Algorithms for Nearest Neighbors

State-of-the-Art

Yury Lifshits

Steklov Institute of Mathematics at St.Petersburg

Yandex Tech Seminar, April 2007

Outline

Part I

What are nearest neighbors about?

Short overview of applications

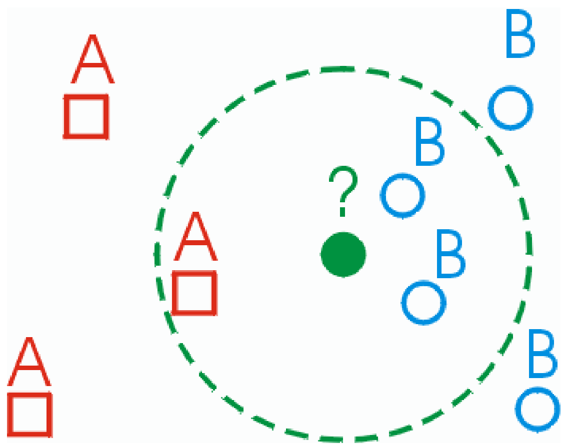
Variations of the problem

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 in Web Technologies

- Text classification (YaCa)
- Personalized news aggregation (Yandex Lenta)
- Recommendation systems (MoiKrug, Yandex Market)
- On-line advertisement distribution systems (Yandex Direct)
- Long queries in web search (Yandex Search)
- Content-similar pages, near-duplicates (Yandex Search)
- Semantic search

Data Model

Formalization for nearest neighbors consists of:

- Representation format for objects
- Similarity function

Remark 1: Usually there is original and "reduced" representation for every object

Remark 2: Accuracy of NN-based classification, prediction or recommendations depends solely on a data model, no matter what specific exact NN algorithm we use.

Variations of Data Model (1/3)

- Vector Model
  - Similarity: scalar product, cosine
- Set Model
  - Similarity: size of intersection
- String Model
  - Similarity: Hamming distance

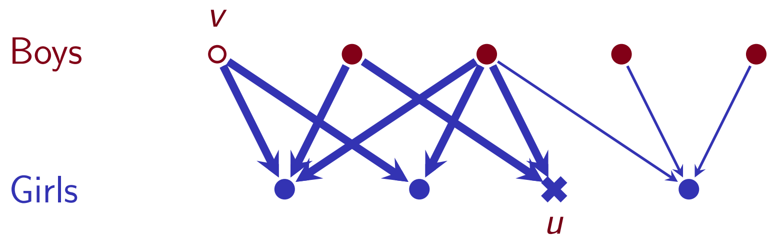
Variations of Data Model (2/3)

- Sparse Vector Model: query time should be in $o(d)$
  - Similarity: scalar product
- Small graphs
  - Similarity: structure/labels matching

More data models?

Variations of Data Model (3/3)

- **New** 3-step bipartite model
  - Similarity: number of 3-step paths



Variations of the Computation Task

- Approximate nearest neighbors
- Multiple nearest neighbors
- Nearest assignment
- All over-threshold neighbor pairs
- Nearest neighbors in dynamically changing database: moving objects, deletes/inserts, changing similarity function

Part II
Overview of Algorithmic Techniques

Partitioning, look-up and embedding-based approaches for **vector model**

New rare-point method (joint work with Hinrich Schütze)

Linear Scan

What is the most obvious solution for nearest neighbors?

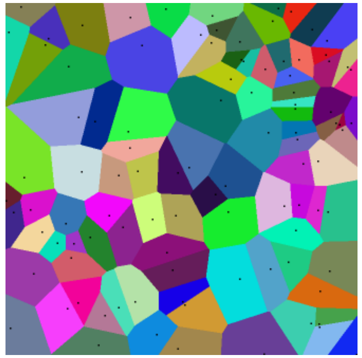
Answer:
compare query object with every object in database

Advantages:
No preprocessing
Exact solution
Works in any data model

Directions for improvement:
order of scanning, pruning

Voronoi diagrams

Voronoi diagram is a mapping transforming every point p in database to the polygon of points for which p is the nearest neighbor



Can we generalize one-dimensional binary search?

KD-Trees

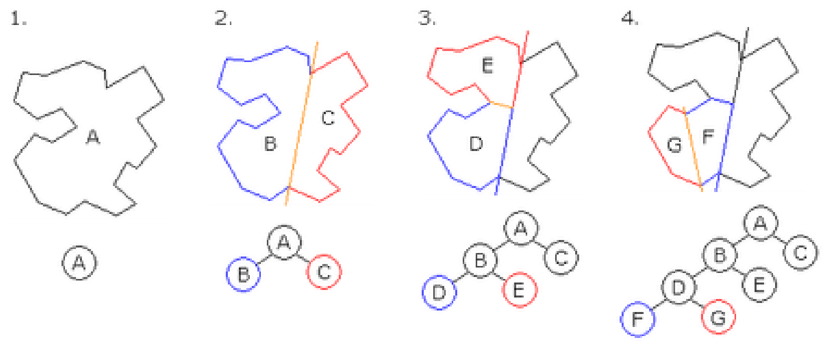
Preprocessing:
Build a kd-tree: for every internal node on level l we make partitioning based on the value of $l \mod d$-th coordinate

Query processing:
Go down to the leaf corresponding to the the query point and compute the distance;

(Recursively) Go one step up, check whether the distance to the second branch is larger than that to current candidate neighbor
if "yes" go up, else check this second branch

BSP-Trees

Generalization: BSP-tree allows to use any hyperplanes in tree construction



VP-Trees

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

Inner branch: $B(p, r(1 + \varepsilon))$

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 $d(p, q)$

Inverted Index

How can we use sparseness of vectors in database?

Preprocessing:

For very coordinate store a list of all points in database with nonzero value on it

Query processing:

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

Locality-Sensitive Hashing

Desired hash family $\mathcal{H}$:

- If $\|p - q\| \leq R$ then $\mathcal{P}r_{\mathcal{H}}[h(p) = h(q)] \geq p_1$
- If $\|p - q\| \geq cR$ then $\mathcal{P}r_{\mathcal{H}}[h(p) = h(q)] \leq p_2$

Preprocessing:

Choose at random several hash functions from $\mathcal{H}$

Build inverted index for hash values of object in database

Query processing:

Retrieve all object that have at least one common hash value with query object; 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

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

Kleinberg Embedding-Based Approach

Preprocessing:
Choose at random linear transformation
$A : R^d \to R^l, \qquad l \ll d$
Apply A to all points in database and make some tricky preprocessing for images

Query processing:
Compute $A(q)$;
Find its nearest neighbor in R^l;

Part IV
Further Work

Directions for Applied Research

Directions for Theoretical Research

Questions to Practitioners

Directions for Applied Research

Attractive goals: **NN-based recommendation system, NN-based ads distribution system**

- Choose reasonable data model, add some assumptions about the nature of database and queries
- Find (theoretically) the best solutions in the resulting formalization
- Perform experimental analysis of obtained solutions
- Develop a prototype product

Directions for Theoretical 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

Questions to Practitioners

- What is your experience in using nearest neighbors? What algorithm/data model are used? Do you face any scalability/accuracy problems? What is a bottleneck subproblem?
- Are you interested to apply NN approach in any of your future products?
- Give us benchmark data
- Give us names and contacts of potentially interested engineers

Summary

- Nearest neighbors is one of the key algorithmic problems for web technologies
- Key ideas: look-up tables, partitioning techniques, embeddings
- Further work: from algorithms to prototype products, from heuristics to theory, from canonical problem to new data models and new search tasks

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