Restrictions on Input

Algorithms for Nearest Neighbor Search: Lecture 4

Yury Lifshits
http://yury.name

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

RuSSIR
Russian Summer School in Information Retrieval 2007

Making Nearest Neighbors Easier

Tractable solution: $poly(n)$ preprocessing, $poly \log(n)$ search time

General case of nearest neighbors seems to be intractable

Making Nearest Neighbors Easier

Tractable solution: $poly(n)$ preprocessing, $poly \log(n)$ search time

General case of nearest neighbors seems to be intractable

Any assumption that makes the problem easier?

Making Nearest Neighbors Easier

Tractable solution: *poly*(n) preprocessing, *poly* $\log(n)$ search time

General case of nearest neighbors seems to be intractable

Any assumption that makes the problem easier?

Two approaches:

- Define intrinsic dimension of search domain and assume it is small (usually constant or $\mathcal{O}(\log\log n)$)
- Fix some probability distribution over inputs and queries. Find an algorithm which is fast with high probability over inputs/query

Outline

1. Nearest Neighbors in Small Doubling Dimension

Outline

1. Nearest Neighbors in Small Doubling Dimension
2. Disorder Method: A Combinatorial Solution of Nearest Neighbors

Outline

1. Nearest Neighbors in Small Doubling Dimension
2. Disorder Method: A Combinatorial Solution of Nearest Neighbors
3. Probabilistic Analysis: Zipf Model

Outline

1. Nearest Neighbors in Small Doubling Dimension
2. Disorder Method: A Combinatorial Solution of Nearest Neighbors
3. Probabilistic Analysis: Zipf Model
4. Open Problems

Chapter XI

Nearest Neighbors in Small Doubling Dimension

Mini-plan:

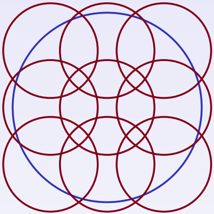
Notion of doubling dimension

Solving 3-approximate nearest neighbors

From 3-approximation to $(1 + \varepsilon)$-approximation

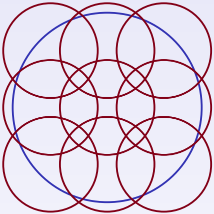
Notion of Doubling Dimension

Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r



Notion of Doubling Dimension

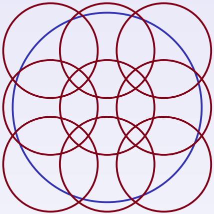
Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r



Doubling dimension: logarithm of doubling constant $\dim(\mathbb{U}) = \log \lambda$

Notion of Doubling Dimension

Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r

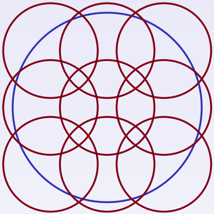


Doubling dimension: logarithm of doubling constant
$\dim(\mathbb{U}) = \log \lambda$

Exercise: Prove that for Euclidean space $\dim(\mathbb{R}^d) = \mathcal{O}(d)$

Notion of Doubling Dimension

Doubling constant λ for search domain $\mathbb{U}$: minimal value such that for every r and every object $p \in \mathbb{U}$ the ball $B(p, 2r)$ has cover of at most λ balls of radius r



Doubling dimension: logarithm of doubling constant
$\dim(\mathbb{U}) = \log \lambda$

Exercise: Prove that for Euclidean space $\dim(\mathbb{R}^d) = \mathcal{O}(d)$

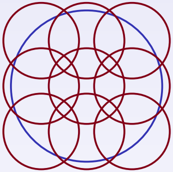
Exercise: Prove that $\forall S \subset \mathbb{U}: \quad \dim(S) \leq 2\dim(\mathbb{U})$

Doubling Dimension and r-Nets

Set $T \subset \mathbb{U}$ is an r-net for $S \subset \mathbb{U}$ iff

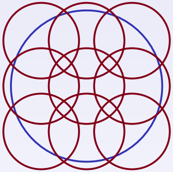
(1) $\forall p, p' \in T : \; d(p, p') > r$

(2) $\forall s \in S \;\; \exists p \in T : \; d(s, p) < r$



Doubling Dimension and r-Nets

Set $T \subset \mathbb{U}$ is an r-net for $S \subset \mathbb{U}$ iff
(1) $\forall p, p' \in T: \; d(p, p') > r$
(2) $\forall s \in S \;\; \exists p \in T: \; d(s, p) < r$



Lemma (Cover Lemma)

Every ball $B(p, r)$ has δr-net of cardinality at most $(\frac{1}{\delta})^{\mathcal{O}(dim(\mathbb{U}))}$

Cover Lemma: Proof

Greedy algorithm:

