Homework 3

10-605/10-805: Machine Learning with Large Datasets

Due Tuesday, October 4th at 11:59 PM Eastern Time

Instructions: Submit your solutions via Gradescope, *with your solution to each subproblem on a separate page*, i.e., following the template below. Note that Homework 3 consists of two parts: this written assignment, and a programming assignment. The written part is worth **60%** of your total HW3 grade. The programming part makes up the remaining 40%.

Submitting via Gradescope: When submitting on Gradescope, you must assign pages to each question correctly (it prompts you to do this after submitting your work). This significantly streamlines the grading process for the course staff. Failure to do this may result in a score of 0 for any questions that you didn't correctly assign pages to. It is also your responsibility to make sure that your scan/submission is legible so that we can grade it.

1 Written Section [60 Points]

1.1 Count-Min Sketch Basics (15 pts)

(a) Let's use count-min sketch to estimate the counts of a stream of 5 items: x_1, x_2, x_3, x_4, x_5, using 3 hash functions: h_1, h_2, h_3, each with 3 buckets. The hash functions have the following collision matrix:

collision matrix			
h_1	x_1, x_3	x_4, x_5	x_2
h_2	x_2, x_5	x_3	x_1, x_4
h_3	x_1	x_2, x_3	x_4, x_5

Suppose you see the following stream of data: $\{x_1, x_2, x_5, x_3, x_3, x_4, x_4, x_4\}$.

i. *[2 points]* What is the value of $\hat{c}_1$, the approximate count of item x_1, based on this sketch?

ii. *[2 points]* What is the value of $\hat{c}_4$, the approximate count of item x_4, based on this sketch?

iii. *[5 points]* Let c_s be the true count of item x_s. What is the average absolute error of approximation for all items, i.e. $\frac{1}{5}\sum_{s=1}^{5} |\hat{c}_s - c_s|$?

(b) Consider the approximation guarantee for count-min sketch presented in lecture: given $r > \log_2(\frac{1}{\delta})$ hash functions, each with $m > \frac{2}{\epsilon}$ buckets, then

$$P\left(\hat{c}_s \geq c_s + \epsilon\|c\|_1\right) \leq \delta$$

i. *[3 points]* How would the number of required buckets change if our tolerance on the discrepancy ϵ is halved?

ii. *[3 points]* How would the number of hash functions change if we allowed the probability δ to be twice as large?

1.2 An Unbiased Sketch (30 pts)

The count-min sketch presented in lecture is biased, as the estimated count $\hat{c}_s$ for item $s \in \{1, \dots, k\}$ is always higher than (or equal to) the true count c_s. In this problem, we will explore an *unbiased* sketch.

Let g be a hash function chosen from a family G of independent hashes, such that g maps each item s to either $+1$ or -1 with equal probability:

$$P(g(s) = +1) = P(g(s) = -1) = 1/2.$$

Let's start with a simple version of the sketch, which uses one additional hash function, h, which maps each item s to one of m buckets. Assume that g and h are independent. When we observe an item s in the sequence, we simply update the sketch counter, C, as:

$$C[h(s)] = C[h(s)] + g(s).$$

Then, if we would like to predict the count for item s, we return:

$$\hat{c}_s = C[h(s)]\ g(s).$$

Given this sketch, please answer the following questions:

a) *[8 points]* First, show that this simple sketch is unbiased, i.e., $\mathbb{E}[\hat{c}_s] = c_s$. (*Hint: Find an expression for $\hat{c}_s$ in terms of $g(s)$ and c_s, and use linearity of expectation.*)

b) *[8 points]* Let's understand how much the estimates vary. Prove that: $Var(\hat{c}_s) = \frac{1}{m}\sum_{j\in\{1,\dots,k\},j\neq s} c_j^2$, where m is the number of buckets for the hash function h.

c) *[6 points]* We will now bound the probability of getting a bad estimate $\hat{c}_s$, i.e., one that differs substantially from the true count c_s. Show that:

$$P(|\hat{c}_s - c_s| \geq \epsilon \|\mathbf{c}_{-s}\|_2) \leq \frac{1}{\epsilon^2 m},$$

where $\mathbf{c}_{-s}$ is a vector of length $(k-1)$ containing the counts for all items except s. *(Hint: Consider Chebyshev's inequality.)*

d) Similar to count-min sketch, we can improve the results from (c) by using r hash functions for mapping the items to buckets $(h_1, \ldots, h_r)$, and r hash functions for assigning signs $(g_1, \ldots g_r)$. In particular, setting $m > \frac{3}{\epsilon^2}$ and $r > (log(1/\delta))$, we can guarantee:

$$P(|\hat{c}_s - c_s| \geq \epsilon \|\mathbf{c}_{-s}\|_2) \leq \delta\,.$$

Note that while this result appears similar to the one we had for count-min sketch, it requires more buckets, m. Recall that for count-min sketch with $m > \frac{2}{\epsilon}$ and $r > (log(1/\delta))$, we had the bound:

$$P(|\hat{c}_s - c_s| \geq \epsilon \|\mathbf{c}\|_1) \leq \delta\,.$$

With these results in mind, answer the questions below.

i. *[4 points]* Explain a scenario where count-min sketch would be preferable to the unbiased sketch we derived (using the above bounds).

ii. *[4 points]* Explain a scenario where the unbiased sketch we derived would be preferable to count-min sketch (using the above bounds).

