CMU SCS

15-826: Multimedia Databases and Data Mining

Lecture #10: Fractals - case studies - I

C. Faloutsos

CMU SCS

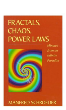
Must-read Material

- 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.

CMU SCS

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)



CMU SCS

Outline

Goal: 'Find similar / interesting things'

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

CMU SCS

Indexing - Detailed outline

- primary key indexing
- secondary key / multi-key indexing
- spatial access methods
  - z-ordering
  - R-trees
  - misc
- fractals
  - intro
  - applications
- text
- ...

CMU SCS

Indexing - Detailed outline

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

CMU SCS

(Fractals mentioned before:)

- for performance analysis of R-trees
- fractals for dim. reduction

CMU SCS

Case study#1: R-tree performance

Problem

- Given
  - N points in E-dim space
- Estimate # disk accesses for a range query
  ($q1 \times ... \times q_E$)

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

CMU SCS

Case study#1: R-tree performance

Problem

- Given
  - N points in E-dim space
  - with fractal dimension D
- Estimate # disk accesses for a range query
  ($q1 \times ... \times q_E$)

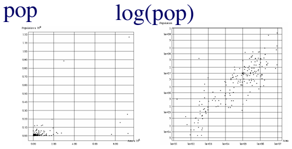
(assume: 'good' R-tree, with tight, cube-like MBRs)
Typically, in DB Q-opt: uniformity + independence

CMU SCS

Examples: World's countries

- BUT: area vs population for ~200 countries (1991 CIA fact-book).

pop log(pop)



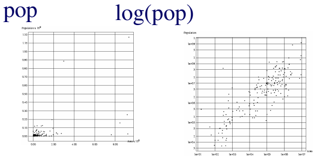
area log(area)

CMU SCS

Examples: World's countries

- neither uniform, nor independent!

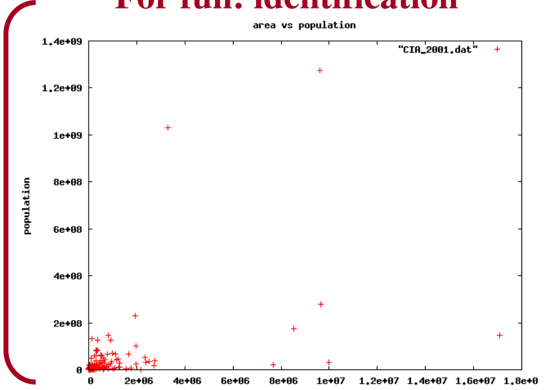
pop log(pop)



area log(area)

