15-826: Multimedia Databases and Data Mining

Lecture #25: Time series mining and forecasting
Christos Faloutsos

Must-Read Material

- Chungmin Melvin Chen and Nick Roussopoulos, Adaptive Selectivity Estimation Using Query Feedbacks, SIGMOD 1994

Thanks

Deepay Chakrabarti (CMU)
Spiros Papadimitriou (CMU)
Prof. Byong-Kee Yi (Pohang U.)
Outline

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

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

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

Motivation - Applications (cont’d)

• civil/automobile infrastructure
  – bridge vibrations [Oppenheim+02]
  – road conditions / traffic monitoring

Motivation - Applications (cont’d)

• Weather, environment/anti-pollution
  – volcano monitoring
  – air/water pollutant monitoring
Motivation - Applications (cont’d)

• Computer systems
  – ‘Active Disks’ (buffering, prefetching)
  – web servers (ditto)
  – network traffic monitoring
  – ...

Stream Data: Disk accesses

Problem #1:

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

lynx caught per year
(packets per day; temperature per day)
Problem #2: Forecast

Given \( x_t, x_{t-1}, \ldots \), forecast \( x_{t+1} \)

Problem #2': Similarity search

E.g., Find a 3-tick pattern, similar to the last one

Problem #3:

- Given: A set of correlated time sequences
- Forecast ‘Sent(t)’
Important observations

Patterns, rules, forecasting and similarity indexing are closely related:
• To do forecasting, we need
  – to find patterns/rules
  – to find similar settings in the past
• To find outliers, we need to have forecasts
  – (outlier = too far away from our forecast)

Outline

• Motivation
• Similarity Search and Indexing
  • Linear Forecasting
  • Bursty traffic - fractals and multifractals
  • Non-linear forecasting
  • Conclusions

Outline

• Motivation
• Similarity search and distance functions
  – Euclidean
  – Time-warping
• …
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

\[ D(\vec{x}, \vec{y}) = \sum_{i=1}^{n} (x_i - y_i)^2 \]
\[ L_p(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i|^p \]

- \( L_1 \): city-block = Manhattan
- \( L_2 \): Euclidean
- \( L_\infty \)

Observation #1

- Time sequence -> n-d vector

\[
\begin{align*}
\text{Day-n} & \\
\ldots & \\
\text{Day-2} & \\
\text{Day-1} & \\
\end{align*}
\]
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

‘stutters’:
**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 

\[ x_1, x_2, \ldots, x_i; \quad y_1, y_2, \ldots, y_j \]

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

**Time warping**

VERY SIMILAR to the string-editing distance

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

- Complexity: $O(M*N)$ - quadratic on the length of the strings
- **Many** variations (penalty for stutters; limit on the number/percentage of stutters; …)
- popular in voice processing [Rabiner + Juang]

**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]

**Other Distance functions**

- In [Keogh+, KDD’04]: parameter-free, MDL based
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

Linear Forecasting
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

Reference

[Yi+00] Byoung-Kee Yi et al.: Online Data Mining for Co-Evolving Time Sequences, ICDE 2000. (Describes MUSCLES and Recursive Least Squares)
Problem#2: Forecast

• Example: give \( x_{t-1}, x_{t-2}, \ldots \), 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: \( x_{t-1}, x_{t-2}, \ldots \),
(up to a window of \( w \))

Formally:

\[ x_t \approx a_1 x_{t-1} + \ldots + a_w x_{t-w} + \text{noise} \]
(Problem: Back-cast; interpolate)

- Solution - interpolate: try to express \( x_t \) as a linear function of the past AND the future:
  \( x_{t+1}, x_{t+2}, \ldots, x_{t+\text{future}}; x_{t-1}, \ldots, x_{t-\text{past}} \)
  (up to windows of \( \text{w}_{\text{past}}, \text{w}_{\text{future}} \))

- EXACTLY the same algo's

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:
**Linear Auto Regression:**

- 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!
More details:

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

$X_{[N \times w]} \times a_{[w \times 1]} = y_{[N \times 1]}$

• 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:
• \( X_{[N \times w]} \times a_{[w \times 1]} = y_{[N \times 1]} \)

Ind-var1 \quad \text{Ind-var-w}

\begin{align*}
\begin{bmatrix}
X_{11}, & X_{12}, & \ldots, & X_{1w} \\
X_{21}, & X_{22}, & \ldots, & X_{2w} \\
\vdots & \vdots & \ddots & \vdots \\
X_{N1}, & X_{N2}, & \ldots, & X_{Nw}
\end{bmatrix}
\times
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_w
\end{bmatrix}
&=
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{bmatrix}
\end{align*}

Q2: How to estimate \( a_1, a_2, \ldots, a_w = a \)?

