ECE228 Machine Learning for Physical Applications (Spring 2023): Assignment 1

Andy Nguyen

April 20, 2023

1 Written Questions [30 Points]

1.1 Probability

1. (10 points) Let X_1 and X_2 be independent, continuous random variables uniformly distributed on $[a, b]$ where $a < b$. Let $X = \max(X_1, X_2)$. Compute

   (a) (2 points) Expection $E(X)$ The probability density function (pdf) is defined as $f_X(x) = \frac{\partial F_X(x)}{\partial x}$ where $F_X(x)$ is the cumulative density function (cdf) of X and is defined as $F_X(x) = P(X < x)$. Because X_1, X_2 are independent we can write the cdf as

$$F_X(x) = P(\max(X_1, X_2) < x) \tag{1}$$
$$= P(X_1 < x)P(X_2 < x) \tag{2}$$
$$= F_{X_1}(x)F_{X_2}(x) \tag{3}$$

   which is the product of the cdfs of X_1 and X_2. The cdf of a uniformly distributed random variable is:

$$F_X(x) = \frac{x - a}{b - a} \tag{4}$$

   Therefore, the cdf of X becomes:

$$F_{X_1}(x)F_{X_2}(x) = \frac{(x - a)^2}{(b - a)^2} \tag{5}$$

   Then the pdf becomes:

$$f_X(x) = \frac{\partial F_X(x)}{\partial x} \tag{6}$$
$$= \frac{\partial}{\partial x}(\frac{x - a}{b - a})^2 \tag{7}$$

$$E(X) = \int_a^b x f_X(x) dx \tag{8}$$
$$= \int_a^b \frac{2x^2}{(b-a)^2} - \frac{2ax}{(b-a)^2} dx \tag{9}$$
$$= \frac{1}{(b-a)^2} \int_a^b \frac{2x^3}{3} - ax^2 dx|_a^b \tag{10}$$
$$= \frac{1}{3(b-a)^2}(2b^3 - 3ab^2 + a^3) \tag{11}$$

(b) (4 points) Variance $Var(X)$

$$Var(X) = E(X^2) \tag{12}$$
$$= \int_a^b x^2 \frac{2(x-a)}{(b-a)^2} dx \tag{13}$$
$$= \frac{1}{(b-a)^2} \int_a^b 2x^3 - 2ax^2 dx \tag{14}$$
$$= \frac{1}{6(b-a)^2}[3b^4 - 4ab^4 - a^4] \tag{15}$$

(c) (4 points) Covariance $Cov(X, X_1)$, for simplicity, you can just consider $a = 0, b = 1$.

$$E(X) = \frac{1}{3(b-a)^2}(2b^3 - 3ab^2 + a^3) \tag{16}$$
$$= \frac{2}{3} \tag{17}$$

$$E(X_1) = \frac{b+a}{2} \qquad = 1/2 \tag{18}$$

$$Cov(X, X_1) = E((X - E(X))(X_1 - E(X_1)) \tag{19}$$
$$= E((max(X_1, X_2 - \frac{2}{3})(X_1 - \frac{1}{2})) \tag{20}$$

When X_1 is the maximum:

$$E[(max(X1, X2) - 2/3)(X1 - 1/2)] = E((X_1 - 2/3)(X_1 - 1/2)) \tag{21}$$
$$= \int 0^1 (x_1 - 2/3)(x_1 - 1/2) dx_1 \tag{22}$$
$$= 1/18 \tag{23}$$

When X_1 is the maximum:

$$E[(max(X1, X2) - 2/3)(X1 - 1/2)] = E((X_1 - 2/3)(X_1 - 1/2)) \tag{24}$$

$$= \int 0^1 \int 0^x 1(x1 - 2/3)(x2 - 1/2)dx2dx1 \tag{25}$$

$$= -1/36 \tag{26}$$

So the total covariance is $Cov(X, X_1) = \frac{1}{18} - \frac{1}{36} = \frac{1}{36}$.

