Carnegie Mellon

amazon

T25. Forecasting Big Time Series: Theory and Practice



Christos Faloutsos, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Yuyang (Bernie) Wang


@Summit 5- Ground Level, Egan

Outline

- Motivation
- Part 1: Classical methods
  - Similarity Search and Indexing
  - DSP (Digital Signal Processing)
  - Linear Forecasting
  - Non-linear forecasting
  - Tensors
  - Conclusions
- Part 2: Modern methods – Neural Networks

Problem definition

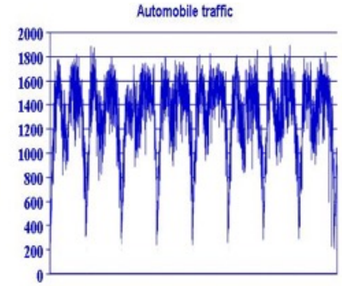
- Given: one or more sequences

  $x_1, \; x_2, \; \ldots, \; x_t, \; \ldots$

  $(y_1, y_2, \ldots, y_t, \ldots$

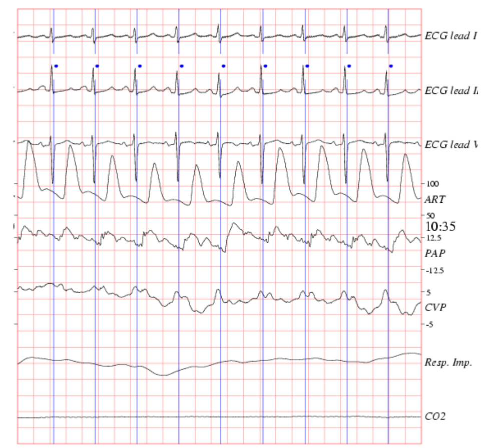
  $\ldots)$

- Find
  - **Forecast**; similar sequences
  - patterns; clusters; outliers



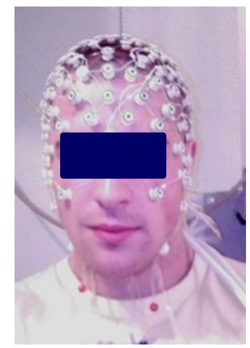
Motivation - Applications

- Financial, sales, economic series
- Medical
  - ECG
  - reactions to new drugs
  - elderly care



physionet.org

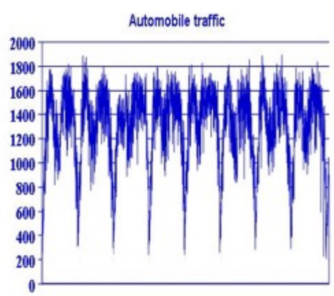
EEG - epilepsy



from wikipedia

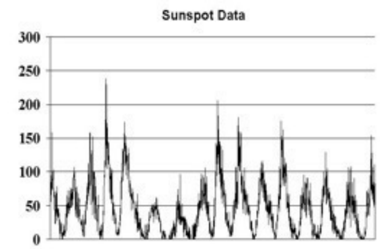
Motivation - Applications (cont'd)

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



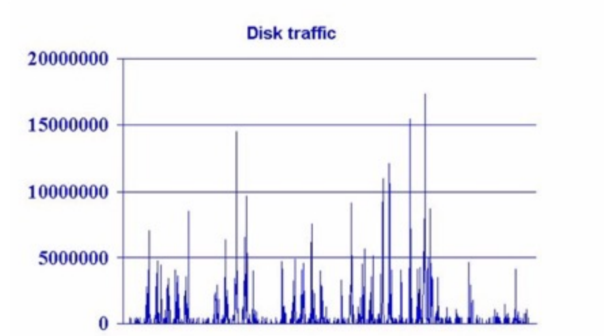
Motivation - Applications (cont'd)

- Weather, environment/anti-pollution
  - volcano monitoring
  - air/water pollutant monitoring
  - Sunspots (magnetic storms)



Motivation - Applications (cont'd)

- Computer systems
  - Disks (buffering, prefetching)
  - web servers (ditto)
  - network traffic monitoring
  - ...

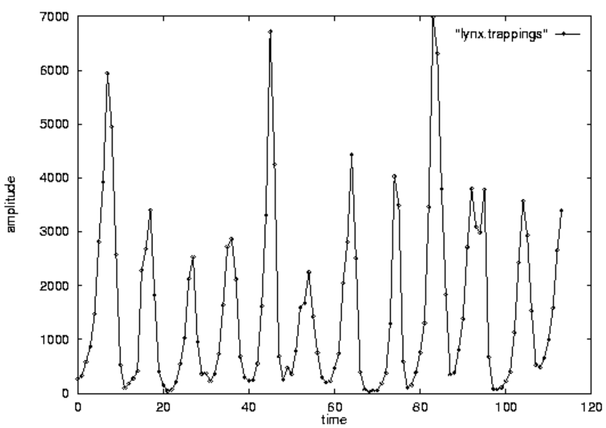


Problem #1:

Goal: given a signal (eg., #packets over time)

Find: patterns, periodicities, and/or compress

count

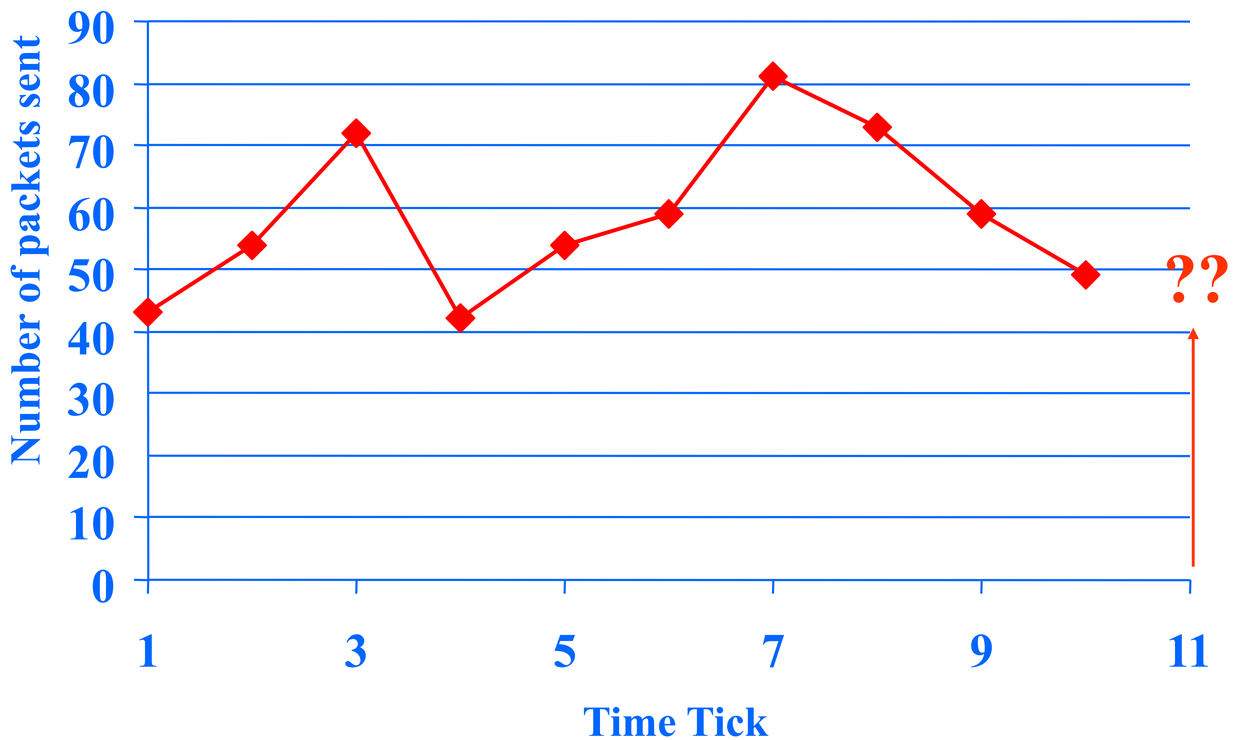


lynx caught per year
(packets per day;
temperature per day)

year

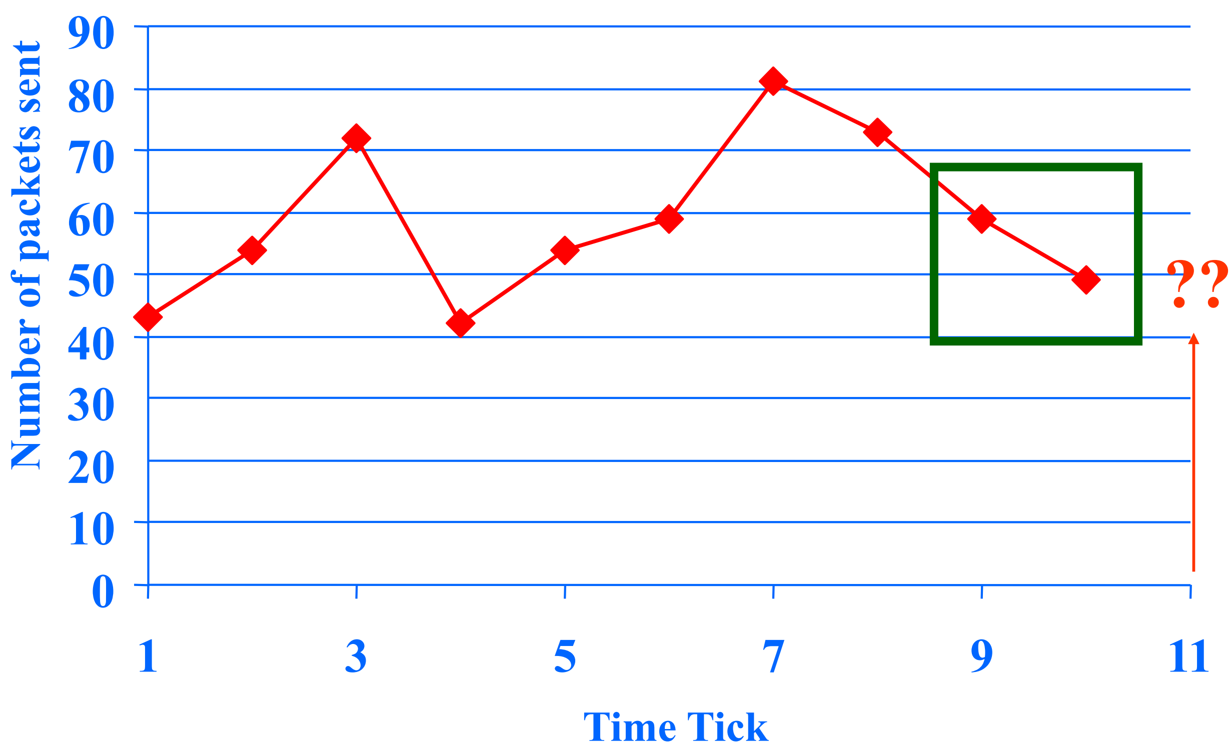
Problem#2: Forecast

Given x_t, x_{t-1}, …, forecast x_{t+1}



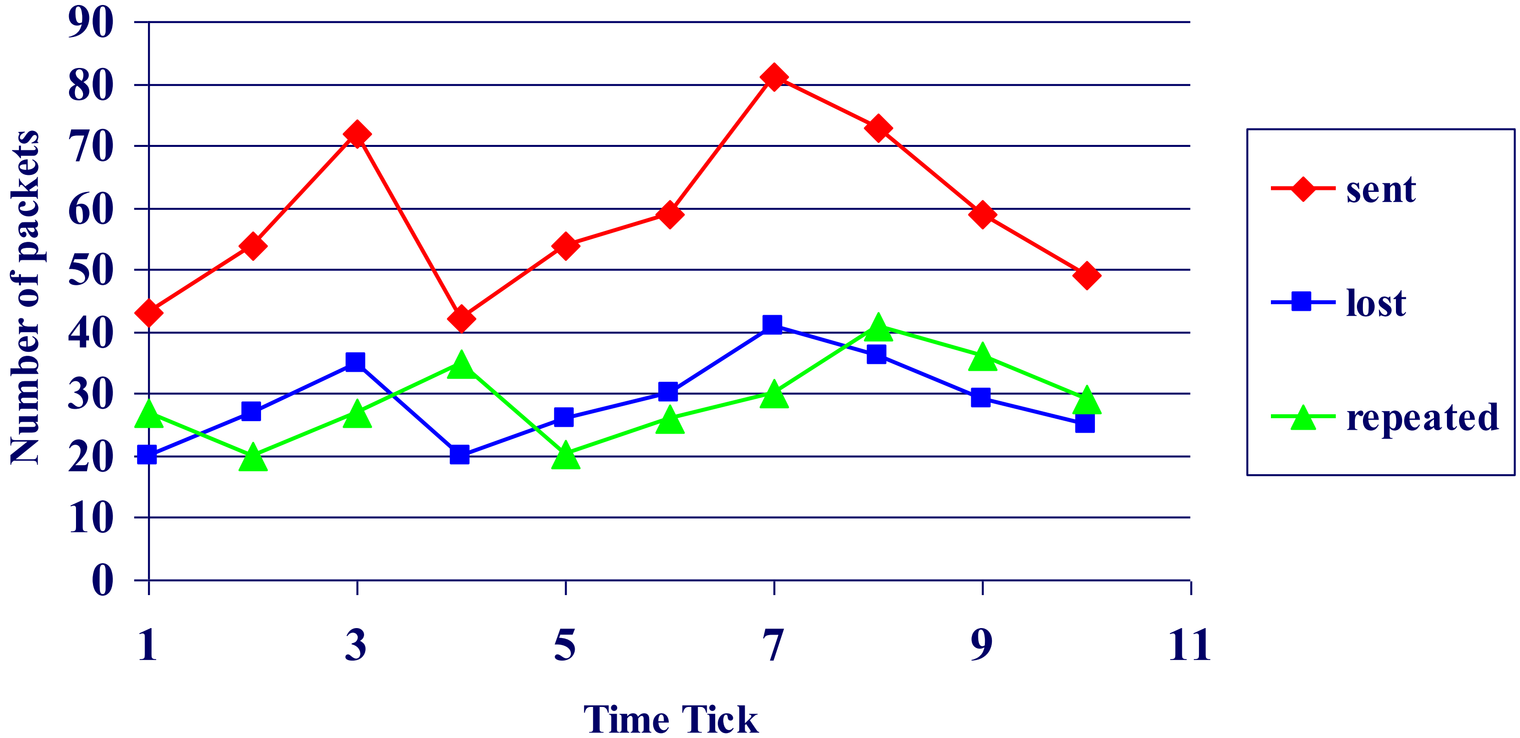
Problem#2': Similarity search

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

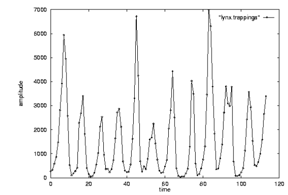


Problem #3:

- Given: A set of **correlated** time sequences
- Forecast 'Sent(t)'

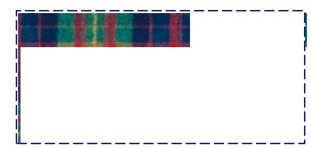


Important observations



Patterns, rules, forecasting and similarity indexing are closely related:

- To do **forecasting**, we need
  - to find **patterns**/rules
  - compress
  - to find **similar** settings in the past



- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)

Outline

- Motivation
- Part 1: classical methods
  - Similarity Search and Indexing
  - DSP
  - Linear Forecasting
  - Non-linear forecasting
  - Tensors
  - Conclusions

Detailed Outline

- Motivation
- Part 1
  - Similarity Search and Indexing
    - distance functions: Euclidean;Time-warping
    - indexing
    - feature extraction
  - DSP
  - ...

Importance of distance functions

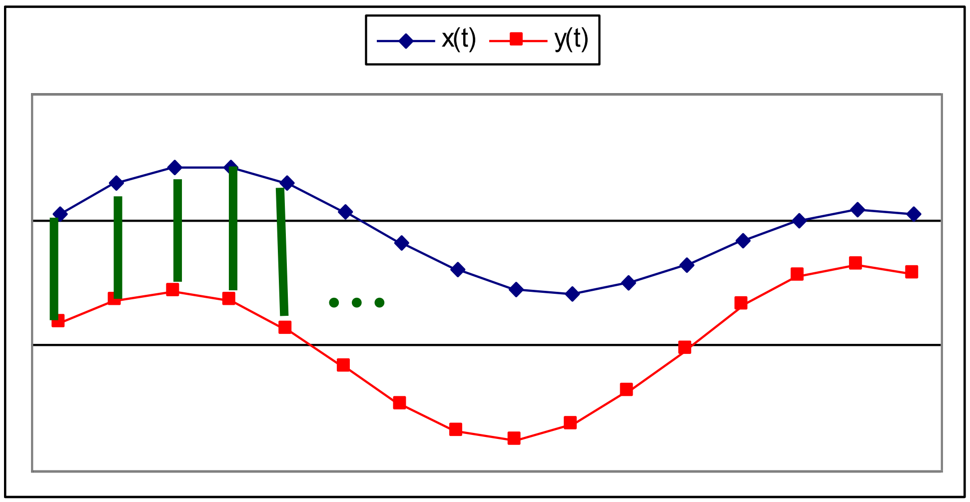
Subtle, but **absolutely necessary**:

- A 'must' for similarity indexing (-> forecasting)
- A 'must' for clustering

Two major families

- Euclidean and Lp norms
- Time warping and variations

Euclidean and Lp



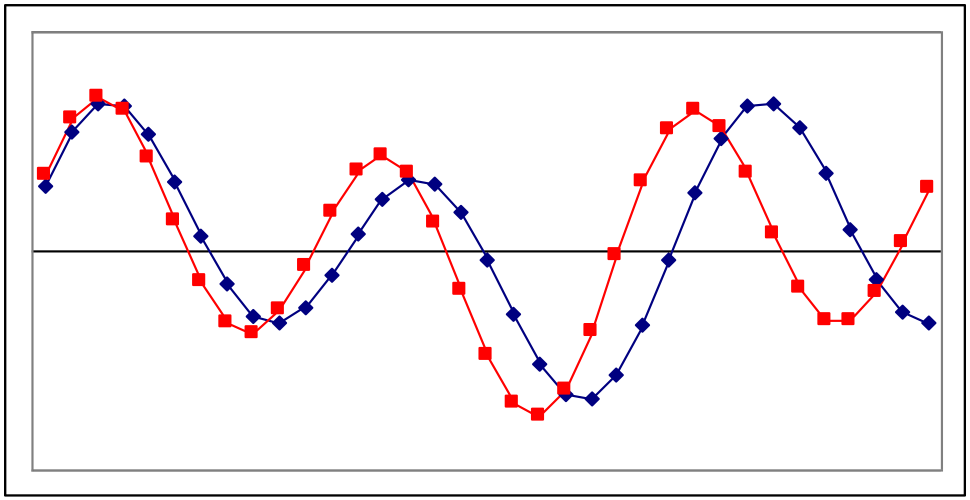
$$D(\vec{x},\vec{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$$

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

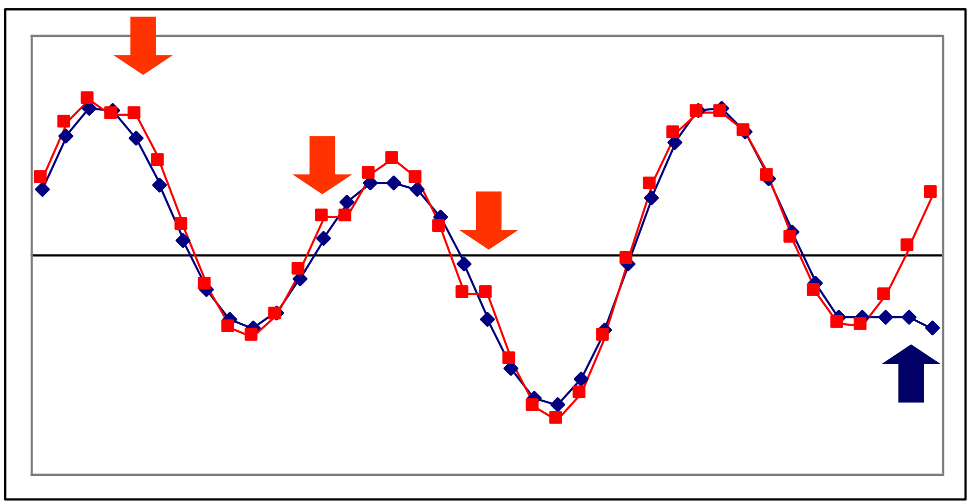
- L_1: city-block = Manhattan
- L_2 = Euclidean
- L_∞

Skip

Time Warping



'stutters':



Skip

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]

Skip

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 Lp-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

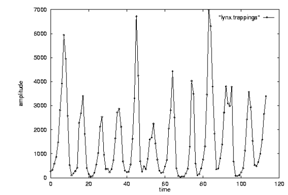
Prevailing distances:

- **Euclidean** and
- time-warping

Outline

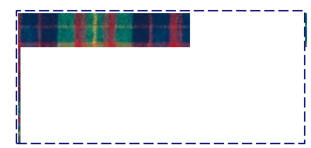
- Motivation
- Part 1: classical methods
  - Similarity Search and Indexing
    - distance functions
    - indexing
    - feature extraction
  - DSP
  - ...

