New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6763: Homework 3.
Due Monday, April 11th, 2022, 11:59pm.

Collaboration is allowed on this problem set, but solutions must be written-up individually. Please list collaborators for each problem separately, or write "No Collaborators" if you worked alone.

Problem 1: Sparse Regression is Only Easier

(10 pts) Often in machine learning it is desirable to choose a "sparse" model that only involves a subset of features. Sparse models can help avoid overfitting and are sometimes easier to interpret. Suppose we run some feature selection algorithm to select $d' < d$ features (i.e., columns) from a data matrix $A \in \mathbb{R}^{n \times d}$ with n examples. Let $A' \in \mathbb{R}^{n \times d'}$ be the data matrix restricted to just those features. Now we want to solve a least squares regression problem $\min_x \|A'x - b\|_2^2$. Show that the condition number of this convex optimization problem is no worse than that of $\min_x \|Ax - b\|_2^2$.

Hint: Use the second order definition of β-smoothness and α-strong-convexity in terms of the largest and smallest eigenvalues of the Hessian Matrix, respectively.

Hint 2: Recall that for a square symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_{\max} = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = \lambda_{\min}$, by the Courant-Fischer Min-Max Theorem[1] we have that

$$\lambda_{\max}(\mathbf{A}) = \max_{x \in \mathbb{R}^n} \frac{x^T A x}{\|x\|_2^2}, \qquad \lambda_{\min}(\mathbf{A}) = \min_{x \in \mathbb{R}^n} \frac{x^T A x}{\|x\|_2^2}$$

Problem 2: Approximating Eigenvalues Moments

(15 pts) Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square symmetric matrix, which means it it guaranteed to have an eigendecomposition with real eigenvalues, $\lambda_1 \geq \ldots \geq \lambda_n$. While computing these eigenvalues naively takes $O(n^3)$ time, it turns out that we can compute their *sum* much more quickly: with n operations. This is because $\sum_{i=1}^n \lambda_i$ is exactly equal to the trace of $\mathbf{A}$, i.e. the sum of its diagonal entries $\text{tr}(\mathbf{A}) = \sum_{i=1}^n \mathbf{A}_{ii}$. We can also compute the sum of squared eigenvalues in $O(n^2)$ time by taking advantage of the fact that $\sum_{i=1}^n \lambda_i^2 = \|\mathbf{A}\|_F^2$ where $\|\mathbf{A}\|_F^2 = \sum_{i,j} \mathbf{A}_{i,j}^2$ is the Frobenius norm. What about $\sum_{i=1}^n \lambda_i^3$? Or $\sum_{i=1}^n \lambda_i^4$? It turns out that no exact algorithms faster than a full eigendecomposition are known.

In this problem, however, we show how to *approximate* $\sum_{i=1}^n \lambda_i^k$ for any positive integer k in $O(n^2 k)$ time. By the way, this is not a contrived problem – it has a ton of applications in machine learning and data science.

(a) Show that $\sum_{i=1}^n \lambda_i^k = \text{tr}(\mathbf{A}^k)$ where $\mathbf{A}^k$ denotes the chain of matrix products $\mathbf{A} \cdot \mathbf{A} \cdot \ldots \cdot \mathbf{A}$, repeated k times. For the remainder of the problem we use the notation $\mathbf{B} = \mathbf{A}^k$.

(b) Let $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^n$ be m independent random vectors, all with i.i.d. $\{+1, -1\}$ uniform random entries. Let $Z = \frac{1}{m} \sum_{i=1}^m (\mathbf{x}^{(i)})^T \mathbf{B} \mathbf{x}^{(i)}$. We will show that Z is a good estimator for $\text{tr}(\mathbf{B})$ and thus for $\sum_{i=1}^n \lambda_i^k$. Give a short argument that Z can be computed in $O(n^2 k m)$ time (recall that $\mathbf{B} = \mathbf{A}^k$).

(c) Prove that:

$$\mathbb{E}[Z] = \text{tr}(\mathbf{B}) \qquad \text{and} \qquad \text{Var}[Z] \leq \frac{4}{m} \|\mathbf{B}\|_F^2$$

Hint: Use linearity of variance but be careful about what things are independent!

