CS480/680: Introduction to Machine Learning

Homework 1

Due: 11:59 pm, June 03, 2021, submit on Crowdmark and LEARN.

Include your name and student number!

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!

[Text in square brackets are hints that can be ignored.]

Exercise 1: Perceptron Implementation (5 pts)

Convention: All algebraic operations, when applied to a vector or matrix, are understood to be element-wise (unless otherwise stated).

Algorithm 1: The perceptron algorithm.

```
Input: $X \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{-1,1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, max_pass $\in \mathbb{N}$
Output: $\mathbf{w}, b, mistake$
for $t = 1, 2, \ldots,$ max_pass do
    $mistake(t) \leftarrow 0$
    for $i = 1, 2, \ldots, n$ do
        if $y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \leq 0$ then
            $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$                // $\mathbf{x}_i$ is the $i$-th row of $X$
            $b \leftarrow b + y_i$
            $mistake(t) \leftarrow mistake(t) + 1$
```

Implement the perceptron in Algorithm 1. Your implementation should take input as $X = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{-1,1\}^n$, an initialization of the hyperplane parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and the maximum number of passes of the training set [suggested max_pass $= 500$]. Run your perceptron algorithm on the spambase dataset (use the version on the course website), and plot the number of mistakes (y-axis) w.r.t. the number of passes (x-axis).

Ans:

Exercise 2: Regression Implementation (8 pts)

Recall that ridge regression refers to

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

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 pt) Show that ridge regression can be rewritten as a non-regularized linear regression problem. That is, prove 1 is equivalent to

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

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.

Ans:

2. (1 pt) Show that the derivatives of 1 are

$$\frac{\partial}{\partial \mathbf{w}} = \tfrac{1}{n} X^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + 2\lambda \mathbf{w} \tag{3}$$

$$\frac{\partial}{\partial b} = \tfrac{1}{n} \mathbf{1}^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}). \tag{4}$$

Ans:

3. (2 pts) Implement ridge regression using the closed form solution for linear regression as derived in lecture. You may find the function `numpy.linalg.solve` useful.

Ans:

4. (2 pts) Implement the gradient descent algorithm for solving ridge regression. The following incomplete pseudo-code may of help. Your training loss should monotonically decrease during iteration; if not try to tune your step size η, e.g. make it smaller.

Ans:

Algorithm 2: Gradient descent for ridge regression.

```
Input: $X \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{w}_0 = \mathbf{0}_d$, $b_0 = 0$, max_pass $\in \mathbb{N}$, $\eta > 0$, tol $> 0$
Output: $\mathbf{w}, b$
for $t = 1, 2, \ldots,$ max_pass do
    $\mathbf{w}_t \leftarrow$
    $b_t \leftarrow$
    if $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| \leq$ tol then        // can use other stopping criteria
        break
$\mathbf{w} \leftarrow \mathbf{w}_t,\ b \leftarrow b_t$
```

Ans:

5. (2 pts) Test your two implementations on the Boston housing dataset (to predict the median house price, i.e., y). Use the train and test splits provided on course website. Try $\lambda \in \{0, 10\}$ and report your training error, training loss and test error for each. For the gradient descent algorithm, plot the training loss over iterations. Compare the running time of the two approaches, which is faster? Overall, which approach do you think is better? Explain why.

Ans:

Exercise 3: Playing with Regression (3 pts)

You may use the Python package scikit-learn for this exercise (and only for this exercise).

Train (unregularized) linear regression, ridge regression, and lasso on the mystery datasets A, B, and C on the course website (using X_train and Y_train for each dataset). For ridge regression and lasso, use regularization parameters 1 and 10 (note that you will have to divide these by n for lasso in scikit-learn – explain why). Report the average mean squared error on the test set for each method. Which approach performs best in each case? Plot the five parameter vectors obtained for each dataset on the same histogram, so they're all visible at once (change the opacity setting for the bars if necessary): specifically, for each parameter vector, plot a histogram of its value in each coordinate.

Ans:

Exercise 4: Nearest Neighbour Regression (7 pts)

1. (3 pts) Implement k-nearest neighbour regression, for a dataset of n X, y pairs where $X \in \mathbb{R}^d$ and $y \in \mathbb{R}$. This is similar to k-nearest neighbour classification, but instead of aggregating labels by taking the majority, we average the y values of the k nearest neighbours. Use ℓ_2 distance as the distance metric, that is, the distance between points X_1 and X_2 is $\|X_1 - X_2\|_2$. Ensure that your implementation is $O(nd)$ time for all values of k, and argue that this is the case.

Ans:

2. (2 pts) For the training sets of the $d = 1$ mystery datasets D and E on the course website, compute a) the (unregularized) least squares linear regression solution and b) the k-nearest neighbour regression solution with each integer k from 1 to 9. For each dataset, on one plot with the x and y axes corresponding to the x and y values, display the least squares solution, the 1-nearest neighbour solution, and the 9-nearest neighbour solution. Be sure to plot these solutions for all points between the minimum and maximum x values, not just for the points in the dataset. On another plot, with k on the x axis and test mean-squared error on the y axis, display the error of the k-NN solution for each value of k. On the same plot, include the test error of the least squares solution as a horizontal line. When does each approach perform better, and why?

   Ans:

3. (2 pts) For the training set of the $d = 20$ mystery dataset F on the course website, compute a) the (unregularized) least squares linear regression solution and b) the k-nearest neighbour regression solution with each integer k from 1 to 9. Plot, with k on the x axis and test mean-squared error on the y axis, display the error of the k-NN solution for each value of k. On the same plot, include the test error of the least squares solution as a horizontal line. Which approach performs better, and why? Hint: consider inspecting distances between the test points and their k nearest neighbours in the training set.

   Ans: