Branch and Bound

Algorithms for Nearest Neighbor Search: Lecture 1

Yury Lifshits
http://yury.name

Steklov Institute of Mathematics at St.Petersburg
California Institute of Technology

RuSSIR
Russian Summer School in Information Retrieval
2007

Outline

1. Welcome to Nearest Neighbors!

Outline

1. Welcome to Nearest Neighbors!
2. Branch and Bound Methodology

Outline

1. Welcome to Nearest Neighbors!
2. Branch and Bound Methodology
3. Around Vantage-Point Trees

Outline

1. Welcome to Nearest Neighbors!
2. Branch and Bound Methodology
3. Around Vantage-Point Trees
4. Generalized Hyperplane Trees and Relatives

Outline

1. Welcome to Nearest Neighbors!
2. Branch and Bound Methodology
3. Around Vantage-Point Trees
4. Generalized Hyperplane Trees and Relatives
5. M-Trees

Chapter I

Welcome to Nearest Neighbors!

Informal Statement

To preprocess a database of n objects
so that given a query object,
one can effectively determine
its nearest neighbors in database

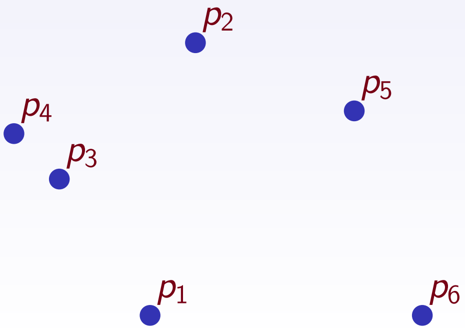
More Formally

Search space: object domain $\mathbb{U}$, similarity function σ

Input: database $S = \{p_1, \ldots, p_n\} \subseteq \mathbb{U}$

Query: $q \in \mathbb{U}$

Task: find $\operatorname{argmax}_{p_i} \sigma(p_i, q)$



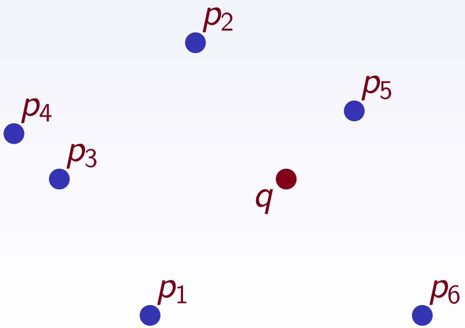
More Formally

Search space: object domain $\mathbb{U}$, similarity function σ

Input: database $S = \{p_1, \ldots, p_n\} \subseteq \mathbb{U}$

Query: $q \in \mathbb{U}$

Task: find $\operatorname{argmax}_{p_i} \sigma(p_i, q)$



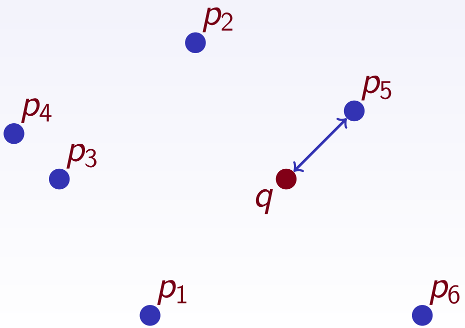
More Formally

Search space: object domain $\mathbb{U}$, similarity function σ

Input: database $S = \{p_1, \ldots, p_n\} \subseteq \mathbb{U}$

Query: $q \in \mathbb{U}$

Task: find $\operatorname{argmax}_{p_i} \sigma(p_i, q)$



Applications (1/5) Information Retrieval

- Content-based retrieval (magnetic resonance images, tomography, CAD shapes, time series, texts)
- Spelling correction
- Geographic databases (post-office problem)
- Searching for similar DNA sequences
- Related pages web search
- Semantic search, concept matching

Applications (2/5) Machine Learning

- kNN classification rule: classify by majority of k nearest training examples. E.g. recognition of faces, fingerprints, speaker identity, optical characters
- Nearest-neighbor interpolation

Applications (3/5) Data Mining

- Near-duplicate detection
- Plagiarism detection
- Computing co-occurrence similarity (for detecting synonyms, query extension, machine translation...)

