New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6763: Homework 4.
Due Thursday, December 15th, 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: Accelerated Gradient Descent Through the Polynomial Lens

(15 pts) In Lecture 7, we saw how to analyze gradient descent for $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$, which has gradient $\nabla f(\mathbf{x}) = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{A}^T\mathbf{b}$. The dominant cost for each gradient descent iteration is multiplying $\mathbf{x}$ by $\mathbf{A}^T\mathbf{A}$ to compute the gradient, which takes $O(nd)$ time when A is $n \times d$.

We obtained a convergence bound depending on the largest and smallest eigenvalues of $\mathbf{A}^T\mathbf{A}$, which we denote λ_1 and λ_d respectively. We did so by rearranging the gradient descent update rule:

$$\begin{aligned}
\mathbf{x}^{(i)} &= \mathbf{x}^{(i-1)} - \eta\left(2\mathbf{A}^T\mathbf{A}\mathbf{x}^{(i-1)} - 2\mathbf{A}^T\mathbf{b}\right) \\
\mathbf{x}^{(i)} - \mathbf{x}^* &= \mathbf{x}^{(i-1)} - \eta\left(2\mathbf{A}^T\mathbf{A}\mathbf{x}^{(i-1)} - 2\mathbf{A}^T\mathbf{A}\mathbf{x}^*\right) - \mathbf{x}^* \qquad \text{since } \nabla f(\mathbf{x}^*) = \mathbf{0}, \text{ so } \mathbf{A}^T\mathbf{A}\mathbf{x}^* = \mathbf{A}^T\mathbf{b} \\
\mathbf{x}^{(i)} - \mathbf{x}^* &= (\mathbf{I} - 2\eta\mathbf{A}^T\mathbf{A})(\mathbf{x}^{(i-1)} - \mathbf{x}^*).
\end{aligned}$$

By induction, it follows that the error $\mathbf{x}^{(i)} - \mathbf{x}^*$ equals $\mathbf{x}^{(i)} - \mathbf{x}^* = (\mathbf{I} - 2\eta\mathbf{A}^T\mathbf{A})^i(\mathbf{x}^{(0)} - \mathbf{x}^*)$. This allowed us to obtain a convergence bound by arguing that, if we set $\eta = 1/2\lambda_1$ where λ_1 is the largest eigenvalue of $\mathbf{A}^T\mathbf{A}$, then $(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}^T\mathbf{A})^i$ has top eigenvalue $< \epsilon$ after $i = O(\frac{\lambda_1}{\lambda_d}\log(1/\epsilon))$ iterations. In this problem you will prove an "accelerated" version of this bound with a significantly improve condition number dependence of $O(\sqrt{\frac{\lambda_1}{\lambda_d}}\log(1/\epsilon))$ iterations.

1. Let p be a degree q polynomial. I.e. $p = c_0 + c_1 x + \ldots + c_q x^q$. Show that, for any p with $c_0 + c_1 + \ldots + c_q = 1$ and any starting vector $\mathbf{x}^{(0)}$, we can compute in q iterations (i.e., using q gradient computations and up to $O(ndq)$ additional runtime) a vector $\mathbf{x}^{(q)}$ such that:

$$\mathbf{x}^{(q)} - \mathbf{x}^* = p\left(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}^T\mathbf{A}\right)(\mathbf{x}^{(0)} - \mathbf{x}^*).$$

2. Prove that for $q = O(\sqrt{\frac{\lambda_1}{\lambda_d}}\log(1/\epsilon))$, there exists a polynomial p with coefficients $c_0 + c_1 + \ldots + c_q = 1$ such that the top eigenvalue of $p\left(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}^T\mathbf{A}\right) \leq \epsilon$. **Hint:** You might want to use Claim 4 in the supplemental notes on the Lanczos method posted for Lecture 10.

