CIS 520, Machine Learning, Fall 2021
Homework 3
Due: Sunday, Oct 3rd, 11:59pm
Submit to Gradescope

Instructions. Please write up your responses to the following problems clearly and concisely. We require you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**

Collaboration. You are allowed and encouraged to work together. You may discuss the **written homework** to understand the problem and reach a solution in groups. However, **it is recommended that each student also write down the solution independently and without referring to written notes from the joint session.** You must understand the solution well enough to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

Assignment Policies

Instructions. Please write up your responses to the following problems clearly and concisely. We require you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**

Learning Objectives

After completing this assignment, you will:

- Understand how linear regression interacts with the number of training examples
- Be able to recognize tradeoffs in performance between OLS and gradient descent linear regression
- Understand how linear regression relates to squared error
- Be able to compute the MLE of any distribution

Deliverables

This homework can be completed individually or in groups of 2. You need to make one submission per group. Make sure to add your team member's name on Gradescope when submitting the homework's written and coding part. Please view the following link if you are not familiar with adding group member's name for Gradescope submission - https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members

1. **A PDF compilation of `hw3.tex` with team member's names in the agreement**
2. **A `.ipynb` file with the functions implemented**

Homework Submission Instructions

Written Homeworks

All written homework **must** be submitted as a PDF to Gradescope. **Handwritten assignments (scanned or otherwise) will not be accepted.** We require the use of LaTeXto generate your final PDF. We will be posting the homeworks in both PDF and LaTeX source form, and we encourage you to use this source as a template for your submission. **We recommend using Overleaf,** a free online LaTeX editor, though you are welcome to edit assignments however you like.

Coding Homeworks

All coding assignments will be done in Jupyter Notebooks. We will provide a `.ipynb` template for each assignment as well as function stubs for you to implement. Your final submission will be a `.ipynb` file compiled from your notebook submitted to Gradescope. Though you are free to use your own installation of Jupyter for the assignments, **we recommend using Google Colab**, which provides a Jupyter environment connected to Google Drive along with a hosted runtime containing a CPU and GPU.

1 Regularization Penalties [25 points]

(This question is manually graded, no need to submit your code). We saw in class that one can use a variety of regularization penalties in linear regression.

$$\widehat{w} = \arg\min_{w} \quad \|y - Xw\|_2^2 + \lambda \|w\|_p^p$$

Consider the three cases, $p = 0$, 1, and 2. (Where, to be precise, the exponent p isn't there for $p = 0$.) We want to know what effect these different penalties have on estimates of w.

Let's see this using a simple problem. Use the provided dataset **q1.pkl**. You can load the data through the following Python code:

```
import pickle
with open('hw3_q1.pkl', 'rb') as f:
    data = pickle.load(f)

print(type(data))
```

```
print(data.keys())
print(data['x'].shape)
print(data['y'].shape)
```

Assume the constant term (bias) in the regression is zero. Assume $\lambda = 1$, except for question (1.1, 1.2). You don't need to write code that solves these problems in their full generality; instead, feel free to use Python to do the main calculations.

1. **[3 points]** If we assume that the response variable is distributed according to $y_i \sim N(w \cdot x_i, \sigma^2)$ (no regularization penalty is needed), then what is the MLE estimate $\widehat{w}_{MLE}$ of w? Write down the closed form solution for w, and then show its value.

2. **[2 points]** Given $\lambda = 2$, what is $\widehat{w}$ for $p = 2$? Write down the closed form solution for w, and then show its value.

3. **[2 points]** Given $\lambda = 1$, what is $\widehat{w}$ for $p = 1$? Feel free to use sklearn's Lasso model or Scipy function *scipy.optimize.fmin*. Write down the value of w.

4. **[4 points]** Given $\lambda = 1$, what is $\widehat{w}$ for $p = 0$? Note that since L0 norm is not a "real" norm, the penalty expression is a little different:

   $$\widehat{w} = \arg\min_{w} \quad \|y - Xw\|_2^2 + \lambda \|w\|_0$$

   Also, for the L_0 norm, you will have to solve the (combinatorially many) cases where different components of w are set to zero, then add the L_0 penalty to each based on the number of features. There are 8 cases for 3 unknown w_i. Write down the value of w.

