Homework Assignment # 3

Due: Friday, March 11, 2022, 11:59 p.m.
Total marks: 100

Question 1. [25 MARKS]

In Assignment 2, you learned $p(y|x, w) = \mathcal{N}(\mu = xw, \sigma^2)$ where we assumed fixed variance $\sigma^2 = 1$. Now let's assume that $p(y|x, \mathbf{w}) = \mathcal{N}(\mu = xw_1, \sigma^2 = \exp(xw_2))$ for $\mathbf{w} = (w_1, w_2)$. The objective is the negative log-likelihood, written as a sum over all datapoints in dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$:

$$c(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} c_i(\mathbf{w}) \quad \text{where } c_i(\mathbf{w}) = -\ln p(y_i|x_i, \mathbf{w})$$

(a) [5 MARKS] Compute the gradient of c_i. Show your steps.

(b) [5 MARKS] Let $\mathbf{w}_t = (w_{t,1}, w_{t,2})$ be the weights on iteration t. Write the stochastic gradient descent update, with a mini-batch size of 1 (one sample), for a given sample (x_i, y_i).

(c) [5 MARKS] Unlike all the other objectives we have considered, this objective is non-convex. Explain why that is a problem in a couple of sentences.

(d) [5 MARKS] It is useful to reason about the behavior of our learning systems. Let Algorithm 1 be the (mini-batch SGD) algorithm to learn w for $p(y|x, w) = \mathcal{N}(\mu = xw, \sigma^2 = 1)$, and let Algorithm 2 be the (mini-batch SGD) algorithm to learn $\mathbf{w}$ for $p(y|x, \mathbf{w}) = \mathcal{N}(\mu = xw_1, \sigma^2 = \exp(xw_2))$. Imagine we run them both on Monday (with random seed 1) and w converges to $w = 0.1$ and $\mathbf{w}$ converges to $\mathbf{w} = (0.1, 0.05)$. Then we run them both on Tuesday (with random seed 2) and again w converges to $w = 0.1$ but now $\mathbf{w}$ converges to $\mathbf{w} = (0.04, 0.1)$! How is this possible?

(e) [5 MARKS] It is also useful to reason about how to model our problem. Provide one reason why it might be preferable to learn $p(y|x, \mathbf{w}) = \mathcal{N}(\mu = xw_1, \sigma^2 = \exp(xw_2))$ rather than $p(y|x, w) = \mathcal{N}(\mu = xw, \sigma^2 = 1)$.

Question 2. [55 MARKS]

In this question, you will implement multivariate linear regression, and polynomial regression, and learn their parameters using mini-batch Gradient Descent. You will implement 3 different stepsize adaptation rules—ConstantLR, HeuristicLR, and AdaGrad—and compare their performance. You will implement linear regression first, with all the stepsize adaptation rules. Then you will use this implementation to do polynomial regression, by creating polynomial features and then calling the linear regression procedure. You will compare linear regression and polynomial regression with a constant learning rate. Then you will compare the three different stepsize adaptation rules, for polynomial regression where stepsize adaptation matters even more than linear regression. Initial code has been given to you in a notebook, called `A3.jl`, to run the regression algorithms on a dataset. Detailed information for each question is given in the notebook. Baseline algorithms (random and mean) are sanity checks; we should be able to outperform them.

You will be running the algorithms on the GraduateAdmissions_v1.0 data set on Kaggle, which has $n = 500$ samples and $d = 7$ features. The features are composed of GPA, TOEFL grades and a few other criteria. You are asked to train some models to predict the admission probability based

on these features. The features are augmented to have a column of ones (to create the bias term), in `A3.jl` (not in the data file itself).

(a) [5 MARKS] Implement the epoch function in the specified cell, needed to do mini-batch gradient descent.

(b) [5 MARKS] Implement the mean squared error loss function, to use for linear regression with mini-batch gradient descent.

(c) [5 MARKS] Implement the gradient of the mean squared error loss function, for a given mini-batch.

(d) [5 MARKS] Fill in the necessary implementation details of the update rule for mini-batch gradient descent using a constant learning rate under `ConstantLR`.

(e) [5 MARKS] Implement the update rule for mini-batch gradient descent using the heuristic learning rate under `HeuristicLR`, described in the notebook.

(f) [10 MARKS] Implement the update rule for mini-batch gradient descent using AdaGrad under `AdaGrad`.

(g) [10 MARKS] Complete the part in the code needed to implement polynomial features with a $p = 2$ degree polynomial to create polynomial regression.

(h) [5 MARKS] Compare the `Linear` vs. `Polynomial` models and report their mean errors on the admissions dataset.

(i) [5 MARKS] Compare the different learning rate rules on the `LinearModel` and provide 1 sentence commenting on the results.