(d) Show that if $m = O(\frac{1}{\epsilon^2})$ then, with probability 9/10,

$$|\,\text{tr}(\mathbf{B}) - Z| \leq \epsilon \|\mathbf{B}\|_F.$$

[1] See https://en.wikipedia.org/wiki/Min-max_theorem

(e) Argue that, when $\mathbf{A}$ is positive semidefinite, $\epsilon\|\mathbf{B}\|_F \leq \epsilon \operatorname{tr}(\mathbf{B})$, so the above guarantee actually gives the relative error bound,

$$(1-\epsilon)\operatorname{tr}(\mathbf{B}) \leq Z \leq (1+\epsilon)\operatorname{tr}(\mathbf{B}),$$

all with just $O(n^2k/\epsilon^2)$ computation time.

Problem 3: Count-Sketch is always at least as good as Count-Min

(10 pts) Recall that, for any $\epsilon \in (0,1)$, Count-Sketch can recover an approximation $\tilde{f}$ to the frequency vector $f \in \mathbb{R}^n$ of a stream, such that for all $i \in [n]$ we have

$$|\tilde{f}_i - f_i| \leq \epsilon\|f_{\text{tail}(1/\epsilon^2)}\|_2$$

Where recall $f_{\text{tail}(k)}$ is the result of setting the k largest coordinates (in magnitude) equal to 0. To accomplish this, Count-Sketch uses $O(\frac{1}{\epsilon^2}\log n)$ words of space. Similarly, we saw that Count-Min produces an approximation $\tilde{f}$ such that $|\tilde{f}_i - f_i| \leq \epsilon\|f_{\text{tail}(1/\epsilon)}\|_1$ using $O(\frac{1}{\epsilon}\log n)$ bits of space.

Since for any vector f we have $\|f\|_2 \leq \|f\|_1$, the guarantee of Count-Sketch is superior. However, the space required to obtain this guarantee is larger by a factor of $O(1/\epsilon)$. Therefore, it may seem like Count-Sketch and Count-Min are incomparable. In this problem, you will show that is not the case.

1. Show that for any $2 \leq k \leq n/2$ and any vector $f \in \mathbb{R}^n$, we have

$$\|f_{\text{tail}(k)}\|_2 \leq O\left(\frac{1}{\sqrt{k}}\right)\|f_{\text{tail}(k/2)}\|_1$$

For simplicity, you may assume that k is even.

Hint: You may want to use Hölder's inequality,[2] which states that for any two vectors $x, y \in \mathbb{R}^n$, and any $p, q \in [1,\infty]$ such that $1/p + 1/q = 1$ (where $1/\infty$ is interpreted as 0), we have $\sum_i x_i y_i \leq \|x\|_p\|y\|_q$.

2. Given the above, show how Count-Sketch can be used to obtain a vector $\tilde{f}$ such that $|\tilde{f}_i - f_i| \leq \epsilon\|f_{\text{tail}(1/\epsilon)}\|_1$ using $O(\frac{1}{\epsilon}\log n)$ bits of space. Conclude that Count-Sketch is always at least as good or better than Count-min, when using the same amount of space (up to a constant).

Problem 4: Locating Points via the SVD

(15 pts) Suppose you are given all pairs distances between a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$. You can assume that $d \ll n$. Formally, you are given an $n \times n$ matrix $\mathbf{D}$ with $\mathbf{D}_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$. You would like to recover the location of the original points, at least up to possible rotation and translation (which do not change pairwise distances).

Since we can only recover up to a translation, it may be easiest to assume that the points are centered around the origin. I.e. that $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$.

1. Under this assumption, describe a polynomial time algorithm for learning $\sum_{i=1}^n \|\mathbf{x}_i\|_2^2$ from $\mathbf{D}$. Hint: expand $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ as $(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)$ and go from there.

2. Next, describe a polynomial time algorithm for learning $\|\mathbf{x}_i\|_2^2$ for each $i \in 1, \ldots, n$.

3. Finally, describe an algorithm for recovering a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ which realize the distances in $\mathbf{D}$. Hint: This is where you will use the SVD! It might help to know (and prove to yourself) that $\mathbf{D}$ has rank $\leq d + 2$.

4. Implement your algorithm and run it on the U.S. cities dataset provided in `UScities.txt`. Note that the distances in the file are unsquared Euclidean distances, so you need to square them to obtain $\mathbf{D}$. Plot your estimated city locations on a 2D plot and label the cities to make it clear how the plot is oriented. Submit these images and your code with the problem set (in the same file, as plaintext) – I don't need to be able to run the code.

[2] https://en.wikipedia.org/wiki/Min-max_theorem