Algorithms for Finding Nearest Neighbors (and Relatives)

Piotr Indyk

Helsinki, May 2007
Definition

- Given: a set $P$ of $n$ points in $\mathbb{R}^d$
- Nearest Neighbor: for any query $q$, returns a point $p \in P$ minimizing $||p-q||$
- $r$-Near Neighbor: for any query $q$, returns a point $p \in P$ s.t. $||p-q|| \leq r$ (if it exists)
Nearest Neighbor: Motivation

- Learning: nearest neighbor rule
MNIST data set “2”
Nearest Neighbor: Motivation

- Learning: nearest neighbor rule
- Database retrieval
- Vector quantization, compression/clustering
Brief History of NN
The case of $d=2$

- Compute Voronoi diagram
- Given $q$, perform point location
- Performance:
  - Space: $O(n)$
  - Query time: $O(\log n)$
The case of $d > 2$

- Voronoi diagram has size $n^{O(d)}$
- We can also perform a linear scan: $O(dn)$ time
- That is pretty much all what known for exact algorithms with theoretical guarantees
- In practice:
  - kd-trees work “well” in “low-medium” dimensions
Approximate Near Neighbor

- c-Approximate Nearest Neighbor: build data structure which, for any query $q$
  - returns $p' \in P$, $||p-q|| \leq cr$,
  - where $r$ is the distance to the nearest neighbor of $q$
Plan

• Intro
• (Main memory) data structures:
  – Today: Kd-trees
    • Low-medium dimensions
    • A proud member of a (huge) family of tree-based data structures
  – Tomorrow: Locality Sensitive Hashing (LSH)
    • Dimensionality does not really matter (but other things do)
Kd-tree
Kd-trees [Bentley’75]

• Not the most efficient solution in theory
• Everyone uses it in practice
• Algorithm:
  – Choose x or y coordinate (alternate)
  – Choose the median of the coordinate; this defines a horizontal or vertical line
  – Recurse on both sides
• We get a binary tree:
  – Size: $O(N)$
  – Depth: $O(\log N)$
  – Construction time: $O(N \log N)$
Kd-tree: Example

Each tree node \( v \) corresponds to a region \( \text{Reg}(v) \).
Searching in \textit{kd}-trees

- Range Searching in 2D
  - Given a set of \textit{n} points, build a data structure that for any query rectangle $R$, reports all points in $R$
Kd-tree: Range Queries

1. Recursive procedure, starting from \( v=\text{root} \)

2. Search \((v,R)\):
   a) If \( v \) is a leaf, then report the point stored in \( v \) if it lies in \( R \)
   b) Otherwise, if \( \text{Reg}(v) \) is contained in \( R \), report all points in the subtree of \( v \)
   c) Otherwise:
      • If \( \text{Reg(left}(v)) \) intersects \( R \), then Search(left(v),R)
      • If \( \text{Reg(right}(v)) \) intersects \( R \), then Search(right(v),R)
Query demo

Helsinki, May 2007
Query Time Analysis

• We will show that Search takes at most $O(n^{1/2}+P)$ time, where $P$ is the number of reported points
  – The total time needed to report all points in all sub-trees (i.e., taken by step b) is $O(P)$
  – We just need to bound the number of nodes $v$ such that $\text{Reg}(v)$ intersects $R$ but is not contained in $R$. In other words, the boundary of $R$ intersects the boundary of $\text{Reg}(v)$
  – Will make a gross overestimation: will bound the number of $\text{Reg}(v)$ which are crossed by any of the 4 horizontal/vertical lines
Query Time Continued

• What is the max number $Q(n)$ of regions in an $n$-point kd-tree intersecting (say, vertical) line?
  – If we split on $x$, $Q(n)=1+Q(n/2)$
  – If we split on $y$, $Q(n)=2*Q(n/2)+2$
  – Since we alternate, we can write $Q(n)=3+2Q(n/4)$

• This solves to $O(n^{1/2})$
Exercises

• Construct a set of \( n \) points, and a range query \( R \) such that:
  – \( R \) does not contain any of the points
  – The search procedure takes \( \Omega(n^{1/2}) \) time

• What happens if the query range is a circle, not a square?
Back to \((1+\epsilon)\)-Nearest Neighbor

• We will solve the problem using kd-trees
• “Analysis”…under the assumption that all leaf cells of the kd-tree for \(P\) have bounded aspect ratio
• Assumption somewhat strict, but satisfied in practice for most of the leaf cells
• We will show
  – \(O(\log n \times O(1/\epsilon^d))\) query time
  – \(O(n)\) space (inherited from kd-tree)
ANN Query Procedure

- Locate the leaf cell containing $q$
- Enumerate all leaf cells $C$ in the increasing order of distance from $q$ (denote it by $r$)
- Keep updating $p'$ so that it is the closest point seen so far
  - Note: $r$ increases, $\text{dist}(q,p')$ decreases
- Stop if $\text{dist}(q,p') < (1+\varepsilon) \cdot r$

Helsinki, May 2007
Analysis

• Let $R$ be the value of $r$ before the last cell was examined.
• Each cell $C$ seen (except maybe for the last one) has diameter $> \varepsilon R$.
• …Because if not, then the point $p$ in $C$ would have been a $(1+\varepsilon)$-approximate nearest neighbor (by now), so we would have stopped earlier.
  \[
  \text{dist}(q,p) \leq \text{dist}(q,C) + \text{diameter}(C) \leq R + \varepsilon R = (1+\varepsilon)R
  \]
• The number of cells with diameter $\varepsilon R$, bounded aspect ratio, and touching a ball of radius $R$ is at most $O(1/\varepsilon)^d$.
  – Ball of radius $R$ has volume $O(R)^d$.
  – Each cell has volume $\Omega((\varepsilon R/\sqrt{d}))^d$.
Refs


• D Lowe, 1992.

Helsinki, May 2007