An Algorithm for Approximate Closest-point Queries

Kenneth L. Clarkson
AT&T Bell Laboratories
Murray Hill, New Jersey 07974
e-mail: clarkson@research.att.com

Abstract

This paper gives an algorithm for approximately solving the *post office problem*: given n points (called sites) in d dimensions, build a data structure so that, given a query point q, a closest site to q can be found quickly. The algorithm is also given a relative error bound ϵ, and depends on a ratio ρ, which is no more than the ratio of the distance between the farthest pair of sites to the distance between the closest pair of sites. The algorithm builds a data structure of size $O(n\eta)O(1/\epsilon)^{(d-1)/2}$ in time $O(n^2\eta)O(1/\epsilon)^{(d-1)}$. Here $\eta = \log(\rho/\epsilon)$. With this data structure, a site is returned whose distance to a query point q is within $1+\epsilon$ of the distance of the closest site. A query needs $O(\log n)O(1/\epsilon)^{(d-1)/2}$ time, with high probability.

1 Introduction

The post-office problem is the following: given a set S of n points (called sites) in d dimensions, build a data structure so that given a query point q, the closest site to q can be found quickly. Many data structures have been proposed for this problem; their query times generally are faster than the trivial $O(n)$, and often are $O(\log n)$, as $n \to \infty$. The dependence of the query time on the dimension d is generally very steep for the nontrivial algorithms, at least $2^{\Omega(d)}$. Even heuristic algorithms that are fast in practice, such as bucketing and kd-trees, also have this exponential dependence. This is unfortunate, since many of the most interesting potential applications have d at least 10, and often much greater. Thus the promise of a query time of $O(\log n)$ is crushed by the "curse of dimensionality."

This paper gives a modest but significant improvement in the dimension dependence of one recent algorithm, that of Arya and Mount.[AM93] They attack the problem of *approximate* solution to the post-office problem; their procedure returns an *ϵ-closest* site, one whose distance to the query point q is within $1+\epsilon$ of closest. Here $\epsilon > 0$ is input when their data structure is built. The algorithm given here takes their approach to what is arguably its logical conclusion, reduces the query time from $\tilde{O}(\log^3 n)$ to $\tilde{O}(\log n)$,[1] and reduces the "constant" factors for the dimension from $O(1/\epsilon)^d$ to about $O(1/\epsilon)^{(d-1)/2}$. (The latter reduction applies to the storage and query-time bounds.) The new algorithm has an additional factor in storage and preprocessing time of $\log\rho$, where ρ is bounded above by the ratio of the distance between the sites farthest apart to the distance of the sites closest together. (In fact ρ is roughly the maximum, over all sites s, of the ratio of the distance to s to its farthest Delaunay neighbor, to the distance of s to its closest Delaunay neighbor. Hence $\log\rho$ is negligible relative to other factors.)

Arya *et al.* have more recently described a different approximation algorithm, based on quadtree techniques, that has better storage and preprocessing bounds than those given here, but with a query time that has a $\Omega(1/\epsilon)^d$ dependence.[AMN+94]

The new algorithm uses a technique previously used for polytope approximation [Cla93] in order to approximate a certain Voronoi region for each site by a simpler Voronoi region for the site. The approximation problem is solved using randomization. The algorithm also builds a data structure similar to a skip list, and uses randomization for that.[Pug90]

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association of Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

10th Computational Geometry 94-6/94 Stony Brook, NY, USA
© 1994 ACM 0-89791-648-4/94/0006..$3.50

[1]Here $\tilde{O}(g(n))$ means $O(g(n))$ with probability at least $1 - 1/n^2$.

The following section first describes the general approach, and then develops that approach in the following subsections.

2 The algorithm

Following Arya and Mount, the general idea is to find, for each site s, a list of sites N_s with the following property: if s is not the closest site to the query point q, then there is a site in N_s closer to q than s. With this property, a simple search procedure will lead to the closest site: pick any site s; if a site $t \in N_s$ is closer to q, assign t to s and repeat; otherwise return s as closest.

This approach is not so interesting just yet. The list N_s must be the set of Delaunay neighbors of s, as the interested reader can easily show. This makes for a space requirement of $\Omega(n^2)$ in the worst case, for $d > 2$. Also, the query time is $\Omega(n)$ in the worst case: there is no speedup over the obvious algorithm. (For uniformly distributed points, the query time is more like $O(n^{1/d})$, so this approach is not entirely useless, however.)

