CSCI 374: Homework 1

Due: Sep 21, 2022 at 11:59pm EST

Name: Kelly McCarthy

Instructions: This homework requires answering some open-ended questions, short proofs, and programming. This is an individual assignment, not group work. Though you may discuss the problems with your classmates, you must solve the problems and write the solutions independently. As stated in the syllabus, copying code from a classmate or the internet (even with minor changes) constitutes plagiarism. You are required to submit your answers in pdf form (use LaTeX) in a file called `hw1.pdf` to Gradescope under "HW1". Code should also be submitted to Gradescope in a file called `hw1.py` under "HW1 Programming". A LaTeX template for the pdf submission and a code skeleton for the code submission are available on Piazza. Please do **not** modify the base code in the coding skeleton; simply augment it with your solution. Late submissions do not receive full credit, except in extenuating circumstances such as medical or family emergency. Submissions submitted 0-24 hours late are eligible for only 90%, 24-48 hours late for 80%, 48-72 hours late for 70%, and later than 72 hours for 0% of the total credit for this assignment. Late days may be used (if available) to avoid these penalties.

The total assignment is worth 50 points.

Problem 1 (5 points)

This is an open-ended question to get you a head start on thinking about what you would like to do for your final project. Write a brief description of the kinds of data you have worked with (if you have not worked with data before, it is ok to say so) and would like to work with in this course. What kinds of predictive models would you like to build with such data, and for what purpose? There are no right or wrong answers; a paragraph or two is sufficient.

I would say that most of my experience with working with data has come from my CS courses at Williams. In a 134 lab, I recall working with a dataset that contained the instructors at Williams and their alma maters. Iris's 378 HAII course was probably the most experience I've had with data processing. We filtered through home loan data sets, online dating profiles, among other quantitative and categorical data. Additionally, in an ethnographic research project I conducted in a Public Health course at Williams, I have worked with student response data. This mostly came in "degrees of agreement" in which students responded on a 0-5 scale of their agreement with a given statement. Overall, I do not have an extensive background in data processing but I'm excited to learn! I think the sort of data I would like to work with is a combination of qualitative (both nominal and ordinal) as well as quantitative. I'm not sure if this is too specific, but I am really interested in breast cancer research and other predictive problem-solving approaches. Some predictive models I would like to explore are: forecast models, decision trees, and neural networks. Forecast models appeal to me because I like the idea of trying to predict some future numerical outcome based on historical data. I just started learning more about decision trees in 361 so I think looking into that would be mutually beneficial for my learning in both courses. Neural networks were briefly introduced in 378 and I am just so intrigued to learn more.

Problem 2 (7 points)

(i) Consider the absolute value function defined as,

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -x & \text{if } x < 0. \end{cases}$$

Prove that the absolute value function is differentiable everywhere except at $x = 0$.

Hint: it is sufficient to state what the derivatives are for $x \neq 0$. For $x = 0$, show that the limit that defines the derivative at $x = 0$ does not exist. (5 points)

Here are some helpful LaTeX macros to help you structure your proof:

Proof.

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$= \text{Another notation for derivative} = f'(x) = \lim_{x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Case 1: Approaching from right

$If x > 0$

$y = x$

Work:

$$\lim_{x \to 0} \frac{x}{x} = \lim_{x \to 0} 1 = 1$$

$\underline{Example}$:$(1, 1), (3, 3)$

$$\lim_{x \to 0} \frac{\Delta y}{\Delta x} = \frac{3-1}{3-1} = \frac{2}{2} = 1(constant, positive 1)$$

Case 2: Approaching from left

$If x < 0$

$y = -x$

Work :

$$\lim_{x \to 0} \frac{-x}{x} = \lim_{x \to 0} -1 = -1$$

$\underline{Example}$:$(-1, 1), (-3, 3)$

$$\lim_{x \to 0} \frac{\Delta y}{\Delta x} = \frac{3-1}{-3+1} = \frac{2}{2} = -1(constant, negative 1)$$

Proving that $|x|$ is not differentiable at 0:

$$y' = \frac{df}{dx}(0) = \lim_{x \to 0} \frac{f(x) - f(0)}{x - 0} = \lim_{x \to 0} \frac{|x| - |0|}{x - 0} = \lim_{x \to 0} \frac{|x|}{x}$$

From earlier we know that the left limit = -1 and the right limit = 1.