5. **[5 points]** Write a paragraph describing the relation between the estimates of w in the four cases (i.e. the four estimates of w from the first four parts of this question), explaining why that makes sense given the different penalties.

6. **[9 points]** When $\lambda > 0$, we make a trade-off between minimizing the sum of squared errors and the magnitude of $\widehat{w}$. In the following questions, we will explore this trade-off further. For the following, use the same data from q1.pkl.

   (a) **[1 point]** For the MLE estimate of w (as in 4.1), write down the value of the ratio

   $$||\widehat{w}_{MLE}||_2^2 \;/\; ||y - X\widehat{w}_{MLE}||_2^2.$$

   (b) i. **[3 points]** Suppose the assumptions of linear regression are satisfied. Let's say that with N training samples (assume $N >> P$, where P is the number of features), you compute $\widehat{w}_{MLE}$. Then let's say you do the same, this time with $2N$ training samples. How do you expect $||y - X\widehat{w}_{MLE}||_2^2$ to change when going from N to $2N$ samples? When $N >> P$, does this sum of squared errors for linear regression directly depend on the number of training samples?

   ii. **[3 points]** Likewise, if you double the number of training samples, how do you expect $||\widehat{w}_{MLE}||_2^2$ to change? Does $||\widehat{w}_{MLE}||_2^2$ for linear regression directly depend on the number of training samples in the large-N limit?

   (c) **[2 point]** Using any method (e.g. trial and error, random search, etc.), find a value of λ for which the estimate $\widehat{w}$ satisfies

   $$0.8 < ||\widehat{w}||_2^2 \;/\; ||\widehat{w}_{MLE}||_2^2 < 0.9.$$

2 Feature Selection [27 points]

In this question, we will do both **streamwise regression** and **stepwise regression** to select features for a toy example dataset. Please refer to the lecture slides for detailed algorithms.

The training dataset includes three data samples shown in the following table. Each sample has three features x_1, x_2, x_3, and a truth value y. We will do OLS (ordinary least squares) regression with L_0 regularization as the model to predict the value of y. For simplicity, we only consider the parameter w with no bias term in the OLS model. Therefore, the OLS regression with L_0 regularization model is expressed as

$$\text{argmin}_w ||y - Xw||_2^2 + \lambda ||w||_0$$

	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{y}$
Sample 1	2	4	3.3	5
Sample 2	1	2	1.3	8
Sample 3	1	1	5.8	2

1. **[10 points]** Streamwise regression.

   Please follow the algorithm in the lecture slide. In the simplest version, for each round, w is computed based on all the features in current round model under OLS only without regularization, and then Err is computed based on that w.

   To calculate w, you can reuse the code in HW2. (Please do NOT submit your code.)

   (a) **[6 points]** Try adding each feature in the order of x_1, x_2, x_3. Assume $\lambda = 0.2$, and apply L_0 regularization. Please report your Err_0 (the error of a model with no features) to Err_3, the fitted coefficients of each model, and the final selected feature(s).

   (b) **[2 points]** Now, try adding each feature in the order of x_2, x_3, x_1 with $\lambda = 0.2$ and L_0 regularization. Report the final selected feature(s) (no need to show the intermediate calculations for this part).

   (c) **[2 points]** What do you learn from the results in (a)(b)?

2. **[17 points]** Stepwise regression.

   Please follow the algorithm in the lecture slide. Assume $\lambda = 0.2$, and apply L_0 regularization. In the simplest version, for each round, w is computed based on all the features in current round model under OLS only without regularization, and then Err is computed based on that w.

   Complete the following:

   (a) **[1 point]** Compute the initial Err_{old}

   (b) **[6 point]** Try adding each feature respectively, compute current Err respectively

   (c) **[2 point]** Pick which feature to add to the model and the updated Err_{old}

   (d) **[4 point]** Repeat adding each feature respectively with computed Err, pick features (if any) to add to the model, and report when to halt

   (e) **[2 point]** Report the final selected feature(s)

   (f) **[2 point]** From the results, discuss some of the pros and cons of stepwise regression compared to streamwise regression.

3 Kernel Regression (Programming) [18 points]

In this question, you are going to implement a Kernel Regression model using Gaussian kernel method. You will use **hw3_q3.zip** to do the experiment.

1. **[12 points]** Build the model.

