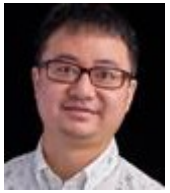
Carnegie Mellon

amazon

Forecasting Big Time Series: Theory and Practice



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


https://lovvge.github.io/Forecasting-Tutorial-WWW-2020/

Overall Outline

- Part 1: Fundamentals
- Part 2:
  - NN – from old to new
  - CNNs / Wavenet / Transformer-based models
  - Deep Probabilistic Models
  - System, practice, and the AWS Forecasting Stack

Part 1 – Fundamentals - Outline

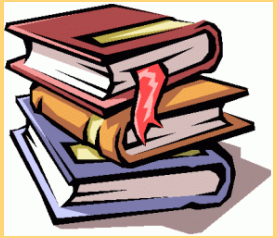
- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP (Digital Signal Processing)
- P1.3. Linear Forecasting
- P1.4. Non-linear forecasting
- P1.5. Tensors
- Conclusions

‘Recipe’ Structure:

- Problem definition

- Short answer/solution

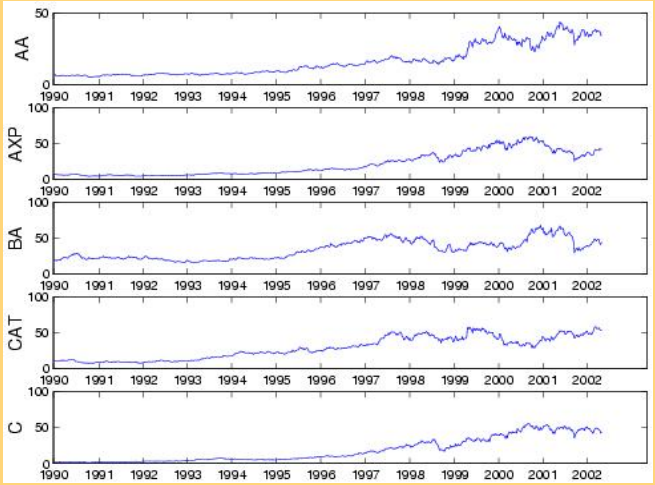
- LONG answer – details



- Conclusion/short-answer

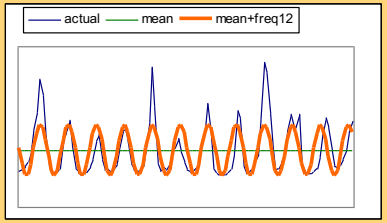
Problem:

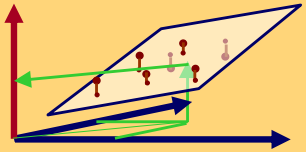
Q: mine/forecast (one, or more) time sequences

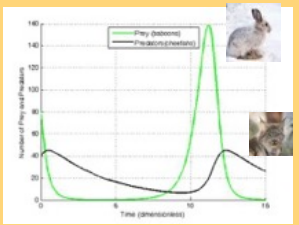


Answers

- P1.1. Similarity search: **Euclidean**/time-warping; **feature extraction** and **SAMs**
- P1.2. Periodicities: **DFT/DWT**
- P1.3. Linear Forecasting: **AR** (Box-Jenkins)
- P1.4 Non-linear forecasting: **lag-plots**
  - Gray-box modeling: **Lotka-Volterra**
- P1.5. Tensors: **PARAFAC**

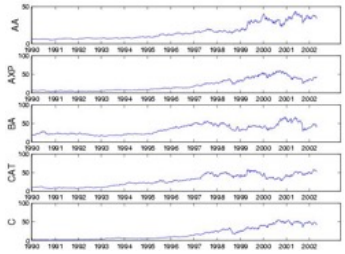


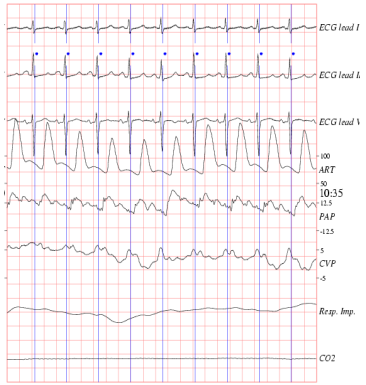




Motivation - Applications

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





Carnegie Mellon

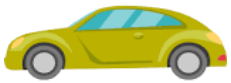
amazon

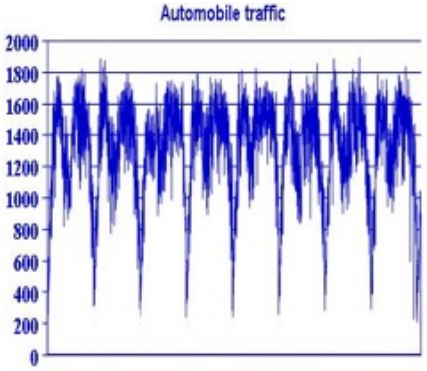
EEG - epilepsy

from wikipedia

Motivation - Applications (cont'd)

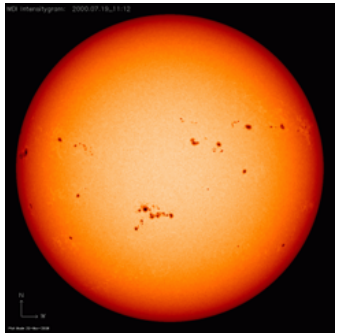
- civil/automobile infrastructure
  - bridge vibrations [Oppenheim+02]
  - road conditions / traffic monitoring

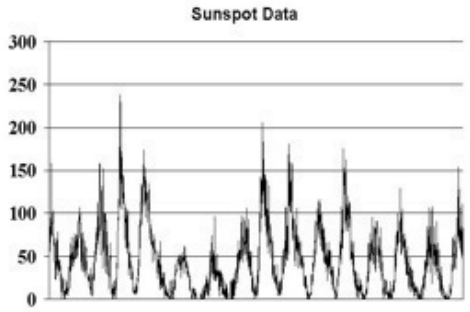




Motivation - Applications (cont'd)

- Weather, environment/anti-pollution
  - volcano monitoring
  - air/water pollutant monitoring
  - sunspots



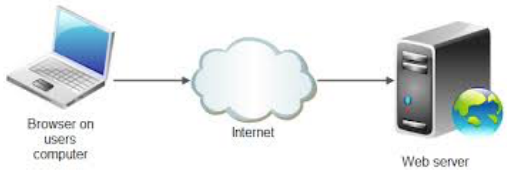


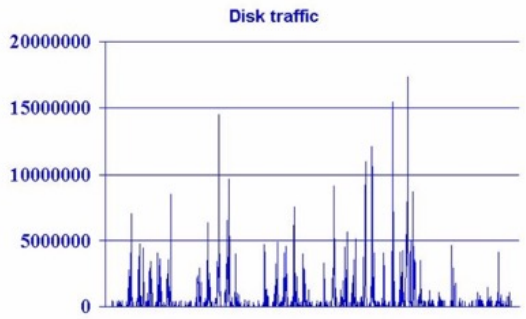
Carnegie Mellon

amazon

Motivation - Applications (cont'd)

- Computer systems
  - web servers (caching, prefetching)
  - network traffic monitoring
  - ...



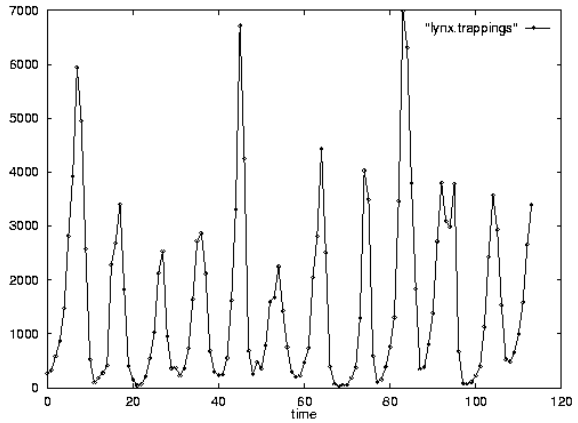


Problem #1:

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

Find: patterns, periodicities, and/or compress

count

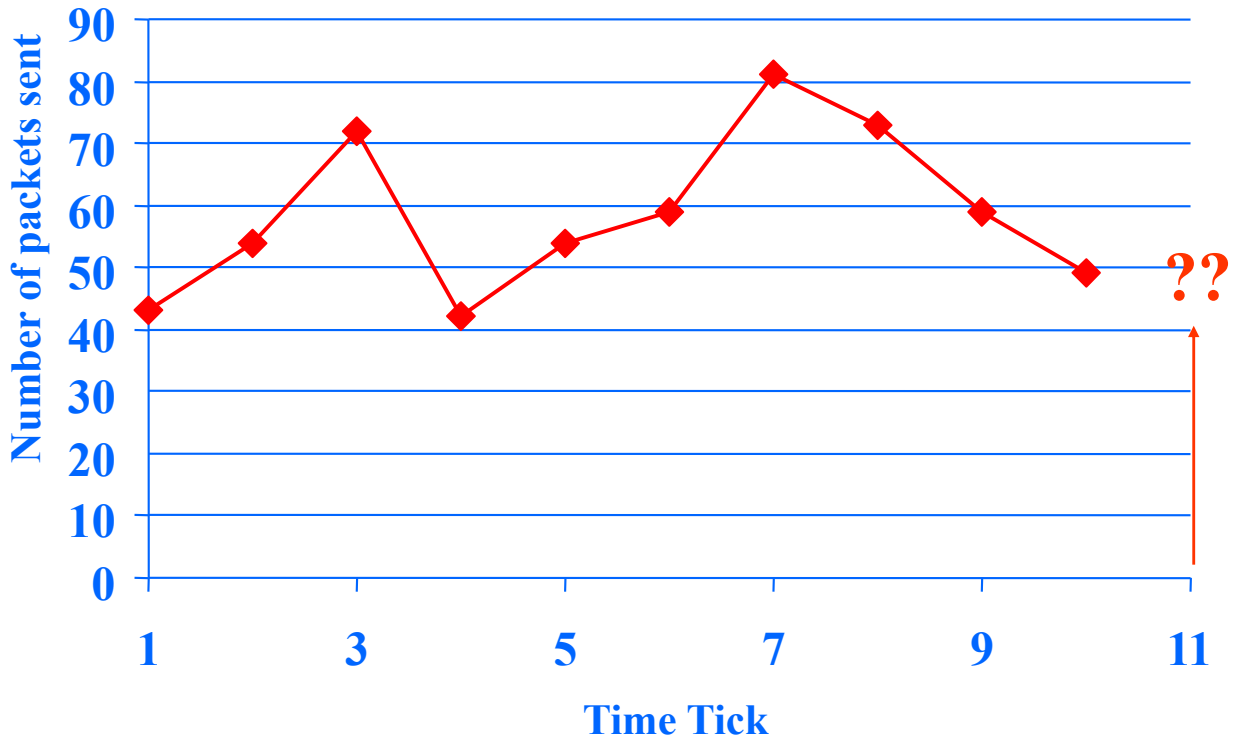


lynx caught per year
(packets per day;
temperature per day)

year

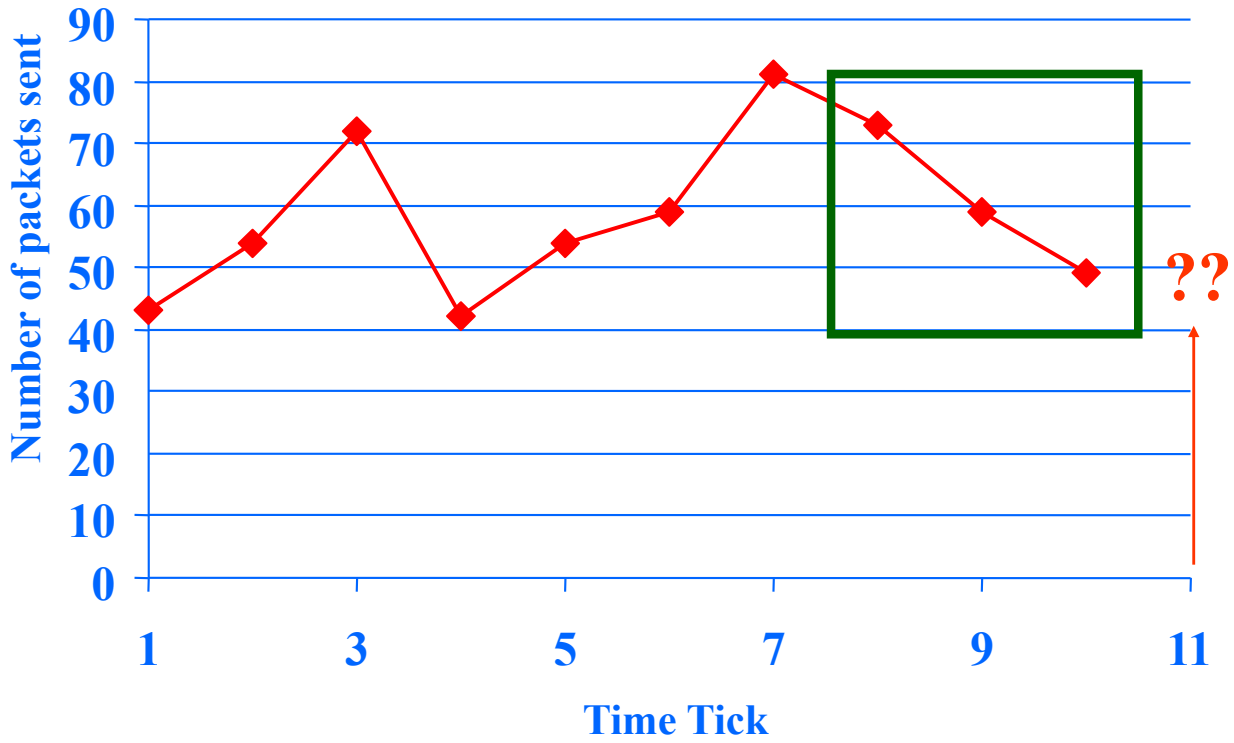
Problem#2: Forecast

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



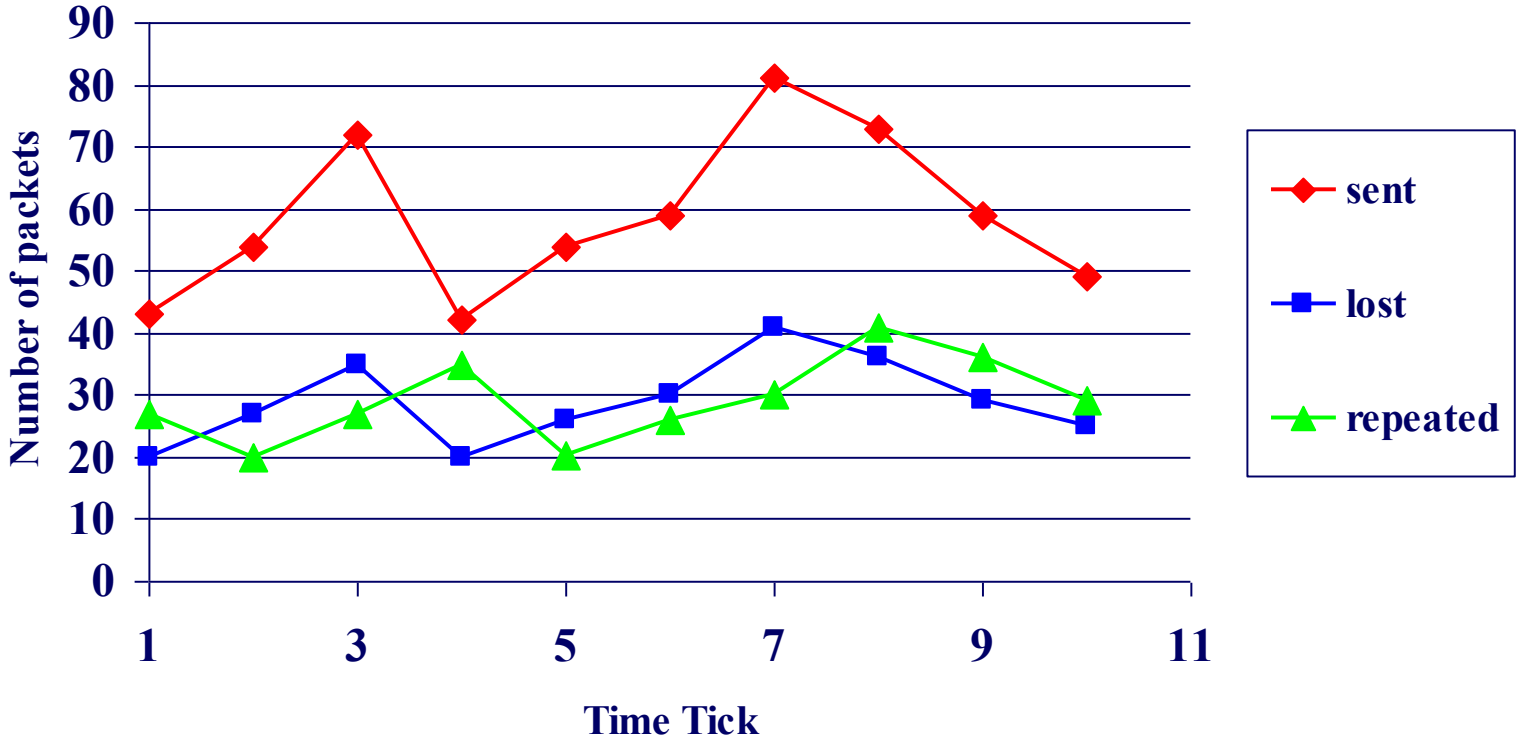
Problem#2': Similarity search

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

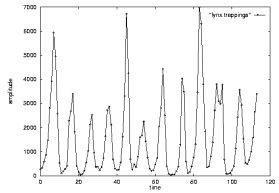


Problem #3:

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



Important observations



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

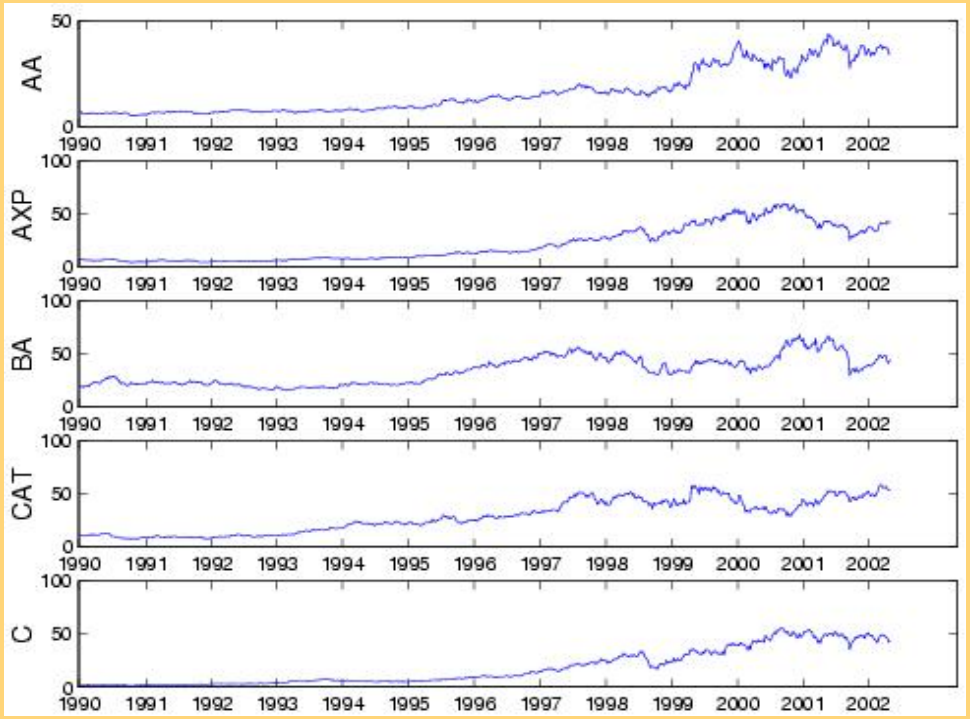
- To do **forecasting**, we need
  - to find **patterns**/rules
  - compress
  - to find **similar** settings in the past
- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)

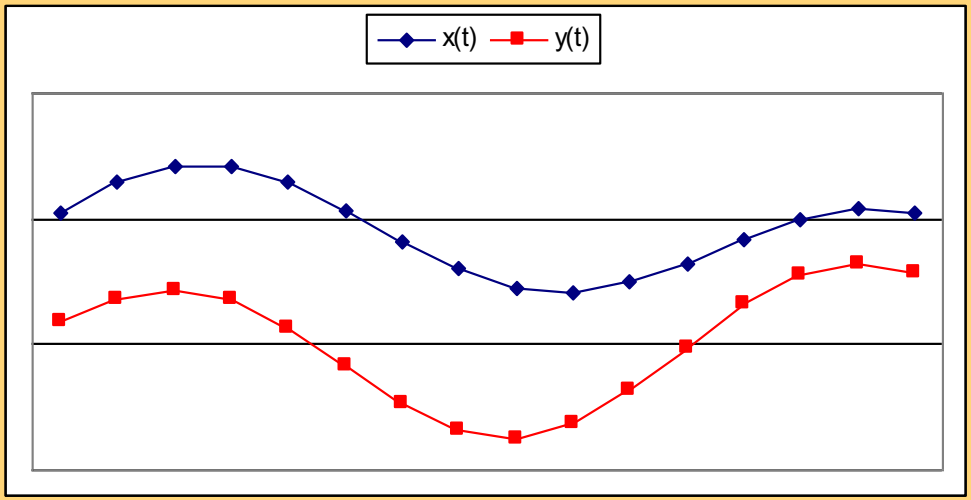
Part 1 - Outline

- Motivation
- P1.1. Similarity Search and Indexing
  - distance functions: Euclidean;Time-warping
  - indexing
  - feature extraction
- P1.2. DSP
- P1.3. Linear forecasting
- ...

P1.1 - Problem:

Q: How similar are two sequences?

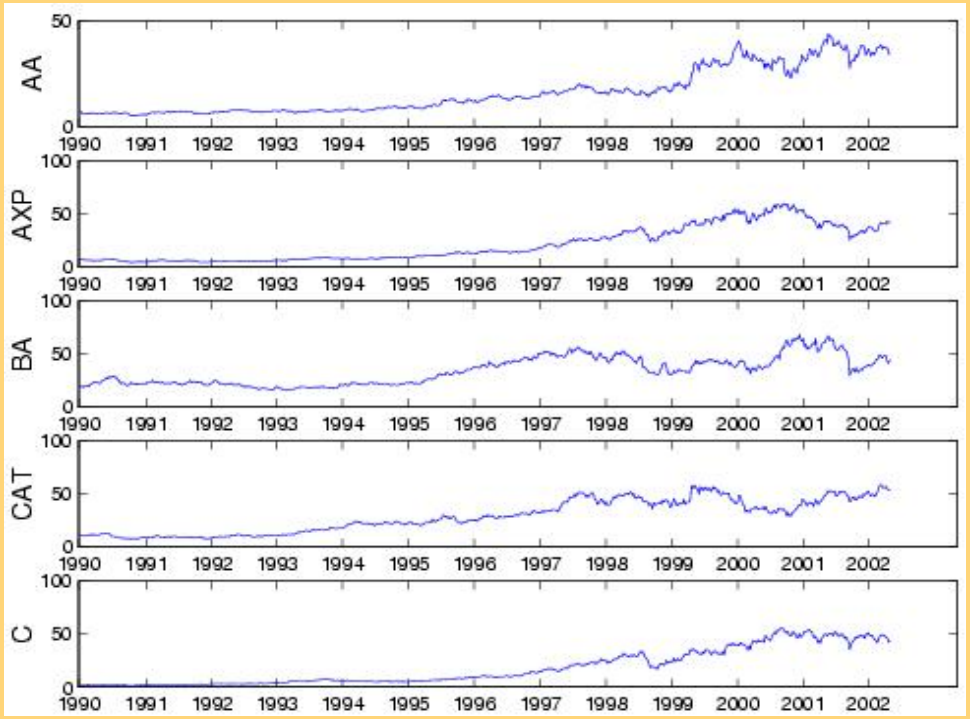




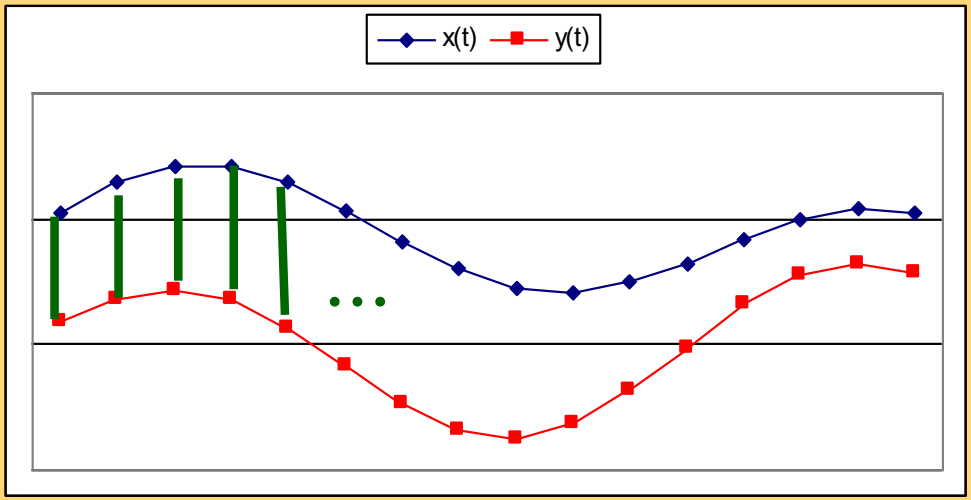
P1.1 - Answer:

Q: How similar are two sequences?

A: Euclidean distance (<-> cosine similarity)



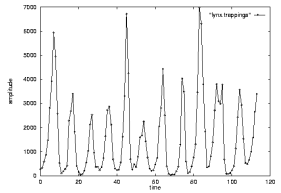
$$D(\vec{x}, \vec{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$$



Part 1 - Outline

- Motivation
- P1.1. Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
- P1.2. DSP
- P1.3. Linear forecasting
- ...

Important observations

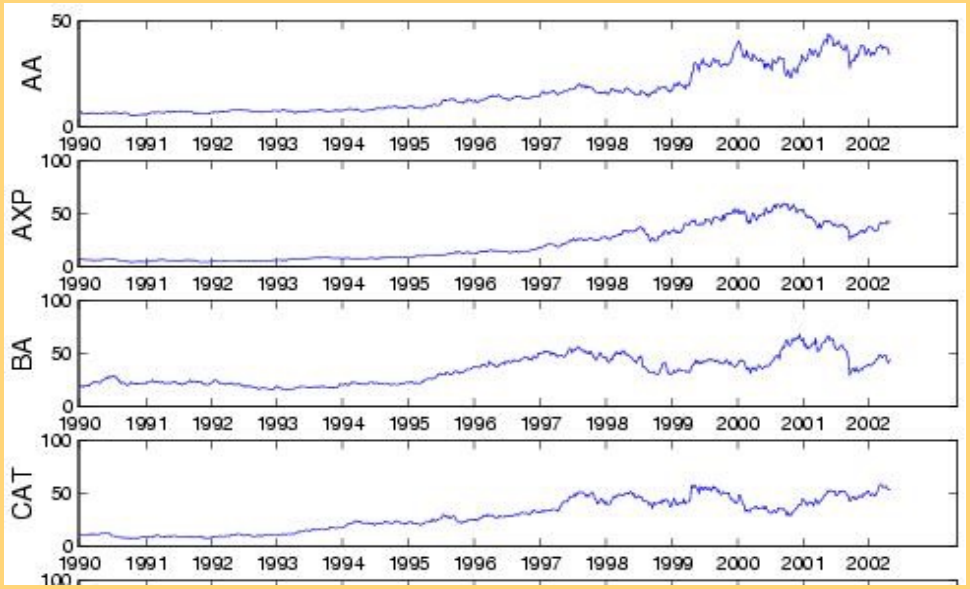


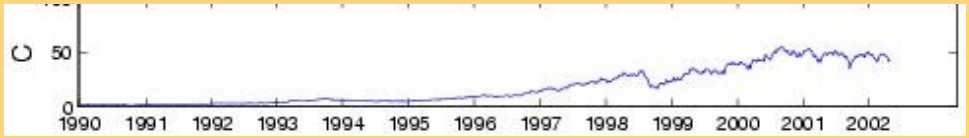
Patterns, rules, forecasting and similarity indexing are closely related:

- 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)

Problem:

Q: find quickly stocks like ‘C’ (or customers like ‘smith’)

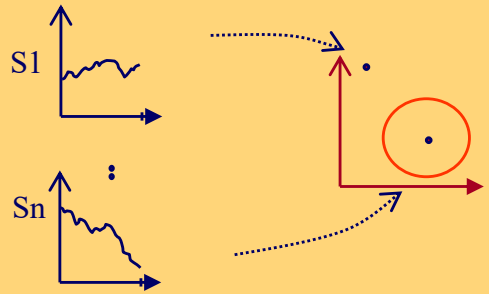


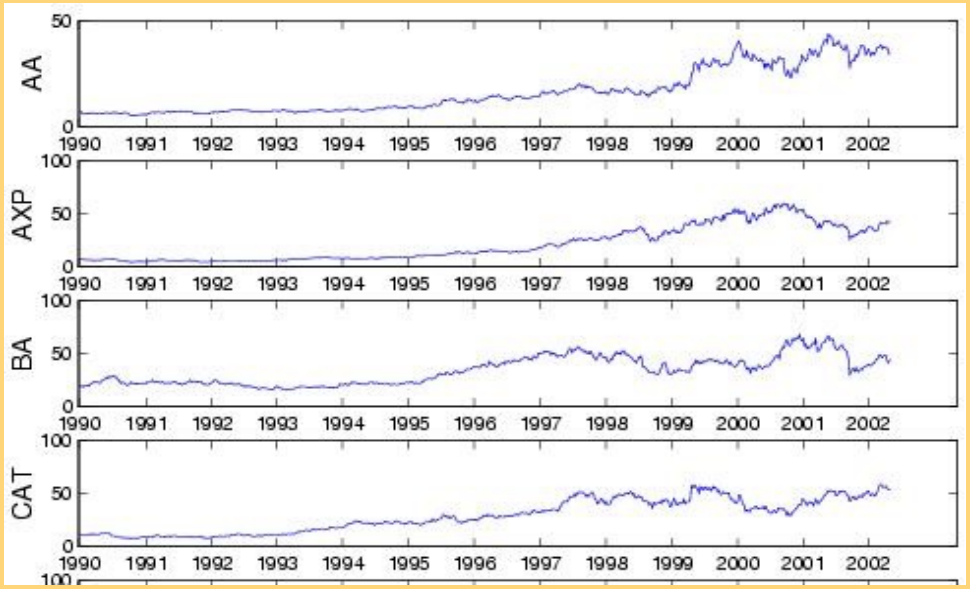


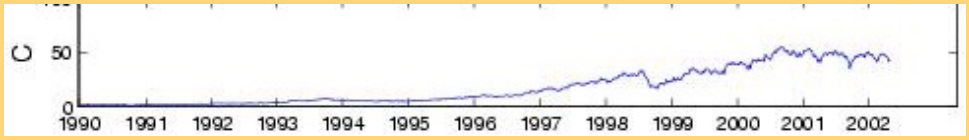
Answer:

Q: find quickly stocks like 'C' (or customers like 'smith')

A: summarize seq. to a few numbers/features (eg., avg, stdv, Fourier coeff.)







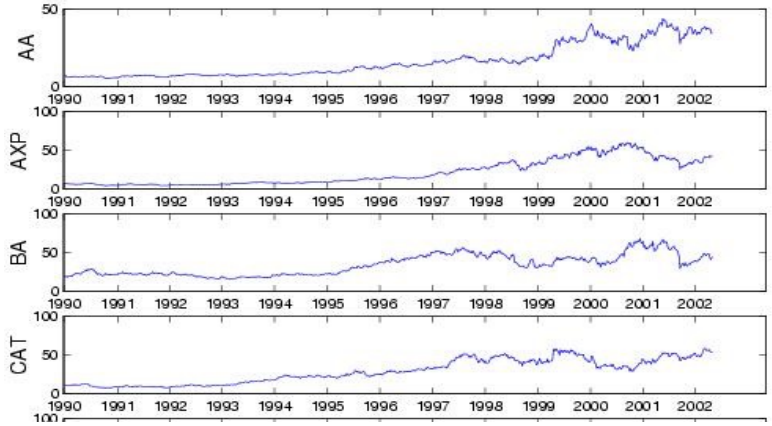
Idea: 'GEMINI'

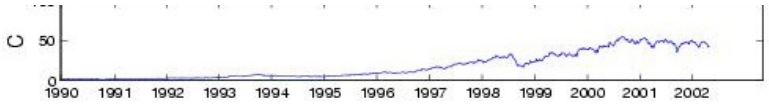
Eg., '*find stocks similar to MSFT*'

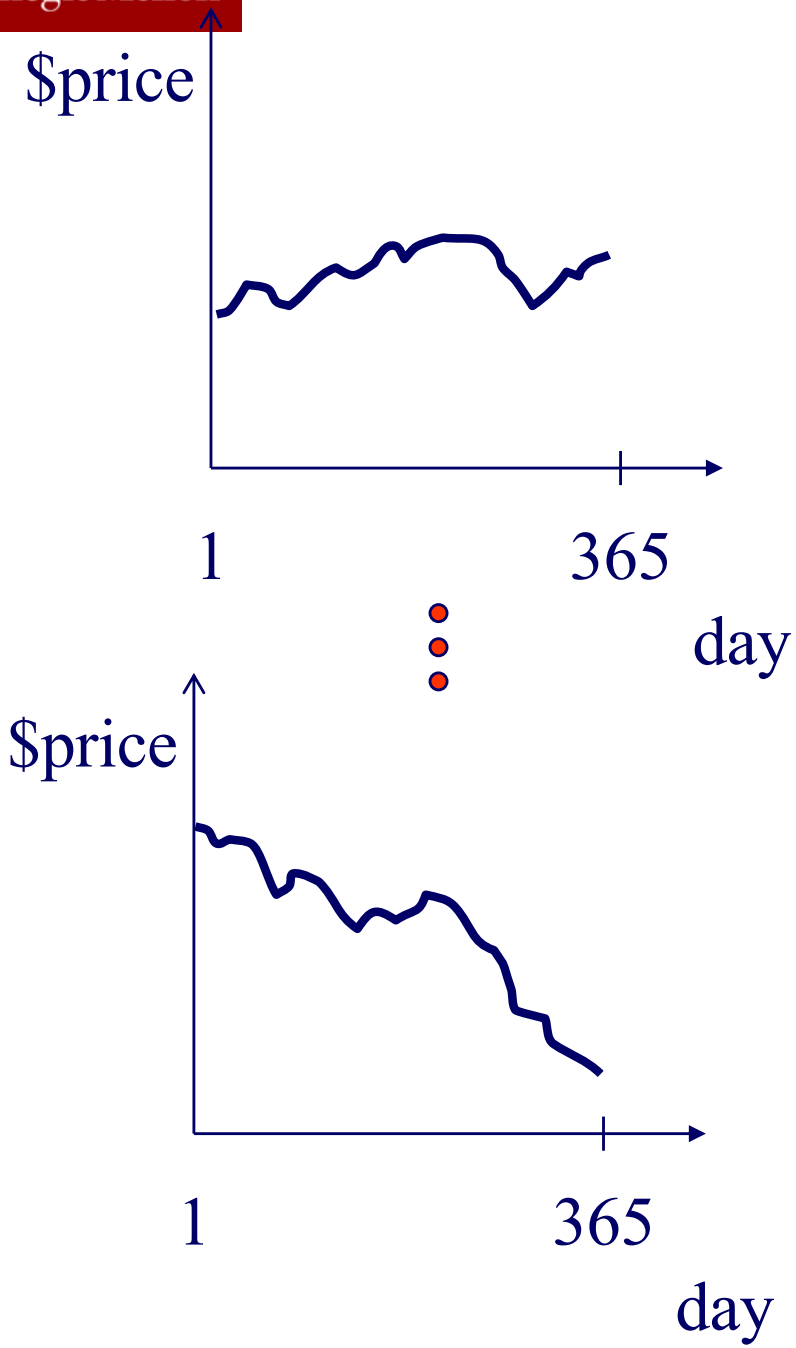
Seq. scanning: too slow

How to accelerate the search?

