Outline

- Motivation
- Similarity search – distance functions
- Linear Forecasting
- Bursty traffic - fractals and multifractals
- Non-linear forecasting
- Conclusions

Motivation - Applications

- Financial, sales, economic series
- Medical
  - ECGs +; blood pressure etc monitoring
  - reactions to new drugs
  - elderly care

Motivation - Applications (cont’d)

- ‘Smart house’
  - sensors monitor temperature, humidity, air quality
- video surveillance

Problem definition

- Given: one or more sequences
  \( x_1, x_2, \ldots, x_t, \ldots \)
  \( y_1, y_2, \ldots, y_p, \ldots \)
- Find
  - similar sequences; forecasts
  - patterns; clusters; outliers

Thanks

Deepay Chakrabarti (CMU)
Prof. Dimitris Gunopulos (UCR)
Spiros Papadimitriou (CMU)
Mengzhi Wang (CMU)
Prof. Byoung-Kee Yi (Pohang U.)
**Motivation - Applications**

(continuation)

- Civil/automobile infrastructure
  - Bridge vibrations [Oppenheim+02]
  - Road conditions / traffic monitoring

**Weather, environment/anti-pollution**

- Volcano monitoring
- Air/water pollutant monitoring

**Motivation - Applications**

(continuation)

- Computer systems
  - ‘Active Disks’ (buffering, prefetching)
  - Web servers (ditto)
  - Network traffic monitoring
  - ...

**Settings & Applications**

- One or more sensors, collecting time-series data

Each sensor collects data \((x_1, x_2, \ldots, x_t, \ldots)\)
**Goal #1:**
Finding patterns in a single time sequence

**Problem #1:**
Given a signal (e.g., packets over time)
Find: patterns, periodicities, and/or compress

**Goal #2:**
Finding patterns in many time sequences

**Problem #2:** Forecast
Given \( x_1, x_2, \ldots, x_t \), forecast \( x_{t+1} \)

**Problem #2'**: Similarity search
E.g., Find a 3-tick pattern, similar to the last one
Differences from DSP/Stat

- Semi-infinite streams
  - need on-line, 'any-time' algorithms
- Can not afford human intervention
  - need automatic methods
- Sensors have limited memory / processing / transmitting power
  - need for (lossy) compression

Important topics NOT in this tutorial:

- Continuous queries
  - [Babu+Widom] [Gehrke+] [Madden+]
- Categorical data streams
  - [Hatonen+96]
- Outlier detection (discontinuities)
  - [Breunig+00]
- Related (see D. Shasha’s tutorial)

Outline

- Motivation
- Similarity Search and Indexing
  - DSP
  - Linear Forecasting
  - Bursty traffic - fractals and multifractals
  - Non-linear forecasting
- Conclusions
Importance of distance functions
Subtle, but absolutely necessary:
• A ‘must’ for similarity indexing (-> forecasting)
• A ‘must’ for clustering
Two major families
– Euclidean and Lp norms
– Time warping and variations

Euclidean and Lp
\[ D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (x_i - y_i)^2 \]
\[ L_p(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \]
• \( L_1 \): city-block = Manhattan
• \( L_2 \): Euclidean
• \( L_\infty \)

Observation #1
• Time sequence -> n-d vector

Observation #2
Euclidean distance is closely related to
– cosine similarity
– dot product
– ‘cross-correlation’ function

Time Warping
• allow accelerations - decelerations
  – (with or w/o penalty)
• THEN compute the (Euclidean) distance (+ penalty)
• related to the string-editing distance

Time Warping
‘stutters’: day-n day-2 day-1
Time warping

Q: how to compute it?
A: dynamic programming
\[ D(i, j) = \text{cost to match} \]
prefix of length \(i\) of first sequence \(x\) with prefix of length \(j\) of second sequence \(y\)

Thus, with no penalty for stutter, for sequences \(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j\)

\[ D(i, j) = \begin{cases} 
D(i-1, j-1) & \text{no stutter} \\
D(i, j-1) & \text{x-stutter} \\
D(i-1, j) & \text{y-stutter}
\end{cases} \]