   Given a training dataset $S = ((\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n))$, and kernel function $K(\cdot, \cdot)$, the predicted value $\widehat{y}$ of an input data $\mathbf{x}$ is:

   $$\widehat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x_i}) y_i}{\sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x_i})}$$

   , where $K(\mathbf{x}, \mathbf{x_i}) = \exp(\frac{-||\mathbf{x}-\mathbf{x_i}||_2^2}{\sigma^2})$

   Fill in your code for function **GaussianKernel** and **kernelRegression**. For $\sigma = \{0.01, 0.05, 0.1, 0.15, 0.2, 0.5, 1.0\}$, fill in your code for function **evaluation** to compute the means squared error of the model for each σ value.

   The grading rubric is :

   - **[4 point]** correct **GaussianKernel** function
   - **[6 point]** correct **kernelRegression** function
   - **[2 point]** correct **evaluation** function

2. **[6 points]** Analysis of the model.

   Similar to what you did for HW2, plot a figure for each sigma value of predicted regression line and scatter plot of data points in the test set. Report the sigma value with the smallest MSE for the test set. How the value of sigma affects the kernel regression model?

   The grading rubric is (this part is manually graded):

   - **[2 point]** figures with correct MSE values
   - **[2 point]** sigma value with the smallest MSE
   - **[2 point]** effect of sigma value

4 Gradient Descent on Logistic Regression (Programming) [30 points]

In this question, we will try to use logistic regression to solve a binary classification problem. Given some information of a house, such as area and the number of living rooms, would it be expensive? We would like to predict 1 if it is expensive, and 0 otherwise. We will use the **hw3_house_sales.zip** dataset.

We will first implement it with python Scikit learn package, and then try to implement it by updating weights with gradient descent. We will derive the gradient formula, and use standard gradient descent and AdaGrad to calculate the weights.

1. **[4 points] Logistic regression with Scikit**

   Fill in the `logisticRegressionScikit()` function using the Scikit toolbox. We will not use any penalty here, so set the parameters `penalty = 'none', solver = 'saga'`. You should also specify `fit_intercept=false`, since the constant column is already added in the dataset. Also, we will use

2000 iterations for a fair comparison to later algorithms, so also set the parameter `max_iter=2000`.(3 points).

Report the weights and prediction accuracy here in your submitted PDF file (1 point).

2. **[8 points] Gradient derivation**

   Assume a sigmoid is applied to a linear function of the input features:

$$h_w(x) = \frac{1}{1 + e^{-w^T x}}$$

   Assume also that

$$P(Y = 1|x; w) = h_w(x)$$
$$P(Y = 0|x; w) = 1 - h_w(x)$$

   Calculate the maximum likelihood estimation, then formulate the gradient ascent rule.

   The grading rubric is:

   - **[2 point]** Writing out the log likelihood
   - **[4 point]** Calculating the derivative
   - **[2 point]** Writing out the update formula

3. **[9 points] Logistic regression with standard gradient descent**

   Fill in the `LogisticRegressionGD()` function. To do that, two helper functions `sigmoid_activation()`, to calculate the sigmoid function result, and `model_optimize()`, to calculate the gradient of w, will be needed. Both helper functions can be used in the following AdaGrad optimization function. Use a learning rate of 10^{-4}, run with 2000 iterations. Keep track of the accuracy every 100 iterations in the training set (no need to report). It will be used later.

   Report weights, training accuracy and test accuracy here in your submitted PDF file (1 point).

   The grading rubric is:

   - **[2 point]** correct **sigmoid_activation()** function
   - **[3 point]** correct **model_optimize()** function
   - **[3 point]** correct **LogisticRegressionGD()** function

4. **[4 points] Logistic regression with AdaGrad**

   Fill in the **LogisticRegressionAda()** function. Use a learning rate of 10^{-4}, run with 2000 iterations. Keep track of the accuracy every 100 iterations in the training set (no need to report). It will be used later. (3 points).

   Report weights, training accuracy and test accuracy here in your submitted PDF file (1 point).

5. **[5 points] Comparison of Scikit, GD and AdaGrad convergence**

   Plot the accuracy of GD and AdaGrad over the 2000 iterations on the training set (2 points). What do you observe? Which one has better accuracy on the test dataset (1 point)? Why might that be the case (2 point)?