1. Start from empty T
2. Find some object in S which is still δr-far from all objects in T, add it to T
3. Stop when all objects in S are within δr from some point in T

Cover Lemma: Proof

Greedy algorithm:

1. Start from empty T
2. Find some object in S which is still δr-far from all objects in T, add it to T
3. Stop when all objects in S are within δr from some point in T

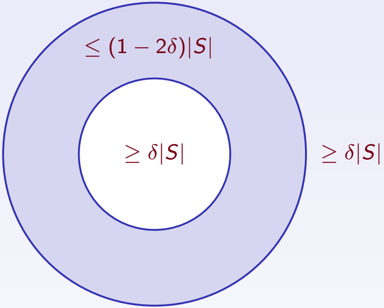
Upper bound on size:

- Apply definition of doubling constant to $B(p, r)$ recursively until getting $\frac{\delta r}{3}$-cover
- This cover has size $(\frac{1}{\delta})^{\mathcal{O}(\dim(\mathbb{U}))}$
- Every element of this cover can contain at most one object from T

Ring-Separator Lemma

Triple $(p, r, 2r)$ is δ-**ring-separator** for S iff

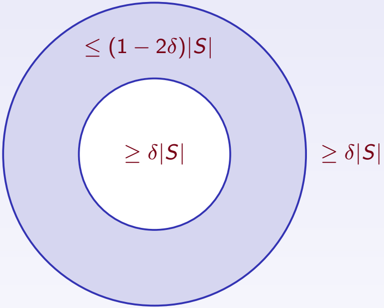
1. $|S \cap B(p, r)| \geq \delta |S|$
2. $|S / B(p, 2r)| \geq \delta |S|$



Ring-Separator Lemma

Triple $(p, r, 2r)$ is
δ-ring-separator
for S iff

1. $|S \cap B(p, r)| \geq \delta|S|$
2. $|S / B(p, 2r)| \geq \delta|S|$



Lemma (Ring-Separator Lemma)

For every S there is ring-separator with $\delta \geq (\frac{1}{2})^{\mathcal{O}(dim(S))}$

Ring-Separator Lemma: Proof

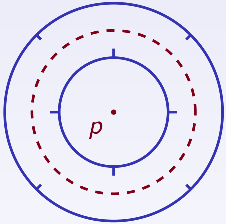
- Fix $\delta = (\frac{1}{2})^{c\dim(S)}$ for some large c
- For every p choose the maximal r_p such that $|B(p, r_p)| < \delta|S|$
- Let p_0 be the one having minimal r_{p_0}
- If none of triples $(p, r_p, 2r_p)$ is δ ring-separator build an r_{p_0}-net for $B(p_0, 2r_{p_0})$:
  - Start from r_0, and set $A := B(p_0, 2r_{p_0})/B(p_0, r_{p_0})$
  - Iteratively add some point p from A to net, update $A := A/B(p, r)$
- Since A decreased by at most $2\delta|S|$ points each time there must be many points in cover. Since it is r_{p_0}-net for $B(p_0, 2r_{p_0})$ there must be few points. Contradiction

Ring-Separator Tree

Krauthgamer&Lee'05

Preprocessing:

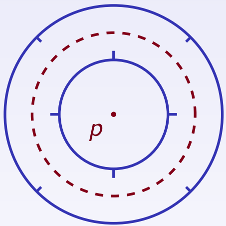
1. Find $(\frac{1}{2})^{\mathcal{O}(\dim(S))}$ ring-separator $(p, r, 2r)$ for S
2. Put objects from $B(p, 2r)$ to inner branch
3. Put objects from $S/B(p, r)$ to outer branch
4. Recursively repeat



Ring-Separator Tree

Krauthgamer&Lee'05

Preprocessing:



1. Find $(\frac{1}{2})^{\mathcal{O}(\dim(S))}$ ring-separator $(p, r, 2r)$ for S
2. Put objects from $B(p, 2r)$ to inner branch
3. Put objects from $S/B(p, r)$ to outer branch
4. Recursively repeat

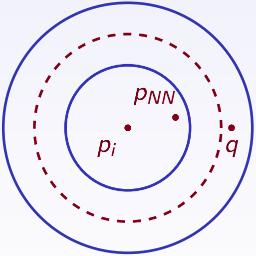
Search:

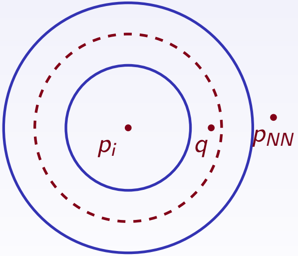
1. For every node $(p, r, 2r)$: if $d(q, p) \leq 3r/2$ go only to inner branch otherwise go only to outer branch
2. Return the best object considered in search