Applications (3/5) Data Mining

- Near-duplicate detection
- Plagiarism detection
- Computing co-occurrence similarity (for detecting synonyms, query extension, machine translation...)

Key difference:

Mostly, off-line problems

Applications (4/5) Bipartite Problems

- Recommendation systems (most relevant movie to a set of already watched ones)
- Personalized news aggregation (most relevant news articles to a given user's profile of interests)
- Behavioral targeting (most relevant ad for displaying to a given user)

Applications (4/5) Bipartite Problems

- Recommendation systems (most relevant movie to a set of already watched ones)
- Personalized news aggregation (most relevant news articles to a given user's profile of interests)
- Behavioral targeting (most relevant ad for displaying to a given user)

Key differences:

Query and database objects have different nature
Objects are described by features and connections

Applications (5/5) As a Subroutine

- Coding theory (maximum likelihood decoding)
- MPEG compression (searching for similar fragments in already compressed part)
- Clustering

Variations of the Computation Task

Solution aspects:

- Approximate nearest neighbors
- Dynamic nearest neighbors: moving objects, deletes/inserts, changing similarity function

Variations of the Computation Task

Solution aspects:

- Approximate nearest neighbors
- Dynamic nearest neighbors: moving objects, deletes/inserts, changing similarity function

Related problems:

- Nearest neighbor: nearest museum to my hotel
- Reverse nearest neighbor: all museums for which my hotel is the nearest one
- Range queries: all museums up to 2km from my hotel
- Closest pair: closest pair of museum and hotel
- Spatial join: pairs of hotels and museums which are at most 1km apart
- Multiple nearest neighbors: nearest museums for each of these hotels
- Metric facility location: how to build hotels to minimize the sum of "museum — nearest hotel" distances

Brief History

1908 Voronoi diagram

Brief History

1908 Voronoi diagram

1967 kNN classification rule by Cover and Hart

Brief History

1908 Voronoi diagram

1967 kNN classification rule by Cover and Hart

1973 Post-office problem posed by Knuth

Brief History

1908 Voronoi diagram

1967 kNN classification rule by Cover and Hart

1973 Post-office problem posed by Knuth

1997 The paper by Kleinberg, beginning of provable upper/lower bounds

Brief History

- **1908** Voronoi diagram
- **1967** kNN classification rule by Cover and Hart
- **1973** Post-office problem posed by Knuth
- **1997** The paper by Kleinberg, beginning of provable upper/lower bounds
- **2006** Similarity Search book by Zezula, Amato, Dohnal and Batko

Brief History

- **1908** Voronoi diagram
- **1967** kNN classification rule by Cover and Hart
- **1973** Post-office problem posed by Knuth
- **1997** The paper by Kleinberg, beginning of provable upper/lower bounds
- **2006** Similarity Search book by Zezula, Amato, Dohnal and Batko
- **2008** First International Workshop on Similarity Search. Consider submitting!

Tutorial Outline

Four lectures:

1. Branch-and-bound: various tree-based data structures for general metric space

Tutorial Outline

Four lectures:

1. Branch-and-bound: various tree-based data structures for general metric space
2. Other use of triangle inequality: Walks, matrix methods, specific tricks for Euclidean space

Tutorial Outline

Four lectures:

1. Branch-and-bound: various tree-based data structures for general metric space
2. Other use of triangle inequality: Walks, matrix methods, specific tricks for Euclidean space
3. Mapping-based techniques: Locality-sensitive hashing, random projections

Tutorial Outline

Four lectures:

1. Branch-and-bound: various tree-based data structures for general metric space
2. Other use of triangle inequality: Walks, matrix methods, specific tricks for Euclidean space
3. Mapping-based techniques: Locality-sensitive hashing, random projections
4. Restrictions on input: Intrinsic dimension, probabilistic analysis and open problems

Tutorial Outline

Four lectures:

1. Branch-and-bound: various tree-based data structures for general metric space
2. Other use of triangle inequality: Walks, matrix methods, specific tricks for Euclidean space
3. Mapping-based techniques: Locality-sensitive hashing, random projections
4. Restrictions on input: Intrinsic dimension, probabilistic analysis and open problems