1.3 Locality-Sensitive Hashing (15 pts)

In lecture, we saw that LSH can be used for applications such as nearest neighbors classification and clustering. We mentioned that if two points $\mathbf{x}$ and $\mathbf{y}$ have small *cosine distance*, then the resulting bit vectors formed by using LSH, $\mathbf{x}'$ and $\mathbf{y}'$, will have small *Hamming distance*. In this problem, we will explore this result in more detail.

First, let's recall the definitions of cosine similarity, cosine distance, and Hamming distance.

Cosine similarity measures the cosine of the angle between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, and can be calculated as:

$$s_{cos}(\mathbf{x}, \mathbf{y}) = \cos(\theta(\mathbf{x}, \mathbf{y})) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2}$$

Cosine distance is measured as $d_{cos}(\mathbf{x}, \mathbf{y}) = 1 - s_{cos}(\mathbf{x}, \mathbf{y})$.

Hamming distance between two *binary* (0-1) vectors $\mathbf{x}', \mathbf{y}'$ of length k measures the number of differing bits, and is calculated as:

$$d_h(\mathbf{x}', \mathbf{y}') = \sum_{i=1}^{k} |\mathbf{x}'_i - \mathbf{y}'_i|$$

(a) *[1 points]* Define $\mathbf{a} = [0, 0, 1, 1, 0]$, $\mathbf{b} = [0, 1, 0, 0, 1]$. What's the *cosine similarity* between $\mathbf{a}$ and $\mathbf{b}$?

(b) *[1 points]* Define $\mathbf{a} = [0, 0, 1, 1, 0]$, $\mathbf{b} = [0, 1, 0, 0, 1]$. What's the *Hamming distance* between $\mathbf{a}$ and $\mathbf{b}$?

(c) *[3 points]* For a random hyperplane, $\mathbf{z}$, define the following hash function:

$$h_z(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \mathbf{z} \cdot \mathbf{x} \geq 0 \\ 0 & \text{if} \quad \mathbf{z} \cdot \mathbf{x} < 0 \end{cases}$$

For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^k$, prove that:

$$P[h_z(\mathbf{x}) = h_z(\mathbf{y})] = 1 - \frac{\theta(\mathbf{x}, \mathbf{y})}{\pi},$$

where $\theta(\mathbf{x}, \mathbf{y})$ is the angle between vectors $\mathbf{x}, \mathbf{y}$ measured in radians.

To prove this, use the fact that $P[\mathbf{z} \cdot \mathbf{x} \geq 0, \mathbf{z} \cdot \mathbf{y} < 0] = \frac{\theta(\mathbf{x}, \mathbf{y})}{2\pi}$, i.e., the probability that a random hyperplane separates two vectors is proportional to the angle between them.

(d) *[6 points]* In LSH, we estimate $P[h_z(\mathbf{x}) = h_z(\mathbf{y})]$ by selecting b random hyperplanes, and calculating bit vectors $\mathbf{x}'_i = h_i(\mathbf{x})$ and $\mathbf{y}'_i = h_i(\mathbf{y})$ for $i = 1, \ldots, b$. Suppose that after running this procedure, we compute the Hamming distance and find: $d_h(\mathbf{x}', \mathbf{y}') = h$. Using this estimate and part (c), show that:

$$d_{cos}(\mathbf{x}, \mathbf{y}) \approx 1 - \cos\left(\frac{h}{b}\pi\right)$$

Note that this equation shows that if $h = b$, i.e., the bit vectors have a large hamming distance, the cosine distance will also be large ($d_{cos}(\mathbf{x}, \mathbf{y}) = 2$). Similarly, for $h = 0$, we will have $d_{cos}(\mathbf{x}, \mathbf{y}) = 0$. (*Hint: The proof should only be a few lines.*)

(e) Let's go through a simple example using LSH. Suppose we have two vectors, $\mathbf{x} = [3, 4, 5, 6]$ and $\mathbf{y} = [4, 3, 2, 1]$.

