

# CMPSCI 390A Homework 2

YOUR NAME HERE

Assigned: Feb 8 2018; Due: Feb 11 2018 @ 11:00 am ET

## Abstract

This assignment is the first part of a two-part assignment on linear regression. In this assignment, you will solve analytically for the partial derivative of a loss function with respect to the model parameters (weights), and you will begin the basic setup of a program to perform linear regression on a provided data set. To submit this assignment, upload a `.pdf` to Gradescope containing your responses to the written response questions below. You are required to use L<sup>A</sup>T<sub>E</sub>X for your write up. We suggest using overleaf.com since they do not require installing anything on your computer and provide free accounts. When submitting your answers, use the template L<sup>A</sup>T<sub>E</sub>X code provided and put your answers below the question they are answering. Do not forget to put your name on the top of the `.pdf`. To submit the assignment's coding portion, upload a single python file called `my_hw2.py`, to the Gradescope programming assignment for Homework 2. An auto-grader will grade your code to check your code for correct output. As such, your program must meet the requirements specified below. We will also be using cheating detection software, so, as a reminder, you are allowed to discuss the homework with other students, but you must write your code on your own.

## 1 Derivatives of a Loss function (50 points)

In this part of the assignment, you will compute the partial derivatives of various functions related to linear regression. Consider the linear parametric model:

$$f_w(x_i) = \sum_{j=1}^m w_j x_{i,j}.$$

One way to solve for the optimal weights,  $w^*$ , is to perform gradient descent on the loss function  $l$  (in this assignment we will consider multiple possible loss functions,  $l_1, l_2$ , etc.). The gradient descent algorithm begins with an initial parameter vector,  $w$ , and then repeatedly performs the following update for all  $j \in \{1, \dots, m\}$ :

$$w_j \leftarrow w_j - \alpha \frac{\partial l(w)}{\partial w_j},$$

where  $\alpha$  is a small constant called the step-size. In the next lecture, we will discuss using the gradient descent algorithm to compute  $w^*$ . In this assignment, you will begin working with some of the terms and tools we will use in that lecture. If you are not familiar with the gradient descent algorithm above and how you can use it to find  $w^*$ , fear not—this will be covered in the lecture, and this assignment will prepare you for that lecture!

Notice that to implement the gradient descent update above, you will need to compute the partial derivative of the loss function with respect to each of the parameters, i.e.,  $\frac{\partial l(w)}{\partial w_j}$ . This part of the assignment aims to familiarize yourself with taking partial derivatives like the ones needed to implement regression algorithms. For each function below, write an expression for the derivative with respect to the weight  $w_j$ . When deriving an answer, you may reuse the answer to a previous question if it is convenient.

To give you an example of what we expect for an answer and how to format it we provide the following example. Consider the function  $g(w) = \sum_{k=1}^m a_k w_k$ , where  $a_k = 1$  if  $k = j$  and 0 otherwise.

$$\frac{\partial g(w)}{\partial w_j} =$$

In formatting your answers, you can use the `&` symbol to align multiple lines (see the source code for the equation block above for an example).

Write expressions for the following derivatives.

1. (10 points)  $f_w(x_i)$

$$\frac{\partial f_w(x_i)}{\partial w_j} =$$

2. (10 points)  $l_1(w) = (y_1 - f_w(x_1))^2$

$$\frac{\partial l_1(w)}{\partial w_j} =$$

3. (10 points)  $l_2(w) = \sum_{i=1}^n f_w(x_i)$

$$\frac{\partial l_2(w)}{\partial w_j} =$$

4. (20 points)  $l_3(w) = \frac{1}{n} \sum_{i=1}^n (y_i - f_w(x_i))^2$

$$\frac{\partial l_3(w)}{\partial w_j} =$$

## 2 Programming

In this section of the assignment, you will be setting up a basic Python script to perform linear regression on a data set. The program you will write in this assignment will load a data set, initialize the weights of a linear model, and make predictions on the data set using the model. You will extend this program to find the optimal model parameters for a data set in the next assignment. An autograder will check your code and thus, needs to meet the specifications described here. Your code should be contained in a single file called `my_hw2.py`. No external dependencies other than NumPy will be allowed. A template file, `my_hw2.py`, and a data set, `hw2_data.csv`, for this assignment can be found in the folder HW2.

The data set in `hw2_data.csv` is synthetic, meaning it does not represent any real-world data. This data set has six attributes and six data points. The first five columns of the file correspond to the features, and the sixth column is the label for that feature vector. Each row of the file corresponds to a different data point, i.e., the first row corresponds to the data point  $(x_1, y_1)$ . The template code already provides a function for loading this data set, so you do not have to worry about parsing the feature vectors and labels from the file.

To complete this assignment, you need to perform the following steps:

1. Complete the Python setup instructions.
2. Familiarize yourself with the code in the template file. There are three functions in the file: `load_dataset`, `predict`, and `main`. The `load_dataset` function parses a `.csv` file and returns two arrays. the first array is a 2D NumPy array with shape  $(n, m)$  containing the list of feature vectors for the data set. See the Python setup instructions for guidelines on interacting with a NumPy array. The second array is a 1D NumPy array of shape  $(n,)$  containing the labels that correspond to each feature vector. The `predict` function takes as input a list of feature vectors and a weight vector and returns a list of predictions for each feature vector. The `main` function loads the data set in the file `hw2_data.csv`, initializes weights,  $w$ , for the linear model, makes predictions of the labels on the data set using weights  $w$ , and then prints out the predictions and actual labels. Not all of these functions are complete, and your task is to implement them correctly.
3. Complete both TODOs in the template file.
  - (a) The first TODO in the template file is to initialize the weights,  $w$ , of the linear model. These weights should be a 1D NumPy array with shape  $(m,)$ , where  $m$  is the length of the feature vectors in the dataset. You must initialize the weights  $w$  as follows:

$$w = [0, 1, \dots, m - 1].$$

Note that this is not a useful initialization, and we only require it for this assignment to make sure your predictions are correct. We will test your code using datasets with different varying lengths of feature vectors, so you cannot hardcode this value.

- (b) The second TODO in the template file is to complete the function `predict`. To complete this function, you need to write code that makes a prediction of the label for each feature vector and store that prediction in the array `predictions`, i.e.,

$$\forall i \text{ predictions}[i] = f_w(x_i).$$

Note that we created the labels for this dataset using the predictions of a correct implementation. So you will know your program is correct when the printed predictions match the labels. In future assignments, predictions will not perfectly match the labels.

4. Submit your `my_hw2.py` file to gradescope.