6.838: Geometric Computing

Fall 2003

Problem Set 4
Due on **Tuesday, December 2**

Mandatory Part

Problem 1. Minkowski Sum
Show that a Minkowski Sum of two convex polygons is convex.

Problem 2. Norms
Show that for any $x \in \Re^d$, $\|x\|_2 \leq \|x\|_1 \leq \sqrt{d}\|x\|_2$.

Problem 3. Fast Approximate Nearest Neighbor
Construct a new data structure for the Approximate Near Neighbor problem in $\Re^d$ under l_2 norm. Your data structure should have the following parameters:

- Approximation factor: $O(d^{1.5})$
- Space: $O(dn)$
- Query time: $O(d)$

Your data structure can be randomized.

Problem 4. LSH in $\Re^d$
In the class, we have seen how to embed $\{0 \ldots M\}^d$ equipped with l_1 norm, into the Hamming space $\{0,1\}^{Md}$. This automatically yields a randomized data structure solving a c-approximate Near Neighbor with query time $O(dMn^{1/c})$, for $c = 1 + \epsilon > 1$.

Show how to extend the latter data structure so that it works for points in $\Re^d$ (again, the distance is defined by the l_1 norm). Your data structure should support queries in time $O((d/\epsilon)^{O(1)}n^{1/c})$.

Hint: do NOT try to construct an embedding of $(\Re^d, l_1)$ into $(\{0 \ldots M\}^d, l_1)$. Instead, try to come up with an algorithmic method for replacing real coordinates by small integers.

Optional Theoretical Part

Problem A. $(1,2)-B$ **metrics**

In the class, we have seen how to construct an exact embedding of a given metric $M = (X, D)$, $|X| = n$, into l_∞^n. In this problem we consider embeddings of a special subclass of metrics called $(1,2)-B$ *metrics*. A metric is a $(1,2)-B$ metric if it satisfies the following two very particular conditions:

1. All non-zero distances are either 1 or 2
2. For any point $p \in X$, the number of points $q \in X$ such that $D(p,q) = 1$ is at most B.

Show that there is a constant C such that any metric M satisfying the above conditions can be embedded exactly into l_∞^d where $d = CB \log n$.

Hint: Use probabilistic method, similar to the proof of Matousek's theorem.

Note: You might wonder: why anyone would be interested in $(1,2)-B$ metrics ? It turns out that it is possible to show that, for a certain constant $A > 1$, it is NP-hard to find an A-approximate solution the Traveling Salesman Problem for such metrics (this is a much stronger fact than the NP-hardness of the *exact* TSP showed in the Intro to Algorithms class). This remains true even if B is constant.

The embedding implies that the problem is equally hard even if the metric is induced by n points living in l_∞ with dimension $d = O(\log n)$. So, any A-approximation algorithm for this problem is unlikely to run in time $2^{2^{o(d)}}$. Otherwise, we would have an algorithm solving an NP-hard problem in time $2^{2^{o(d)}} = 2^{2^{o(\log n)}} = 2^{n^{o(1)}}$, i.e., in sub-exponential time, which is conjectured to be impossible.

So, the problem of approximately solving TSP in d-dimensional l_∞ norm suffers from doubly exponential dependence on d. This is a "super-curse of dimensionality" !

Problem B. Exact Nearest Neighbor in l_∞.

Design a data structure for the exact Nearest Neighbor problem for $\Re^d$ equipped with the l_∞ norm. Your data structure should use $n^{O(d)}$ space and $O(d \log n)$ query time.

Hint: What does a unit ball in l_∞ look like ?

Optional Programming Part

Design a Java applet that animates the Locality-Sensitive Hashing algorithm. Since high-dimensional displays are still waiting to be invented, the implementation should focus on point-sets in two dimensions. To make life simpler, the distance between the points should be measured using the l_1 norm. The points can be then embedded into Hamming space, as seen in the lecture.

The applet should enable the following:

- Let the user enter the data points (pretty much as in earlier applets). Each point should have a distinct label, either a number or a letter. It would be nice if the last point entered (a "query") had different color than the others.
- Let the user specifiy the following parameters:

  - k: the numbers of bits sampled
  - l: the number of hash tables

- For each hash table, display the assignment of points to buckets.

You are welcome to add other features, if you would like to. In particular, it would be nice if the user could specify r and c, and the applet would mark all points as "good", "bad" and "in the middle". Another idea is that the applet suggests the values of l and k (via the formulas seen in the lecture). But it should not enforce them on the user.

Note: The embedding from l_1 to Hamming space increases the dimension by a lot. If this is a problem, you can limit the coordinates to, say, the set $\{0 \dots 31\}$. The final dimension of the Hamming space should be displayed somewhere, so that the user knows how to choose k.