Indexing and Mining Time Sequences

Christos Faloutsos and Lei Li
CMU
Outline

• Motivation
• Similarity Search and Indexing
• DSP (Digital Signal Processing)
• Linear Forecasting
• Kalman filters
• fractals and multifractals
• Non-linear forecasting
• Conclusions
Problem definition

- **Given**: one or more sequences
  \[ x_1, x_2, \ldots, x_t, \ldots \]
  \[ (y_1, y_2, \ldots, y_t, \ldots) \]

- **Find**
  - similar sequences; forecasts
  - patterns; clusters; outliers
Motivation - Applications

- Financial, sales, economic series
- Medical
  - reactions to new drugs
  - elderly care
ECG - physionet.org
EEG - epilepsy

from wikipedia
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

![Disk traffic graph]

- #bytes
- time

200000000
150000000
100000000
50000000
0
Problem #1:

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

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

Given $x_t$, $x_{t-1}$, ..., forecast $x_{t+1}$
Problem#2’: Similarity search

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

![Graph showing number of packets sent over time ticks 1 to 11]
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)
Important topics NOT in this tutorial:

- Continuous queries
  - [Babu+Widom] [Gehrke+] [Madden+]
- Categorical data streams
  - [Hatonen+96]
- Outlier detection (discontinuities)
  - [Breunig+00]
Outline

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

• Motivation
• Similarity Search and Indexing
  – distance functions: Euclidean; Time-warping
  – indexing
  – feature extraction
• DSP
• ...
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

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

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

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

\[ \sum \text{diagram of } x(t) \text{ and } y(t) \]
Observation #1

- Time sequence -> n-d vector

Day-n

Day-2

Day-1
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’: 

\[ \text{Diagram showing time warping with red and blue lines, and arrows indicating 'stutters'.} \]
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 \(y\)
Time Warping

Thus, with no penalty for stutter, for sequences

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

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

• Chen + Ng [vldb’04]: ERP ‘Edit distance with Real Penalty’: give a penalty to stutters

• Keogh+ [kdd’04]: VERY NICE, based on information theory: compress each sequence (quantize + Lempel-Ziv), using the other sequences’ LZ tables

On The Marriage of $L_p$-norms and Edit Distance, Lei Chen, Raymond T. Ng; VLDB’04

Towards Parameter-Free Data Mining, E. Keogh, S. Lonardi, C.A. Ratanamahatana, KDD’04
Conclusions

Prevailing distances:

– Euclidean and

– time-warping
Outline

• Motivation
• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
• DSP
• ...

Indexing

Problem:

• given a set of time sequences,
• find the ones similar to a desirable query sequence
distance function: by expert
Idea: ‘GEMINI’

Eg., ‘find stocks similar to MSFT’
Seq. scanning: too slow
How to accelerate the search?

[Faloutsos96]
‘GEMINI’ - Pictorially

\[ S_1 \]

\[ S_n \]

1 day

365 day

1 day

\[ F(S_1) \]

\[ F(S_n) \]

eg., std

eg., avg
GEMINI

Solution: Quick-and-dirty' filter:

• extract $n$ features (numbers, eg., avg., etc.)
• map into a point in $n$-d feature space
• organize points with off-the-shelf spatial access method (‘SAM’)
• discard false alarms
Examples of GEMINI

• Time sequences: DFT (up to 100 times faster) [SIGMOD94];
• [Kanellakis+], [Mendelzon+]
Indexing - SAMs

Q: How do Spatial Access Methods (SAMs) work?

A: they group nearby points (or regions) together, on nearby disk pages, and answer spatial queries quickly (‘range queries’, ‘nearest neighbor’ queries etc)

For example:
R-trees

- [Guttman84] eg., w/ fanout 4: group nearby rectangles to parent MBRs; each group -> disk page
R-trees

- eg., w/ fanout 4:
R-trees

- eg., w/ fanout 4:
R-trees - range search?
R-trees - range search?
Conclusions

• Fast indexing: through GEMINI
  – feature extraction and
  – (off the shelf) Spatial Access Methods [Gaede+98]
Outline

• Motivation
• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
• DSP
• ...

(C) 2010, C. Faloutsos, Lei Li
Outline

• Motivation

• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
    • DFT, DWT, DCT (data independent)
    • SVD, etc (data dependent)
    • MDS, FastMap
DFT and cousins

• very good for compressing real signals
• more details on DFT/DCT/DWT: later
DFT and stocks

- Dow Jones Industrial index, 6/18/2001-12/21/2001
DFT and stocks

- Dow Jones Industrial index, 6/18/2001-12/21/2001
- just 3 DFT coefficients give very good approximation
Outline

• Motivation

• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
    • DFT, DWT, DCT (data independent)
    • SVD etc (data dependent)
    • MDS, FastMap
SVD

• **THE optimal method for dimensionality reduction**
  – (under the Euclidean metric)
Singular Value Decomposition (SVD)

- SVD (~LSI ~ KL ~ PCA ~ spectral analysis...)

Details: [Press+], [Faloutsos96]
SVD

• **Extremely** useful tool
  – (also behind PageRank/google and Kleinberg’s algorithm for hubs and authorities)

• But may be slow: $O(N \times M \times M)$ if $N > M$

• any approximate, faster method?
SVD shortcuts

- random projections (Johnson-Lindenstrauss thm [Papadimitriou+ pods98])
Random projections

• pick ‘enough’ random directions (will be ~orthogonal, in high-d!!)
• distances are preserved probabilistically, within epsilon
• (also, use as a pre-processing step for SVD [Papadimitriou+ PODS98])
Outline

• Motivation

• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
    • DFT, DWT, DCT (data independent)
    • SVD etc (data dependent), ICA
    • MDS, FastMap
Citation

• **AutoSplit: Fast and Scalable Discovery of Hidden Variables in Stream and Multimedia Databases**, Jia-Yu Pan, Hiroyuki Kitagawa, Christos Faloutsos and Masafumi Hamamoto

PAKDD 2004, Sydney, Australia
Motivation:
(Q1) Find patterns in data

- Motion capture data (broad jumps)
PCA sometimes misses essential features

- Best SVD axis: not always meaningful!
Motivation:
(Q1) Find patterns in data

- Human would say
  - Pattern 1: along diagonal
  - Pattern 2: along vertical axis
- How to find these automatically?
Motivation:
(Q2) Find hidden variables

Find common hidden variables, and weights.

