PRINCETON UNIVERSITY FALL '16 COS 521:ADVANCED ALGORITHMS

Homework 4

Out: *Dec 7* Due: *Dec 16*

1. Consider a set of n objects (images, songs etc.) and suppose somebody has designed a *distance* function $d(\cdot)$ among them where $d(i,j)$ is the distance between objects i and j. We are trying to find a geometric realization of these distances. Of course, exact realization may be impossible and we are willing to tolerate a factor 2 approximation. We want n vectors $u_1, u_2, \ldots, u_n$ such that $d(i,j) \leq |u_i - u_j|_2 \leq 2d(i,j)$ for all pairs i, j. Describe a polynomial-time algorithm that determines whether such u_i's exist.

2. The course webpage links to a grayscale photo. Interpret it as an $n \times m$ matrix and run SVD on it. What is the value of k such that a rank k approximation gives a reasonable approximation (visually) to the image? What value of k gives an approximation that looks high quality to your eyes? Attach both pictures and your code. (In matlab you need mat2gray function.) Extra credit: Try to explain from first principles why SVD works for image compression at all.

3. Suppose we have a set of n images and for some multiset E of image pairs we have been told whether they are *similar* (denoted +edges in E) or *dissimilar* (denoted −edges). These ratings were generated by different users and may not be mutually consistent (in fact the same pair may be rated as + as well as −). We wish to *partition* them into clusters $S_1, S_2, S_3, \ldots$ so as to maximise:

   (# of +edges that lie within clusters) + (# of −edges that lie between clusters).

   Show that the following SDP is an upperbound on this, where $w^+(ij)$ and $w^-(ij)$ are the number of times pair i, j has been rated + and − respectively.

$$\begin{aligned} \max \sum_{(i,j)\in E} & w^+(ij)(x_i \cdot x_j) + w^-(ij)(1 - x_i \cdot x_j) \\ & |x_i|_2^2 = 1 \qquad \forall i \\ & x_i \cdot x_j \geq 0 \qquad \forall i \neq j. \end{aligned}$$

4. For the problem in the previous question describe a clustering into 4 clusters that achieves an objective value 0.75 times the SDP value. (Hint: Use Goemans-Williamson style rounding but with two random hyperplanes instead of one. You may need a quick matlab calculation just like GW.)

5. Suppose you are given m halfspaces in $\Re^n$ with rational coefficients. Describe a polynomial-time algorithm to find the largest *sphere* that is contained inside the polyhedron defined by these halfspaces.

6. Let f be an n-variate convex function such that for every x, every eigenvalue of $\nabla^2 f(x)$ lies in $[m, M]$. Show that the optimum value of f is lowerbounded by $f(x) - \frac{1}{2m}|\nabla f(x)|_2^2$ and upperbounded by $f(x) - \frac{1}{2M}|\nabla f(x)|_2^2$, where x is any point. In other words, if the gradient at x is small, then the value of f at x is near-optimal. (Hint: By the mean value theorem, $f(y) = f(x) + \nabla f(x)^T(y-x) + \frac{1}{2}(y-x)^T\nabla^2 f(z)(y-x)$, where z is some point on the line segment joining x, y.)

7. (Yet another convex minimization based algorithm for submodular minimization)

   In Lecture 16, we saw that one can minimize an arbitrary submodular function by minimizing the convex Lovász extension of the function over the solid hypercube. This exercise presents another convex minimization based approach for the problem. This approach gives a fast-in-practice algorithm for submodular minimization used in many Machine Learning applications.

   Recall the notation from Lecture 16: Let $f : 2^{[n]} \to \mathbb{R}$ be a submodular function with $f(\emptyset) = 0$ (recall the notation used : f is a *set* function defined on all possible subsets of the ground set $[n] = \{1, 2, \ldots, n\}$ of size n). The base polyhedron of f, $\mathcal{B}_f$ is the convex set in $\mathbb{R}^n$ defined by the inequality constraints:

$$x \in \mathcal{B}_f \Leftrightarrow \{ \sum_{i \in S} x_i \leq f(S) \ \forall S \subseteq [n]; \sum_{i \in n} x_i = f([n])\}. \tag{1}$$

   a) Prove the "dumbbell lemma": Suppose $J \subseteq N$ is such that $f(J) \leq f(K)$ for any $K \supseteq J$ or $K \subseteq J$. Then, $f(J) = \min_{S \subseteq N} f(S)$.

   b) Let x^* be the optimal solution to $\min_{x \in \mathcal{B}_f} \|x\|_2^2$ and set $J = \{i \mid x_i^* < 0\}$. Prove that $f(J) = \min_{S \subseteq [n]} f(S)$. (Hint: Show that J gives a "tight" inequality in $\mathcal{B}_f$, i.e, $\sum_{j \in J} x_j^* = f(J)$ and use the dumbbell lemma.)

   c) Show that there's an efficient separation oracle for $\mathcal{B}_f$ that uses only an evaluation oracle for f. Conclude that one can thus use the ellipsoid algorithm to minimize an arbitrary submodular function.

   In practice, instead of the ellipsoid method, one uses an iterative procedure known as *Wolfe's method* for implementing the algorithm suggested by b) above.

8. (Tensor Power Iteration, Updated)

   In Lecture 12, we saw the power method for computing the eigenvector of a real symmetric matrix $M \in \mathbb{R}^{n \times n}$ corresponding to its largest singular value (assuming that there's some gap between the largest and the second largest singular values).

   Now, consider the 3 tensor $T = \sum_{i \leq k} \lambda_i \cdot a_i \otimes a_i \otimes a_i$ for $a_i \in \mathbb{R}^n$ satisfying $\|a_i\|^2 = 1$ for each i and $\langle a_i, a_j \rangle = 0$ whenever $i \neq j$. In Lecture 19, we saw Jennrich's algorithm that gives us a method to compute the a_is. This problem explores the analog of the power method for computing some single component of the tensor T.

   We first need a notion of "tensor-vector multiplication". For any x, define $T \cdot x$ as the vector $z \in \mathbb{R}^n$ such that $z_i = \sum_{j,k \in n} T_{i,j,k} x_j x_k$. In the *tensor power method*, we start with a random vector $x_0 \in \mathbb{R}^n$ satisfying $\|x_0\|_2 = 1$ and repeatedly compute $x_i = \frac{T \cdot x_{i-1}}{|T \cdot x_{i-1}|_2}$ for $i = 1, 2, \ldots, r$.

(a) Show that for large enough t, x_t converges to some a_i.

(b) Give a tight estimate of the minimum t required to ensure $\|x_t - a_i\|_2 \leq \epsilon$ in terms of λ_is, n, ϵ and the initial choice x_0.

While we understand the power method for matrices very well, theoretical analyses of tensor power iteration is known only in very special cases (even though the method works well in practice in very general situations.)