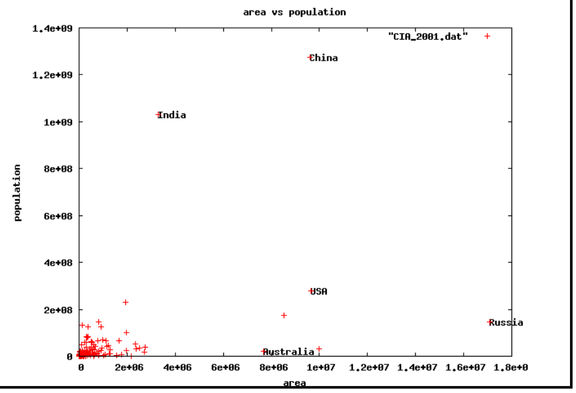
CMU SCS

For fun: identification



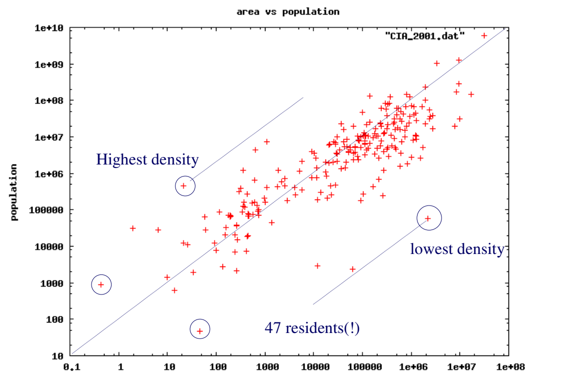
CMU SCS

For fun: identification



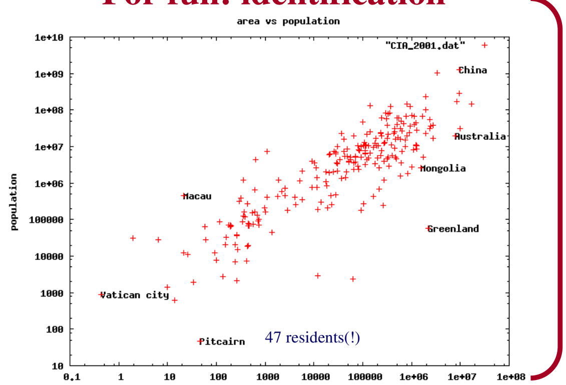
CMU SCS

For fun: identification



CMU SCS

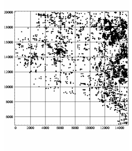
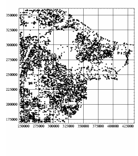
For fun: identification



Examples: TIGER files

- neither uniform, nor independent!

MG county LB county

How to proceed?

- recall the [Pagel+] formula, for range queries of size q1 x q2

$$\#DiskAccesses(q1,q2) = sum\ (x_{i,1} + q1) * (x_{i,2} + q2)$$

But:

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

How to proceed?

But:

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

Answer (jumping ahead):

$$\boxed{s = (C/N)^{1/D0}}$$

How to proceed?

But:

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

Answer (jumping ahead):

$s = (C/N)^{1/D0}$ ← **Hausdorff fd**

side of (parent) MBR

# of data points

page capacity

Let's see the rationale

$\boxed{s = (C/N)^{1/D0}}$

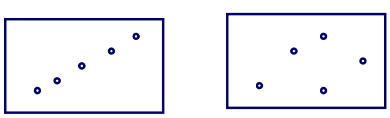
proof

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



CMU SCS

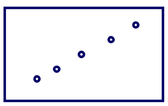
proof

R-trees - performance analysis

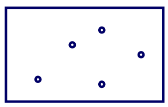
Q: OK - so we are told that the **Hausdorff** fractal dim. = D0 - Next step?

(also know that there are at most *C* points per page)

D0=1



D0=2

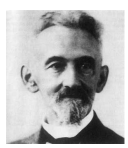


CMU SCS

proof

R-trees - performance analysis

Hint: dfn of Hausdorff f.d.:



Felix Hausdorff (1868-1942)

CMU SCS

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 D0:

$$D_0 = -\frac{\partial \log(N(r))}{\partial \log(r)}$$

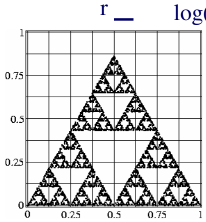
CMU SCS

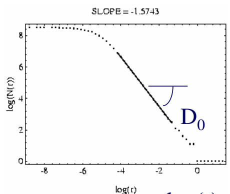
proof

Reminder

- Hausdorff fd:

r log(#non-empty cells)





log(r)

CMU SCS

proof

Reminder

- dfn of Hausdorff fd implies that

$N(r) \sim r^{(-D0)}$

\# non-empty cells of side *r*

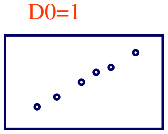
CMU SCS

proof

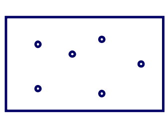
R-trees - performance analysis

Q (rephrased): what is the side s1, s2, ... of parent nodes, given *N* data points, packed by *C,* with f.d. = *D0*

D0=1



D0=2

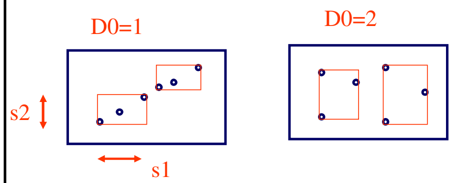


CMU SCS

proof

R-trees - performance analysis

Q (rephrased): what is the side s1, s2, ... of parent nodes, given *N* data points, packed by *C,* with f.d. = *D0*

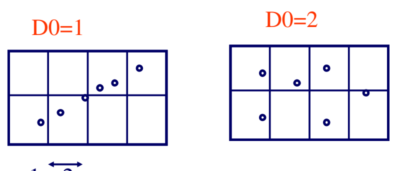


CMU SCS

proof

R-trees - performance analysis

Q (rephrased): what is the side s1, s2, ... of parent nodes, given *N* data points, packed by *C,* with f.d. = *D0*



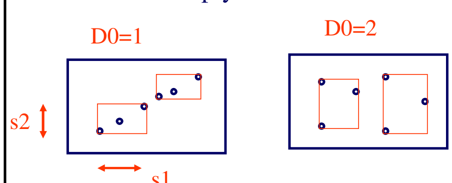
CMU SCS

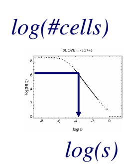
proof

R-trees - performance analysis

A: (educated guess)

- s=s1=s2 (= ...) - square-like MBRs
- N/C non-empty cells = K * $s^{(-D0)}$





CMU SCS

proof

R-trees - performance analysis

Details of derivations: in [PODS 94].