1.2 Maximize Likelihood Estimation

2. (10 points) A discrete random variable X follows a Poisson distribution with parameter λ if

$$P(X = k) = \frac{\lambda^k}{k!}e^{-\lambda}, k \in \{0, 1, 2, \ldots\}$$

Imagine that we are investigating the number of commuting vehicles arriving at a station (per hour) from 10 am to 6 pm in a day. Assume that the vehicles arriving follow a Poisson distribution with parameter λ. The table displays the hourly recorded vehicles (with "1" representing the first observed hour, "2" representing the second observed hour, etc):

Hour	1	2	3	4	5	6	7	8
The number of vehicles	12	19	10	14	13	12	18	8

Let $G = (G_1, \ldots, G_n)$ be a random vector where G_i is the number of vehicles observed at hour i:

(a) (4 points) Write the log-likelihood function of G given λ.

$$L(G_1, G_2, ...G_n|\lambda) = \Pi_{i=1}^n P(G_i = k) \tag{27}$$

$$= \Pi_{i=1}^n \frac{\lambda^{k^{(i)}}}{k^{(i)!}}e^{-\lambda} \tag{28}$$

$$logL = l(G|\lambda) \tag{29}$$

$$l = \Sigma_{i=1}^n klog(\lambda) - log(k!) - \lambda \tag{30}$$

(b) (4 points) Compute the MLE for λ in the general case $G = (G_1, \ldots, G_n)$.

$$\lambda^* = max_\lambda \Sigma_{i=1}^n klog(\lambda) - log(k!) - \lambda \tag{31}$$

$$= -n\lambda + log(\lambda)\Sigma_{i=1}^n k^{(i)} = 0 \tag{32}$$

$$n\lambda = \Sigma_{i=1}^n k^{(i)} \tag{33}$$

$$\lambda^* = \frac{1}{n}\Sigma_{i=1}^n k^{(i)} \tag{34}$$

Note: In the above equation, since $log(k!)$ was not dependent on lambda, it was taken out of the max equation. $log(\lambda)$ was a constant, so it was taken out also.

(c) (2 points) Compute the MLE for λ using the observation data in table.

$$\lambda^* = \frac{1}{n}\Sigma_{i=1}^{n} k^{(i)} \tag{35}$$

$$= \frac{12+19+10+14+13+12+18+8}{8} \qquad = 13.25 \tag{36}$$

1.3 Linear Regression and Regularization

3. (10 points) Regularization is an important technique to avoid overfitting problem. For linear regression, we have mentioned both LASSO (which uses the L1 norm as a penalty), and ridge regression (which uses the squared L2 norm as a penalty). In practice, the scaling factor of these penalties has a significant impact on the behavior of these methods, and must often be chosen empirically for a particular dataset. In this problem, we look at what happens when we choose our regularization factor poorly.

   Recall that loss function to be optimized under ridge regression is

$$L_{ridge} = \sum_{i=1}^{n}(y_i - (\hat{w}_0 + x_i^T\hat{w}))^2 + \lambda\|\hat{w}\|_2^2$$

   where $x_i \in \mathbb{R}^d$, and λ is the regularization constant, and

$$\lambda\|\hat{w}\|_2^2 = \lambda\sum_{i=1}^{d}(\hat{w}_i)^2.$$

   The loss function to be optimized under LASSO regression is

$$L_{LASSO} = \sum_{i=1}^{n}(y_i - (\hat{w}_0 + x_i^T\hat{w}))^2 + \lambda\|\hat{w}\|_1$$

   where

$$\lambda\|\hat{w}\|_1 = \lambda\sum_{i=1}^{d}|\hat{w}_i|.$$

   (a) (5 points) Discuss how choosing a too small λ affects the magnitude of the following quantities. Please describe the effects for both ridge and LASSO

   - The error on the training set
   - The error on the testing set
   - The elements of $\hat{w}$
   - The number of nonzero element of $\hat{w}$