Dow Jones Industrial Average
Motivation:

(Q2) Find hidden variables

Caterpillar

Intel

B<sub>1</sub>,CAT
B<sub>1</sub>,INTC

B<sub>2</sub>,CAT
B<sub>2</sub>,INTC

Hidden variable 1

Hidden variable 2

KDD 2010
(c) 2010, C. Faloutsos, Lei Li
Motivation:
(Q2) Find hidden variables

Caterpillar

0.94

0.63

Intel

0.64

0.03

“Hidden variable 1”

“Hidden variable 2”
Motivation:
(Q2) Find hidden variables

![Graph showing trends and correlations between Caterpillar (CAT) and Intel (INTC).](image)

“General trend”

“Internet bubble”
Outline

• Motivation
• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
    • DFT, DWT, DCT (data independent)
    • SVD (data dependent)
    • MDS, FastMap
MDS / FastMap

• but, what if we have NO points to start with?
  (eg. Time-warping distance)
• A: Multi-dimensional Scaling (MDS) ;
  FastMap
MDS/FastMap

\[
\begin{array}{ccccc}
O1 & O2 & O3 & O4 & O5 \\
O1 & 0 & 1 & 1 & 100 & 100 \\
O2 & 1 & 0 & 1 & 100 & 100 \\
O3 & 1 & 1 & 0 & 100 & 100 \\
O4 & 100 & 100 & 100 & 0 & 1 \\
O5 & 100 & 100 & 100 & 1 & 0 \\
\end{array}
\]
Multi Dimensional Scaling

MDS
FastMap

• Multi-dimensional scaling (MDS) can do that, but in $O(N^{**2})$ time
• FastMap [Faloutsos+95] takes $O(N)$ time
FastMap: Application

VideoTrails [Kobla+97]

scene-cut detection (about 10% errors)
Outline

- Motivation
- Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
    - DFT, DWT, DCT (data independent)
    - SVD (data dependent)
    - MDS, FastMap, IsoMap etc
Variations

• Isomap [Tenenbaum, de Silva, Langford, 2000]
• LLE (Local Linear Embedding) [Roweis, Saul, 2000]
• MVE (Minimum Volume Embedding) [Shaw & Jebara, 2007]
Variations

• Isomap [Tenenbaum, de Silva, Langford, 2000]

• LLE (Local Linear Embedding) [Roweis, Saul, 2000]

• MVE (Minimum Volume Embedding) [Shaw & Jebara, 2007]
Outline

• Motivation
• Similarity Search and Indexing
  – distance functions
  – indexing
  – feature extraction
    • DFT, DWT, DCT (data independent)
    • SVD (data dependent)
    • MDS, FastMap
Conclusions - Practitioner’s guide

Similarity search in time sequences

1) establish/choose distance (Euclidean, time-warping, …)

2) extract features (SVD, DWT, MDS), and use an SAM (R-tree/variant) or a Metric Tree (M-tree)

2’) for high intrinsic dimensionalities, consider sequential scan (it might win…)
Books

References


• Berry, Michael: http://www.cs.utk.edu/~lsi/
References


References


References


References


References


References

• Oppenheim, I. J., A. Jain, et al. (March 2002). A MEMS Ultrasonic Transducer for Resident Monitoring of Steel Structures. SPIE Smart Structures Conference SS05, San Diego.


References

• Traina, C., A. Traina, et al. (October 2000). Fast feature selection using the fractal dimension,. XV Brazilian Symposium on Databases (SBBD), Paraiba, Brazil.
References

• Dennis Shasha and Yunyue Zhu *High Performance Discovery in Time Series: Techniques and Case Studies* Springer 2004


• Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. *The Design of an Acquisitional Query Processor for Sensor Networks*. SIGMOD, June 2003, San Diego, CA.
References

• Lawrence Saul & Sam Roweis. *An Introduction to Locally Linear Embedding* (draft)


References

Part 2: DSP (Digital Signal Processing)
Outline

• Motivation
• Similarity Search and Indexing
• DSP (DFT, DWT)
• Linear Forecasting
• Kalman filters
• Bursty traffic - fractals and multifractals
• Non-linear forecasting
• Conclusions
Outline

• DFT
  – Definition of DFT and properties
  – how to read the DFT spectrum

• DWT
  – Definition of DWT and properties
  – how to read the DWT scalogram
Introduction - Problem #1

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

- lynx caught per year
  (packets per day; automobiles per hour)
What does DFT do?