Finally, expected side *s* of parent MBRs:

$$\boxed{s = (C/N)^{1/D0}}$$

Q: sanity check: how does *s* change with *D0?*

A:

CMU SCS

R-trees - performance analysis

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

Finally, expected side *s* of parent MBRs:

$$s = (C/N)^{1/D0}$$

Q: sanity check: how does *s* change with *D0?*

A: *s* grows with *D0*

Q: does it make sense?

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

CMU SCS

R-trees - performance analysis

Q: Final-final formula (# disk accesses for range queries *q1* x *q2* x ...):

A:

CMU SCS

R-trees - performance analysis

Q: Final-final formula (# disk accesses for range queries *q1* x *q2* x ...):

A: # of parent-node accesses:

$N/C * (s + q1) * (s + q2) * \ldots (s + q_E)$

A: # of grand-parent node accesses

CMU SCS

R-trees - performance analysis

Q: Final-final formula (# disk accesses for range queries *q1* x *q2* x ...):

A: # of parent-node accesses:

$N/C * (s + q1) * (s + q2) * \ldots (s + q_E)$

A: # of grand-parent node accesses

$N/(C\text{^}2) * (s' + q1) * (s' + q2) * \ldots (s' + q_E)$

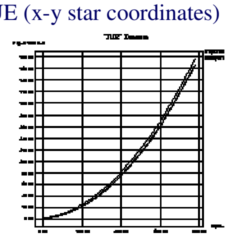
$s' = (C\text{^}2/N)^{1/D0}$

CMU SCS

R-trees - performance analysis

Results: IUE (x-y star coordinates)

leaf accesses



(a) IUE - Leaf accesses vs. query s

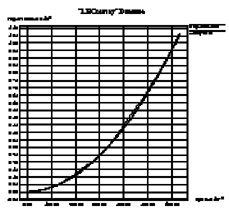
query side

CMU SCS

R-trees - performance analysis

Results: LB County

leaf accesses



(b) LBCounty - Leaf accesses vs. query side

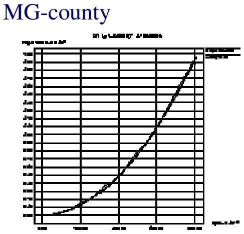
query side

CMU SCS

R-trees - performance analysis

Results: MG-county

leaf accesses



(c) MGCounty - Leaf accesses vs. query side

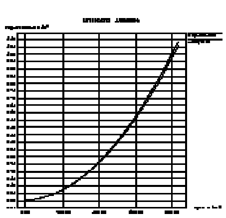
query side

CMU SCS

R-trees - performance analysis

Results: 2D- uniform

leaf accesses



(d) 2D-UNIFORM - Leaf accesses vs. query side

query side

CMU SCS

R-trees - performance analysis

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

CMU SCS

Indexing - Detailed outline

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

CMU SCS

Case study #2: Dim. reduction

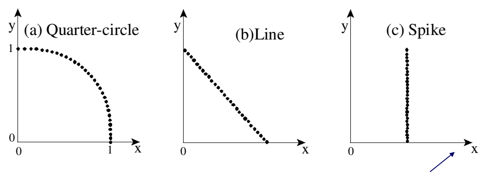
Problem definition: ‘Feature selection’
- given N points, with E dimensions
- keep the k most ‘informative’ dimensions

[Traina+,SBBD’00]

Caetano Traina Agma Traina Leejay Wu

CMU SCS

Dim. reduction - w/ fractals



not informative

CMU SCS

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?

