CSCI 5525: Advanced Machine Learning (Spring 2023)

Homework 1

(Due Tue, Jan. 31, 11:59 PM CST)

1. **(10 points)** A generalization of the least squares problem adds an affine function to the objective,

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \mathbf{a}^\top \mathbf{w} + b$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{w} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^n, \mathbf{a} \in \mathbb{R}^d$, and $b \in \mathbb{R}$. Assume the columns of $\mathbf{X}$ are linearly independent. This generalized problem can be solved by reducing it to a standard least squares problem, using a trick called *completing the square*.

Show that the objective of the problem above can be expressed in the form

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \mathbf{a}^\top \mathbf{w} + b = \|\mathbf{X}\mathbf{w} - \mathbf{y} + \mathbf{f}\|_2^2 + g$$

where $\mathbf{f} \in \mathbb{R}^n, g \in \mathbb{R}$. Then solve the generalized least squares problem

$$\text{argmin}_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y} + \mathbf{f}\|_2^2 + g.$$

2. **(25 points)** In this problem, you will write Python code to implement k-fold cross-validation from scratch. Write the function `my_cross_val(model, loss_func, X, y, k)` which performs k-fold cross-validation on the data (X, y) using `model` and returns the loss value using `loss_func` for each validation fold. You can assume the value of `loss_func` will be either `'mse'` or `'err_rate'` which correspond to the mean squared error (MSE) and error rate loss functions which are computed as

$$\text{MSE:} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{1}$$

$$\text{Error rate:} \quad \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}[y_i \neq \hat{y}_i] \tag{2}$$

where n is the number of data points, y_i is the target value or label, and $\hat{y}_i$ is the predicted target value or label.

You may also assume X is an $n \times d$ matrix where n is the number of data points and d is the number of features, and y is an n-dimensional vector of target values or labels. Moreover, the function will be called with parameter `model` which is an instance of a class object with methods `model.fit(X', y')` and `model.predict(X'')` where X', y', and X'' are subsets of X and y. The `model.fit` method will not return anything and `model.predict` will return a list of predictions of length equal to the number of rows in X''. Test your code with the script `hw1_q2.py`. Note, you cannot use any machine learning packages (like scikit-learn) in this problem.

3. **(25 points)** Here we consider the regression problem of predicting the median house value for California districts. We will be using the California housing dataset[1] which comes packaged with scikit-learn[2]. The dataset has 20640 data points, 8 features (median income, median house age, average number of rooms, average number of bedrooms, population, average number of household members, latitude, and longitude) and 1 target variable (median house value for California districts).

   We will consider both ridge regression and lasso methods. Write Python code to implement ridge regression from scratch in the class `MyRidgeRegression`. For lasso, you may use the scikit-learn class `sklearn.linear_model.Lasso`[3]. For both algorithms, choose the optimal regularization parameter λ by using your cross-validation function `my_cross_val` to run k-fold cross-validation for $k = 10$, using MSE loss, and $\lambda = \{0.01, 0.1, 1, 10, 100\}$.

   Using `my_cross_val`, report the mean squared error (MSE) in each fold as well as the mean and standard deviation across folds. Which value of λ is optimal for each method? Using the optimal value of λ, train a single model for each method using all the training data and then predict using the test data. Report the MSE on the test data. Which model performs the best? In addition to `MyRidgeRegression.py` (details below), add your code to `hw1_q3.py` and use it to test. Note, you must implement the MSE without using any machine learning packages like scikit-learn.

   `MyRidgeRegression.py` is a class which must have the following:

```
class MyRidgeRegression:

   def __init__(self, lambda_val):
       ...
   def fit(self, X, y):
       ...
   def predict(self, X):
       ...
```

   Your class `MyRidgeRegression` **should not** inherit any base class. The `fit` method does not return anything and the `predict` method should return a list of predictions of length equal to the number of rows in `X`.