Since the left limit does not equal the right limit, the limit that defines the derivative at x = 0 DNE.

So, the absolute value is not differentiable at x = 0.

Another important note is that since the absolute value

function has a ”corner” in it, this is often a mark of non-differentiability.

□

(ii) Comment (in a sentence or two) on how this affects the use of the mean absolute deviation as a loss function for machine learning. (2 points)

The mean absolute deviation as a loss function is continuous with respect to its input x, but it is not differentiable at x = 0. An important characteristic of the loss function is that it is numerically easy to work with - smooth, continuous, and differentiable. MAD only satisfies 2 out of these 3 because it is only sub-differentiable, not entirely differentiable. Differentiability is important because it gives the algorithm a "sense of direction" so when we have to use sub-differentiability, we have to designate a "proxy" gradient for the one spot that is not differentiable.

Problem 3 (5 points)

Provide 1 advantage and disadvantage for:

(i) Zero-order optimization methods in comparison to first-order methods
(ii) Second-order optimization methods in comparison to first-order methods

(i) Zero-order optimization methods in comparison to first-order methods
advantage: Since zero-order uses gradient estimations, it is useful for derivatives that are computationally complex and difficult to compute
disadvantage: zero-order methods are computationally inefficient and can take many iterations before finding a good/optimal solution. Using first order methods and following the negative gradient direction is more efficient than random search from zero-order.

(ii) Second-order optimization methods in comparison to first-order methods
advantage: Second order can provide some extra information on how far in the direction of the gradient we should go.
disadvantage: It is expensive to compute the Hessian matrix. This is a matter of computing a d x d matrix as opposed to a d-dimensional gradient vector

Problem 4 (13 points)

This question pertains to the convergence of gradient descent, and is meant to help develop better intuition about why it works. Recall the update rule for gradient descent at each step:

$$\theta^{(i+1)} = \theta^{(i)} - \alpha \nabla L(\theta^{(i)}).$$

Though gradient descent is a first-order method, most analyses of its convergence rely on twice differentiability and constraints on the smoothness of the loss function. For this proof, we assume $L(\theta)$ is twice differentiable and adopt a weak notion of smoothness known as Lipschitz continuity.

A function $L(\theta)$ is said to be Lipschitz continuous if the magnitude of its gradient does not vary too much with respect to its inputs. Formally, $L(\theta)$ is Lipschitz continuous with some constant $K > 0$ if the magnitude of the difference between the gradients for *any* two points $\theta^{(1)}$ and $\theta^{(2)}$ is less than or equal to K times the magnitude of the difference between the points themselves.[1]

Now consider the second-order Taylor approximation of $L(\theta)$ around $\theta^{(i)}$ (this is just a generalization of the Taylor expansion you may already know about to vector-valued inputs):

$$L(\theta) = L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(\theta - \theta^{(i)}) + \frac{1}{2}(\theta - \theta^{(i)})^T \nabla^2 L(\theta)(\theta - \theta^{(i)}).$$

Under our assumptions above, the loss function satisfies the following inequality:

$$L(\theta) \leq L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(\theta - \theta^{(i)}) + \frac{1}{2}(\theta - \theta^{(i)})^T K(\theta - \theta^{(i)}).$$

(i) Use the above relation (plugging in $\theta = \theta^{(i+1)}$ according to the gradient descent update rule) to prove that $L(\theta^{(i+1)}) \leq L(\theta^{(i)})$ if we use any constant learning rate $\alpha \leq 1/K$. That is, the Lipschitz constant K provides an intuitive bound on a learning rate α that will guarantee that the loss decreases (or at worst remains the same) at every step of gradient descent. (10 points)

[1] In math, $||\nabla L(\theta)^{(1)} - \nabla L(\theta)^{(2)}|| \leq K||\theta^{(1)} - \theta^{(2)}||$, for any $\theta^{(1)}, \theta^{(2)}$. Here $||X||$ denotes the magnitude of a vector X defined as the square root of the dot product of X with itself: $\sqrt{X.X}$.