The regularization term adds a penalty term to the linear regression cost functions to prevent the weights from being too large. In terms of LASSO regression, because the penalty term for the weights is l_1 distance, it will be easier to do feature selection. This means that there will be more nonzero elements in LASSO than ridge regression. Since linear regression does not have this penalty term, the weights will be unregularized and can grow too large. When testing on validation sets, linear regression can have weights that are too large based on the training set. Ridge and LASSO regression will lower the error on the training and testing set with the regularized term with smaller weights. A too small λ will result on the error of the training set and testing set being larger than a good λ. The weights will be larger because there is a smaller penalty on the weights, therefore could lead to overfitting which induces higher error on the training and testing sets. For ridge regression, this will not really affect the number of nonzero elements of the weights, but for LASSO regression, this will make less nonzero elements of $\hat{w}$.

(b) (5 points) Now discuss briefly how choosing a too large λ affects the magnitude of the same quantities in the previous question. Again describe the effects for both ridge and LASSO, or state why the effects will be the same.
If λ is too large, the weights will be shrunk towards zero making the model more simpler. This can lead to a higher increase in the bias which may result in a higher error. This is the same case for the testing set because the model may be too simple and underfit the data leading to a higher error. For LASSO regression, a large lambda will induce higher number of nonzero weights as there is a higher penalty. For ridge regression, this effect will be less so because with a large lambda, this does not lead to the weights being exactly zero.

2 Coding Questions: LASSO [70 Points]

2.1 Implement Coordinate Descent to Solve the Lasso. (35 Points)

1. (35 points) In this section, you are going to implement the coordinate descent algorithm in Algorithm **??**. During each iteration t, we first evaluate $\hat{\boldsymbol{y}}^{(t)}$ and update the weight coefficient w_0 according to what you derived in question (a).

   Next, we update the weight coefficients at each dimension using the equation you derived in (b).

   (a) (5 points) Derive the update rule for w_0. (Hint: update w_0 using $\hat{\boldsymbol{y}}^{(t)}$)

$$w_0^* = min||y - Xw - w_0||_2^2 \qquad = w_0^2 - 2(y - Xw)^T w_0 + c = 0 \tag{37}$$

$$=> w_0^* = y - Xw \tag{38}$$

   Note: c is a constant term that is deleted because in the min function, it is not dependent on w_0 which is our minimizing term.

(b) (10 points) Now, we will update each dimension of the weight vector step by step. For the update of w_k, we treat other dimensions of the weight vector as fixed, and we are using the k-th column (i.e., k-th feature) of the data matrix X for predicting the response y, and leverage the residual prediction error to update w_k. Specially, let denote $a_k = 2\sum_{i=1}^{N} X_{ik}^2$, $c_k \leftarrow 2\sum_{i=1}^{N} \boldsymbol{X}_{ik}\left(y_i - (w_0^{(t+1)} + \sum_{j<k} w_j^{(t+1)} \cdot \boldsymbol{X}_{ij} + \sum_{j>k} w_j^{(t)} \cdot \boldsymbol{X}_{ij}\right)$. Derive the update rule for w_k and justify your answers.

$$w^* = min||y - Xw - w_0||_2^2 + \lambda|w| \tag{39}$$

$$=> \Sigma_{i=1}^{N} 2X_{ik}^2 w_k^2 - 2(y_i - w_0 - \Sigma_{j\neq k}^{d} X_{ij}w_j)X_{ik}w_k + \lambda|w_k| = 0 \tag{40}$$

$$=> a_k w_k^2 - c_k w_k + \lambda|w_k| = 0 \tag{41}$$

Where $a_k = 2\Sigma_{i=1}^{N} X_{ik}^2$ and $c_k = 2\sum_{i=1}^{N} \boldsymbol{X}_{ik}\left(y_i - (w_0^{(t+1)} + \sum_{j<k} w_j^{(t+1)} \cdot \boldsymbol{X}_{ij} + \sum_{j>k} w_j^{(t)} \cdot \boldsymbol{X}_{ij}\right)$. After computing the subdifferential of the Lasso cost function and equating to zero to find the minimum, we achieve this update rule:

