Homework Assignment # 2

Due: Friday, March 12, 2021, 11:59 p.m. Mountain time
Total marks: 100

Question 1. [25 MARKS]

Imagine that you would like to predict if your favorite table will be free at your favorite restaurant. The only additional piece of information you can collect, however, is if it is sunny or not sunny. Therefore, you would like to predict whether the table will be free or not given the weather. You collect paired samples from visit of the form (is sunny, is table free), where it is either sunny (1) or not sunny (0) and the table is either free (1) or not free(0).

(a) [10 MARKS] How can this be formulated as a maximum likelihood problem? Explain what the distributions are, what parameters need to be learned and write the (log) likelihood explicitly for those distributions and parameters. You do not need to solve this maximum likelihood problem.

(b) [10 MARKS] Assume you have collected data for the last 10 days and computed the maximum likelihood solution to the problem formulated in (a). You do not actually have to do this, just assume that you did and now have estimated the parameter for your distribution. If it is sunny today, how would you predict if your table will be free?

(c) [5 MARKS] Imagine you could further gather information about if it is morning, afternoon, or evening. How does this change the maximum likelihood problem? You do not need to write the log likelihood explicitly for this question, just explain how the distributions and parameters change.

Question 2. [20 MARKS]

Assume that X is a random variable with density corresponding to an equal mixture of two Gaussians, with unknown means μ_1, μ_2 and unknown variances σ_1, σ_2:

$$p(x) = 0.5\mathcal{N}(\mu_1, \sigma_1^2) + 0.5\mathcal{N}(\mu_2, \sigma_2^2). \tag{1}$$

Assume you are given a dataset of n iid samples from this distribution: $\mathcal{D} = \{x_i\}_{i=1}^n$. Your goal is to estimate μ_1, μ_2, σ_1 and σ_2.

(a) [10 MARKS] Write down the negative log-likelihood for this problem, for the given dataset $\mathcal{D} = \{x_i\}_{i=1}^n$. Simplify as far as you can, by explicitly writing the densities for a Gaussian. Hint: You will not be able to simplify very far, and you will be stuck with a few exponentials that the logs cannot cancel.

(b) [10 MARKS] Derive update rules to estimate μ_1, μ_2, σ_1, and σ_2. More specifically, derive the gradient descent update rule, for the negative log likelihood you provided above.

Question 3. [55 MARKS]

In this question, you will implement an algorithm to estimate $p(y|x)$, for a batch of data of pairs of (x, y): $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where both $x_i, y_i \in \mathbb{R}$. We have provided you with a simple code-base in python, to load data and run an algorithm to estimate the parameters for $p(y|x)$. The main script is called `script_regression.py`. It uses a simplified Kaggle data-set[1], where we use only the weight (x) and height (y) of 10,000 people. The provided data-loader function splits the

[1] `https://www.kaggle.com/mustafaali96/weight-height`

data-set into a training set and testing set, after centering the variables so that the mean values for random variables x, and y will be zero. The python file `regressionalgorithms.py` has two baseline algorithms, for comparison: a mean predictior and random predictor. These baselines are sanity checks: your algorithm should be able to outperform random predictions, and the mean value of the target in the training set. When using `load_height_weight` function, set `trainsize` and `testsize` equal to 4000 and 1000 respectively. Because of randomization in some of the learning approaches, we run each of the algorithms 50 times for different random training/test splits, and report the average error and standard error, to see how well each algorithm averaged across splits.

(a) [10 MARKS] Let's start simple, and assume $p(y|x) = \mathcal{N}(\mu = bx, \sigma^2 = 1.0)$ for an unknown weight $b \in \mathbb{R}$. For fun, you can derive the stochastic gradient descent update for the negative log-likelihood for this problem. But, we also provide it for you here. For a randomly sampled (x_i, y_i), the stochastic gradient descent update to b is:

$$b_{t+1} = b_t - \eta \left(x_i b_t - y_i\right) x_i \tag{2}$$

You will run this iterative update by looping over the entire dataset multiple times. Each pass over the entire dataset is called an *epoch*. At the beginning of each epoch, you should randomize the order of the samples (however, for this assignment, if you leave out this step, you will not get penalized). Then you iterate over the entire dataset, updating with Equation (2). Implement this algorithm to estimate b, by completing the code in function `SimpleRegression`. Set the number of epochs to 10. The given barebones code already has epochs as part of the parameters in `SimpleRegression`. The default value is set to 1, but you can change the number of epochs in `script_regression` by passing the desired parameter value of 10.

