NYU CS-GY 6763: Algorithmic Machine Learning and Data Science Fall 2023

Homework 3

Name: Solution key

Problem 1

1. **Expectation Calculation.** As in class, we have that $\mathbb{E}[\|\Pi x\|_2^2] = \mathbb{E}[\langle \pi, x\rangle^2]$, where π is a single unscaled row from the matrix Π. I.e. π has length n and contains random ± 1 entries. We have:

$$\mathbb{E}[\langle \pi, x\rangle^2] = \mathbb{E}\left[\left(\sum_{j=1}^n \pi_j x_j\right)^2\right] = \mathbb{E}\left[\sum_{j=1}^n \pi_j^2 x_j^2\right] + \mathbb{E}\left[\sum_{i\neq j}^n \pi_i\pi_j x_j x_i\right]$$
$$= \sum_{j=1}^n \mathbb{E}\left[\pi_j^2\right] x_j^2 + \sum_{i\neq j}^n \mathbb{E}\left[\pi_i\pi_j\right] x_j x_i.$$

The last equality follows from linearity of expectation. Since π_i is independent of π_j, we have that for $j \neq i$, $\mathbb{E}[\pi_i\pi_j] = \mathbb{E}[\pi_i]\mathbb{E}[\pi_j] = 0$. On the other hand $\pi_j^2 = 1$ deterministically, so we have $\mathbb{E}\left[\pi_j^2\right] = 1$. Plugging in above, we find that

$$\mathbb{E}[\langle \pi, x\rangle^2] = \sum_{j=1}^n x_j^2 + \sum_{i\neq j}^n 0 \cdot x_j x_i = \sum_{j=1}^n x_j^2 = \|x\|_2^2,$$

as desired.

Variance Calculation. Since $\|\Pi x\|_2^2 = \frac{1}{k}\sum_{i=1}^k \langle \pi^i, x\rangle^2$, where $\pi^1, \ldots, \pi^k$ are the unscaled rows of Π, we first observe that $\mathrm{Var}[\|\Pi x\|_2^2] = \frac{1}{k}\mathrm{Var}[\langle \pi, x\rangle^2]$ for a single random ± 1 vector π. So we just need to bound $\mathrm{Var}[\langle \pi_i, x\rangle^2]$. This gets a bit tricky! There are many ways to do it, but I think the easiest way is to take advantage of linearity of variance by writing:

$$\langle \pi, x\rangle^2 = \sum_{j=1}^n \pi_j^2 x_j^2 + 2\sum_{i>j} \pi_i\pi_j x_i x_j.$$

The terms in the first part of the sum are actually deterministic, since $\pi_j = 1$. The terms in the second part of the sum are random, but they are *pairwise independent* since $\pi_i\pi_j$ is random ± 1 and independent from any $\pi_i\pi_k$, $\pi_k\pi_j$, or $\pi_k\pi_\ell$. They are not mutually independent, but we only need pairwise independence to apply linearity of variance. Note that to make this claim it's important that I used the form $2\sum_{i>j}$ instead of $\sum_{i\neq j}$. If I did the later, there would be repeated random variables in the sum ($\pi_i\pi_j x_i x_j$ and $\pi_j\pi_i x_j x_i$). Writing the other way removes duplicates.

$$\mathrm{Var}[\langle \pi, x\rangle^2] = \sum_{j=1}^n \mathrm{Var}[\pi_j^2 x_j^2] + 4\sum_{i>j}\mathrm{Var}[\pi_i\pi_j x_j x_i] = 0 + 4\sum_{i>j} x_j^2 x_i^2.$$

Then finally we observe that:

$$\|x\|_2^4 = \|x\|_2^2 \cdot \|x\|_2^2 = (x_1^2 + \ldots + x_n^2) \cdot (x_1^2 + \ldots + x_n^2) \geq 2 \sum_{i>j} x_j^2 x_i^2.$$

Putting this together we have that $\mathrm{Var}[\langle \pi, x\rangle^2] \leq 2\|x\|_2^4$ and the result follows since $\mathrm{Var}[\|\Pi x\|_2^2] = \frac{1}{k}\mathrm{Var}[\langle \pi, x\rangle^2]$ as claimed above.

