Homework 3

TAs: Gunjan, Kathryn, Sean

10-405/605: Machine Learning with Large Datasets

Due Friday, February 21st at 11:59 PM Eastern Time

Submit your solutions via Gradescope.

IMPORTANT: Be sure to highlight where your solutions are for each question when submitting to Gradescope otherwise you will be marked 0 and will need to submit regrade request for every solution un-highlighted in order for fix it!

Note that Homework 3 consists of two parts: this written assignment and a programming assignment. Remember to fill out the collaboration section found at the end of this homework as per the course policy.

All students are required to complete the following:

1. Written Section 1.1 *[25 points]*
2. Programming Section 2.1 *[30 points]*
3. Programming Section 2.2 *[45 points]*

All students are required to complete **all** sections of the homework. Your homework score will be calculated as a percentage over the maximum points you can earn, which is 100.

Programming: The programming in this homework is **NOT** autograded however you are required to upload your completed notebooks to Gradescope, otherwise you will not receive credit for the programming sections.

1 Written Section [25 Points]

1.1 An Unbiased Sketch (25 pts)

The count-min sketch presented in lecture is biased, as the estimated count $\hat{c}_s$ for item $s \in \{1, \ldots, k\}$ 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) *[10 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) *[7 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. You'll need the variance of the sketch which is given by

$$Var(\hat{c}_s) = \frac{1}{m} \sum_{j \in \{1,\ldots,k\}, j \neq s} c_j^2$$

where m is the number of buckets for the hash function h.

(Hint: Consider Chebyshev's inequality: $P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$.)

c) 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).

2 Programming [75 Points]

Logistics and AWS

For this homework you will be diving into using AWS's Elastic MapReduce (EMR) systems. As such we strongly recommend that you start this homework early.

If you are enrolled in the course and applied for them via the Google form, you should have $150 AWS credits applied to your account. While we expect that all AWS based homework will cost you $60 in total, this extra amount is there in case it takes you more. If you use up all of your $150 your credit card which you opened your AWS account with will be charged! Please be sure to set up usage warnings to make sure that you are not overspending. Here is a list of everything you will create which is chargeable in AWS:

1. EC2 Instances
2. EBS Volumes
3. EMR Instances
4. S3 Bucket Usage - Very small cost

When you are not working on the homework we strongly recommend you terminate/delete instances/volumes as the cost for leaving these open, even idle, can build up fast. The only thing we suggest keeping open is the S3 bucket until you are completely finished with HW3, since this costs very little. You should plan to run cheap instances while developing your code on a subset of data and only run it on something larger (and more expensive) once you have everything ready.

HW3 AWS Requirements and Getting lab files

For this part of the programming, you will need to use a S3 bucket and EMR Instances. We recommend writing your code using one m4.xlarge 'worker' instance using a subset of the data and using two m4.xlarge 'worker' instances when you have finished your notebook and need to run on the full dataset. **Make sure that your AWS is current set to N.Virginia US-EAST-1.**

You can obtain the notebook `'HW3-LSH.ipynb'` in the homework 3 handout .zip file. You should upload this to your Jupyterlab hosted on your AWS EMR cluster. **Be sure to download your completed notebook before you terminate your AWS cluster, otherwise you will lose all completed data.**

You will need to set up an EMR cluster by following the instructions in the AWS Guide

Preparing for submission

We provide several public tests via `assert` in the notebook, which will help you know if you are completing the notebook correctly. You may want to pass all those tests before submitting your homework. There is no Autograder for this homework, however in order to be given any credit for your programming section, you **MUST** upload your completed notebook to the HW3 Programming submission slot on Gradescope.

2.1 Locality Sensitive Hashing [30 Points]

In this coding section, we will explore a variant of approximate nearest neighbors (ANN) search for images based on locality-sensitive hashing (LSH). As discussed in class, LSH consists of a hashing function which encourages collisions between nearby points in the original space. Choosing the right hashing function is somewhat of an art and depends largely on the domain. In this assignment, we will develop the *cross-polytope hash* for image data. Roughly, the hash first projects an input sample to a lower dimensional space (by using a JL style projection) and then selects the coordinate with the highest magnitude in the resulting projected vector. The **index** of this coordinate, along with its sign, serves as a hash for the given input. One could then perform multiple such projections and stack all the individual hash values to form a **fingerprint**. The number of projections and their dimensionality is a hyperparameter that we need to tune.