Question 3. [20 MARKS]

Suppose you are rating apples for quality, to ensure the restaurants you serve get the highest quality apples. The ratings are $\{1, 2, 3\}$, where 1 means bad, 2 is ok and 3 is excellent. But, you want to err on the side of giving lower ratings: you prefer to label an apple as bad if you are not sure, to avoid your customers being dissatisfied with the apples. Better to be cautious, and miss some good apples, than to sell low quality apples.

You decide to encode this into the cost function. Your cost is as follows

$$\text{cost}(\hat{y}, y) = \begin{cases} |\hat{y} - y| & \hat{y} \leq y \\ 2|\hat{y} - y| & \hat{y} > y \end{cases} \tag{1}$$

This cost is twice as high when your prediction for quality $\hat{y}$ is greater than the actual quality y. The cost is zero when $\hat{y} = y$. To make your predictions, you get access to a vector of attributes (features) $\mathbf{x}$ describing the apple.

(a) [10 MARKS] Assume you have access to the true distribution $p(y|\mathbf{x})$. You want to reason about the optimal predictor, for each $\mathbf{x}$. Assume you are given a feature vector $\mathbf{x}$. Define

$$c(\hat{y}) \doteq \mathbb{E}[\text{cost}(\hat{y}, Y)|\boldsymbol{X} = \mathbf{x}]$$

Let $p_1 = p(y = 1|\mathbf{x})$, $p_2 = p(y = 2|\mathbf{x})$ and $p_3 = p(y = 3|\mathbf{x})$. Write down $c(\hat{y})$ for each $\hat{y} \in \{1, 2, 3\}$, in terms of p_1, p_2, p_3.

(b) [10 MARKS] Now notice that the optimal predictor is $f^*(\mathbf{x}) = \arg\min_{\hat{y} \in \{1,2,3\}} c(\hat{y})$. In practice,

you won't have access to $p(y|\mathbf{x})$, but you can approximate it. Imagine you have a procedure to learn this p (it is not hard to do, the algorithm is called multinomial logistic regression). Assume you can query this learned function, `phat` for an input vector `x`. This function returns a 3-dimensional vector of the estimated probabilities $\hat{p}_1, \hat{p}_2, \hat{p}_3$ for $y = 1, y = 2, y = 3$ respectively given `x`. Your goal is to return predictions using $f(\mathbf{x}) = \arg\min_{\hat{y} \in \{1,2,3\}} \hat{c}(\hat{y})$ where $\hat{c}$ is the same as c that you derive in part a, but with the true p replaced with our estimates $\hat{p}$.

Write a piece of pseudocode that implements this f, namely that inputs `x` and returns a prediction $\hat{y}$. Err on the side of the pseudocode being more complete. The goal of this question is to show that you can go from an abstract description of our predictor to a more concrete implementation.

Homework policies:

Your assignment should be submitted as two pdf documents and a .jl notebook, on eClass. There is no need to submit the 3 into a zip file. One pdf is for the written work, the other pdf is generated from .jl notebook. The pdf version of .jl notebook can be generated by clicking the button on the top-right corner of the notebook and choose an option.

Ths first pdf containing your answers of the write-up questions must be written legibly and scanned or must be typed (e.g., Latex). This .pdf should be named Firstname_LastName_Sol.pdf, For your code, we want you to submit it both as .pdf and .jl. To generate the .pdf format of a Pluto notebook, you can easily click on the circle-triangle icon on the right top corner of the screen, called Export, and then generate the .pdf file of your notebook. The .pdf of your Pluto notebook as Firstname_LastName_Code.pdf while the .jl of your Pluto notebook as Firstname_LastName.jl. All code should be turned in when you submit your assignment.

Because assignments are more for learning, and less for evaluation, grading will be based on coarse bins. **The grading is atypical**. For grades between (1) 80-100, we round-up to 100; (2) 60-80, we round-up to 80; (3) 40-60, we round-up to 60; and (4) **0-40, we round down to 0**. The last bin is to discourage quickly throwing together some answers to get some marks. The goal for the assignments is to help you learn the material, and completing less than 50% of the assignment is ineffective for learning.

We will not accept late assignments. Plan for this and aim to submit at least a day early. If you know you will have a problem submitting by the deadline, due to a personal issue that arises, please contact the instructor as early as possible to make a plan. If you have an emergency that prevents submission near the deadline, please contact the instructor right away. Retroactive reasons for delays are much harder to deal with in a fair way.

All assignments are individual. All the sources used for the problem solution must be acknowledged, e.g. web sites, books, research papers, personal communication with people, etc. Academic honesty is taken seriously; for detailed information see the University of Alberta Code of Student Behaviour.

Good luck!