CS589: Machine Learning - Fall 2025

Homework 4: Regression

Assigned: Friday, October 17. Due: Thursday, October 23 at 8:00pm

Getting Started: This assignment consists of written problems and coding problems. Download the assignment archive from Canvas and unzip the file. Starter code for coding problems is provided in the code directory. For general clarification questions, please submit public posts to Campuswire. For questions related to your specific solutions, please submit private posts on Campuswire. In-person help is available through regularly scheduled office hours.

Due Date and Late Work: The assignment is due at 8:00pm ET on October 23rd, 2025. Students can submit up to 11:59pm on the due date with no penalty. Work submitted up to 11:59pm on one day after the assignment is due is subject to a penalty of 10%. Work submitted up to 11:59pm two days after the assignment is due is subject to a penalty of 20%. Gradescope will close at 11:59pm two days after the assignment is due and work can not be submitted for credit after this point.

How to Submit: Your written report must be submitted to Gradescope as a PDF file. You must select the page on which each answer appears when uploading your report to Gradescope. You are encouraged to typeset your PDF solutions using LaTeX. The source of this assignment is provided to help you get started. You may also submit a PDF containing scans of *clear* hand-written solutions. Work that is illegible will not count for credit. For this assignment, code should be submitted to Gradescope as Python 3.10+ Jupyter Notebooks. Autograding will not be used for this assignment. You may submit both the code and the report as many times as you like before Gradescope closes. Only your final submission will be graded. Any late penalties will be based on the timestamp of your final submission as determined by Gradescope.

Academic Honesty Reminder: Homework assignments are individual work. Being in possession of another student's solutions, code, code output, or plots/graphs for any reason is considered cheating. Sharing your solutions, code, code output, or plots/graphs with other students for any reason is considered cheating. Copying solutions from external sources (books, web pages, etc.) is considered cheating. Collaboration indistinguishable from copying is considered cheating. Posting your code to public repositories like GitHub (during or after the course) is not allowed. Manual and algorithmic cheating detection are used in this class. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Generative AI Use Reminder: The use of generative AI tools to help with coding is permitted for programming problems in this course. The use of generative AI for all other work is considered cheating.

1. (*10 points*) **The Constant Regressor**. Consider a model that predicts a constant value for all inputs, $f(\mathbf{x}) = b$. Given a dataset, $D = \{(\mathbf{x}_i, y_i)|1 \leq i \leq N\}$, show that the value of b that minimizes the mean squared error $\frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2$ is $\hat{b} = \frac{1}{N}\sum_{i=1}^{N} y_i$. Show your work.

2. (*15 points*) **Ridge Regression Derivation.** Consider the ridge regression learning problem for a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)|1 \leq i \leq N\}$ with $\mathbf{x}_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}$. For simplicity in this problem, we will consider a linear regression model with no bias: $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$. The optimization problem is defined as shown below.

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left(\left(\frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \right) + \lambda \|\mathbf{w}\|_2^2 \right).$$

a. (*5 pts*) To begin, re-write the objective function (including the regularizer) in matrix form, similar to the OLS linear regression case. Your answer should be given in terms of an output matrix $\mathbf{Y}$ of size $N \times 1$ containing the regression target value y_i on row i, an input matrix $\mathbf{X}$ of size $N \times D$ where $\mathbf{X}_{id}$ is the value of feature d for data case i, and the weight vector $\mathbf{w}$. Show your work and give your final answer in your report.

b. (*5 pts*) Now, derive the gradient of the objective function in matrix form. You will need the matrix calculus results $\nabla(\mathbf{w}^\top \mathbf{x}) = \mathbf{x}$ and $\nabla(\mathbf{w}^\top \mathbf{A}\mathbf{w}) = (\mathbf{A} + \mathbf{A}^\top)\mathbf{w}$ where $\mathbf{w}$ and $\mathbf{x}$ are column vectors of length D and $\mathbf{A}$ is a $D \times D$ matrix. Show your work and give your final answer in your report.

c. (*5 pts*) Lastly, solve the gradient equation to show that the optimal ridge regression parameters are given by $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + N\lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}$. Show your work.