Important observations



Patterns, rules, forecasting and similarity indexing are closely related:

- To do forecasting, we need
  - to find patterns/rules
  - compress
  - **to find similar settings in the past**

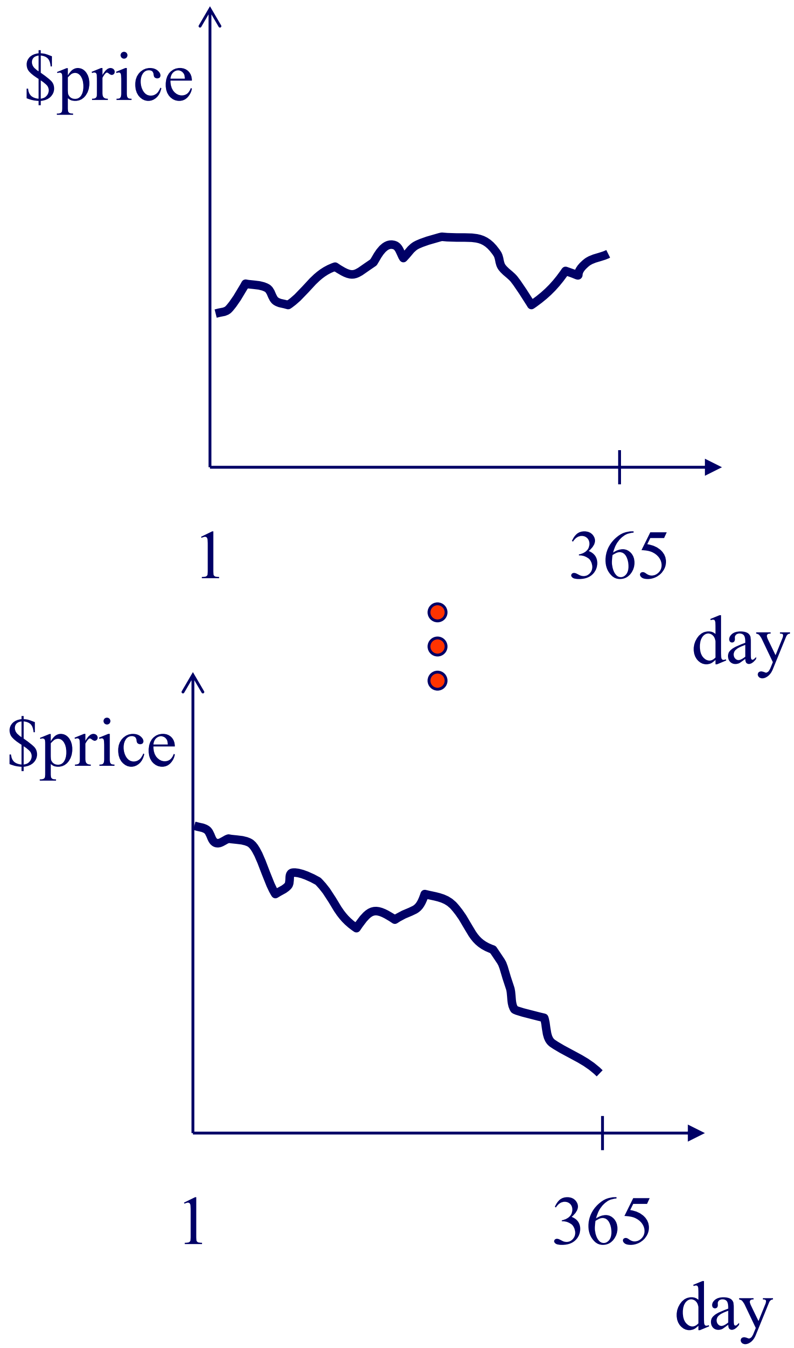


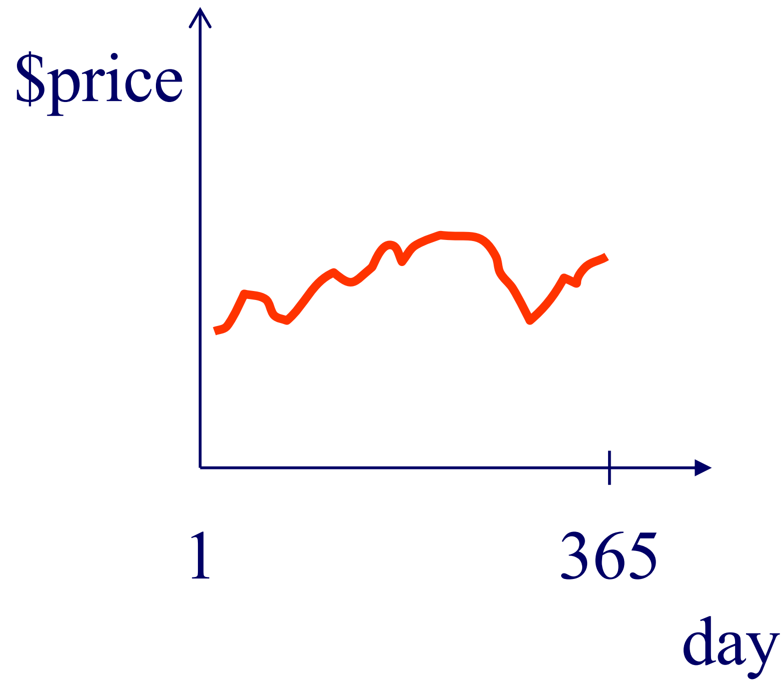
- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)

Indexing

Problem:

- given a set of time sequences,
- find the ones similar to a desirable query sequence





distance function: by expert

Idea: 'GEMINI'

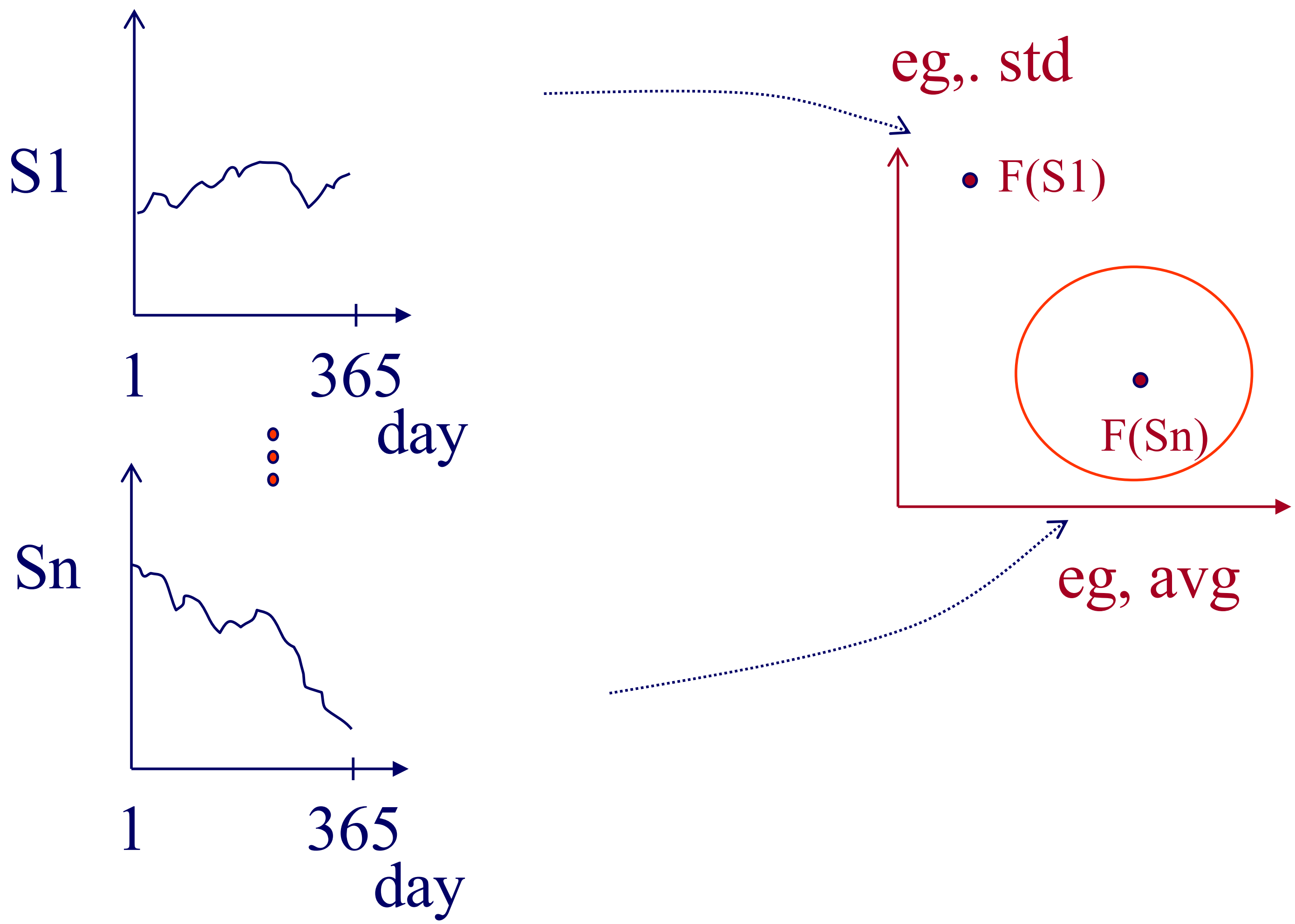
Eg., '*find stocks similar to MSFT*'

Seq. scanning: too slow

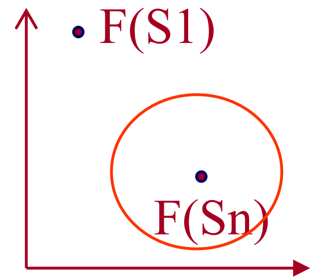
How to accelerate the search?

[Faloutsos96]

‘GEMINI’ - Pictorially



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

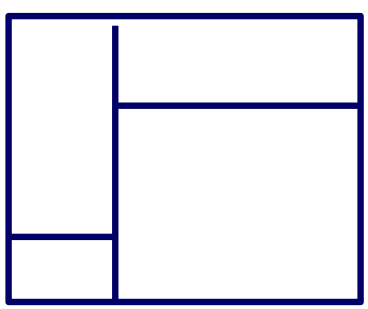
- Time sequences: DFT (up to 100 times faster) [SIGMOD94];
- [Kanellakis+], [Mendelzon+]

Indexing - SAMs

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

A: they group nearby points (or regions) together, and answer spatial queries quickly ('range queries', 'nearest neighbor' queries etc)

k-d tree



Conclusions

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

But: features?

- How to extract ‘a few, good features’?
- A1: SVD
- A2: Fourier; Wavelets

But: features?

- How to extract 'a few, good features'?
- A1: SVD
- A2: Fourier; Wavelets

Feature extraction
= (lossy) compression
= Dimensionality reduction
= Embedding
= Auto-encoding

Conclusions - Practitioner's guide

Similarity search in time sequences

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

2) extract features (SVD, DWT), and use an SAM (R-tree/k-d-tree/variant)

2') for high <u>intrinsic</u> dimensionalities, consider sequential scan (it might win…)

Books

- William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery: *Numerical Recipes in C*, Cambridge University Press, 1992, 2nd Edition. (Great description, intuition and code for SVD)
- C. Faloutsos: *Searching Multimedia Databases by Content*, Kluwer Academic Press, 1996 (introduction to SVD, and GEMINI)

References

- Agrawal, R., K.-I. Lin, et al. (Sept. 1995). Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases. Proc. of VLDB, Zurich, Switzerland.

References

- Ciaccia, P., M. Patella, et al. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. VLDB.
- Guttman, A. (June 1984). R-Trees: A Dynamic Index Structure for Spatial Searching. Proc. ACM SIGMOD, Boston, Mass.

References

- Gaede, V. and O. Guenther (1998). "Multidimensional Access Methods." Computing Surveys 30(2): 170-231.

References

- Gunopulos, D. and G. Das (2001). Time Series Similarity Measures and Time Series Indexing. SIGMOD Conference, Santa Barbara, CA.
- Eamonn J. Keogh, Themis Palpanas, Victor B. Zordan, Dimitrios Gunopulos, Marc Cardle: Indexing Large Human-Motion Databases. VLDB 2004: 780-791

References

- Hatonen, K., M. Klemettinen, et al. (1996). Knowledge Discovery from Telecommunication Network Alarm Databases. ICDE, New Orleans, Louisiana.
- Jolliffe, I. T. (1986). Principal Component Analysis, Springer Verlag.

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.
- Rabiner, L. and B.-H. Juang (1993). Fundamentals of Speech Recognition, Prentice Hall.

References

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

Part 1.2: DSP (Digital Signal Processing)

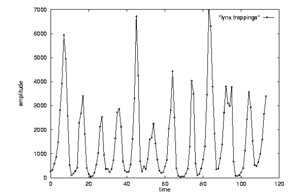
Outline

- Motivation
- Part 1: classical methods
  - Similarity Search
  - DSP (DFT, DWT)
  - Linear Forecasting
  - Non-linear forecasting
  - Tensors
  - Conclusions

Outline

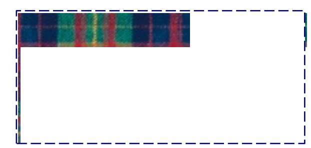
- DFT
  - Definition of DFT and properties
  - how to read the DFT spectrum
- DWT
  - Definition of DWT and properties
  - how to read the DWT scalogram

Important observations



Patterns, rules, forecasting and similarity indexing are closely related:

- To do forecasting, we need
  - to find patterns/rules
  - **compress**
  - to find similar settings in the past



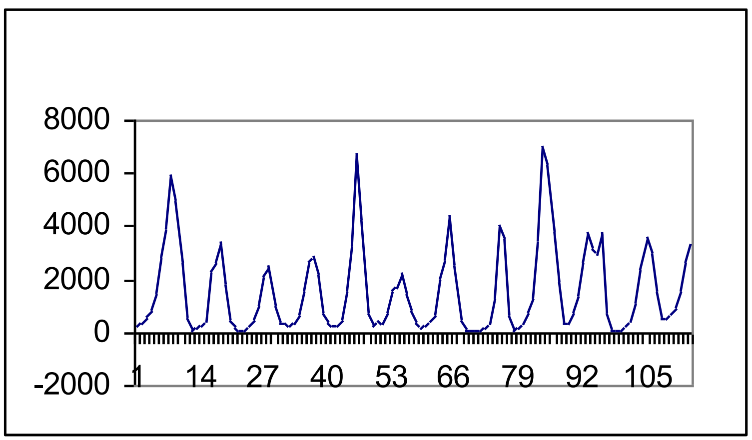
- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)

Introduction - Problem#1

Goal: given a signal (eg., packets over time)

Find: patterns and/or compress

count



year

lynx caught per year
(packets per day;
automobiles per hour)

What does DFT do?

A: highlights the periodicities

Skip

DFT: definition

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

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t * \exp(-j 2\pi\, tf / n) \qquad f = 0, \ldots, n-1$$

$$(j = \sqrt{-1})$$

inverse DFT

$$x_t = 1/\sqrt{n} \sum_{t=0}^{n-1} X_f * \exp(+j 2\pi\, tf / n)$$

DFT: definition

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

  x = [1,2,1,2];

  X = Fourier[x];

  Plot[Abs[X]];

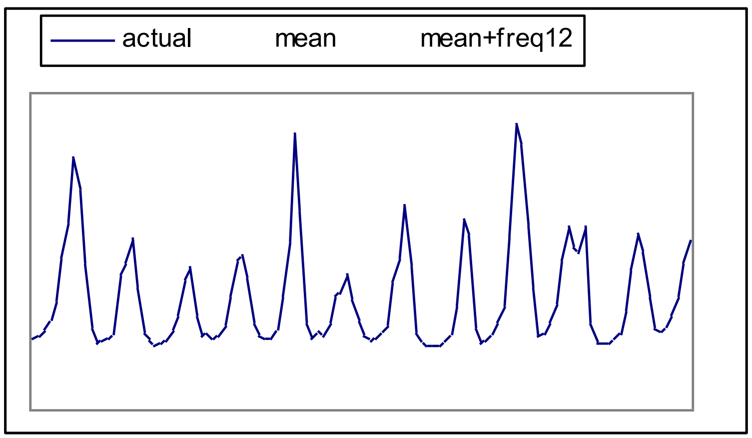
DFT: Amplitude spectrum

Amplitude: $A_f^{\,2} = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$

x

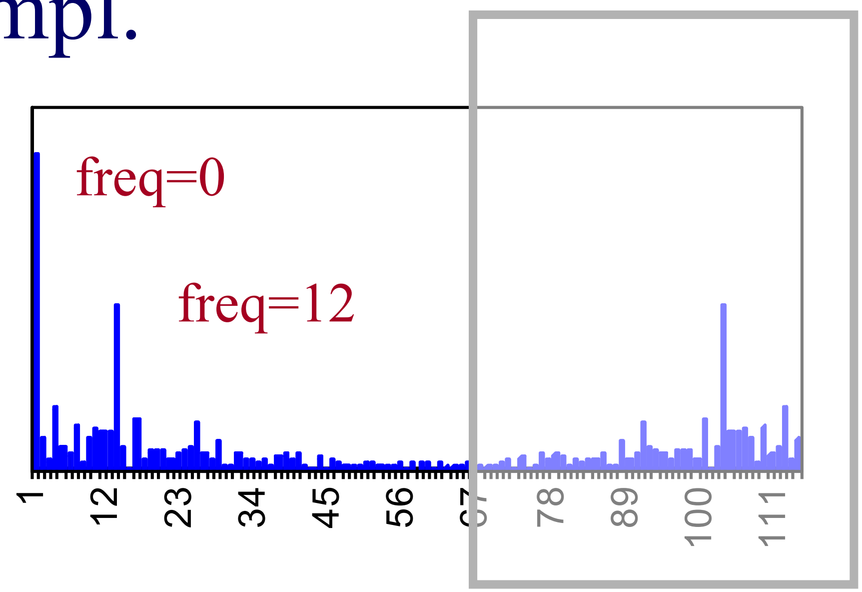
Plot[Abs[Fourier[x]]];

count



year

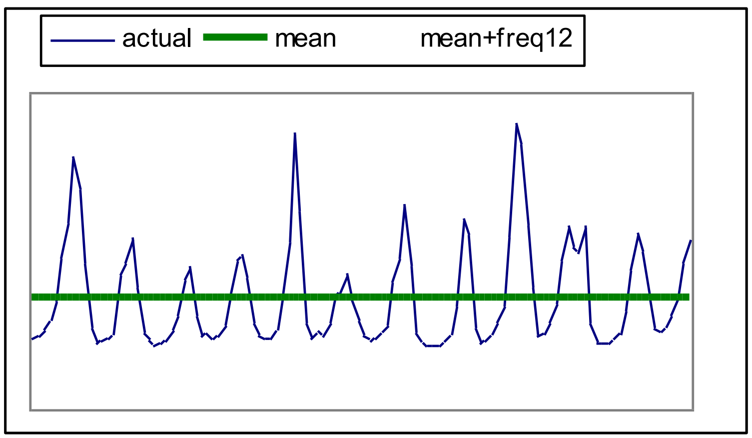
Ampl.



Freq.

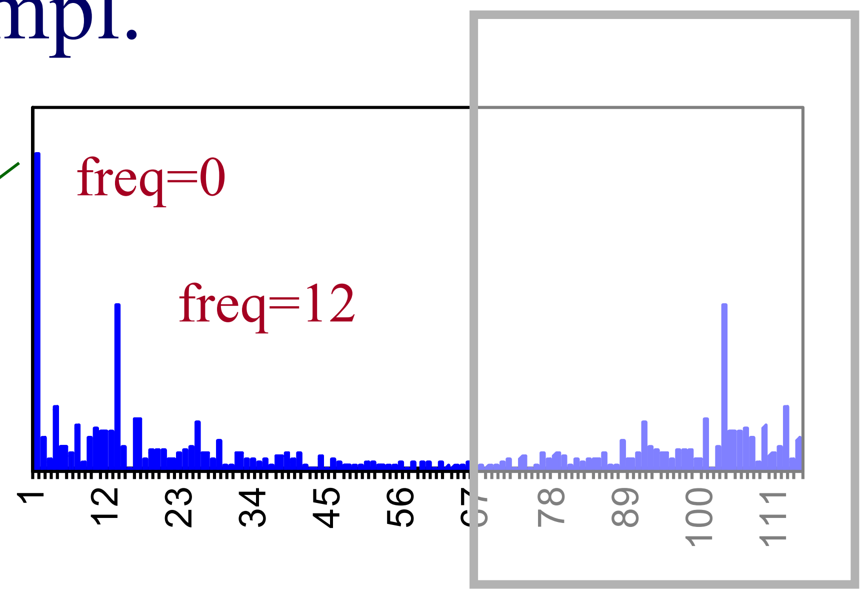
DFT: Amplitude spectrum

count



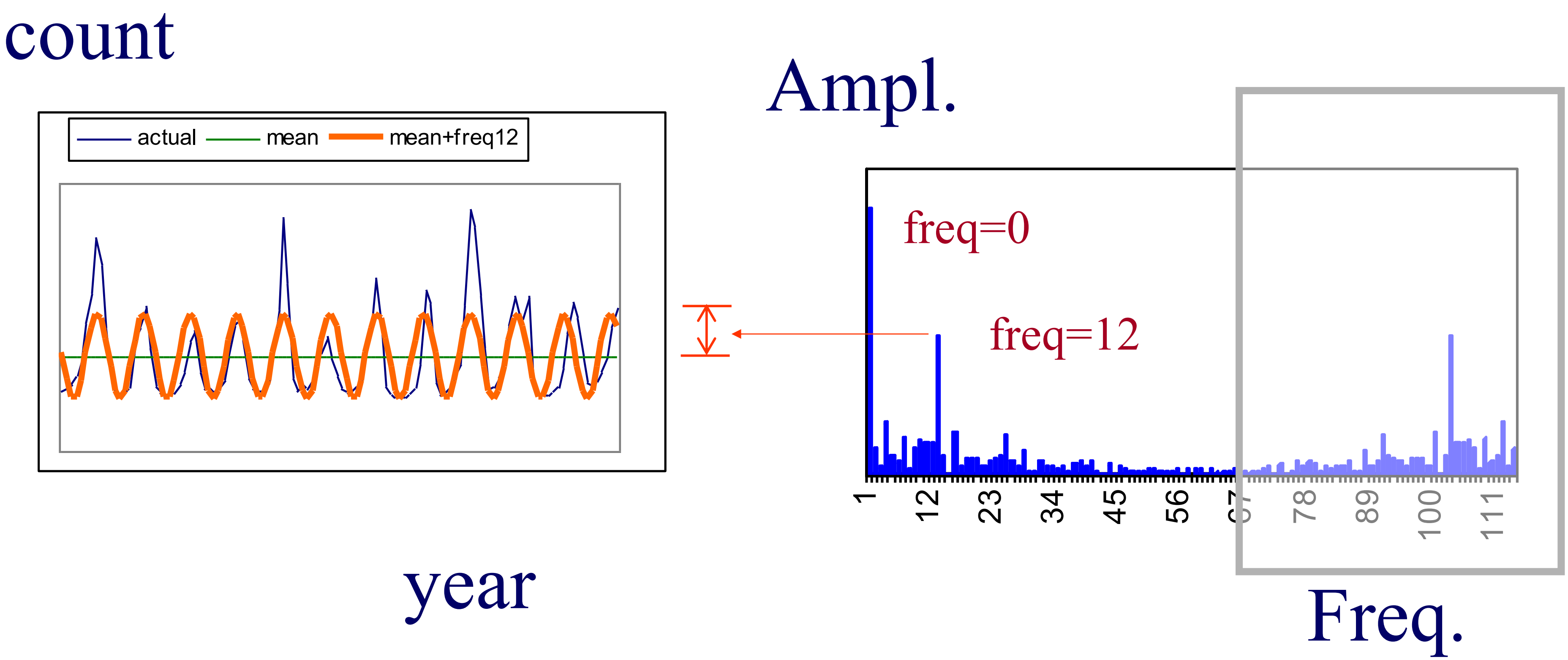
year

Ampl.



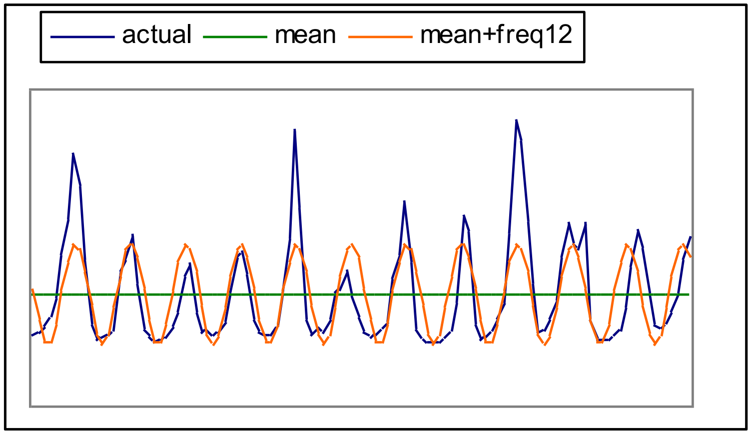
Freq.

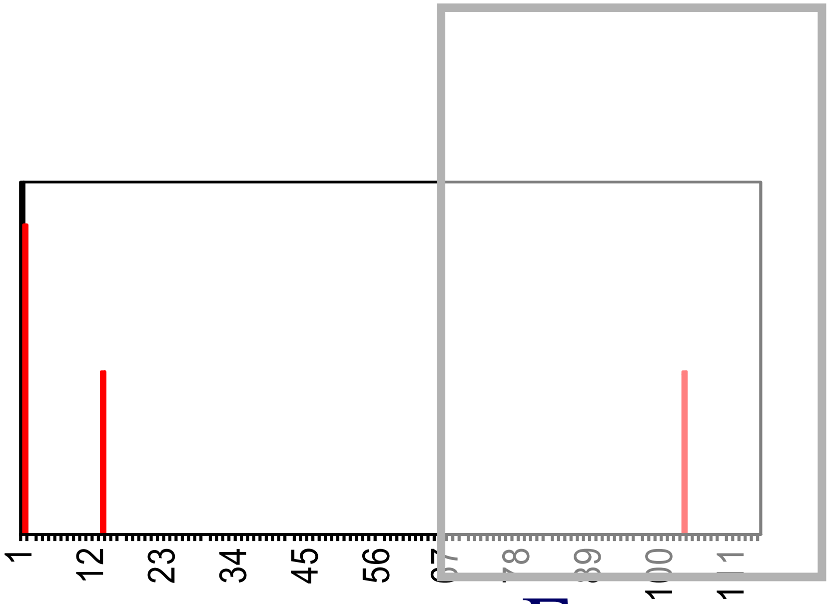
DFT: Amplitude spectrum



DFT: Amplitude spectrum

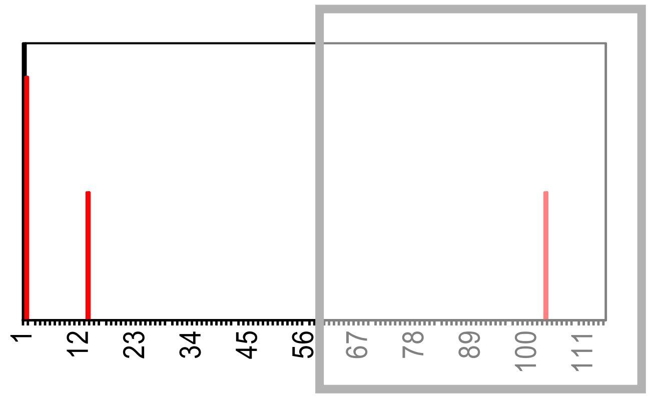
- excellent approximation, with only 2 frequencies!
- so what?