A2: with Least Squares fit

\[ a = (X^T \times X)^{-1} \times (X^T \times y) \]

(Moore-Penrose pseudo-inverse)

\( a \) is the vector that minimizes the RMSE from \( y \)

<identical math with ‘query feedbacks’>
More details

- Q2: How to estimate $a_1, a_2, \ldots, a_w = a$?
- A2: with Least Squares fit
  $$a = (X^T \times X)^{-1} \times (X^T \times y)$$
  Identical to earlier formula (proof?)
  $$a = V \times \Lambda^{-1} \times U^T \times y$$
  Where
  $$X = U \times \Lambda \times V^T$$

More details

- Straightforward solution:
  $$a = (X^T \times X)^{-1} \times (X^T \times y)$$
  - $a$: Regression Coeff. Vector
  - $X$: Sample Matrix

  - Observations:
    - Sample matrix $X$ grows over time
    - needs matrix inversion
    - $O(Nw^2)$ computation
    - $O(Nw)$ storage

Even more details

- Q3: Can we estimate $a$ 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?!)

---

CMU SCS

15-826
Even more details

- Q3: Can we estimate $a$ 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: $(X^T X)$

More details

At the $N+1$ time tick:

$$X_{N+1} \quad X_N \quad X_{N+1}$$

More details

- Let $G_N = (X_N^T X_N)^{-1}$ (‘gain matrix’)
- $G_{N+1}$ can be computed recursively from $G_N$
EVEN more details:

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

where \( I_{w \times w} \) is a row vector.

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

Let’s elaborate

(VERY IMPORTANT, VERY VALUABLE!)

EVEN more details:

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

\[ a = \begin{bmatrix} [w \times 1] & [(N+1) \times w] & [(N+1) \times 1] \\ [w \times (N+1)] & [w \times (N+1)] \end{bmatrix} \]
EVEN more details:

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

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

\[ \left[ w \times (N+1) \right] \]
EVEN more details:

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

**SCALAR!**

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

Altogether:

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

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

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

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

Altogether:

\[ G_0 = \delta I \quad \text{IMPORTANT!} \]

where

\[ I: \text{w x w identity matrix} \]
\[ \delta: \text{a large positive number (say, 10^4)} \]
Comparison:

- **Straightforward Least Squares**
  - Needs huge matrix, growing in size \( O(N \times w) \)
  - 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

\( N = 10^6, \ w = 1-100 \)

Pictorially:

- Given:

```
Independent Variable
```

```
Dependent Variable
```

Pictorially:

```
Independent Variable
```

```
Dependent Variable
```

\( \rightarrow \) new point
Pictorially:
RLS: quickly compute new best fit

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

Adaptability - ‘forgetting’
Adaptability - ‘forgetting’

- Independent Variable
  - e.g., #packets sent

- Dependent Variable
  - e.g., #bytes sent

Trend change

- (R)LS with no forgetting

- (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
Reference


Answer:

- ‘AWSOM’ (Arbitrary Window Stream fOrecasting 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")
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: $O(lgN + mk^2) \approx O(lgN)$
  - Time: $O(k^2) \approx O(1)$
- Where
  - $N$: number of points (so far)
  - $k$: number of regression coefficients; fixed
  - $m$: number of linear models; $O(lgN)$

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?

Solution:

Least Squares, with
- Dep. Variable: Repeated(t)
- Indep. Variables: Sent(t-1) … Sent(t-w); Lost(t-1) … Lost(t-w); Repeated(t-1), ...
- (named: ‘MUSCLES’ [Yi+00])

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; #packets per time unit)
  - 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”
Linear forecasting - 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]
- (AWSOM: no human intervention)

Resources: software and urls

- free-ware: ‘R’ for stat. analysis (clone of Splus)
  http://cran.r-project.org/
- python script for RLS
  http://www.cs.cmu.edu/~christos/SRC/rls-all.tar
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
Bursty Traffic & Multifractals

SKIP, if you have done HW2, Q3: Foils use D1 (‘information fractal dimension’), While HW2-Q3 uses D2 (‘correlation’ f.d.)

Outline

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


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
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

\[ \begin{array}{c}
20 \\
\wedge
\end{array} \begin{array}{c}
80 \\
\wedge
\end{array} \]

\[ \begin{array}{c}
100 \\
\wedge
\end{array} \begin{array}{c}
100 \\
\wedge
\end{array} \]

\[ \begin{array}{c}
0 \\
\wedge
\end{array} \begin{array}{c}
0 \\
\wedge
\end{array} \]

\[ \begin{array}{c}
50000 \\
\wedge
\end{array} \begin{array}{c}
80000 \\
\wedge
\end{array} \]

\[ \begin{array}{c}
100 \\
\wedge
\end{array} \begin{array}{c}
100 \\
\wedge
\end{array} \]

\[ \begin{array}{c}
0 \\
\wedge
\end{array} \begin{array}{c}
0 \\
\wedge
\end{array} \]

\[ \begin{array}{c}
50000 \\
\wedge
\end{array} \begin{array}{c}
80000 \\
\wedge
\end{array} \]
Parameter estimation

• Q2: How to estimate the bias factor $b$?