i. *[1 points]* Use one random vector: $\mathbf{z}_1 = [0.16, -0.26, -1.12, 0.01]$ to perform LSH. Compute the error of the resulting estimated cosine distance, i.e., $|1 - \cos\left(\frac{h}{b}\pi\right) - d_{cos}(\mathbf{x}, \mathbf{y})|$.

ii. *[1 points]* Now let's try with more vectors. Use four random vectors: $\mathbf{z}_1 = [0.16, -0.26, -1.12, 0.01]$, $\mathbf{z}_2 = [1.08, -1.14, 0.8, -0.8]$, $\mathbf{z}_3 = [1.94, 0.08, 1.83, 0.17]$, $\mathbf{z}_4 = [-0.88, -1.01, 1.36, -0.07]$ to perform LSH. Compute the error of the estimated cosine distance.

iii. *[2 points]* In practice, it can be sufficient to consider random vectors only consisting of $\{-1, +1\}$ when performing LSH. Use the signs of the previous random vectors, i.e., $\mathbf{z}_1 = [+1, -1, -1, +1]$, $\mathbf{z}_2 = [+1, -1, +1, -1]$, $\mathbf{z}_3 = [+1, +1, +1, +1]$, $\mathbf{z}_4 = [-1, -1, +1, -1]$, to perform LSH. Compute the error of the estimated cosine distance.

2 Programming Section [40 Points]

2.1 Introduction

This assignment involves training a logistic regression model with high-dimensional data using OHE and feature hashing.

You will build a machine learning pipeline to apply logistic regression to a high-dimensional, click-through-rate prediction problem.

3 Logistics

We provide the code template for this assignment in *one* Jupyter notebook. What you need to do is to follow the instructions in the notebook and implement the missing parts marked with '`<FILL_IN>`' or '`# YOUR CODE HERE`'. Most of the '`<FILL_IN>/YOUR CODE HERE`' sections can be implemented in just one or two lines of code.

3.1 Getting lab files

You can obtain the notebooks `'hw3.ipynb'` in the homework 3 handout .zip file.

Next, import the notebook into your Databricks account, which provides you a well-configured Spark environment and will definitely save your time (see the next section for details).

3.2 Preparing for submission

We provide several public tests via `assert` in the notebook. You may want to pass all those tests before submitting your homework. You can individually submit a notebook for debugging but **make sure to submit the notebook for your final submission to receive full credit.**

In order to enable auto-grading, please do not change any function signatures (e.g., function name, parameters, etc) or delete any cells. If you do delete any of the provided cells (even if you re-add them), the autograder will fail to grade your homework. If you do this, you will need to re-download the homework files and fill in your answers again and resubmit.

3.3 Submission

1. Export solution notebook as an IPython notebook file on Databricks via `File -> Export -> IPython Notebook`
2. Submit completed notebook via Gradescope.

3.4 Setting up environments on Databricks

We provide step-by-step instructions on how to configure your Databricks platform. We will also introduce it in detail during the recitation on September 2nd. Here are the links to the recitation 1 materials for your convenience: Recitations 1 Slides, Lab Notebook.

1. Sign up for the **Community Edition** of Databricks here: https://one.com/tr-databricks.
2. Import the notebook file we provide on your homepage: `Workspace -> Users -> Import`
3. Create a cluster: `Clusters -> Create Cluster`. You can use any cluster name as you like. When configuring your cluster, make sure to choose **runtime version** `11.1`. Note: It may take a while to launch the cluster, please wait for its status to turn to '`active`' before start running.
4. Installing third-party packages that will be used in the homework on Databricks: `Clusters -> Cluster name -> Libraries -> Install New`. Then select `PyPI`, enter the package name as `nose`. Finally click `Install` to install it.
5. You can start to play with the notebook now!

Note: Databricks Community Edition only allows you to launch one 'cluster'. If the current cluster is 'terminated', then you can either (1) delete it, and then create a new one, or (2) activate and attach to the existing cluster when running the notebook. Make sure to install nose.

3.5 Logistic Regression

In this section you will go through the steps for creating a click-through rate (CTR) prediction pipeline. You will work with the a sample data that we sampled from Criteo Labs dataset. You will use a logistic regression model and featurize the data by constructing your own one-hot-encoding (OHE).

This excercise covers:

- **Part 1: Featurize categorical data using one-hot-encoding (OHE)**
- **Part 2: Construct an OHE dictionary**
- **Part 3: Parse CTR data and generate OHE features**
- **Part 4: CTR prediction and logloss evaluation**
- **Part 5: Reduce feature dimension via feature hashing**

See the notebook for detailed descriptions and instructions of each question.

4 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?

   (b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

2. (a) Did you give any help whatsoever to anyone in solving this assignment?

   (b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. (a) Did you find or come across code that implements any part of this assignment?

   (b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).