   Please write up your results and submit them in a PDF document. For each method and value of λ, report the validation set MSE for each of the $k = 10$ folds, the mean MSE over the k folds, and the standard deviation of the MSE over the k folds. Make a table to present the results. Include a column in the table for each fold, and add two columns at the end to show the overall mean error rate and standard deviation over the k folds. For example:

MSE for Ridge Regression											
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
#	#	#	#	#	#	#	#	#	#	#	#

[1] `https://scikit-learn.org/stable/datasets/real_world.html#california-housing-dataset`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html#sklearn.datasets.fetch_california_housing`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`

4. **(40 points)** Here we consider a binary classification problem using the provided dataset 'hw1_q4_dataset.csv' which has 2000 datapoints with 2 features from 2 classes. We will use Fisher's linear discriminant analysis (LDA) for this problem. Recall, Fisher's LDA computes a $\mathbf{w}$ vector which is used to project the data to $\mathbb{R}$ via $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. Once each data point has been projected, a threshold $\lambda \in \mathbb{R}$ is chosen such that $\mathbf{x}$ is predicted to belong to one class if $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \geq \lambda$ and the other class otherwise. Write Python code to implement Fisher's LDA from scratch in class `MyLDA`. Using your function `my_cross_val`, run $k = 10$ fold cross-validation with `loss_func = 'err_rate'` to choose the optimal threshold value λ by minimizing classification error rate. You must decide on the set of λ values to experiment with. I encourage you to visualize the projected data points to see what values make sense. Note, be careful with how you select which class you predict a data point belongs to using λ.

   Using `my_cross_val`, report the classification error rate in each fold as well as the mean and standard deviation across folds. Which value of λ is optimal? Using the optimal value of λ, train a single model using all the training data and then predict using the test data. Report the classification error rate on the test data. In addition to `MyLDA.py` (details below), add your code to `hw1_q4.py` and use it to test.

   `MyLDA.py` is a class which must have the following:

```
class MyLDA:

   def __init__(self, lambda_val):
       ...
   def fit(self, X, y):
       ...
   def predict(self, X):
       ...
```

   Your class `MyLDA` **should not** inherit any base class. The `fit` method does not return anything and the `predict` method should return a list of predictions of length equal to the number of rows in `X`.

   Please write up your results and submit them in a PDF document. Report the validation set error rates for each of the $k = 10$ folds and λ, the mean error rate over the k folds, and the standard deviation of the error rates over the k folds. Make a table to present the results. Include a column in the table for each fold, and add two columns at the end to show the overall mean error rate and standard deviation over the k folds. For example:

Error rates for LDA											
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	Mean	SD
#	#	#	#	#	#	#	#	#	#	#	#

Instructions

You must complete this homework assignment individually. You may discuss the homework at a high-level with other students but make sure to include the names of the students in your README file. You may not use any AI tools (like GPT-3, ChatGPT, etc.) to complete the homework. Code can only be written in Python 3.6+; no other programming languages will be accepted. One should be able to execute all programs from the Python command prompt or terminal. Make sure to include a requirements.txt, yaml, or other files necessary to set up your environment. Please specify instructions on how to run your program in the README file.

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures, if any, should be included in the PDF report.

In your code, you can only use machine learning libraries such as those available from scikit-learn as specified in the problem description. You may use libraries for basic matrix computations and plotting such as numpy, pandas, and matplotlib. Put comments in your code so that one can follow the key parts and steps in your code.

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**

  1. `[YOUR_NAME]_hw1_solution.pdf`: A document which contains the written solutions to all problems.
  2. `my_cross_val.py`: Code for Problem 2.
  3. `hw1_q3.py` and `MyRidgeRegression.py`: Code for Problem 3.
  4. `hw1_q4.py` and `MyLDA.py`: Code for Problem 4.
  5. `README.txt`: README file that contains your name, student ID, email, instructions on how to run your code, any assumptions you are making, and any other necessary details.
  6. Any other files, except the data, which are necessary for your code.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems at a high level only. Each person must write up the final solutions individually. You need to list in the README.txt which problems were a collaborative effort and with whom. Please refer to the syllabus for more details. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online,
- Look up things/post on sites like Quora, StackExchange, etc.