EXHIBIT 6

Excluded Middle Vantage Point Forests for Nearest Neighbor Search

Peter N. Yianilos*

July 20, 1998
(revised August 1, 1999)

Abstract

The excluded middle vantage point forest is a new data structure that supports *worst case* sublinear time searches in a metric space for nearest neighbors within a fixed radius τ of arbitrary queries. Worst case performance depends on the dataset but is not affected by the distribution of queries.

Our analysis predicts vp-forest performance in simple settings such as L_p spaces with uniform random datasets — and experiments confirm these predictions. Another contribution of the analysis is a new perspective on the *curse of dimensionality* in the context of our methods and kd-trees as well. In our idealized setting the dataset is organized into a forest of $O(N^{1-\rho})$ trees, each of depth $O(\log N)$. Here ρ may be viewed as depending on τ, the distance function, and on the dataset. The radius of interest τ is an input to the organization process and the result is a linear space data structure specialized to answer queries within this distance. Searches then require $O(N^{1-\rho} \log N)$ time, or $O(\log N)$ time given $O(N^{1-\rho})$ processors.

Our conclusion is that these new data structures exhibit useful behavior only for small radius searches, where despite their variation in search times, conventional kd-trees perform much better.

Keywords: Nearest neighbor search, Vantage point tree (vp-tree), kd-tree, Computational geometry, Metric space.

*The author is with NEC Research Institute, 4 Independence Way, Princeton, NJ. Email: pny@research.nj.nec.com

1 Introduction

We consider the radius-limited nearest neighbor problem in a metric space. That is, given an N point dataset and a radius of interest τ, produce a data structure and associated search algorithm to rapidly locate the dataset point nearest to any query q. Our focus is on practical data structures that provide worst-case search time bounds.

Vantage point trees (vp-trees) [32] and kd-trees [15, 16, 4, 3] organize an N point dataset so that sublinear time nearest neighbor searches may be performed on an *expected* basis for some fixed distribution. Performance depends on the dataset and on the assumed distribution of queries.

The excluded middle vantage point forest (vp-forest) is a new related data structure that supports *worst case* sublinear time searches for nearest neighbors within a fixed radius τ of arbitrary queries. Worst case performance depends on the dataset but is not affected by the distribution of queries.

The dataset is organized into a forest of $O(N^{1-\rho})$ trees, each of depth $O(\log N)$. Here ρ may be viewed as depending on τ, the distance measure, and on the dataset. The radius of interest τ is an input to the organization process and the result is a data structure specialized to answer queries within this distance.

Each element of the dataset occurs in exactly one tree so that the entire forest remains linear space. Searches follow a single root-leaf path in each tree. There is no backtracking when the search is limited to neighbors within distance τ. Along its way *every* neighbor within τ is necessarily encountered. The

query's effect is to guide the descent through each tree.

There are no significant ancillary computational burdens at search time. So upon creating the forest, the user simply adds the depths of each tree in the forest to arrive at the maximum number of distance evaluations each search will require. Each tree may be searched independently. Searches then require $O(\log N)$ time given one processor for each tree in the forest.

We also discuss design variations that trade space for reductions in search time — and compare forests with single trees constructed similarly.

The general idea behind vp-forests is easily understood. Both vp-trees and kd-trees recursively divide the dataset. At each node the remaining dataset elements have an associated value, and the node has a corresponding fixed threshold that is roughly central in the distribution of values. Elements below this threshold are assigned to, say, the left child, and those above to the right. For kd-trees these values are those of individual coordinates within each data vector. For vp-trees they are the distance of a metric space element to some fixed *vantage point*.

Elements near to the threshold lead to backtracking during search. When building a vp-forest, such elements are deleted from the tree and added instead to a *bucket*. Once the tree is complete, the bucket is organized into a tree in the same way, resulting in another (smaller) bucket of elements. This continues until the forest is built. This effectively eliminates backtracking. Because elements near to the threshold are recursively deleted, and this threshold lies near the middle of the distribution of values, we refer to our data structure as an *excluded middle vantage point forest*. Both vp-trees and kd-trees may be regarded as trivial instances of vp-forests with no excluded middle.

We present an idealized analysis that allows us to predict vp-forest performance in simple settings such as L_p spaces with uniform random datasets. Experiments are reported that confirm these predictions. One contribution of this analysis is an interesting new perspective on the so called *curse of dimensionality* (that is that nearest neighbor search increases in difficulty with dimension). In Euclidean space given a uniform random dataset drawn from the hypercube, we observe that the worst-case difficulty of vp-forest search for any fixed τ ought to be asymptotically constant with respect to dimension — and our experiments confirm this. Using L_1 we expect constant difficulty if τ is allowed to increase with $d^{1/2}$ (d denotes dimension), and this too is confirmed by experiment.

