Reverse kNN search in Arbitrary Dimensionality

Instructor: Cyrus Shahabi
Algorithms for finding NN

- Elementary methods:
  - Search Algorithm
  - Indexing Data Structure
  - BF
  - DFS
  - R-tree
  - R*-tree
  - NN solution

- More advanced methods:
  - Search Algorithm
  - Branch & Bound Methods
  - Indexing Data Structure
  - NN solution
  - Mindist
  - Maxdist
  - Minmaxdist
Reverse Nearest Neighbors Queries

What are the fire locations I’m nearest to?

Which houses I’m the closest restaurant to?
RNN Definition

- A data point $p$ is the reverse nearest neighbor of query point $q$, if there is no point $p'$ such that $\text{dist}(p', p) < \text{dist}(q, p)$, i.e. $q$ is the NN of $p$.

$\text{NN}(p_2) = \text{NN}(p_3) = q$
$\text{RNN}(q) = \{p_2, p_3\}$

- Is RNN a symmetric relation?
Related Work

Main idea
- Pre-computing
- Filter/refinement

Methods
- KM
- YL
- SAA
- SFT

RNN Algorithms
KM

- Original RNN method
  For all \( p \):
  1. Pre-compute \( \text{NN}(p) \)
  2. Represent \( p \) as a vicinity circle
  3. Index the MBR of all circles by an R-tree

  (Named RNN-tree)
  4. \( \text{RNN}(q) \) = all circles that contain \( q \)

- Needs two trees: RNN-tree & R-tree
KM (Cont.)

• YL: Merges the trees
• What happens if we insert $p_5$?
  $RNN(p_5)=$?

  1. Find all points that have $p_5$ as their new NN
  2. Update the vicinity circles of those points in the index
  3. Compute $NN(p_5)$ and insert the corresponding circle in the index
• Drawbacks?

Techniques that rely on pre-processing cannot deal efficiently with updates
SAA

• Elimination of the need for pre-computing all NNs in filter/ refinement methods

• SAA:
  – Divide the space around query into six equal regions
  – Find NN(q) in all regions (candidate keys)
    (prove by contradiction: p1 rNN(q) but p2 not!
  – Either (i) or (ii) holds for each candidate key p
    • (i) p is in RNN(q)
    • (ii) No RNN(q) in Si
  – RNN(q) = \{p_6\}

• Any Drawbacks?

The number of regions increases exponentially with the dimensionality
SFT

1. Find the \( k \)NNs of the query \( q \) (\( k \) candidates)
2. Eliminate the points that are closer to other candidates than \( q \).
3. Apply *Boolean range queries* to determine the actual RNNs
   - A Boolean range query terminates as the first data point is found
   - Drawbacks?

False misses
Choosing a proper \( k \)
Concluding former methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Dynamic data</th>
<th>Arbitrary dimensionality</th>
<th>Exact result</th>
</tr>
</thead>
<tbody>
<tr>
<td>KM, YL</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SAA</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SFT</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Half-plane pruning

• Can $p'$ be an RNN of $q$?

• If $p_1, p_2, \ldots, p_n$ are $n$ data points, then any node whose MBR falls inside $\bigcup_{i=1..n} \Pi p_i(p_i, q)$ cannot contain any RNN result.

• E.g., points inside N2 would have either $p_1$ or $p_2$ as their NN, hence they are not RNN of $q$
• Pruning an R-tree MBR:

• Drawbacks?

\[ O(n^2) \] processing time in terms of bisector trimming for computing \( N^{res} \)
Computation of intersections does not scale with dimensionality
• Approximating the residual MBR
• An MBR can be pruned if its residual region is empty

• The approximation is a superset of the real residual region

• We can prune an MBR if its approximate residual is empty

• Good news:

\[ O(n) \] processing time for computing \( N^{resM} \)

No more hyper-polyhedrons to make the intersection computation complex
TPL Algorithm

- The big picture
  - Uses best-first search
  - Utilizes one R-tree as the data structure
  - Includes filtering/ refinement phases
  - Uses candidate points to prune entries
  - Filters visited entries to obtain the set $S_{cnd}$ of candidates
  - Adds pruned entries to set $S_{rfn}$
  - $S_{rfn}$ is used in the refinement step to eliminate false hits
TPL Example