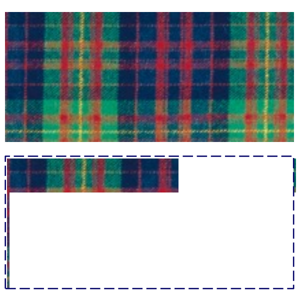




DFT: Amplitude spectrum

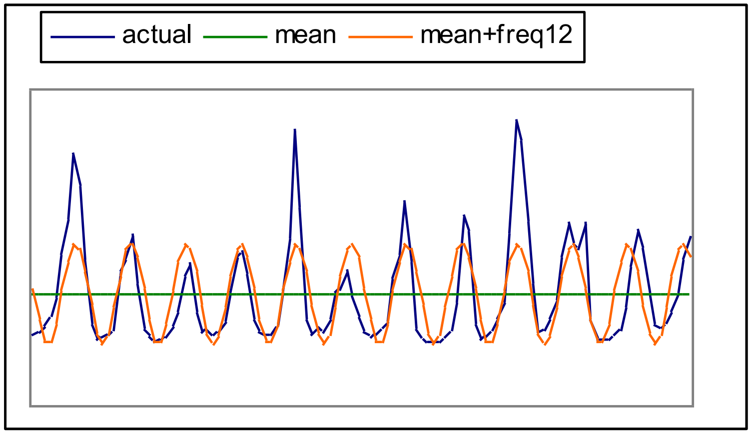
- excellent approximation, with only 2 frequencies!
- so what?
- A1: **(lossy) compression**
- A2: pattern discovery





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

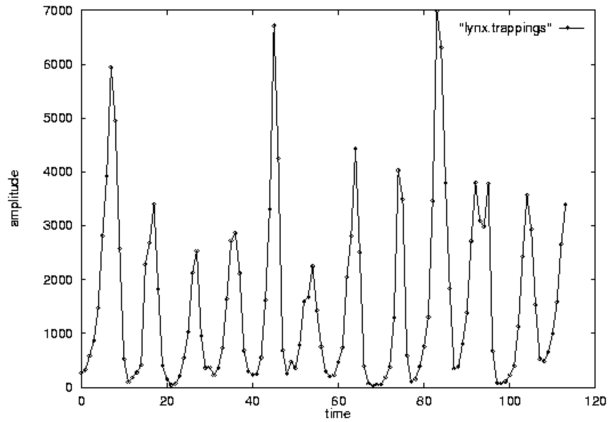
- Motivation
- Part 1: classical methods
  - Similarity Search and Indexing
  - DSP
    - DFT
    - DWT
      - Definition of DWT and properties
      - how to read the DWT scalogram

Problem #1:

Goal: given a signal (eg., #packets over time)

Find: patterns, periodicities, and/or **compress**

count



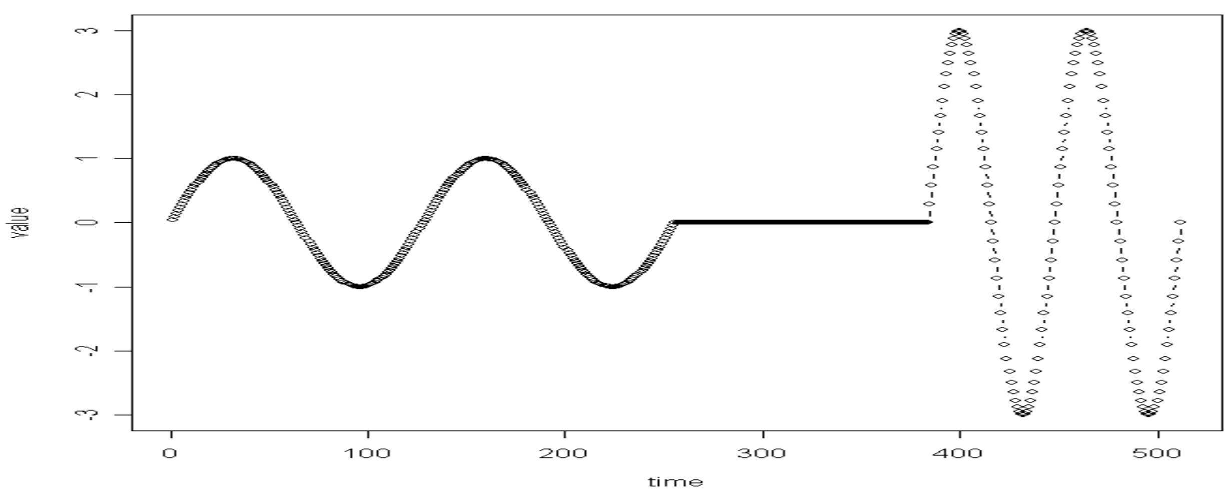
lynx caught per year
(packets per day;
virus infections per month)

year

Wavelets - DWT

- DFT suffers on short-duration waves (eg., baritone, silence, soprano)

value

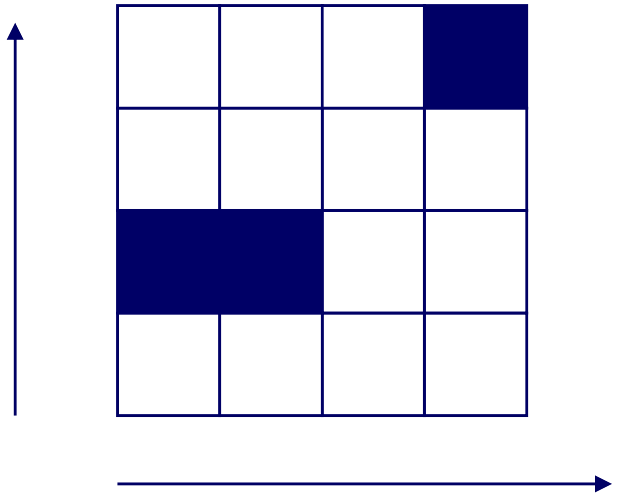


time

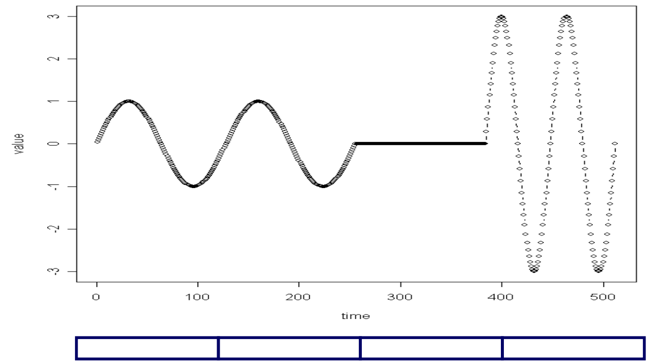
Wavelets - DWT

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

freq



value

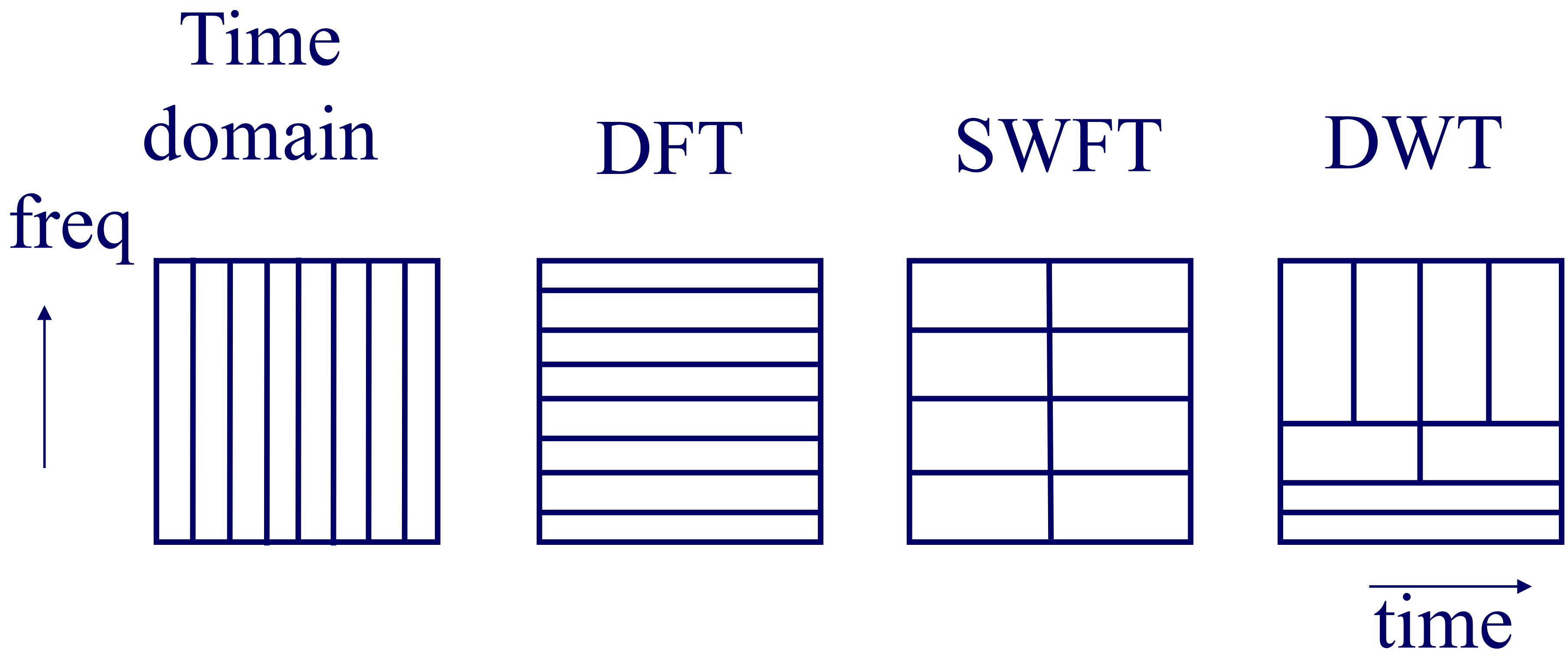


ime

time

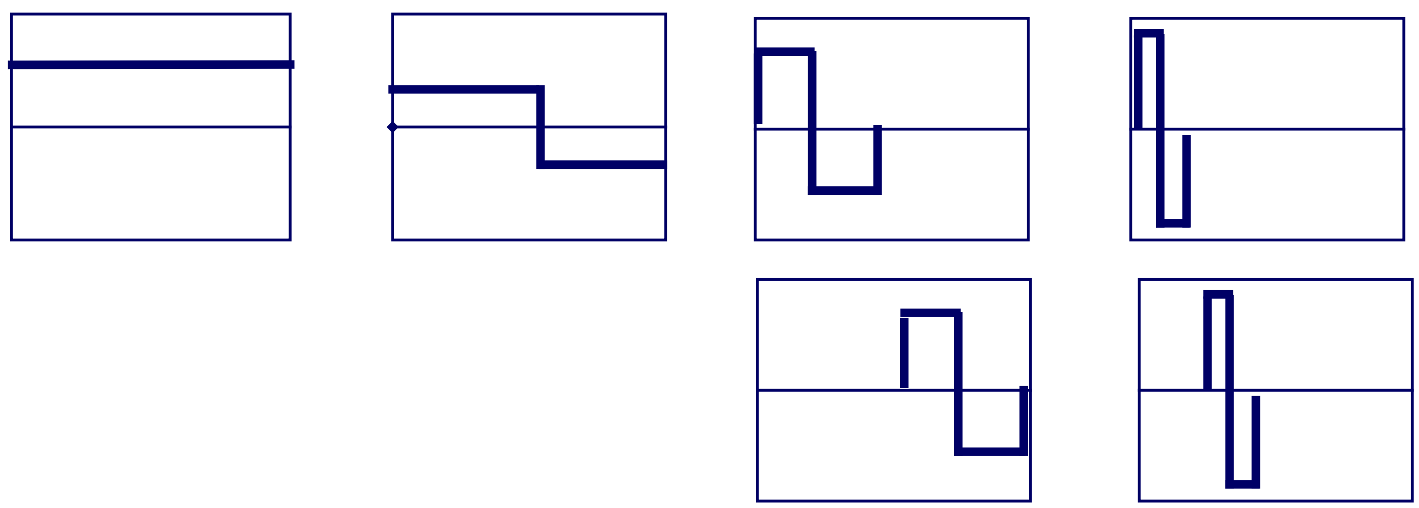
Wavelets - DWT

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



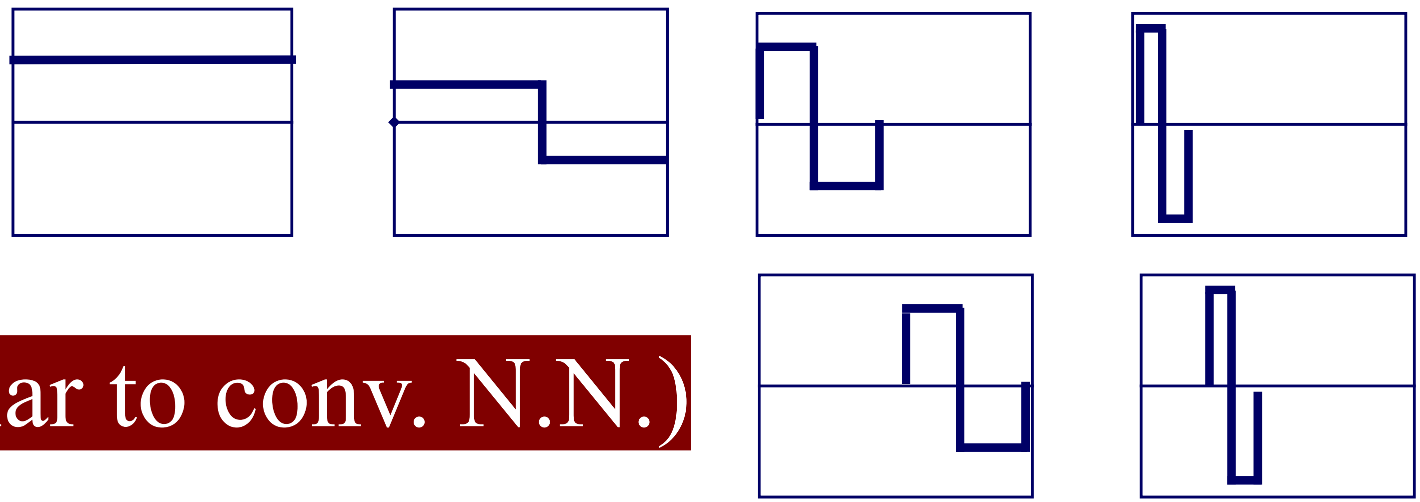
Haar Wavelets

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



Haar Wavelets

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



(Similar to conv. N.N.)

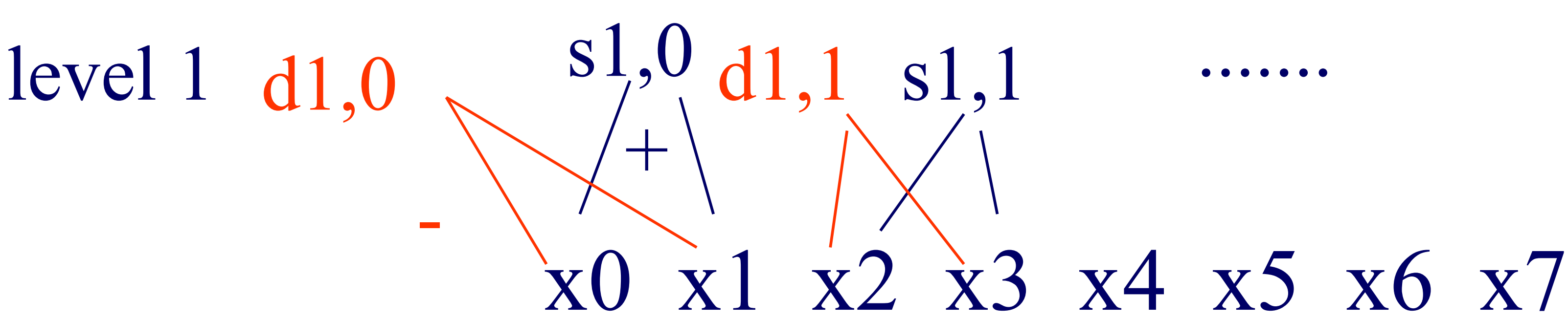
Skip

Wavelets - construction

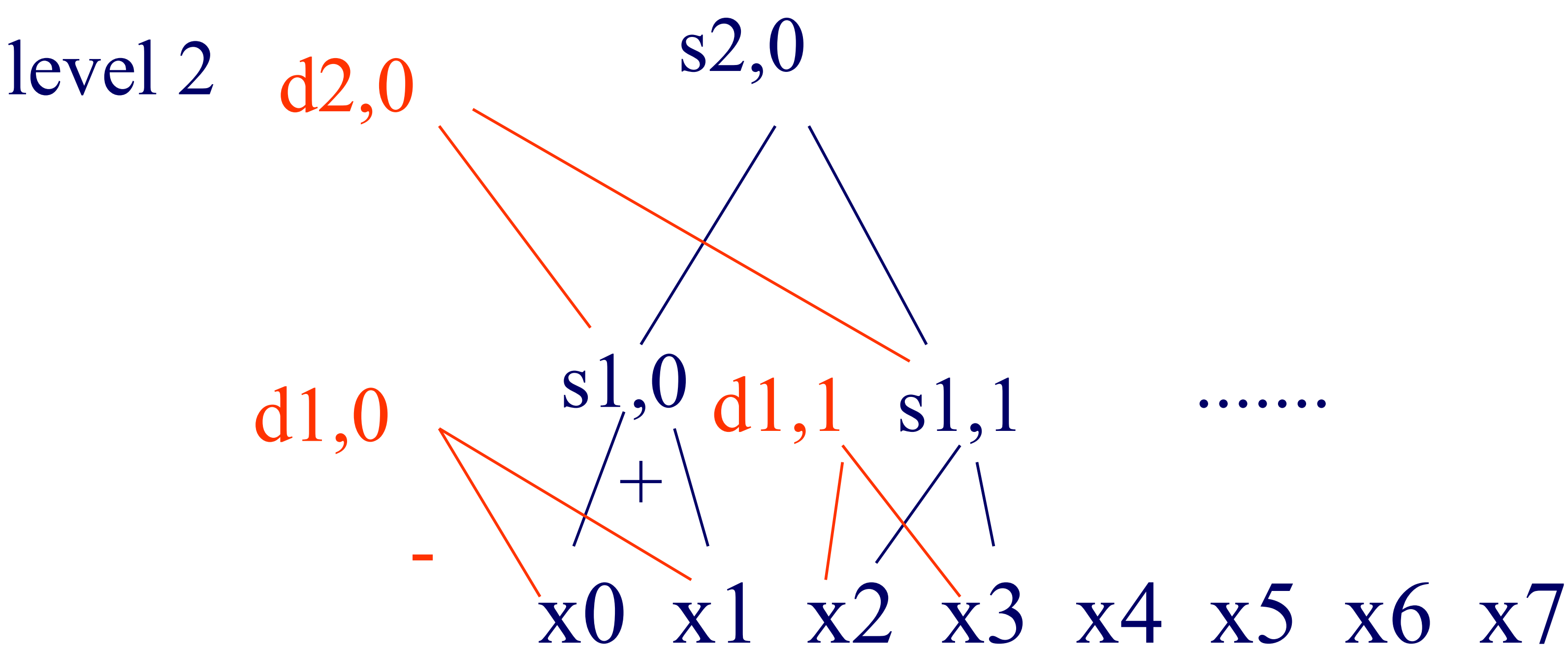
x0 x1 x2 x3 x4 x5 x6 x7

Skip

Wavelets - construction



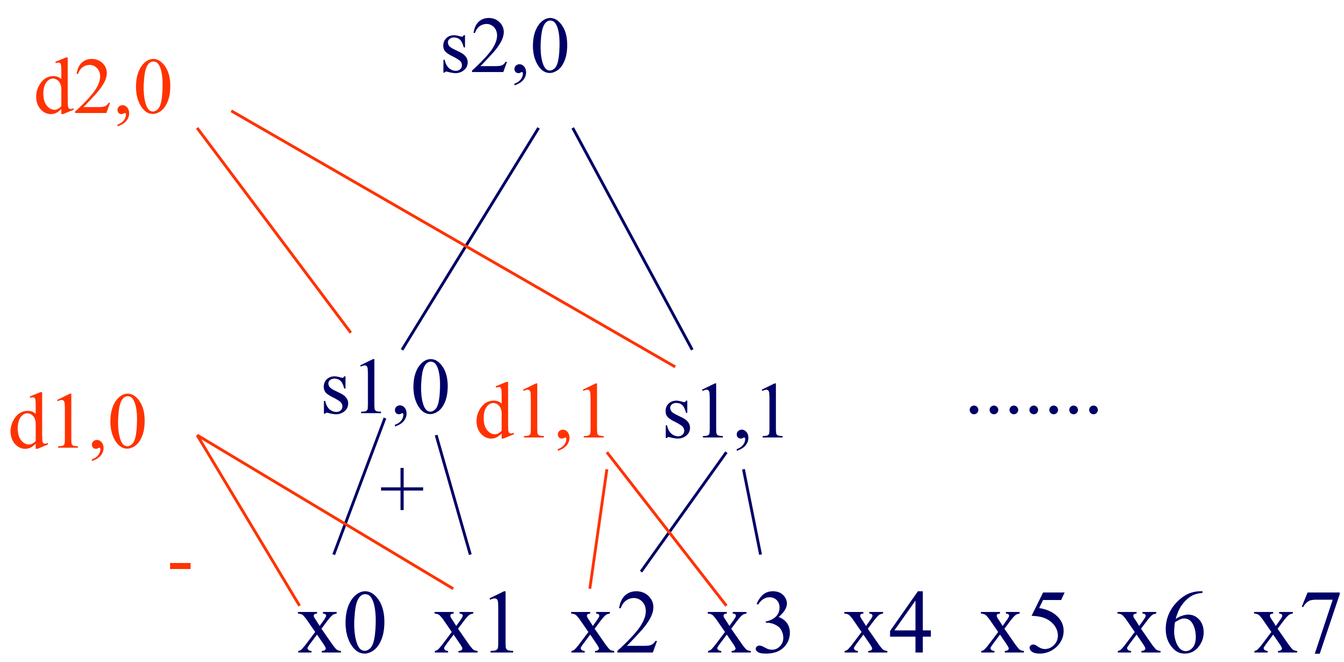
Wavelets - construction



Wavelets - construction

Skip

etc ...

