log-likelihood?*
2. **GaussX vs. QuadLog.** *Hint: GaussX induces quadratic* log-odds/*decision boundaries. How does that compare to logistic regression with quadratic features?*
3. **LinLog vs. QuadLog.** *Hint: Is every linear decision boundary a special case of a quadratic one? Think model* nesting *and expressivity.*
4. **GaussI vs. QuadLog.** *Hint: Combine your conclusions from the above comparisons.*
5. Now define the *training misclassification rate*:

$$R(M) \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{y_i \neq \hat{y}_M(\boldsymbol{x}_i)\}\,, \quad \hat{y}_M(\boldsymbol{x}) = \mathbf{1}\big\{p_M(y{=}1 \mid \boldsymbol{x}) \geq \tfrac{1}{2}\big\}\,.$$

   Is it true in general that $L(M) > L(M')$ implies $R(M) < R(M')$? Explain why or why not. *Hint:* The log-likelihood evaluates the *probability assignments*; the 0–1 error depends only on the *argmax* (thresholded label). Can you find situations where probabilities improve (are better calibrated) but the predicted labels do not change, or even get worse?

8 MLE and Model Optimization: Gradient Descent and Newton's Method [15 pts]

For each user $i = 1, \ldots, N$, a large language model (LLM) presents a candidate set $\mathcal{C}_i$ of $J_i \geq 2$ possible responses. Each candidate $j \in \mathcal{C}_i$ has a feature vector $\boldsymbol{x}_{ij} \in \mathbb{R}^d$ (e.g., text embedding, length, safety/helpfulness score). The observed choice $y_i \in \mathcal{C}_i$ is the response the user prefers (e.g., clicks on or upvotes). Thus, the training data is $\big(\{\boldsymbol{x}_{ij}\}_{j\in\mathcal{C}_i}, \mathcal{C}_i, y_i\big)_{i=1}^{N}$.

Model: Multinomial Logit (MNL)

$$P(y_i = j \mid \mathcal{C}_i, \{\boldsymbol{x}_{ij}\}_{j\in\mathcal{C}_i}) = \frac{\exp(\boldsymbol{\beta}^\top \boldsymbol{x}_{ij})}{\sum_{\ell\in\mathcal{C}_i} \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_{i\ell})}.$$

Questions:

1. Derive the log-likelihood $\ell(\boldsymbol{\beta})$ and show that

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{N} \Big(\boldsymbol{\beta}^\top \boldsymbol{x}_{iy_i} - \log \sum_{\ell\in\mathcal{C}_i} \exp(\boldsymbol{\beta}^\top \boldsymbol{x}_{i\ell})\Big).$$

2. Derive the gradient $\nabla\ell(\boldsymbol{\beta})$. Give the update rules for:
   - **Gradient Descent (GD)** with learning rate η.
   - **Newton's Method** using the Hessian $H(\boldsymbol{\beta})$.
3. **Discussion:** Compare GD vs. Newton:
   - Which method has tunable parameters (e.g., learning rate)?
   - Which one typically converges faster, and why?

This model is closely related to preference learning for LLM fine-tuning*, where the LLM generates multiple responses and we learn from user feedback which ones are preferred. Understanding the optimization methods here helps connect classical ML with modern alignment techniques like RLHF.*

9 Computing the Posterior Predictive [10 pts]

You first need to generate some 1D data by yourself as follows. Sample 20 independent pair samples (x_i, y_i) from

$$y = sin(x) + \epsilon, \quad x \in [0, 2\pi] \tag{1}$$

$$\epsilon \sim \mathcal{N}(0, 1) \tag{2}$$

You can sample $\{x_i\}$ uniformly from $[0, 2\pi]$ and inject iid noise to obtain your 1d data.

We aim to fit a Bayesian polynomial regression model to these 1d data. Suppose you have a 1d regression model of the form

$$f(x; \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$$

We will use a Gaussian prior:

$$p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \tau^2 \mathbf{I})$$

and write the likelihood as follows:

$$p\left(\mathcal{D} \mid \boldsymbol{w}, \sigma^2\right) = \prod_{n=1}^{N} p\left(y_n \mid \boldsymbol{w}^\top \phi(\boldsymbol{x}), \sigma^2\right)$$

You are asked to compute the posterior predictive at some grid points in range $[0, 2\pi]$. Consider both the Plugin approximation and the Posterior predictive. Try to illustrate your results using plots like Figure 11.21 (a) - (d) in our textbook. Implement by Python by yourself.