A: highlights the periodicities
DFT: definition

• For a sequence $x_0, x_1, \ldots x_{n-1}$
• the (\textbf{n-point}) Discrete Fourier Transform is
• $X_0, X_1, \ldots X_{n-1}$:

\begin{align*}
X_f &= 1 / \sqrt{n} \sum_{t=0}^{n-1} x_t \times \exp(-j 2\pi tf / n) \quad f = 0, \ldots, n-1 \\
(j &= \sqrt{-1}) \\
x_t &= 1 / \sqrt{n} \sum_{t=0}^{n-1} X_f \times \exp(+j 2\pi tf / n)
\end{align*}
DFT: definition

- **Good news**: Available in **all** symbolic math packages, e.g., in ‘mathematica’

  ```
  x = [1,2,1,2];
  X = Fourier[x];
  Plot[ Abs[X] ];
  ```
DFT: Amplitude spectrum

Amplitude: \[ A_f^2 = \text{Re}^2(X_f) + \text{Im}^2(X_f) \]
DFT: examples

flat

Amplitude

time

freq
DFT: examples

Low frequency sinusoid

![Graph showing a low frequency sinusoid in the time and frequency domains.](attachment:image.png)
DFT: examples

- Sinusoid - symmetry property: $X_f = X^*_{n-f}$
DFT: examples

• Higher freq. sinusoid
DFT: examples

\[ \text{examples} \]

\[ = \]

\[ + \]

\[ + \]
DFT: examples

examples

Ampl.

Freq.
Outline

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

• Motivation
• Similarity Search and Indexing
• DSP
  – DFT
    • Definition of DFT and properties
    • how to read the DFT spectrum
  – DWT
DFT: Amplitude spectrum

Amplitude:  \[ A_f^2 = \text{Re}^2 (X_f) + \text{Im}^2 (X_f) \]
DFT: Amplitude spectrum

count

Ampl.

year

Freq.

freq=0

freq=12
DFT: Amplitude spectrum

- **X-axis (Year)**
- **Y-axis (Count)**
- **Amplitude (Ampl.)**
- **Frequency (Freq.)**

- **Legend**:
  - Actual
  - Mean
  - Mean + freq=12

- **Key Points**:
  - freq=0
  - freq=12

---

KDD 2010        (c) 2010, C. Faloutsos, Lei Li 121
DFT: Amplitude spectrum

• excellent approximation, with only 2 frequencies!

• so what?
DFT: Amplitude spectrum

• excellent approximation, with only 2 frequencies!
• so what?
• A1: (lossy) compression
• A2: pattern discovery
DFT: Amplitude spectrum

- excellent approximation, with only 2 frequencies!
- so what?
- A1: (lossy) compression
- A2: pattern discovery
DFT - Conclusions

- It spots periodicities (with the ‘amplitude spectrum’)
- can be quickly computed ($O(n \log n)$), thanks to the FFT algorithm.
- *standard* tool in signal processing (speech, image etc signals)
- (closely related to DCT and JPEG)
Outline

• Motivation
• Similarity Search and Indexing
• DSP
  – DFT
  – DWT
  • Definition of DWT and properties
  • how to read the DWT scalogram
Problem #1:

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

- lynx caught per year
- (packets per day; virus infections per month)

![Graph showing lynx trapping count over time](chart)

KDD 2010
Wavelets - DWT

- DFT is great - but, how about compressing a spike?

![Graph showing a spike in time series data]
Wavelets - DWT

- DFT is great - but, how about compressing a spike?
- A: Terrible - all DFT coefficients needed!
Wavelets - DWT

- DFT is great - but, how about compressing a spike?
- A: Terrible - all DFT coefficients needed!
Wavelets - DWT

• Similarly, DFT suffers on short-duration waves (eg., baritone, silence, soprano)
Wavelets - DWT

- Solution#1: Short window Fourier transform (SWFT)
- But: how short should be the window?
Wavelets - DWT

- **Answer:** *multiple window sizes!* -> DWT

- **Time domain**
  - **DFT**
  - **SWFT**
  - **DWT**

freq

↑

<table>
<thead>
<tr>
<th>freq</th>
<th>DFT</th>
<th>SWFT</th>
<th>DWT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

↑

time
Haar Wavelets

- subtract sum of left half from right half
- repeat recursively for quarters, eight-ths, ...
Wavelets - construction

x0  x1  x2  x3  x4  x5  x6  x7
Wavelets - construction

level 1  d1,0  s1,0  d1,1  s1,1  ......  
    / + \
   /    \  
  x0   x1   x2   x3   x4   x5   x6   x7
Wavelets - construction

level 2

\( d_{2,0} \quad \text{s2,0} \)

\( d_{1,0} \quad \text{s1,0} \quad d_{1,1} \quad \text{s1,1} \quad \ldots \)

\( x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \)
Wavelets - construction

e tc ...

d2,0

s2,0

s1,0 d1,1 s1,1

x0 x1 x2 x3 x4 x5 x6 x7

d1,0

+ −
Wavelets - construction

Q: map each coefficient on the time-freq. plane

\[ \begin{align*}
&d_{2,0} \\
&s_{2,0} \\
&d_{1,0} \\
&s_{1,0} \quad d_{1,1} \quad s_{1,1} \\
&x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7
\end{align*} \]
Wavelets - construction

Q: map each coefficient on the time-freq. plane

\[ \begin{align*}
  x_0 & \quad x_1 & \quad x_2 & \quad x_3 & \quad x_4 & \quad x_5 & \quad x_6 & \quad x_7 \\
  d_{1,0} & \quad s_{1,0} & \quad d_{1,1} & \quad s_{1,1} & \quad \cdots \\
  d_{2,0} & \quad s_{2,0} \\
\end{align*} \]
#!/usr/bin/perl

# expects a file with numbers
# and prints the dwt transform
# The number of time-ticks should be a power of 2
# USAGE
#   haar.pl <fname>

my @vals=();
my @smooth; # the smooth component of the signal
my @diff;   # the high-freq. component

# collect the values into the array @val
while(<>){
    @vals = ( @vals, split );
}

my $len = scalar(@vals);
my $half = int($len/2);
while($half >= 1 ){  
    for(my $i=0; $i< $half; $i++){
        $diff [$i] = ($vals[2*$i] - $vals[2*$i + 1] )/ sqrt(2);
        print "t", $diff[$i];
        $smooth [$i] = ($vals[2*$i] + $vals[2*$i + 1] )/ sqrt(2);
    }
    print "n";
    @vals = @smooth;
    $half = int($half/2);
}

print "t", $vals[0], "n" ;  # the final, smooth component
Wavelets - construction

Observation 1:
‘+’ can be some weighted addition
‘-’ is the corresponding weighted difference
(‘Quadrature mirror filters’)

Observation 2: unlike DFT/DCT,
there are *many* wavelet bases: Haar, Daubechies-4, Daubechies-6, Coifman, Morlet, Gabor, ...
Wavelets - how do they look like?