3. (*30 points*) **Ridge Regression Implementation** In this question, you will implement and test the full version of ridge regression, including a bias term that is not regularized. Your implementation must be in NumPy (use of `scikit-learn` or other libraries is not allowed for the model implementation). The regression function for the model is given by $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ with $\mathbf{w}, \mathbf{x}$ length D column vectors and b a scalar. To learn the model, we define the parameter vector $\theta = [w_1, ..., w_D, b]^\top$ as a column vector that contains the weights followed by the bias. The formula for learning the parameters of this version of the model is given below.

$$\hat{\theta} = (\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}} + N\Lambda)^{-1} \tilde{\mathbf{X}}^\top \mathbf{Y}.$$

Here, the matrix $\tilde{\mathbf{X}}$ is an augmented data matrix of shape $N \times (D + 1)$ where each row i contains the feature vector for data case i, followed by a constant 1: $\tilde{\mathbf{X}}_i = [x_{i1}, ..., x_{iD}, 1]$. Λ is a diagonal matrix of size $(D + 1) \times (D + 1)$ where all of the diagonal entries are equal to the regularization parameter strength λ except for $\Lambda_{D+1,D+1}$, which is 0. This ensures the bias is not regularized.

a. (*5 pts*)To begin, complete the `predict` function in the `RidgeRegressor` class defined in `ridge.ipynb`. There is nothing to include in your report for this question.

b. (*5 pts*) Now, complete the `fit` function in the `RidgeRegressor` class defined in `ridge.ipynb`. There is nothing to include in your report for this question.

c. (*5 pts*) Setting $\lambda = 0$ yields a special case where ridge regression simplifies to standard OLS linear regression. As a check on your implementation, use the provided data set to fit your model to the training data set.

Report the train, validation and test MSEs. Now, use the same training data to fit scikit-learn's implementation of OLS linear regression `sklearn.linear_model.LinearRegression` and report the train, validation and test MSES. If your implementation is correct, the MSE values of the two implementations should match when $\lambda = 0$. Add your code to `ridge.ipynb`.

d. (*5 pts*) In the limit as λ goes to infinity, the ridge regression model simplifies to the Constant Regressor model introduced in Question 1. To test this case, fit your model to the training set using $\lambda = 10^5$. Report the train, validation and test MSEs. Now fit the Constant Regressor model to the training data set and compute its train, validation and test MSEs. If your implementation is correct, the MSE values of the two implementations should approximately match. Add your code to `ridge.ipynb`.

e. (*5 pts*) In the next two question parts, you will conduct a train-validation-test experiment to determine the optimal value of λ. Use the 50 logarithmically spaced values of λ from 10^{-5} to 10^0 defined in the starter code for this question. As your answer to the question, provide a graph showing the training and validation MSE curves as a function of the value of λ. Your graph should use a log axis to show the values of λ (in matplotlib, use a semilogx plot type). Label all axes and provide a legend. Add your code for this experiment to `ridge.ipynb`.

f. (*5 pts*) Finally, using the output from the previous part, determine the optimal value of λ. Report the optimal value of lambda along with the train, validation and test MSEs for the corresponding optimal model. Add your code to `ridge.ipynb`.