Not covered: low-dimensional solutions, experimental results, parallelization, I/O complexity, lower bounds, applications

Chapter II

Branch and Bound Methodology

General Metric Space

Tell me definition of metric space

General Metric Space

Tell me definition of metric space

$M = (\mathbb{U}, d)$, distance function d satisfies:

Non negativity: $\forall s, t \in \mathbb{U} : \quad d(s,t) \geq 0$
Symmetry: $\forall s, t \in \mathbb{U} : \quad d(s,t) = d(t,s)$
Identity: $d(s,t) = 0 \Rightarrow s = t$
Triangle inequality: $\forall r, s, t \in \mathbb{U} : \quad d(r,t) \leq d(r,s) + d(s,t)$

General Metric Space

Tell me definition of metric space

$M = (\mathbb{U}, d)$, distance function d satisfies:

Non negativity: $\forall s, t \in \mathbb{U}: \quad d(s,t) \geq 0$
Symmetry: $\forall s, t \in \mathbb{U}: \quad d(s,t) = d(t,s)$
Identity: $d(s,t) = 0 \Rightarrow s = t$
Triangle inequality: $\forall r, s, t \in \mathbb{U}: \quad d(r,t) \leq d(r,s) + d(s,t)$

Basic Examples:

- Arbitrary metric space, oracle access to distance function
- k-dimensional Euclidean space with Euclidean, weighted Euclidean, Manhattan or L_p metric
- Strings with Hamming or Levenshtein distance

Metric Spaces: More Examples

- Finite sets with Jaccard metric $d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$
- Correlated dimensions: $\bar{x} \cdot M \cdot \bar{y}$ distance
- Hausdorff distance for sets

Metric Spaces: More Examples

- Finite sets with Jaccard metric $d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$
- Correlated dimensions: $\bar{x} \cdot M \cdot \bar{y}$ distance
- Hausdorff distance for sets

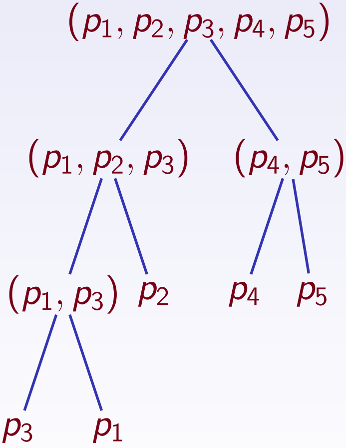
Similarity spaces (no triangle inequality):

- Multidimensional vectors with scalar product similarity
- Bipartite graph, co-citations similarity for vertices in one part
- Social networks with “number of joint friends” similarity

Branch and Bound: Search Hierarchy

Database $S = \{p_1, \ldots, p_n\}$ is represented by a tree:

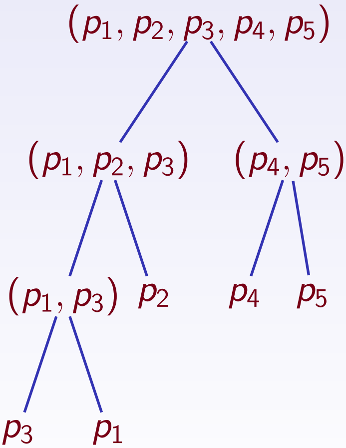
- Every node corresponds to a subset of S
- Root corresponds to S itself
- Children's sets cover parent's set
- Every node contains a "description" of its subtree providing easy-computable lower bound for $d(q, \cdot)$ in the corresponding subset



Branch and Bound: Range Search

Task: find all $i \quad d(p_i, q) \leq r$:

1. Make a depth-first traversal of search hierarchy
2. At every node compute the lower bound for its subtree
3. Prune branches with lower bounds above r



B&B: Nearest Neighbor Search

Task: find $\operatorname{argmin}_{p_i} d(p_i, q)$:

1. Pick a random p_i, set $p_{NN} := p_i, r_{NN} := d(p_i, q)$
2. Start range search with r_{NN} range
3. Whenever meet p' such that $d(p', q) < r_{NN}$, update $p_{NN} := p', r_{NN} := d(p', q)$

