Algorithms for Nearest Neighbors: Theoretical Aspects

Yury Lifshits

Steklov Institute of Mathematics at St.Petersburg

Kolmogorov Seminar, April 2007

Outline

1. Problem Statement
   - Applications
   - Three Data Models

Outline

1. Problem Statement
   - Applications
   - Three Data Models
2. Three Relaxed Versions of Nearest Neighbors
   - Super-Nearest Neighbors
   - Approximate Nearest Neighbors
   - Nearest Rare Neighbors

Outline

1. Problem Statement
   - Applications
   - Three Data Models
2. Three Relaxed Versions of Nearest Neighbors
   - Super-Nearest Neighbors
   - Approximate Nearest Neighbors
   - Nearest Rare Neighbors
3. Nearest Neighbors in Zipf Model

Outline

Part I

What are nearest neighbors about?

Industrial applications

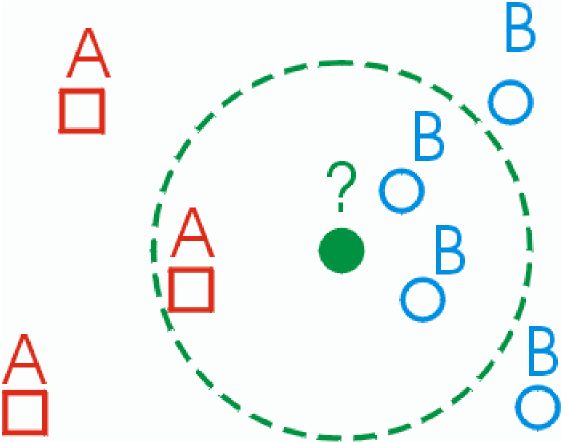
Three data models

Informal Problem Statement

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

First Application (1960s)

Nearest neighbors for classification:



Picture from `http://cgm.cs.mcgill.ca/ soss/cs644/projects/perrier/Image25.gif`

Applications

What applications of nearest neighbors do you know?

Applications

What applications of nearest neighbors do you know?

- Statistical data analysis, e.g. medicine diagnosis
- Pattern recognition, e.g. for handwriting
- Code plagiarism detection
- Coding theory
- Future applications: recommendation systems, ads distribution, personalized news aggregation

Data Model in General

Formalization for nearest neighbors consists of:

- Representation format for objects
- Similarity function

Vector Model

Database: points in R^d

Similarity: scalar product

Vector Model

Database: points in R^d

Similarity: scalar product

Constraints:
$poly(n+d)$ for preprocessing time,
$d \cdot polylog(n+d)$ for query

Sparse Vector Model

Database: points in R^d,
every point has at most $k \ll d$ nonzero coordinates

Similarity: scalar product

Sparse Vector Model

Database: points in R^d,
every point has at most $k \ll d$ nonzero coordinates

Similarity: scalar product

Constraints:
$poly(n + d)$ for preprocessing time,
$poly(k) \cdot polylog(n + d)$ for query

Set Model

Database: n subsets of T, having size at most k

$|T| = m$

Similarity: size of intersection

Set Model

Database: n subsets of T, having size at most k
$|T| = m$

Similarity: size of intersection

Constraints:
$poly(n+m)$ for preprocessing time,
$poly(k) \cdot polylog(n+m)$ for query

Set Model

Database: n subsets of T, having size at most k
$|T| = m$

Similarity: size of intersection

Constraints:
$poly(n+m)$ for preprocessing time,
$poly(k) \cdot polylog(n+m)$ for query

More data models?

Part II

Three Relaxed Versions of Nearest Neighbors

Super-Nearest Neighbors

Idea

We will search for nearest neighbors only within $B(q, \tau)$

Definition

p is nearest τ-neighbor for q iff $d(p, q) \leq \tau$ and p is in fact the nearest neighbor for q

Yianilos Theorem

Consider some **nice** metric space $\mathcal{S}$ and probability distribution P over it

Yianilos Theorem

Consider some **nice** metric space $\mathcal{S}$ and probability distribution P over it

Theorem (Nearest τ-Neighbors)