* Figures of this example are obtained from [2]
Filtering step

Action | Heap | $S_{cnd}$ | $S_{rfn}$
--- | --- | --- | ---
Visit root | $\{N_{10}, N_{11}, N_{12}\}$ | {} | {}
<table>
<thead>
<tr>
<th>Action</th>
<th>Heap</th>
<th>$S_{cnd}$</th>
<th>$S_{rfn}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit $N_{10}$</td>
<td>${N_3, N_{11}, N_2, N_1, N_{12}}$</td>
<td>${}$</td>
<td>${}$</td>
</tr>
<tr>
<td>Action</td>
<td>Heap</td>
<td>$S_{cmd}$</td>
<td>$S_{rfn}$</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>Visit $N_3$</td>
<td>${N_{11}, N_2, N_1, N_{12}}$</td>
<td>${p_1}$</td>
<td>${p_3}$</td>
</tr>
<tr>
<td>Action</td>
<td>Heap</td>
<td>$S_{cmd}$</td>
<td>$S_{rfn}$</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------</td>
<td>-----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Visit $N_{11}$</td>
<td>${N_5, N_2, N_1, N_{12}}$</td>
<td>${p_1}$</td>
<td>${p_3, N_4, N_6}$</td>
</tr>
<tr>
<td>Action</td>
<td>Heap</td>
<td>$S_{cmd}$</td>
<td>$S_{rfn}$</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Visit $N_5$</td>
<td>${N_2, N_1, N_{12}}$</td>
<td>${p_1, p_2}$</td>
<td>${p_3, N_4, N_6, p_6}$</td>
</tr>
<tr>
<td>Action</td>
<td>Heap</td>
<td>$S_{cnd}$</td>
<td>$S_{rfn}$</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Visit $N_1$</td>
<td>${N_{12}}$</td>
<td>${p_1, p_2, p_5}$</td>
<td>${p_3, N_4, N_6, p_6, N_2, p_7}$</td>
</tr>
<tr>
<td>Action</td>
<td>Heap</td>
<td>$S_{cnd}$</td>
<td>$S_{rfn}$</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>{}</td>
<td>${p_1, p_2, p_5}$</td>
<td>${p_3, N_4, N_6, p_6, N_2, p_7, N_{12}}$</td>
</tr>
</tbody>
</table>
Refinement Heuristics

- Let $P_{rfn}$ be the set of points and $N_{rfn}$ be the set of nodes in $S_{rfn}$
- A candidate point can be eliminated if it is closer to another candidate point than to the query
- A point $p$ from $S_{cnd}$ can be discarded as a false hit if either of the following hold:
  (i) there is a point $p' \in P_{rfn}$ such that $\text{dist}(p, p') < \text{dist}(p, q)$
  (ii) There is a node MBR $N \in N_{rfn}$ such that $\text{minmaxdist}(p, N) < \text{dist}(p, q)$
- A point $p$ from $S_{cnd}$ can be reported as an actual result if the following conditions hold:
  (i) There is no point $p' \in P_{rfn}$ such that $\text{dist}(p, p') < \text{dist}(p, q)$
  (ii) For every node $N \in N_{rfn}$ : $\text{mindist}(p, N) \geq \text{dist}(p, q)$
- If none of the above works, visit all node MBRs $N \in N_{rfn}$ where $\text{mindist}(p, N) < \text{dist}(p, q)$ and use the mentioned heuristics considering the newly visited entries
<table>
<thead>
<tr>
<th>Action</th>
<th>$\text{Scnd}$</th>
<th>$\text{Srfn}$</th>
<th>Actual results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalidate $p_1$</td>
<td>${p_2, p_5}$</td>
<td>${N_4, N_6, N_2, N_{12}}$</td>
<td>${}$</td>
</tr>
<tr>
<td>Validate $p_5$</td>
<td>${p_2}$</td>
<td>${N_4, N_6, N_2, N_{12}}$</td>
<td>${p_5}$</td>
</tr>
<tr>
<td>Remove $N_6, N_2$</td>
<td>${p_2}$</td>
<td>${N_4, N_{12}}$</td>
<td>${p_5}$</td>
</tr>
<tr>
<td>Action</td>
<td>Scnd</td>
<td>Srfn</td>
<td>Actual results</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Access $N_4$</td>
<td>${p_2}$</td>
<td>${N_4, N_{12}}$</td>
<td>${p_5}$</td>
</tr>
<tr>
<td>Invalidate $p_2$</td>
<td>${}$</td>
<td>${p_4, p_8, N_{12}}$</td>
<td>${p_5}$</td>
</tr>
</tbody>
</table>
RkNN pruning

- Return all points that have q as one of their \( k \) nearest neighbors

- Let \( \{\sigma_1, \sigma_2, \ldots, \sigma_k\} \) be a subset of \( \{p_1, p_2, \ldots, p_n\} \). Each of the \( \binom{n}{k} \) subsets, prunes the area \( \bigcap_{i=1}^{k} \text{PL} \sigma_i(\sigma_i, q) \).  

\( N_1 \)

\( \perp(p_1, q) \)

\( \perp(p_2, q) \)

\( q \)

\( p \)

\( p_1 \)

\( p_2 \)

\( \perp(p_2, q) \)

\( q \)

\( p_2 \)

\( p_3 \)

\( \perp(p, q) \)

\( \perp(p_1, q) \)

\( \perp(p_2, q) \)

\( \perp(p_3, q) \)
kTPL Algorithm

• Same filtering as TPL
• Same refining with the following exceptions:
  – A point can be pruned if $k$ points are found within distance $dist(p,q)$ from $p$
  – A counter is associated with each point (initialized to $k$) and decreases when such a point is found
  – A candidate is eliminated if counter $= 0$
  – No prior knowledge of number of points in a node, so no application of $\min\max dist(p, N) < dist(p, q)$ in pruning
  – A point $p$ can be pruned if a node $N$ is found such that $\max dist(p, N) < dist(p, q)$ and $\min\_card(N) \geq \text{counter}(p)$
Experiments

- RNN queries on real data
Conclusion

• TPL is good in that it
  – Supports arbitrary values of k
    • KM
  – Can deal efficiently with database updates
    • KM
  – Is applicable to data of dimensionality more than two
    • SAA
  – Retrieves exact results
    • SFT
  – Results in fast results!
References

2. A presentation by Jalal Kazemitabar in csci587 Fall’2010