$$w_k = \frac{c_k + \lambda}{a_k} if w_k < -\lambda \tag{42}$$

$$w_k = 0 if -\lambda \leq w_k \leq \lambda \tag{43}$$

$$w_k = \frac{c_k - \lambda}{a_k} if w_k > \lambda \tag{44}$$

$$\tag{45}$$

2.2 Test Lasso Solver on a Synthetic Dataset. (15 Points)

2. (15 points) We will now try out your solver with some synthetic data. A benefit of the Lasso is that if we believe many features are irrelevant for predicting $\boldsymbol{y}$, the Lasso can be used to enforce a sparse solution, effectively differentiating between the relevant and irrelevant features. We will find out how it works in this synthetic dataset.

Suppose $\boldsymbol{x} \in \mathbb{R}^d, y \in \mathbb{R}$ and pairs of data $(\boldsymbol{x}_i, y_i)$ are generated independently according to the model

$$y_i = w_o^* + w_1^* x_{i,1} + w_2^* x_{i,2} + \ldots + w_k^* x_{i,k} + \epsilon_i \tag{46}$$

where $k < d$ and $\epsilon_i \sim N(0, \sigma^2)$ is some Gaussian noise. Note that since $k < d$, the features $k+1$ through d are unnecessary for predicting y. You are now tasked with the following:

(a) (10 points) Let $N = 50, d = 80, k = 6$, and σ = 1. Generate some data by drawing each element of $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ from a standard normal distribution $N(0, 1)$. Let $w_0^* = 0$ and create a $\boldsymbol{w}^*$ by setting the first k elements to ±10 (you can choose any sign pattern) and the remaining elements to 0. Finally, generate a Gaussian noise vector ϵ with variance σ^2 and form the training data set $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}^* + w_0^* + \epsilon$.

With your synthetic data, solve multiple Lasso problems on a regularization path, starting at λ_{max} and decreasing λ by a constant ratio until a few features are chosen correctly. Record the sparsity pattern of your Lasso solution $\hat{\boldsymbol{w}}$ and compare it to that of the true sparsity pattern of model parameters $\boldsymbol{w}^*$. Specially, you need to generate two plots: 1) the first plot shows how the precision rate (y-axis) changes with respect to the value of λ (you can either use the raw λ value or use the normalized λ/λ_{max} for x-axis). Precision is defined as the number of correct nonzeros in $\hat{\boldsymbol{w}}$/total number of nonzeros in $\hat{\boldsymbol{w}}$). 2) the second plot shows how the recall rate (y-axis) changes with respect to the value of λ (you can either use the raw λ value or use the normalized λ/λ_{max} for x-axis). The recall rate is defined as the number of correct nonzeros in $\hat{\boldsymbol{w}}/k$.