CMU SCS

Dim. reduction

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

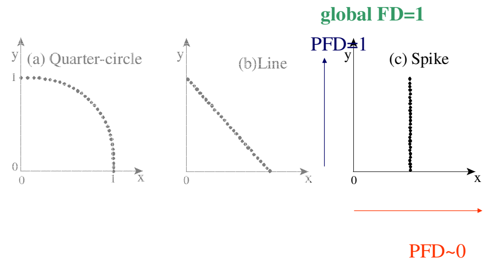
$$y = f(x,z,w)$$

we can drop y

(hence: '*partial fd*' (PFD) of a set of attributes = the fd of the dataset, when projected on those attributes)

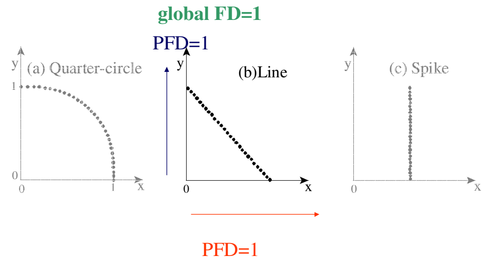
CMU SCS

Dim. reduction - w/ fractals



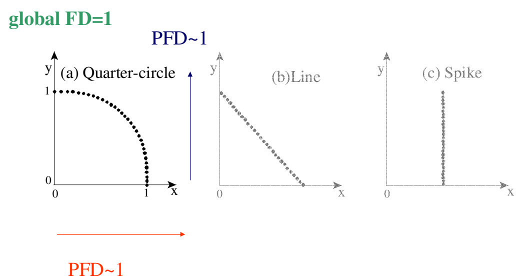
CMU SCS

Dim. reduction - w/ fractals



CMU SCS

Dim. reduction - w/ fractals



CMU SCS

Dim. reduction - w/ fractals

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

CMU SCS

Dim. reduction - w/ fractals

- 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.

CMU SCS

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.

Dim. reduction - w/ fractals

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

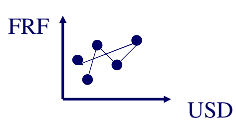
Dim. reduction - w/ fractals

- 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)

Dim. reduction - w/ fractals

- 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)

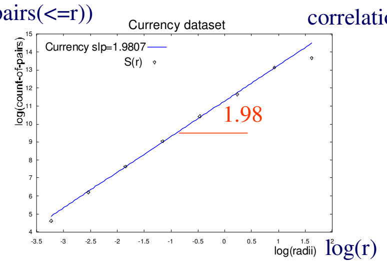
e.g.:



CMU SCS

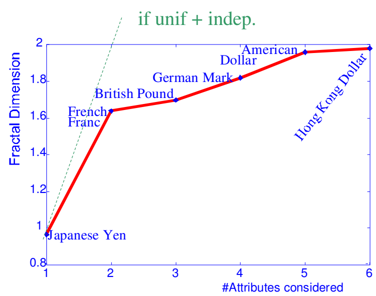
E.g., on the 'currency' dataset

log(#pairs(<=r)) correlation integral



CMU SCS

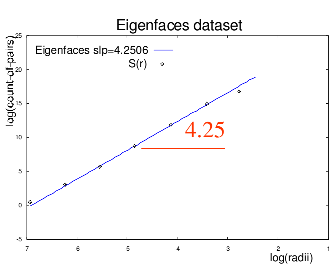
E.g., on the 'currency' dataset



CMU SCS

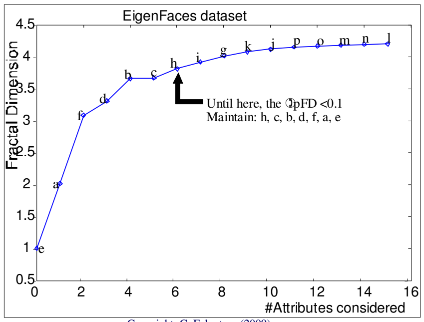
E.g., on the eigenface dataset

16-d vectors, one for each of ~1K faces



CMU SCS

E.g., on the eigenface dataset

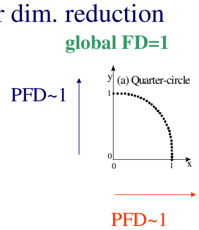


CMU SCS

Dim. reduction - w/ fractals

Conclusion:

- can do non-linear dim. reduction



CMU SCS

References

- [PODS94] Faloutsos, C. and I. Kamel (May 24-26, 1994). *Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension*. Proc. ACM SIGACT-SIGMOD-SIGART PODS, Minneapolis, MN.
- [Traina+, SBBD'00] Traina, C., A. Traina, et al. (2000). *Fast feature selection using the fractal dimension*. XV Brazilian Symposium on Databases (SBBD), Paraiba, Brazil.