[Faloutsos96]

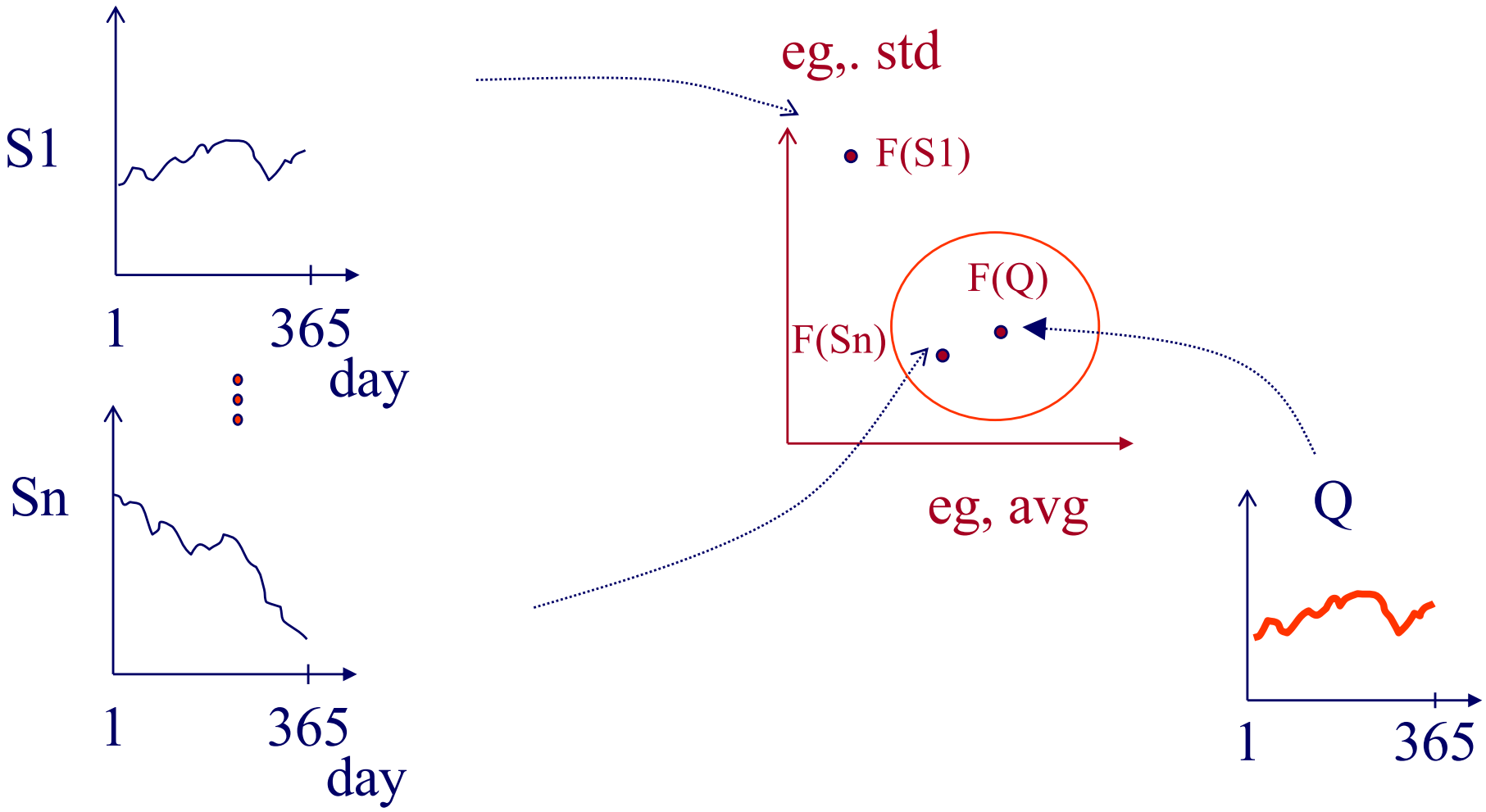




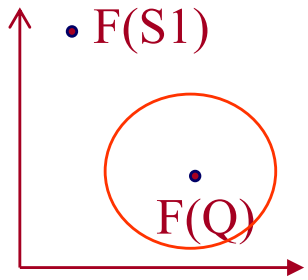


distance function: by expert

‘GEMINI’ - Pictorially



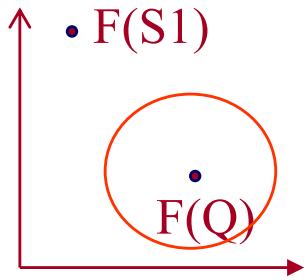
GEMINI



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

Conclusions



- Fast indexing: through GEMINI
  - feature extraction and
  - (off the shelf) Spatial Access Methods [Gaede+98] (eg, Rtree, kdtree, even in python)

But: features?

- How to extract 'good features'?
- A1:
  - **SVD** (Singular Value Decomposition); or
  - **ICA** (Independent Component Analysis)
- A2:
  - **DFT** (Discrete Fourier Transform)
  - **DWT** (Discrete Wavelet Transform)

Important observations:

Feature extraction =

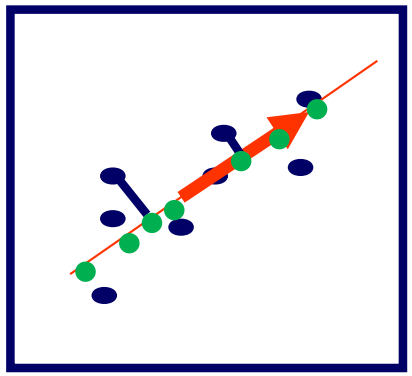
= *(lossy) compression*

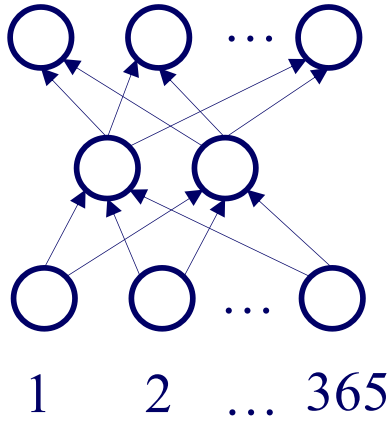
= *Dimensionality reduction*

= *Embedding*

= *Latent/hidden variable detection*

= *AutoEncoding*

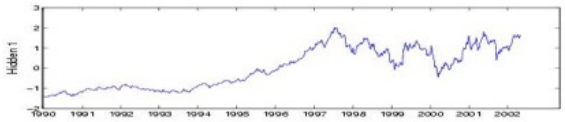


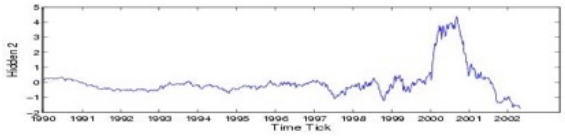


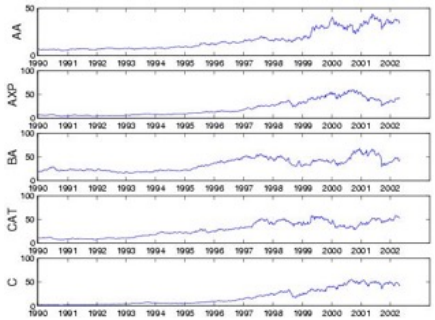
Part 1 - Outline

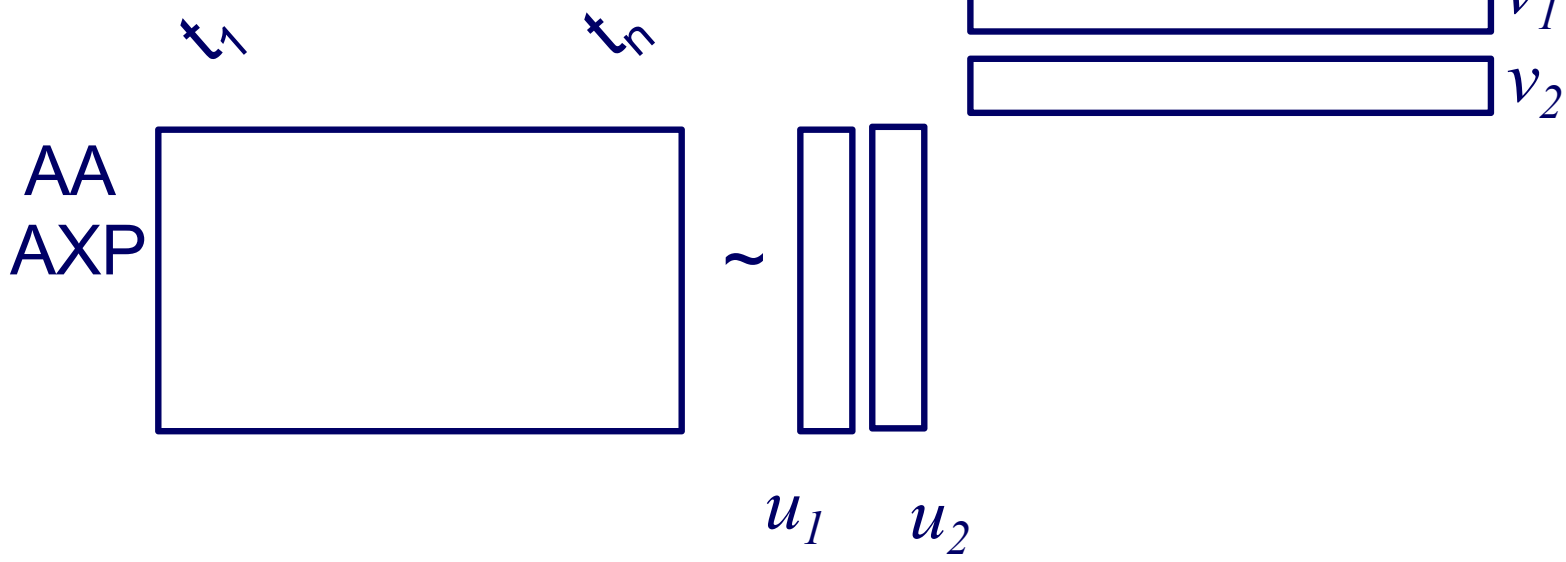
- Motivation
- P1.1. Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
    - SVD etc (data dependent)
    - (DFT, DWT, DCT (data independent))
- P1.2. DSP (DFT, DWT)

SVD










```
# svd code
import numpy as np
u, s, vh = np.linalg.svd(b)
```

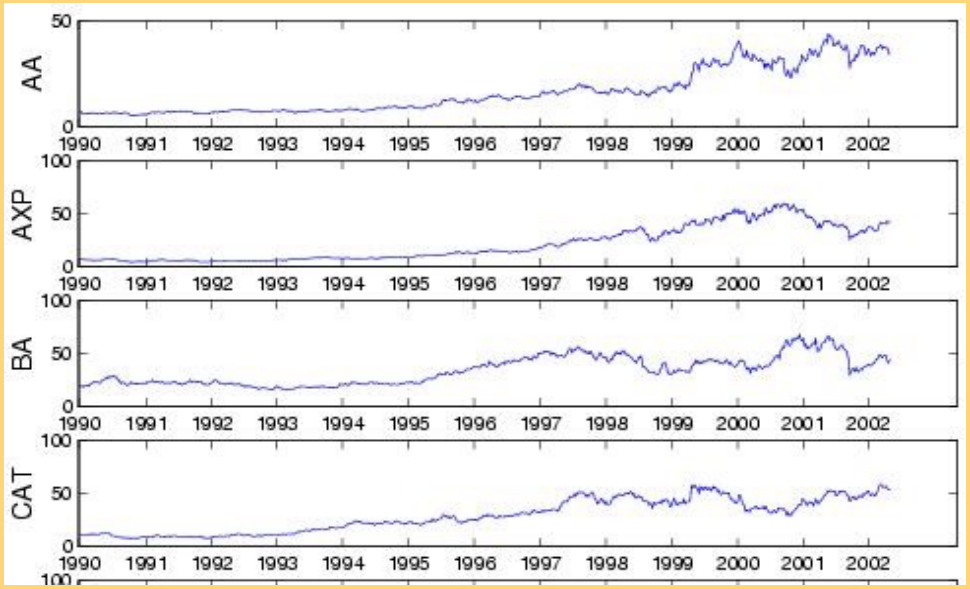
U **Σ** $\mathbf{V}^T$

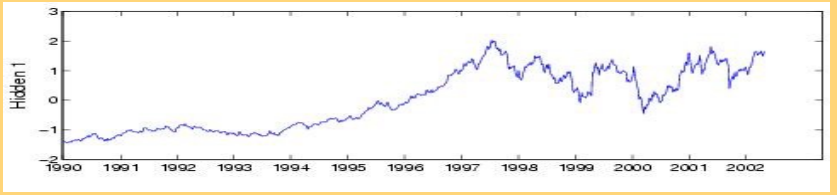
Carnegie Mellon

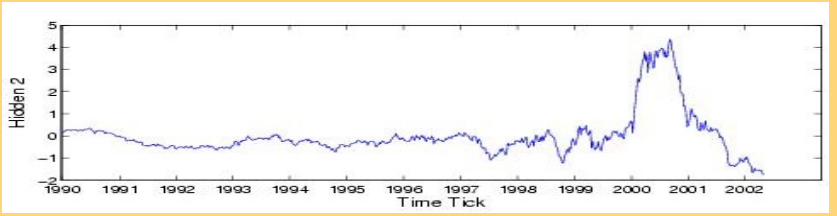
amazon

Answer

Q: how to extract features (commonalities)? (given the data)
A: SVD, ICA







Carnegie Mellon

amazon

Citations

SVD • Jolliffe, I. T. (1986). *Principal Component Analysis*, Springer Verlag.

ICA • *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

Books + lecture notes

- 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)

- CMU class slides: SVD, and ICA

References

- Foltz, P. W. and S. T. Dumais (Dec. 1992). "*Personalized Information Delivery: An Analysis of Information Filtering Methods*." Comm. of ACM (CACM) 35(12): 51-60.
- 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.
- 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

Carnegie Mellon

amazon

Part 1.2: DSP (Digital Signal Processing)

Part 1 - Outline

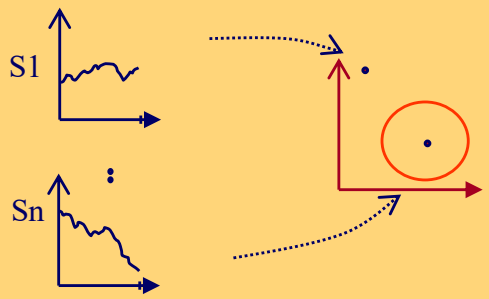
- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP (DFT, DWT)
  - Fourier (DFT)
  - Wavelets (DWT)
- P1.3. Linear Forecasting
- P1.4. Non-linear forecasting
- P1.5. Tensors
- Conclusions

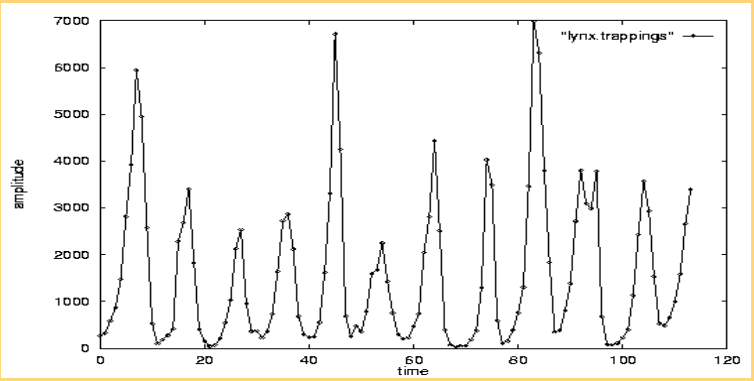
Part 1.2 - Outline

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

P1.2 - Problem

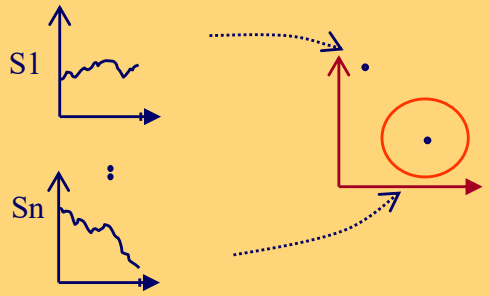
Q: How to summarize / extract few features

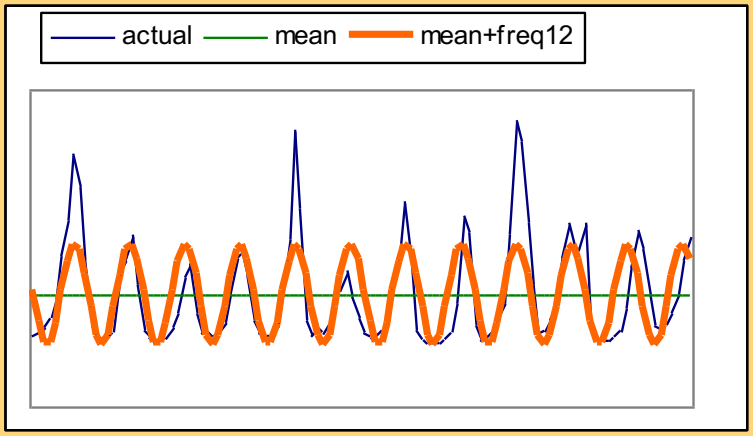




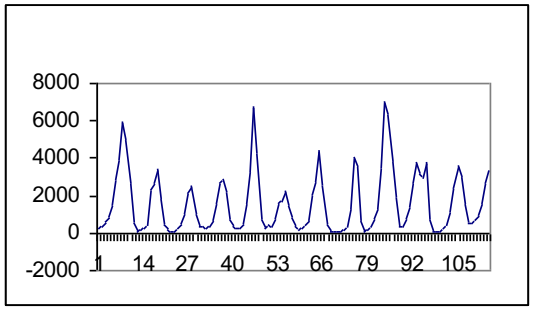
P1.2 - Answer:

Q: How to summarize / extract few features
A1: Data dep.: SVD, ICA
A2: Data indep.: Fourier; Wavelets



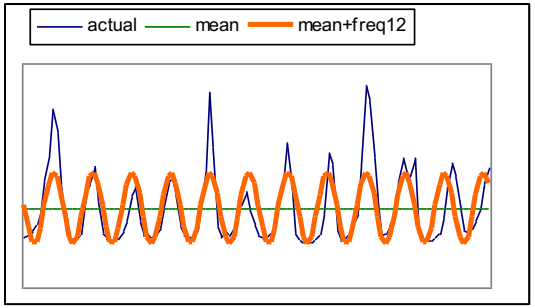


What does DFT do?