3-NN via Ring-Separator Tree

Notation: $p_1, \ldots, p_k$ are the centers of visited rings

- If $p_{NN}(q) = p_k$ we are done
- If not, let us consider p_i where we miss the right branch. There are two cases:

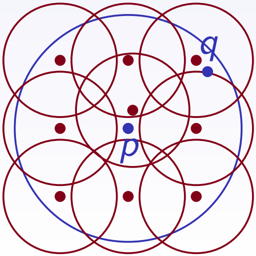




- Anyway, p_i at most 3 time worse than $p_{NN}(q)$

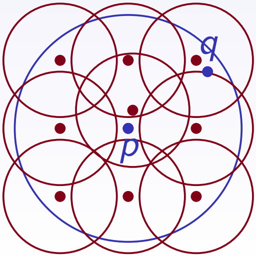
From 3-NN to r-NN: Reduction Algorithm

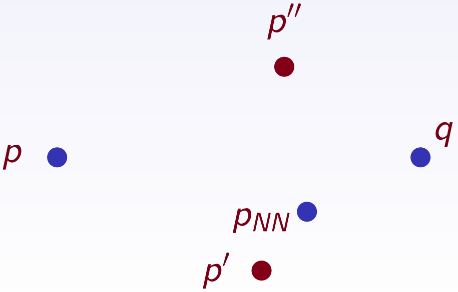
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q



From 3-NN to r-NN: Reduction Algorithm

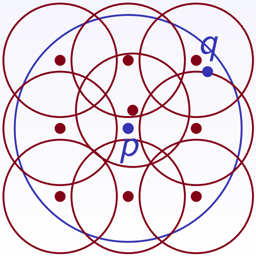
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

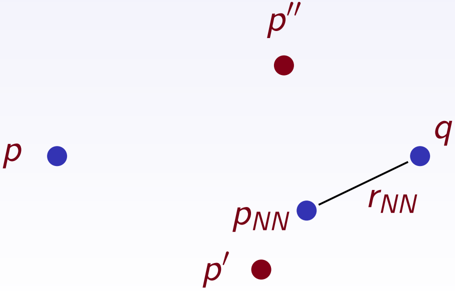




From 3-NN to r-NN: Reduction Algorithm

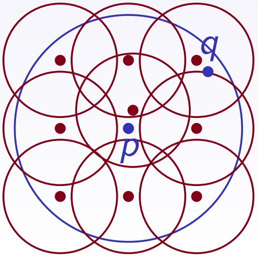
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

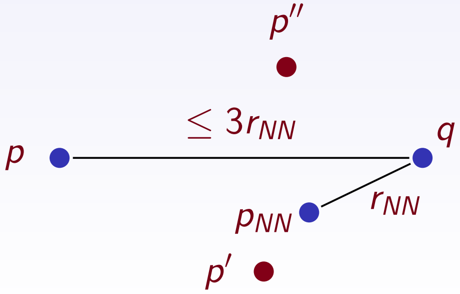




From 3-NN to r-NN: Reduction Algorithm

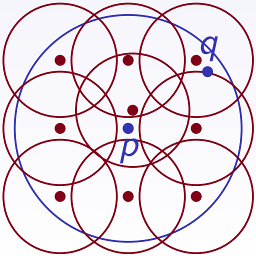
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

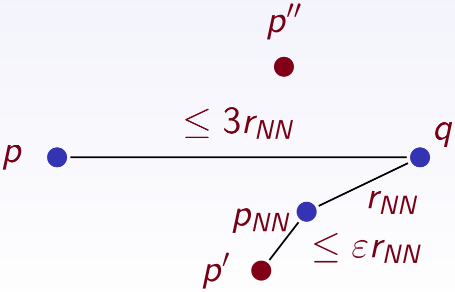




From 3-NN to r-NN: Reduction Algorithm

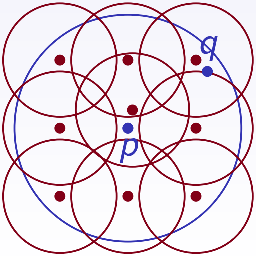
1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q

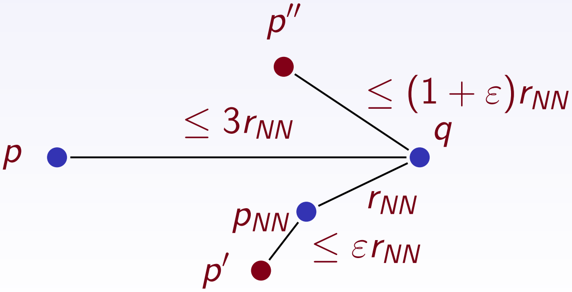




From 3-NN to r-NN: Reduction Algorithm

1. Find 3-approximate nearest neighbor p for q
2. Quickly build a $\varepsilon\frac{d(p,q)}{3}$ cover for $B(p, 4\frac{d(p,q)}{3})$. See the next slide
3. Return an object in cover that is the closest to q