Time warping

VERY SIMILAR to the string-editing distance

\[ D(i, j) = \|x[i] - y[j]\| + \min \begin{cases} 
D(i-1, j-1) & \text{no stutter} \\
D(i, j-1) & \text{x-stutter} \\
D(i-1, j) & \text{y-stutter}
\end{cases} \]

Other Distance functions

- piece-wise linear/flat approx.; compare pieces [Keogh+01] [Faloutsos+97]
- ‘cepstrum’ (for voice [Rabiner+Juang])
  – do DFT; take log of amplitude; do DFT again!
- Allow for small gaps [Agrawal+95]
  See tutorial by [Gunopulos Das, SIGMOD01]
Conclusions

Prevailing distances:
- Euclidean and
- time-warping

Outline

- Motivation
- Similarity search and distance functions
  - Linear Forecasting
    - Bursty traffic - fractals and multifractals
    - Non-linear forecasting
  - Conclusions

Forecasting

"Prediction is very difficult, especially about the future." - Nils Bohr

http://www.hfac.uh.edu/MediaFutures/thoughts.html

Outline

- Motivation
- ...
- Linear Forecasting
  - Auto-regression: Least Squares; RLS
  - Co-evolving time sequences
  - Examples
  - Conclusions

Problem#2: Forecast

- Example: give $x_0, x_1, \ldots$, forecast $x_t$
**Forecasting: Preprocessing**

MANUALLY:
- remove trends
- spot periodicities
  - 7 days

**Problem#2: Forecast**

- Solution: try to express $x_t$ as a linear function of the past: $x_{t-2}, x_{t-3}, \ldots$, (up to a window of $w$)
- Formally:

$$x_t = a_1 x_{t-1} + \ldots + a_w x_{t-w} + \text{noise}$$

**Linear Regression: idea**

- express what we don’t know (= ‘dependent variable’)
- as a linear function of what we know (= ‘indep. variable(s)’)

**Linear Auto Regression:**

<table>
<thead>
<tr>
<th>Time</th>
<th>Packet Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>55</td>
</tr>
</tbody>
</table>

- lag $w=1$
- Dependent variable = # of packets sent ($S(t)$)
- Independent variable = # of packets sent ($S(t-1)$)
Outline

• Motivation
• ...
• Linear Forecasting
  – Auto-regression: Least Squares; RLS
  – Co-evolving time sequences
  – Examples
  – Conclusions

More details:

• Q1: Can it work with window \( w > 1 \)?
• A1: YES!

\[ \begin{align*}
X_t \quad & \quad \downarrow \quad X_{t-1} \\
\uparrow & \quad \quad \uparrow \\
x_i \quad & \quad x_{i-1} \\
x_{i-2} & \quad x_{i-2}
\end{align*} \]

More details:

• Q1: Can it work with window \( w > 1 \)?
• A1: YES! (we’ll fit a hyper-plane, then!)

\[ \begin{align*}
X_t \quad & \quad \downarrow \quad X_{t-1} \\
\uparrow & \quad \quad \uparrow \\
x_i \quad & \quad x_{i-1} \\
x_{i-2} & \quad x_{i-2}
\end{align*} \]

More details:

• Q1: Can it work with window \( w > 1 \)?
• A1: YES! (we’ll fit a hyper-plane, then!)

\[ \begin{align*}
X_t \quad & \quad \downarrow \quad X_{t-1} \\
\uparrow & \quad \quad \uparrow \\
x_i \quad & \quad x_{i-1} \\
x_{i-2} & \quad x_{i-2}
\end{align*} \]

More details:

• Q1: Can it work with window \( w > 1 \)?
• A1: YES! The problem becomes:
  \[ X_{[N \times w]} \times a_{[w \times 1]} = y_{[N \times 1]} \]
• OVER-CONSTRAINED
  – \( a \) is the vector of the regression coefficients
  – \( X \) has the \( N \) values of the \( w \) indep. variables
  – \( y \) has the \( N \) values of the dependent variable
More details:

- \( \mathbf{X}_{[N \times w]} \mathbf{a}_{[w \times 1]} = \mathbf{y}_{[N \times 1]} \)