2. This just follows directly from Chebyshev's.

3. It's almost the same analysis as in part 1. The first thing to observe is that:

$$\langle \Pi x, \Pi y\rangle = \frac{1}{k}\sum_{i=1}^{k} \langle \pi^i, x\rangle \langle \pi^i, y\rangle.$$

So we have that $\mathbb{E}[\langle \Pi x, \Pi y\rangle] = \mathbb{E}[\langle \pi, x\rangle\langle \pi, y\rangle]$ and $\mathrm{Var}[\langle \Pi x, \Pi y\rangle] = \frac{1}{k}\mathrm{Var}[\langle \pi, x\rangle\langle \pi, y\rangle]$, where π is a single random ± 1 vector. We also have that

$$\langle \pi, x\rangle\langle \pi, y\rangle = \left(\sum_{j=1}^{n} \pi_j x_j\right) \cdot \left(\sum_{j=1}^{n} \pi_j y_j\right) = \sum_{i=1}^{n} \pi_i^2 x_i y_i + \sum_{j \neq i} \pi_i \pi_j x_i y_j.$$

From this it's clear that

$$\mathbb{E}[\langle \Pi x, \Pi y\rangle] = \mathbb{E}[\langle \pi, x\rangle\langle \pi, y\rangle] = \sum_{i=1}^{n} x_i y_i = \langle x, y\rangle,$$

as desired.

The variance calculation is also a bit tricky since we need to make sure our sums involve pairwise independent random variables. We have that:

$$\langle \pi, x\rangle\langle \pi, y\rangle = \sum_{i=1}^{n} \pi_i^2 x_i y_i + \sum_{j>i} \pi_i \pi_j (x_i y_j + x_j y_i).$$

Applying linearity of variance, we find that

$$\begin{aligned}
\mathrm{Var}[\langle \pi, x\rangle\langle \pi, y\rangle] = \sum_{j>i} (x_i y_j + x_j y_i)^2 &= \sum_{j>i} x_i^2 y_j^2 + x_j^2 y_i^2 + 2 x_i x_j y_i y_j \\
&\leq 2 \sum_{j>i} x_i^2 y_j^2 + x_j^2 y_i^2 \\
&\leq 2(x_1^2 + \ldots + x_n^2)(y_1^2 + \ldots + y_n^2) \\
&= 2\|x\|_2^2 \|y\|_2^2.
\end{aligned}$$

In second to last inequality we have used that for any a, b, $2ab \leq a^2 + b^2$, which follows from the fact that $(a-b)^2 \geq 0$ for all a, b (this is technically called the AM-GM inequality).

Overall, we get a variance bound of:

$$\mathrm{Var}[\langle \Pi x, \Pi y\rangle] \leq \frac{2}{k}\|x\|_2^2 \|y\|_2^2.$$

Once they get the mean and variance, the bound just follows from applying Chebyshev inequality again.

Problem 2

1. Construct 2 length U binary vectors x and y where $x_i = 1$ if $i \in X$ and 0 otherwise, and $y_i = 1$ if $i \in Y$ and 0 otherwise. Note that $|X \cap Y|$ is exactly equal to $\langle x, y \rangle$, so we can estimate the quantity using sketches Πx and Πy. If we set $k = O(1/\epsilon^2)$, then with $9/10$ probability we will have:

$$|\langle \Pi x, \Pi y \rangle| \leq \epsilon \|x\|_2 \|y\|_2$$

Note that $\|x\|_2^2 = |X|$ and $\|y\|_2^2 = |Y|$, which yields the bound.