From 3-NN to r-NN: Net Construction

Preprocessing:

1. For every i build 2^i-net for S (every lower level contains all points from the higher level)
2. Compute children pointers: from every element p of 2^i-net to all balls of 2^{i-1}-net required to cover $B(p, 2^i)$
3. Compute brother pointers: from every element p of 2^i-net to all elements p' from 2^i-net needed for covering $B(p, 2^i)$
4. Compute parent pointers: from every element p of 2^{i-1}-net to the element p' from 2^i-net within 2^i from it

From 3-NN to r-NN: Net Construction

Preprocessing:

1. For every i build 2^i-net for S (every lower level contains all points from the higher level)
2. Compute children pointers: from every element p of 2^i-net to all balls of 2^{i-1}-net required to cover $B(p, 2^i)$
3. Compute brother pointers: from every element p of 2^i-net to all elements p' from 2^i-net needed for covering $B(p, 2^i)$
4. Compute parent pointers: from every element p of 2^{i-1}-net to the element p' from 2^i-net within 2^i from it

On-line net construction:

1. Go up by parent pointers until meeting ball big enough
2. Use brother pointer
3. Go by children pointers until getting cover small enough

Other Definitions of Intrinsic Dimension

- Box dimension is the minimal d that for every r our domain $\mathbb{U}$ has r-net of size at most $(1/r)^{d+o(1)}$
- Karger-Ruhl dimension of database $S \subset \mathbb{U}$ is the minimal d that for every $p \in S$ and every r the following inequality holds:
  $|B(p, 2r) \cap S| \leq 2^d |B(p, r) \cap S|$
- Measure-based dimensions
- Disorder dimension (see next chapter)

Other Definitions of Intrinsic Dimension

- Box dimension is the minimal d that for every r our domain $\mathbb{U}$ has r-net of size at most $(1/r)^{d+o(1)}$
- Karger-Ruhl dimension of database $S \subset \mathbb{U}$ is the minimal d that for every $p \in S$ and every r the following inequality holds:
  $|B(p, 2r) \cap S| \leq 2^d |B(p, r) \cap S|$
- Measure-based dimensions
- Disorder dimension (see next chapter)

Exercise: prove that

$$\forall S \subset \mathbb{U} : \quad \dim_{\mathrm{Doub}}(S) \leq 4 \dim_{\mathrm{KR}}(S)$$

Chapter XII

Disorder Method:
A Combinatorial Solution of Nearest Neighbors

Concept of Disorder

Sort all objects in database S by their similarity to p
Let $\mathrm{rank}_p(s)$ be position of object s in this list

Concept of Disorder

Sort all objects in database S by their similarity to p
Let $\mathrm{rank}_p(s)$ be position of object s in this list

Disorder inequality for some constant D:

$$\forall p, r, s \in \{q\} \cup S: \qquad \mathrm{rank}_r(s) \leq D \cdot (\mathrm{rank}_p(r) + \mathrm{rank}_p(s))$$

Minimal D providing disorder inequality is called **disorder constant** of a given set

Concept of Disorder

Sort all objects in database S by their similarity to p
Let $\text{rank}_p(s)$ be position of object s in this list

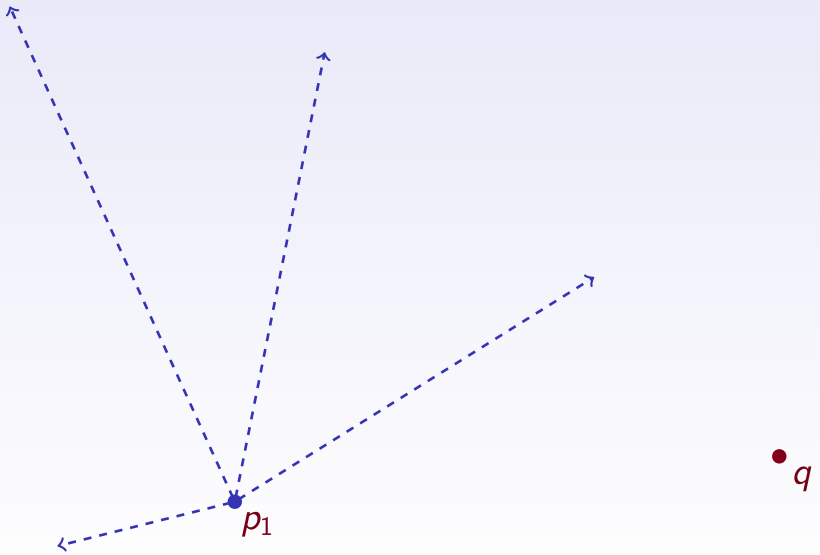
Disorder inequality for some constant D:

$$\forall p, r, s \in \{q\} \cup S : \qquad \text{rank}_r(s) \leq D \cdot (\text{rank}_p(r) + \text{rank}_p(s))$$