- E.g., Daubechies-4
Wavelets - how do they look like?

- E.g., Daubechies-4
Wavelets - how do they look like?

- E.g., Daubechies-4
Outline

• Motivation
• Similarity Search and Indexing
• DSP
  – DFT
  – DWT
    • Definition of DWT and properties
    • how to read the DWT scalogram
Wavelets - Drill#1:

- Q: baritone/silence/soprano - DWT?
Wavelets - Drill#1:

• Q: baritone/silence/soprano - DWT?
Wavelets - Drill#2:

• Q: spike - DWT?

\[ f(t) \]

\[ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \]
Wavelets - Drill#2:

- Q: spike - DWT?

\[
\begin{array}{cccc}
& & & \downarrow \\
\vdots & \vdots & \vdots & \vdots \\
f & & & t \\
\hline
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
0.00 & 0.00 & 0.71 & 0.00 \\
0.00 & 0.50 & \text{-0.35} & 0.35 \\
\end{array}
\]
Wavelets - Drill#3:

- Q: weekly + daily periodicity, + spike - DWT?

![Waveform Diagram]
Wavelets - Drill#3:

• Q: weekly + daily periodicity, + spike - DWT?

![Waveform Diagram]

\[ f(t) \]
Wavelets - Drill#3:

Q: weekly + daily periodicity, + spike - DWT?
Wavelets - Drill#3:

- Q: weekly + daily periodicity, + spike - DWT?
Wavelets - Drill#3:

- Q: weekly + daily periodicity, + spike - DWT?
Wavelets - Drill#3:

• Q: DFT?
Advantages of Wavelets

- Better compression (better RMSE with same number of coefficients - used in JPEG-2000)
- Fast to compute (usually: $O(n)$!)
- Very good for ‘spikes’
- Mammalian eye and ear: Gabor wavelets
Overall Conclusions

• DFT, DCT spot periodicities
• **DWT**: multi-resolution - matches processing of mammalian ear/eye better
• All three: powerful tools for **compression**, **pattern detection** in real signals
• All three: included in math packages
  – (matlab, ‘R’, mathematica, … - often in spreadsheets!)
Overall Conclusions

- DWT: very suitable for self-similar traffic
- DWT: used for summarization of streams
  [Gilbert+01], db histograms etc
Resources - software and urls

Resources: software and urls

- *xwpl*: open source wavelet package from Yale, with excellent GUI
- [http://monet.me.ic.ac.uk/people/gavin/java/waveletDemos.html](http://monet.me.ic.ac.uk/people/gavin/java/waveletDemos.html): wavelets and scalograms
Books


Additional Reading

Part 3: Linear Forecasting
Outline

• Motivation
• Similarity Search and Indexing
• DSP
• Linear Forecasting
• Kalman filters
• 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 $x_{t-1}$, $x_{t-2}$, ..., 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: \( x_{t-2}, x_{t-2}, \ldots \)

(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:

$$x_{t+1}, x_{t+2}, \ldots x_{t+w_{future}}; x_{t-1}, \ldots x_{t-w_{past}}$$

(up to windows of $$w_{past}, w_{future}$$)

• 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)’)

<table>
<thead>
<tr>
<th>patient</th>
<th>weight</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

Body weight

Body height
**Linear Auto Regression:**

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent(t)</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>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>??</td>
</tr>
</tbody>
</table>
Linear Auto Regression:

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

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent ( t-1 )</th>
<th>Packets Sent ( t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

‘lag-plot’

Number of packets sent \( t \) vs. Number of packets sent \( t-1 \)
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!)
More details:

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

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

Ind-var1 \hspace{1cm} Ind-var-w

time

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

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

Ind-\( \text{var1} \) \hspace{1cm} \text{Ind-\( \text{var-w} \)}

time

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

- Q2: How to estimate $a_1, a_2, \ldots, a_w = \mathbf{a}$?
- A2: with Least Squares fit
  \[
  \mathbf{a} = (X^T \times X)^{-1} \times (X^T \times \mathbf{y})
  \]
- (Moore-Penrose pseudo-inverse)
- $\mathbf{a}$ is the vector that minimizes the RMSE from $\mathbf{y}$
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) - pictorially:
Even more details

• Given:

![Graph showing a scatter plot with a linear trend line between independent and dependent variables.]

Dependent Variable

Independent Variable
Even more details

new point
Even more details

RLS: quickly compute new best fit

Even more details

RLS: quickly compute new best fit

new point
Even more details

• **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
    \[ O(1 \times w^2) \]

\[
N = 10^6, \quad w = 1-100
\]
Even more details

• Q4: can we ‘forget’ the older samples?
• A4: Yes - RLS can easily handle that

[Yi+00]:
Adaptability - ‘forgetting’

- Independent Variable: eg., #packets sent
- Dependent Variable: eg., #bytes sent
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])
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]
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)
BREAK!