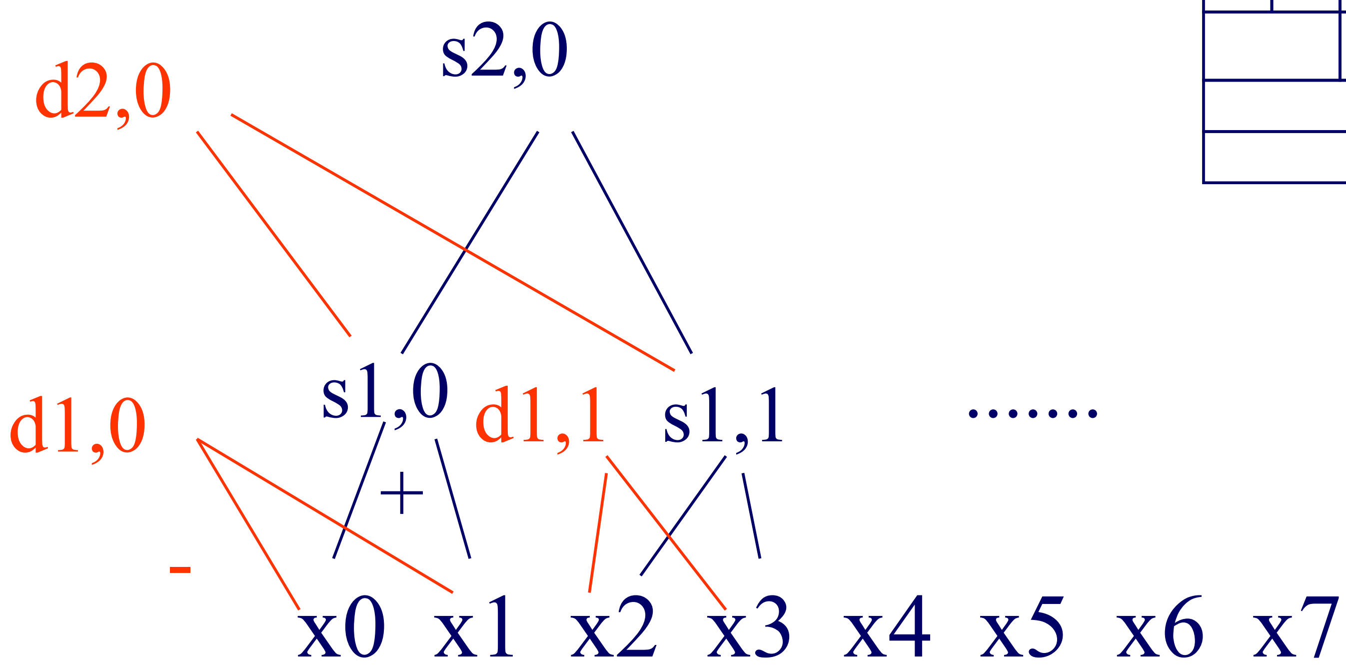


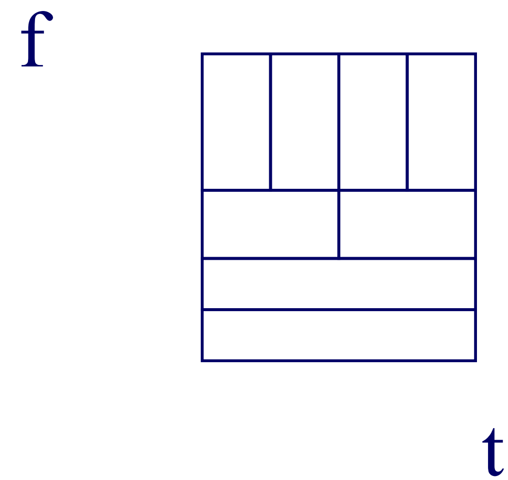
Skip

Wavelets - construction

Q: map each coefficient

on the time-freq. plane



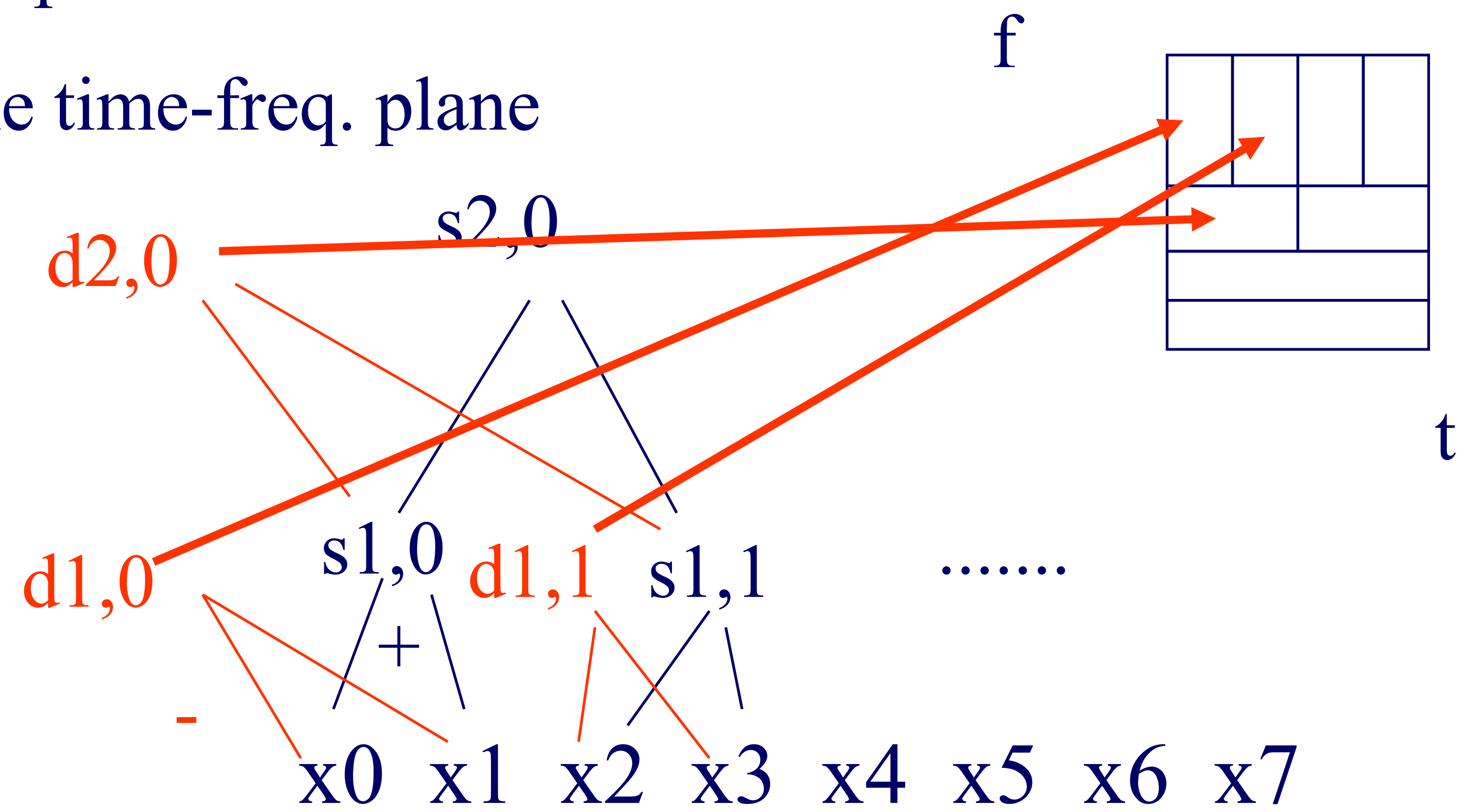


Skip

Wavelets - construction

Q: map each coefficient

on the time-freq. plane



Haar wavelets - code

```
#!/usr/bin/perl5
# 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

Observation1:

- '+' can be some weighted addition
- '-' is the corresponding weighted difference ('Quadrature mirror filters')

Observation2: unlike DFT/DCT,

- there are *many* wavelet bases: **Haar**, Daubechies-4, Daubechies-6, Coifman, Morlet, Gabor, ...

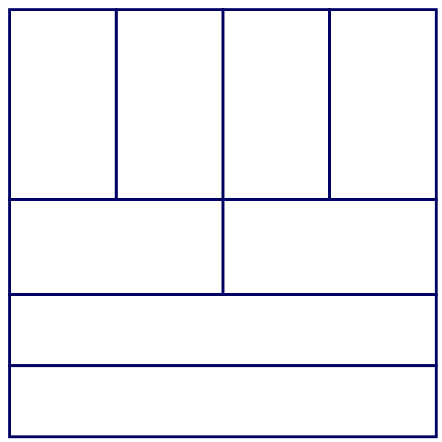
Outline

- Motivation
- Part 1: classical methods
  - Similarity Search and Indexing
  - DSP
    - DFT
    - DWT
      - Definition of DWT and properties
      - how to read the DWT scalogram

Wavelets - Drill#1:

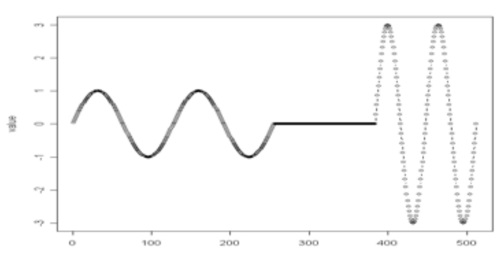
- Q: baritone/silence/soprano - DWT?

f



t

value

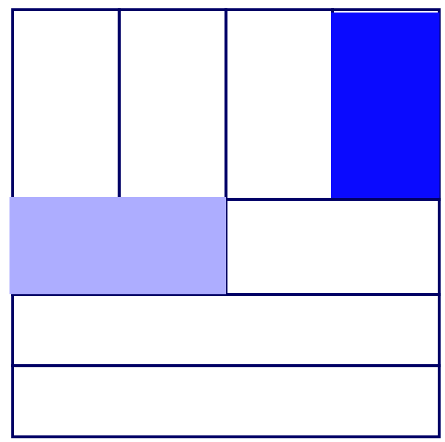


time

Wavelets - Drill#1:

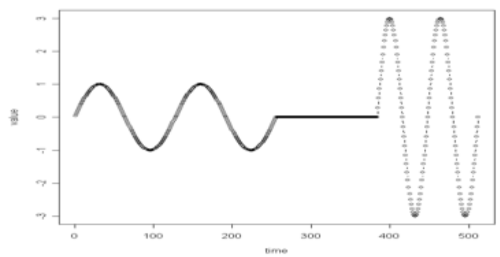
- Q: baritone/silence/soprano - DWT?

f



t

value



time

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

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

Books

- William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery: *Numerical Recipes in C*, Cambridge University Press, 1992, 2nd Edition. (Great description, intuition and code for DFT, DWT)
- C. Faloutsos: *Searching Multimedia Databases by Content*, Kluwer Academic Press, 1996 (introduction to DFT, DWT)

Additional Reading

- [Gilbert+01] Anna C. Gilbert, Yannis Kotidis and S. Muthukrishnan and Martin Strauss, *Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries*, VLDB 2001

Part 1.3: Linear Forecasting

Outline

- Motivation
- Part 1: classical methods
  – Similarity Search and Indexing
  – DSP
  – Linear Forecasting
  – Non-linear forecasting
  – Tensors
  – Conclusions

Forecasting

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

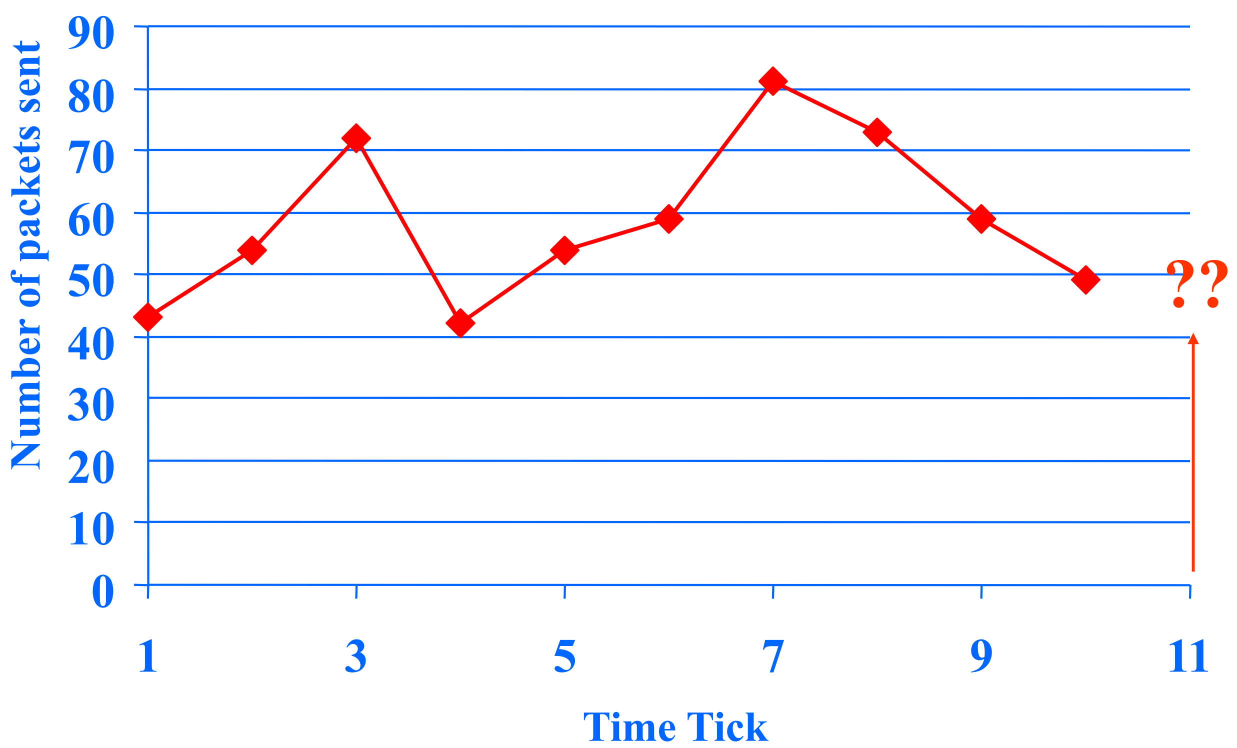
http://www.hfac.uh.edu/MediaFutures/thoughts.html

Outline

- Motivation
- Part 1: classical methods
  - ...
  - Linear Forecasting
    - Auto-regression: Least Squares; RLS
    - Co-evolving time sequences
    - Examples
    - Conclusions

Problem#2: Forecast

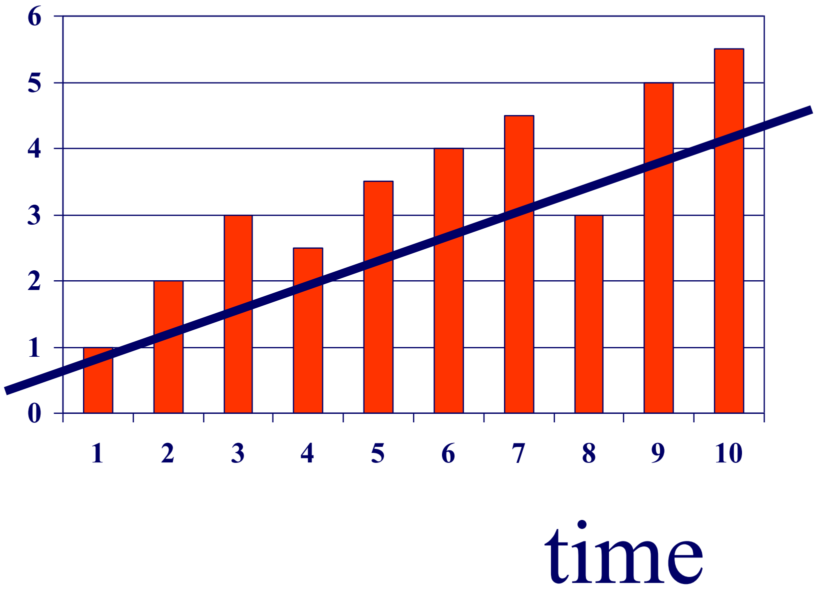
- Example: give x_{t-1}, x_{t-2}, …, forecast x_t



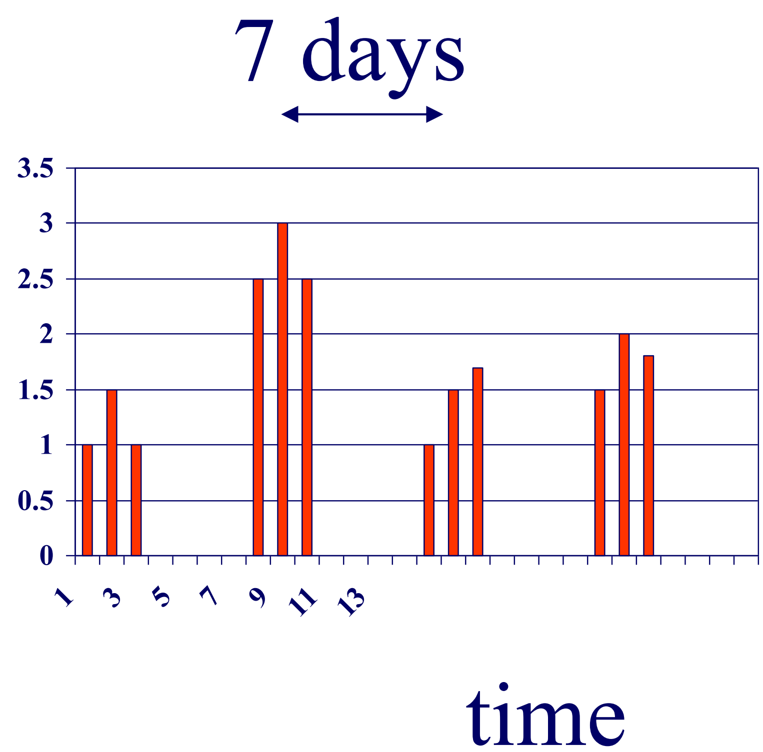
Forecasting: Preprocessing

MANUALLY:

remove trends



spot periodicities



Problem#2: Forecast

- Solution: try to express

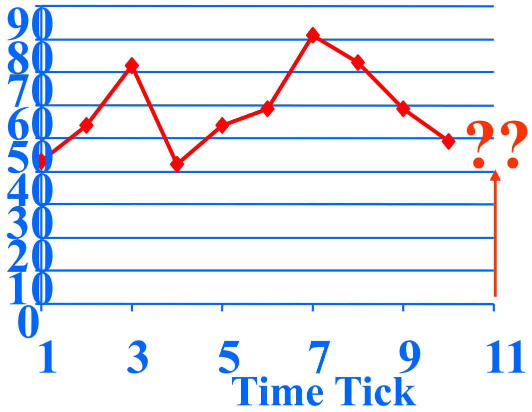
  x_t

  as a linear function of the past: x_{t-2}, x_{t-2}, …,

  (up to a window of w)

Formally:

$$\boxed{x_t \approx a_1 x_{t-1} + \ldots + a_w x_{t-w} + noise}$$



(Problem: Back-cast; interpolate)

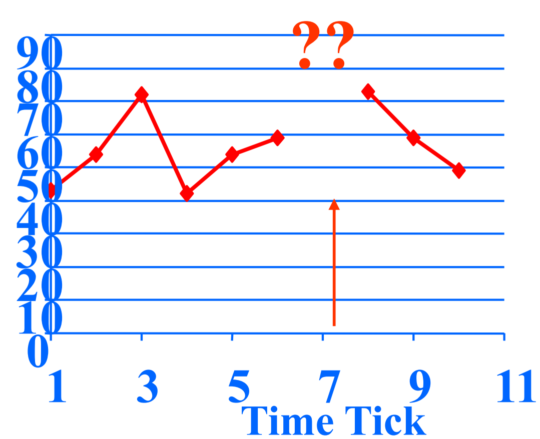