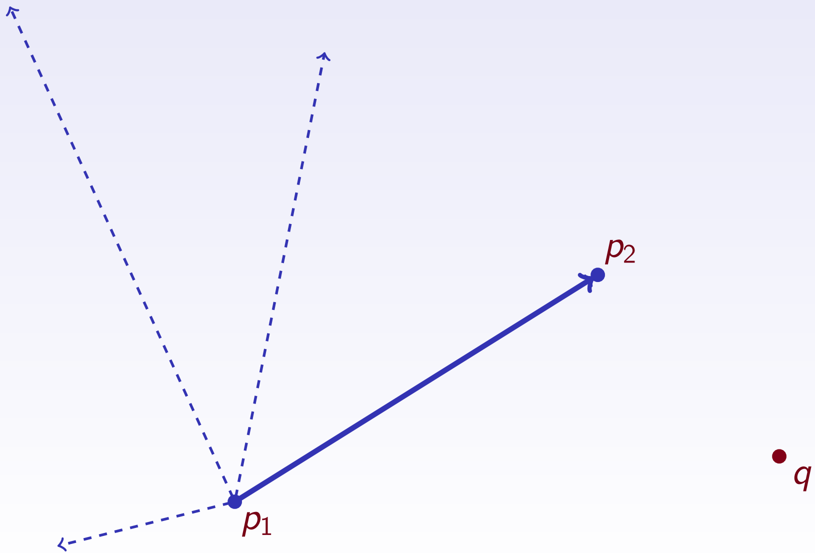
Minimal D providing disorder inequality is called **disorder constant** of a given set

For "regular" sets in d-dimensional Euclidean space $D \approx 2^{d-1}$

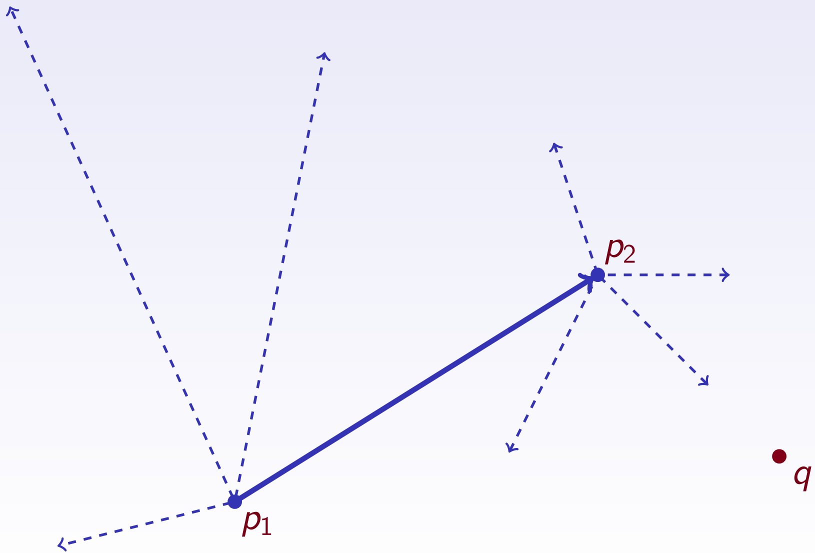
Ranwalk Informally (1/2)



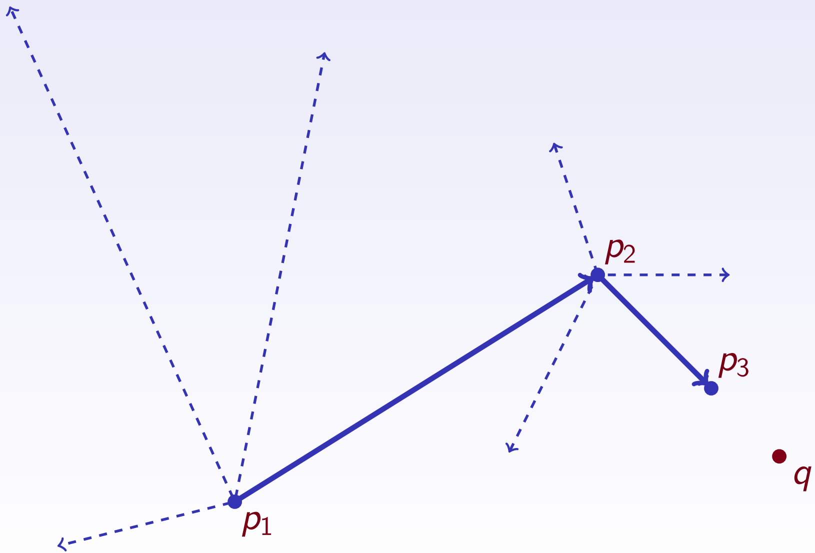
Ranwalk Informally (1/2)