B&B: Best Bin First

Task: find $\text{argmin}_{p_i}\, d(p_i, q)$:

1. Pick a random p_i, set $p_{NN} := p_i, r_{NN} := d(p_i, q)$
2. Put the root node into **inspection queue**
3. Every time: take the node with a smallest lower bound from inspection queue, compute lower bounds for children subtrees
4. Insert children with lower bound below r_{NN} into inspection queue; prune other children branches
5. Whenever meet p' such that $d(p', q) < r_{NN}$, update $p_{NN} := p', r_{NN} := d(p', q)$

Some Tree-Based Data Structures

Sphere Rectangle Tree k-d-B tree
Geometric near-neighbor access tree Excluded
middle vantage point forest mvp-tree Fixed-height fixed-queries
tree Vantage-point tree
R*-tree Burkhard-Keller tree BBD tree Voronoi tree Balanced
aspect ratio tree Metric tree vp^s-tree M-tree
SS-tree R-tree Spatial approximation tree Multi-vantage
point tree Bisector tree mb-tree
Generalized hyperplane tree
Hybrid tree Slim tree Spill Tree Fixed queries tree X-tree
k-d tree Balltree Quadtree Octree
SR-tree Post-office tree

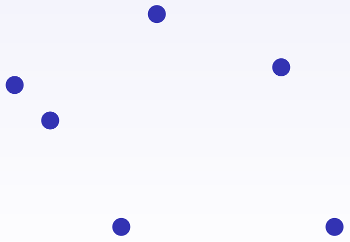
Chapter III

Vantage-Point Trees and Relatives

Vantage-Point Partitioning

Uhlmann'91, Yianilos'93:

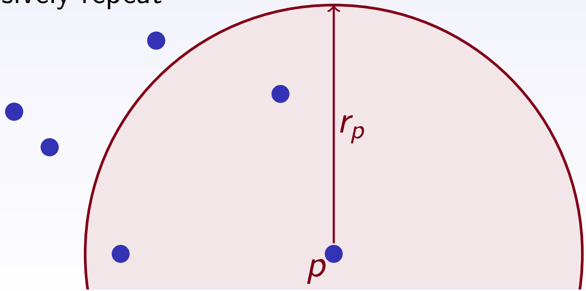
1. Choose some object p in database (called pivot)
2. Choose partitioning radius r_p
3. Put all p_i such that $d(p_i, p) \leq r$ into "inner" part, others to the "outer" part
4. Recursively repeat



Vantage-Point Partitioning

Uhlmann'91, Yianilos'93:

1. Choose some object p in database (called pivot)
2. Choose partitioning radius r_p
3. Put all p_i such that $d(p_i, p) \leq r$ into "inner" part, others to the "outer" part
4. Recursively repeat



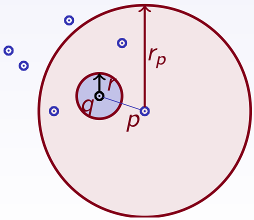
Pruning Conditions

For r-range search:

If $d(q,p) > r_p + r$ prune the inner branch

If $d(q,p) < r_p - r$ prune the outer branch

For $r_p - r \leq d(q,p) \leq r_p + r$ we have to inspect both branches



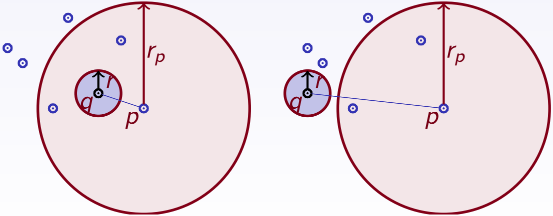
Pruning Conditions

For r-range search:

If $d(q,p) > r_p + r$ prune the inner branch

If $d(q,p) < r_p - r$ prune the outer branch

For $r_p - r \leq d(q,p) \leq r_p + r$ we have to inspect both branches



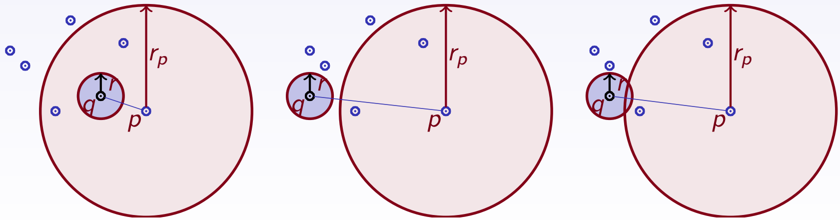
Pruning Conditions

For r-range search:

If $d(q, p) > r_p + r$ prune the inner branch

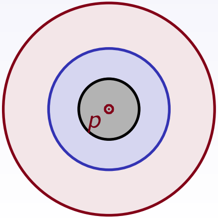
If $d(q, p) < r_p - r$ prune the outer branch

For $r_p - r \leq d(q, p) \leq r_p + r$ we have to inspect both branches



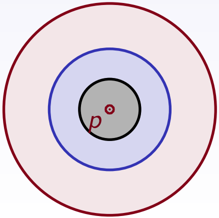
Variations of Vantage-Point Trees

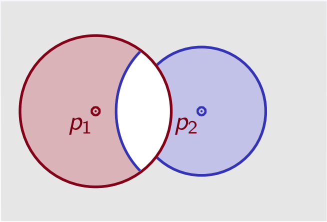
- **Burkhard-Keller tree:** pivot used to divide the space into m rings Burkhard&Keller'73
- **MVP-tree:** use the same pivot for different nodes in one level Bozkaya&Ozsoyoglu'97
- **Post-office tree:** use $r_p + \delta$ for inner branch, $r_p - \delta$ for outer branch McNutt'72



Variations of Vantage-Point Trees

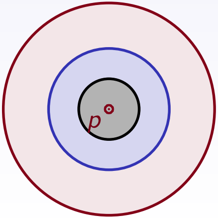
- **Burkhard-Keller tree:** pivot used to divide the space into m rings   Burkhard&Keller'73
- **MVP-tree:** use the same pivot for different nodes in one level   Bozkaya&Ozsoyoglu'97
- **Post-office tree:** use $r_p + \delta$ for inner branch, $r_p - \delta$ for outer branch   McNutt'72

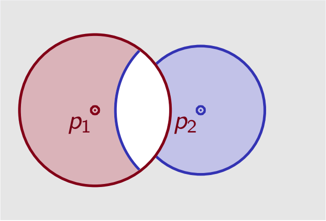


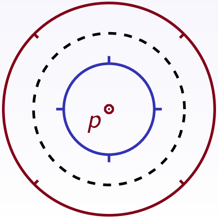


Variations of Vantage-Point Trees

- **Burkhard-Keller tree:** pivot used to divide the space into m rings Burkhard&Keller'73
- **MVP-tree:** use the same pivot for different nodes in one level Bozkaya&Ozsoyoglu'97
- **Post-office tree:** use $r_p + \delta$ for inner branch, $r_p - \delta$ for outer branch McNutt'72







Chapter IV

Generalized Hyperplane Trees and Relatives

Generalized Hyperplane Tree

Partitioning technique (Uhlmann'91):

- Pick two objects (called pivots) p_1 and p_2
- Put all objects that are closer to p_1 than to p_2 to the left branch, others to the right branch
- Recursively repeat

Generalized Hyperplane Tree

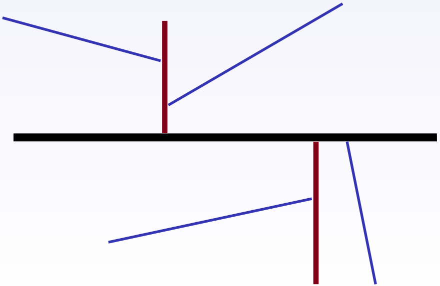
Partitioning technique (Uhlmann'91):

- Pick two objects (called pivots) p_1 and p_2
- Put all objects that are closer to p_1 than to p_2 to the left branch, others to the right branch
- Recursively repeat

Generalized Hyperplane Tree

Partitioning technique (Uhlmann'91):

- Pick two objects (called pivots) p_1 and p_2
- Put all objects that are closer to p_1 than to p_2 to the left branch, others to the right branch
- Recursively repeat

