Outline

Goal: ‘Find similar / interesting things’

- Intro to DB
- Indexing - similarity search
- Data Mining

Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
- spatial access methods
  - problem dfn
  - z-ordering
  - R-trees
  - misc
- text
- ...

SAMs - Detailed outline

- spatial access methods
  - problem dfn
  - z-ordering
  - R-trees
  - misc topics
    - grid files
    - dimensionality curse; dim. reduction
    - metric trees
    - other nn methods
- text, ...

Grid files

- problem: spatial queries in k-d point-sets
- Main idea: try to generalize hashing to k-d
- (how?)

Grid files

- A: put a grid
- specs: [Nievergelt +, 84]
  - symmetric to all attributes
  - 2 disk accesses for exact match queries
  - adaptive to non-uniform distr.
- Q: details?
Grid files

- cuts: all the way through
- cuts: at ½, ¾, ¼ etc; but on demand
- each cell -> disk page

Thus, we only need:
- cut-points for each axis
- k-d directory

x-cuts
\[ \frac{1}{4}, \frac{1}{2} \]
y-cuts
\[ \frac{1}{2} \]

Search (for exact match) – eg., (0.3; 0.3)

Thus, we only need:
- cut-points for each axis
- k-d directory

x-cuts
\[ \frac{1}{4}, \frac{1}{2} \]
y-cuts
\[ \frac{1}{2} \]

Grid files

- specs: [Nievergelt +, 84]
  - symmetric to all attributes
  - 2 disk accesses for exact match queries
  - adaptive to non-uniform distr.

Grid files

- partially match – eg., 0<x<0.3

Thus, we only need:
- cut-points for each axis
- k-d directory

x-cuts
\[ \frac{1}{4}, \frac{1}{2} \]
y-cuts
\[ \frac{1}{2} \]
**Grid files**

**Partial match** – eg., $0 < x < 0.3$

- $x$-cuts: $\frac{1}{4}$, $\frac{1}{2}$
- $y$-cuts: $\frac{1}{2}$

**Grid files**

**Exactly the symmetric algo** for eg., $0 < y < 0.3$

- $x$-cuts: $\frac{1}{4}$, $\frac{1}{2}$
- $y$-cuts: $\frac{1}{2}$

**Grid files**

- specs: [Nievergelt +, 84]
  - $x$ symmetric to all attributes
  - $x$ 2 disk accesses for exact match queries
  - adaptive to non-uniform distr.

**Grid files**

**Q: How to split an overflowing page?**

- $x$-cuts: $\frac{1}{4}$, $\frac{1}{2}$
- $y$-cuts: $\frac{1}{2}$

**Grid files**

**A: pick the ‘best’ axis, and cut all the way through**

- $x$-cuts: $\frac{1}{4}$, $\frac{1}{2}$
- $y$-cuts: $\frac{1}{2}$

**Grid files**

**A: pick the ‘best’ axis, and cut all the way through...**

- $x$-cuts: $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$
- $y$-cuts: $\frac{1}{2}$, $\frac{3}{8}$
Grid files

... updating the directory appropriately (ouch!)

\[
\begin{array}{ccc}
\frac{1}{4} & 3/8 & 1/2 \\
\frac{1}{4} & 1/2 & \ \\
\end{array}
\]

\[
\begin{array}{cc}
\text{x-cuts} & \frac{1}{4} \\
\text{y-cuts} & \frac{1}{2} \\
\end{array}
\]

Grid files - disadvantages

• #1: problems in high-d: directory splits can be expensive
  - symmetric to all attributes
  - 2 disk accesses for exact match queries
  - adaptive to non-uniform distr.

Grid files

• specs: [Nievergelt +, 84]

Grid files - disadvantages

• #2: even in low-d, suffers on correlated attributes:

Grid files

• it meets the three goals
• had follow-up work [twin grid files, multi-level; etc]
• BUT: has some disadvantages (which ones?)