Our analysis also suggests that kd-trees should exhibit the same dimension invariance for fixed search radii, and experiments confirm this. For a worst case query, kd-tree search visits essentially the entire dataset, but on average performs far less work than the vp-forest.

The vp-forests described in this report stimulated the developments of [33], but do not themselves appear to have immediate practical value.

We conclude our introduction with a brief discussion of the nearest neighbor search problem and literature. See [32] for additional discussion.

Nearest neighbor search is an important task for non-parametric density estimation, pattern recognition, information retrieval, memory-based reasoning, and vector quantization. See [11] for a survey.

The notion of a mathematical metric space [20] provides a useful abstraction for *nearness*. Examples of metric spaces include Euclidean space, the Minkowski L_p spaces, and many others. Exploiting the metric space triangle inequality to eliminate points during nearest neighbor search has a long history. Our work belongs to this line. This paper extends it by i) introducing structures that give worst case time bounds for limited radius searches, ii) providing analysis for them, iii) introducing a method for trading space for time, and iv) defining our data structures and algorithms in terms of abstract *projectors*, which combines approaches that use distance from a distinguished element with those such as kd-trees that use the value of a distinguished coordinate.

In early work Burkhard and Keller [7] describe methods for nearest neighbor retrieval by evaluating distances from distinguished elements. Their data structures are multi-way trees corresponding to integral valued metrics.

Fukunaga in [18, 19] exploits clustering techniques [27] to produce a hierarchical decomposition of Euclidean Space. During a branch and bound search,

the triangle inequality is used to rule out an entire cluster if the query is far enough outside of it. While exploring a cluster he observes that the triangle inequality may be used to eliminate some distance computations. A key point missed is that when the query is well inside of a cluster, the exterior need not be searched.

Collections of graphs are considered in [14] as an abstract metric space with a metric assuming discrete values only. This work is related to the constructions of [7]. In their concluding remarks the authors clearly anticipate generalization to continuous settings such as $\mathbb{R}^n$.

The idea that vantage points near the *corners* of the space are better than those near the center was described in [28] and much later in [32].

More recent papers describing vantage-point approaches are [30, 29, 25] and [32] who describe variants of what we refer to as a vantage-point tree. Also see [10] for very recent work on search in metric spaces.

The well-known *kd-tree* of Friedman and Bentley [15, 16, 4, 3] recursively divides a pointset in $\mathbb{R}^d$ by projecting each element onto a distinguished coordinates. Improvements, distribution adaptation, and incremental searches, are described in [13], [21], and [6] respectively. In our framework kd-trees correspond to *unit vector projection* with the canonical basis.

More recently, the Voronoi digram [2] has provided a useful tool in low- dimensional Euclidean settings – and the overall field and outlook of Computational Geometry has yielded many interesting results such as those of [31, 9, 8, 17] and earlier [12]. It appears that [12] may be the first work focusing on worst case bounds.

Very recently Kleinberg [22] gives two algorithms for an approximate form of the nearest neighbor problem. The space requirements of the first are prohibitive but the second, which almost always finds approximate nearest neighbors seems to to be of more practical interest. The recent work reported in [1] also considers an approximate form of the problem. Their analysis gives exponential dependence on d but the heuristic version of their approach they describe may be of practical interest.

For completeness, early work dealing with two special cases should be mentioned. Retrieval of similar binary keys is considered by Rivest in [26] and the L_∞ setting is the focus of [34].

See [5] for worst case data structures for the *range search* problem. This problem is related to but distinct from nearest neighbor search since a neighbor can be nearby even if a single coordinate is distant. But the L_∞ nearest neighbor problem may be viewed as an instance of range search. Their paper also describes a particular approach to trading space for time via an overlapping cover. Our discussion of this topic in section 5 also takes this general approach.

2 Vantage Point Forests

We begin by formalizing the ideas and construction sketched in the introduction.

Definition 1 *Consider an ordered set* $X = \{x_1, \ldots, x_N\}$ *and a value* $m \in [0, 1]$. *Let* $w = \lfloor mN \rfloor$ *and* $a = \lfloor (N - w)/2 \rfloor$. *Then the m-split of X consists of left, middle, and right subsets defined by:*

$$\begin{aligned} L &= \{x_i | i \leq a\} \\ M &= \{x_i | i > a, i \leq a + w\} \\ R &= \{x_i | i > a + w\} \end{aligned}$$

That is, a balanced 3-way partition of X with a central proportion of approximately m.