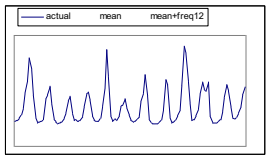
What does DFT do?

A: highlights the periodicities

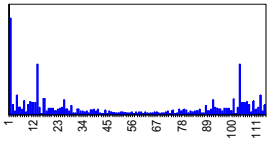


DFT: definition

- For a sequence $x_0, x_1, \ldots x_{n-1}$



- the (**n-point**) Discrete Fourier Transform is
- $X_0, X_1, \ldots X_{n-1}$:



$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t * \exp(-j2\pi\, tf\,/\,n) \qquad f = 0,\ldots,n-1$$

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

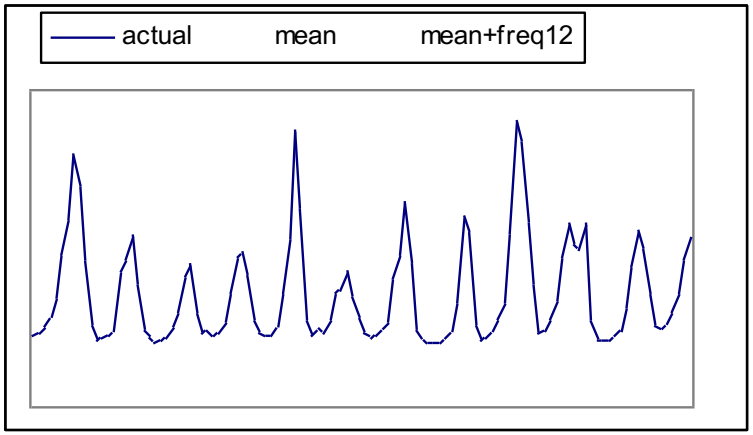
inverse DFT

$$x_t = 1/\sqrt{n} \sum_{t=0}^{n-1} X_f * \exp(+j2\pi\, tf\,/\,n)$$

DFT: Amplitude spectrum

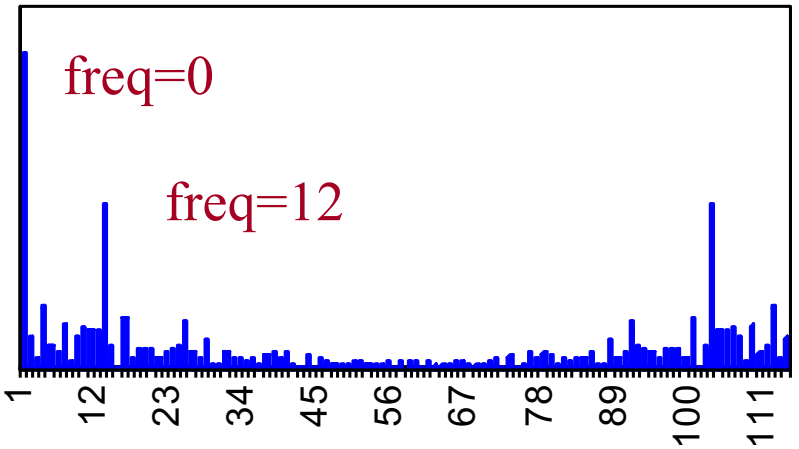
Amplitude: $A_f^2 = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$

count



year

Ampl.

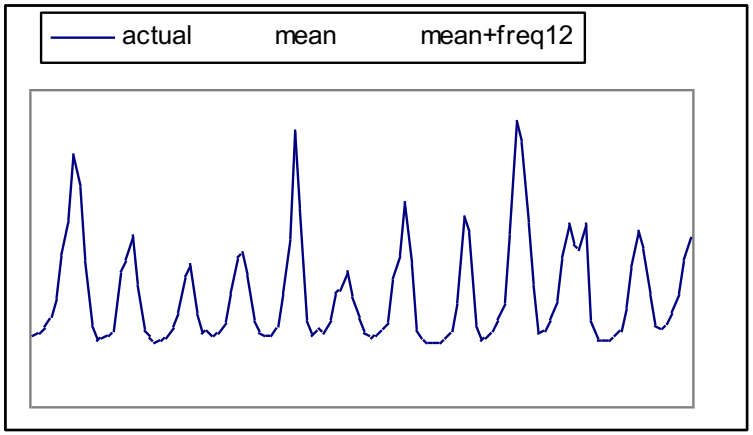


Freq.

DFT: Amplitude spectrum

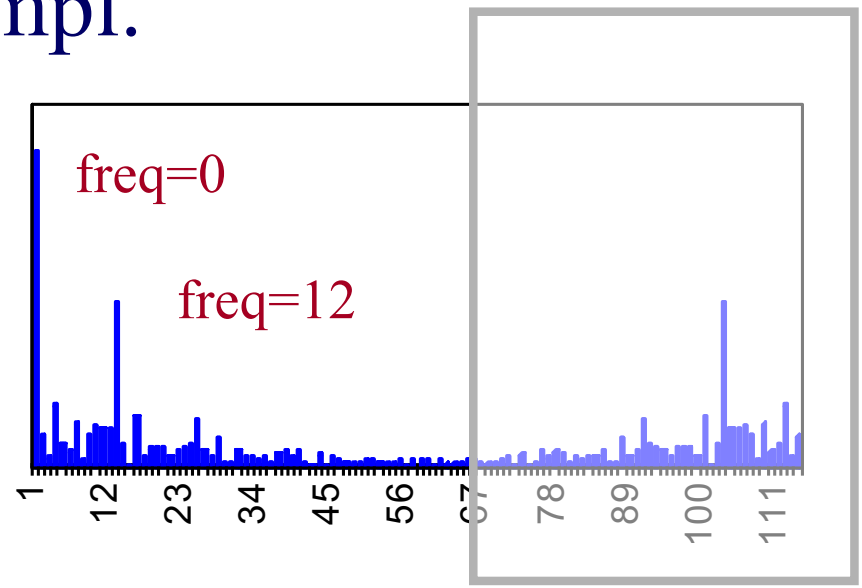
Amplitude: $A_f^2 = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$

count



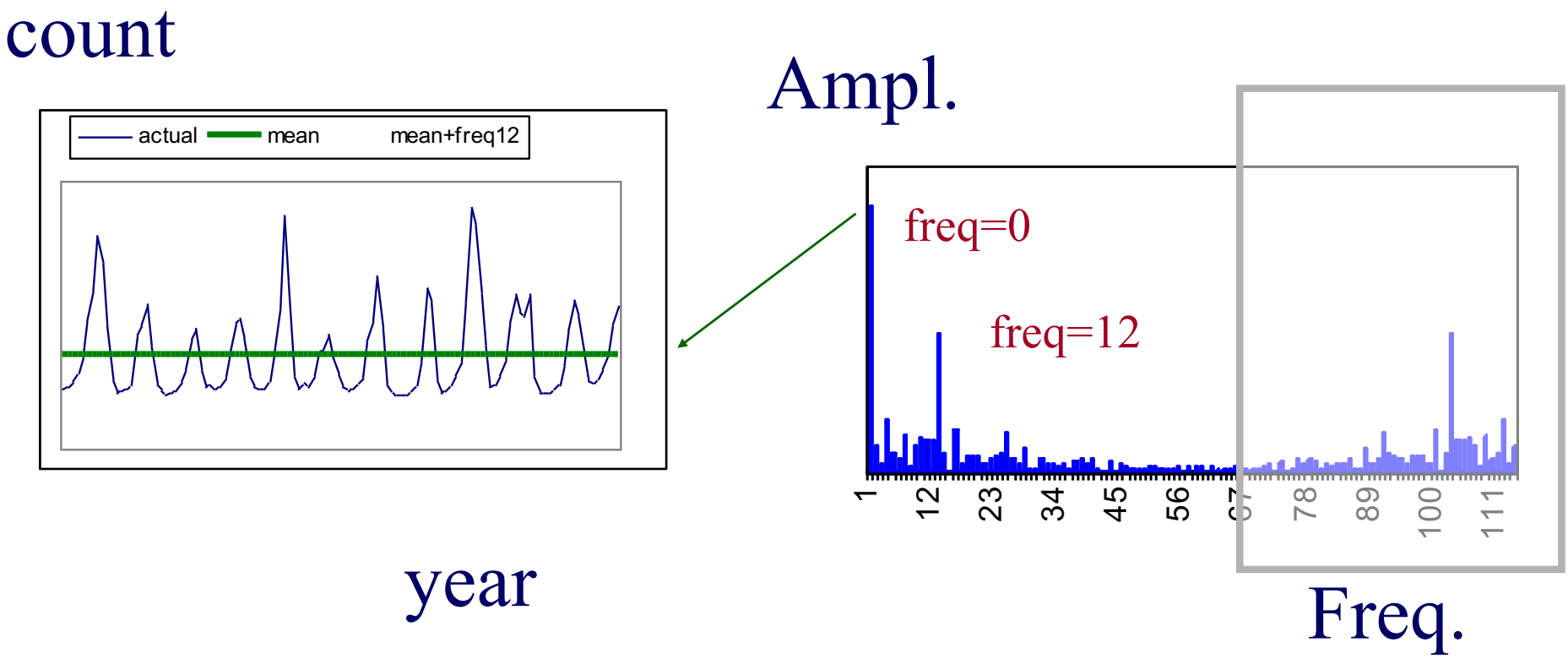
year

Ampl.

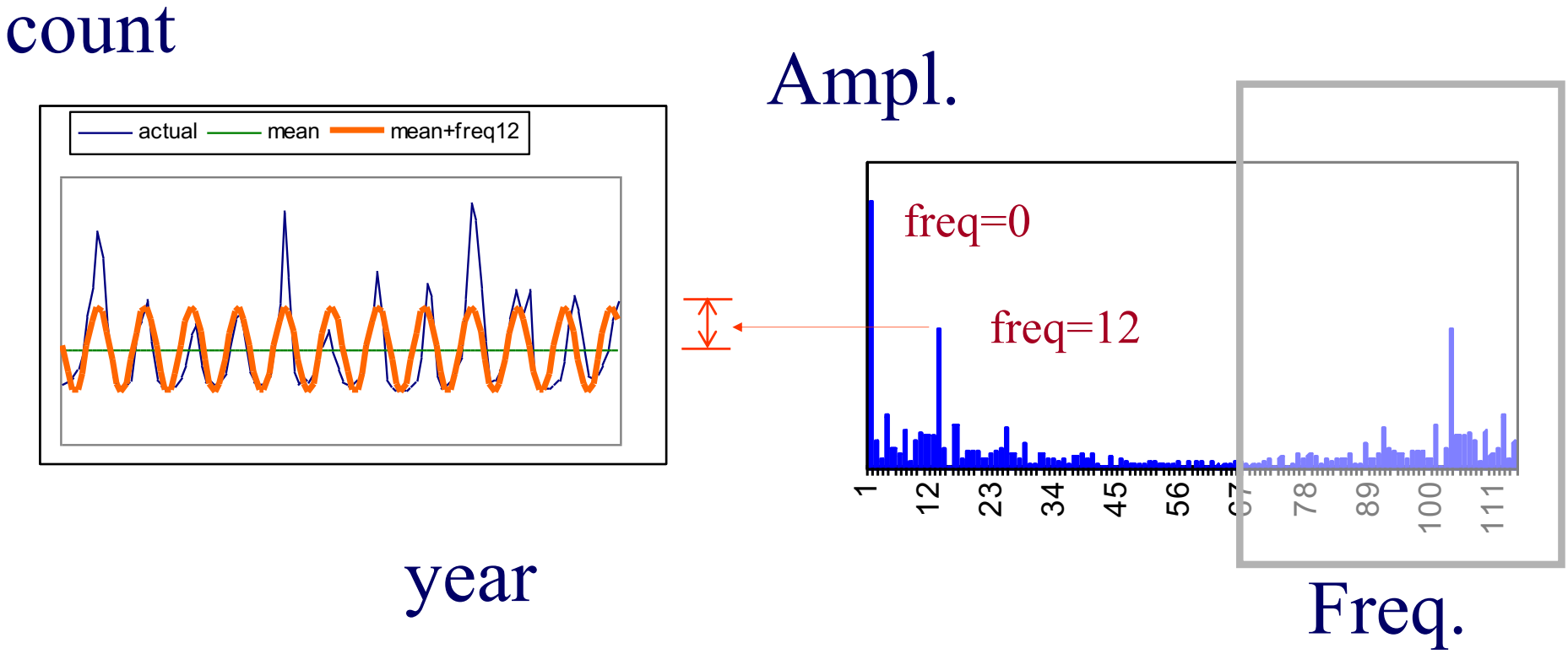


Freq.

DFT: Amplitude spectrum

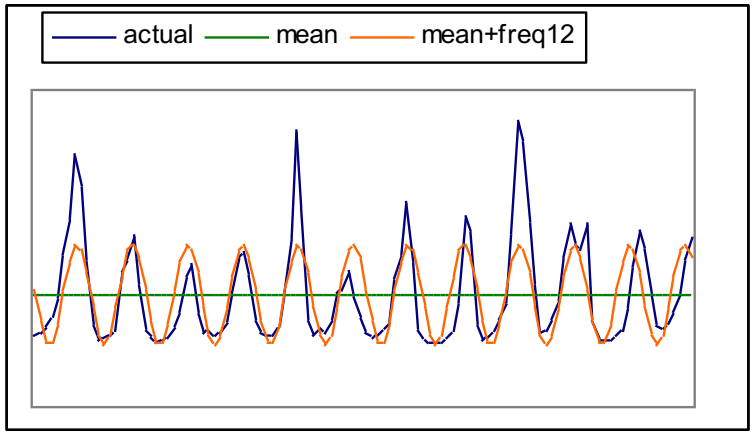


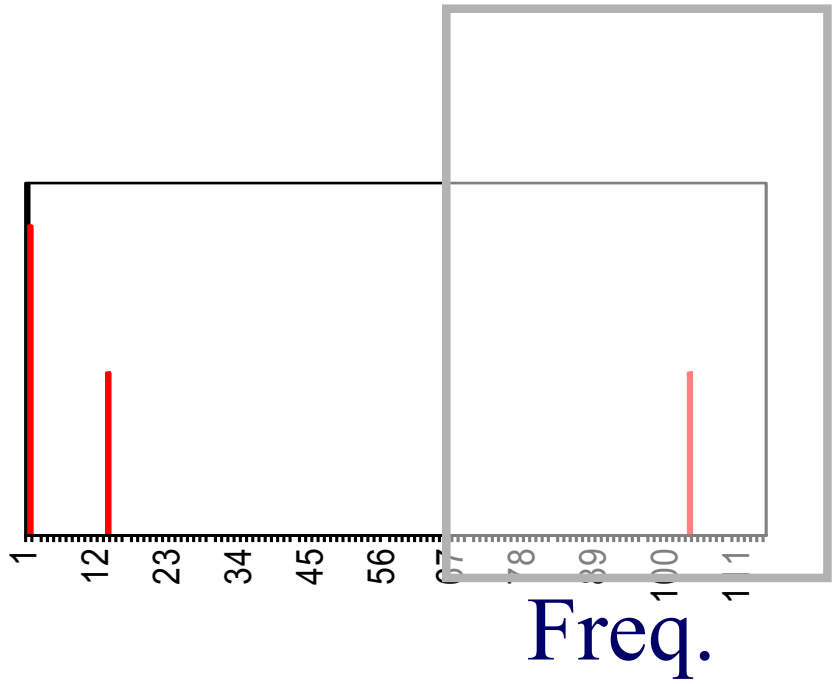
DFT: Amplitude spectrum



DFT: Amplitude spectrum

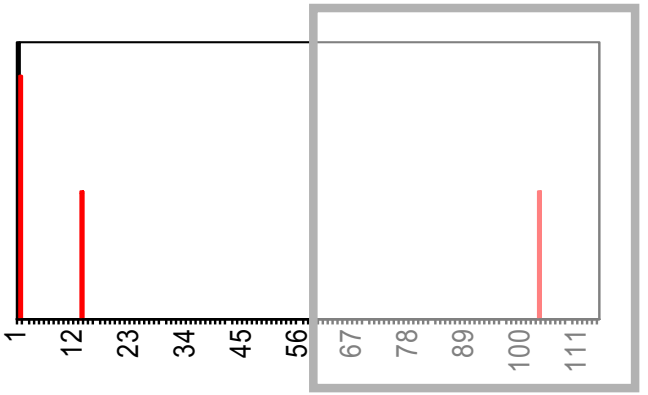
- excellent approximation, with only 2 frequencies! (2 int, 3 float)
- so what?





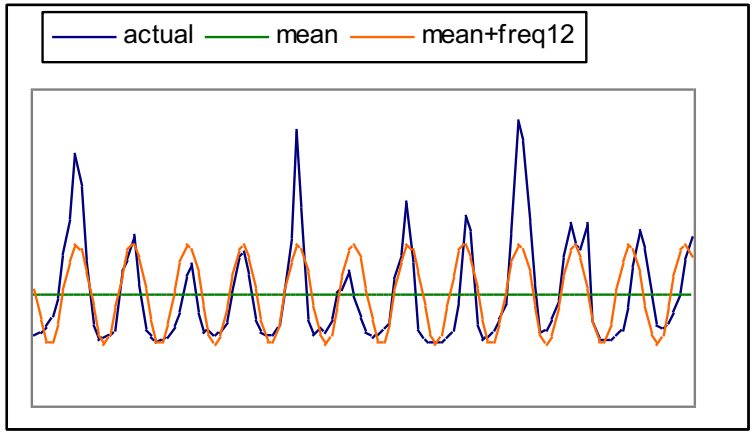
DFT: Amplitude spectrum

- 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

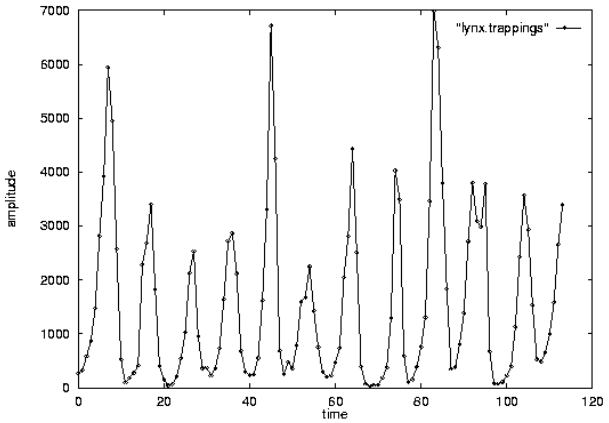
- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP
  - DFT
  - DWT
    - Definition of DWT and properties
    - how to read the DWT scalogram
- P1.3. Linear Forecasting
- ….