- Solution - interpolate: try to express

  x_t

  as a linear function of the past AND the future:

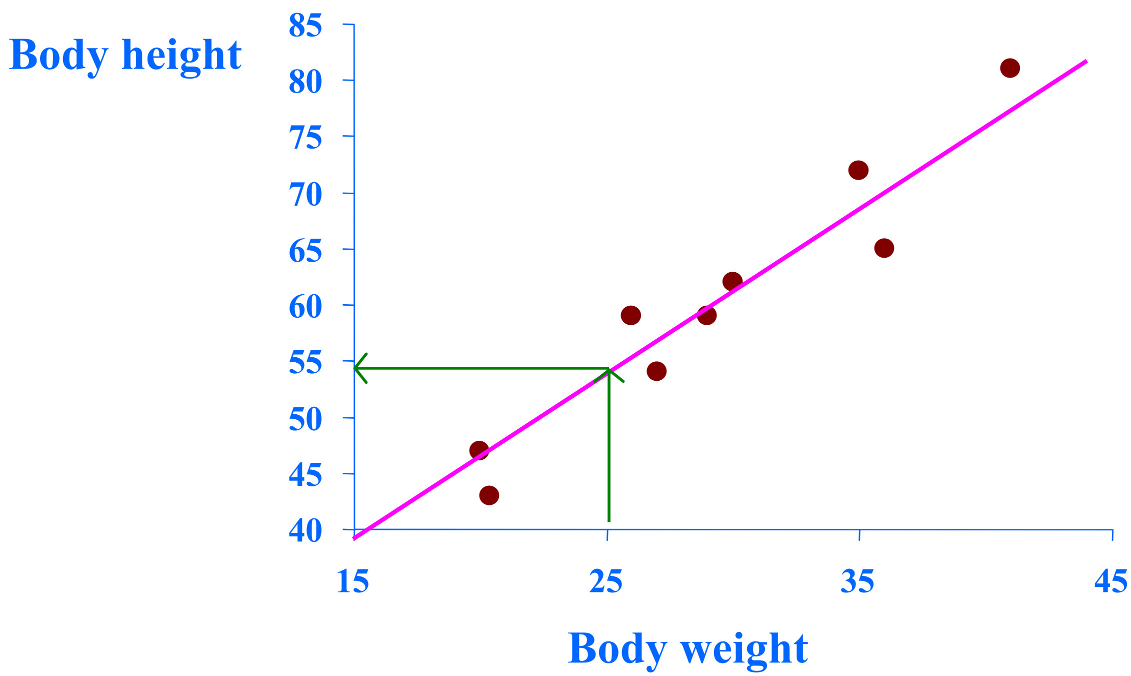
  x_{t+1}, x_{t+2}, … $x_{t+wfuture;}$ x_{t-1}, … $x_{t-wpast}$

  (up to windows of w_{past} , w_{future})
- EXACTLY the same algo’s



Linear Regression: idea

patient	*weight*	*height*
1	27	43
2	43	54
3	54	72
	…	
…		…
N	**25**	**??**



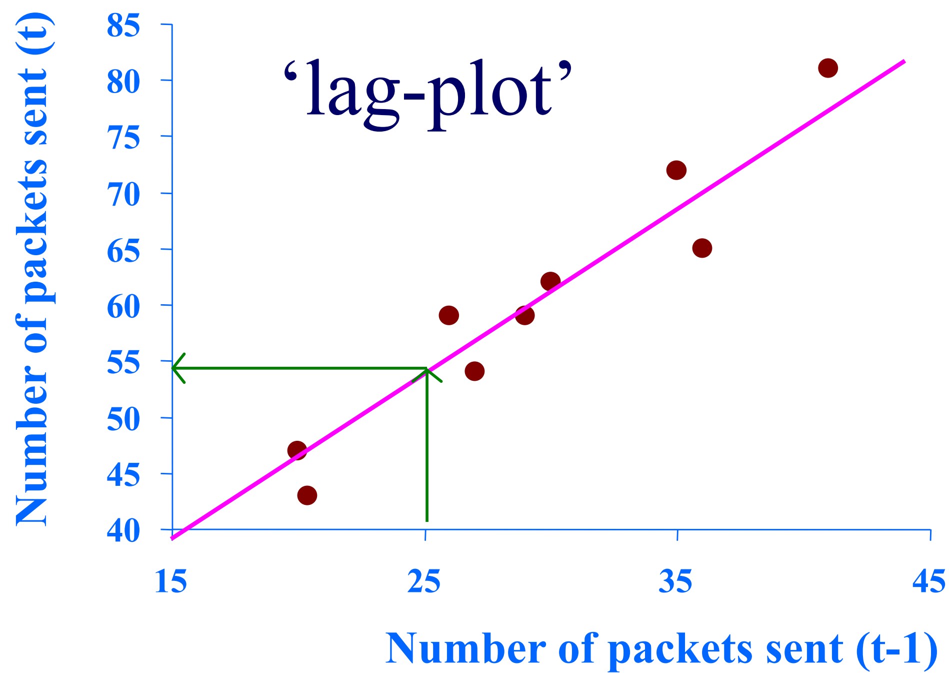
- express what we don't know (= 'dependent variable')
- as a linear function of what we know (= 'indep. variable(s)')

Linear Auto Regression:

Time	*Packets Sent(t)*
1	43
2	54
3	72
…	…
N	**??**

Linear Auto Regression:

Time	*Packets Sent (t-1)*	*Packets Sent(t)*
1	-	43
2	43	54
3	54	72
	…	
…		…
N	25	??



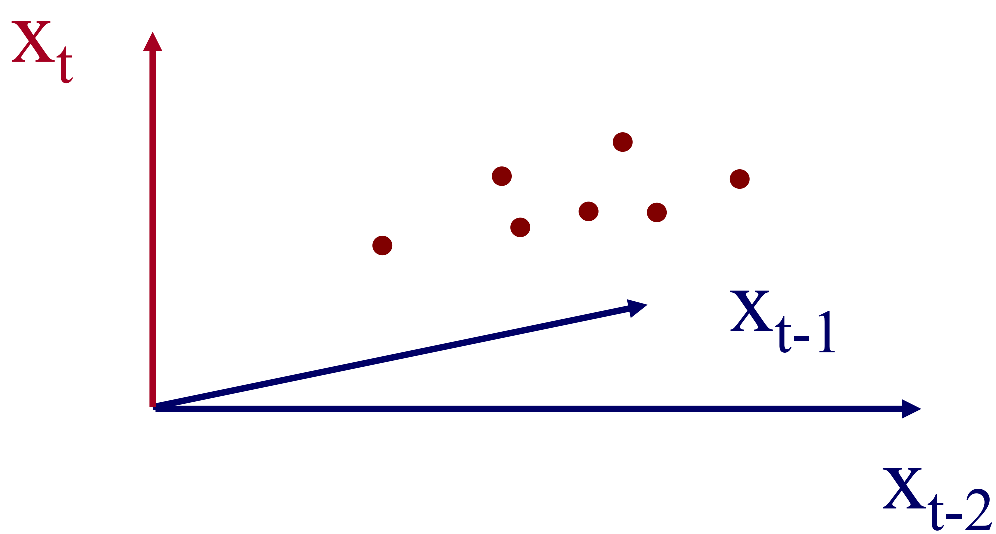
- lag w=1
- Dependent variable = # of packets sent (S [t])
- Independent variable = # of packets sent (S[t-1])

Outline

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

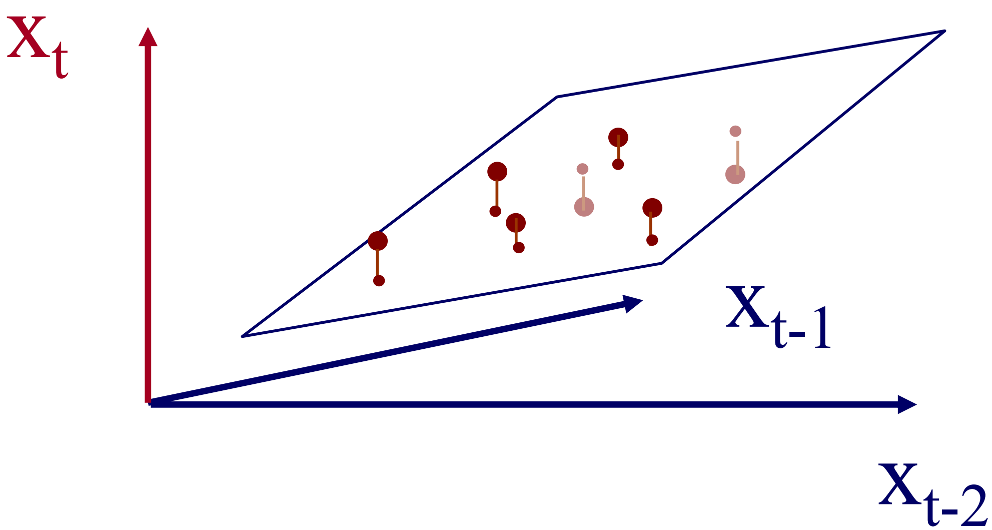
More details:

- Q1: Can it work with window w>1?
- A1: YES!



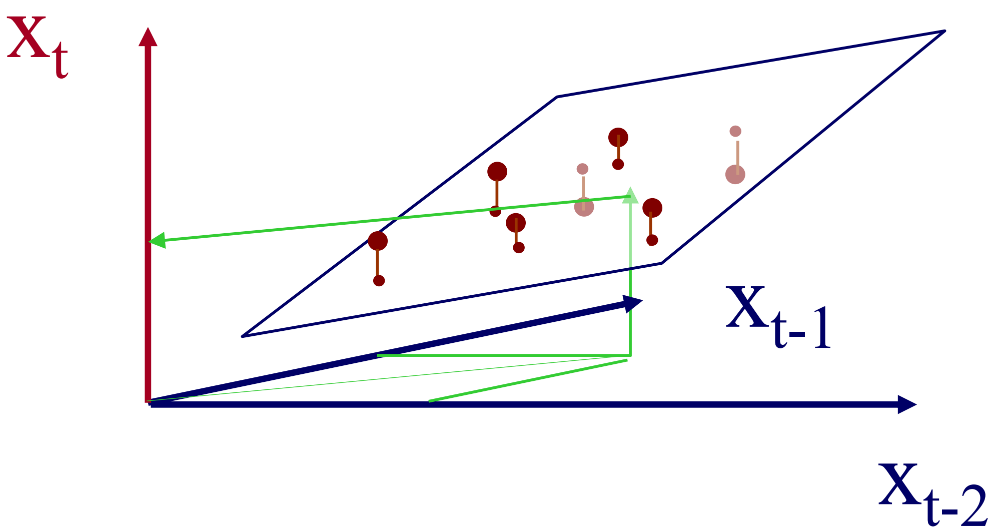
More details:

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



More details:

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



Skip

More details:

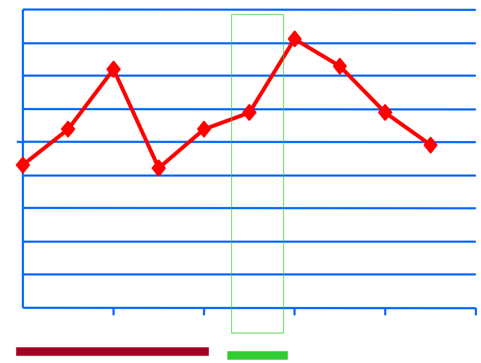
- Q1: Can it work with window w>1?
- A1: YES! The problem becomes:

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

- 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

Skip

More details:



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

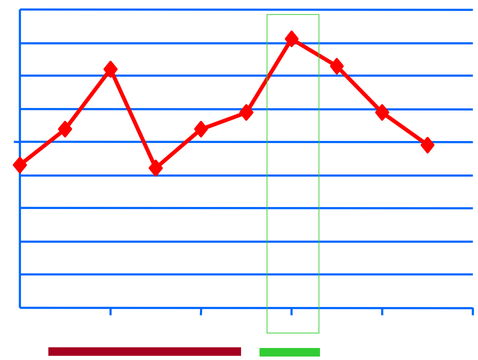
Ind-var1 Ind-var-w

time

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

Skip

More details:



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

Ind-var1 Ind-var-w

time

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

Skip

More details

- Q2: How to estimate $a_1, a_2, \ldots a_w = \mathbf{a}$?
- A2: with Least Squares fit

$$\boxed{\mathbf{a} = (\mathbf{X}^T \times \mathbf{X})^{-1} \times (\mathbf{X}^T \times \mathbf{y})}$$

- (Moore-Penrose pseudo-inverse)
- **a** is the vector that minimizes the RMSE from **y**

Skip

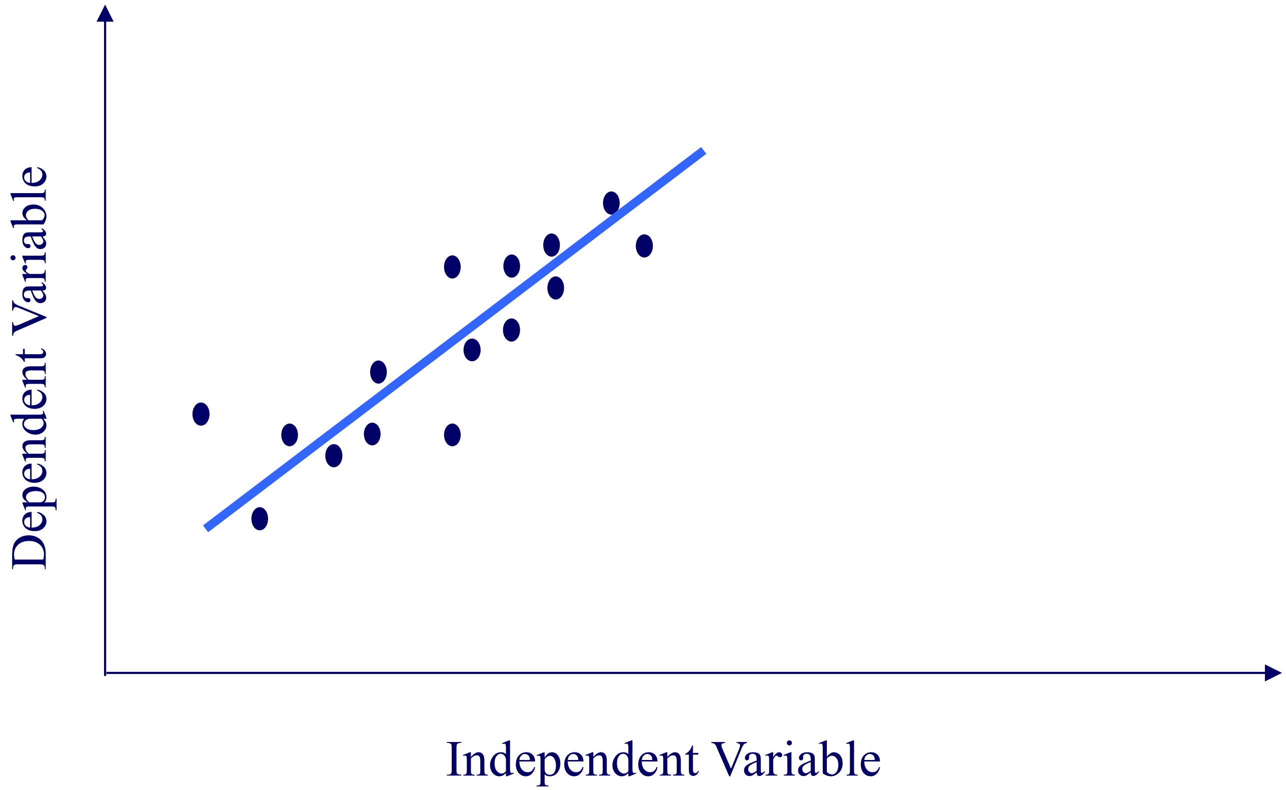
Even more details

- 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:

Skip

Even more details

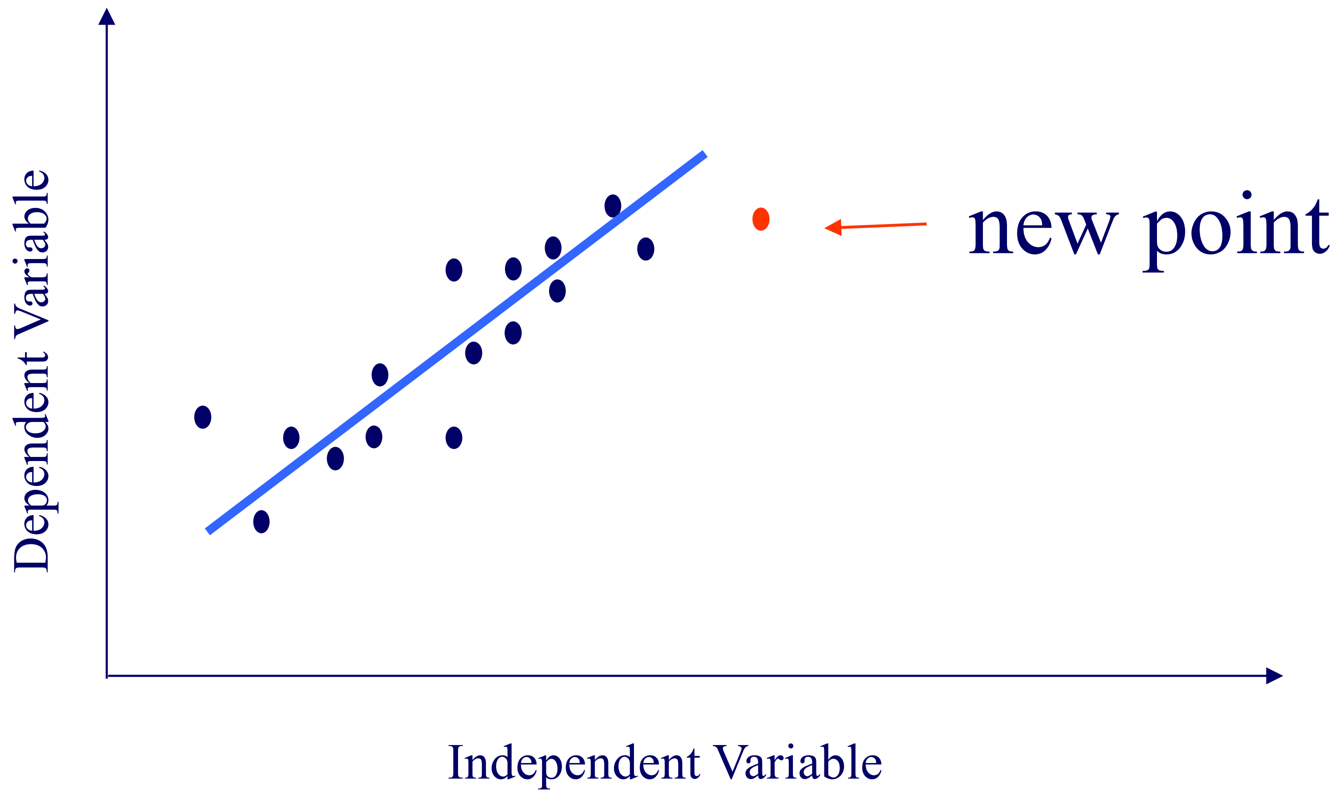
- Given:



Skip

Even more details

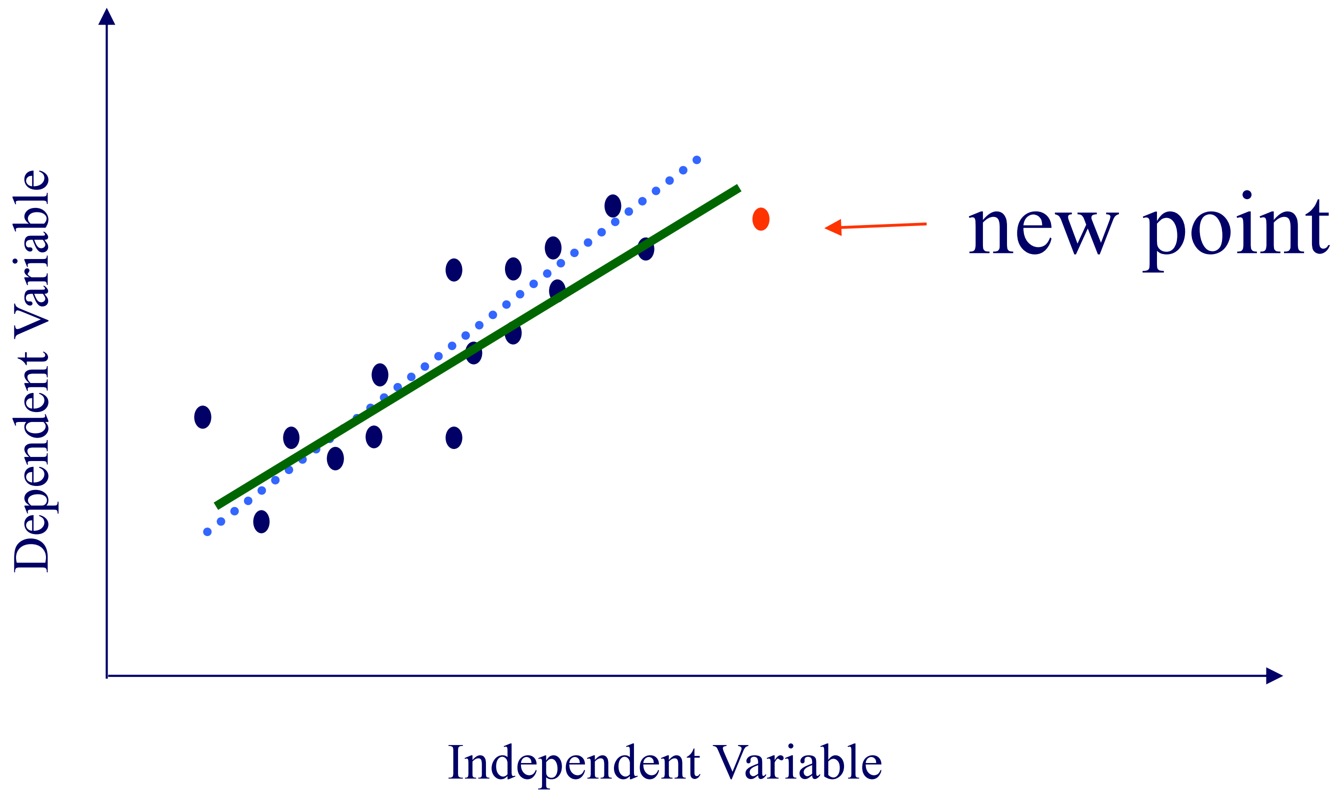
•



Skip

Even more details

RLS: quickly compute new best fit



Skip

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 $O(1 \times w^2)$

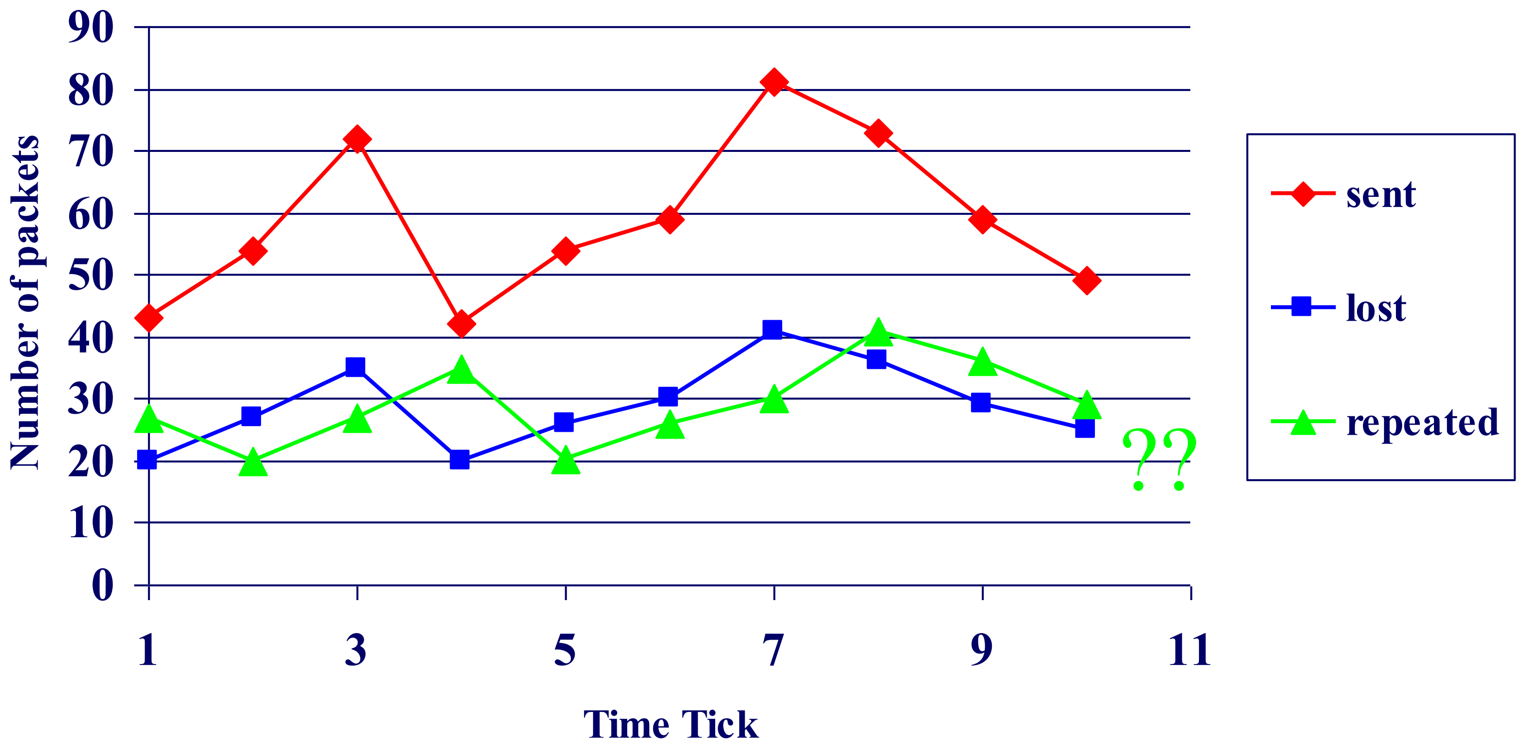
$N = 10^6, \quad w = 1\text{-}100$

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)**'



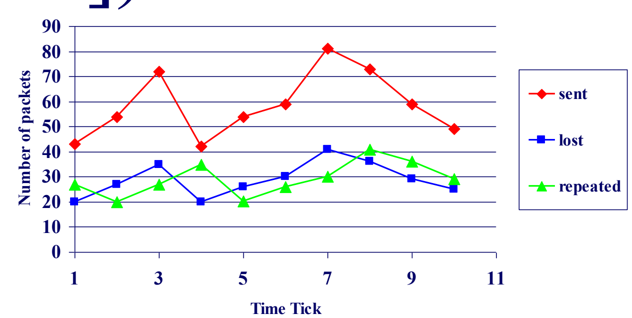
Solution:

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

- George E.P. Box and Gwilym M. Jenkins and Gregory C. Reinsel, *Time Series Analysis: Forecasting and Control*, Prentice Hall, 1994 (the classic book on ARIMA, 3rd ed.)