Next: Kalman filters
Indexing and Mining Time Sequences
Part 4
Kalman Filters

Christos Faloutsos
Lei Li
CMU
Outline

• Intuition, example, and definition
• Extensions
• Kalman filters at work
Intuition

• Tracking moving objects, estimate velocity and acceleration on the fly

from FIFA 2010

RoboCup 2010

KDD 2010

Copyright: C. Faloutsos & L. Li, 2010
Linear Dynamical System

- **Known parameters**
  - Original Kalman Filters [Kalman 1960, Rauch 1965]

- **Unknown parameters**
  - Parameter estimation through EM algorithm [Shumway et al 1982, Ghahramani 1996]
Kalman Filters

Given observations of the soccer ball position \( t=1..T \), “Model parameters”

Goal: two types of prediction

Kalman filtering: Estimate the true position, velocity & acceleration based on the previous observations

Kalman smoothing: Estimate for every time tick, based on all observations
Kalman Filters (intuition)

t=1, soccer with initial pos, vel and acc.
To estimate the future
Kalman Filters (intuition)

t=2, according to Newton’s law, it should be...
Kalman Filters (intuition)

t=2, according to Newton’s law, it should be... however, imperfect soccer/kick movement…
Kalman Filters (intuition)

Now take a photo, due to imperfect camera...
Kalman Filters (intuition)

What is the best estimate for next time tick?
Kalman Filters (intuition)

What is the best estimate for next time tick?

\[ a_1 \]

\[ v_1 \]

\[ p_1 \]

\[ \hat{a}_2 \]

\[ \hat{v}_2 \]

\[ \hat{p}_2 \]
Some math notation

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>transition matrix</td>
</tr>
<tr>
<td>C</td>
<td>transmission/ projection/ output matrix</td>
</tr>
<tr>
<td>Q</td>
<td>transition covariance</td>
</tr>
<tr>
<td>R</td>
<td>transmission/ projection/ output covariance</td>
</tr>
</tbody>
</table>

‘Newton’s dynamics’:

\[
\begin{align*}
    v_2 &= A v_1 + \hat{a}_2 \hat{v}_2 \\
    \hat{v}_2 &= C \hat{a}_2 \\
    \hat{a}_2 &= Q \hat{v}_1 \\
    \hat{v}_1 &= R \hat{a}_1
\end{align*}
\]
Example

hidden states  \( z_1 = (p_1, v_1, a_1)^T \)

observation  \( x_1 = (\text{observed}_1) \)

transition matrix  \( A = \begin{pmatrix} 1,1,1/2 \\ 0,1,1 \\ 0,0,1 \end{pmatrix} \)

transition covariance  \( Q \)

output matrix  \( C = (1,0,0) \)

output covariance  \( R \)
Example

hidden states \( z_1 = (p_1, v_1, a_1)^T \)

observation \( x_1 = (\text{observed}_1) \)

transition matrix \( A = \begin{pmatrix} 1, 1, 1/2 \\ 0, 1, 1 \\ 0, 0, 1 \end{pmatrix} \)

output matrix \( C = (1, 0, 0) \)

\[ p_2 = p_1 + v_1 \Delta t + 0.5a_1 \Delta t^2 \]
\[ v_2 = v_1 + a_1 \Delta t \]
\[ a_2 = a_1 \]
Kalman Filtering (intuition)

Step 1, forecast next time tick before observation

‘Newton’s dynamics’:

Transition matrix $A$
Kalman Filtering (intuition)

Step 2: adjust estimation after observation

\[ \text{Diagram: } a_1, v_1, p_1, \hat{a}_2, \hat{v}_2, \hat{p}_2 \]
Example: Kalman filtering (forward)

Given:
a sequence of observations, Model parameters (A, C …)
Goal: remove noise and forecast real position

Position

Time

observed
Example: Kalman filtering (forward)

Position

observed

estimated

t=1

Time

Kalman. A new approach to linear filtering and prediction problems. 1960]
Example: Kalman filtering

\[
\hat{z}_n = A \cdot \hat{z}_{n-1} + K_n \cdot (x_n - C \cdot A \cdot \hat{z}_{n-1})
\]

\[
\hat{V}_n = (I - K_n) \cdot P_{n-1}
\]

\[
K_n = P_{n-1} \cdot C^T \cdot (C \cdot P_{n-1} \cdot C^T + R)^{-1}
\]

\[
P_{n-1} = A \cdot \hat{V}_{n-1} \cdot A^T + Q
\]

Intuition: #2 may be close to #1
Using the “Newton Dynamics”
Example: Kalman filtering

\[ \hat{z}_n = A \cdot \hat{z}_{n-1} + K_n \cdot (x_n - C \cdot A \cdot \hat{z}_{n-1}) \]

\[ \hat{V}_n = (I - K_n) \cdot P_{n-1} \]

\[ K_n = P_{n-1} \cdot C^T \cdot (C \cdot P_{n-1} \cdot C^T + R)^{-1} \]

\[ P_{n-1} = A \cdot \hat{V}_{n-1} \cdot A^T + Q \]
Kalman Filters

Given observations of the soccer position \( t=1..T \), and model parameters \( (A, C \ldots) \)

Goal: two types of prediction

Kalman filtering: Estimate the true position, velocity & acceleration based on the previous observations

Kalman smoothing: Estimate for every time tick, based on all observations
Kalman Smoothing