Algorithm 1 *Given a collection of points H and a 1-1 projection function* $\pi_G : H \to \mathbb{R}$ *defined for any nonempty* $G \subseteq H$, *define* $\pi_G(G)$ *to be the ordered set of distinct real values corresponding to the image of G under* π_G. *Now for* $m \in [0, 1]$:

1. *consider the m-split* L, M, R *of* $\pi(G)$, *and define the split* G_L, G_M, G_R *of G corresponding to the preimages of* L, M, R *respectively.*

2. *Given* $G \subseteq H$ *construct a binary tree by forming* G_L, G_M, G_R, *discarding* G_M, *and then doing the same recursively for* G_L *and* G_R *until single elements remain forming the tree's leaves.*

3. *Starting with H build a tree T_1 as described above and denote its membership by M_1. Let $H_0 = H$ and define $H_1 = H_0 - M_1$, i.e. the elements discarded building T_1.*

4. *For $k > 1$ and $H_{k-1} \neq \emptyset$ define T_k as the tree built as above for H_{k-1}, denote the tree's membership by M_k, and define $H_k = H_{k-1} - M_k$.*

Definition 2 *We refer to the result of algorithm 1 as the idealized excluded middle vantage point forest induced by π with central proportion m. When H is a metric space and the projection functions π satisfy $|\pi(x) - \pi(y)| \leq d(x, y), \forall x, y \in H$, this forest is of use for nearest neighbor search and we further define τ to be one half of the minimum diameter of a middle set discarded during construction.*

We remark that each tree of the forest may be viewed as analogous to the Cantor set from real analysis. That is, the subset of $[0,1]$ constructed by removing the central third of the interval — and proceeding recursively for both the left and right thirds. Our forest then corresponds to a decomposition of the space into a union of Cantor sets.

Two important examples of a suitable family of π functions are *vantage point projection* for general metric spaces, and *unit vector projection* for Euclidean space.

A vantage point projector π_p is defined for any $p \in H$ by $\pi_p(x) = d(p, x)$. The split points in tree construction correspond to abstract spheres about p. It is easily verified that $|\pi_p(x) - \pi_p(y)| \leq d(x, y)$ as required. The range of this projector is the nonnegative reals. Letting $m = 0$ gives rise to a *vantage point tree* [32].

The unit vector projector π_p is defined for any $p \neq 0$ as $\pi_p(x) =< p, x > /\|p\|$. This is easily seen to satisfy the required inequality as well, and its range is not limited to nonnegative values. Here the split points correspond to hyperplanes. Choosing p as canonical unit vectors and letting $m = 0$ builds a form of kd-tree [15, 16, 4, 3]. It is important to note that $< p, x >$ may be computed more rapidly for canonical unit vectors — in constant time with respect to dimension. Also, we remark that whenever orthogonal vectors are used, projection distances are, in a sense, additive, and that this fact can be exploited (as kd-trees do) when designing solutions specialized for L_p spaces. We do not consider either of these optimizations in this paper.

Proposition 1 *Consider an idealized Vantage Point Forest with central proportion m and corresponding value τ. Define $\rho = 1/(1 - \log_2(1 - m))$. Then:*

1. *There are $\Theta(N^{1-\rho})$ trees in the forest, having maximum depth $\Theta(\log N)$.*

2. *A search for nearest neighbors within distance τ of a query requires $\Theta(N^{1-\rho} \log N)$ time, and linear space — independent of the query.*

3. *The search requires $\Theta(\log N)$ time given $\Theta(N^{1-\rho})$ independent processors.*

4. *Assuming each projector π_G can be constructed in constant time, and that it can also be evaluated in constant time, and that $m > 0$, then $O(N^{2-\rho})$ time is required to construct the forest.*

proof: At each step in the construction the central proportion of m elements is removed so that the left and right subsets are each of size $(1 - m)/2$. The first tree's depth is then $[1/log_2(2/(1 - m))]log_2 N = \Theta(\log N)$. So the number of elements left in the tree is $N^{1/(1-log_2(1-m))} = N^{\rho}$.

The number of elements left after the first tree has been constructed is then $N - N^{\rho}$. After the second $N - N^{\rho} - (N - N^{\rho})^{\rho}$ are left, and so on. Clearly $\Omega(N^{1-\rho})$ trees are required to reduce the population to any fixed size.

Once the population has been reduced to $1/2$ of its original size, the number of elements removed as each tree is built will have declined to $(N/2)^{\rho} = (1/2)^{\rho} N^{\rho}$. Since $(1/2)^{\rho} > 1/2$ no fewer than $N^{\rho}/2$ are removed. So the number of steps required to reach $N/2$ is no more than $N^{1-\rho}$. The number required to reach $N/4$ is then $(1/2)^{1-\rho} N^{1-\rho}$ — and so on forming a geometric series. So the number required to reach any fixed level is $O(N^{1-\rho})$. The number of trees in the forest is then $\Theta(N^{1-\rho})$ — and the total space is clearly linear.

A search for nearest neighbors within distance τ is then made by following a single root-leaf path in

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.