15-826: Multimedia Databases and Data Mining

Lecture #27: 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 (UT-Austin)
Spiros Papadimitriou (Rutgers)
Prof. Byoung-Kee Yi (Samsung)

Outline

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

- Given: one or more sequences $x_1, x_2, \ldots, x_t, \ldots$; $(y_1, y_2, \ldots, y_n, \ldots)$
- 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

- 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 $x_p$, $x_{t-1}$, ..., 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

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

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

- L₁: city-block = Manhattan
- L₂ = Euclidean
- Lₙ∞
Observation #1

• Time sequence -> n-d vector

Day-1
Day-2

Day-n

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 \( y \)

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

**Full-text scanning**

- Approximate matching - **string editing** distance:

\[ d(\text{‘survey’}, \text{‘surgery’}) = 2 \]

\[ = \min \text{# of insertions, deletions, substitutions to transform the first string into the second} \]

SURVEY

SURGERY

**Time warping**

Thus, with no penalty for stutter, for sequences

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

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

**Time warping**

VERY SIMILAR to the string-editing distance

\[ D(i, j) = \| x[i] - y[j] \| + \min \begin{cases} 
D(i-1, j-1) & \text{no stutter} \\
D(i, j-1) & \text{x-stutter} \\
D(i-1, j) & \text{y-stutter}
\end{cases} \]
Full-text scanning
if s[i] = t[j] then
   cost(i, j) = cost(i-1, j-1)
else
   cost(i, j) = min (1 + cost(i, j-1) // deletion
                     1 + cost(i-1, j-1) // substitution
                     1 + cost(i-1, j) // insertion)

Time warping
VERY SIMILAR to the string-editing distance

String editing
\[ D(i, j) = \| x[i] - y[j] \| + \min \left\{ \begin{array}{l} D(i-1, j-1) \\ D(i, j-1) \\ D(i-1, j) \end{array} \right\} \]

Time-warping
\[ cost(i, j) = \min \left\{ \begin{array}{l} 1 + cost(i-1, j-1) // sub. \\ 1 + cost(i, j-1) // del. \\ 1 + cost(i-1, j) // ins. \end{array} \right\} \]

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

(Describes MUSCLES and Recursive Least Squares)

Problem#2: Forecast

• Example: give $x_{t-1}$, $x_{t-2}$, ..., 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:
\[
x_t \approx a_1 x_{t-1} + \ldots + a_w x_{t-w} + \text{noise}
\]

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

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>63</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>97</td>
</tr>
</tbody>
</table>

- lag \( w=1 \)
- Dependent variable = # of packets sent \( S[t] \)
- Independent variable = # of packets sent \( S[t-1] \)

'lag-plot'

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!)

\[ x_{t-2} \text{ } x_{t-1} \text{ } x_t \]

More details:

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

\[ x_{t-2} \text{ } x_{t-1} \text{ } x_t \]

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:

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

Ind-var1 Ind-var-w
time

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

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 \( y \)

\(<\text{identical} \text{ math with ‘query feedbacks’}>\)

More details

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

A2: with Least Squares fit

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

(Moore-Penrose pseudo-inverse)

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

\(<\text{identical} \text{ math with ‘query feedbacks’}>\)

More details

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

A2: with Least Squares fit

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

(Moore-Penrose pseudo-inverse)

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

\(<\text{identical} \text{ math with ‘query feedbacks’}>\)
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?!)

More details

At the $N+1$ time tick:

\[
x_{N+1} \quad x_N \quad x_{N+1}
\]

\[ \begin{array}{c}
\text{w} \\
\text{w} \\
\text{w} \\
\end{array} \]

- Let $G_N = (X_N^T \times X_N)^{-1}$ (`gain matrix`)
- $G_{N+1}$ can be computed recursively from $G_N$
EVEN more details:

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

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

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

Let’s elaborate

(VERY IMPORTANT, VERY VALUABLE!)

EVEN more details:

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

\[ [w \times 1] \hspace{1cm} [(N+1) \times w] \hspace{1cm} [(N+1) \times 1] \]

\[ [w \times (N+1)] \hspace{1cm} [w \times (N+1)] \]

EVEN more details:

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

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

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

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

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

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

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

Altogether:

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

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

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

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

\[ G_0 = \delta I \]

where

I: \( w \times w \) identity matrix
\( \delta \): a large positive number (say, \( 10^4 \))

IMPORTANT!

Comparison:

- **Straightforward Least Squares**
  - Needs huge matrix (growing in size) \( O(N \times w) \)
  - Costly matrix operation \( O(N \times w^2) \)

- **Recursive LS**
  - Need much smaller, fixed size matrix \( O(w \times w) \)
  - Fast, incremental computation \( O(1 \times w^2) \)
  - no matrix inversion

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

Pictorially:

- **Given:**

[Graph showing linear relationship between independent and dependent variables]

- **New point**

[Graph showing linear relationship with a new data 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’
Adaptability - ‘forgetting’

- RLS: can *trivially* handle ‘forgetting’

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”

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)

Linear forecasting - Outline

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

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
Outline

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

Recall: Problem #1:

Goal: given a signal (eg., #bytes over time)
Find: patterns, periodicities, and/or compress

#bytes

Bytes per 30’
(packets per day; earthquakes per year)

time

Reference:


Full thesis: CMU-CS-05-185
Performance Modeling of Storage Devices using Machine Learning Mengzhi Wang, Ph.D. Thesis
Abstract, .ps.gz, .pdf

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

**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 (eg., 80%)

Binary multifractals

Could you use IFS?
To generate such traffic?
Could you use IFS?
To generate such traffic?
A: Yes – which transformations?

A:
\[ x' = x / 2 \quad (p = 0.2) \]
\[ x' = x / 2 + 0.5 \quad (p = 0.8) \]

Parameter estimation
• Q2: How to estimate the bias factor \( b \)?
• A: MANY ways [Crovella+96]
  – Hurst exponent
  – variance plot
  – even DFT amplitude spectrum! (‘periodogram’)
  – Fractal dimension (D2)
    • Or D1 (‘entropy plot’ [Wang+02])
**Fractal dimension**

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

- \( \text{Dim} = 1 \)
- \( \text{Dim} = 0 \)
- \( 0 < \text{Dim} < 1 \)

---

**Estimating ‘b’**

- **Exercise:** Show that

\[
D_2 = - \log_2 \left( b^2 + (1-b)^2 \right)
\]

Sanity checks:
- \( b = 1.0 \quad D_2 = ?? \)
- \( b = 0.5 \quad D_2 = ?? \)

---

**Fractals, again**

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

---

**Cantor dust**

- Eliminate the middle third
- Recursively!
Cantor dust

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

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:

• Crovella, M. and A. Bestavros (1996). Self-
  Similarity in World Wide Web Traffic, Evidence
  and Possible Causes. Sigmetrics.
• [ieeeTN94] W. E. Leland, M.S. Taqqu, W.
  Willinger, D.V. Wilson, *On the Self-Similar
  Nature of Ethernet Traffic*, IEEE Transactions on
Further reading


Outline

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

Reference:

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:
\[ x_t = ax_{t-1}(1-x_{t-1}) + \text{noise} \]
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

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?)

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, …).

Theoretical foundation

- 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]

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

Logistic Parabola

Comparison of prediction to correct values

Our Prediction from here

Lag-plot

Lag-plot

ARIMA: fails

Timesteps

Value

Timesteps
Datasets

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

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

Comparison of prediction to correct values
Conclusions

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

References


Overall conclusions

• Similarity search: Euclidean/time-warping; feature extraction and SAMs
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’)

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)