2. The first thing to note is that $\frac{1}{S} - 1$ is exactly a distinct elements estimator for $X \cup Y$ because $\min(C_i^X, C_i^Y) = \min_{v \in X \cup Y} h_i(v)$. Accordingly, as shown in class (slide 30 of lecture 2), if we set $k = O(1/\epsilon^2)$, then with probability $19/20$,

$$\left| \left(\frac{1}{S} - 1 \right) - |X \cup Y| \right| \leq \epsilon |X \cup Y|.$$

The next thing to note is that k'/k is exactly the MinHash estimator for the Jaccard similarity between X and Y, which we denote $J = \frac{|X \cap Y|}{|X \cup Y|}$. As shown on Ed, if we set $k = O(1/\epsilon^2)$, then with probability $19/20$,

$$\left| J - k'/k \right| \leq \epsilon \cdot \sqrt{J}.$$

By a union bound, we have that both approximation inequalities hold with probability $9/10$ and thus:

$$(1-\epsilon)|X \cup Y| \cdot (J - \epsilon\sqrt{J}) \leq \frac{k'}{k}\left(\frac{1}{S} - 1\right) \leq (1+\epsilon)|X \cup Y| \cdot (J + \epsilon\sqrt{J}). \quad (1)$$

Noting that $J \cdot |X \cup Y| =)|X \cap Y|$ we simplify the left hand side of (1) to:

$$\begin{aligned}
(|X \cup Y| - \epsilon|X \cup Y|) \cdot (J - \epsilon\sqrt{J}) &\geq |X \cup Y| - \epsilon|X \cup Y| - \epsilon|X \cup Y|\sqrt{j} \\
&= |X \cap Y| - \epsilon|X \cap Y| - \epsilon\sqrt{|X \cup Y||X \cap Y|} \\
&\geq |X \cap Y| - 2\epsilon\sqrt{|X \cup Y||X \cap Y|}.
\end{aligned}$$

The last step follow from the fact that $|X \cup Y| \geq |X \cap Y|$. Similarly, we can upper bound the right hand side of (1) by:

$$|X \cap Y| + (2+\epsilon)\epsilon\sqrt{|X \cup Y||X \cap Y|}.$$

Adjusting the constant factor on ϵ (setting $\epsilon \leftarrow \epsilon/3$), we conclude that with $k = O(1/\epsilon^2)$,

$$|X \cap Y| - \epsilon\sqrt{|X \cup Y||X \cap Y|} \leq \frac{k'}{k}\left(\frac{1}{S} - 1\right) \leq |X \cap Y| + \epsilon\sqrt{|X \cup Y||X \cap Y|},$$

which proves the bound.

3. The hashing based bound is *strictly better*. In particular, Let $a = X - |X \cap Y|$, $b = |X \cap Y|$, and $c = Y - |X \cap Y|$. We have that $|X| = (a+b)$, $|Y| = (b+c)$, and $|X \cap Y| = (ab+c)$. So, the JL upper bound is equal to:

$$\epsilon(a+b)(b+c) = \epsilon\left((a+b+c)b + ac\right).$$

On the other hand, the hashing based method achieves an upper bound of just

$$\epsilon(a+b+c)b,$$

which will be a lot smaller for sets with low Jaccard similarity (small intersection compared to union).

Problem 3

1. For any vector x, let z be the point on the hyperplane closest to x. Now:

$$\langle x, a\rangle = \langle x - z, a\rangle + \langle z, a\rangle = \langle x - z, a\rangle + c = \|x - z\|_2 + c \geq c + \epsilon.$$

In the second step we used that $\langle z, a\rangle = c$ since z is on the hyperplane. And in the next step we use that $x - z$ must be perpendicular to the hyperplane (for z to be the closest point). And thus $x - z$ is *parallel* to a. Since a is a unit vector, $\langle x - z, a\rangle = \|x - z\|_2$. The proof for any y on the other size of the hyperplane is the same, but in that case, $y - z$ points directly opposite of a

2. To show that there exists a good separating hyperplane for the dimension reduced data, we exhibit one: consider the hyperplane given by parameters $\Pi a/\|\Pi a\|_2, c/\|\Pi a\|_2$.