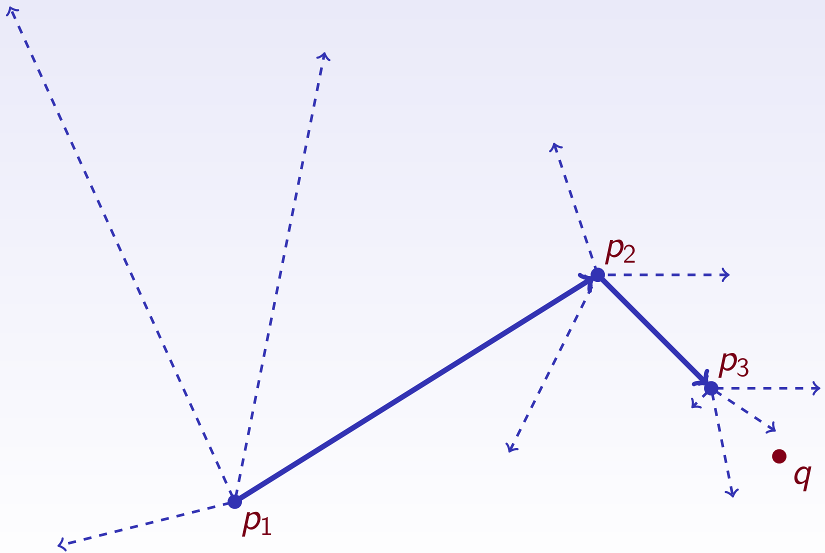
Ranwalk Informally (1/2)



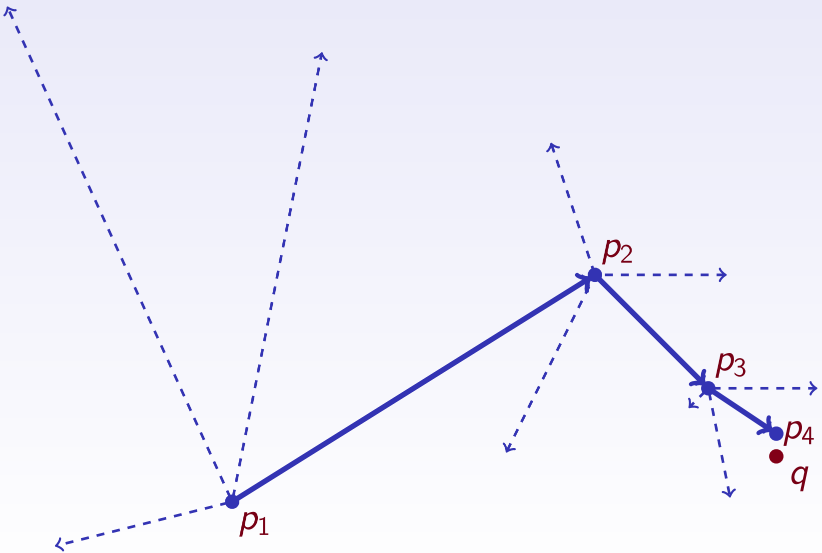
Ranwalk Informally (1/2)



Ranwalk Informally (1/2)



Ranwalk Informally (1/2)



Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)
4. Among all bus routes from p_3 choose the one going most closely to q, move there (p_4)

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)
4. Among all bus routes from p_3 choose the one going most closely to q, move there (p_4)
5. Repeat this $\log n$ times and return the final city

Ranwalk Informally (2/2)

Hierarchical greedy navigation:

1. Start at random city p_1
2. Among all airlines choose the one going most closely to q, move there (say, to p_2)
3. Among all railway routes from p_2 choose the one going most closely to q, move there (p_3)
4. Among all bus routes from p_3 choose the one going most closely to q, move there (p_4)
5. Repeat this $\log n$ times and return the final city

Transport system: for level k choose c random arcs to $\frac{n}{2^k}$ neighborhood

Ranwalk Algorithm

Preprocessing:

- For every point p in database we sort all other points by their similarity to p

  Data structure: n lists of $n-1$ points each.

