ME 360, Fall 24 **PROBLEM SET 5** **Due @ 3PM on Friday, Oct 5**

Reading: Class notes: week 4.

Problems:

1. In class, we derived the gradient-descent algorithm for linear regression in the case of scalar real-valued inputs, $\{x^i : i = 1, 2, \ldots, N\}$, and outputs, $\{y^i : i = 1, 2, \ldots, N\}$, without a bias term. As we already did in Homework 4, we will extend this to include an additional bias weight, w_0.

The linear regression model with a bias term is now given by:

$$y = wx + w_0 \tag{1}$$

where w is the slope (weight) and w_0 is the bias.

(a) In Homework 4, you considered the loss function

$$J(w, w_0) = \frac{1}{2N} \sum_{i=1}^{N} \left(y^i - (wx^i + w_0)\right)^2 \tag{2}$$

and showed that

$$\frac{\partial J}{\partial w} = -\frac{1}{N} \sum_{i=1}^{N} \left(y^i - (wx^i + w_0)\right) x^i \tag{3a}$$

$$\frac{\partial J}{\partial w_0} = -\frac{1}{N} \sum_{i=1}^{N} \left(y^i - (wx^i + w_0)\right) \tag{3b}$$

(b) **Compute the second partial derivatives** of $J(w, w_0)$ with respect to w and w_0.

$$\begin{bmatrix} \frac{\partial^2 J}{\partial w_0^2} & \frac{\partial J}{\partial w \partial w_0} \\ \frac{\partial^2 J}{\partial w_0 \partial w} & \frac{\partial^2 J}{\partial w^2} \end{bmatrix} = \begin{bmatrix} 1 & * \\ * & * \end{bmatrix}$$

This is referred to as the Hessian matrix. Compute the determinant of this matrix. If it is positive, then one can conclude that the least square solution (that you obtained in Homework 4) is a minimizer.

Hint: let $v = (1, 1, \ldots, 1) \in \mathbb{R}^N$ and $w = (x^1, x^2, \ldots, x^N) \in \mathbb{R}^N$. From the cosine law, we know that $|v \cdot w| \leq |v| \cdot |w|$ where $|w| = (\sum_{i=1}^{N} (x^i)^2)^{\frac{1}{2}}$ (and $|v|$ is defined similarly) and $v \cdot w = \sum_{i=1}^{N} (1 \cdot x^i)$.

(c) **Write a gradient descent algorithm** for updating the weights w and w_0.

$$w[k+1] = w[k] - *$$
$$w_0[k+1] = w_0[k] - *$$

(d) **Write a stochastic gradient descent (SGD) algorithm** for updating the weights w and w_0.

$$w[k+1] = w[k] - *$$
$$w_0[k+1] = w_0[k] - *$$

Based on your results in the previous problem, write and/or run Python functions for the various algorithms. Example scripts are available on GitHub. You have a choice to:

1. Either code in the functions yourself; or
2. Simply download and run the code that has been provided. If you do so, please make sure that you understand the algorithms as well as their implementations in the Python script. Consult with the TA in case of any questions.

2. Functions for data generation and least square.

(a) **Function for data generation**. Fix the true values of the weights to $w = 1$ and $w_0 = 1$. Therefore, the true model (1) is $y = x + 1$.

Write a Python function to generate noisy samples based on the Algorithm 1. Alternatively, you may download the Python code. Run the function to generate $N = 100$ samples.

Algorithm 1 Noisy data from straight line

```
Initialise $L = 5,\ N = 100, w = 1, w_0 = 1$
Let $\Delta = \frac{2L}{N}$
for $i = 0, 2, \ldots, N-1$ do
    $x[i] = -L + i\Delta$
    Sample $n[i] \sim \mathcal{N}(0, 1)$  ▷ This is noise, a Gaussian random variable with mean zero and variance 1
    $y[i] = wx[i] + w_0 + n[i]$
end for
return $x$ and $y$
```

(b) **Function for computing the least square solution**. Recall the formula for the least square solution

$$w^{\text{opt}} = \frac{\sum_{i=1}^{N}(x^i - \hat{x})(y^i - \hat{y})}{\sum_{i=1}^{N}(x^i - \hat{x})^2}, \quad w_0^{\text{opt}} = \hat{y} - w^{\text{opt}}\hat{x} \tag{4}$$

where $\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x^i$ and $\hat{y} = \frac{1}{N}\sum_{i=1}^{N} y^i$.

Write a Python function for the Algorithm 2. Alternatively, you may download the Python code. Run the function to compute the least square solution based on the data generated in part (a).

