15-826: Multimedia Databases and Data Mining

Lecture #10: Fractals - case studies - I

C. Faloutsos

Must-read Material

• Christos Faloutsos and Ibrahim Kamel, *Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension*, Proc. ACM SIGACT-SIGMOD-SIGART PODS, May 1994, pp. 4-13, Minneapolis, MN.

Optional Material

Optional, but very useful: Manfred Schroeder *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise* W.H. Freeman and Company, 1991 (on reserve in the WeH library)

Reminder

• Code at
  www.cs.cmu.edu/~christos/SRC/fdnq_h.zip

Also, in ‘R’
> library(fdim);
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
  - z-ordering
  - R-trees
  - misc
- fractals
  - intro
  - applications
- text

(Fractals mentioned before:)

- for performance analysis of R-trees
- fractals for dim. reduction
Case study#1: R-tree performance

Problem

- Given
  - N points in E-dim space

- Estimate # disk accesses for a range query
  \((q_1 x ... x q_E)\)

(assume: ‘good’ R-tree, with tight, cube-like MBRs)

Typically, in DB Q-opt: uniformity + independence
Examples: World’s countries


For fun: identification

Examples: World’s countries

- neither uniform, nor independent!
For fun: identification

Examples: TIGER files

• neither uniform, nor independent!

How to proceed?

• recall the \[ \text{Pagel}^+ \] formula, for range queries of size \( q_1 \times q_2 \)

\[
\#\text{DiskAccesses}(q_1,q_2) = \sum (x_{i,1} + q_1) \times (x_{i,2} + q_2)
\]

But:

formula needs to know the \( x_{ij} \) sizes of MBRs!
How to proceed?

But:
formula needs to know the $x_{i,j}$ sizes of MBRs!

Answer (jumping ahead):

\[ s = \left( \frac{C}{N} \right)^{1/D_0} \]

Let’s see the rationale

\[ s = \left( \frac{C}{N} \right)^{1/D_0} \]

R-trees - performance analysis

I.e: for range queries - how many disk accesses, if we just now that we have
- $N$ points in $E$-d space?
A: can not tell! need to know distribution
R-trees - performance analysis

Q: OK - so we are told that the Hausdorff fractal dim. = D0 - Next step?
(also know that there are at most C points per page)

\[ D0 = 1 \quad D0 = 2 \]

Assumption 1: square-like parents (s*s)
Assumption 2: fully packed (C points each)
Assumption 3: non-overlapping

\[ s1 = s2 = s \]

Hint: dfn of Hausdorff f.d.:

Felix Hausdorff (1868-1942)
Reminder: Hausdorff or box-counting fd:
- Box counting plot: \( \log(N(r)) \) vs \( \log(r) \)
- \( r \): grid side
- \( N(r) \): count of non-empty cells
- (Hausdorff) fractal dimension \( D_0 \):
  \[
  D_0 = -\frac{\partial \log(N(r))}{\partial \log(r)}
  \]

Reminder
- Hausdorff fd:
  \[
  \log(\text{#non-empty cells}) = \text{slope} \cdot \log(r) + \text{intercept} = D_0 \cdot \log(r)
  \]

Reminder
- dfn of Hausdorff fd implies that
  \[
  N(r) \sim r^{-D_0}
  \]
  # non-empty cells of side \( r \)

R-trees - performance analysis
Q (rephrased): what is the side \( s_1, s_2, ... \) of parent nodes, given \( N \) data points, packed by \( C \), with f.d. = \( D_0 \)
- \( D_0 = 1 \)
- \( D_0 = 2 \)
R-trees - performance analysis

Q (rephrased): what is the side $s_1, s_2, \ldots$ of parent nodes, given $N$ data points, packed by $C$, with f.d. = $D_0$

A: (educated guess)
- $s = s_1 = s_2 = \ldots$ - square-like MBRs
- $N/C$ non-empty cells = $K \cdot s^{(-D_0)}$

Details of derivations: in [PODS 94].

