Assignment 1

ML Class: CS 4375

February 6, 2025

1 Assignment Policies for CS 4375

The following are the policies regarding this assignment.

1. This assignment needs be done individually by everyone.
2. You are expected to work on the assignments on your own. If I find the assignments of a group of (two or more) students very similar, the group will get zero points towards this assignment. You may possibly also be reported to the judiciary committee.
3. Please use Python for writing code. You can submit the code as a Jupyter notebook
4. For the theory questions, you can use Latex or Word. You may also submit scanned copies of what you have written down ASSUMING IT CAN BE CLEARLY READ AND UNDERSTOOD. Also, make sure the scan is of good quality.
5. This Assignment is for 25 points.
6. We will deduct two points for every day you are late!
7. This will be due Friday, February 21st.

2 Questions

1. **Loss Functions for Linear Regression (6 points):** Assume that the hypothesis function is $f(w, b, x) = w^T x + b$. In the standard linear regression case, given an instance x_i, y_i on the training set, the loss function is defined as $L_i(w, b) = [f(w, b, x_i) - y_i]^2$. Imagine that we are still interested in posing the optimization problem (over the dataset) as:

$$\min_{w,b} \sum_{i=1}^{n} L_i(w, b) \tag{1}$$

What if we were to use some slightly different loss functions? Please look at the following four loss functions and answer the questions below:

(a) $L_i(w, b) = |f(w, b, x_i) - y_i|^3$

(b) $L_i(w, b) = [f(w, b, x_i) - y_i]^3$

(c) $L_i(w, b) = \exp[f(w, b, x_i) - y_i]$

(d) $L_i(w, b) = \max(0, 1 - y_i f(w, b, x_i))$

Part 1: Please answer exactly why these loss functions may or may not be a good choice for regression. Also, comment on what approach would you use to solve the corresponding problem (e.g., gradient descent or sub-gradient descent).
Part 2: Compute the gradients or sub-gradients of the loss function in each of the cases above.

Part 1 is for four points and Part 2 is for two points.

2. **Loss Functions for Classification (6 Points):** Again, assume that the function is $f(w, b, x) = w^T x + b$. In the case of SVM and Perceptrons, we saw the following two loss functions: $L_i(w, b) = \max(0, -y_i f(w, b, x_i))$ for Perceptron and $L_i(w, b) = \max(0, 1 - y_i f(w, b, x_i))$ for Hinge Loss (SVM). Similar to question 1, let us see if the following loss functions are good choices.

   (a) $L_i(w, b) = \max(0, 1 - y_i f(w, b, x_i))^2$
   (b) $L_i(w, b) = [y_i - f(w, b, x_i]^4$
   (c) $L_i(w, b) = \exp[f(w, b, x_i) - y_i]$
   (d) $L_i(w, b) = \exp[-y_i f(w, b, x_i)]$

   Part 1: Please answer exactly why these loss functions may or may not be a good choice for classification. Also, comment on what approach would you use to solve the corresponding problem (e.g., gradient descent or sub-gradient descent).
   Part 2: Compute the gradients or sub-gradients of the final loss function in each of the cases above.

   Part 1 is for four points and Part 2 is for two points.

3. **Polynomial and Higher order Features (3 Points):** Consider a dataset where the true relationship between the input x and output y is given by:

$$y = 4x^3 - 2x^2 + 3x + 5 + \epsilon$$

   where ϵ is Gaussian noise.

   (a) If you fit a linear regression model (without polynomial features), what issues might arise?
   (b) What would be the polynomial features you would use, and if you fit a linear model on the polynomial features what would be the optimal weight vector?

4. **Implement Linear Regression from Scratch (5 Points):** Implement a Linear Regression from scratch. Implement two loss functions (i) the mean squared loss (MSE) and (ii) the mean absolute error (MAE)loss. The MSE Loss is $L_i(w, b) = [f(w, b, x_i) - y_i]^2$ and the MAE Loss is $L_i(w, b) = |f(w, b, x_i) - y_i|$.

   Implement a very simple gradient descent algorithm in python. Test your implementation on the house price prediction dataset from the demo on Linear Regression - `https://colab.research.google.com/drive/1myH2V4xbKXZzdF-49ma_vWHLfArTyO2W#scrollTo=XVmWAIEIUsdF`.

   Finally, compare your solution to the solution obtained by sklearn (from the demo) and comment on it briefly. You do not need to consider the more complicated quadratic case. Just the simple linear regression on the features is enough for this.

   You can use sklearn to load the dataset and do the splits but not do anything more (e.g., form the loss function, optimize it, or directly call linear regression from sklearn – you will be comparing to it but you should compare your implementation to sklearn's implementation.

5. **Implement SVM and Perceptrons from Scratch (5 points)** Use the simple dataset (using makeblobs function) that we used in the SVM demo (`https://colab.research.google.com/drive/1z368aCHvKDy92E0dJXtQnTFgF3O1HOWo`). Implement the Perceptron Loss and Hinge Loss functions. Next, implement a simple subgradient descent algorithm and optimize the perceptron and hinge losses. Compare the solutions obtained by the two.