For any fixed database $DB \subset \mathcal{S}$ of size n and for any $M > 1$ there exists $\tau > 0$ such that we can construct a binary tree for DB which answers nearest τ-neighbor queries using at most $M \cdot (\log n + 1)$ expected metric evaluations

Approximate Nearest Neighbors

Definition

p is ε-approximate nearest neighbor for q
iff $\forall p' \in DB: \qquad d(p,q) \leq (1+\varepsilon)d(p',q)$

VP-Trees for Approximate NN

Partitioning condition: $d(p, x) \;<?\; r$

Inner branch: $B(p, r(1+\delta))$, where $\delta = \frac{1}{1+\varepsilon}$

Outer branch: $R^d / B(p, r(1-\delta))$

VP-Trees for Approximate NN

Partitioning condition: $d(p, x) \ <? \ r$

Inner branch: $B(p, r(1 + \delta))$, where $\delta = \frac{1}{1+\varepsilon}$

Outer branch: $R^d / B(p, r(1 - \delta))$

Search:

If $d(p, q) < r$ go to inner branch

If $d(p, q) > r$ go to outer branch

VP-Trees for Approximate NN

Partitioning condition: $d(p,x) \ <? \ r$
Inner branch: $B(p, r(1+\delta))$, where $\delta = \frac{1}{1+\varepsilon}$
Outer branch: $R^d / B(p, r(1-\delta))$

Search:
If $d(p,q) < r$ go to inner branch
If $d(p,q) > r$ go to outer branch and return minimum between obtained result and $d(p,q)$

Rare Neighbors

Definition

p is an r-rare neighbor for q
iff p and q have common nonzero coordinate
which is nonzero for at most r points in DB

Cheating

We will search only for neighbors that have at least one common rare feature with query object

Rare-Point Method

Preprocessing:

For every rare feature store a list of all objects in database having it

Rare-Point Method

Preprocessing:

For every rare feature store a list
of all objects in database having it

Query processing:

Retrieve all point that have at least one
common rare feature with the query object;
Perform linear scan on them

Part III
Probabilistic Analysis

Probabilistic assumptions about data collection can lead to provably efficient solutions for nearest neighbors

This section represents joint work with Benjamin Hoffmann and Dirk Nowotka

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 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

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*

Part IV
Further Work

Directions for Research

Three Specific Open Problems

Directions for Further Research

- Develop techniques for proving hardness of some computational problems with preprocessing. Find theoretical limits for some specific families of algorithms

Directions for Further Research

- Develop techniques for proving hardness of some computational problems with preprocessing. Find theoretical limits for some specific families of algorithms
- Extend classical NN algorithms to new data models and new task variations

Directions for Further Research

- Develop techniques for proving hardness of some computational problems with preprocessing. Find theoretical limits for some specific families of algorithms
- Extend classical NN algorithms to new data models and new task variations
- Develop theoretical analysis of existing heuristics. Average case complexity is particulary promising. Find subcases for which we can construct provably efficient solutions

Directions for Further Research

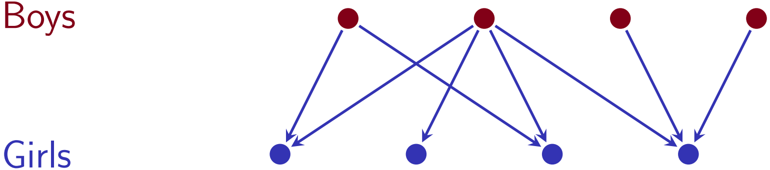
- Develop techniques for proving hardness of some computational problems with preprocessing. Find theoretical limits for some specific families of algorithms
- Extend classical NN algorithms to new data models and new task variations
- Develop theoretical analysis of existing heuristics. Average case complexity is particulary promising. Find subcases for which we can construct provably efficient solutions
- Compare NN-based approach with other methods for classification/recognition/prediction problems