Finally, expected side $s$ of parent MBRs:
$$s = (C/N)^{1/D_0}$$

Q: sanity check: how does $s$ change with $D_0$?
A:
R-trees - performance analysis

Details of derivations: in [Kamel+, PODS 94].

Finally, expected side $s$ of parent MBRs:

$$s = \frac{(C/N)^{1/D_0}}{s}$$

Q: sanity check: how does $s$ change with $D_0$?
A: $s$ grows with $D_0$
Q: does it make sense?

Q: does it suffer from (intrinsic) dim. curse?

Q: Final-final formula (# disk accesses for range queries $q_1 \times q_2 \times \ldots$):

A:

# of parent-node accesses:

$$\frac{N}{C} \times (s + q_1) \times (s + q_2) \times \ldots \times (s + q_E)$$

A: # of grand-parent node accesses

$$\frac{N}{(C^2)} \times (s' + q_1) \times (s' + q_2) \times \ldots \times (s' + q_E)$$

$$s' = ??$$
R-trees - performance analysis

Q: Final-final formula (# disk accesses for range queries \( q_1 \times q_2 \times \ldots \)):
A: # of parent-node accesses:
\[
\frac{N}{C} \times (s + q_1) \times (s + q_2) \times \ldots \times (s + q_E)
\]
A: # of grand-parent node accesses
\[
\frac{N}{C^2} \times (s' + q_1) \times (s' + q_2) \times \ldots \times (s' + q_E)
\]
\[s' = \left(\frac{C^2}{N}\right)^{1/D_0}\]

R-trees - performance analysis

Results:

IUE (x-y star coordinates)

# leaf accesses

query side

R-trees - performance analysis

Results:

LB County

# leaf accesses

query side

R-trees - performance analysis

Results:

MG-county

# leaf accesses

query side
R-trees - performance analysis

Results:

2D-uniform

# leaf accesses

query side

Conclusions: usually, <5% relative error, for range queries

Indexing - Detailed outline

- fractals
  - intro
  - applications
    - disk accesses for R-trees (range queries)
      - dimensionality reduction
      - dim. curse revisited
      - quad-tree analysis [Gaede+]
    - ....

Case study #2: Dim. reduction

Problem definition: ‘Feature selection’
- given $N$ points, with $E$ dimensions
- keep the $k$ most ‘informative’ dimensions

[Traina+,SBBD’00]
**Dim. reduction**

Problem definition: ‘Feature selection’
- given $N$ points, with $E$ dimensions
- keep the $k$ most ‘informative’ dimensions

Re-phrased: spot and drop attributes with strong (non-)linear correlations

Q: how do we do that?

---

**Dim. reduction - w/ fractals**

A: Hint: correlated attributes do not affect the intrinsic/fractal dimension, e.g., if

$$y = f(x,z,w)$$

we can drop $y$

(hence: ‘partial fd’ (PFD) of a set of attributes = the fd of the dataset, when projected on those attributes)
Dim. reduction - w/ fractals

- (problem: given N points in E-d, choose k best dimensions)
- Q: Algorithm?

- A: e.g., greedy - forward selection:
  - keep the attribute with highest partial fd
  - add the one that causes the highest increase in pfd
  - etc., until we are within $\epsilon$ from the full f.d.
Dim. reduction - w/ fractals

- (backward elimination: ~ reverse)
  - drop the attribute with least impact on the p.f.d.
  - repeat
  - until we are *epsilon* below the full f.d.

Q: what is the smallest # of attributes we should keep?

- A: we should keep at least as many as the f.d. (and probably, a few more)

Results: E.g., on the ‘currency’ dataset
- (daily exchange rates for USD, HKD, BP, FRF, DEM, JPY - i.e., 6-d vectors, one per day - base currency: CAD)

E.g.: FRF

USD
E.g., on the ‘currency’ dataset

\[ \log(\#\text{pairs}(<r)) \]

correlation integral

Dim. reduction - w/ fractals

Conclusion:
- can do non-linear dim. reduction

References