Problem #1:

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

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

count



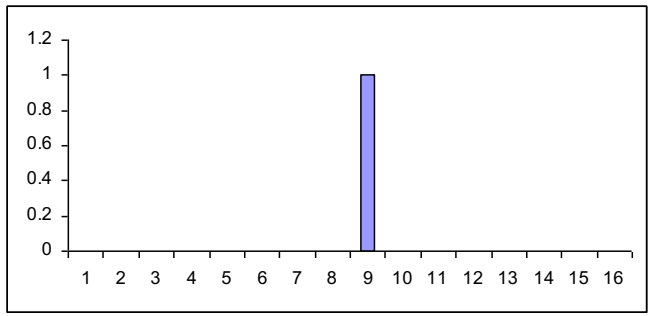
lynx caught per year
(packets per day;
virus infections per month)

year

Wavelets - DWT

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

value

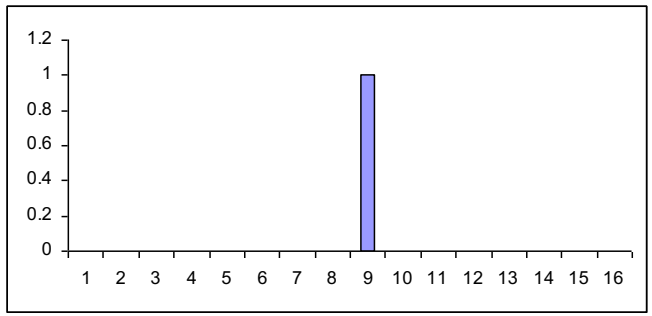


time

Wavelets - DWT

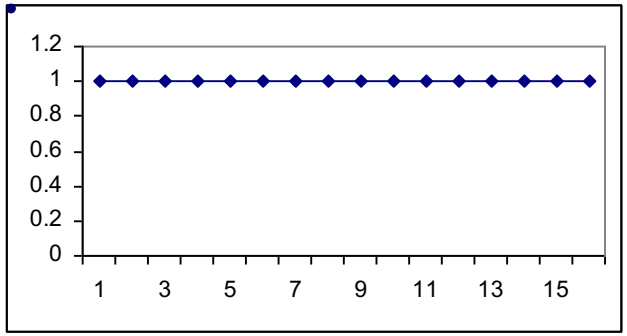
- DFT is great - but, how about compressing a spike?
- A: Terrible - all DFT coefficients needed!

value



time

Ampl.

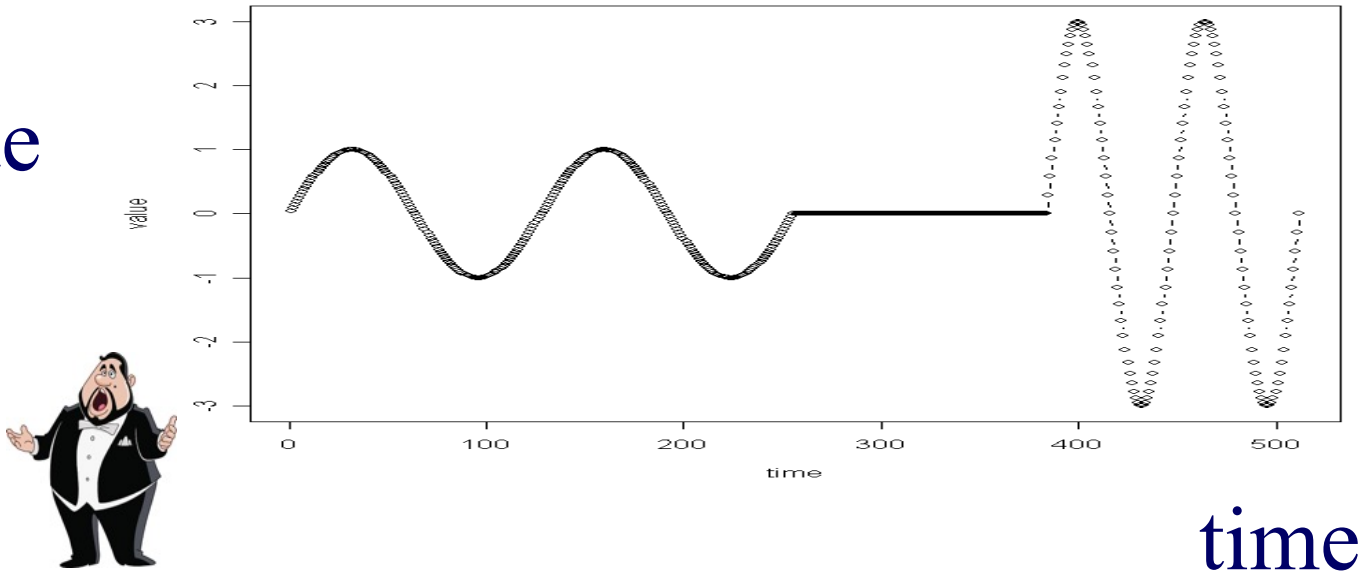


Freq.

Wavelets - DWT

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

value

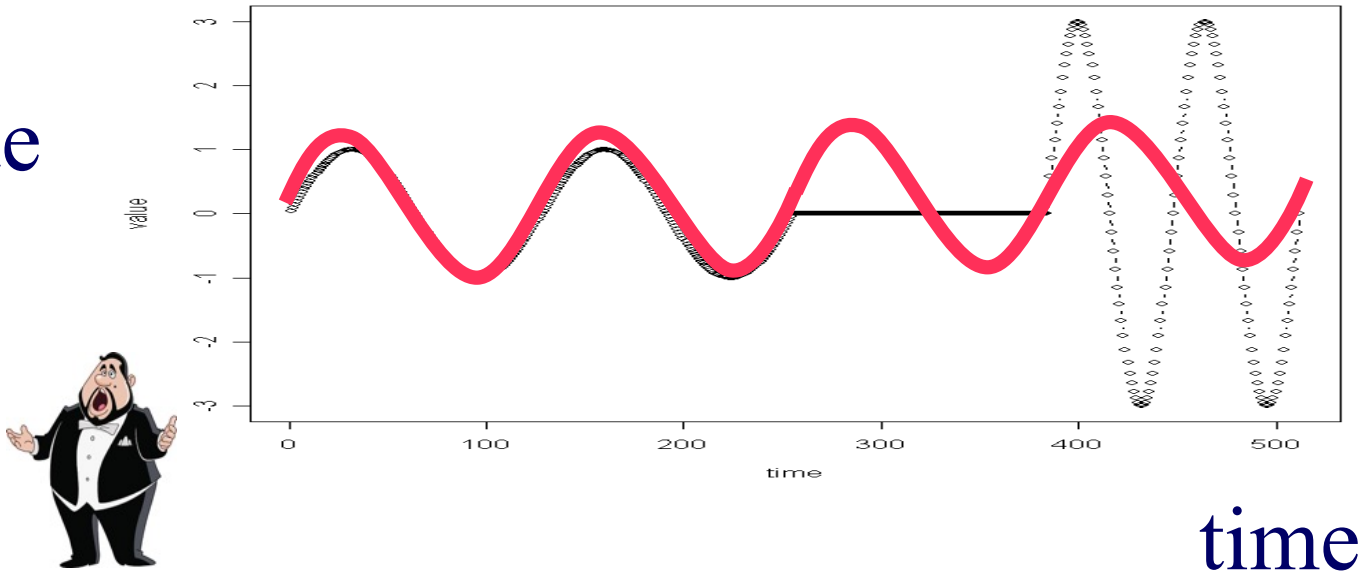


time

Wavelets - DWT

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

value

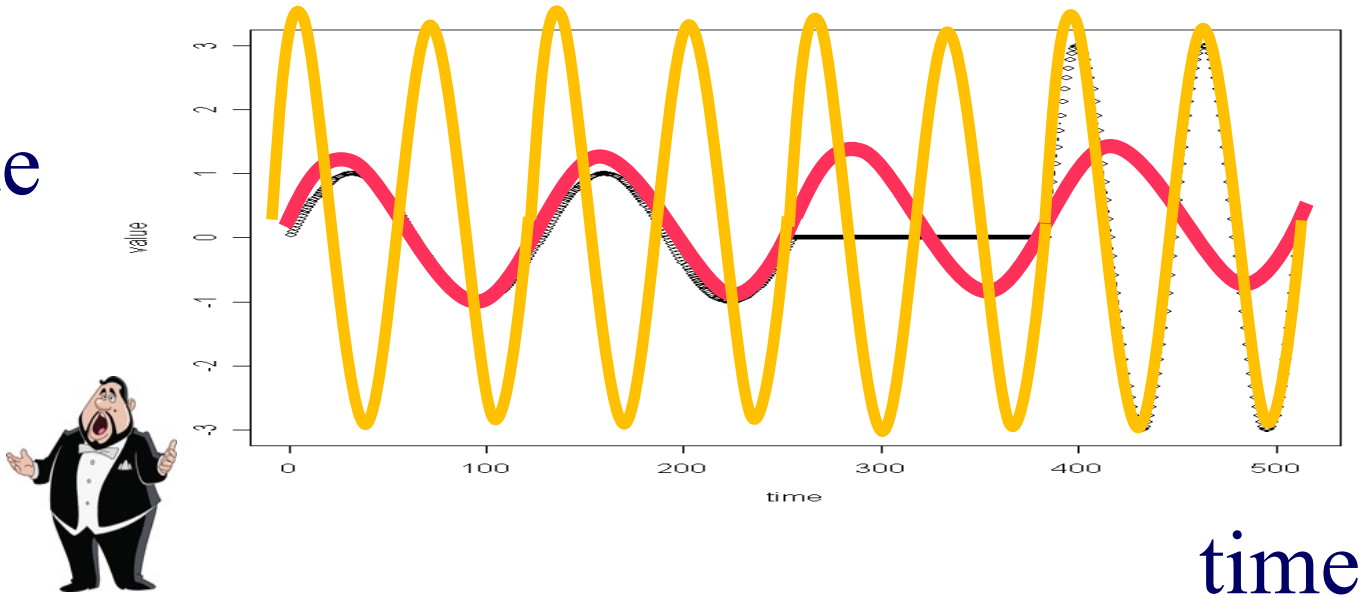


time

Wavelets - DWT

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

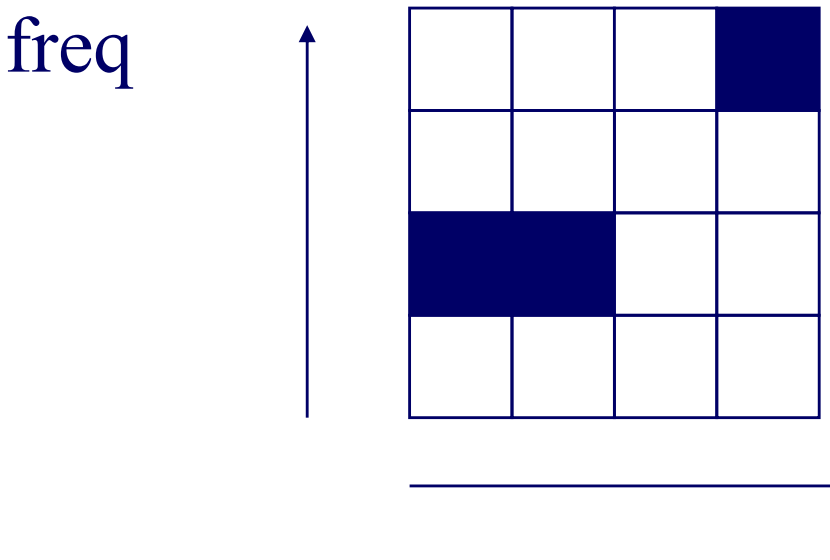
value

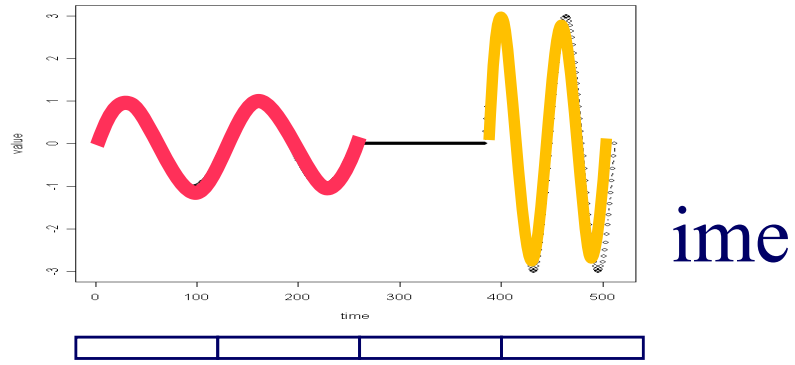


time

Wavelets - DWT

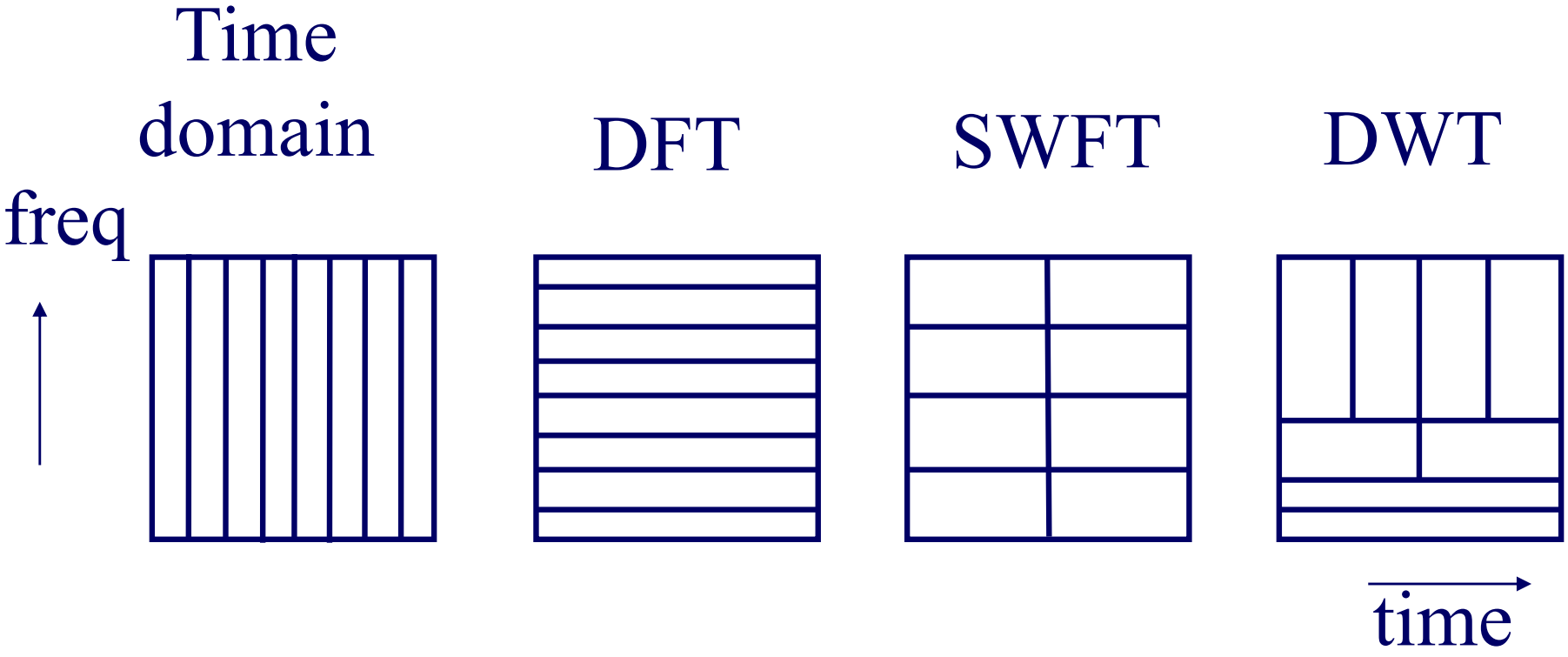
- Solution#1: Short window Fourier transform (SWFT)
- But: how short should be the window?





Wavelets - DWT

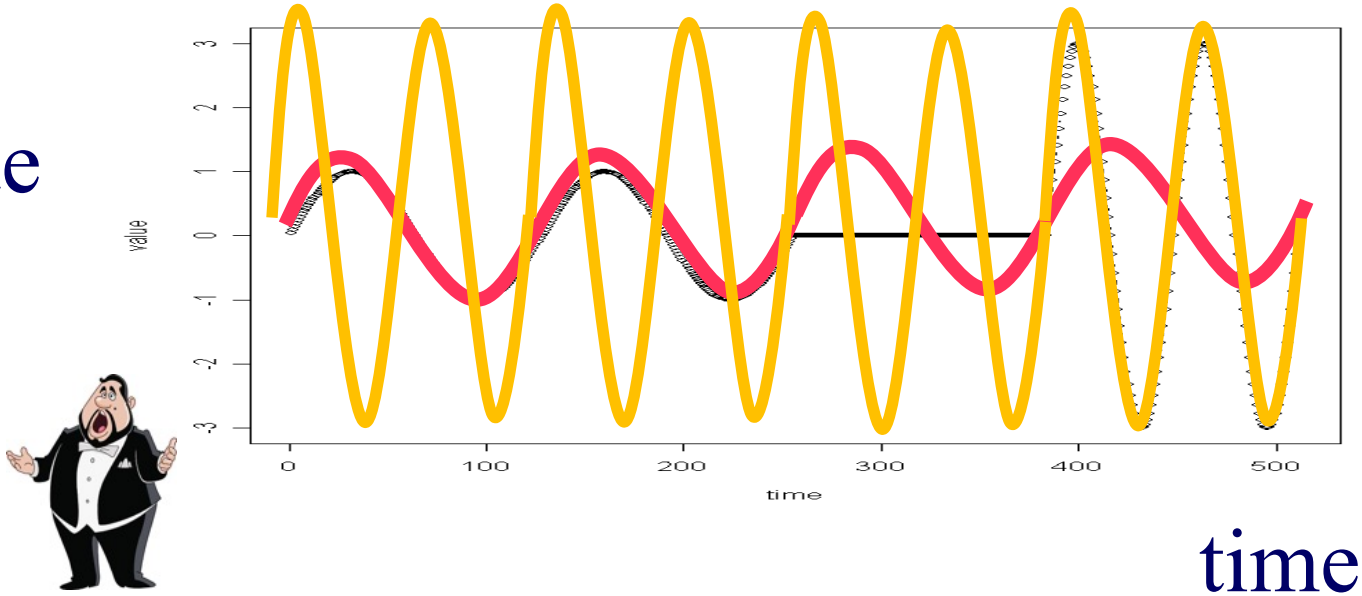
- Answer: **multiple** window sizes! -> DWT



(Basis functions of DFT)

- Sine waves of several frequencies (1, 2, 3, ...)