Include the query image in the neighbor set for questions (a) and (b) below. Include the query image in the neighbor set for questions (a)-(d) below.

(a) *[10 points]* Visualize up to 20 LSH neighbors for the test image 'stegosaurus/image_0043.jpg'.

(b) *[10 points]* Visualize up to 20 kNN neighbors for the test image 'stegosaurus/image_0043.jpg' using VGG embeddings.

(c) *[5 points]* What can you say about the neighbors returned by LSH versus those returned by kNN+VGG? Which one returns more "semantically" similar neighbors (i.e., the content of the images is the same), and which one returns "pixel"-similar neighbors (i.e., corresponding pixel values are similar)?

(d) *[5 points]* Assume we are performing millions of queries. What can you say about the runtime of each method (LSH vs kNN)? Include any preprocessing steps in your calculation.

2.2 Machine Learning at Scale [45 Points]

Introducton

For the second programming part of the homework, you will be working with the data collected from the Million Song Dataset from the UCI Machine Learning Repository. We have provided the processed dataset in CSV format in an S3 Bucket for you to use in this assignment. You will perform exploratory data analysis (EDA) and data cleaning, and then train models with the original features. You will then perform feature engineering similar to what we did in Homework 1 (TF-IDF and Bag-of-Words), and then train models with these new features.

Getting Started

Open JuptyerHub on your EMR cluster, similar to what you did in part 1, and upload your ‘`HW3-MSD.ipynb`’ notebook to begin working on it!

The S3 Bucket containing the dataset files is: `10405-spring2025-hw3-msp` .

2.2.1 Exploratory Data Analysis *[6 points]*

(a) *[1 points]* Explain why the two features seem problematic (after performing .summary() operation).

(b) *[1 points]* Histograms (remember to label them)

(c) *[1 points]* Explain what is strange about `year`'s distribution and what might cause this. Describe how you could filter `year` to make its histogram look more balanced.

(d) *[1 points]* New histogram for `year`.

(e) *[1 points]* Provide plots for the three pairs. Describe your findings.

(f) *[1 points]* Think about what simple technique you could use to visualize large datasets while retaining a similar data distribution. Briefly describe what you did.

2.2.2 Data Cleaning *[5 points]*

(a) *[1 points]* Your justification for dropping the two features.

(b) *[1 points]* Compare the two numbers and explain the advantages and potential problem of doing this step. What other techniques could you use to potentially do better?

(c) *[1 points]* State the two features.

(d) *[1 points]* Explain your proposed solution and discuss its pros and cons.

(e) *[1 points]* Report the percentage:

2.2.3 Baseline *[9 points]*

(a) *[1 points]* Explain why treating this as a classification problem might be a sensible choice.

(b) *[1 points]* Report what percentage of songs are assigned the "popular" label.

(c) *[1 points]* Explain why we shift the year.

(d) *[1 points]* Explain what scaling means and why we want to perform scaling before the learning step.

(e) *[2 points]* Explain the difference between these two metrics and when AUC might be more useful than accuracy.

(f) *[3 points]* Calculate the train and test AUC of both models and report them.

Models	Train AUC	Test AUC
Logistic Regression		
Random Forest		

2.2.4 Featurization: Bag-of-Words and TF-IDF *[7 points]*

(a) *[2 points]* Explain what the `vocabSize` hyperparameter means in the context of Bag-of-Words.

(b) *[2 points]* Other than featurizing texts, what other feature engineering would you do on the dataset? Briefly describe one.

(c) *[3 points]* Explain where this number "31" comes from.

2.2.5 Modeling with New Features *[9 points]*

(a) *[4 points]* Evaluate train and test AUC for each model and report them.

Models	Train AUC	Test AUC
Logistic Regression		
Random Forest		

(b) *[5 points]* Include the plot and your explanations.

2.2.6 Do Your Best *[6 points]*

(a) *[2 points]* Your final AUC:.

(b) *[2 points]* Your model and hyperparameters.

(c) *[2 points]* Describe your approach.

2.2.7 Reflection *[3 points]*

What challenges did you face in HW3 Programming 2.2? How did you overcome these challenges? What did you learn from HW3 Programming 2.2?

3 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.).