COMP6321: Machine Learning

Assignment 2

Due: 11:59 PM (EST), October 10, 2024, submit on Moodle.

Include your name and student ID!

- Submit your write-up in PDF (we do not accept hand-written submissions) and all source code as an ipynb file (with proper documentation). Your writeup should include written answers to all questions, including the reported results and plots of coding questions in a single PDF file. Please save the output of each cell or your coding questions may NOT be graded. Write a script for each programming exercise so the TAs can easily run and verify your results. Make sure your code runs!
- Text in square brackets are hints that can be ignored.

NOTE For all the exercises, you are only allowed to use the basic Python, Numpy, and matplotlib (or other libraries for plotting), unless specified otherwise.

Exercise 1: Regression Implementation (9 pts)

Recall that ridge regression refers to

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \overbrace{\underbrace{\frac{1}{2n}\|X\mathbf{w}+b\mathbf{1}-\mathbf{y}\|_2^2}_{\text{error}}+\lambda\|\mathbf{w}\|_2^2}^{\text{loss}}, \tag{1}$$

where $X \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ are the given dataset and $\lambda \geq 0$ is the regularization hyperparameter. If $\lambda = 0$, then this is the standard linear regression problem. Observe the distinction between the error (which does not include the regularization term) and the loss (which does).

1. (1.75 pts) Show that ridge regression can be rewritten as a non-regularized linear regression problem with data augmentation. That is, prove 1 is equivalent to

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left\|\begin{bmatrix} X & \mathbf{1}_n \\ \sqrt{2\lambda n} I_d & \mathbf{0}_d \end{bmatrix}\begin{bmatrix}\mathbf{w}\\ b\end{bmatrix} - \begin{bmatrix}\mathbf{y}\\ \mathbf{0}_d\end{bmatrix}\right\|_2^2, \tag{2}$$

   where I_d is the d-dimensional identity matrix, and $\mathbf{0}_k$ and $\mathbf{1}_k$ are zero and one column vectors in k dimensions.

2. (2.25 pts) Implement the Ridge regression algorithm using the closed form solution for linear regression. Do not use any library like Scikit Learn that already has linear regression implemented. Feel free to use general libraries for array and matrix operations such as numpy. You may find the function `numpy.linalg.solve` useful. Test Ridge regression implementation on the Boston housing dataset (to predict the median house price, i.e., y). Use the train and test splits provided on Moodle. Try $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$ and report your training error, and test error for each.

3. (2.25 pts) Repeat step 2 but solve Ridge regression using the gradient descent algorithm. Try $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$ and report your training error, and test error for each.

4. (2.25 pts) Repeat step 3, but this time solve Lasso regression using gradient descent. For Lasso regression, the regularization term in equation (1) is $\lambda||w||_2$ i.e. you only have the norm and not the square of the norm. Try $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$ and report your training error, and test error for each.

5. (0.25 pts) Do you think gradient descent is better than the closed form solution of Ridge regression? Explain why.

6. (0.25 pts) Print the θ values found using gradient descent for both Ridge and Lasso regression. Are these two values different? If so, can you explain how the Lasso and Ridge regression algorithms affected the θ values?

Exercise 2: Decision Trees (7 pts)

In this exercise, you will implement decision trees for binary classification. Use the provided stub files for training and test data.

Recall: decision trees are constructed by repeatedly splitting of nodes. We split a node by measuring the loss of splitting with respect to each possible feature and threshold, and split based on the feature and threshold that minimizes this loss. Mathematically:

$$X_L = \{x \ : \ x \in X \wedge x[i] \leq j\},$$

$$X_R = \{x \ : \ x \in X \wedge x[i] > j\},$$

where $x[i]$ is the ith coordinate of point x. The vector of labels y is split into vectors y_L and y_R using the same indices.

The loss of splitting a training dataset into a left and right half is computed as

$$\ell(X, y, i, j) = \frac{|y_L|}{|y|}\ell(y_L) + \frac{|y_R|}{|y|}\ell(y_R)$$

We will consider the following loss functions ℓ, specialized for binary classification.[a] Define $\hat{p}$ for a vector of labels y to be $\frac{|\{y_j=1:y_j\in y\}|}{|y|}$, that is, the fraction of labels which are 1. We have the following three loss functions.

Misclassification error:

$$\min\{\hat{p}, 1-\hat{p}\}$$

Gini coefficient:

$$\hat{p}(1-\hat{p})$$

Entropy:

$$-\hat{p}\log_2(\hat{p}) - (1-\hat{p})\log_2(1-\hat{p})$$

We do not split a node if it is pure (i.e., consists entirely of either 0's or 1's), or if a split would exceed a maximum depth hyperparameter provided to the decision tree (recall that the depth of a single-node tree is 0).

1. (6 pts) Implement and train decision trees on the provided dataset. Create a different plot for each of the three loss functions (misclassification error, Gini index, and entropy). The x-axis of each plot should show the maximum depth of the tree (starting from 0), and the y-axis should indicate the accuracy. Include two trend lines, one for the training accuracy and test accuracy.

2. (1 pt) Observe and comment on how the different loss functions perform, and how train and test accuracy change as a function of the maximum depth.

[a] Note that these are slightly different from what we discussed in class, since we are only focusing on binary classification. Additionally, there was some implicit rescaling which we omit here.