4. (*15 points*) **Regularization Paths.** In this question, you will examine what happens to the model coefficients when fitting models using ridge regression and Lasso while varying the regularization parameter strength. For this question, you will use scikit-learn's implementations sklearn.linear_model.Ridge and sklearn.linear_model.Lasso. In these implementations, the regularization strength parameter is α (alpha). Answer the following questions. Starter code is in `regularization_paths.ipynb`.

a. (*5 pts*) Using sklearn.linear_model.Ridge, learn the model for the values of α provided in the starter code using the provided training data set. For each fit model, record the learned weights. The learned weights for the fit model are stored in the `coef_` member variable. As your answer to this question in your report, provide a single graph with one trend line per weight showing how that weight changes as a function of the regularization parameter strength. Your graph should use a log axis to show the values of α (in matplotlib, use a semilogx plot type). There are eight features in the data set so there are eight weights and the graph should contain eight lines. Label all axes and provide a legend. Add your code to `regularization_paths.ipynb`.

b. (*5 pts*) Now repeat the experiment from part (a) using sklearn.linear_model.Lasso. As your answer to this question in your report, provide a single graph with one trend line per weight as in (a). Add your code to `regularization_paths.ipynb`.

c. (*5 pts*) The graphs produced in (a) and (b) are called "regularization paths" as they plot the trajectory of each weight as a function of regularization strength. What key difference can be seen when comparing the Ridge and Lasso regularization paths? How do these differences relate to properties of the Ridge and Lasso models?

5. (*30 points*) **Bagging.** In this question, you will use scikit-learn's implementation of the decision tree regressor model, `DecisionTreeRegressor`, as a base to implement bagging. We will use the California Housing dataset as the data set for this question. Code to load the data set is included in the starter notebook for this question, `bagging.ipynb`.

a. (*5 pts*) To begin, add code to `bagging.ipynb` to randomly split the data set into a training set and a test set of size 80% and 20% respectively. Use the scikit-learn function `train_test_split` from `sklearn.model_selection.train` with `random_state=0`. As your answer to this question, include in your report the size of the training and test sets.

b. (*5 pts*) Next, implement the function `resample` in `bagging.ipynb`. The function takes as input a feature matrix X of shape (N, D) and an array of corresponding regression outputs y of shape $(N,)$. It returns a new feature matrix Xnew of shape (N, D) and an array of regression outputs ynew of shape $(N,)$ created by sampling N data cases from the input data set (X,y) with replacement. It also returns an array of indices ind of shape $(N,)$ to keep track of the index j of original data case corresponding to each new data case i. Specifically, if Xnew[i, :] and ynew[i] correspond to X[j, :] and y[j], then ind[i] = j. As your answer to this question, apply your resample function to the training data set. Using the ind array, determine the percent of the original training data cases that are not included in the resampled data set.

c. (*5 pts*) Now, use your resample function and the `DecisionTreeRegressor` model to implement the fit function in the provided BaggedTrees class in `bagging.ipynb`. The init function for the model takes the number of trees to use K and stores it in self.K. You will need to sample K data sets, learn one decision tree model on each resampled data set, and store the K models to create the bagged ensemble. For the `DecisionTreeRegressor` base model, set the maximum depth to 20. Leave the other model hyper-parameters at their default values. There is nothing to include in your report for this question.

d. (*5 pts*) Next, implement the predict function in the provided BaggedTrees class in `bagging.ipynb`. You will need to use each of the K learned model to make predictions, then average their outputs to obtain a single prediction for each input. There is nothing to include in your report for this question.

e. (*5 pts*) Lastly, learn your BaggedTree model using $K \in \{1, 10, 25, 50, 100\}$. For each learned ensemble, compute the training set MSE and the test set MSE. Make a plot showing the training and test set MSEs verses the ensemble size K. Make sure to include a legend and label your axes. Add your code to `bagging.ipynb`.

f. (*5 pts*) Using the plot from the previous part, how does using a single decision tree with max depth 20 compare (e.g., K=1) compare to using an ensemble of multiple trees? Do we need to worry about overfitting as the value of K increases? Explain your answers.

6. (*0 points*) If you used generative AI tools to help complete any programming questions on this assignment, please briefly describe which tools you used, how you used them, and for which problems you used them. If you did not use generative AI tools, please indicate that as your response to this question.