Performance Comparison of Multi-Dimensional Indexing Methods for k-Nearest Neighbor Search

Lijuan Zhang

Advisor: Alexander Thomasian
Introduction

- Similarity search is a popular paradigm in multimedia databases
- Objects represented by N-dimensional feature vectors
- Similarity defined as the (Euclidean) distance of the endpoints of feature vectors
- Similarity search amounts to k-NN (k-Nearest Neighbor) search in a high dimensional vector space
- What’s k-NN Search?
- How to speed up the k-NN Search?
K-NN Search

- Find the k images which are most similar to the following image.
Query Results
K-NN Search

- **Brute force**
  - Sequential scan
    Compute euclidean distance of feature vector of query point wrt all objects in databases, pick k closest points.

- **Speeding up k-NN Search**
  - Clustering
  - Indexing
  - Clustering + Indexing
  - Approximate k-NN Search
Multi-dimensional Indexing

Disk resident
- R-tree based
  • R-tree, R*-tree, R+-tree, SS-tree, SR-tree, X-tree
- kd-tree based
  • K-D-B tree, hB-tree, Hybrid-tree
- Metric-trees
  • VP-tree, M-tree, iDistance

Main memory
- quad-tree, k-d tree, ordered partition tree
R-tree (Guttman’84)

- Extension of B-tree
- Height-balanced
- Data-driven (no nodes for empty space)
- Partition space using MBRs (Minimum Bounding Rectangles)
- Minimizes the area
- Allows overlap
  - Multiple paths may need to be searched
R-tree
R*-tree (Beckmann+ ’90)

- A variant of the R-tree
- Minimizes
  - Overlap
  - Area
  - Perimeter
- Forced reinsert
SR-tree (Katayama’97)

- An extension of R*-tree and SS-tree
- SS-tree partitions space by hyperspheres
- SR-tree partitions space by the intersection of hyperspheres and hyperrectangles

Advantages:
- Partitions neighborhoods to smaller regions
- Improves the disjointness among the regions
SR-tree
Hybrid Tree (Chakrabarti+’99)

- Combines positive aspects of data and space partitioning index structures.
- Nodes represented as modified kd-trees.
  - Splits high-dimensional space like a kd-tree
  - Allows overlap
Ordered Partition Tree (Kim+’86)

- A kd-tree like main memory index
- Partition space in a round robin manner
- For each dimension, specify the number of ways to be partitioned
- For the leaf nodes, specify the capacity
Ordered Partition Tree

\[ X = (1,2,5), (3,8,7), (9,10,8), (12,9,2), (8,7,20), (6,6,23), (0,3,27), (2,13,9), (11,11,15), (14,17,13), (7,4,12), (10,12,3) \]
Our Solution

- Main memory
  - Scalability
  - Disk resident
    - I/O cost
- Combined solution
  Clustering + Indexing + Sequential Disk Access
Index Construction

- Cluster the dataset
- Build index for each cluster
- Save index on disk
  - Transfer the linked representation into a Contiguous Memory Area (CMA)
  - Adjust pointers wrt CMA base address
  - Record addresses of words containing pointers
  - Write the CMA onto disk as a blob
Index File Format

I: internal and leaf nodes, II: a list of positions for the all points, III: a list of pointers from the leaf nodes, IV: the data for all the points.
K-NN Processing

Initializes

Preprocess

Find the primary cluster

KNN search on the primary cluster

Is there any candidate cluster for search?

KNN search on the candidate cluster

Merge the two search results

k nearest neighbors of the query
K-NN Processing

Search a cluster

- Load the index into main memory
  - Read in the index body
  - Read in the address lookup table
  - Adjust relative addresses to absolute addresses
- Search on the main memory index
Comparison of Index Size

SYN64 : 16 clusters : average dimension 13

index size (Mbytes)

R*-tree
SR-tree
OP-tree

Lijuan Zhang @ GSSD
OP-tree Speedup vs R*-tree & SR-tree

SYN64 : 20NN : 16 clusters

Lijuan Zhang @ GSSD
Comparison of Elapsed Time

TXT55 : 32 clusters : 20NN

Elapsed Time (sec)

Dimensions

SR-tree

OP-tree
Comparison of Index Sizes

SYN64 : 8 clusters

Index Size (pages)

Dimensions

R*-tree
Hybrid-tree
SR-tree
OP-tree

Lijuan Zhang @ GSSD
OP-tree Speedup vs R*-tree & SR-tree

SYN64 : 8 clusters : 20NN

- R*-tree
- SR-tree

Lijuan Zhang @ GSSD
Comparison of Estimated Time

SYN64 : 8 clusters : 20NN

- R*-tree
- Hybrid-tree
- SR-tree
- OP-tree

Dimensions

Lijuan Zhang @ GSSD
Conclusions

- OP-tree index read from disk in one sequential scan.
- Other Indexing Structures access individual pages, which ensue in costly random accesses.
- With the advent of giant main memories, the days of paginated indexing structures may be over!
Thanks