• A: MANY ways [Crovella+96]
  – Hurst exponent
  – variance plot
  – even DFT amplitude spectrum! (‘periodogram’)
  – More robust: ‘entropy plot’ [Wang+02]

Entropy plot

• 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 plot

- Entropy $E(n)$ after $n$ levels of splits
- $n=1$: $E(1) = - p_1 \log_2(p_1) - p_2 \log_2(p_2)$

Real traffic

- Has linear entropy plot ($\rightarrow$ self-similar)
Observation - intuition:

intuition: slope =
intrinsic dimensionality =
info-bits per coordinate-bit
– unif. Dataset: slope =?
– multi-point: slope = ?

# of levels (n)

Observation - intuition:

intuition: slope =
intrinsic dimensionality =
info-bits per coordinate-bit
– unif. Dataset: slope =1
– multi-point: slope = 0

# of levels (n)

Entropy plot - Intuition

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

Dim = 1

<table>
<thead>
<tr>
<th>Is MSB 0?</th>
<th>‘info’ value = E(1): 1 bit</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Is next MSB =0?</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Info value =1 bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>= E(2) - E(1) = slope!</td>
</tr>
<tr>
<td>Is next MSB =0?</td>
</tr>
</tbody>
</table>
Entropy plot

- Repeat, for all points at same position:
  - we need 0 bits of info, to determine position
  - \( \rightarrow \) slope = 0 = intrinsic dimensionality

\( \text{Dim} = 0 \)

Real (and 80-20) datasets can be in-between: bursts, gaps, smaller bursts, smaller gaps, at every scale

\( \text{Dim} = 1 \)
\( \text{Dim} = 0 \)
\( 0 < \text{Dim} < 1 \)
(Fractals, again)

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

Cantor dust

- Eliminate the middle third
- Recursively!
Cantor dust

_____________________

____________________

____________________

_____________________

____________________

____________________

____________________

____________________

____________________

____________________

____________________

____________________

____________________
Cantor dust

Dimensionality?
(no length; infinite # points!)
Answer: $\log_2 / \log_3 = 0.6$

Some more entropy plots:

• Poisson vs real
  
  Poisson: slope = 1 -> uniformly distributed

b-model

• b-model traffic gives perfectly linear plot
• Lemma: its slope is $slope = -b \log_2 b \cdot (1-b) \log_2 (1-b)$
• Fitting: do entropy plot; get slope; solve for $b$
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
  http://repository.cs.vt.edu/
  lbl-conn-7.tar.Z

Model validation

• Linear entropy plots

Bias factors $b$: 0.6–0.8
smallest $b$ / smoothest: nntp traffic
Web traffic - results

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

\[ \text{Prob}(>l) \]

(a) lbl-diff
(b) lbl-nntp
(c) lbl-sntp
(d) lbl-ftp

How to give guarantees?

\[ (\text{queue length } l) \]

20% of the requests will see queue lengths <100

Conclusions

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

- Fractals: Manfred Schroeder: *Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise*
  W.H. Freeman and Company, 1991 (Probably the BEST book on fractals!)

Further reading:

Outline

• Motivation
• ...
• Linear Forecasting
• Bursty traffic - fractals and multifractals
• Non-linear forecasting
• Conclusions
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, \(x_{t+1}, x_{t+2}, \ldots\)

Datasets

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

Lag-plot
ARIMA: fails
How to forecast?

• ARIMA - but: linearity assumption

Lag-plot
ARIMA: fails

How to forecast?

• ARIMA - but: linearity assumption

• ANSWER: ‘Delayed Coordinate Embedding’ = Lag Plots [Sauer92]
  ~ nearest-neighbor search, for past incidents

General Intuition (Lag Plot)

Interpolate these…
To get the final prediction

Lag = 1,
k = 4 NN

4-NN New Point

x_t

x_{t-1}
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])

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
\[
\begin{align*}
\frac{dH}{dt} &= rH - aHP \\
\frac{dP}{dt} &= bHP - mP
\end{align*}
\]

- 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, …).

But the delay vector space is a faithful reconstruction of the internal system state
- So prediction in delay vector space is as good as prediction in state space
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]

Lag-plot

ARIMA: fails
Datasets

LORENZ: Models convection currents in the air
dx/dt = a(y - x)
dy/dt = x(b - z) - y
dz/dt = xy - cz
Datasets

- LASER: fluctuations in a Laser over time (used in Santa Fe competition)
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; **AWSOM**
Overall conclusions

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

• Non-linear forecasting: lag-plots (Takens)