Given: all observation $x_1, \ldots, x_n$

Estimate: the hidden state for every time tick $z_t (t=1..n)$

Difference from Kalman filtering:

bring future observation back in history estimate
Recap: Kalman filtering

\[ \hat{z}_n = A \cdot \hat{z}_{n-1} + K_n \cdot (x_n - C \cdot A \cdot \hat{z}_{n-1}) \]

\[ \hat{V}_n = (I - K_n) \cdot P_{n-1} \]

\[ K_n = P_{n-1} \cdot C^T \cdot (C \cdot P_{n-1} \cdot C^T + R)^{-1} \]

\[ P_{n-1} = A \cdot \hat{V}_{n-1} \cdot A^T + Q \]
Example: Kalman Smoothing

\[
\hat{z}_n = \hat{z}_n + J_n \cdot (\hat{z}_{n+1} - A \cdot \hat{z}_n)
\]
\[
\hat{V}_n = \hat{V}_n + J_n \cdot (\hat{V}_{n+1} - P_n) \cdot J_n^T
\]
\[
J_n = \hat{V}_n \cdot A^T \cdot P_n^{-1}
\]
Example: Kalman Smoothing

\[ \hat{Z}_n = \hat{z}_n + J_n \cdot (\hat{z}_{n+1} - A \cdot \hat{z}_n) \]

\[ \hat{V}_n = \hat{V}_n + J_n \cdot (\hat{V}_{n+1} - P_n) \cdot J_n^T \]

\[ J_n = \hat{V}_n \cdot A^T \cdot P_n^{-1} \]

Backward

Position

estimated

observed

Time

KDD 2010

Copyright: C. Faloutsos & L. Li, 2010
Example: Kalman Smoothing

Backward

Position

estimated
observed

Reconstructed signal after smoothing

Time

KDD 2010
Copyright: C. Faloutsos & L. Li, 2010
Outline

• Intuition, example, and definition
  – Original Kalman
  – Kalman filters with parameter estimation
• Extensions
• Kalman filters at work
What if not know the model parameters?

- E.g. Datacenter sensor temperatures
- no longer “Newton dynamics”

Transition matrix $A = \cdots$
output matrix $C = \cdots$
Graphical Model Representation

Model parameters:
\[ \theta = \{ \mu_0, Q_0, A, Q, C, R \} \]

\[ z_1 = \mu_0 + \omega_0 \]
\[ z_{n+1} = A \cdot z_n + \omega_n \]
\[ x_n = C \cdot z_n + \varepsilon_n \]
## Linear Dynamical Systems: parameters

<table>
<thead>
<tr>
<th>name</th>
<th>meaning &amp; example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_0$</td>
<td>initial state for hidden variable</td>
</tr>
<tr>
<td></td>
<td>e.g. initial position, velocity &amp; acceleration</td>
</tr>
<tr>
<td>$A$</td>
<td>transition matrix</td>
</tr>
<tr>
<td></td>
<td>how the states move forward, e.g. soccer flying in the air</td>
</tr>
<tr>
<td>$C$</td>
<td>transmission/ projection/ output matrix</td>
</tr>
<tr>
<td></td>
<td>hidden state $\rightarrow$ observation, e.g. camera taking picture of the soccer</td>
</tr>
<tr>
<td>$Q_0$</td>
<td>Initial covariance</td>
</tr>
<tr>
<td>$Q$</td>
<td>transition covariance</td>
</tr>
<tr>
<td></td>
<td>how precision is the soccer motion</td>
</tr>
<tr>
<td>$R$</td>
<td>transmission/ projection covariance</td>
</tr>
<tr>
<td></td>
<td>i.e. observation noise; e.g. how accurate is the camera</td>
</tr>
</tbody>
</table>
Estimating parameters of LDS

- **Given**: a sequence of observations (e.g. car positions)
- **Find**: best-fit parameters
- **Basic principle**: maximum likelihood
- **Through**: Expectation-Maximization Alg.
Learning LDS: EM alg.

• **E-step:** Kalman filtering-smoothing
  – Estimate the hidden variables based on observation and current parameters

• **M-step:**
  – Update parameters \((A, C...)\) to maximize

\[
L(\theta; \mathcal{X}) = \mathbb{E}_{\mathcal{X}, \mathcal{Z} | \theta}[-D(\bar{z}_1, \mu_0, Q_0) - \sum_{t=2}^{T} D(\bar{z}_t, A\bar{z}_{t-1}, Q) - \sum_{t=1}^{T} D(\bar{x}_t, C\bar{z}_t, R) - \frac{1}{2} \log |Q_0| - \frac{T-1}{2} \log |Q| - \frac{T}{2} \log |R|]
\]

[KDD 2010 Copyright: C. Faloutsos & L. Li, 2010 32]

[Shumway et al 1982, Ghahramani 1996]
EM alg. Intuition

E-step: compute hidden states using Kalman filtering-smoothing (exactly as before)

Position

Reconstructed signal after smoothing

Time
EM alg.

M-step: Maximizing the log-likelihood

\[
L(\theta; \mathcal{X}) = \mathbb{E}_{x,z|\theta}[-D(\tilde{z}_1, \tilde{\mu}_0, Q_0) - \sum_{t=2}^{T} D(\tilde{z}_t, A \tilde{z}_{t-1}, Q) - \sum_{t=1}^{T} D(\tilde{x}_t, C \tilde{z}_t, R) - \frac{1}{2} \log |Q_0| - \frac{T-1}{2} \log |Q| - \frac{T}{2} \log |R|]
\]

by solving

\[
\frac{\partial L}{\partial A} = 0
\]

\[
\frac{\partial L}{\partial C} = 0
\]

\[
\ldots
\]
Outline

• Intuition, example, and definition
• Extensions
  – Handling missing values
  – Switching LDS
  – Particle filters
• Kalman filters at work
Motivation Examples

• Motion Capture:
  – Markers on human actors
  – Cameras used to track the 3D positions
  – Duration: 100-500
  – 93 dimensional body-local coordinates after preprocessing (31-bones)

• Sensor data missing due to:
  – Low battery
  – RF error

From mocap.cs.cmu.edu
Illustration of data

sensor 1
sensor 2
...
sensor \_m
blackout
DynaMMo: Intuition

Position of Left hand marker

Recover using Correlation among multiple sequences

Position of right hand marker

missing

[Lei Li et al DynaMMo. KDD 2009]
DynaMMo: Intuition

Position of Left hand marker

Position of right hand marker

Recover using Dynamics temporal moving pattern

missing
LDS with Missing Value

hidden states

observation

partially observed
DynaMMo Illustration: estimate all colored elements

Details in [Li+2009]
DynaMMo Illustration: step 1
estimate hidden variables

Details in [Li+2009]
DynaMMo Illustration: step 2

estimate missing values

Details in [Li+2009]
DynaMMo Illustration: step 3
update model parameters

Details in [Li+2009]
DynaMMo Intuition, forward-backward algorithm

• How to recover the missing values?
DynaMMo: How to Recover?

hidden variables e.g. velocity, acceleration

projection matrix $C$

$x_1$
DynaMMyo: How to Recover?

