Algorithms for Nearest Neighbors

Background and Two Challenges

Yury Lifshits
Steklov Institute of Mathematics at St.Petersburg
http://logic.pdmi.ras.ru/~yura

McGill University, July 2007

Outline

Outline

Outline

1. Formulating the Problem
2. Nearest Neighbors for Texts
3. Proving Hardness of Nearest Neighbors

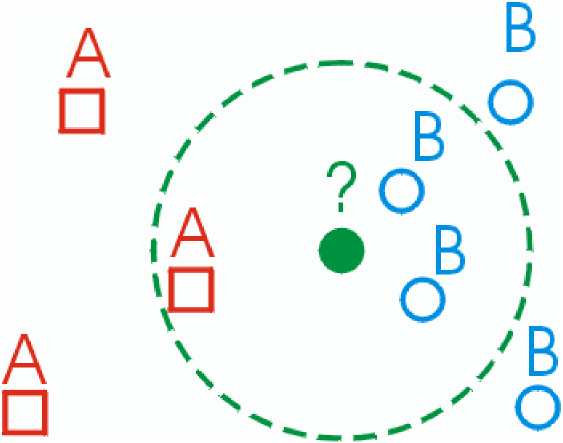
Part I
Formulating the Problem

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?

- Text classification
- Statistical data analysis, e.g. medicine diagnosis
- Pattern recognition: characters, faces
- Code plagiarism detection
- Coding theory
- Data compression

Applications

What applications of nearest neighbors do you know?

- Text classification
- Statistical data analysis, e.g. medicine diagnosis
- Pattern recognition: characters, faces
- Code plagiarism detection
- Coding theory
- Data compression
- **Web:** recommendation systems, on-line ads, personalized news aggregation, long queries in web search, near-duplicates detection

Data Model in General

Formalization for nearest neighbors consists of:

- Representation format for objects
- Similarity function

Basic Data Models (1/2)

- Vector Model
  - Similarity: l^2, scalar product, cosine

Basic Data Models (1/2)

- Vector Model
  - Similarity: l^2, scalar product, cosine
- String Model
  - Similarity: Hamming distance, edit distance

Basic Data Models (1/2)

- Vector Model
  - Similarity: l^2, scalar product, cosine
- String Model
  - Similarity: Hamming distance, edit distance
- Black-box model
  - Similarity: given by oracle
    The only knowledge is triangle inequality

Basic Data Models (2/2)

- Set Model
  - Similarity: size of intersection

Basic Data Models (2/2)

- Set Model
  - Similarity: size of intersection
- Small graphs
  - Similarity: structure/labels matching

Algorithmic Approaches to NN

- Divide and conquer
- Traversal techniques
- Look-up techniques
- Contractive and low-distortion embeddings
- Tournament algorithms

Part II
Nearest Neighbors for Texts

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

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

Open Problem: solve NN for sparse vector model within given constraints

Inverted Index

Preprocessing:

For very term store a list of all documents in database with nonzero weight on it

Query processing:

Retrieve all point that have at least one common term with the query documet; Perform linear scan on them

Rare-Point Method

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

Rare-Point Method

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

Preprocessing:

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

Rare-Point Method

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

Preprocessing:
For very 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

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, Lifshits and 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*

Part III
Proving Hardness of Nearest Neighbors

Inclusions with Preprocessing (1/2)

Input

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

Query task

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

Constraints

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

Inclusions with Preprocessing (1/2)

Input

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

Query task

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

Constraints

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

Open problem: is there an algorithm satisfying given constraints?

Inclusions with Preprocessing (2/2)

Reformulation in SAT style:

Input
DNF formula $\mathcal{F}$ on n variables, without negations

Query task
Given an assignment x to evaluate $\mathcal{F}(x)$

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

Inclusions with Preprocessing (2/2)

Reformulation in SAT style:

Input

DNF formula $\mathcal{F}$ on n variables, without negations

Query task

Given an assignment x to evaluate $\mathcal{F}(x)$

Constraints

Data storage after preprocessing $poly(|\mathcal{F}|)$

Time for query processing $poly(n)$

Open problem: is there an algorithm satisfying given constraints?

“NP Analogue” for Search Problems

Every problem in **SEARCH class** is characterized by poly-time computable Turing Machine M:

Input

Strings $x_1, \ldots, x_n$, $\qquad |x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable problems in SEARCH

Input

Strings $x_1, \ldots, x_n$, $|x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable problems in SEARCH

Input

Strings $x_1, \dots, x_n$, $\qquad |x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable solution

Preprocessing in $poly(m, n)$ space

Query processing in $poly(m, \log n)$ time with RAM access to preprocessed database

Tractable problems in SEARCH

Input

Strings $x_1, \ldots, x_n$, $|x_i| = m$

Query task

Given string y of length m to answer whether $\exists i : M(x_i, y) = yes$

Tractable solution

Preprocessing in $poly(m, n)$ space

Query processing in $poly(m, \log n)$ time with RAM access to preprocessed database

Inclusions is in SEARCH. Is it tractable?

Complete problems in SEARCH (1/2)

Program Search problem:

Input

Turing machines $P_1 \ldots, P_n$

Query task

Given string y of length m to answer
whether $\exists i : P_i(y) = yes$ after at most m steps

Complete problems in SEARCH (1/2)

Program Search problem:

Input

Turing machines $P_1 \ldots, P_n$

Query task

Given string y of length m to answer
whether $\exists i : P_i(y) = yes$ after at most m steps

Open problem: is Program Search tractable?

Complete problems in SEARCH (2/2)

Parallel Run problem:

Input

$x_1 \ldots, x_n$

Query task

Given poly-time computable P to answer whether $\exists i : P(x_i) = yes$

Complete problems in SEARCH (2/2)

Parallel Run problem:

Input

$x_1 \ldots, x_n$

Query task

Given poly-time computable P to answer whether $\exists i : P(x_i) = yes$

Open problem: is Parallel Run tractable?

Conclusions

Call for Feedback

- Any relevant work?
- How to improve this talk for the next time?

Call for Feedback

- Any relevant work?
- How to improve this talk for the next time?
- **Give my open problems to your friends!**

Summary

- Nearest neighbors for texts can be modelled by Euclidean space with sparse vectors

Summary

- Nearest neighbors for texts can be modelled by Euclidean space with sparse vectors
- No exact algorithms for NN in texts are known so far

Summary

- Nearest neighbors for texts can be modelled by Euclidean space with sparse vectors
- No exact algorithms for NN in texts are known so far
- New approach to lower bounds for NN: SEARCH class and its complete problems

Summary

- Nearest neighbors for texts can be modelled by Euclidean space with sparse vectors
- No exact algorithms for NN in texts are known so far
- New approach to lower bounds for NN: SEARCH class and its complete problems

Thanks for your attention! Questions?

References (1/2)

Search "**Lifshits**" or visit 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