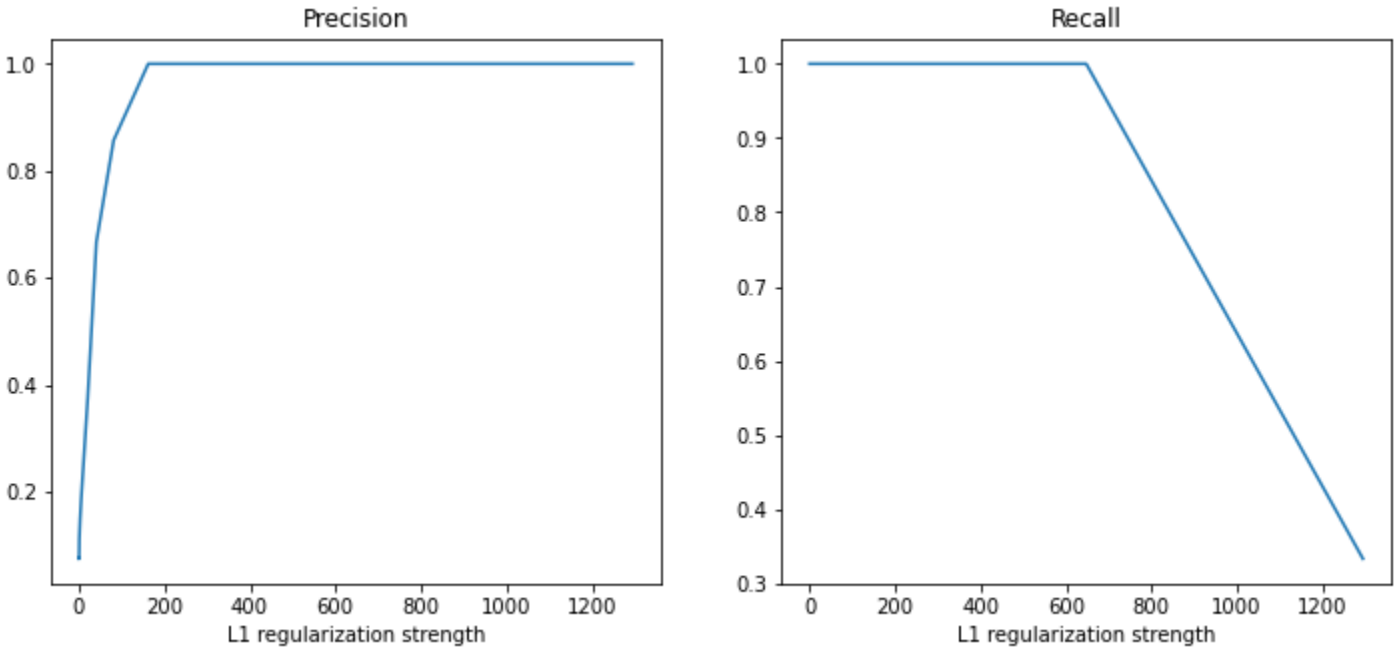

Figure 1: Precision and Recall rate of

(b) (5 points) Select the appropriate value for λ and generate the output weight vector $\hat{w}$. Please report based on your best-performing model, your chosen λ value and the corresponding $\hat{\boldsymbol{w}}$ as well as the true weight vector $\boldsymbol{w}$, for simplicity you can just report the nonzeros $\boldsymbol{w}^*$ and $\hat{\boldsymbol{w}}$ elements, and the precision/recall rates.

```
regs[i]
```

```
418.8990446421737
```

```
w_star, w0_star = lasso(X, y, regs[i])
```

```
w_star
```

```
array([[ 4.89708608],
       [ 2.56568901],
       [-5.94693123],
       [-6.17705571],
       [-3.84439451],
       [ 4.02466183],
```

Figure 2: λ = 418.9 and there are 6 nonzero values for w^*

2.3 Solve a Real-world Problem: Water Quality Prediction (20 Points)

(a) (12 points) Solve Lasso to predict water quality for **the first location with "Location ID"=0**. Divide the training data into 80% training set and 20% validation set. We will provide dataload codes for you and you can directly get

$$X_{train}, y_{train}, X_{val}, y_{val}, X_{test}, y_{test}.$$

Starting at λ_{max}, run Lasso on the training set, decreasing λ using previous solutions as initial conditions to each problem. Stop when you have considered enough λ 's that, based on the validation error, you can choose a good solution with confidence (for instance, when the validation error begins increasing or stops decreasing significantly). At each solution, record the root-mean-squared-error (RMSE) on **training and validation** data for generating plots later. In addition, record the number of nonzeros in each solution.

Produce two plots and include in your homework PDF submission: Plot 1) How the training RMSE and validation RMSE changes with respect to λ using the plot function we provided; Plot 2) How the number of nonzeros in the final weight estimates changes with respect to λ.