Algorithm 2 Lease squares solution

```
Require: $x, y$ from Algorithm 1
Compute $\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x[i]$,   $\hat{y} = \frac{1}{N}\sum_{i=1}^{N} y[i]$  ▷ These computations may be done either using a for loop or the mean function in numpy.
Compute $c_1 = \sum_{i=1}^{N}(x[i] - \hat{x})(y[i] - \hat{y})$ and $c_2 = \sum_{i=1}^{N}(x[i] - \hat{x})^2$  ▷ These computations may be done either using a for loop or the cov function in numpy.
Compute $w^{\text{opt}}$ and $w_0^{\text{opt}}$ using equation (4).
Compute and store $J(w^{\text{opt}}, w_0^{\text{opt}})$ for making plots.
return $w^{\text{opt}}$ and $w_0^{\text{opt}}$
```

For this problem, turn in a scatter plot of the data (in the x-y plane) together with a fit obtained using the least square algorithm.

Algorithm 3 Gradient descent

```
Require: $x, y$ from Algorithm 1
Initialise $w^{\text{gd}} = 0$ and $w_0^{\text{gd}} = 0$ and $K = 50$, $\epsilon = 0.01$
for $k = 1, 2, \ldots, K$ do
    Compute $g_1 = \frac{\partial J}{\partial w}$ and $g_0 = \frac{\partial J}{\partial w_0}$ using equation (3).
    $w^{\text{gd}} \leftarrow w^{\text{gd}} - \epsilon g_1$
    $w_0^{\text{gd}} \leftarrow w_0^{\text{gd}} - \epsilon g_0$
    Compute and store $J(w^{\text{gd}}, w_0^{\text{gd}})$ for making plots.
end for
return $w^{\text{gd}}$ and $w_0^{\text{gd}}$
```

3. In this problem, you will code the gradient descent and the stochastic gradient descent algorithm.

(a) **Function for computing the gradient descent solution**. See Algorithm 3 for pseudo code.

(b) **Function for computing the SGD solution**. See Algorithm 4 for pseudo code. Some explanation for the pseudo code: At the k-th time-step, select a sample (x^i, y^i) randomly from the data set. The gradients at the sample are

$$\left(\frac{\partial J}{\partial w}\right)_i = -\left(y^i - (wx^i + w_0)\right) x^i \tag{5a}$$

$$\left(\frac{\partial J}{\partial w_0}\right)_i = -\left(y^i - (wx^i + w_0)\right) \tag{5b}$$

Algorithm 4 Stochastic gradient descent

```
Require: $x, y$ from Algorithm 1
Initialise $w^{\text{sgd}} = 0$ and $w_0^{\text{sgd}} = 0$ and $K = 50$, $\epsilon = 0.01$
for $k = 1, 2, \ldots, K$ do
    Uniformly sample a random integer $i$ from the set $\{1, 2, \ldots, N\}$.
    Compute $g_1 = (\frac{\partial J}{\partial w})_i$ and $g_0 = (\frac{\partial J}{\partial w_0})_i$ for data point $i$, using equation (5).
    $w^{\text{sgd}} \leftarrow w^{\text{sgd}} - \epsilon g_1$
    $w_0^{\text{sgd}} \leftarrow w_0^{\text{sgd}} - \epsilon g_0$
    Compute and store $J(w^{\text{sgd}}, w_0^{\text{sgd}})$ for making plots.    ▷ Note that in computing $J$ we use all data points.
end for
return $w^{\text{sgd}}$ and $w_0^{\text{sgd}}$
```

Turn in a plot showing the trajectory of the loss function for the gradient descent and the stochastic gradient descent algorithm. The x-axis of the plot is the time k and the y axis is the loss function $J(w[k], w_o[k])$.

In Homework 4, you worked with a dataset of vehicle characteristics that describes a relationship between miles per gallon (MPG), horsepower (HP), and the number of cylinders in the engine. The dataset and example Python scripts are provided, and can be accessed on GitHub. The final problem is to implement the SGD algorithm for this dataset.

4. GD algorithm. A code has been provided that implements the GD algorithm for the binary classifier model and the loss function introduced for the vehicle dataset in Homework 4.

1. In the BinaryClassifier class, an additional tanh activation function is introduced. Explain the purpose of using the tanh function in this context, and why is it useful for binary classification? The hyperbolic tangent function is a smooth approximation of the signum function, and makes the backpropagation in a machine learning algorithm easier.

2. Experiment with different learning rates (10^{-2}, 10^{-4}, 10^{-6}). Turn in a plot showing the trajectory for the loss for the three choices. Discuss the trade-offs between using a high learning rate versus a low learning rate. High learning rate converges faster but it can start oscillating closer to the optimal point. A low learning rate may converge very slowly. Thus a decaying learning rate is typically used.

3. Experiment with two different initial weight vectors (your solutions for Problem 3 in Homework 4). For a fixed learning rate 10^{-4}, run the training process with these two choices. Turn in a plot showing the trajectory for the loss for the two choices. Discuss the effect of weight initialization on the training process. The different weights make the algorithm converge to different points, which are locally optimal, but may not be globally optimal.