Grid files - disadvantages

• #1: problems in high-d: directory splits can be expensive

Grid files - disadvantages

• #2: even in low-d, suffers on correlated attributes:

Grid files - disadvantages

• (Q: how to fix, for 2-d, linearly correlated points?)
Grid files - disadvantages

- (A1: rotate [Hinrichs+]; A2: triangular cells [Rego+])
- if we 'cut' them, then we have $O(\text{volume})$ pieces (while z-ordering: $O(\text{surface})$)
- what to do?

• (A1: rotate [Hinrichs+]; A2: triangular cells)

still has subtle problems) E.g., 1-d 'regions'

Grid files - disadvantages

- #3: how about region data?
- what to do?
- Translation to $2k$ – $d$ points! (clever, BUT, still has subtle problems) E.g., 1-d 'regions'

Grid files - disadvantages

- what to do?
- Translation to $2k$ – $d$ points! (clever, BUT, still has subtle problems) E.g., 1-d 'regions'
Grid files - disadvantages

• what is the problem, then?

Grid files - disadvantages

• what is the problem, then?
• A: dimensionality curse; large query regions

Grid files – conclusions

• works OK in low-d un-correlated points
• but z-ordering/R-trees seem to work better for higher-d
• smart idea to translate k-d rectangles into 2*k - points (but: dim. curse)

SAMs - Detailed outline

• spatial access methods
  – problem dfn
  – z-ordering
  – R-trees
  – misc topics
    • grid files
    • dimensionality curse; dim. reduction
    • metric trees
    • other nn methods
• text, ...

Dimensionality ‘curse’

• Q: What is the problem in high-d?

Dimensionality ‘curse’

• Q: What is the problem in high-d?
• A: indices do not seem to help, for many queries (eg., k-nn)
  – in high-d (& uniform distributions), most points are equidistant -> k-nn retrieves too many near-neighbors
  – [Yao & Yao, ’85]: search effort ~ O( N (1-1/d) )
Dimensionality ‘curse’

• (counter-intuitive, for db mentality)
• Q: What to do, then?

• A1: switch to seq. scanning
  – X-trees [Kriegel+, VLDB 96]
  – VA-files [Schek+, VLDB 98]

• A2: dim. reduction
• A3: consider the ‘intrinsic’/fractal dimensionality
• A4: find approximate nn

Dim. reduction

a.k.a. feature selection/extraction:
• SVD (optimal, to preserve Euclidean distances)
• random projections
• using the fractal dimension [Traina+SBBD2000]

Singular Value Decomposition

(SVD)

• SVD (~LSI ~ KL ~ PCA ~ spectral analysis...)

  LSI: S. Dumais; M. Berry
  KL: eg, Duda+Hart
  PCA: eg., Jolliffe
  MANY more details: soon
Random projections

- random projections (Johnson-Lindenstrauss thm [Papadimitriou+ PODS98])

Dim. reduction - w/ fractals

- Main idea: drop those attributes that don’t affect the intrinsic (‘fractal’) dimensionality [Traina+, SBBD 2000]

Dimensionality ‘curse’

- A1: switch to seq. scanning
- A2: dim. reduction
- A3: consider the ‘intrinsic’/fractal dimensionality
- A4: find approximate nn

Intrinsic dimensionality

- before we give up, compute the intrinsic dim.: 
  - the lower, the better... [Pagel+, ICDE 2000]
  - more details: under ‘fractals’

intrinsic d = 2

intrinsic d = 1
Dimensionality ‘curse’

- A1: switch to seq. scanning
- A2: dim. reduction
- A3: consider the ‘intrinsic’/ fractal dimensionality
- A4: find approximate nn

Approximate nn

- [Arya + Mount, SODA93], [Patella+ ICDE 2000]
- Idea: find k neighbors, such that the distance of the k-th one is guaranteed to be within epsilon of the actual.

SAMs - Detailed outline

- spatial access methods
  - problem dfn
  - z-ordering
  - R-trees
  - misc topics
    • grid files
    • dimensionality curse; dim. reduction
    • metric trees
    • other nn methods
  • text, ...

Conclusions

- Dimensionality ‘curse’:
  - for high-d, indices slow down to ~O(N)
- If the intrinsic dim. is low, there is hope
- otherwise, do seq. scan, or sacrifice accuracy (approximate nn)

References

References cnt’d