“Geometric” data structures:
Announcements:

• HW3 posted

• Today:
  – Review: LSH for Euclidean distance
  – Other ideas: KD-trees, ball trees, cover trees
Image Search...

Organic Authority

5 Bitter Melon Recipes: The Ancient Healing Fruit

bitter melon stir fry

Images may be subject to copyright.
LSH for Euclidean distance

- The family of hash functions:

- Recall $R$, $cR$, $P1$, $P2$

- Pre-processing time:

- Query time:
What other guarantees might we hope for?

- Recall sorting:
- LSH:
- Voronoi:

- How about other "geometric" data structures?
- What is the ‘key’ inequality to exploit?
Smarter approach: **kd-trees**

- Structured organization of documents
  - Recursively partitions points into axis aligned boxes.
- Enables more efficient pruning of search space
  - Examine nearby points first.
  - Ignore any points that are further than the nearest point found so far.

**kd-trees** work “well” in “low-medium” dimensions

- We’ll get back to this...
Start with a list of $d$-dimensional points.

<table>
<thead>
<tr>
<th>Pt</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1.00</td>
<td>4.31</td>
</tr>
<tr>
<td>3</td>
<td>0.13</td>
<td>2.85</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Split the points into 2 groups by:

- Choosing dimension $d_j$ and value $V$ (methods to be discussed...)
- Separating the points into $x_{d_j}^i > V$ and $x_{d_j}^i \leq V$. 

<table>
<thead>
<tr>
<th>Pt</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>1.00</td>
<td>4.31</td>
</tr>
<tr>
<td>3</td>
<td>0.13</td>
<td>2.85</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

©Sham Kakade 2017
Consider each group separately and possibly split again (along same/different dimension).

- Stopping criterion to be discussed...
- Consider each group separately and possibly split again (along same/different dimension).
  - Stopping criterion to be discussed...
 KD-Tree Construction

- Continue splitting points in each set
  - creates a binary tree structure
- Each leaf node contains a list of points
Keep one additional piece of information at each node:

- The (tight) bounds of the points at or below this node.
KD-Tree Construction

- Use heuristics to make splitting decisions:
- Which dimension do we split along?
- Which value do we split at?
- When do we stop?
Many heuristics...

median heuristic

center-of-range heuristic
Nearest Neighbor with KD Trees

- Traverse the tree looking for the nearest neighbor of the query point.
Examine nearby points first:

- Explore branch of tree closest to the query point first.
Nearest Neighbor with KD Trees

- Examine nearby points first:
  - Explore branch of tree closest to the query point first.
When we reach a leaf node:

- Compute the distance to each point in the node.
When we reach a leaf node:

- Compute the distance to each point in the node.
Then backtrack and try the other branch at each node visited
Each time a new closest node is found, update the distance bound
Using the distance bound and bounding box of each node:

- Prune parts of the tree that could NOT include the nearest neighbor
Nearest Neighbor with KD Trees

Using the distance bound and bounding box of each node:
- Prune parts of the tree that could NOT include the nearest neighbor
Using the distance bound and bounding box of each node:

- Prune parts of the tree that could NOT include the nearest neighbor
For (nearly) balanced, binary trees...

Construction

- Size:
- Depth:
- Median + send points left right:
- Construction time:

1-NN query

- Traverse down tree to starting point:
- Maximum backtrack and traverse:
- Complexity range:

Under some assumptions on distribution of points, we get $O(\log N)$ but exponential in $d$ (see citations in reading)
Complexity
K-NN with KD Trees

- Exactly the same algorithm, but maintain distance as distance to furthest of current $k$ nearest neighbors
- Complexity is:
**Approximate K-NN with KD Trees**

- Before: Prune when distance to bounding box >
- Now: Prune when distance to bounding box >
- Will prune more than allowed, but can guarantee that if we return a neighbor at distance $r$, then there is no neighbor closer than $r/\alpha$.
- In practice this bound is loose...Can be closer to optimal.
- Saves lots of search time at little cost in quality of nearest neighbor.

©Sham Kakade 2017
What about NNs searches in high dimensions?

- KD-trees:
  - What is going wrong?
  - Can this be easily fixed?

- What do have to utilize?
  - utilize triangle inequality of metric
  - New ideas: ball trees and cover trees
Ball Trees

Ball-tree Example

level 1

level 2

level 3

level 4
Node:
- Every node defines a ball (hypersphere), containing
  - a subset of the points (to be searched)
  - A center
  - A (tight) radius of the points

Construction:
- Root: start with a ball which contains all the data
- take a ball and make two children (nodes) as follows:
  - Make two spheres, assign each point (in the parent sphere) to its closer sphere
  - Make the two spheres in a “reasonable” manner
Ball Tree Search

- Given point $x$, how do find its nearest neighbor quickly?

- Approach:
  - Start: follow a greedy path through the tree
  - Backtrack and prune: rule out other paths based on the triangle inequality
    - (just like in KD-trees)

- How good is it?
  - Guarantees:
  - Practice:
Cover trees

- What about exact NNs in general metric spaces?

- Same Idea: utilize triangle inequality of metric (so allow for arbitrary metric)

- What does the dimension even mean?

- cover-tree idea:
Intrinsic Dimension

- How does the volume grow, from radius $R$ to $2R$?

- Can we relax this idea to get at the “intrinsic” dimension?

- This is the “doubling” dimension:
## NN complexities

<table>
<thead>
<tr>
<th></th>
<th>Query time</th>
<th>Space used</th>
<th>Preprocessing time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vornoi</strong></td>
<td>$O\left(2^d \log n\right)$</td>
<td>$O\left(n^{d/2}\right)$</td>
<td>$O\left(n^{d/2}\right)$</td>
</tr>
<tr>
<td><strong>Kd-tree</strong></td>
<td>$O\left(2^d \log n\right)$</td>
<td>$O\left(n\right)$</td>
<td>$O\left(n \log n\right)$</td>
</tr>
<tr>
<td><strong>LSH</strong></td>
<td>$O\left(n^\rho \log n\right)$</td>
<td>$O\left(n^{1+\rho}\right)$</td>
<td>$O\left(n^{1+\rho} \log n\right)$</td>
</tr>
</tbody>
</table>