Ind-var1  Ind-var-w

\[
\begin{bmatrix}
X_{11}, X_{12}, \ldots, X_{1w} \\
X_{21}, X_{22}, \ldots, X_{2w} \\
\vdots \\
X_{N1}, X_{N2}, \ldots, X_{Nw}
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_w
\end{bmatrix}
= 
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{bmatrix}
\]

More details

- Q2: How to estimate \( a_1, a_2, \ldots a_w = \mathbf{a} \)?
- A2: with Least Squares fit
  \[
  \mathbf{a} = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{X}^T \mathbf{y})
  \]
  (Moore-Penrose pseudo-inverse)
  \( \mathbf{a} \) is the vector that minimizes the RMSE from \( \mathbf{y} \)
  - \(<\text{identical} \) math with ‘query feedbacks’>

Even more details

- Q3: Can we estimate \( \mathbf{a} \) incrementally?
- A3: Yes, with the brilliant, classic method of ‘Recursive Least Squares’ (RLS) (see, e.g., [Yi+00], for details).
- We can do the matrix inversion, WITHOUT inversion! (How is that possible?!)

Even more details

- Q3: Can we estimate \( \mathbf{a} \) incrementally?
- A3: Yes, with the brilliant, classic method of ‘Recursive Least Squares’ (RLS) (see, e.g., [Yi+00], for details).
- We can do the matrix inversion, WITHOUT inversion! (How is that possible?!)
- A: our matrix has special form: \( (\mathbf{X}^T \mathbf{X}) \)

More details

At the \( N+1 \) time tick:
More details

- Let \( G_N = (X_N^T \times X_N)^{-1} \) ("gain matrix")
- \( G_{N+1} \) can be computed recursively from \( G_N \)

\[
G_N = \begin{bmatrix}
    W & \vdots \\
    \vdots & W
\end{bmatrix}
\]

EVEN more details:

\[
G_{N+1} = G_N - [c]^{-1} \times [G_N \times x_{N+1}^T] \times x_{N+1} \times G_N
\]

\[
c = [1 + x_{N+1} \times G_N \times x_{N+1}^T]
\]

Let’s elaborate
(VERY IMPORTANT, VERY VALUABLE!)

EVEN more details:

\[
a = [X_{N+1}^T \times X_{N+1}]^{-1} \times [X_{N+1}^T \times y_{N+1}]
\]

\[
[(N+1) \times w]
\]

EVEN more details:

\[
a = [X_{N+1}^T \times X_{N+1}]^{-1} \times [X_{N+1}^T \times y_{N+1}]
\]

\[
[w \times (N+1)]
\]

EVEN more details:

\[
a = [X_{N+1}^T \times X_{N+1}]^{-1} \times [X_{N+1}^T \times y_{N+1}]
\]

\[
[w \times (N+1)]
\]

EVEN more details:

\[
a = [X_{N+1}^T \times X_{N+1}]^{-1} \times [X_{N+1}^T \times y_{N+1}]
\]

\[
1 \times w \text{ row vector}
\]

\[
G_{N+1} = [X_{N+1}^T \times X_{N+1}]^{-1}
\]

\[
G_{N+1} = G_N - [c]^{-1} \times [G_N \times x_{N+1}^T] \times x_{N+1} \times G_N
\]

\[
c = [1 + x_{N+1} \times G_N \times x_{N+1}^T]
\]

\[
\text{`gain matrix''}
\]

\[
\text{EVEN more details:}
\]
EVEN more details:

\[ G_{N+1} = G_N - \left[c^{-1} \times \left[G_N \times x_{N+1}^T \right] \times x_{N+1} \times G_N \right] \]

\[ c = [1 + x_{N+1} \times G_N \times x_{N+1}^T] \]

Altogether:

\[ a = \left[ X_{N+1}^T \times X_{N+1} \right]^{-1} \times \left[ X_{N+1}^T \times y_{N+1} \right] \]

\[ G_{N+1} = [X_{N+1}^T \times X_{N+1}]^{-1} \]

\[ G_{N+1} = G_N - \left[c^{-1} \times \left[G_N \times x_{N+1}^T \right] \times x_{N+1} \times G_N \right] \]

\[ c = [1 + x_{N+1} \times G_N \times x_{N+1}^T] \]

