New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6763: Homework 3.
Due Monday, November 24th, 2021, 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: Weak Submodular Optimization

(10 pts) Consider the definition of weak submodularity that Teal introduced in class: For a fixed positive integer k, we say that a set function $f : 2^{[n]} \to \mathbb{R}$ is γ_k-weekly submodular for k if, for all $S' \subset \{1, \ldots, n\}$ and $S \subset \{1, \ldots, n\} \setminus S'$ with $|S| \leq k$,

$$\frac{\sum_{e \in S} f(e|S')}{f(S|S')} \geq \gamma_k.$$

A submodular function satisfies this inequality with $\gamma_k = 1$ for all k, but in general we could have $\gamma_k < 1$. Even if this is the case, it turns out that we can still prove that the greedy algorithm is effective at maximizing such weakly-submodular functions.

In particular, suppose we select items $s_1, \ldots, s_k$ and let $S_i = \{s_1, \ldots, s_i\}$ with $S_0 = \{\}$ equal to the empty set. The greedy strategy chooses items via the following rule

$$s_i = \underset{e \in \{1,\ldots,n\} \setminus S_{i-1}}{\arg\max} f(e \mid S_i).$$

Prove that, if $S^* = \arg\max_{S:|S|=k} f(x)$, then

$$f(S_k) \geq (1 - e^{-\gamma_k}) f(s^*).$$

Problem 2: 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 (for example, a submodular maximization based method) 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 $\min_x \|Ax - b\|_2^2$.

Problem 3: 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 $\|\mathbf{A}\|_F^2$ 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 that you can ask me about in office hours!

(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 $O(n^2 km)$ 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,

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

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

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

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

Problem 4: Non-convex Optimization

(10 pts) Consider the problem of computing the top right singular vector $\mathbf{v}_1$ of a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$. As mentioned, it is possible to frame this problem as an optimization problem and solve it with gradient descent. As discussed in class, a benefit of doing so is that it makes it easier to introduce stochastic and online methods, and possible use projection to add additional constraints.

Recall that $\mathbf{A}$'s right singular vectors are equal to the eigenvectors of $\mathbf{A}^T\mathbf{A}$ and the eigenvalues of $\mathbf{A}^T\mathbf{A}$ are equal to $\lambda_1 = \sigma_1^2, \ldots, \lambda_d = \sigma_d^2$, where $\sigma_1 > \ldots \sigma_d > 0$. You can assume there are no repeat singular values for this problem, so these are strict inequalities.

1. **Quick answer:** Assume we have know some coarse upper bound $\tilde{\lambda} \geq \lambda_1$. Let $f(\mathbf{x}) = \tilde{\lambda}\cdot\mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$. It is easy to check that $\mathbf{v}_1 = \arg\min_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x})$ where $\mathcal{S} = \{\mathbf{y} : \|\mathbf{y}\|_2^2 \geq 1\}$. Prove that $f(\mathbf{x})$ is a convex function, but $\mathcal{S}$ is not a convex set.

   Since $\mathcal{S}$ is not convex, our generic analysis of projected gradient descent will give no guarantees for this problem. However, in class we will prove that the method actually does work for this problem – with a correctly chosen step size, it is exactly *exactly equivalent* to power method which we analyze.

2. Consider an alternative approach through unconstrained optimization. Let $g(\mathbf{x}) = -\frac{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$. Now we have that $\mathbf{v}_1 \in \arg\min g(\mathbf{x})$. Prove that g is non-convex and derive an expression for its gradient $\nabla g(\mathbf{x})$. Show that $c\cdot\mathbf{v}_i$ is a statitionary point of g for any right singular vector $\mathbf{v}_i$ and scaling c.

3. g's non-convexity also rules out a direct convergence bound: in theory gradient descent could converge to any singluar vector of $\mathbf{A}$, not the top one. However, we can argue this is unlikely to happen. In particular, we claim that for any $i \neq 1$, $\mathbf{v}_i$ is actually just a *saddle point* of g, not a local minimum. To prove this, show that for any such $\mathbf{v}_i$, and *any* $t > 0$,

   $$\text{There exists a perturbation } \mathbf{z} \text{ with } \|\mathbf{z}\|_2 \leq t \text{ such that} \qquad g(\mathbf{v}_i + \mathbf{z}) < g(\mathbf{v}_i).$$

   If you are interested, you can find a some work on proving gradient methods won't get stuck at saddle points here https://arxiv.org/abs/1703.00887.

Problem 5: 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.