Write your answer here.
Here are some helpful LaTeX macros to help you structure your proof:

Proof.

$$
\begin{aligned}
L(\theta) &\leq L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(\theta - \theta^{(i)}) + \frac{1}{2}(\theta - \theta^{(i)})^T K(\theta - \theta^{(i)}) \\
&\leq L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(\theta^{(i+1)} - \theta^{(i)}) + \frac{1}{2}(\theta^{(i+1)} - \theta^{(i)})^T K(\theta^{(i+1)} - \theta^{(i)}) \\
&\leq L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(\theta^{(i)} - \alpha\nabla L(\theta^{(i)}) - \theta^{(i)}) + \frac{1}{2}(\theta^{(i+1)} - \theta^{(i)})^T K(\theta^{(i+1)} - \theta^{(i)}) \\
&\leq L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(\theta^{(i)} - \alpha\nabla L(\theta^{(i)}) - \theta^{(i)}) + \frac{1}{2}(\theta^{(i)} - \alpha\nabla L(\theta^{(i)}) - \theta^{(i)})^T K(\theta^{(i)} - \alpha\nabla L(\theta^{(i)}) - \theta^{(i)}) \\
&\leq L(\theta^{(i)}) + \nabla L(\theta^{(i)})^T(-\alpha\nabla L(\theta^{(i)})) + \frac{1}{2}(-\alpha\nabla L(\theta^{(i)}))^T K(-\alpha\nabla L(\theta^{(i)})) \\
&\leq L(\theta^{(i)}) + -\alpha(\nabla L(\theta^{(i)})^T(\nabla L(\theta^{(i)}))) + \frac{1}{2}\alpha^2 K(\nabla L(\theta^{(i)}))^T(\nabla L(\theta^{(i)})) \\
&\leq L(\theta^{(i)}) + g(\frac{1}{2}\alpha^2 K - \alpha) \\
&\leq L(\theta^{(i)}) + g(\frac{1}{2}\alpha^2\frac{1}{\alpha} - \alpha) \\
&\leq L(\theta^{(i)}) + g(-\alpha + \frac{\alpha^2}{2\alpha}) \\
&\leq L(\theta^{(i)}) + g(-\alpha + \frac{1}{2}\alpha)
\end{aligned}
$$

The variable 'g' represents the expression $\nabla L(\theta^{(i)})^T(\nabla L(\theta^{(i)}))$. A gradient multiplied by its transpose results in the sum of squares. Since we know that the sum of squares must always be positive, then our variable g will be positive. Therefore, 'g' multiplied by $(-\alpha + \frac{1}{2}\alpha)$ will be a negative number due to the preceding -α.

In conclusion, the expression simplifies and ensures that with some constant learning rate where $\alpha \leq \frac{1}{K}$, the loss decreases (or at the worst remains the same) at every step of gradient descent.

□

(ii) Based on your analysis in (i), propose a learning rate α in relation to K that does not guarantee that the loss function will decrease at the next step. (3 points)

A proposed learning rate that does not guarantee that the loss function will decrease at the next step is $\alpha = \frac{1}{K-1}$. This new learning rate will cause the new α value to be greater than $\frac{1}{K}$. For example, if K was originally 4 then $\frac{1}{K}$ would be equal to $\frac{1}{4}$. However, now that the proposed rate is $\frac{1}{K-1}$ then the new rate is $\frac{1}{3}$ which is larger than $\frac{1}{4}$. If we use an α value that is too large, it is possible that we could overshoot the local minima we are seeking to target with gradient descent. With larger values of α, gradient descent could fail to converge altogether.