OP1: 3-Step NN

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

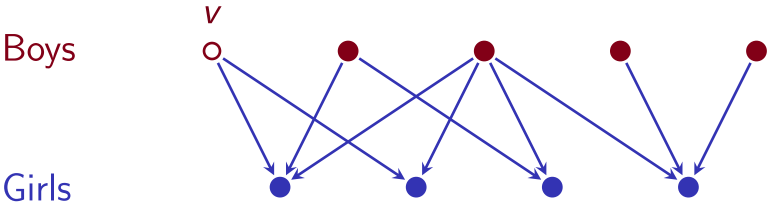
OP1: 3-Step NN

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



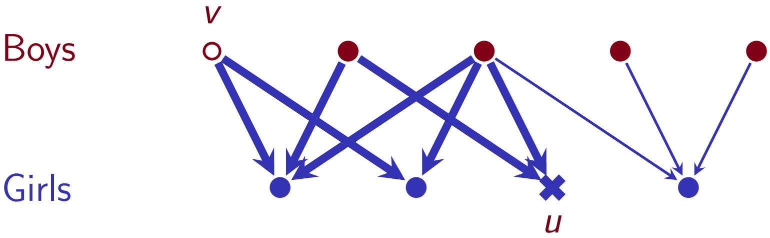
OP1: 3-Step NN

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



OP1: 3-Step NN

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



OP2: 1D Dynamic NN

Input

Database of n points in one-dimensional space and their velocity vectors

Query task

To find the nearest neighbor for a given query point at a given time point

Constraints

Data storage after preprocessing $n \cdot polylog(n)$

Time for query processing $polylog(n)$

OP3: Inclusions with Preprocessing

Input

Family $\mathcal{F}$ of subsets of T

Query task

Given a set $f_{new} \subseteq T$ to decide
whether $\exists f \in \mathcal{F} : \quad f_{new} \subseteq f$

Constraints

Data storage after preprocessing $poly(|\mathcal{F}| + |T|)$
Time for query processing $poly(|T|)$

OP3: Inclusions with Preprocessing

Input

Family $\mathcal{F}$ of subsets of T

Query task

Given a set $f_{new} \subseteq T$ to decide whether $\exists f \in \mathcal{F} : \quad f_{new} \subseteq f$

Constraints

Data storage after preprocessing $poly(|\mathcal{F}| + |T|)$
Time for query processing $poly(|T|)$

Conjecture: this problem CAN NOT be solved within such time/space constraints

Call for Feedback

- Any new ideas how to solve nearest neighbors?
- What kind of formalization should we consider?
- Any relevant work?
- How to improve this talk for the next time?

Summary

- Nearest neighbors is one of the key algorithmic problems for web technologies
- Key ideas: relax search to approximately nearest neighbor, nearest r-rare neighbor or nearest neighbor in τ-neighborhood of query point
- Further work: theoretical analysis of heuristics, extending known solutions to new data models, lower bounds

Summary

- Nearest neighbors is one of the key algorithmic problems for web technologies
- Key ideas: relax search to approximately nearest neighbor, nearest r-rare neighbor or nearest neighbor in τ-neighborhood of query point
- Further work: theoretical analysis of heuristics, extending known solutions to new data models, lower bounds

Thanks for your attention! Questions?

References (1/2)

Contact: http://logic.pdmi.ras.ru/~yura

B. Hoffmann, Y. Lifshits and D. Nowotka

Maximal Intersection Queries in Randomized Graph Models

http://logic.pdmi.ras.ru/~yura/en/maxint-draft.pdf

P.N. Yianilos

Data structures and algorithms for nearest neighbor search in general metric spaces

http://www.pnylab.com/pny/papers/vptree/vptree.ps

J. Zobel and A. Moffat

Inverted files for text search engines

http://www.cs.mu.oz.au/~alistair/abstracts/zm06compsurv.html

K. Teknomo

Links to nearest neighbors implementations

http://people.revoledu.com/kardi/tutorial/KNN/resources.html

References (2/2)

J. Kleinberg

Two Algorithms for Nearest-Neighbor Search in High Dimensions

http://www.ece.tuc.gr/~vsam/csalgo/kleinberg-stoc97-nn.ps

P. Indyk and R. Motwani

Approximate nearest neighbors: towards removing the curse of dimensionality

http://theory.csail.mit.edu/~indyk/nndraft.ps

A. Andoni and P. Indyk

Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions

http://theory.lcs.mit.edu/~indyk/FOCS06final.ps

P. Indyk

Nearest Neighbors Bibliography

http://theory.lcs.mit.edu/~indyk/bib.html