Princeton University
COS 521: Advanced Algorithms
Final Exam Fall 2013

Sanjeev Arora

Due Jan 14 5pm in Sanjeev's office, Room 307. Pls send us email if you need an extension.

Instructions: The test has 7 questions. Finish the test within **48 hours** after first reading it. You can consult any notes/handouts from this class and feel free to quote, without proof, any results from there. You cannot consult any other source or person in any way.

Do not read the test before you are ready to work on it.

Write and sign the honor code pledge on your exam (The pledge is "I pledge my honor that I have not violated the honor code during this examination.")

Sanjeev and Aman will be available Jan 8–14 by email to answer any questions. We will also offer to call you if your confusion does not clear up. In case of unresolved doubt, try to explain your confusion as part of the answer and maybe you will receive partial credit.

1. (15 points) A set of points S in $\Re^n$ is said to be in *isotropic position* if $E_{x\in S}[(w\cdot x)^2] = 1$ for all unit vectors $w \in \Re^n$ (i.e., the expected squared projection in every direction is 1.) Describe a polynomial time algorithm that given a set of points computes a *linear transformation* whose application puts them in isotropic position.

2. (15 points) Suppose you are given the intercity distances (ℓ_2)of n cities in the US. We wish to compute the coordinates of each city (assuming the center of the earth is the point $(0, 0, 0)$ in 3D space). Describe a polynomial-time algorithm to compute the coordinates given the intercity distances. Is the answer unique?

   (Extra credit) Describe conditions under which it recovers the true coordinates even if there is "noise" in the intercity distances.

3. (Learning disks; 20 points) You have to write an algorithm that learns the concept of "disks"in the plane. There is an unknown disk D in $\Re^2$ and points are picked uniformly at random from it. Describe a polynomial time algorithm that learns the disk from these samples. Here "learning"means outputting a disk D' such that (a) no point outside D is in D' (b) A random point in D is also in D' with probability at least $1 - \delta$.

   Describe how many sampled points does your algorithm need.

   (Hint: There are infinitely many disks in the plane, and you need a clever union bound against all incorrect disks.)

4. (Sparse Recovery; 20 points) Normally one solves linear equations by matrix inversion. In *Sparse recovery* one uses a matrix A whose columns are unit vectors that are all "almost orthogonal."Note that if the matrix has n rows it cannot have more than n mutually orthogonal columns.

   (a) First we show that if we only require the columns to be "almost orthogonal"then it can have many more columns, say $m = n^3$.

   Suppose you pick m random unit vectors $A_1, A_2, \ldots, A_m$ in $\Re^n$ by picking each coordinate according to a univariate gaussian with mean 0 and variance $1/n$, and then scaling these vectors to be unit vectors.

   Show that with high probability they satisfy $|A_i \cdot A_j| \leq O(\frac{\sqrt{\log m}}{\sqrt{n}})$ for all i, j.

   (b) Now let A be the matrix whose ith column is A_i. In sparse recovery some unknown vector $x \in \{-1, 1, 0\}^m$ (say, brain wave pattern) is out in nature, and we are able to observe Ax (where measuring of Ax is done by precisely setting up the measuring equipment according to A).

   We say x is *k-sparse* if it has only k nonzeros. Give a polynomial time algorithm that recovers x from Ax if x is k-sparse if $k = o(\sqrt{n}/\sqrt{\log m})$. Your algorithm is given A and Ax.

   (Aside: Note that there are many solutions x; the system is underdetermined. The main point is that the true x is k-sparse, and we are interested in this sparse solution.)

5. (Upperbounds on MAX-CUT; 20 points) For a graph $G = (V, E)$ let its *Laplacian* be the matrix

$$L_{ij} = \begin{cases} \text{degree}(i) & i = j \\ -1 & \{i, j\} \in E \\ 0 & \text{else.} \end{cases}$$

(a) Show that $\frac{n}{4}\lambda_{max}(L)$ is an upperbound on MAX-CUT of G, where λ_{max} is the largest eigenvalue.

(b) Show that for every $u \in \Re^n$ such that $\sum_i u_i = 0$, the following is also an upper bound on MAX-CUT of G:

$$f(u) = \frac{n}{4}\lambda_{max}(L + \text{diag}(u)).$$

(c) Show that f is convex.

(d) Sketch and analyse an algorithm to compute the minimum value of f over all u such that $\sum_i u_i = 0$. It should compute a value within $(1 + \epsilon)$ of the minimum in time polynomial in n and $1/\epsilon$.

6. (Lowerbounds for Experts; 20 points) Recall the experts setup from lectures $8, 10$ with n experts, T days, and $\{0, 1\}$ losses. You have to prove that to get within an $(1 + \epsilon)$ factor of the loss of the best expert requires at least $T = \Omega(\frac{\log n}{\epsilon^2})$ days, matching the upper bound we gave.

Hint: Take a $n \times T$ random matrix with $0/1$ entries, and designate the ith column of the random matrix as payoffs for the experts on day i.

Useful Fact you can use without proof: For a random $n \times T$ matrix with $0/1$ entries, with high probability there is a row with at most $T/2 - \sqrt{T \log n}/2$ 1's.

7. (Nearest Neighbor; 20 points) In the *nearest neighbor (NN) problem* we are given a database of n vectors in $\Re^d$ which has to be stored in a data structure. (Think of d as large; e.g. number of pixels in an image) The algorithm receives a *query* $q \in \Re^d$ and has to return the point ("nearest neighbor") in the database that is *closest* to q. The problem is difficult, so in practice this objective is relaxed to the c-approximate nearest neighbor so that the algorithm is allowed to return *any* point x in the database whose distance from q is at most cR, where R is the distance from q to the nearest neighbor. In this problem we will assume (using bucketing) that $R \in [1, 1 + \epsilon)$ so it suffices to return a point at distance at most c (if one exists). The distance in question is Euclidean (i.e., ℓ_2). The algorithm has to succeed with probability $\Omega(1)$ for each query, where the probability is over the choice of the randomized data structure.

(a) Suppose b is picked randomly from $[0, W-1]$ and we define $h : [0, 1) \to \{0, 1, \ldots, W - 1\}$ as $h(x) = \lfloor x + b \rfloor$. Show that for each $x, y \in [0, 1)$:

$$\Pr[h(x) = h(y)] = 1 - |x - y|.$$

(b) Show that given any number R constant $c > 0$ there is a W and a map g mapping $\Re^n$ to $\{0, 1, \ldots, W-1\}$ of the form $\lfloor (r \cdot x + b) \bmod W \rfloor$ satisfying

$$|x - y|_2 \leq R \Rightarrow \Pr[g(x) = g(y)] \geq p_1$$

$$|x - y|_2 \geq c \cdot R \Rightarrow \Pr[g(x) = g(y)] \leq p_2,$$

where $p_1 > p_2$ are constants.

(c) Now define a hash function $f : \Re^d \to \{0, 1, \ldots, W-1\}^k$ such that

$$|x - y|_2 \leq R \Rightarrow \Pr[f(x) = f(y)] \geq p_1^k$$

$$|x - y|_2 \geq c \cdot R \Rightarrow \Pr[f(x) = f(y)] \leq p_2^k,$$

(d) Put together these ideas to describe a data structure of size $O(n^{1+\rho})$ which allows c-approximate NN to be answered for any query q in n^ρ time, where $\rho \approx \frac{\log 1/p_1}{\log 1/p_2}$. Find a rough expression for ρ as a function of c.

(e) (Extra credit) Show that ρ can be made as small as $1/c$.