For more interesting results, make the problem easier: instead of the closest site, find an ϵ-closest site. (Again, such a site has a distance to the query point that is within $1+\epsilon$ of the distance of the closest site's, for $\epsilon > 0$.) This is the approximate query problem solved by Arya and Mount. They used a collection of narrow cones to obtain their lists, in a way similar to Yao's use of them for finding minimum spanning trees[Yao82]. Here the approach is to go from the desired conditions on the lists to a problem similar to polytope approximation.

The modified construction begins as follows. For each site s, consider a list L_s with the following property: for any q, if there is $b \in S$ with

$$d(q,s) > (1+\epsilon)d(q,b),$$

then there is $b' \in L_s$ with

$$d(q,s) > (1+\epsilon')d(q,b'),$$

where $\epsilon' \equiv \epsilon/2$. (Note that if $\epsilon = 0$, then $L_s = N_s$.) Using such lists, the search procedure starts at any site s. If there is $t \in L_s$ with $d(q,s) > (1+\epsilon')d(q,t)$, then assign t to s and repeat. Otherwise, return s. With L_s as defined, the returned s is ϵ-closest. Note that the procedure makes progress at each step: the distance of the current site decreases by $1/(1+\epsilon')$.

Consider the condition satisfied by L_s in a contrapositive way. Fixing $s \in S$, let $\mathcal{N}_\epsilon(S)$ be defined by

$$\mathcal{N}_\epsilon(S) \equiv \{q \mid d(q,s) \le (1+\epsilon)d(q,b) \text{ for all } b \in S\},$$

so that

$$\mathcal{N}_{\epsilon'}(L_s) = \{q \mid d(q,s) \le (1+\epsilon')d(q,b) \text{ for all } b \in L_s\}.$$

The condition on L_s is equivalent to $\mathcal{N}_{\epsilon'}(L_s) \subset \mathcal{N}_\epsilon(S)$. The set $\mathcal{N}_\epsilon(S)$ is the Voronoi region of s in a certain multiplicatively weighted Voronoi diagram. The L_s we want is a small one that such that $\mathcal{N}_{\epsilon'}(L_s)$ is inside $\mathcal{N}_\epsilon(S)$; the problem of finding such an L_s can be solved by techniques previously applied to polytope approximation.[Cla93]

The following subsection discusses the problem of finding L_s; §2.2 bounds the size of such a list; §2.3 shows how to use the lists to obtain fast query times. Finally, §3 makes some concluding remarks.

2.1 Finding L_s

To find L_s, we'll translate the problem to $d+1$ dimensions using standard "lifting map" techniques.

If we put s at the origin, the region $\mathcal{N}_\epsilon(S)$ is the intersection of all regions of the form

$$\{z \mid z^2 \le (1+\epsilon)^2(z-b)^2\},$$

where $b \in S$. The condition here is $z^2/(1+\epsilon)^2 \le (z-b)^2$, or $\alpha z^2 \ge 2b\cdot z - b^2$, where $\alpha \equiv 1 - 1/(1+\epsilon)^2 \approx 2\epsilon$. Now let (z,y) denote a point in R^{d+1}, with $z \in R^d$ and $y \in R$. We have

$$\mathcal{N}_\epsilon(S) = \{z \mid \alpha y \ge 2b\cdot z - b^2 \text{ and } y = z^2\}.$$

So for $b \in S$, let $\mathcal{H}_{\epsilon,b}$ denote the halfspace

$$\mathcal{H}_{\epsilon,b} \equiv \{(z,y) \mid \alpha y \ge 2b\cdot z - b^2\},$$

and $\mathcal{P}_\epsilon(S)$ denote the polytope which is the intersection of such halfspaces,

$$\mathcal{P}_\epsilon(S) \equiv \bigcap_{b\in S} \mathcal{H}_{\epsilon,b}.$$

Then

$$\mathcal{N}_\epsilon(S) = \{z \mid (z,y) \in \mathcal{P}_\epsilon(S) \text{ and } y = z^2\}.$$

Let $\Psi \equiv \{(z,y) \mid y \ge z^2\}$. Then for $\mathcal{N}_{\epsilon'}(L_s) \subset \mathcal{N}_\epsilon(S)$, it is enough that

$$\mathcal{P}_{\epsilon'}(L_s) \cap \Psi \subset \mathcal{P}_\epsilon(S) \cap \Psi. \tag{1}$$

The problem of finding L_s satisfying this condition can be solved using techniques as for polytope approximation[Cla93], which are very similar to techniques described long ago for linear programming[Cla88]. The general idea is this: give each site an integral weight, initially one. Take a random subset R of S, choosing each site with probability proportional to its weight, and making R about the right size. Test if R satisfies (1); if so, return it. Otherwise, we can derive from the testing of R a subset of S that we can expect to be small, and that must contain a site of an appropriate L_s. Double the weights of the sites in that subset. Repeat this process until done.