(b) [5 MARKS] Run `script_regression.py`, with algorithms `Random`, `Mean` and `SimpleRegression`. Set the stepsize for `SimpleRegression` to 0.01 and report the average performance and standard deviation of the error for each of the three approaches.

(c) [10 MARKS] Next implement a mini-batch approach to estimate b, in `BatchSimpleRegression`. The idea is similar to stochastic gradient descent, but now you use blocks (or mini-batches) of N_{batch} samples to estimate the gradient for each update. For each epoch, you iterate over the dataset in order. For the first mini-batch update, the update equation is

$$b_{t+1} = b_t - \eta g_t \qquad \text{where } g_t = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} \left(x_i b_t - y_i\right) x_i \tag{3}$$

Then the next update uses the next mini-batch of points, $(x_{N_{\text{batch}}+1}, y_{N_{\text{batch}}+1}), \ldots, (x_{2N_{\text{batch}}}, y_{2N_{\text{batch}}})$. In total, for one epoch, you will complete n/N_{batch} updates. (Note that the final mini-batch might be smaller than N_{batch} if the number of samples is not divisible by N_{batch}, but for you we have made sure it is $4000/32 = 125$). In `script_regression.py`, set the stepsize equal to $\eta = 0.01$ and the batch size equal to $N_{\text{batch}} = 32$.

(d) [10 MARKS] Next you will implement a more sensible strategy to pick stepsizes. You will implement the heuristic for adaptive stepsizes given in the notes

$$\eta_t = \left(1 + |g_t|\right)^{-1} \tag{4}$$

where g_t is the gradient in the update $w_{t+1} = w_t - \eta_t g_t$. Implement this heuristic both for `SimpleRegression` and `BatchSimpleRegression`. Note for `SimpleRegression`, the update uses $g_t = (x_i b_t - y_i) x_i$ and for `BatchSimpleRegression` the update uses $g_t = \frac{1}{N_{\text{batch}}} \sum_{i=1}^{N_{\text{batch}}} (x_i b_t - y_i) x_i$.

(e) [5 MARKS] Run the mini-batch approach and the stochastic approach, both with the adaptive

stepsizes given by Equation (4). Compare the performance of them before and after using adaptive stepsize selection strategies based on the average error after 10 epochs.

(f) [10 MARKS] Now let's also estimate the conditional variance, rather than assuming it is 1.0. Let's assume that $p(y|x) = \mathcal{N}(\mu = xb, \sigma^2 = \exp(xa))$. Let $w = (a, b)$ and $w_t = (a_t, b_t)$ be the weights on iteration t. If you write down the maximum likelihood problem, and derive the update for both b and a, you get the following update for a randomly sampled x_i, y_i:

$$\frac{\partial c(w_t)}{\partial b} = \frac{-x_i(y_i - x_i b_t)}{\exp(x_i a_t)} \tag{5}$$

$$\frac{\partial c(w_t)}{\partial a} = 0.5 x_i (1 - (y_i - x_i b_t)^2 \exp(-x_i a_t)) \tag{6}$$

where $c(w)$ is the loss function, proportional to the negative log likelihood for this problem. We can use the same heuristic stepsize as above, where now the gradient has two elements, to get the following update equations

$$\eta_t = \left(1 + \sqrt{\left(\frac{\partial c(w_t)}{\partial b}\right)^2 + \left(\frac{\partial c(w_t)}{\partial a}\right)^2}\right)^{-1} \tag{7}$$

$$b_{t+1} = b_t - \eta_t \frac{\partial c(w_t)}{\partial b} \tag{8}$$

$$a_{t+1} = a_t - \eta_t \frac{\partial c(w_t)}{\partial a} \tag{9}$$

Implement these updates in `DistributionRegression` class.

(g) [5 MARKS] Report the performance of `SimpleRegression` and `DistributionRegression`, by reporting the average error and standard error after 1 epoch and after 10 epochs. Additionally, compute the runtimes of the two methods, across the entire experiment, and report these runtimes.

Homework policies:

Your assignment should be submitted as a single pdf document and a zip file with code, on eClass. The answers must be written legibly and scanned or must be typed (e.g., Latex). 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!