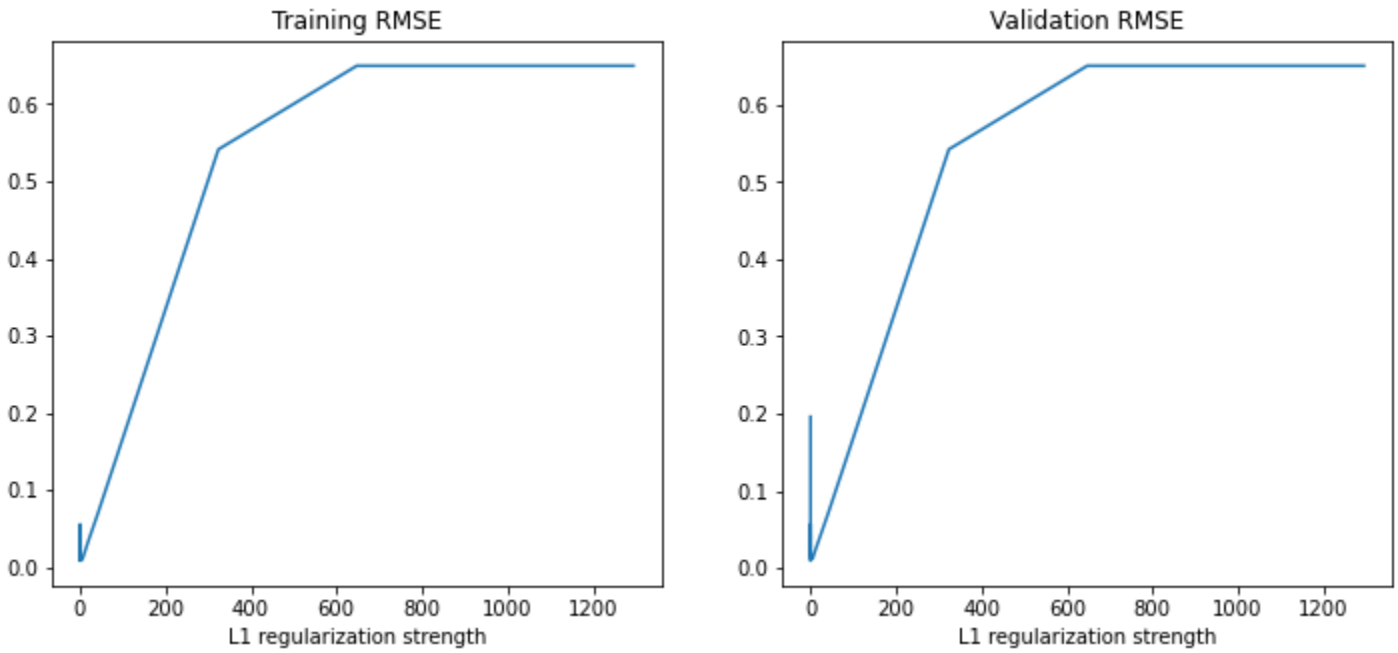

Figure 3: Training and validation RMSE across different L1 regularization strengths

(b) (4 points) Find the λ that achieves the best validation RMSE and test your model on the given **test** data. Report the test RMSE value and corresponding λ.