Here are a few more details of the algorithm: fix some optimal L satisfying (1), where the size c of C is as small as possible. Assume for the moment that c is known. (By using exponentially increasing estimates of c, we lose only a small factor in the size of the returned set, and in running time.) Give each $p \in S$ a weight, initially equal to one. Choose a random subset $R \subset S$ of size $cC_1 d \log c$, where C_1 is a small constant, choosing a site with probability proportional to its weight. For each $b \in S$, check that

$$\mathcal{P}_{\epsilon'}(R) \cap \Psi \subset \mathcal{H}_{\epsilon,b}. \qquad (2)$$

This is a convex programming problem, and as shown by Adler and Shamir, it is solvable in $O(n)$ expected time using a randomized procedure similar to one for linear programming.[AS90] (The base case, with $O(d^2)$ constraints, can be solved in polynomial time.[Vai89]) If all $b \in S$ satisfy (2), then return R as L_s. Suppose the condition fails for some $\hat{b} \in S$. Then there will be some vertex v of $\mathcal{P}_{\epsilon'}(R) \cap \Psi$ not in $\mathcal{H}_{\epsilon,\hat{b}}$; such a vertex can be returned as part of the output of the convex programming algorithm. Find all the halfspaces $\mathcal{H}_{\epsilon',b}$ that do not contain v, for $b \in S$. This set V of halfspaces will have relatively few members, with high probability, and will contain a member of the optimal set C. If the size of set V is less than $C_2 dn/r$, then double the weights of the corresponding sites. (Here C_2 is another small constant.) Repeat this procedure until a list L_s is returned.

An analysis of this procedure appears in [Cla93], and will appear in the full paper.

2.2 The size of L_s

This algorithm returns a set L_s of size within $O(d \log c)$ of the best possible size c, as can be shown as in [Cla93]. How large can c be? We have the following bound, which this subsection will prove.

Theorem 2.1 *The optimal size c of L_s is $O(1/\epsilon)^{(d-1)/2} d \log(\rho/\epsilon)$.*

We'll need the following lemma, due to Dudley.[Dud74]

Lemma 2.2 *Let $P \subset R^d$ be compact and convex, and contained in a ball of radius 1. For ϵ with $0 < \epsilon < 1$, there is a convex polytope $P' \supset P$ with $O(1/\epsilon)^{(d-1)/2}$ facets, and with P' within Haussdorf distance ϵ of P.*

We'll apply this lemma to bound the size of L_s. To do so, split Ψ into slabs

$$\Psi_i \equiv \{(z,y) \mid d_i \le y \le d_{i+1}\},$$

for $i = 0 \ldots m$, where $\sqrt{d_0} \equiv (1/2)\min_{b \in S} \|b\|$, and $d_i = 3d_{i-1}/2$, and m is large enough that $\sqrt{d_m} > (2/\alpha)\max_{b \in S} \|b\|$. We have $m = \log O(\rho/\alpha)$. The following lemma implies that only the points of these slabs need be considered.

Lemma 2.3 *Every halfspace $\mathcal{H}_{\epsilon,b}$ contains all points of Ψ not in some Ψ_i.*

Proof. We need to show that $y \ge z^2$ and either $y \ge d_m$ or $y \le d_0$ imply $\mathcal{H}_{\epsilon,b}$ for any $b \in S$. Since $b \cdot z$ is maximized for given $\|z\|$ when $z = b\|z\|/\|b\|$, when showing that $(z,y) \in \mathcal{H}_{\epsilon,b}$ we can assume $z = \gamma b$ for some γ. If $z^2 \ge d_m$, the minimum value of y to consider is z^2, so it's enough to show that $\alpha z^2 \ge 2b \cdot z - b^2$, or $\alpha\gamma^2 b^2 \ge 2\gamma b^2 - b^2$. This is implied by $\gamma \ge (2/\alpha)$. If $z^2 \le d_m$, then similarly we need $\alpha y \ge 4b^2/\alpha \ge 2\gamma b^2 - b^2$, which holds for $\gamma \le 2/\alpha$. When $y \le d_0$, we have $z^2 \le y \le b^2/4$, and so we need $\alpha z^2 \ge 2\gamma b^2 - b^2$, where $\gamma \le 1/2$. Since $\alpha z^2 \ge 0$, the result follows. □