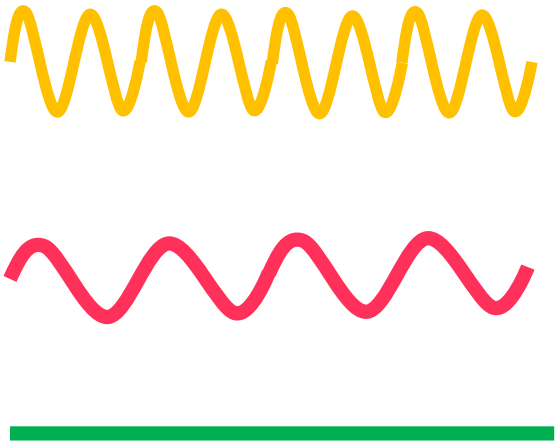
value

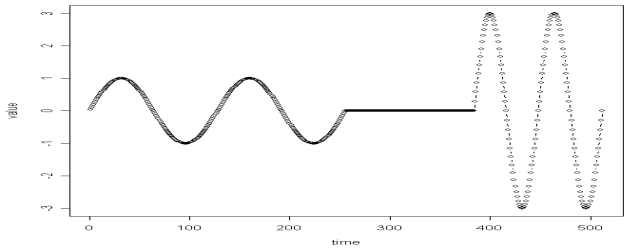


time

(Basis functions of DFT)

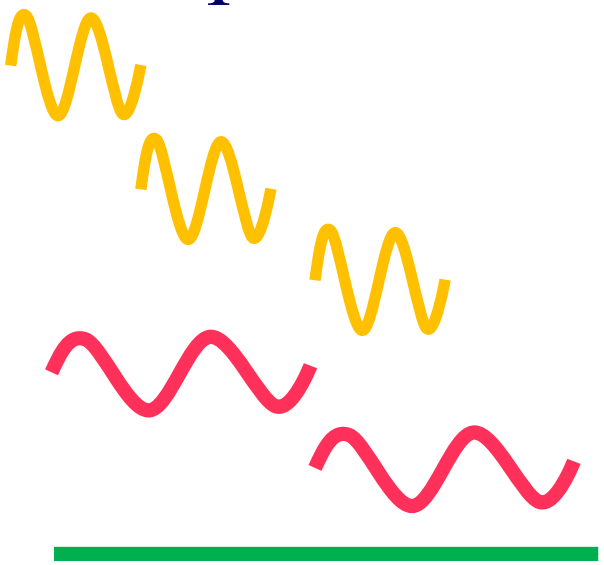
- Sine waves of several frequencies (1, 2, 3, …)

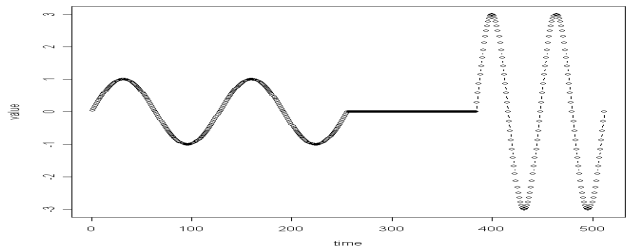




(Basis functions of DWT)

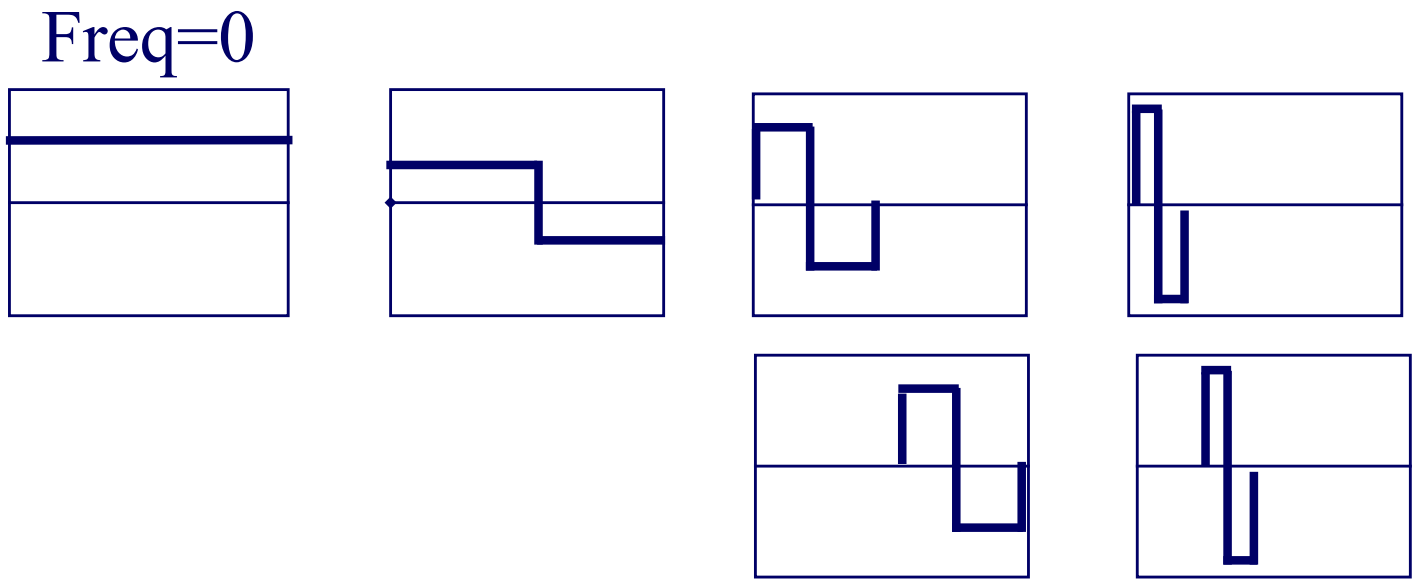
- (~ Sine) waves of several frequencies (1,2,4,8,…)
- and LIMITED timespan: 'little waves'





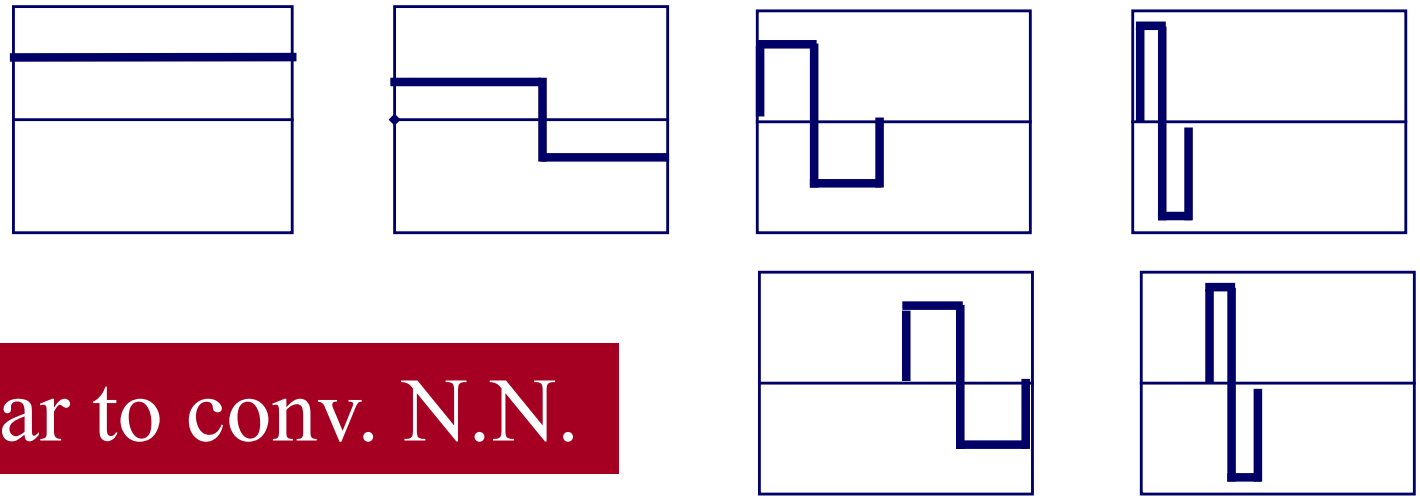
Specifically, Haar Wavelets:

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



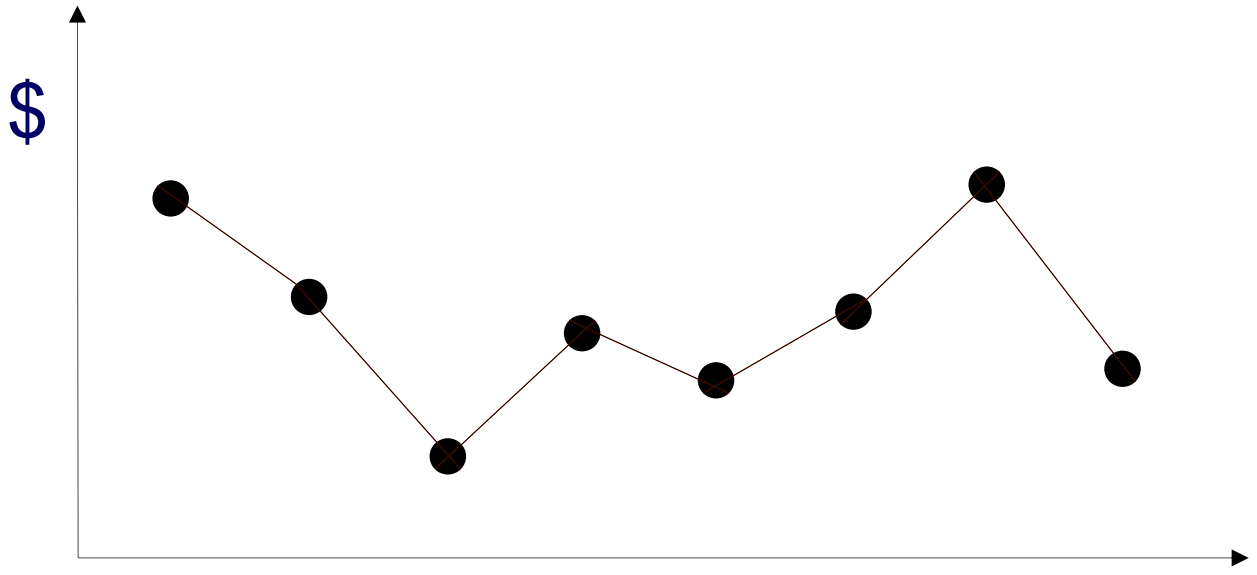
Haar Wavelets

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



Similar to conv. N.N.