Comparison:

- **Straightforward Least Squares**
  - Needs huge matrix (growing in size)
  - \(O(Nw^2)\)
  - Costly matrix operation \(O(Nw^2)\)
- **Recursive LS**
  - Need much smaller, fixed size matrix
  - \(O(w \times w)\)
  - Fast, incremental computation \(O(1 \times w^2)\)
  - No matrix inversion

\[ N = 10^9, \quad w = 1 \text{-} 100 \]

Pictorially:

- **Given:**

  ![Dependent Variable vs. Independent Variable Graph]
Pictorially:

Independent Variable

Dependent Variable

new point

RLS: quickly compute new best fit

Pictorially:

Independent Variable

Dependent Variable

new point

Even more details

• Q4: can we ‘forget’ the older samples?
• A4: Yes - RLS can easily handle that [Yi+00]:

Adaptability - ‘forgetting’

Trend change

(R)LS with no forgetting

Adaptability - ‘forgetting’

Trend change

(R)LS with forgetting

• RLS: can *trivially* handle ‘forgetting’
How to choose ‘\(w\)?’

- goal: capture arbitrary periodicities
- with NO human intervention
- on a semi-infinite stream

Answer:

- ‘AWSOM’ (Arbitrary Window Stream fOrcasting Method) [Papadimitriou+ vldb2003]
- idea: do AR on each wavelet level
- in detail:

AWSOM

AWSOM - idea

More details...

- Update of wavelet coefficients (incremental)
- Update of linear models (incremental; RLS)
- Feature selection (single-pass)
  - Not all correlations are significant
  - Throw away the insignificant ones (‘noise’)

AWSOM
**Results - Synthetic data**

- Triangle pulse
- Mix (sine + square)
- AR captures wrong trend (or none)
- Seasonal AR estimation fails

**Results - Real data**

- Automobile traffic
  - Daily periodicity
  - Bursty “noise” at smaller scales
- AR fails to capture any trend
- Seasonal AR estimation fails

**Results - real data**

- Sunspot intensity
  - Slightly time-varying “period”
- AR captures wrong trend
- Seasonal ARIMA
  - wrong downward trend, despite help by human!

**Complexity**

- Model update
  - Space: $O(\lg N + mk^2) = O(\lg N)$
  - Time: $O(k^2) = O(1)$
- Where
  - $N$: number of points (so far)
  - $k$: number of regression coefficients; fixed
  - $m$: number of linear models; $O(\lg N)$

**Outline**

- Motivation
- ...
- Linear Forecasting
  - Auto-regression: Least Squares; RLS
  - Co-evolving time sequences
  - Examples
  - Conclusions

**Co-Evolving Time Sequences**

- Given: A set of correlated time sequences
- Forecast ‘Repeated(t)’
**Solution:**

Q: what should we do?

**Forecasting - Outline**

- Auto-regression
- Least Squares; recursive least squares
- Co-evolving time sequences
- Examples
- Conclusions

**Examples - Experiments**

- Datasets
  - Modem pool traffic (14 modems, 1500 time-ticks)
  - AT&T WorldNet internet usage (several data streams; 980 time-ticks)
- Measures of success
  - Accuracy: Root Mean Square Error (RMSE)

**Accuracy - “Modem”**

MUSCLES outperforms AR & “yesterday”

**Accuracy - “Internet”**

MUSCLES consistently outperforms AR & “yesterday”
B.II - Time Series Analysis - Outline

- Auto-regression
- Least Squares; recursive least squares
- Co-evolving time sequences
- Examples
- Conclusions

Conclusions - Practitioner’s guide

- AR(IMA) methodology: prevailing method for linear forecasting
- Brilliant method of Recursive Least Squares for fast, incremental estimation.
- See [Box-Jenkins]
- very recently: AWSOM (no human intervention)

Resources: software and urls

- MUSCLES: Prof. Byoung-Kee Yi:
  http://www.postech.ac.kr/~bkyi/
  or christos@cs.cmu.edu
- free-ware: ‘R’ for stat. analysis
  (clone of Splus)
  http://cran.r-project.org/