By Part 2, above, it follows that $\|\mathbf{x}^{(q)} - \mathbf{x}^*\|_2 = \|p\left(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}^T\mathbf{A}\right)(\mathbf{x}^{(0)} - \mathbf{x}^*)\|_2 \leq \epsilon\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2$ as long as we use degree $q = O(\sqrt{\frac{\lambda_1}{\lambda_d}}\log(1/\epsilon))$ – i.e. run for $O(\sqrt{\frac{\lambda_1}{\lambda_d}}\log(1/\epsilon))$ iterations.

Problem 2: Matrix Concentration from Scalar Concentration

(15 pts) This problem asks you to prove a simplified (and slightly weaker) version of the matrix concentration result used in Lecture 10. Construct a random *symmetric* matrix $R \in \mathbb{R}^{n \times n}$ by setting $R_{ij} = R_{ji}$ to $+1$ or -1, uniformly at random. Prove that, with high probability,

$$\|R\|_2 \leq c\sqrt{n \log n},$$

for some constant c. This is much better than the naive bound of $\|R\|_2 \leq \|R\|_F = n$ and it's nearly tight: we always have that $\|R\|_2^2 \geq \|R\|_F^2/n$ (do you see why?) so $\|R\|_2 \geq \sqrt{n}$ no matter what.

Here are a few hints that might help you along:

- Recall that for a matrix R, $\|R\|_2 = \max_{x \in \mathbb{R}^n} \frac{\|Rx\|_2}{\|x\|_2}$. When R is symmetric, it also holds that $\|R\|_2 = \max_{x \in \mathbb{R}^n} \frac{|x^T R x|}{x^T x}$.
- Try to first bound $\frac{|x^T R x|}{x^T x}$ for one particular x. You might want to use a Hoeffding bound.
- Then try to extend the result to hold for all x simultaneously, using an ϵ-net argument.

Problem 3: Spectral Methods for Cliques

(10 pts) A common tasks in data mining is to identify large *cliques* in a graph. For example, in social networks, large cliques can be indicators of fraudulent accounts or networks of accounts designed to promote certain content. In this problem, we consider a spectral heuristic for finding a large clique based on the top eigenvector of the graph adjacency matrix A:

- Compute the leading eigenvector v_1 of A.
- Let $i_1, \ldots, i_k \in \{1, \ldots, n\}$ be the indices of the k entries in v_1 with largest absolute value.
- Check if nodes $i_1, \ldots, i_k$ form a k-clique.

We will analyze this heuristic on a natural random graph model. Specifically, let G be an Erdos-Renyi random graph: we start with n nodes, and for every pair of nodes (i, j), we add an edge between the pair with probability $p < 1$. To simplify the math, also assume that we add a self-loop at every vertex i with probability p. Then, choose a fixed subset S of k nodes to form a clique. Connect all nodes in S with edges and add self-loops. We will argue that, for sufficiently large k, we can expect the heuristic above to identify the nodes in the clique.

1. Let A be the adjacency matrix of a random graph generated as above. What is $\mathbb{E}[A]$? Prove that the rank of $\mathbb{E}[A]$ is 2. In other words, the matrix only has two non-zero eigenvalues.
2. Derive expressions for the two non-zero eigenvalues of $\mathbb{E}[A]$, and their corresponding eigenvectors. **Hint:** First argue that, up to multiplying by a constant, any eigenvector v must have $v[i] = 1$ for all $i \notin S$ and $v[i] = \alpha$ for all $i \in S$, where α is a constant. Then use some high school algebra 2!
3. Using your results from (2) above, argue that, up to a positive scaling, the top eigenvector v_1 has $v[i] = 1$ for all $i \notin S$ and $v[i] = \alpha$ for all $i \in S$, where $\alpha > 1$. In other words, the largest entries of v_1 exactly correspond to the nodes in the clique!
4. To prove the algorithm works, it is possible to use a matrix concentration inequality to argue that the top eigenvector of A is close to that of $E[A]$. Instead of doing that, let's verify things experimentally. Generate a graph G according to the prescribed model with $n = 900$, $k = |S| = 30$, and $p = .1$. Compute the top eigenvector of A and look at its 30 largest entries in magnitude. What fraction of nodes in the clique S are among these 30 entries? Repeat the experiment and report the average fraction recovered.