Query processing:

1. Step 0: choose a random point p_0 in the database.
2. From $k=1$ to $k=\log n$ do Step k: Choose $D' := 3D(\log\log n + 1)$ random points from $\min(n, \frac{3Dn}{2^k})$-neighborhood of p_{k-1}. Compute similarities of these points w.r.t. q and set p_k to be the most similar one.
3. If $\mathrm{rank}_{p_{\log n}}(q) > D$ go to step 0, otherwise search the whole D^2-neighborhood of $p_{\log n}$ and return the point most similar to q as the final answer.

Analysis of Ranwalk

Theorem (Goyal, YL, Schütze. 2007)

Assume that database points together with query point $S \cup \{q\}$ *satisfy disorder inequality with constant* D*:*

$$rank_x(y) \leq D(rank_z(x) + rank_z(y)).$$

Then Ranwalk algorithm always answers nearest neighbor queries correctly. It uses the following resources:
Preprocessing space: $\mathcal{O}(n^2)$.
Preprocessing time: $\mathcal{O}(n^2 \log n)$.
Expected query time: $\mathcal{O}(D \log n \log \log n + D^2)$.

Arwalk Algorithm

Preprocessing:

- For every point p in database we sort all other points by their similarity to p. For every *level number* k from 1 to $\log n$ we store pointers to $D' = 3D(\log\log n + \log 1/\delta)$ random points within $\min(n, \frac{3Dn}{2^k})$ most similar to p points.

Query processing:

1. Step 0: choose a random point p_0 in the database.
2. From $k = 1$ to $k = \log n$ do Step k: go by p_{k-1} pointers of level k. Compute similarities of these D' points to q and set p_k to be the most similar one.
3. Return $p_{\log n}$.

Analysis of Algorithm

Theorem (Goyal, YL, Schütze. 2007)

Assume that database points together with query point $S \cup \{q\}$ *satisfy disorder inequality with constant* D*:*

$$rank_x(y) \leq D(rank_z(x) + rank_z(y)).$$

Then for any probability of error δ *Arwalk algorithm answers nearest neighbor query within the following constraints:*
Preprocessing space: $\mathcal{O}(nD \log n(\log \log n + \log 1/\delta))$.
Preprocessing time: $\mathcal{O}(n^2 \log n)$.
Query time: $\mathcal{O}(D \log n(\log \log n + \log 1/\delta))$.

Chapter XIII

Probabilistic Analysis: Zipf Model

Probabilistic Analysis in a Nutshell

- We define a probability distribution over databases

Probabilistic Analysis in a Nutshell

- We define a probability distribution over databases
- We define probability distribution over query objects

Probabilistic Analysis in a Nutshell

- We define a probability distribution over databases
- We define probability distribution over query objects
- We construct a solution that is efficient/accurate with high probability over "random" input/query

Zipf Model

- Terms $t_1, \ldots, t_m$
- To generate a document we take every t_i with probability $\frac{1}{i}$
- Database is n independently chosen documents
- Query document has exactly one term in every interval $[e^i, e^{i+1}]$
- Similarity between documents is defined as the number of common terms

Magic Level Theorem

Magic Level $q = \sqrt{2 \log_e n}$

Theorem (Hoffmann, YL, Nowotka. CSR'07)

1. *With very high probability there exists a document in database having* $q - \varepsilon$ **top** *terms of query document*

2. *With very small probability there exists a document in database having* **any** $q + \varepsilon$ *overlap with query document*

Chapter XIV

Future of Nearest Neighbors

Directions for Further Research (1/2)

- Relational nearest neighbors: using graph structure of underlying domain. Examples: co-occurrence similarity, recommendations via social network
- Nearest neighbors for sparse vectors in Euclidean space
- Low-distortion embeddings for social networks, similarity visualization
- Construct algorithms for learning similarity function

Directions for Further Research (2/2)

- Probabilistic analysis for specific domains: introduce reasonable input distributions and solve nearest neighbors for them
- Disorder method / Intrinsic dimension: fast algorithm for bounded average dimension
- Branch and bound techniques for bichromatic nearest neighbors
- New dynamic aspects: object descriptions are changing in time

OP1: NN for Sparse Vectors

Database: n vectors in $\mathbb{R}^m$ each having at most $k \ll m$ nonzero coordinates

Query: vector in $\mathbb{R}^m$ also having at most $k \ll m$ nonzero coordinates

Similarity: scalar product

OP1: NN for Sparse Vectors

Database: n vectors in $\mathbb{R}^m$ each having at most $k \ll m$ nonzero coordinates