Transition matrix

projection matrix
DynaMMo: How to Recover?
DynaMMo: How to Recover?

\[ \begin{align*}
  z_1 & \times A \\
  \downarrow & \quad \downarrow \quad \downarrow \\
  x_1 & \quad x_2 \\
\end{align*} \]
DynaMMo: How to Recover?

Each similarly backward pass

\[ z_1 \times A \times z_2 \times A \times z_3 \times A \]

\[ C \times \quad C \times \quad C \times \]

\[ x_1 \quad x_2 \quad x_3 \]
DynaMMo Learning (details)

0. Guess Initial

1. Estimate Hidden
   - Fix X, Estimate $P(Z|X;\theta)$:
     - $E(z_n|X;\theta)$,
     - $E(z_n z'_n|X;\theta)$
     - $E(z_n z'_{n+1}|X;\theta)$

2. Recover Missing
   - Fix Z, estimate
     - $E(X_{\text{missing}}|Z;\theta)$
     - Using $E(z_u|X;\theta)$,

3. Update Model
   - Fix both X and Z, estimate new model parameters $\theta$
   - $\arg\max E[\log(X,Z;\theta)]$

Random Guess model parameters $\theta$
Result – Better Missing Value Recovery

Reconstruction error

Dataset:
CMU Mocap #16
mocap.cs.cmu.edu

more results in [Li+2009]
Outline

• Intuition, example, and definition

• Extensions
  – Handling missing values
  – Switching LDS
  – Particle filters

• Kalman filters at work
Switching LDS: Intuition

• Tracking human motion
  – Start from slow walking
  – Transition to running for a while
  – Gradually stop

• Each part corresponds to a different state & dynamics
Switching LDS

S={running, walking}
Switching LDS: example

S=1, walking
A=A_1, C=C_1

S=2, running
A=A_2, C=C_2
Switching LDS in Penalty Kick

Before touching ball
S=1, running towards left
A=A_1, C=C_1

After hitting ball
S=2, kicking towards right
A=A_2, C=C_2

From FIFA 2010
Estimating Parameters for SLDS

Approximate inference: Variational EM

Outline

• Intuition, example, and definition
• Extensions
  – Handling missing values
  – Switching LDS
  – Particle filters
• Kalman filters at work
Particle Filters

- What if non-Gaussian noise?
- Inference using Markov chain Monte Carlo sampling

Outline

• Intuition, example, and definition
• Extensions
• Kalman filters at work
  – Segmentation & Compression
  – Parallel learning on Multi-core
  – Motion Stitching
How to Segment

• Segment by threshold on reconstruction error
Results — Segmentation

• Find the *transition* during “running” to “stop”.

left hip

left femur

reconstruction error

---

KDD 2010
Results — Segmentation

• Find the *transition* during “running” to “stop”.

left hip
left femur
reconstruction error
How to Compress (DynaMMo)

Original data w/ missing values

DynaMMo

hidden variables

keep only a portion (optimal samples)
Outline

• Intuition, example, and definition
• Extensions
• Kalman filters at work
  – Segmentation & Compression
  – Parallel learning on Multi-core
  – Motion Stitching
Challenge illustration

Expectation-Maximization Alg.

Timeline for E-step (forward-backward) in learning LDS

EM can only use single CPU due to data dependency

KDD 2010

Copyright: C. Faloutsos & L. Li, 2010
Parallel learning by Cut-And-Stitch Method

Step 1
Step 2
Step 3
Step 4

Goal: with 2 CPUs

[Li et al 2008b]

KDD 2010
Copyright: C. Faloutsos & L. Li, 2010
It works!

Dataset:
58 motion sequences
CMU Mocap #16
mocap.cs.cmu.edu, tested on NCSA super computer,

Proposed Cut-And-Stitch

EM algorithm

speedup

# of processors
Outline

• Intuition, example, and definition
• Extensions
• Kalman filters at work
  – Segmentation & Compression
  – Parallel learning on Multi-core
  – Motion Stitching
Motion Stitching
A Database Approach

- Select *best stitchable* segments from a set of basic motion pieces and generate new natural motions
Problem Definition

• Given two motion-capture sequences that are to be stitched together, how can we assess the goodness of the stitching?

Which stitching looks best?
Proposed Method: Laziness Score [Li+2008a]

• Conjecture: less human effort $\rightarrow$ more natural

• Proposed: use Kalman filters to estimate position, velocity, acceleration $\rightarrow$ Compute effort/energy
Which continues to?  
Green or Blue?

straight moving  
U-Turn
Result – Laziness-score prefers straightforward moving

- straight moving
- U-Turn
Conclusion

• Intuition, example, and definition
  – Original kalman filter (known parameters)
    • Kalman filtering
    • Kalman smoothing
  – Kalman filters with parameter estimation (EM)

• Extensions
  – Handling missing values
  – Switching linear dynamical
  – Particle filters (MCMC sampling)

• Kalman filters at work
  – Segmentation & compression
  – Parallel learning
  – Motion stitching
References


• Lei Li, James McCann, Christos Faloutsos, and Nancy Pollard. Laziness is a virtue: Motion stitching using effort minimization. In Short Papers Proceedings of EUROGRAPHICS, 2008.