- Brockwell, P. J. and R. A. Davis (1987). Time Series: Theory and Methods. New York, Springer Verlag.

Additional Reading

- [Papadimitriou+ vldb2003] Spiros Papadimitriou, Anthony Brockwell and Christos Faloutsos *Adaptive, Hands-Off Stream Mining* VLDB 2003, Berlin, Germany, Sept. 2003
- [Yi+00] Byoung-Kee Yi et al.: *Online Data Mining for Co-Evolving Time Sequences*, ICDE 2000. (Describes MUSCLES and Recursive Least Squares)

Part 1.4: chaos and non-linear forecasting

Outline

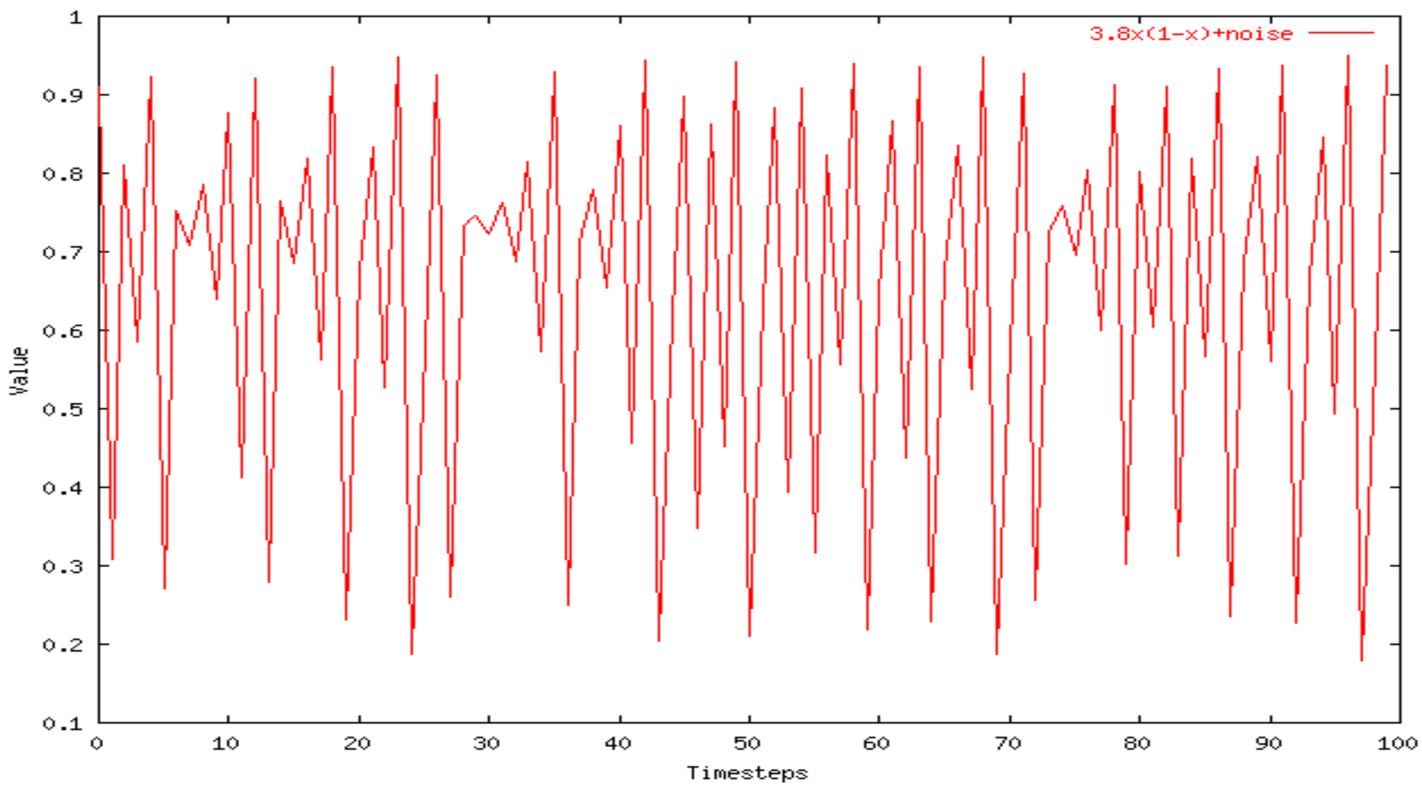
- Motivation
- Part 1: classical methods
  - Similarity Search and Indexing
  - DSP
  - Linear Forecasting
  - Non-linear forecasting
  - Tensors
  - Conclusions

Detailed Outline

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

Recall: Problem #1

Value



Time

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

How to forecast?

- ARIMA - but: linearity assumption

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

ARIMA pitfall

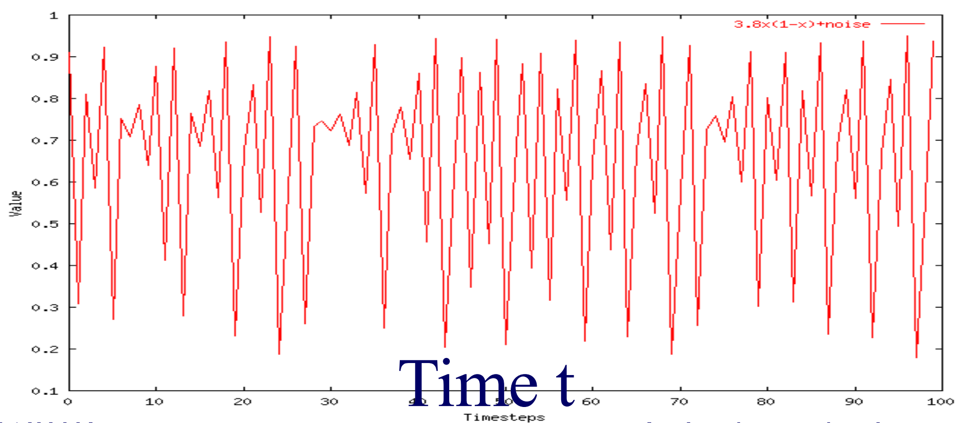
Example: logistic parabola

Models population of flies [R. May/1976]

$$\boxed{x_{t+1} = a x_t \cdot (1 - x_t)}$$

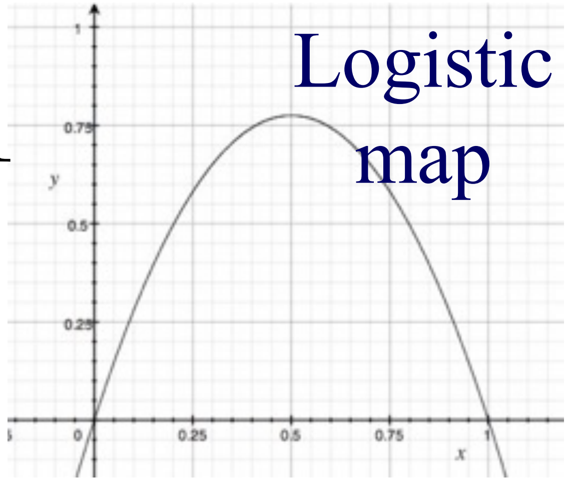
Time-series plot

x_t



Time t

x_{t+1}

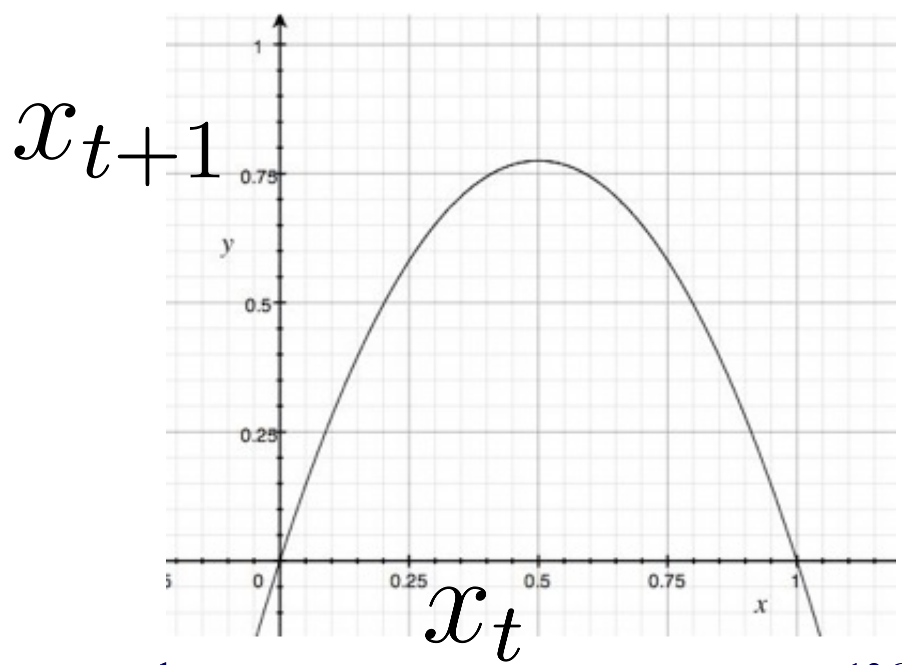


Logistic map

x_t

ARIMA pitfall

Linear equations, e.g., AR, ARIMA, …



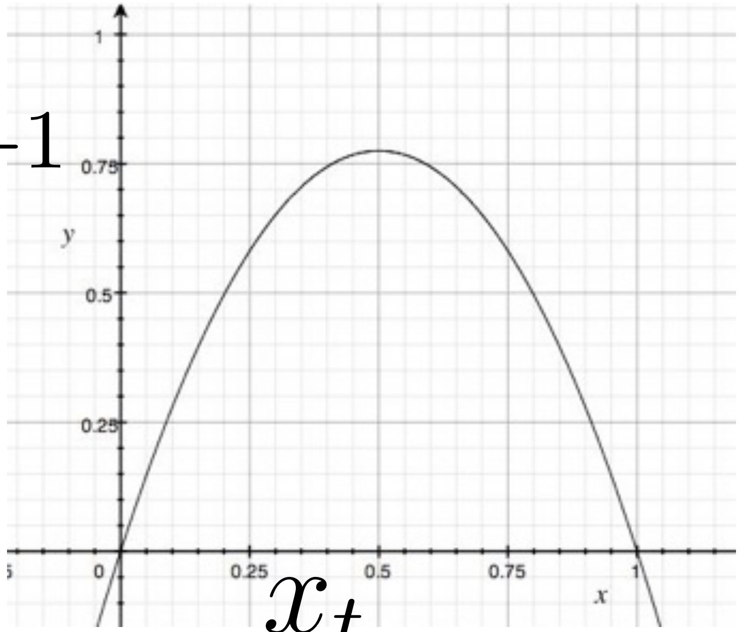
ARIMA pitfall

Linear equations, e.g., AR, ARIMA, …

e.g., AR(1)

$$x_{t+1} = a x_t + \epsilon$$

x_{t+1}



x_t

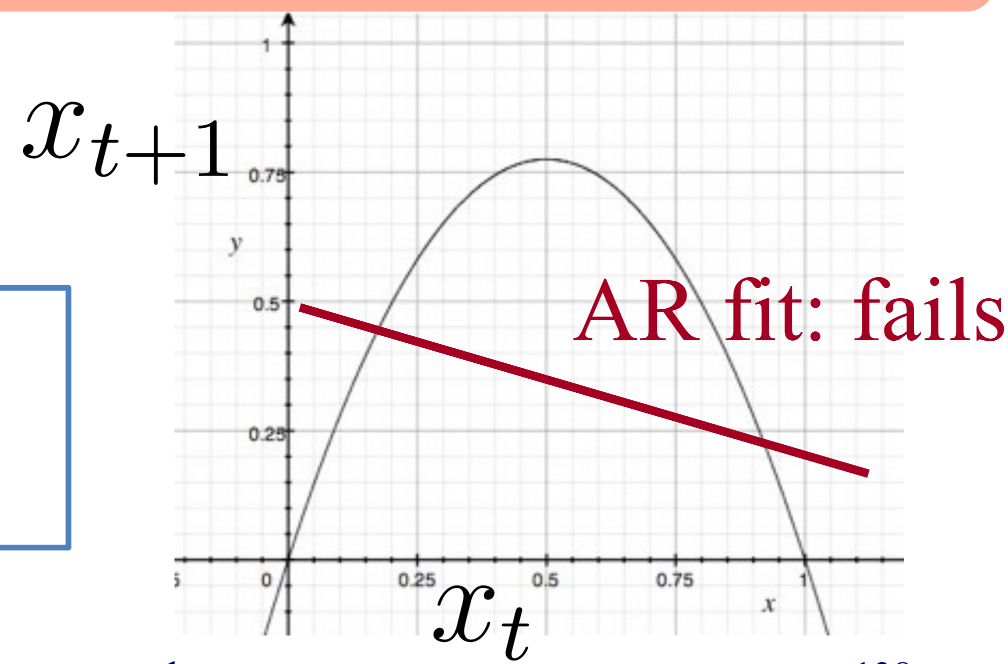
ARIMA pitfall

Linear equations, e.g., AR, ARIMA, …

but: linearity assumption

e.g., AR(1)

$$x_{t+1} = a x_t + \epsilon$$



Solution?

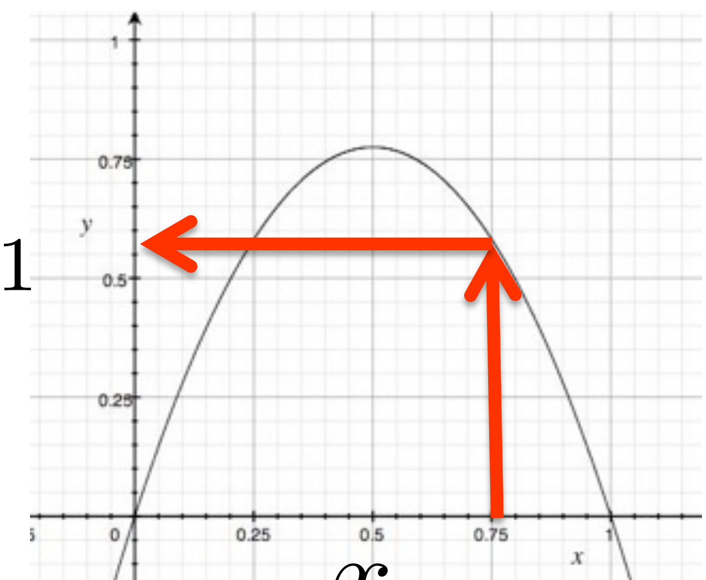
"Delayed Coordinate Embedding"

= Lag Plots

[Sauer92]

k-nearest neighbor search

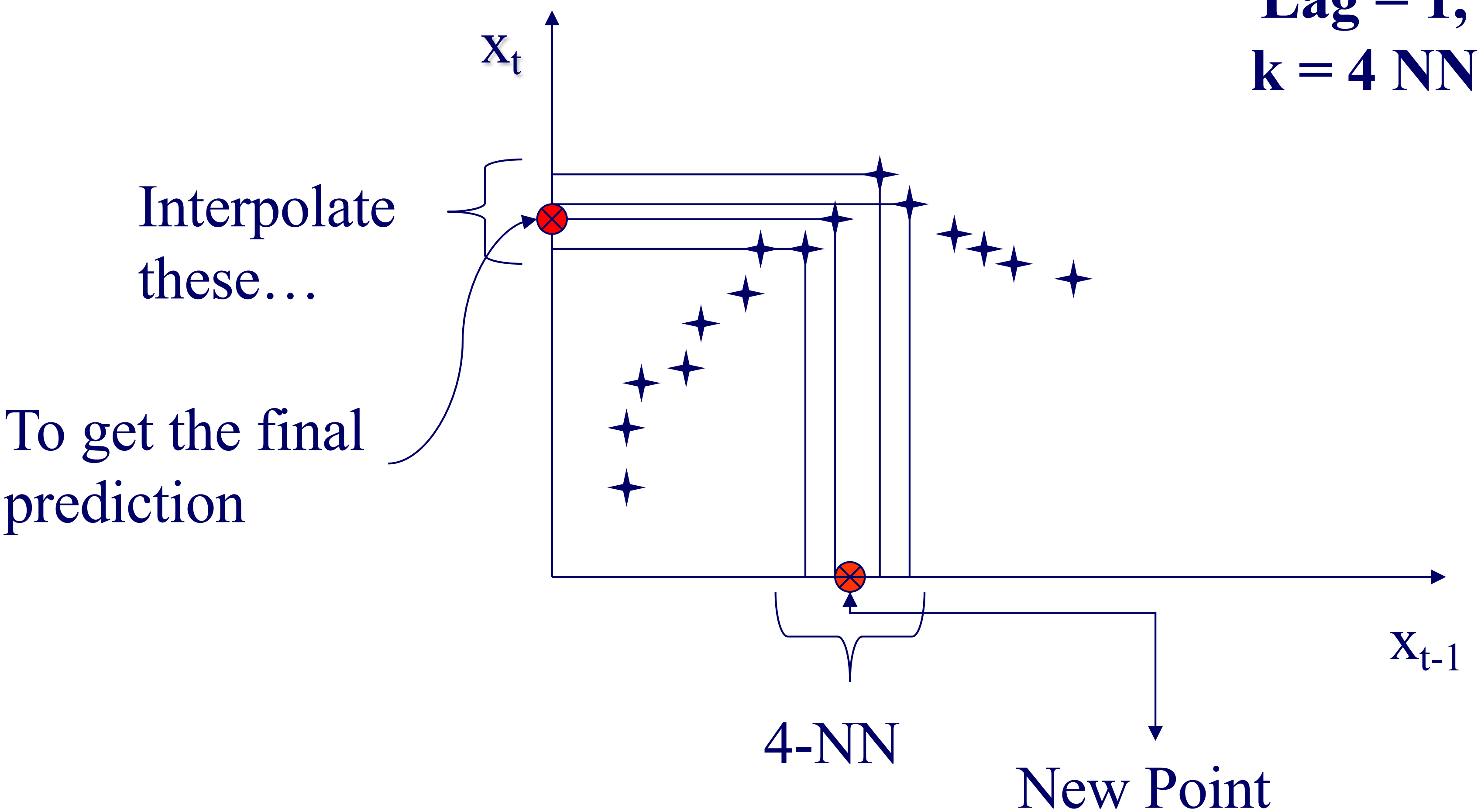
x_{t+1}



x_t

General Intuition (Lag Plot)

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?

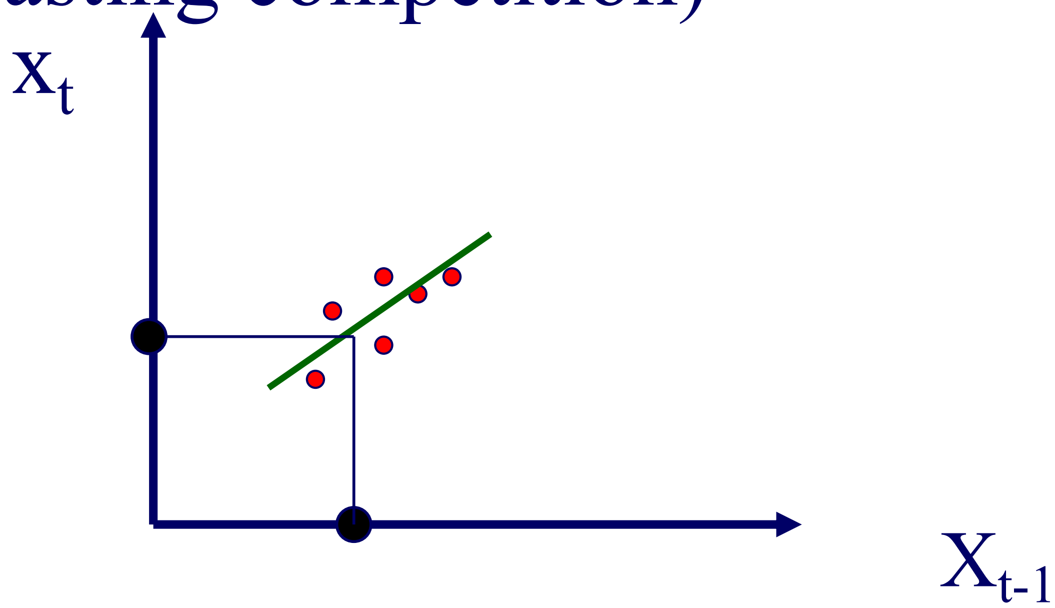
How do we interpolate between the *k* nearest neighbors?

A3.1: Average

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

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!

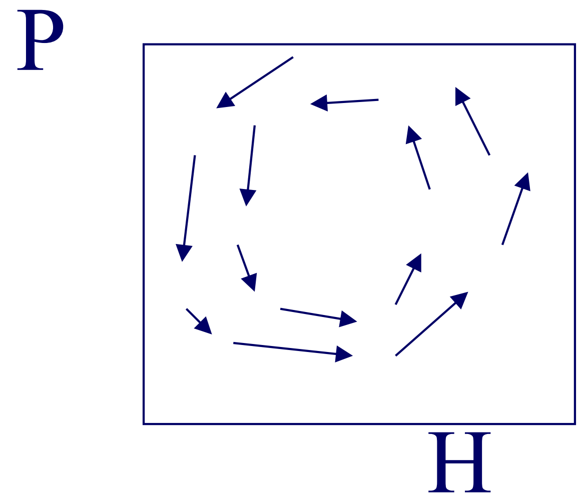
Theoretical foundation

- 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

$$dH/dt = r\,H - a\,H*P$$
$$dP/dt = b\,H*P - m\,P$$



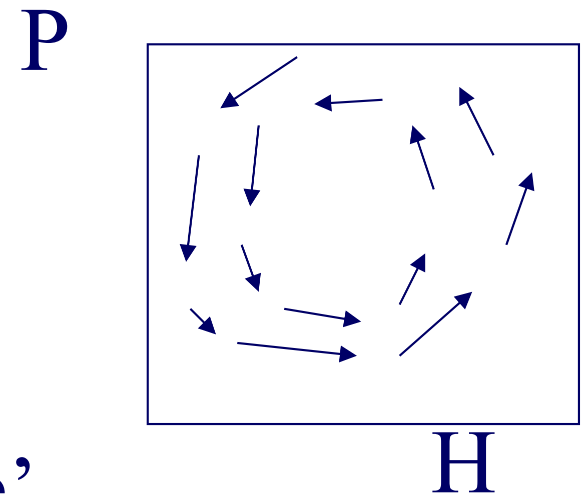
H is count of prey (e.g., hare)
P is count of predators (e.g., lynx)

Suppose only P(t) is observed (t=1, 2, …).

Theoretical foundation

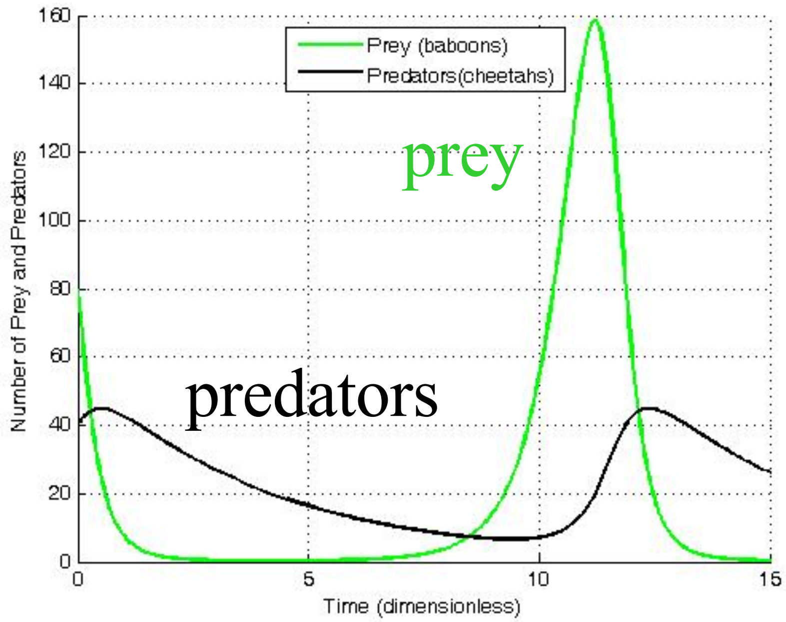
Example: Lotka-Volterra equations

$$dH/dt = r\ H - a\ H*P$$
$$dP/dt = b\ H*P - m\ P$$

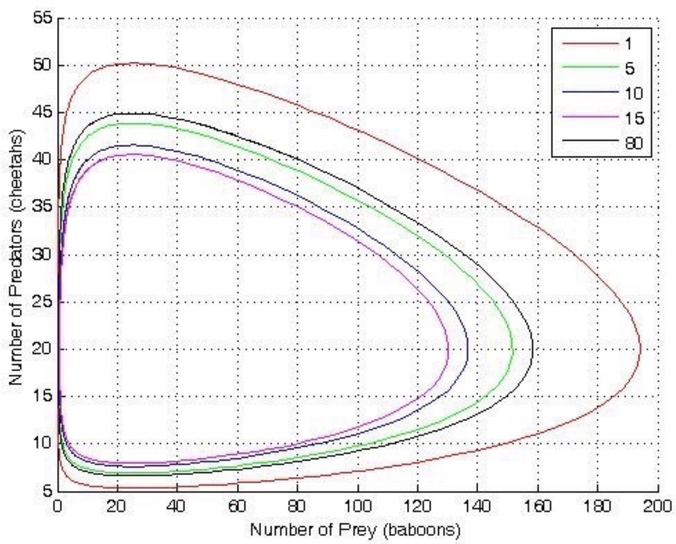


intuition: ‘eliminate the H variable’

Solution to Volterra-Lotka eq.



predators



prey

time

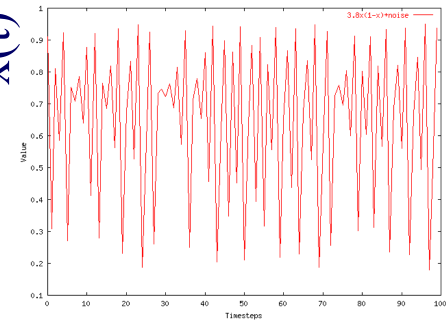
from wikipedia

Detailed Outline