10 Programming I [25 pts]

You may choose either Programming I or Programming II to complete. It is not necessary to do both.

10.1 Ridge Regression

10.1.1 Input Data

You are given three files:

- "train-matrix.txt" contains the training set
- "test-matrix.txt" contains the testing set
- "true-beta.txt" contains the true regression coefficient vector β^*

The format of the "·-matrix.txt" files is as follows:

```
columns-number
rows-number
x11 x12 ... x1n
...
xm1 xm2 ... xmn
y1
y2
...
ym
```

where `volumns-number` is the number of features and the `rows-number` is the number of samples and each x or y value is a float in ASCII. The matrix elements x are the design matrix. y, is the response vector.

The "true-beta.txt" file is formatted as follows:

```
rows-number
b1
b2
...
bn
```

where `rows-number` is the number of features and `b` is β^*. Similarly, `rows-number` is an integer and `b` is a float represented in ASCII.

10.1.2 Submission format

- code.zip (without Input Data included) - **10 pts - you will receive points only if your code runs correctly without errors and finishes within 15 mins**.
- submit a report containing outputs for questions below. This should be included as part of the written assignment. - **10 pts**.
- **Note: Please include all results from your model in the report. You will receive no credits if we have to run the code to get output for Q2 and Q3 below.**

10.1.3 Questions

Given a candidate model with β, the prediction error on a testing set is defined as

$$\frac{1}{m}\|\overline{y} - \overline{X}\beta\|_2^2,$$

where $\overline{y}$ is the response vector of the testing set, $\overline{X}$ is the design matrix of the testing set, and m is the number of samples in the testing set.

1. Please implement the following greedy algorithm, and output the result for X, and Y in "train-matrix.txt" with $K = 6$:

```
Input: $X = [X_{*1}, ..., X_{*d}] \in \mathbb{R}^{n\times d}$, $y \in \mathbb{R}^n$, K
Output: $\mathcal{A}^{(k)}$ and $\beta^{(k)} \in \mathbb{R}^d$
Initialize: $\mathcal{A}^{(0)} = \emptyset$ and $\beta^{(0)} = 0$
for $k = 1, 2, ..., K$
    $i^{(k)} = \arg\max_i |X_{*i}^\top (X\beta^{(k-1)} - y)|$
    $\mathcal{A}^{(k)} = \{i^{(k)}\} \cup \mathcal{A}^{(k-1)}$
    $\beta^{(k)} = \arg\min_\beta \|y - X\beta\|_2^2$ subject to $\beta_j = 0$ for all $j \notin \mathcal{A}^{(k)}$.
end
```

2. Please implement the ridge regression estimator and then use 10-fold cross-validation on "train-matrix.txt" to choose the optimal λ and output it.
   The ridge regression estimator is defined as:

$$\widehat{\beta}^{\text{Ridge}} = \arg\min_\beta \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2.$$

   Select the optimal λ from $\lambda = 0.0125,\ 0.025,\ 0.05,\ 0.1,\ 0.2$. In 10-fold cross-validation: continuously and evenly separate the training data, for example, in the 3rd round, 201 to 300 rows of matrix data are the testing data set, 1 to 200 and 301 to 1000 rows of matrix data are the training data set; choose the optimal λ based on minimum average prediction error.

3. Please use the estimator generated from last step, calculate prediction error on "test-matrix.txt" and $\|\widehat{\beta}^{\text{Ridge}} - \beta^*\|_2^2$.

11 Programming II [25 pts]

(*Note: This is a new programming question designed by our TA. If you have any questions or confusion regarding this question, feel free to contact our TA or me.*)

You are provided with a Jupyter notebook file: `HW1_programming_yacht.ipynb`. Please read the notebook carefully and complete the exercises specified in the `Homework1 section` at the end of the file.

Tasks

1. You are free to apply some feature mapping techniques first and then build the regression model. Also, try to perform the model selection like penalize complex models or search for the best-k features. Build your regression model and report your pipeline.
2. Evaluate the performance of your model and pipeline and compare it against the four pipelines that are introduced in the provided notebook.
3. Employ parameter tuning strategies (e.g., grid search) to improve the performance of your pipeline.
4. Provide a concise report summarizing your approach, along with key insights related to methodology, data characteristics, or parameter tuning.

Environment Setup

The conda environment for this assignment should be created using the provided `requirements.txt`. To reproduce the environment, run the following command:

```
conda env create -f environment.yml
```