Books


Additional Reading

- [Yi+00] Byoung-Kee Yi et al.: *Online Data Mining for Co-Evolving Time Sequences*, ICDE 2000. (Describes MUSCLES and Recursive Least Squares)

Outline

- Motivation
- Similarity search and distance functions
- Linear Forecasting
  - Bursty traffic - fractals and multifractals
- Non-linear forecasting
- Conclusions
Outline

• Motivation
•...
• Linear Forecasting
  • Bursty traffic - fractals and multifractals
    – Problem
    – Main idea (80/20, Hurst exponent)
    – Results

Recall: Problem #1:
Goal: given a signal (e.g., #bytes over time)
Find: patterns, periodicities, and/or compress

Problem #1
• model bursty traffic
• generate realistic traces
• (Poisson does not work)

Motivation
• predict queue length distributions (e.g., to give probabilistic guarantees)
• “learn” traffic, for buffering, prefetching, ‘active disks’, web servers

Q: any ‘pattern’?
• Not Poisson
• spike; silence; more spikes; more silence…
• any rules?
**Solution: Self-similarity**

- # bytes vs. time graph showing self-similarity.

**But:**

- Q1: How to generate realistic traces; extrapolate; give guarantees?
- Q2: How to estimate the model parameters?

**Outline**

- Motivation
- ...
- Linear Forecasting
- Bursty traffic - fractals and multifractals
  - Problem
  - Main idea (80/20, Hurst exponent)
  - Results

**Approach**

- Q1: How to generate a sequence, that is
  - bursty
  - self-similar
  - and has similar queue length distributions

**Approach**

- A: ‘binomial multifractal’ [Wang+02]
  - ~ 80-20 ‘law’:
    - 80% of bytes/queries etc on first half
    - repeat recursively
  - b: bias factor (e.g., 80%)

**Binary multifractals**

- Diagram showing 80% bias factor.

---

Note: The diagrams and graphs are integral to the understanding of the content and are essential for the accurate representation of the document.
• Q2: How to estimate the bias factor $b$?

• Parameter estimation

- Rationale:
  - burstiness: inverse of uniformity
  - entropy measures uniformity of a distribution
  - find entropy at several granularities, to see whether/how our distribution is close to uniform.

- Entropy

- Even DFT amplitude spectrum! (‘periodogram’)

- Rationale:
  - Hurst exponent
  - Variance plot
  - Even DFT amplitude spectrum! (‘periodogram’)
  - More robust: ‘entropy plot’ [Wang+02]
Real traffic

Entropy

\[ E(n) \]

\[ \text{# of levels (n)} \]

- Has linear entropy plot (\( \Rightarrow \) self-similar)

0.73

Observation - intuition:

Entropy

\[ E(n) \]

\[ \text{# of levels (n)} \]

intuition: slope = slope

intrinsic dimensionality = info-bits per coordinate-bit

- unif. Dataset: slope = 1
- multi-point: slope = ?

Entropy plot - Intuition

- Slope \( \sim \) intrinsic dimensionality (in fact, ‘Information fractal dimension’)
- = info bit per coordinate bit - eg

Dim = 1

Pick a point;
reveal its coordinate bit-by-bit - how much info is each bit worth to me?

Entropy plot

- Slope \( \sim \) intrinsic dimensionality (in fact, ‘Information fractal dimension’)
- = info bit per coordinate bit - eg

Dim = 1

Is MSB 0?

‘info’ value = \( E(1) \): 1 bit

Is next MSB = 0?
Entropy plot

- Slope ~ intrinsic dimensionality (in fact, 'Information fractal dimension')
- = info bit per coordinate bit - eg
  Dim = 1
  Info value = 1 bit
  Is MSB 0?
  Is next MSB 0?

• Repeat, for all points at same position:

  Dim=0

(Fractals, again)

- What set of points could have behavior between point and line?