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

Datasets

x(t)

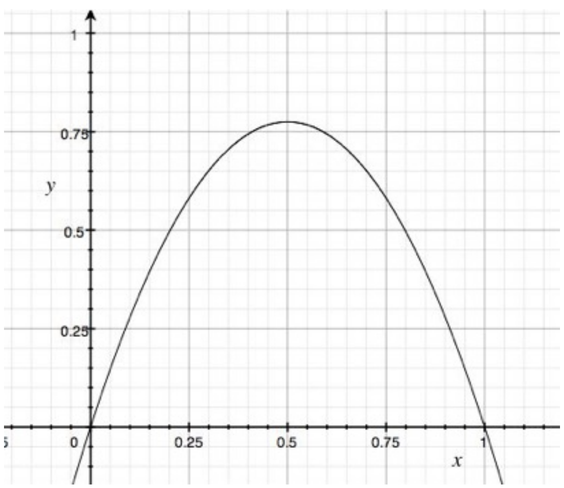


time

Logistic Parabola:

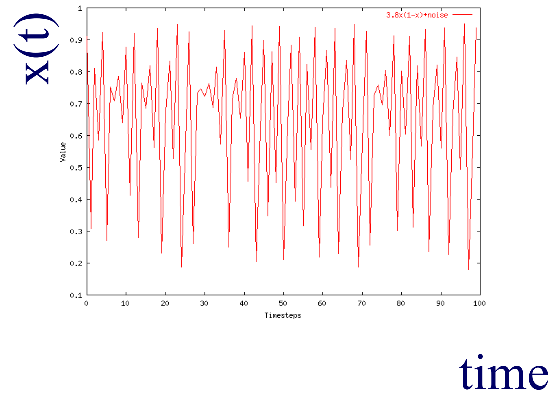
$x_t = a x_{t-1}(1 - x_{t-1}) + \text{noise}$

Models population of flies [R. May/1976]



Lag-plot

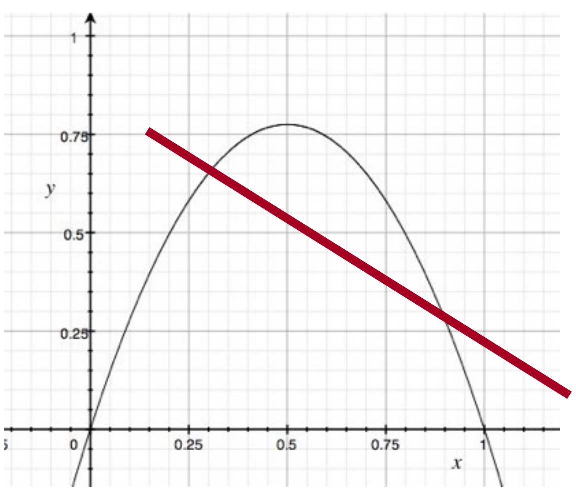
Datasets



Logistic Parabola:

$x_t = ax_{t-1}(1-x_{t-1}) + \text{noise}$

Models population of flies [R. May/1976]



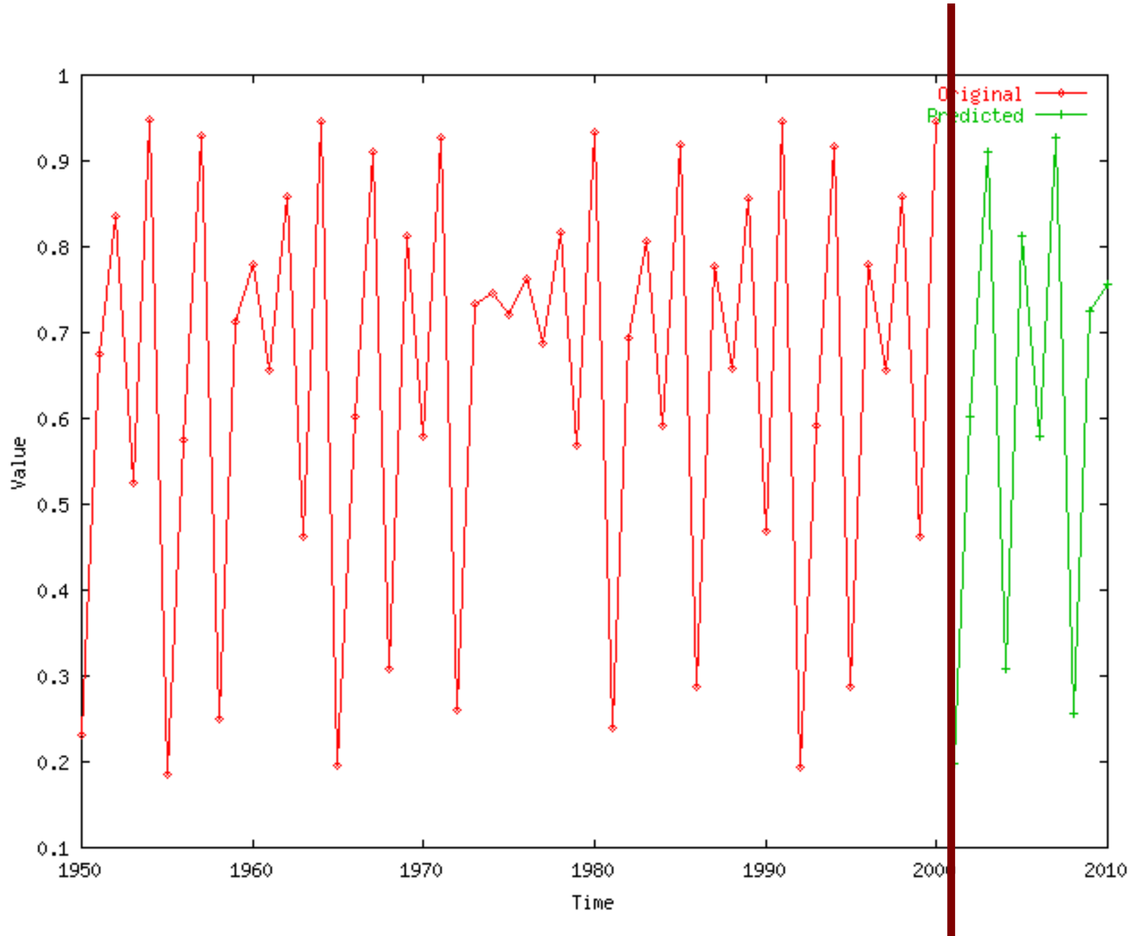
Lag-plot

ARIMA: fails

Logistic Parabola

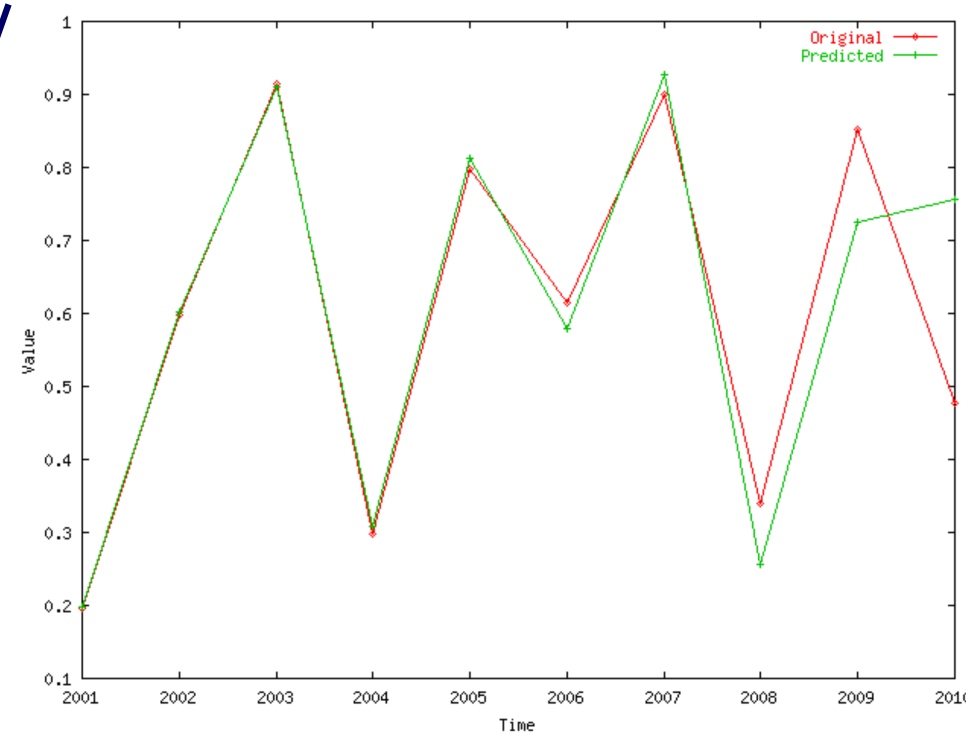
Our Prediction from here

Value



Timesteps

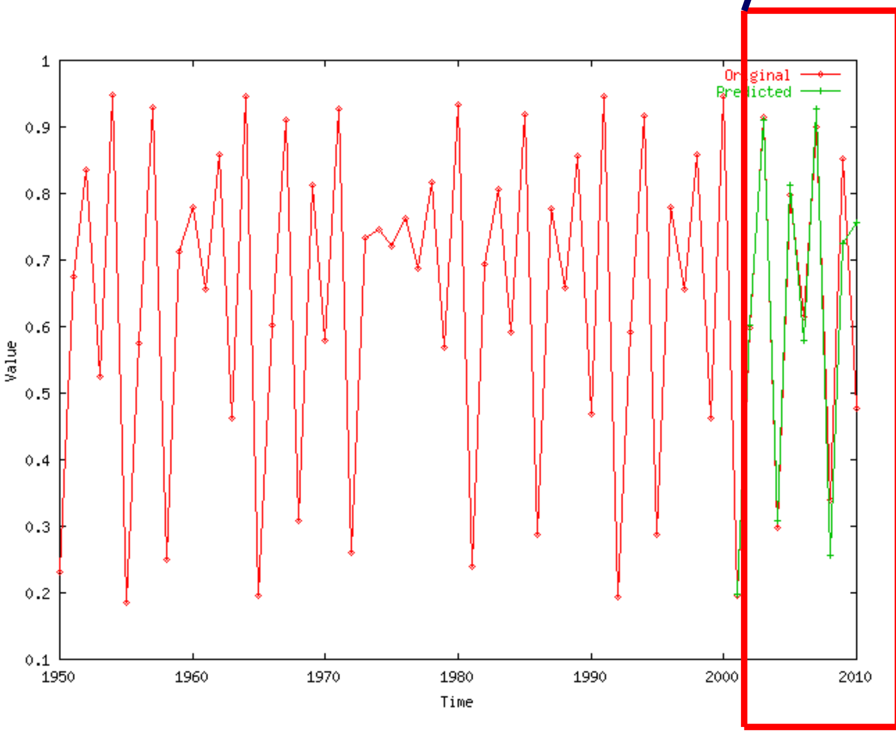
Value



Logistic Parabola

Comparison of prediction
to correct values

Timesteps



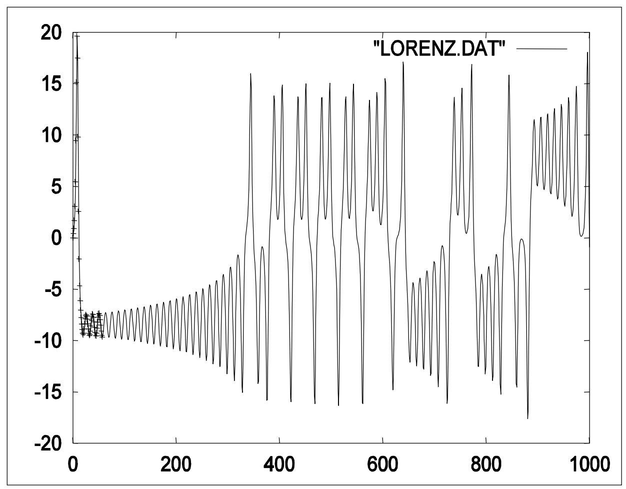
Datasets

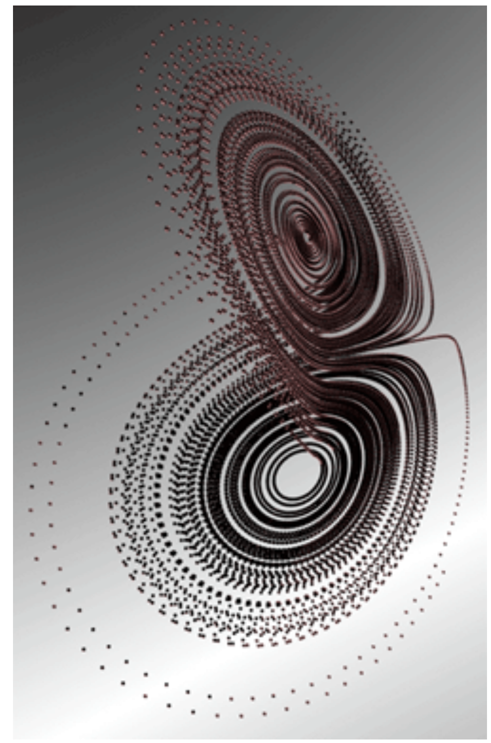
LORENZ: Models convection currents in the air

$$dx / dt = a (y - x)$$

$$dy / dt = x (b - z) - y$$

$$dz / dt = xy - c z$$





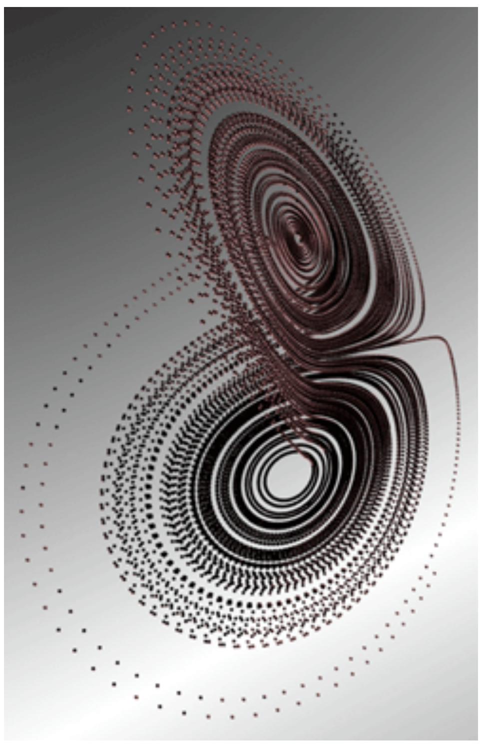
Equations -> prediction (?)

- ~1950s conventional wisdom: '*if we know the equations of a system, we can predict its evolution*'
  - Thus, weather prediction is within reach

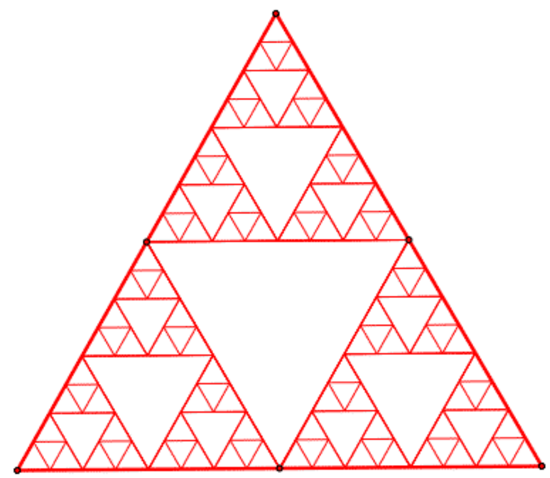
- Lorenz: **not necessarily**, if the equations are non-linear
- 'butterfly effect'

‘strange attractor’ - fractal

Dim: ~2.06

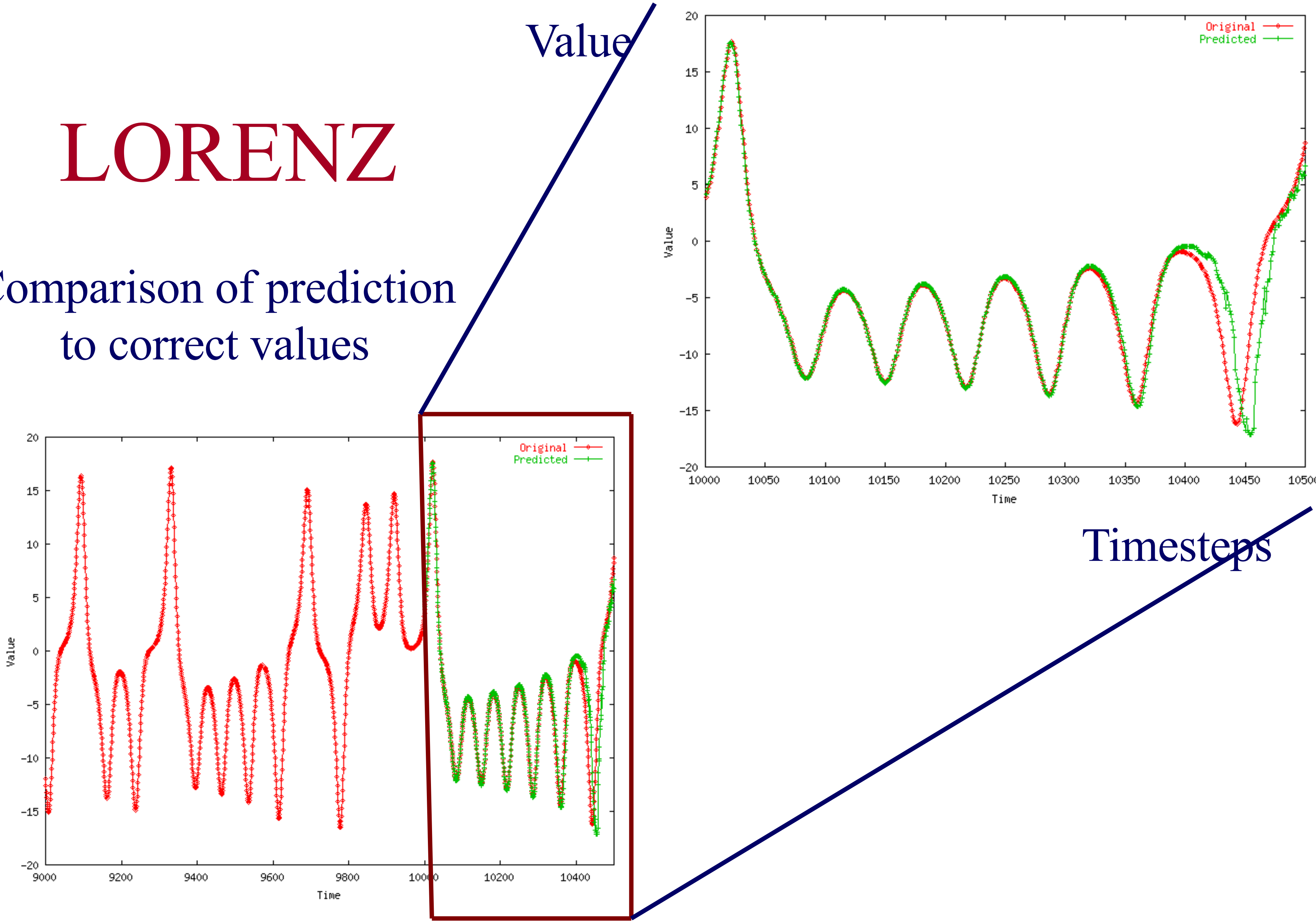


Dim: ~1.58



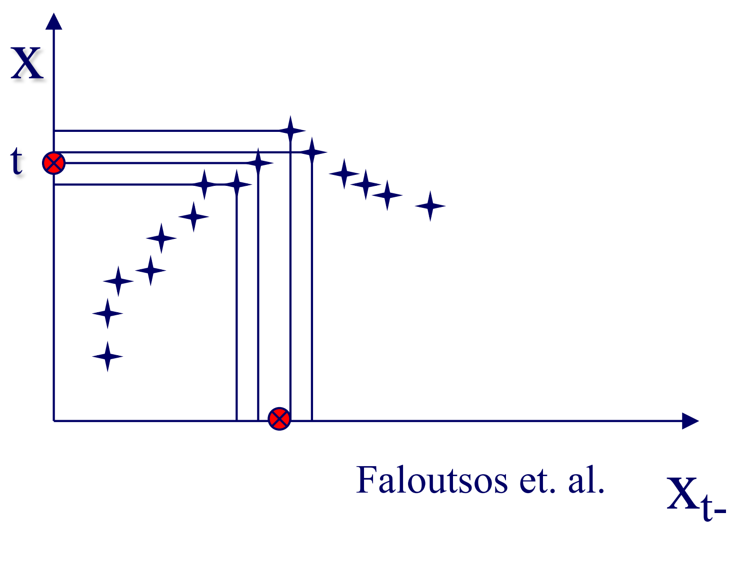
LORENZ

Comparison of prediction to correct values



Conclusions

- Lag plots for non-linear forecasting (Takens' theorem)
- suitable for 'chaotic' signals



References

- Deepay Chakrabarti and Christos Faloutsos *F4: Large-Scale Automated Forecasting using Fractals* CIKM 2002, Washington DC, Nov. 2002.
- Sauer, T. (1994). *Time series prediction using delay coordinate embedding*. (in book by Weigend and Gershenfeld, below) Addison-Wesley.
- Takens, F. (1981). *Detecting strange attractors in fluid turbulence*. Dynamical Systems and Turbulence. Berlin: Springer-Verlag.

References

- Weigend, A. S. and N. A. Gerschenfeld (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison Wesley. (Excellent collection of papers on chaotic/non-linear forecasting, describing the algorithms behind the winners of the Santa Fe competition.)

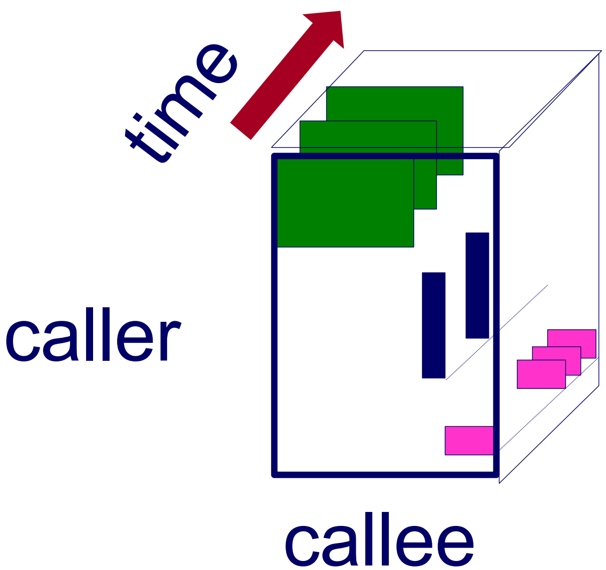
Part 1.5: Tensors – time evolving graphs

Outline

- Motivation
- Part 1: Classical methods
  - Similarity Search and Indexing
  - DSP
  - Linear Forecasting
  - Non-linear forecasting
  - Tensors
  - Conclusions

Problem: co-evolving graphs

- How to forecast?
  - 4M x 4M x 15 days



A: tensors

- Q: what is a tensor?

Tensor examples

- A: N-D generalization of matrix:

arxiv' 17

	data	mining	classif.	tree	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

Tensor examples

- A: N-D generalization of matrix:

arxiv' 19

arxiv' 18

arxiv' 17

	data	mining	classif.	tree	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