Problem 5 (20 points)

In this problem you will implement and apply linear regression based on the code skeleton provided in `hw1.py` and helper functions provided in `data_loader.py`. There are two datasets provided to help test your implementation: an artificially generated dataset in `simulated_data.csv` with features X_1, X_2, X_3 and outcome Y, and a real dataset that examines factors that might be predictive of insurance charges – the features include continuous variables such as age and BMI, and categorical variables such as smoking status and sex. The outcome of interest in the insurance dataset is the charges incurred, i.e., medical costs billed by health insurance.[2]

The helper functions in data loader standardize continuous inputs in the data, converts categorical inputs to dummy 0/1 variables, and adds a column of ones to fit an intercept term. It also performs an 80-20 split of the insurance data into training and test sets.

(i) Implement linear regression using gradient descent.[3] (10 points)

(ii) Implement the R squared metric. Report and briefly comment on the R squared values obtained on the simulated data and the test set of the insurance data. (5 points)

[2] All credit goes to Zach Stednick for scraping and cleaning the data from the public domain: `https://github.com/stedy/Machine-Learning-with-R-datasets`.

[3] Though there exists a closed form solution to gradient descent, we won't use it in this course as no other method we work with will have this property.

R squared simulated data = 0.9635262120732386
R squared insurance data = 0.7584878048782371
These values are what we would generally expect from simulated versus a test set of data. The model performs well with the simulated data, possibly because the data points are more uniform whereas with the test set, there may be some outlying data that causes the model to not produce as high of an R squared value. An R-square of 1, while it measures a perfect 'goodness of fit,' can be very difficult to achieve.

(iii) Assume that you are building this model as a data scientist working for an insurance company. The administration is happy with the predictive accuracy of the model you've created, and are moving for its deployment to real-world settings, where they will use it to decide how much premium to charge to new clients. What are your thoughts on (a) the use of machine learning models (linear regression or otherwise) for this purpose, and (b) incorporating information on predictive but potentially sensitive attributes, e.g., sex, age, BMI, and race, in these models? Feel free to use the fitted weights of your model as a guide for how the outputs of the model would change to guide your discussion. Like question 1, there is not a single right way to answer this question, I am mostly looking for consistent arguments (for or against.) (5 points)

This particular question and the activity as a whole have been very cool to explore. Interestingly enough, I wrote a paper last spring for my Science and Technology Studies course "Critical Introduction to STS" about the BMI metric. I do not agree with the use of machine learning models in instances of deciding how much premium to charge new clients. We have denoted attributes such as sex, age, BMI, and race as "protected characteristics" so when we are leveraging them in a mathematical predictive model, we are inherently not truly "protecting" anything. Last fall I did a Health Care Bias Lab for my final project in my Politics of Algorithms course. When race was used in the model, it overall categorized black patients in a lower risk percentile than white patients, indicating a lesser degree of "need" within the healthcare system. However, this was a biased estimation because there are several variables that influence health expenditures (discrimination, doctor-patient relationship, and outdated ideas of what "health" is). One very outdated metric is the BMI, which is based on the average male body type in the 1830s. Not only has that male body-type changed but it is unreasonable to use it as a metric for other genders. My other concern is the data sets we use when training theses machine learning models to decide how much premium to charge new clients. If they are not representative of the different types of clients (occupation, geographic location, familial dependencies, etc.) then they may favor certain individuals over others. This point may be especially exacerbated if we do use attributes such as sex, age, or race because some groups may have had higher insurance premiums in the past because of discriminatory practices or there is not enough data points to robustly fit all groups.

Important submission note:
Report the numbers you computed (just the Rsquared numbers, not the weights) and answers to the above questions in the PDF that you turn in. Hand in your Python code separately on Gradescope under HW 1 Programming. Please do **not** modify the existing code in the skeleton; simply augment it with your solution. Please make sure your variables and functions are aptly named, and your code is clear and readable, with comments where appropriate. I may try to execute your code on a new dataset as a test case, but don't worry too much about making your code robust in terms of error checking and adversarial inputs – this new dataset will look very similar to the one that's provided.