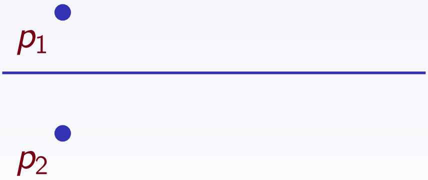


GH-Tree: Pruning Conditions

For r-range search:

If $d(q, p_1) > d(q, p_2) + 2r$ prune the left branch
If $d(q, p_1) < d(q, p_2) - 2r$ prune the right branch

For $|d(q, p_1) - d(q, p_2)| \leq 2r$ we have to inspect both branches

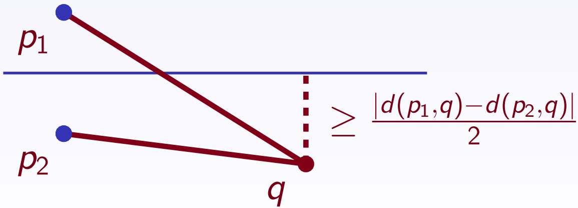


GH-Tree: Pruning Conditions

For *r*-range search:

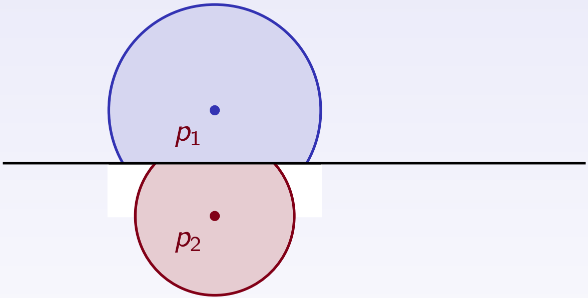
If $d(q, p_1) > d(q, p_2) + 2r$ prune the left branch
If $d(q, p_1) < d(q, p_2) - 2r$ prune the right branch

For $|d(q, p_1) - d(q, p_2)| \leq 2r$ we have to inspect both branches



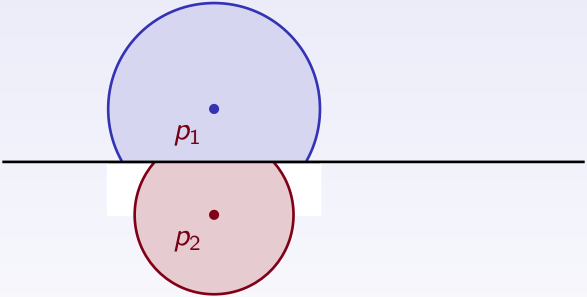
Bisector trees

Let's keep the covering radius for p_1 and left branch, for p_2 and right branch: useful information for stronger pruning conditions



Bisector trees

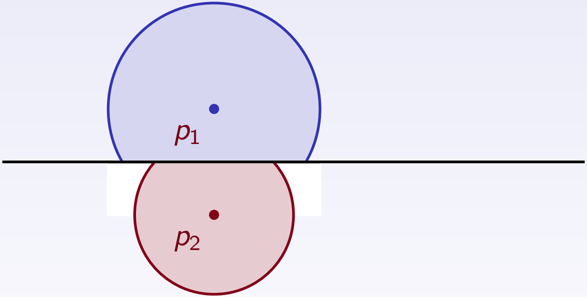
Let's keep the covering radius for p_1 and left branch, for p_2 and right branch: useful information for stronger pruning conditions



Variation: monotonous bisector tree (Noltemeier, Verbarg, Zirkelbach'92) always uses parent pivot as one of two children pivots

Bisector trees

Let's keep the covering radius for p_1 and left branch, for p_2 and right branch: useful information for stronger pruning conditions



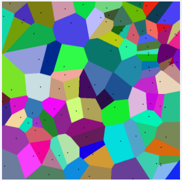
Variation: monotonous bisector tree (Noltemeier, Verbarg, Zirkelbach'92) always uses parent pivot as one of two children pivots

Exercise: prove that covering radii are monotonically decrease in mb-trees

Geometric Near-Neighbor Access Tree

Brin'95:

- Use m pivots
- Branch i consists of objects for which p_i is the closest pivot
- Stores minimal and maximal distances from pivots to all "brother"-branches



Chapter V

M-trees

M-tree: Data structure

Ciaccia, Patella, Zezula'97:

- All database objects are stored in leaf nodes (buckets of fixed size)
- Every internal nodes has associated pivot, covering radius and legal range for number of children (e.g. 2-3)
- Usual depth-first or best-first search

M-tree: Data structure

Ciaccia, Patella, Zezula'97:

- All database objects are stored in leaf nodes (buckets of fixed size)
- Every internal nodes has associated pivot, covering radius and legal range for number of children (e.g. 2-3)
- Usual depth-first or best-first search

Special algorithms for insertions and deletions a-la B-tree

M-tree: Insertions

All insertions happen at the leaf nodes:

1. Choose the leaf node using "minimal expansion of covering radius" principle

M-tree: Insertions

All insertions happen at the leaf nodes:

1. Choose the leaf node using “minimal expansion of covering radius” principle
2. If the leaf node contains fewer than the maximum legal number of elements, there is room for one more. Insert; update all covering radii

M-tree: Insertions

All insertions happen at the leaf nodes:

1. Choose the leaf node using "minimal expansion of covering radius" principle
2. If the leaf node contains fewer than the maximum legal number of elements, there is room for one more. Insert; update all covering radii
3. Otherwise the leaf node is split into two nodes

M-tree: Insertions

All insertions happen at the leaf nodes:

1. Choose the leaf node using “minimal expansion of covering radius” principle
2. If the leaf node contains fewer than the maximum legal number of elements, there is room for one more. Insert; update all covering radii
3. Otherwise the leaf node is split into two nodes
    1. Use two pivots generalized hyperplane partitioning

M-tree: Insertions

All insertions happen at the leaf nodes:

1. Choose the leaf node using "minimal expansion of covering radius" principle
2. If the leaf node contains fewer than the maximum legal number of elements, there is room for one more. Insert; update all covering radii
3. Otherwise the leaf node is split into two nodes
    1. Use two pivots generalized hyperplane partitioning
    2. Both pivots are added to the node's parent, which may cause it to be split, and so on

Exercises

Prove that Jaccard distance $d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$ satisfies triangle inequality

Exercises

Prove that Jaccard distance $d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$ satisfies triangle inequality

Prove that covering radii are monotonically decrease in mb-trees

Exercises

Prove that Jaccard distance $d(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$ satisfies triangle inequality

Prove that covering radii are monotonically decrease in mb-trees

Construct a database and a set of potential queries in some multidimensional Euclidean space for which all described data structures require $\Omega(n)$ nearest neighbor search time

Highlights

- Nearest neighbor search is fundamental for information retrieval, data mining, machine learning and recommendation systems

Highlights

- Nearest neighbor search is fundamental for information retrieval, data mining, machine learning and recommendation systems
- Balls, generalized hyperplanes and Voronoi cells are used for space partitioning

Highlights

- Nearest neighbor search is fundamental for information retrieval, data mining, machine learning and recommendation systems
- Balls, generalized hyperplanes and Voronoi cells are used for space partitioning
- Depth-first and Best-first strategies are used for search

Highlights

- Nearest neighbor search is fundamental for information retrieval, data mining, machine learning and recommendation systems
- Balls, generalized hyperplanes and Voronoi cells are used for space partitioning
- Depth-first and Best-first strategies are used for search

Thanks for your attention! Questions?

References

Course homepage http://simsearch.yury.name/tutorial.html

Y. Lifshits
The Homepage of Nearest Neighbors and Similarity Search
http://simsearch.yury.name

P. Zezula, G. Amato, V. Dohnal, M. Batko
Similarity Search: The Metric Space Approach. Springer, 2006.
http://www.nmis.isti.cnr.it/amato/similarity-search-book/

E. Chávez, G. Navarro, R. Baeza-Yates, J. L. Marroquín
Searching in Metric Spaces. ACM Computing Surveys, 2001.
http://www.cs.ust.hk/~leichen/courses/comp630j/readings/acm-survey/searchinmetric.pdf

G.R. Hjaltason, H. Samet
Index-driven similarity search in metric spaces. ACM Transactions on Database Systems, 2003
http://www.cs.utexas.edu/~abhinay/ee382v/Project/Papers/ft_gateway.cfm.pdf