Consider a given slab Ψ_i. We have

$$\alpha d_i \ge \frac{5}{4}\alpha' d_{i+1}, \qquad (3)$$

for sufficiently small ϵ. Consider the sets

$$\mathcal{P}^a \equiv \bigcap_{b \in S} \{(z,y) \mid \alpha' d_{i+1} \ge 2b \cdot z - b^2\},$$

and

$$\mathcal{P}^b \equiv \bigcap_{b \in S} \{(z,y) \mid \alpha d_i \ge 2b \cdot z - b^2\}.$$

Then (*considering only points in Ψ_i*)

$$\mathcal{P}_{\epsilon'}(S) \subset \mathcal{P}^a \subset \mathcal{P}^b \subset \mathcal{P}_\epsilon(S).$$

Moreover, condition (3) implies that there is a gap between $\mathcal{P}^a$ and $\mathcal{P}^b$ that we'll use shortly. Now let ball

$$B' \equiv \{z \mid z^2 \le d_{i+1}\},$$

so that all points in Ψ_i have z projections in B', and let

$$B \equiv \{z \mid z^2 \le (1+2\epsilon)^2 d_{i+1}\}.$$

Let $\mathcal{P}_p^a \equiv \{z \mid (z,y) \in \mathcal{P}^a\}$, and similarly define $\mathcal{P}_p^b$. Then $\mathcal{P}_p^a \cap B' \subset \mathcal{P}_p^b \cap B$, and by (3), every point of the former is at least

$$\frac{\alpha' d_{i+1}/4}{2\|b\|}$$

from any point not in the latter. If a plane $\mathcal{H}_{\epsilon',b}$ does not contain Ψ_i, and so is relevant here, then as in the proof of Lemma 2.3, we have $d_{i+1} \geq b^2/4$. This says that gap between $\mathcal{P}_p^a \cap B'$ and $\mathcal{P}_p^b \cap B$ is proportional to $\epsilon\sqrt{d_{i+1}}$, and so by appropriate scaling we can apply Lemma 2.2 to find a polytope $\mathcal{P}_p^c$ such that

$$\mathcal{P}_p^a \cap B' \subset \mathcal{P}_p^c \subset \mathcal{P}_p^b \cap B$$

and $\mathcal{P}^c$ has $O(1/\epsilon)^{(d-1)/2}$ facets. This implies that there is a polytope $\mathcal{P}^c$ in R^{d+1} of the same complexity, such that relative to Ψ_i,

$$\mathcal{P}_{\epsilon'}(S) \subset \mathcal{P}^c \subset \mathcal{P}_\epsilon(S).$$

It not hard to show that there is a "coarsening" polytope $\mathcal{P}_{\epsilon'}(L_s^i)$, with at most d times as many facets as $\mathcal{P}^c$, such that $\mathcal{P}_{\epsilon'}(L_s^i) \cap \Psi_i \subset \mathcal{P}_\epsilon(S)$.

By putting the lists L_s^i together into L_s, we obtain the size bound of the theorem of this subsection.

2.3 Solving closest-point problems

How can a fast query time be obtained using the lists L_s? Just as with Arya and Mount's work, a skip-list approach is helpful.[Pug90] Choose a family of subsets of S as follows: let $R_0 \equiv S$; to obtain R_{j+1} from R_j, pick each element of R_j to be in R_{j+1} with probability 1/2. Repeat this process until an empty R_k is obtained. If $s \in R_j$ but not R_{j+1}, say that s has level j. Construct the lists $L_{s,j}$ for each R_j, and so $s \in S$ has lists for each subset up to its level. To answer a query, start with some $s \in R_{k-1}$, and find the ϵ-closest site t_{k-1} in R_{k-1} using the lists $L_{s,k-1}$. Now find an ϵ-closest site t_{k-2} in R_{k-2}, starting with t_{k-1}. Repeat until t_0 is found, and return t_0 as an ϵ-closest site in S.

The correctness of this procedure should be clear. How much time does it take? Since each list is bounded in size by $O(dc\log c)$, where c is $O(1/\epsilon)^{(d-1)/2} d\log(\rho/\epsilon)$, the query time is equal to $O(dc\log c)$ times the number of sites visited in the procedure.