Query: vector in $\mathbb{R}^m$ also having at most $k \ll m$ nonzero coordinates

Similarity: scalar product

Construct an algorithm for solving nearest neighbors on sparse vectors

Constraints: $poly(n, m)$ preprocessing, $poly(k, \log n)$ query

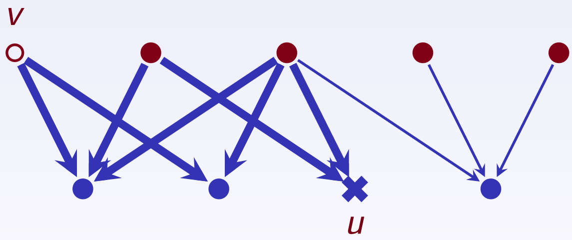
OP2: 3-Step NN

3-step similarity between boy and girl in some bipartite boys-girls graph is equal to number of paths of length 3 between them

n boys

boy degrees $\leq k$

m girls



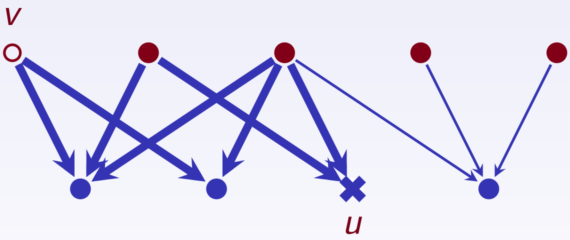
OP2: 3-Step NN

3-step similarity between boy and girl in some bipartite boys-girls graph is equal to number of paths of length 3 between them

n boys

boy degrees $\leq k$

m girls



Construct an algorithm for solving nearest neighbors in bipartite graphs with 3-step similarity

Constraints: $poly(n, m)$ preprocessing, $poly(k, \log n, \log m)$ query

Exercises

Prove that $\dim_{\mathrm{Doub}}(\mathbb{R}^d) = \mathcal{O}(d)$

Exercises

Prove that $\dim_{\text{Doub}}(\mathbb{R}^d) = \mathcal{O}(d)$

Prove that $\forall S \subset \mathbb{U} : \quad \dim_{\text{Doub}}(S) \leq 2\dim_{\text{Doub}}(\mathbb{U})$

Exercises

Prove that $\dim_{\mathrm{Doub}}(\mathbb{R}^d) = \mathcal{O}(d)$

Prove that $\forall S \subset \mathbb{U} : \quad \dim_{\mathrm{Doub}}(S) \leq 2\dim_{\mathrm{Doub}}(\mathbb{U})$

Prove that $\forall S \subset \mathbb{U} : \quad \dim_{\mathrm{Doub}}(S) \leq 4\dim_{\mathrm{KR}}(S)$

Highlights

- Doubling dimension: restriction on size of r-covers.
  Solution: ring-separator tree

Highlights

- Doubling dimension: restriction on size of r-covers.
  Solution: ring-separator tree
- Disorder inequality: replacement of triangle inequality using rank values instead of actual similarity values

Highlights

- Doubling dimension: restriction on size of r-covers. Solution: ring-separator tree
- Disorder inequality: replacement of triangle inequality using rank values instead of actual similarity values
- Probabilistic analysis: Efficient algorithm for texts generated by Zipf model

Highlights

- Doubling dimension: restriction on size of r-covers. Solution: ring-separator tree
- Disorder inequality: replacement of triangle inequality using rank values instead of actual similarity values
- Probabilistic analysis: Efficient algorithm for texts generated by Zipf model

Thanks for your attention! Questions?

References (1/2)

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

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

N. Goyal, Y. Lifshits, H. Schütze
Disorder Inequality: A Combinatorial Approach to Nearest Neighbors Submitted
http://yury.name/papers/goyal2008disorder.pdf

B. Hoffmann, Y. Lifshits, D. Nowotka
Maximal Intersection Queries in Randomized Graph Models CSR'07
http://yury.name/papers/hoffmann2007maximal.pdf

References (2/2)

R. Krauthgamer and J.R. Lee
The black-box complexity of nearest-neighbor search Theoretical Computer Science, 2005
http://www.cs.berkeley.edu/~jrl/papers/nnc.pdf

R. Krauthgamer and J.R. Lee
Navigating nets: simple algorithms for proximity search SODA'04
http://www.cs.berkeley.edu/~robi/papers/KL-NavNets-SODA04.pdf

K.L. Clarkson
Nearest-Neighbor Searching and Metric Space Dimensions
In Nearest-Neighbor Methods for Learning and Vision: Theory and Practice, MIT Press, 2006
http://www.cs.bell-labs.com/who/clarkson/nn_survey/p.pdf