Tensors are useful for 3 or more modes

Terminology: ‘mode’ (or ‘aspect’):

Mode#3

data	mining	classif.	tree	...
13	11	22	55	...
5	4	6	7	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

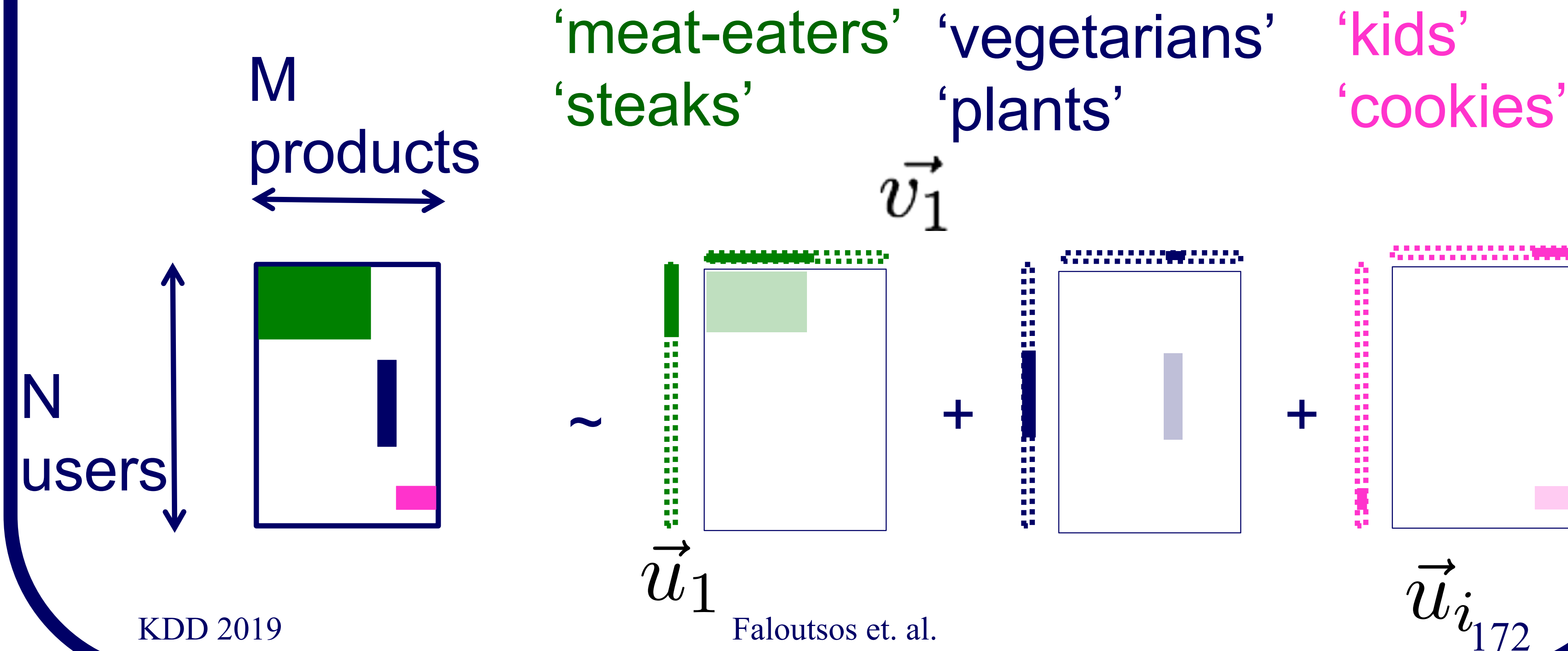
Mode#2

Mode (== aspect) #1

Tensor Basics

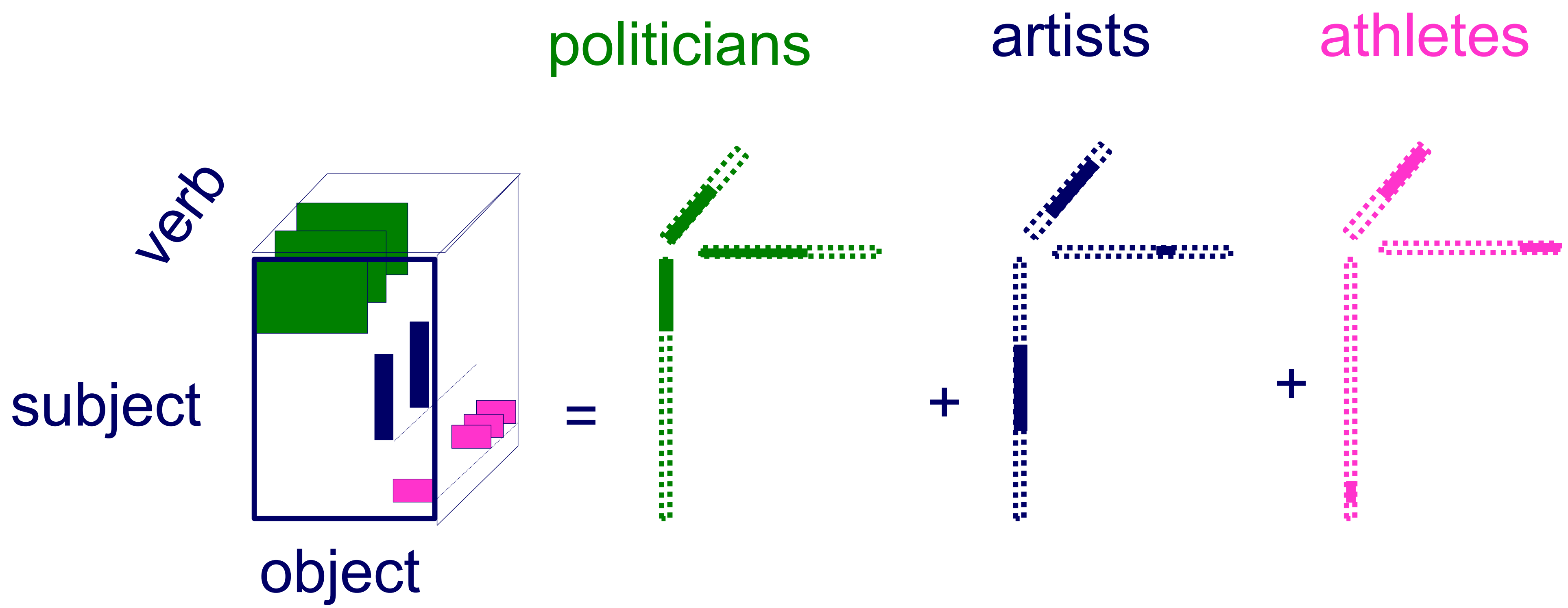
Tensor factorization

- Recall: (SVD) matrix factorization: finds blocks



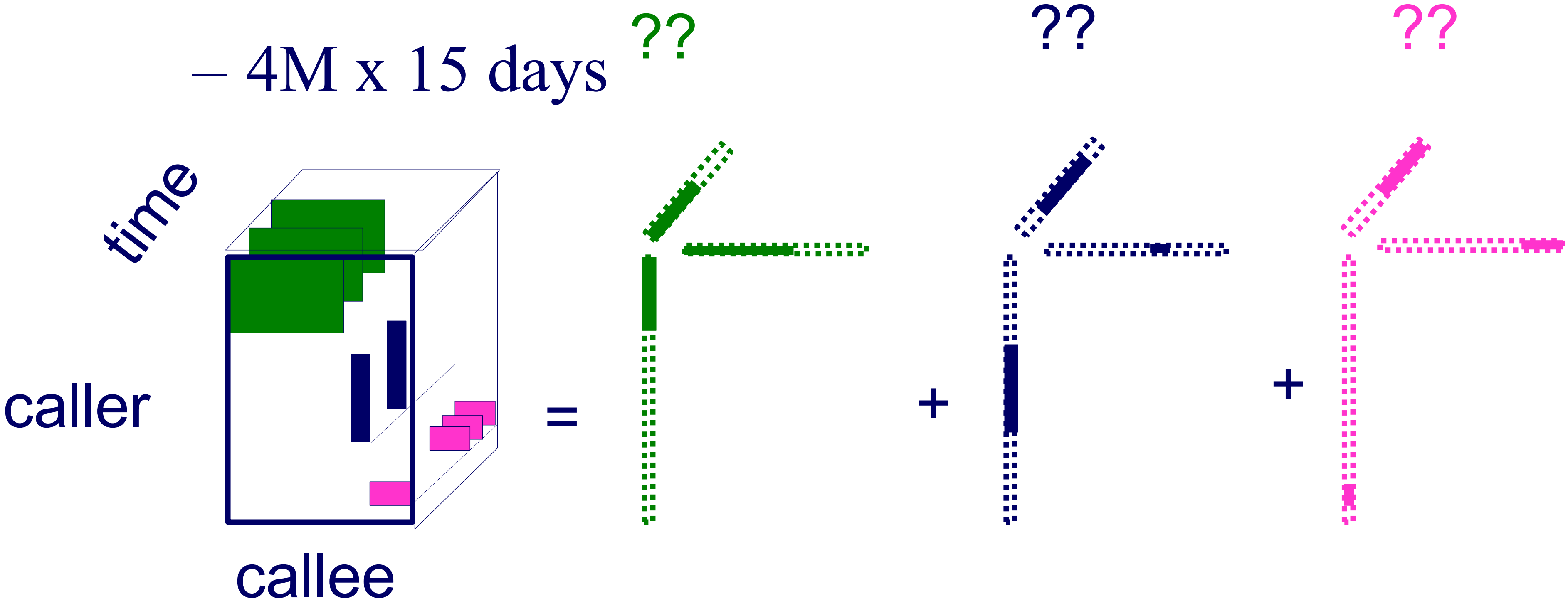
Tensor factorization

- PARAFAC decomposition



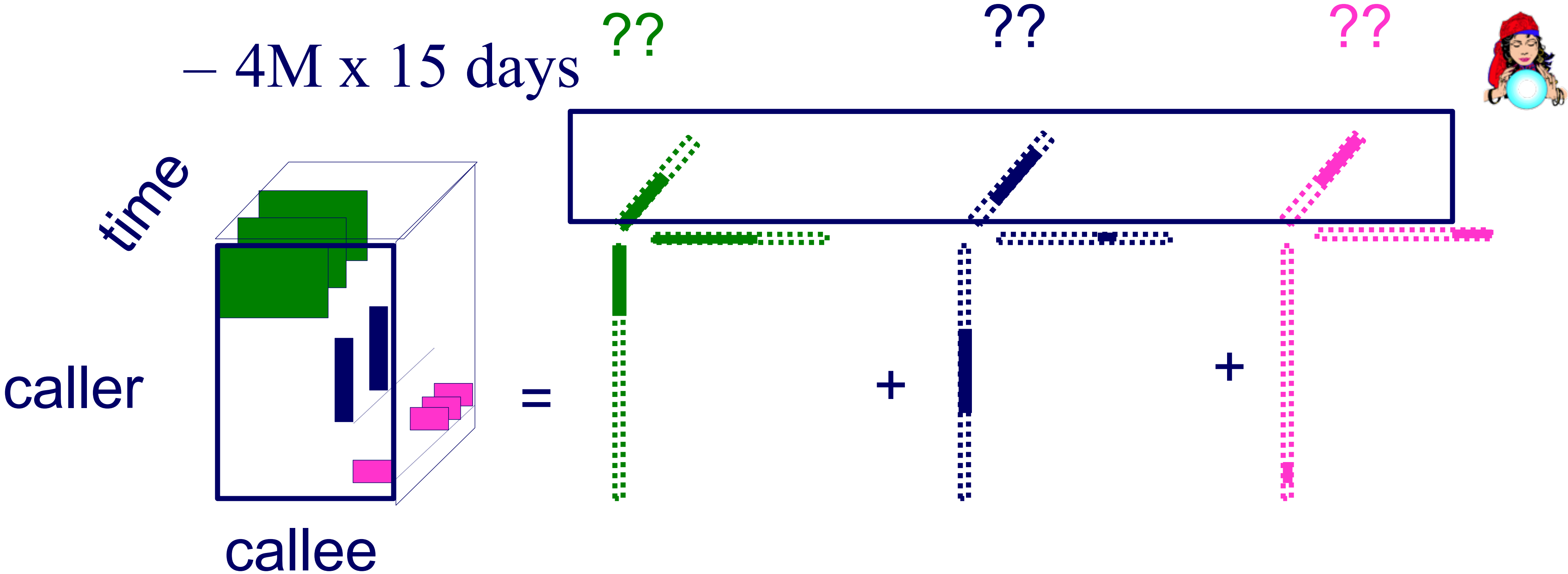
Tensor factorization

- PARAFAC decomposition
- Results for who-calls-whom-when
  - 4M x 15 days



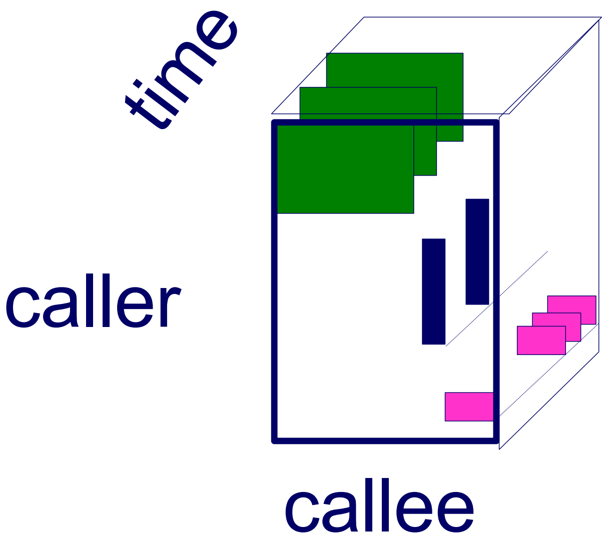
Tensor factorization

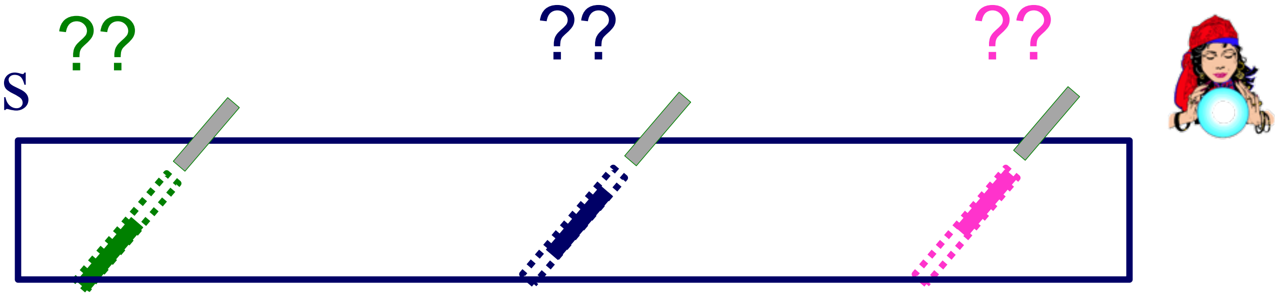
- PARAFAC decomposition
- Results for who-calls-whom-when
  - 4M x 15 days



Tensor factorization

- PARAFAC decomposition
- Results for who-calls-whom-when
  - 4M x 15 days

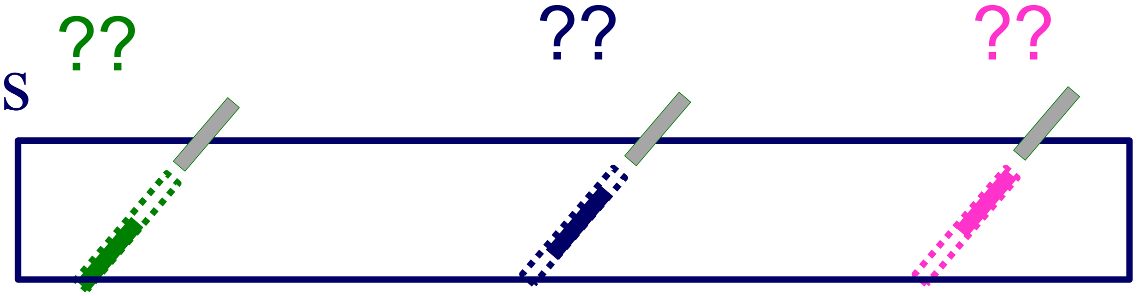


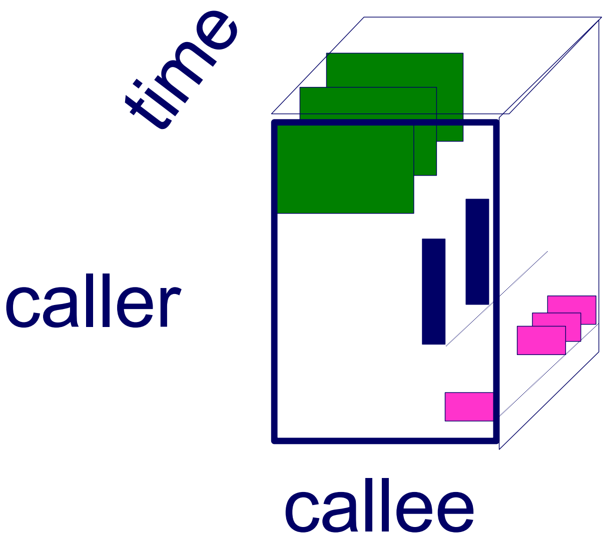


Forecast in, eg, 3, instead of 1M*1M series

Tensor factorization

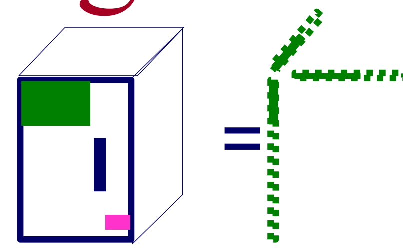
- PARAFAC decomposition
- Results for who-calls-whom-when
  - 4M x 15 days





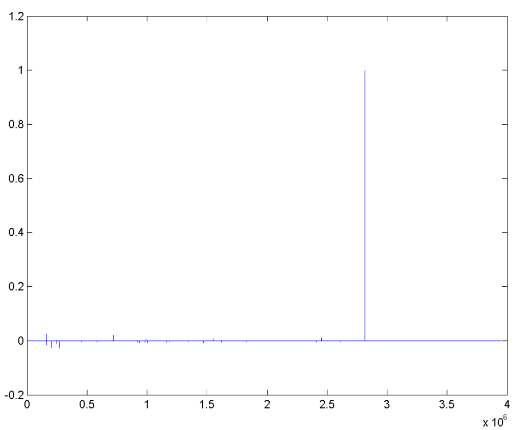
ICDM17: *TensorCast: Forecasting with Context Using Coupled Tensors*
M. Araujo, et al

Detection in time-evolving graphs

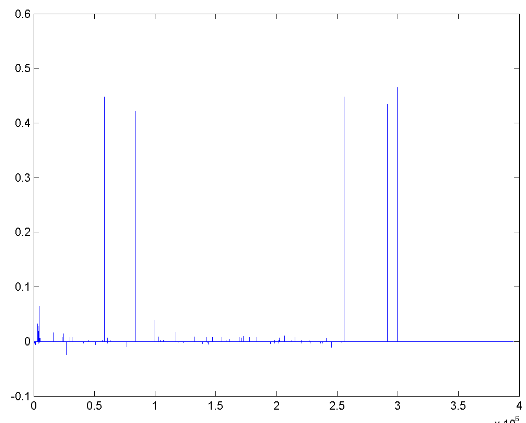


- Strange communities in phone call data:
  - European country, 4M clients, data over 2 weeks

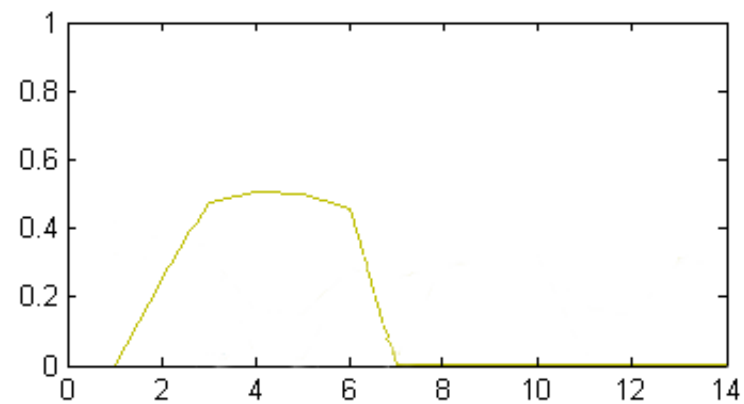
1 caller



5 receivers

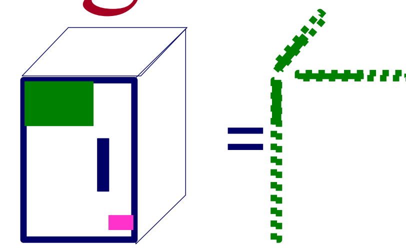


4 days of activity



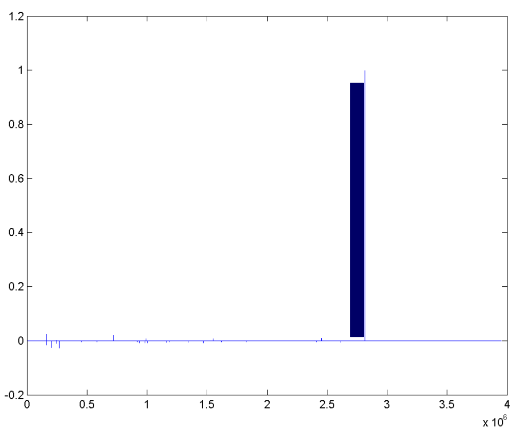
~200 calls to EACH receiver on EACH day!

Detection in time-evolving graphs

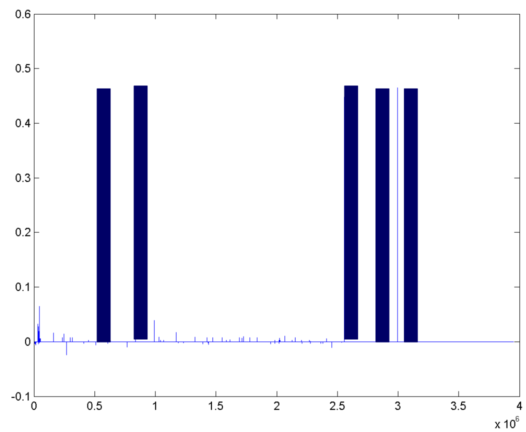


- Strange communities in phone call data:
  - European country, 4M clients, data over 2 weeks

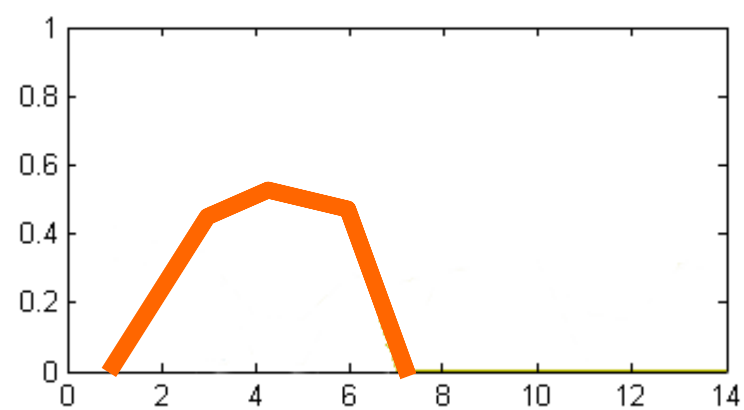
1 caller



5 receivers



4 days of activity

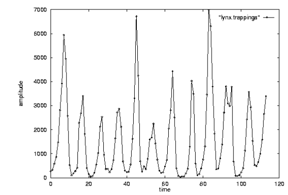


~200 calls to EACH receiver on EACH day!

Overall conclusions

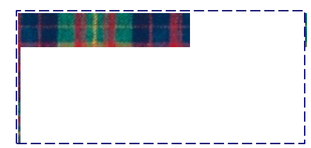
- P1.1. Similarity search: **Euclidean**/time-warping; **feature extraction** and **SAMs**
- P1.2. Signal processing: **DFT**, **DWT** are powerful tools
- P1.3. Linear Forecasting: **AR** (Box-Jenkins)
- P1.4. Non-linear forecasting: **lag-plots** (Takens)
- P1.5. **Tensors**: PARAFAC etc

Important observations



Patterns, rules, forecasting and similarity indexing are closely related:

- To do forecasting, we need
  - to find patterns/rules
  - compress
  - to find similar settings in the past

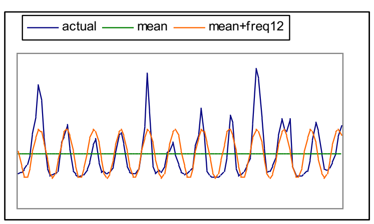


- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)

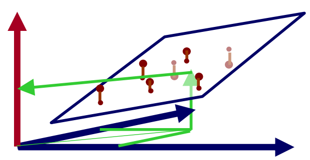
THANK YOU!

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

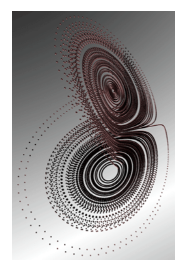
DFT



AR



Non-lin./
chaos



tensors