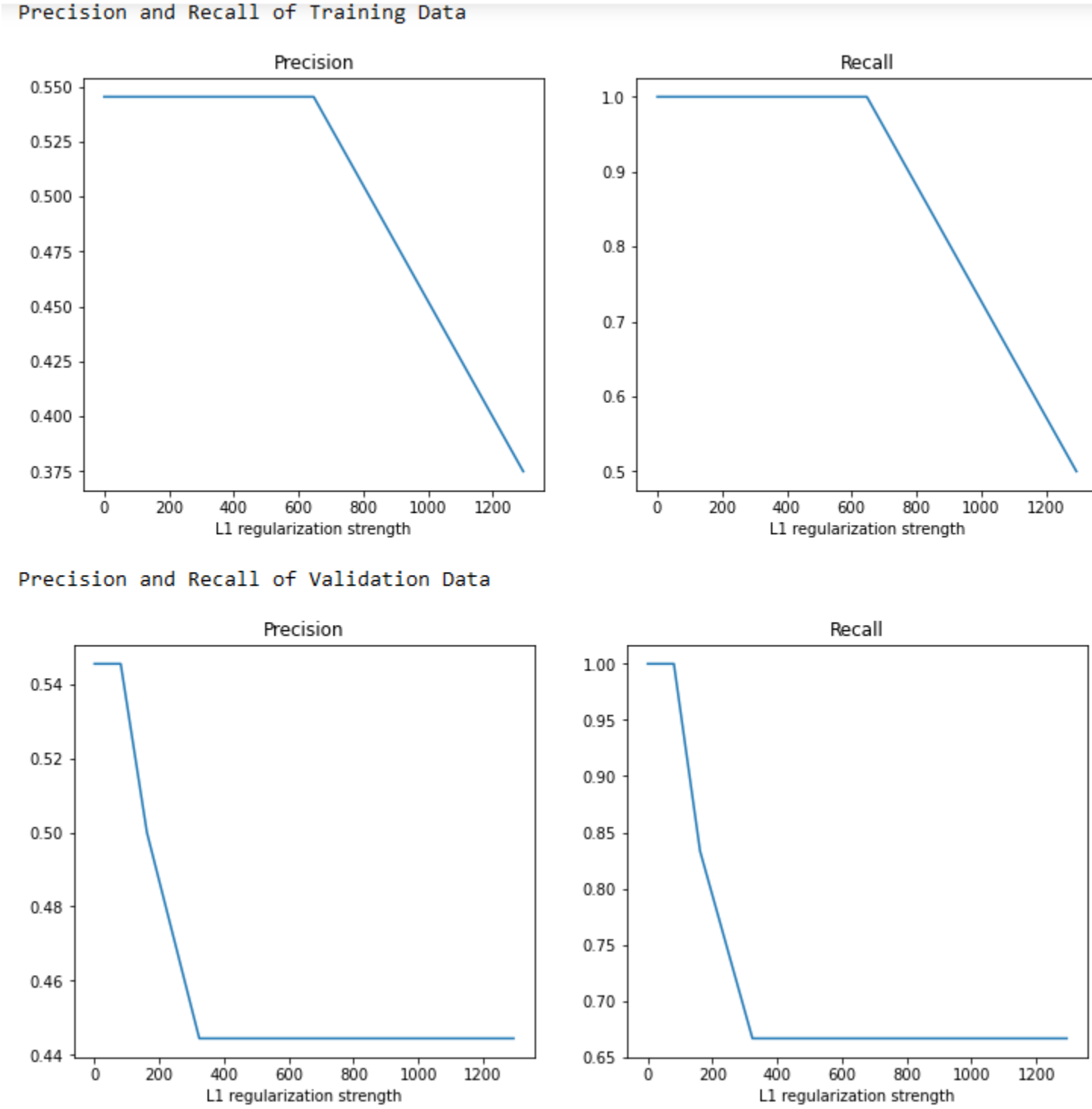

Figure 4: Precision and Recall Values for different L1 regularization strengths across Training and validation data.

```
In [31]: water_w_star, water_w0_star = lasso(X_train_1, y_train_1, regs[1])
         y_pred_test = X_test_1@water_w_star + water_w0_star
         rmse_test = compute_rmse(y_test_1, y_pred_test)
         print("RMSE Test", rmse_test)
         print("Best Lambda Value", regs[1])

         RMSE Test 0.6575831165460446
         Best Lambda Value 647.1056578541386
```

Figure 5: RMSE Test Value and Best Lambda Value for Water Quality Test

(c) (4 points) Inspect your solution in (b) and take a look at the 5 features with the largest weights magnitude. List the names of these features and their weights, and comment on if the weights generally make sense intuitively.

Based on the weights derived from the experiments, the 5 features with the largest weights are maximum specific conductance of water, max pH, max dissolved oxygen, mean temperature and minimum temperature. These make sense because when determining water quality, pH and temperature help determine this. Specific conductance of water will determine how much electric currant can pass through. The pH of water is a measure of how acidic the water. The amount of oxygen dissolved in water is also important because oxygen in water is important to aquatic life. Low dissolved oxygen levels is attributed with pollution. Temperature affects the solubility of oxygen and gases in water. High temperatures can cultivate harmful bacteria and algae that can impact water quality.