Wavelets - construction



x0 x1 x2 x3 x4 x5 x6 x7

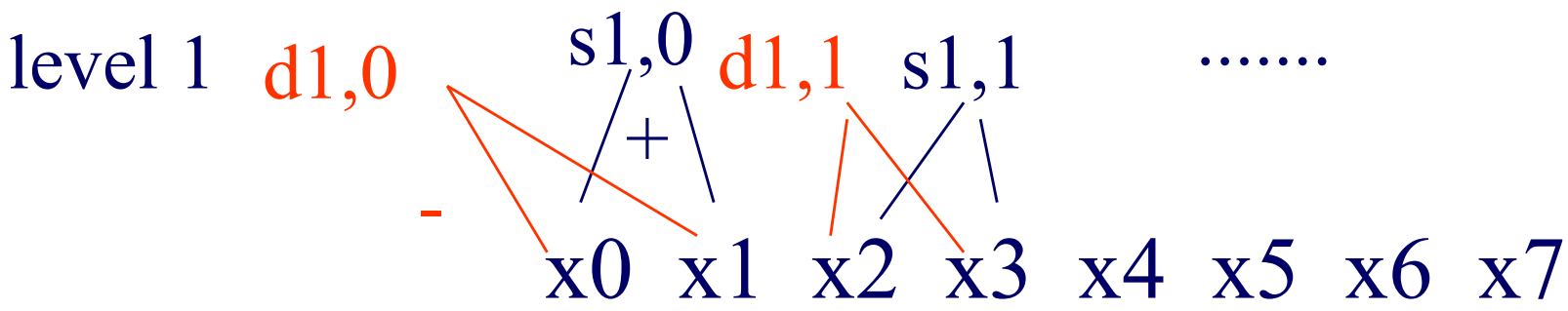
Carnegie Mellon

amazon

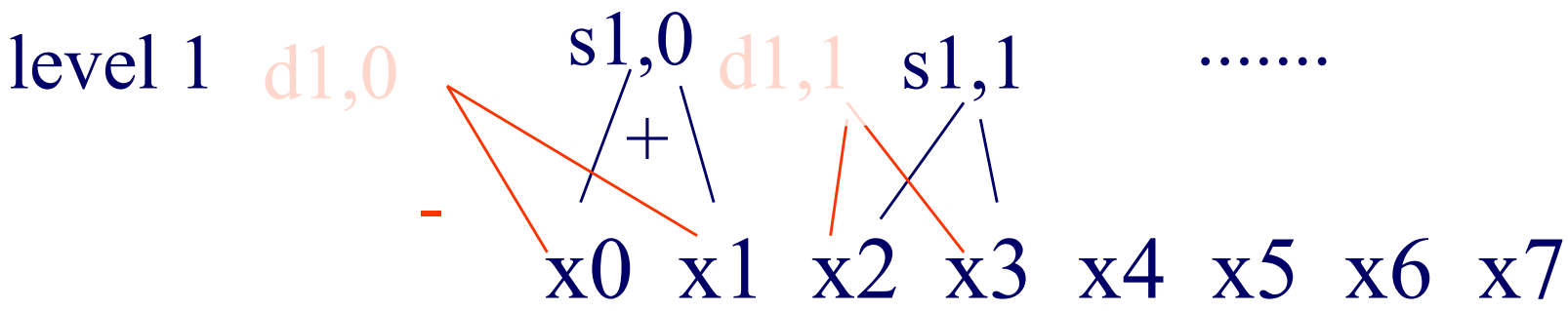
Wavelets - construction

x0 x1 x2 x3 x4 x5 x6 x7

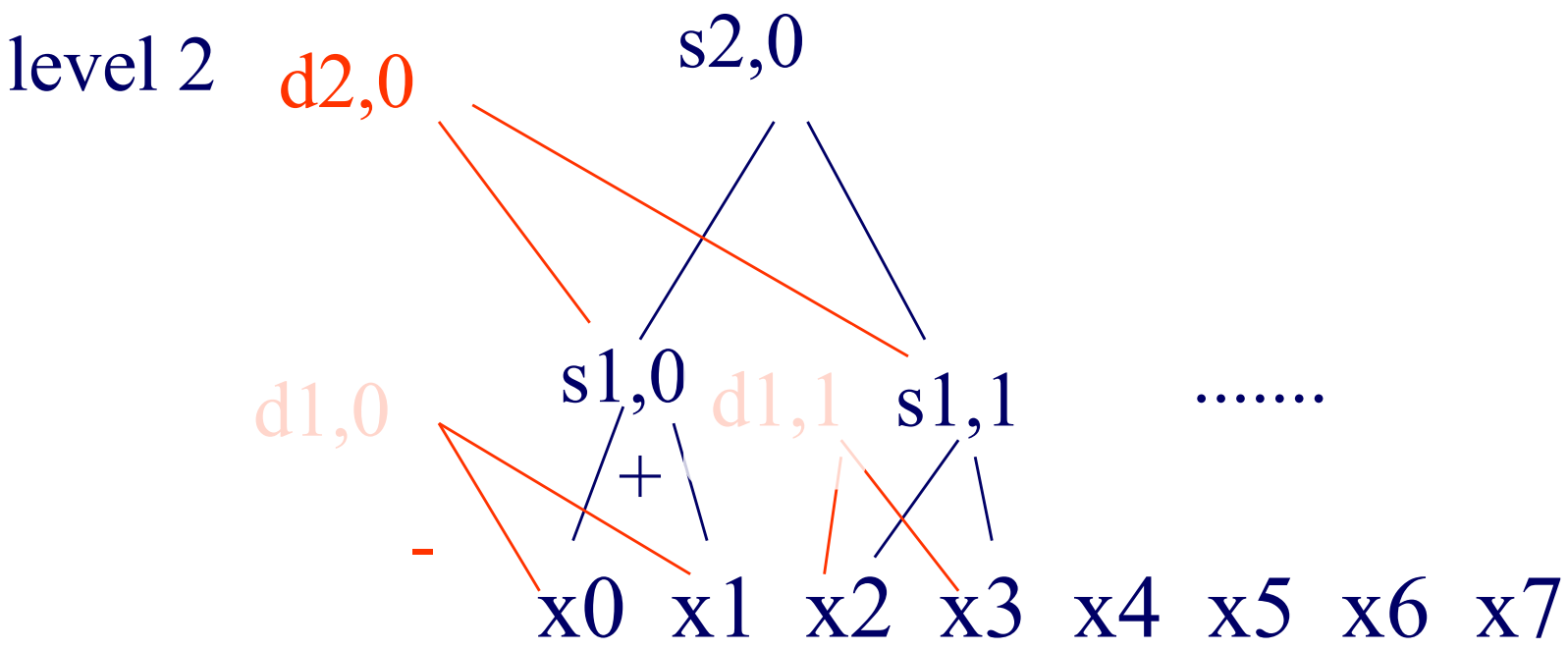
Wavelets - construction



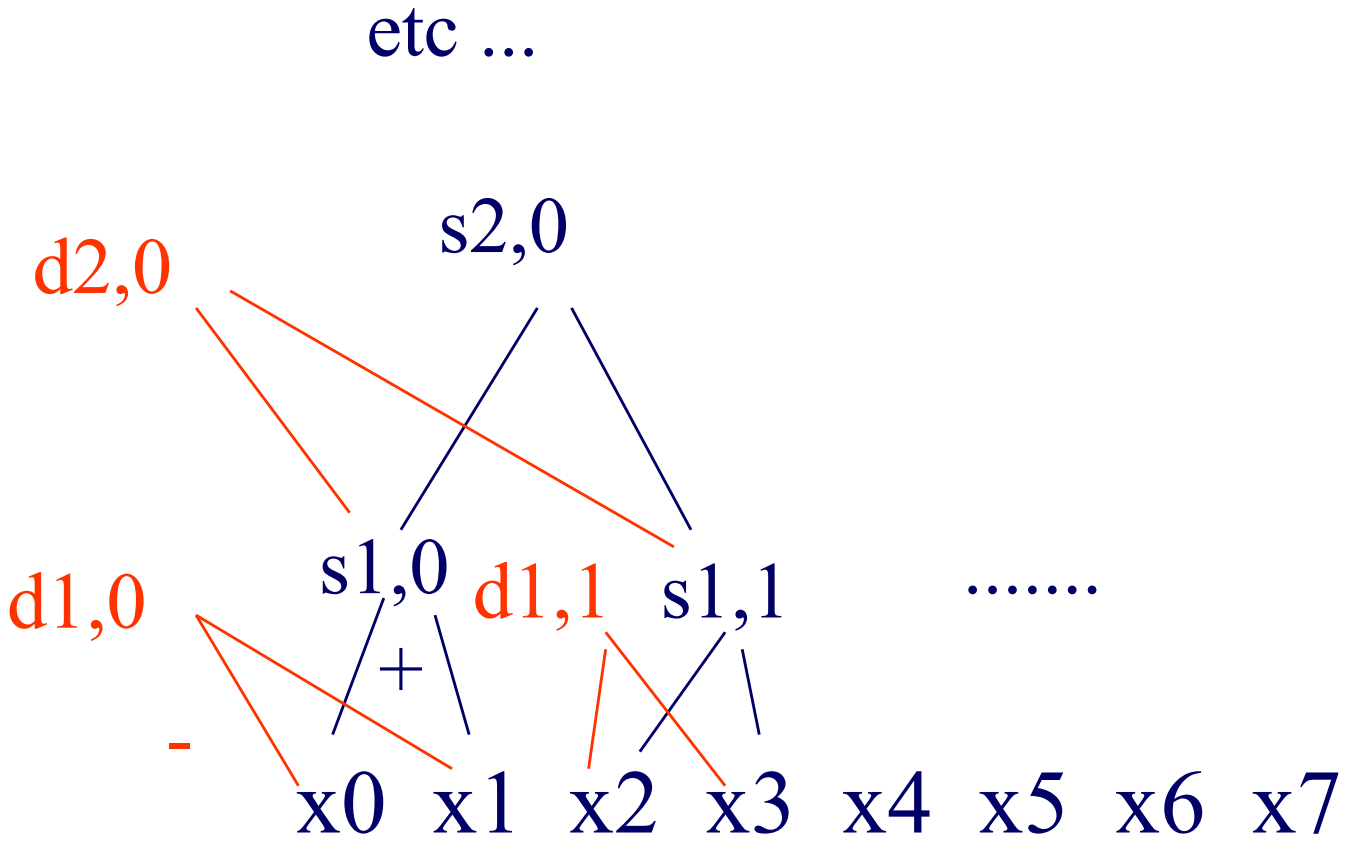
Wavelets - construction



Wavelets - construction



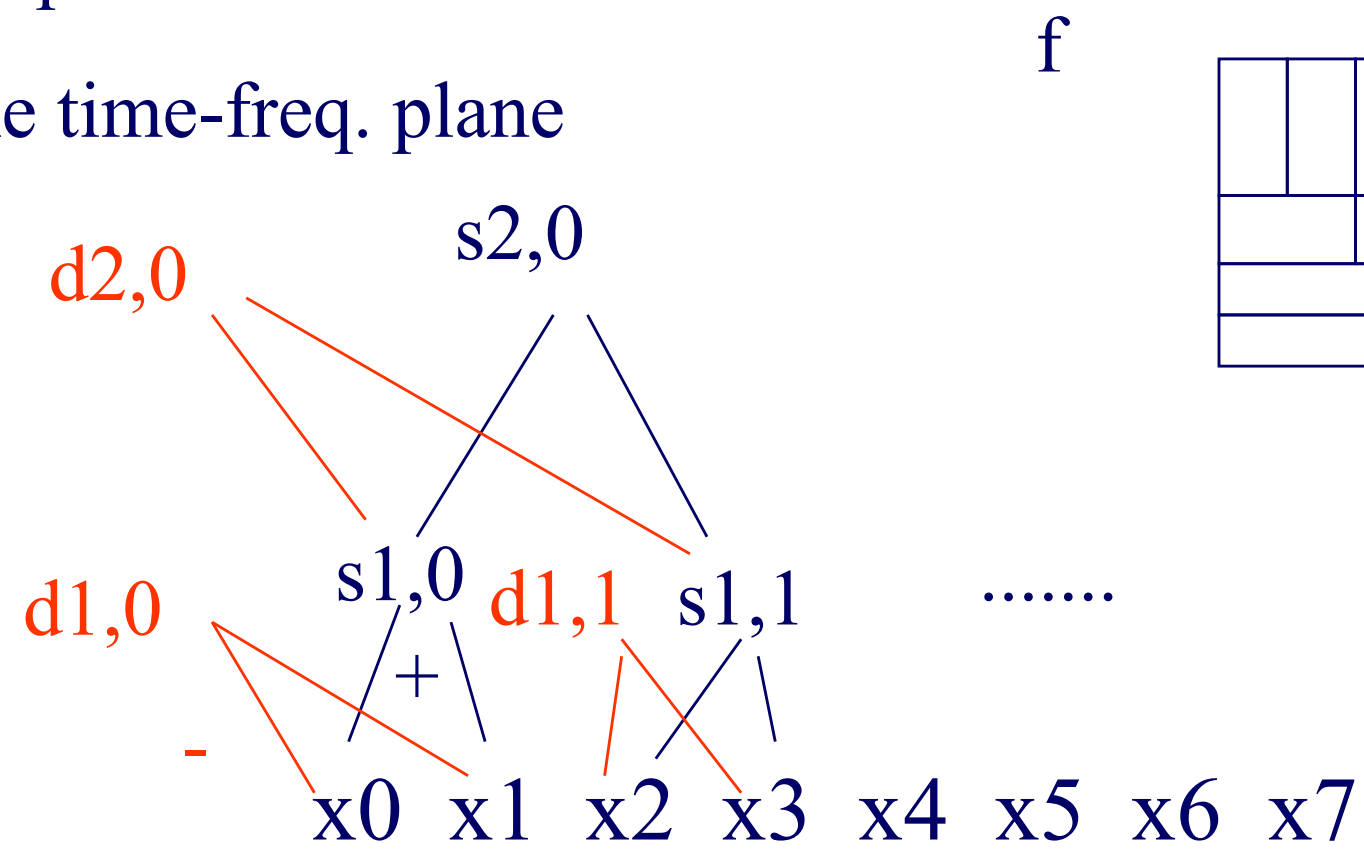
Wavelets - construction



Wavelets - construction

Q: map each coefficient

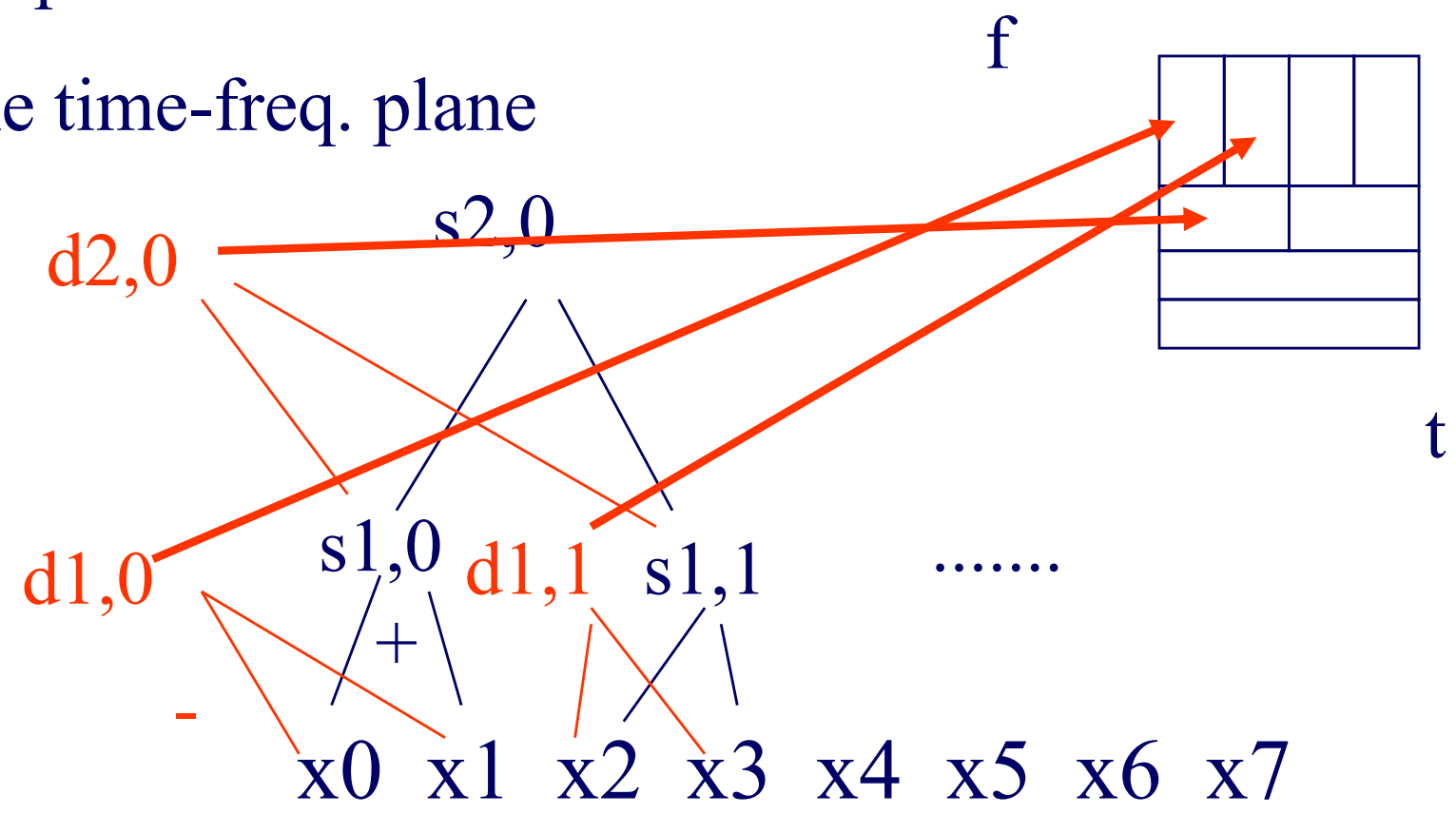
on the time-freq. plane



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,

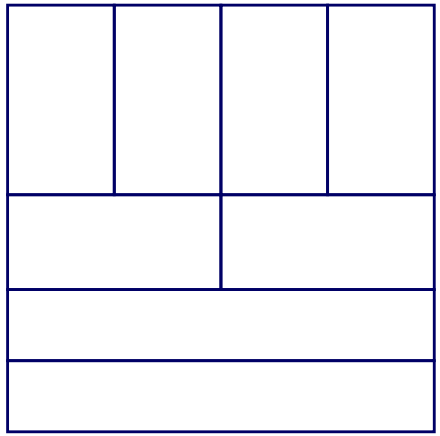
- there are *many* wavelet bases: Haar, Daubechies-4, Daubechies-6, Coifman, Morlet, Gabor, ...

Part 1 - Outline

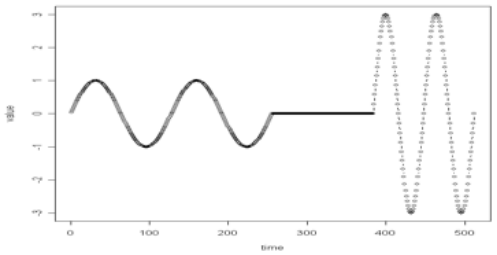
- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP
  - DFT
  - DWT
    - Definition of DWT and properties
    - how to read the DWT scalogram
- P1.3. Linear Forecasting
- …

Wavelets - Drill#1:

- Q: baritone/silence/soprano - DWT?



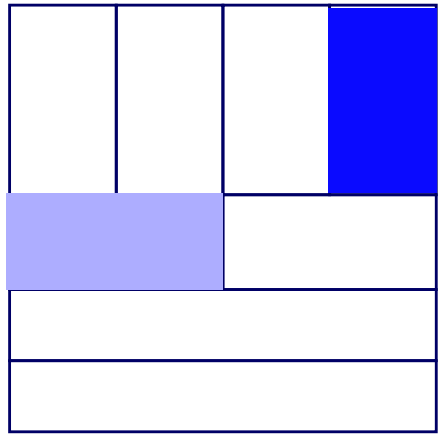
value



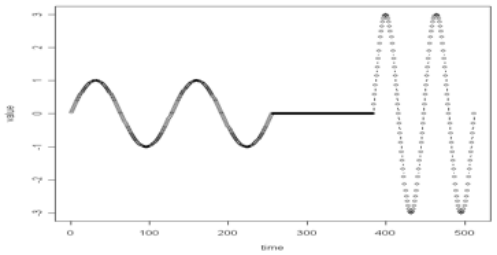
time

Wavelets - Drill#1:

- Q: baritone/silence/soprano - DWT?



value

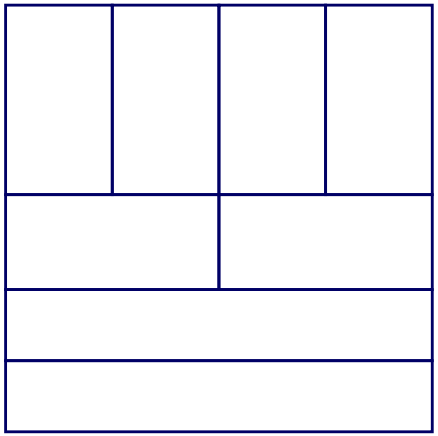


time

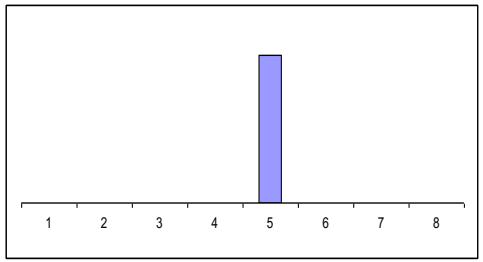
Wavelets - Drill#2:

- Q: spike - DWT?

f



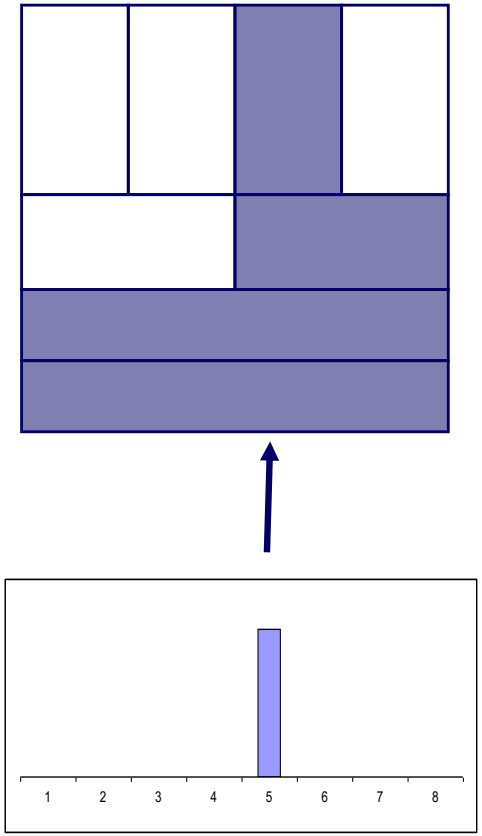
t



Wavelets - Drill#2:

- Q: spike - DWT?

f



t

0.00 0.00 **0.71** 0.00

0.00 **0.50**

-0.35

0.35

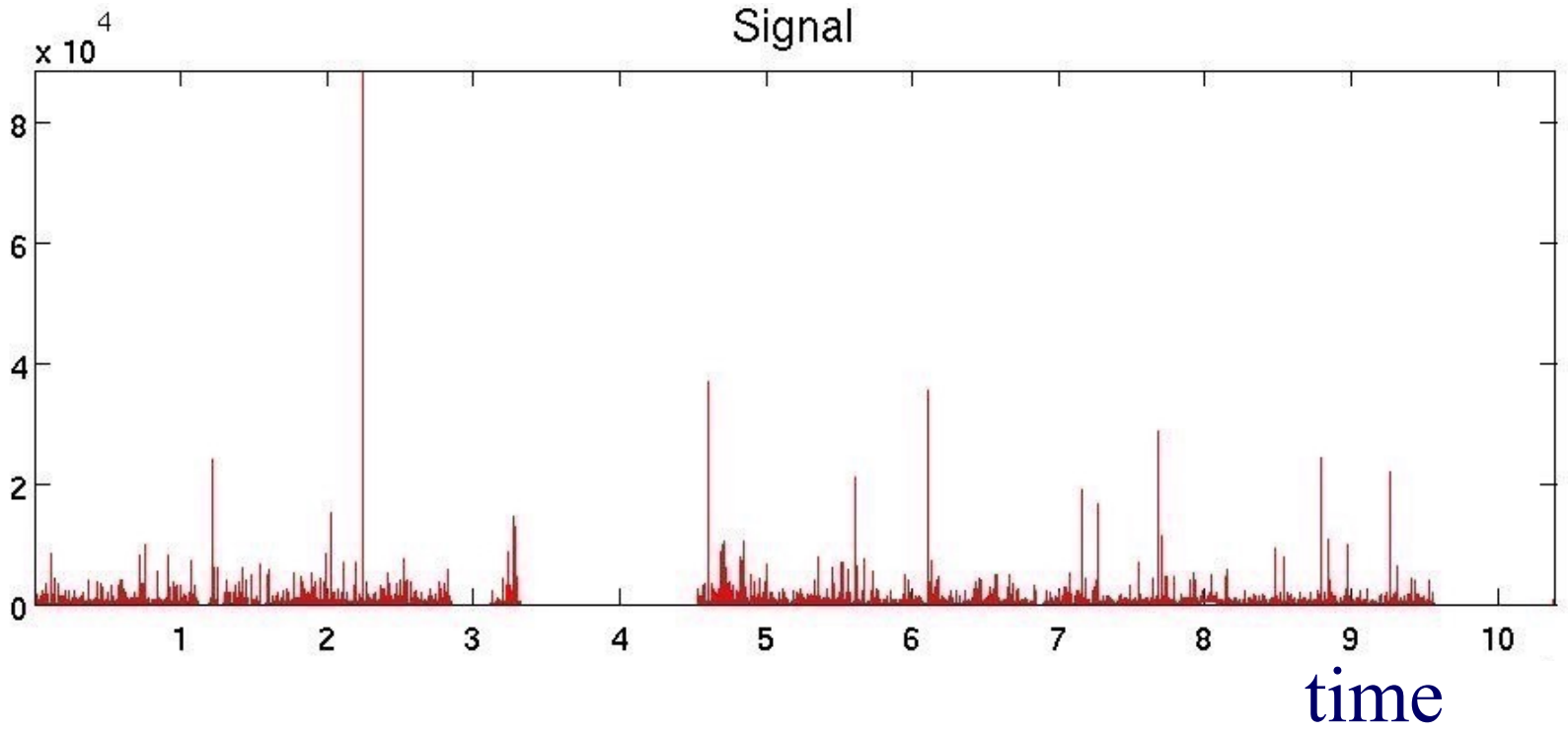
Wavelets in action

Carnegie Mellon

amazon

More examples (BGP updates)