Software

- DynaMMo code (matlab) for missing value, compression & segmentation.
- Parallel learning (in C) for LDS
- [http://www.cs.cmu.edu/~leili/](http://www.cs.cmu.edu/~leili/)
- [http://www.cs.cmu.edu/~leili/pubs/dynamm o.2.1.2.zip](http://www.cs.cmu.edu/~leili/pubs/dynamm o.2.1.2.zip)
- [http://www.cs.cmu.edu/~leili/paralearn/para learn.0.1.zip](http://www.cs.cmu.edu/~leili/paralearn/para learn.0.1.zip) (running on gcc 4.2.0 above)
Part 5: Bursty traffic and multifractals
Outline

• Motivation
• Similarity Search and Indexing
• DSP
• Linear Forecasting
• Kalman filters
• 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

#bytes

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

![Graph showing number of bytes read over time](chart.png)
solution: self-similarity

# bytes

![Graph 1](#)

# bytes

![Graph 2](#)

time

time
solution: self-similarity
solution: self-similarity
solution: self-similarity

# bytes

# bytes

time

time
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

20 \Uparrow 80

\[ x(t) \]
binary multifractals
Parameter estimation

• Q2: How to estimate the bias factor $b$?
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
- $n=1$: $E(1) = - p_1 \log_2(p_1) - p_2 \log_2(p_2)$
Entropy plot

- Entropy $E(n)$ after $n$ levels of splits
- $n=1$: $E(1) = - p_1 \log(p_1) - p_2 \log(p_2)$
- $n=2$: $E(2) = - \sum_i p_{2,i} \log_2(p_{2,i})$
Real traffic

Entropy

\[ E(n) \]

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

# of levels \((n)\)

\(0.73\)
Observation - intuition:

Entropy

\[ E(n) \]

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

\[ \text{Dim} = 1 \]

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’)

• $= \text{info bit per coordinate bit - eg}$

$\text{Dim} = 1$

\[ \text{Is MSB 0?} \]

‘info’ value $= E(1): 1$ bit
Entropy plot

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

\[ \text{Dim} = 1 \]

Is MSB 0?

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
= E(2) - E(1) = slope!
Is MSB 0?
Is next MSB = 0?
Entropy plot

• Repeat, for all points at same position:

Dim=0

Skip
Entropy plot

• Repeat, for all points at same position:
  • we need 0 bits of info, to determine position
  • -> slope = 0 = intrinsic dimensionality

Dim=0
Entropy plot

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

Dim = 1

Dim = 0

0 < Dim < 1
(Fractals)

• 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

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

- Poisson vs real

Poisson: slope = \sim 1 \rightarrow \text{uniformly distributed}
B-model

- b-model traffic gives perfectly linear plot

- Lemma: its slope is
  \[ \text{slope} = -b \log_2 b - (1-b) \log_2 (1-b) \]

- 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
  
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)

\[ \text{Prob} \left( q > l \right) \]

Queue length distribution

(a) lbl-all  
(b) lbl-nntp  
(c) lbl-smtp  
(d) lbl-ftp

How to give guarantees?  
(queue length \( l \))
Web traffic - results

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

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

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
- can give (probabilistic) guarantees
Books

Further reading:


Further reading


Part 6: chaos and non-linear forecasting
Outline

• Motivation
• Similarity Search and Indexing
• DSP
• Linear Forecasting
• Kalman filters
• 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, \(x_{t+1}, x_{t+2}, \ldots\)
How to forecast?

• ARIMA - but: linearity assumption

• ANSWER: ‘Delayed Coordinate Embedding’ = Lag Plots [Sauer92]
General Intuition (Lag Plot)

Interpolate these…

To get the final prediction

Lag = 1,
k = 4 NN

4-NN

New Point

Xt

Xt-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)
Theoretical foundation

Example: Lotka-Volterra equations

\[ \frac{dH}{dt} = rH - aHP \]
\[ \frac{dP}{dt} = bHP - mP \]

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

Suppose only \( P(t) \) is observed \( (t=1, 2, \ldots) \).
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.
Solution to Volterra-Lotka eq.

from wikipedia
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
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
Logistic Parabola

Our Prediction from here

Value

Timesteps
Logistic Parabola

Comparison of prediction to correct values
Datasets

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

Comparison of prediction to correct values
Datasets

• 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
Overall conclusions

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

• Signal processing: **DWT** is a powerful tool
Overall conclusions

• Similarity search: Euclidean/time-warping; feature extraction and SAMs
• Signal processing: DWT is a powerful tool
• Linear Forecasting: AR (Box-Jenkins)
Overall conclusions

• Similarity search: *Euclidean/time-warping; feature extraction and SAMs*
• Signal processing: *DWT is a powerful tool*
• Linear Forecasting: *AR (Box-Jenkins)*
• *Kalman* filters & extensions: forecasting, pattern discovery, segmentation
Overall conclusions

• Similarity search: Euclidean/time-warping; feature extraction and SAMs
• Signal processing: DWT is a powerful tool
• Linear Forecasting: AR (Box-Jenkins)
• Kalman filters & extensions: forecasting, pattern discovery, segmentation
• 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)
• **Kalman** filters & extensions: forecasting, pattern discovery, segmentation
• Bursty traffic: **multifractals** (80-20 ‘law’)
• Non-linear forecasting: **lag-plots** (Takens)
THANK YOU!

christos AT cs.cmu.edu
www.cs.cmu.edu/~christos

leili AT cs.cmu.edu
www.cs.cmu.edu/~leili

www.cs.cmu.edu/~leili/pubs/dynammo.2.1.2.zip