We can apply Problem 2 to claim that, if Π reduces to $O(\log(1/\delta)/\epsilon^2)$ dimensions, then with probability $(1-\delta)$ for *any* $x \in X$ or $\forall y \in Y$,

$$\langle \Pi a, \Pi x\rangle \geq \langle a, x\rangle - \epsilon/2 \geq c + \epsilon/2 \qquad \text{and} \qquad \langle \Pi a, \Pi y\rangle \leq \langle a, y\rangle + \epsilon/2 \leq c - \epsilon/2.$$

Above we use the fact that $\|x\|_2\|\mathbf{a}\|_2 = 1$ and $\|y\|_2\|\mathbf{a}\|_2 = 1$ since all x and y are specified to be unit vectors. Equivalently, we have:

$$\langle \Pi a/\|\Pi a\|_2, \Pi x\rangle \geq c/\|\Pi a\|_2 + \epsilon/2\|\Pi a\|_2 \quad \text{and} \quad \langle \Pi a/\|\Pi a\|_2, \Pi y\rangle \geq c/\|\Pi a\|_2 + \epsilon/2\|\Pi a\|_2. \tag{2}$$

We also have from the distributional JL lemma that, with probability $1-\delta$, $\|\Pi a\|_2 \leq 2$. And if we set $\delta = 1/99(n+1)$, by a union bound we have that (2) holds for all n points in our data set and $\|\Pi a\|_2 \leq 2$ simultaneously with probability $99/100$. This proves the claim with margin $\epsilon/4$.

Problem 4

This problem can be solved in a similar way to the Shazam example from the Lecture 4 notes. You need to optimize over values of s and t, where s is the number of independent locality-sensitive hash functions used in your scheme and t is the number of tables used.

Following the analysis in Lecture 5, given a query vector $\mathbf{y}$ and some database vector $\mathbf{x}$, the probability of $\mathbf{x}$ showing up as a candidate near-duplicate (which will need to be scanned when $\mathbf{y}$ is issued as a query) is equal to:

$$1 - (1 - (1 - \theta(\mathbf{x}, \mathbf{y})/\pi)^s)^t \tag{3}$$

where θ is the angle between vectors $\mathbf{x}$ and $\mathbf{y}$.

Our goal is to find the s, t pair with the smallest value of t which satisfies:

```
% brute force search over t values to find smallest t that works
tvals = 1:100;
% boudaries and heights of the buckets given
cos_sims = [-1:.25:1];
freq = [.01,1.99,14,34,34,14,1.90,.01];
% convert boundaries from cosine similarities to angles
thetas = acos(cos_sims);
% will only use top edge of each bucket for a worst case analysis
thetas = thetas(2:end);
% we want to find any match with cosine similarity >= .98 with prob. ,99
cutoff = acos(.98);
cprob = .99;

% keep track of how many candidate matches there are for a given t
ncandidates = zeros(1,length(tvals));
for t = tvals
    % smallest value of s which ensures we find near-matches
    sopt = floor(log(1 - (1-cprob)^(1/t))/log(1-cutoff/pi));
    % expected number of hits
    hitProbs = 1 - (1 - (1-thetas./pi).^sopt).^t;
    ncandidates(t) = (hitProbs*freq'/100)*1000000000;
end
% answer to part (a)
min(find(ncandidates < 1e6))
% answer to part (b)
min(find(ncandidates < 2e5))
```

1. If $\cos(\theta(\mathbf{x}, \mathbf{y})) \geq .98$, $1 - (1 - (1 - \theta(\mathbf{x}, \mathbf{y})/\pi)^s)^t \geq .99$

2. Based on the histogram data provided, the expected number of candidate near-duplicates in less than 1 million, or 200k, for parts (1) and (2).

Observe that $1-(1-(1-\theta(\mathbf{x}, \mathbf{y})/\pi)^s)^t$ is monotonically decreasing with s and the expected number of duplicates monotonically decreases with s. So, for a given value of t, it suffices to find the largest possible s such that $1 - (1 - (1 - \cos^{-1}(.98)/\pi)^s)^t \geq .99$. I did this by solving for:

$$s = \frac{\log(1 - (1 - .99)^{1/t})}{\log(1 - \cos^{-1}(.98)/\pi)}$$

and taking the floor.

Then an upper bound on the number of expected candidates can be computed for this s. This will be the smallest possible number of expected candidates for the given t.

My code is included below. I obtained solutions of:

- **20 tables** for ≤ 1 million candidates
- **44 tables** for ≤ 200k candidates