It is worthwhile to compare this procedure with one that finds the closest site in R_j at stage j, not just the ϵ-closest. Suppose we have t_j as the ϵ-closest at some stage, but indeed a site t is closest in R_j. When finding the ϵ-closest in R_{j+1}, the approximate procedure will in two steps find a site t' in R_{j+1} such that $d(q,t') \leq d(q,t)/(1+\epsilon')^2$. (Here we assume that the search in R_{j+1} takes at least two steps.) Since $(1+\epsilon')^2 \geq (1+\epsilon)$ for $\epsilon \geq 0$, we know that t' is closer to q than t. The number of sites visited at stage $j+1$ for the exact procedure is proportional to the number of sites of R_{j+1} closer to q than t; hence the number of sites visited for the approximate procedure in R_{j+1} is no more than 2 plus the number for the exact procedure.

To analyze the exact search procedure, we can follow Sen's analysis of skip lists.[Sen] Look at the search procedure "backwards": starting at the closest site to q in R_j, visit sites in R_j in order of increasing distance, until a site also in R_{j+1} is encountered. Call this a *level jump*. Once the level jump occurs, only sites in R_{j+1} are visited. The probability of a level jump at a given visited node is 1/2. Thus the probability that at least k level jumps occur in v node visits is the probability that a binomially distributed random variable has at least k successes in v trials. The query time can be greater than V only if either the number of level jumps exceeds K or if fewer than K level jumps occur in V attempts; the former probability is no more than $n/2^K$, which we'll need less than some probability P_1. This implies $K \geq \lg(n/P_1)$. The probability of fewer than K level jumps in V trials can be bounded using Chernoff bounds for the binomial; letting $\gamma \equiv 2K/V$, it is $\exp(-V(1-\gamma)^2/2)$. The probability that the query time exceeds $2\lg(n/P_1)/\gamma$ for a given point q is therefore at most $P_1 + \exp(-\lg(n/P_1)(1-\gamma)^2/\gamma)$. Hence, setting $P_1 = 1/n^Q$, an $O(Q)\log n$ query time is achievable with failure probability $O(1/n^Q)$.

This analysis applies only to a single given point q; what about arbitrary points? As with similar situations in randomized geometric algorithms, a good query time holds for all points because there are $n^{d^{O(1)}}$ combinatorially distinct classes of points. That is, in an exact search algorithm, two points q_1 and q_2 will have the same sequence of visited sites, and so the same query time, if the distance order on the sites induced by the two points is the same. In other words, whether we sort the sites in order of distance from q_1, or sort them in order of distance from q_2, we get the same sorted order. How many classes of points are distinct in this way? Let $\mathcal{B}$ be the set of $\binom{n}{2}$ perpendicular bisector hyperplanes of pairs of sites, and let $\mathcal{A}(\mathcal{B})$ be the subdivision of R^d induced by those bisectors. Then all points in one cell (block) of $\mathcal{A}(\mathcal{B})$ induce the same distance orders, and so have the same query time. The number of cells of $\mathcal{A}(\mathcal{B})$ is $\binom{\binom{n}{2}}{d} < n^{2d}$. Thus a query time for any point of

$O(\log n)$ occurs with probability $1 - 1/n^{\Omega(1)}$.

Queries can be made a bit faster by splitting up the each list $L_{s,j}$ into lists $L^i_{s,j}$, where the superscript corresponds to the slabs Ψ_i in §2.2. When searching for a given site s at a given stage j, the list $L^i_{s,j}$ with $i = 2\lg d(s,q)$ can be used. This gives a query time independent of ρ.

3 Concluding Remarks

It should be possible to have an algorithm that is polynomial in d, by making ϵ a sufficiently large constant, as in Bern's note.[Ber93]

References

[AM93] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 271–280, 1993.

[AMN+94] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and Angela Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, 1994.

[AS90] I. Adler and R. Shamir. A randomization scheme for speeding up algorithms for linear and convex quadratic programming problems with a high constraints-to-variables ratio. Technical Report 21-90, Rutgers Univ., May 1990. To appear in *Math. Programming.*

[Ber93] M. Bern. Approximate closest-point queries in high dimensions. *Inform. Process. Lett.*, 45:95–99, 1993.

[Cla88] K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 452–456, 1988. Revised version: Las Vegas algorithms for linear and integer programming when the dimension is small (preprint).

[Cla93] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop Algorithms Data Struct.*, Lecture Notes in Computer Science, 1993.

[Dud74] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approximation Theory*, 10:227–236, 1974.

[Pug90] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 35:668–676, 1990.

[Sen] S. Sen. Some oberservations on skip lists.

[Vai89] P. M. Vaidya. A new algorithm for minimizing convex functions over convex sets. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 338–343, 1989.

[Yao82] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11:721–736, 1982.