Cantor dust

- Eliminate the middle third
- Recursively!
Cantor dust

Cantor dust

Cantor dust

Cantor dust

Cantor dust

Cantor dust

Some more entropy plots:

- Poisson vs real

Poisson: slope = -1 -> uniformly distributed

Dimensionality?
(no length; infinite # points!)
Answer: log2 / log3 = 0.6
Outline

- Motivation
- ...
- Linear Forecasting
- Bursty traffic - fractals and multifractals
  - Problem
  - Main idea (80/20, Hurst exponent)
  - Experiments - Results

Experimental setup

- Disk traces (from HP [Wilkes 93])
- Web traces from LBL
  \[ \text{http://repository.cs.vt.edu/}
  \text{lbl-conn-7.tar.Z} \]

Model validation

- Linear entropy plots

Web traffic - results

- LBL, NCDF of queue lengths (log-log scales)

Prob( >l)

(a) lbl-all
(b) lbl-ntpa
(c) lbl-smp
(d) lbl-flo

How to give guarantees?

(queue length l)
**Conclusions**

- Multifractals (80/20, ‘b-model’, Multiplicative Wavelet Model (MWM)) for analysis and synthesis of bursty traffic

**Further reading:**


**Outline**

- Motivation
- ...
- Linear Forecasting
- Bursty traffic - fractals and multifractals
- Non-linear forecasting
- Conclusions

**Books**


**Further reading**

Detailed Outline

- Non-linear forecasting
  - Problem
  - Idea
  - How-to
  - Experiments
  - Conclusions

Recall: Problem #1

Given a time series \{x_t\}, predict its future course, that is, \(x_{t+1}, x_{t+2}, \ldots\)

Questions:

- Q1: How to choose lag \(L\)?
- Q2: How to choose \(k\) (the # of NN)?
- Q3: How to interpolate?
- Q4: why should this work at all?

Q1: Choosing lag \(L\)

- Manually (16, in award winning system by [Sauer94])

How to forecast?

- ARIMA - but: linearity assumption

ANSWER: ‘Delayed Coordinate Embedding’ = Lag Plots [Sauer92]

General Intuition (Lag Plot)

Lag = 1,
\(k = 4\) NN

Interpolate these…

To get the final prediction
Q2: Choosing number of neighbors $k$

- Manually (typically $1-10$)

Q3: How to interpolate?

How do we interpolate between the $k$ nearest neighbors?

A3.1: Average

A3.2: Weighted average (weights drop with distance - how?)

Q3: How to interpolate?

A3.3: Using SVD - seems to perform best ([Sauer94] - first place in the Santa Fe forecasting competition)

Q4: Any theory behind it?

A4: YES!

Theoretical foundation

- Based on the “Takens’ Theorem” ([Takens81])
- which says that long enough delay vectors can do prediction, even if there are unobserved variables in the dynamical system (= diff. equations)

Example: Lotka-Volterra equations

\[
\begin{align*}
dH/dt &= r H - a H^2 P \\
dP/dt &= b H P - m P
\end{align*}
\]

H is count of prey (e.g., hare)
P is count of predators (e.g., lynx)

Suppose only $P(t)$ is observed ($t=1, 2, \ldots$).
Detailed Outline

- Non-linear forecasting
  - Problem
  - Idea
  - How-to
- Experiments
- Conclusions

Datasets

Logistic Parabola:
\[ x_t = ax_{t-1}(1-x_{t-1}) + \text{noise} \]
Models population of flies [R. May/1976]

Datasets

Logistic Parabola:
\[ x_t = ax_{t-1}(1-x_{t-1}) + \text{noise} \]
Models population of flies [R. May/1976]

Logistic Parabola

Our Prediction from here

Comparison of prediction to correct values
Datasets

LORENZ: Models convection currents in the air
\[ \frac{dx}{dt} = a (y - x) \]
\[ \frac{dy}{dt} = x (b - z) - y \]
\[ \frac{dz}{dt} = xy - c z \]

• Laser: fluctuations in a Laser over time (used in Santa Fe competition)

Laser

Comparison of prediction to correct values

Conclusions

• Lag plots for non-linear forecasting (Takens’ theorem)
• Suitable for ‘chaotic’ signals

References

References


Overall conclusions

- Similarity search: Euclidean/time-warping; feature extraction and SAMs
- Signal processing: DWT is a powerful tool
- Linear forecasting: AR (Box-Jenkins) methodology
- Bursty traffic: multifractals (80-20 ‘law’)