#updates



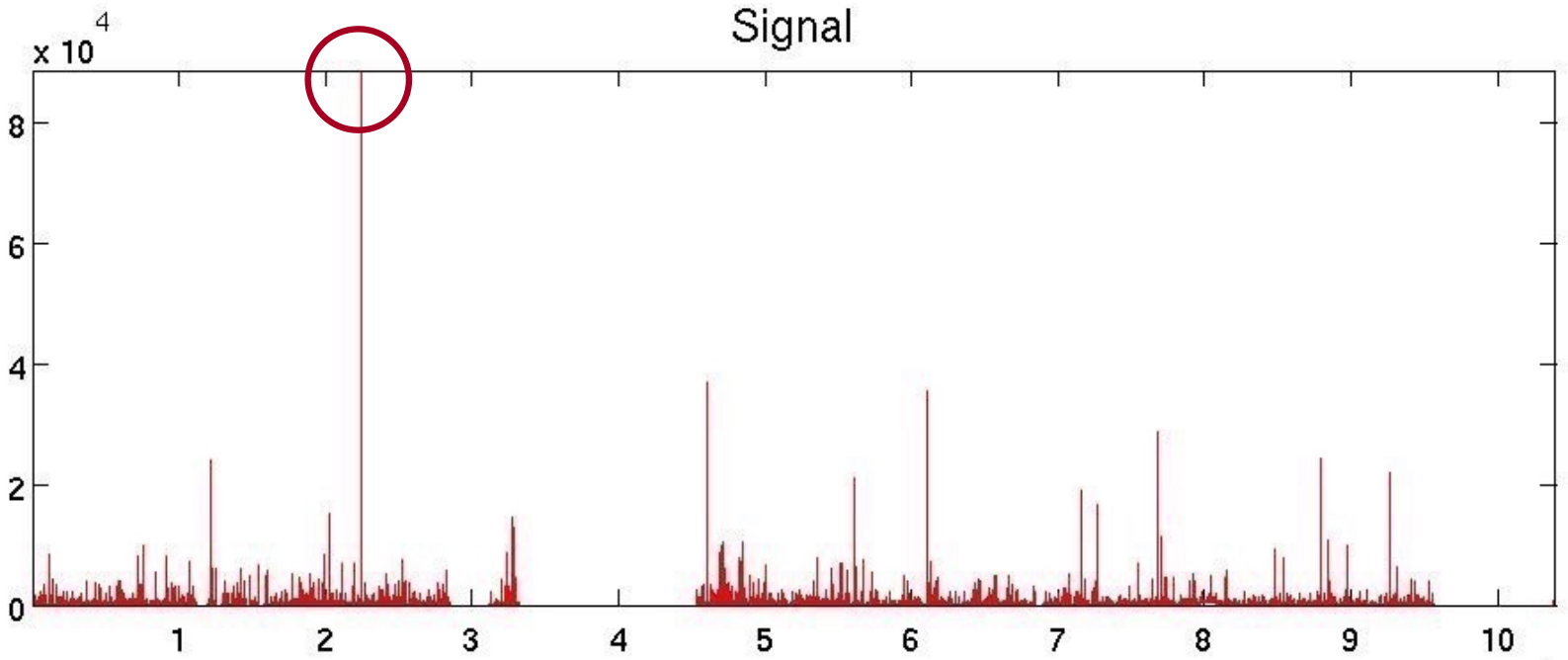
BGP-lens: Patterns and Anomalies in Internet Routing Updates B. Aditya Prakash et al, SIGKDD 2009

Carnegie Mellon

amazon

More examples (BGP updates)

#updates



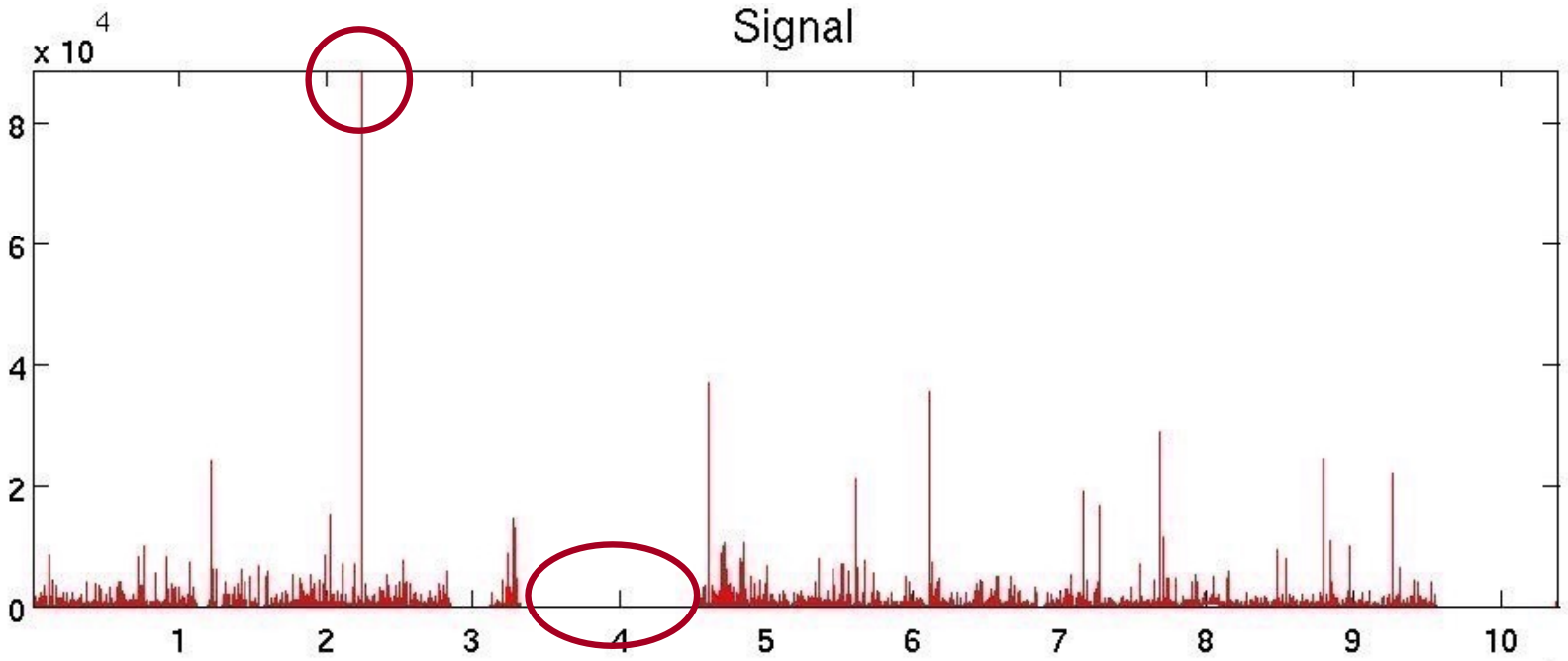
BGP-lens: Patterns and Anomalies in Internet Routing Updates B. Aditya Prakash et al, SIGKDD 2009

Carnegie Mellon

amazon

More examples (BGP updates)

#updates

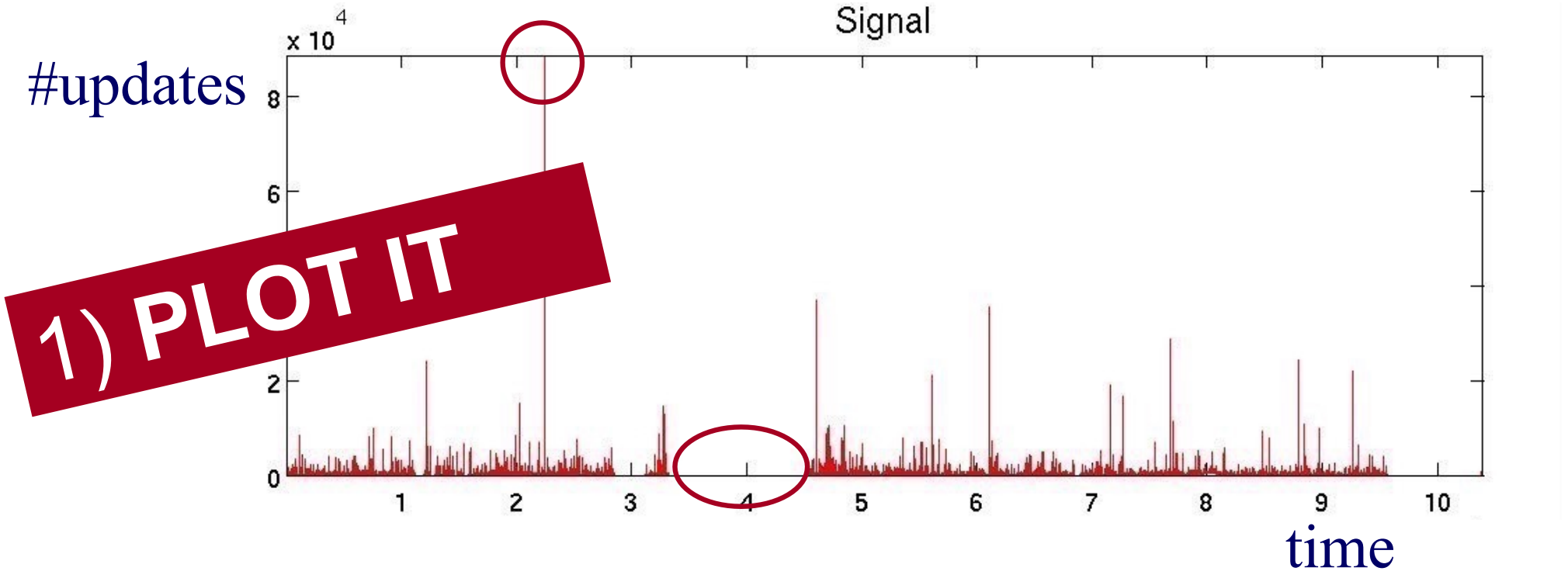


BGP-lens: Patterns and Anomalies in Internet Routing Updates B. Aditya Prakash et al, SIGKDD 2009

Carnegie Mellon

amazon

More examples (BGP updates)



1) PLOT IT

BGP-lens: Patterns and Anomalies in Internet Routing Updates B. Aditya Prakash et al, SIGKDD 2009

More examples (BGP updates)

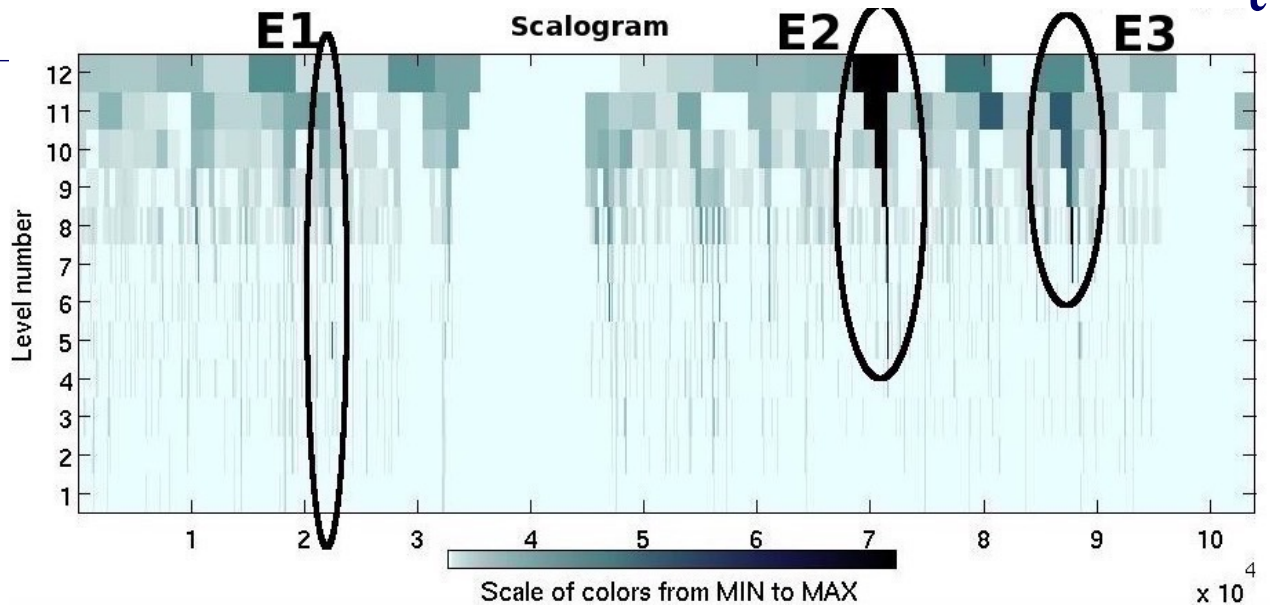
1) PLOT IT

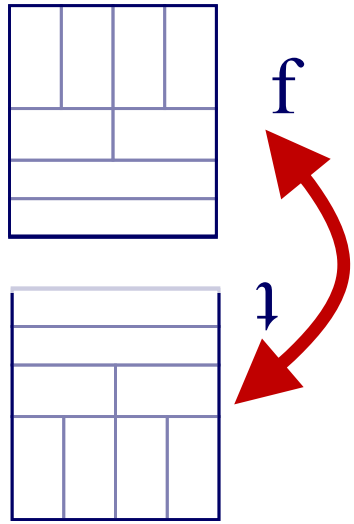
2) DFT / DWT

More examples (BGP updates)

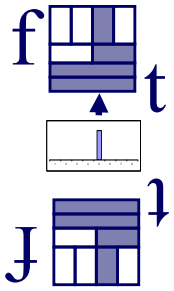
Low freq.: omitted

freq.

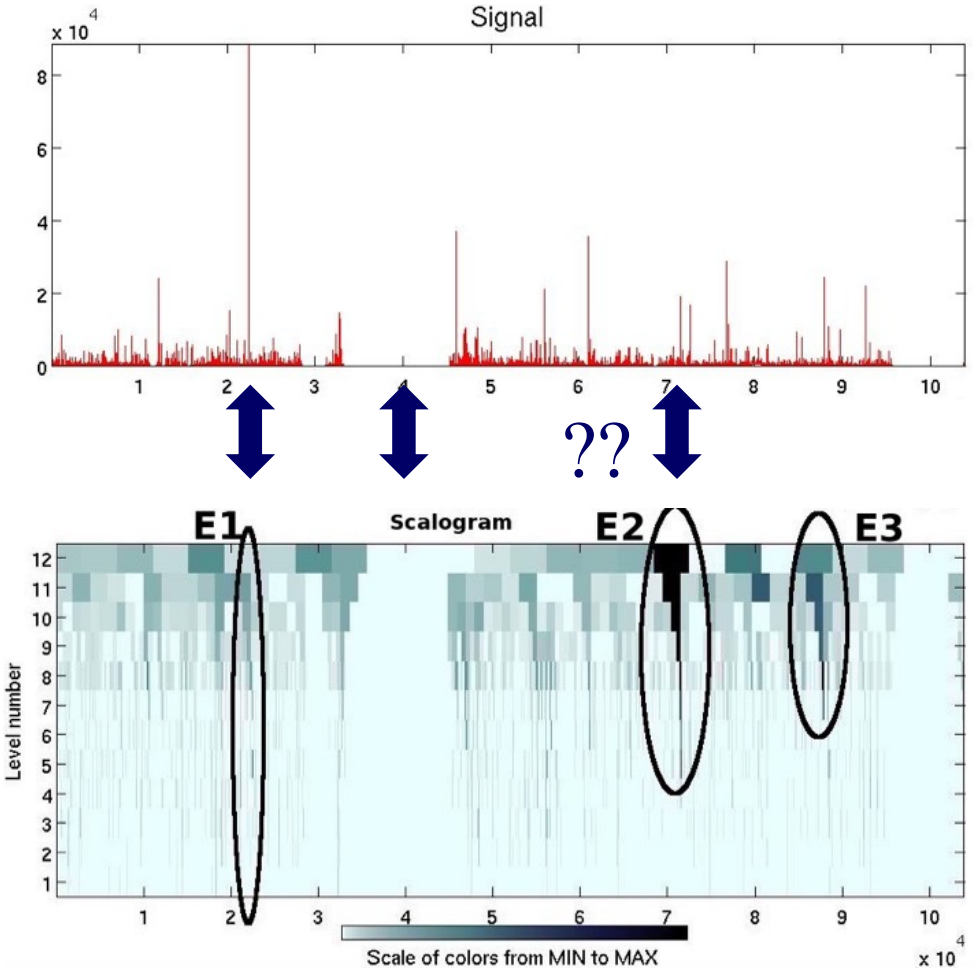




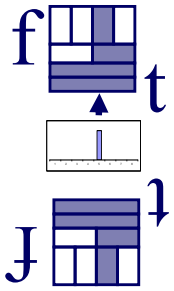
More examples (BGP updates)

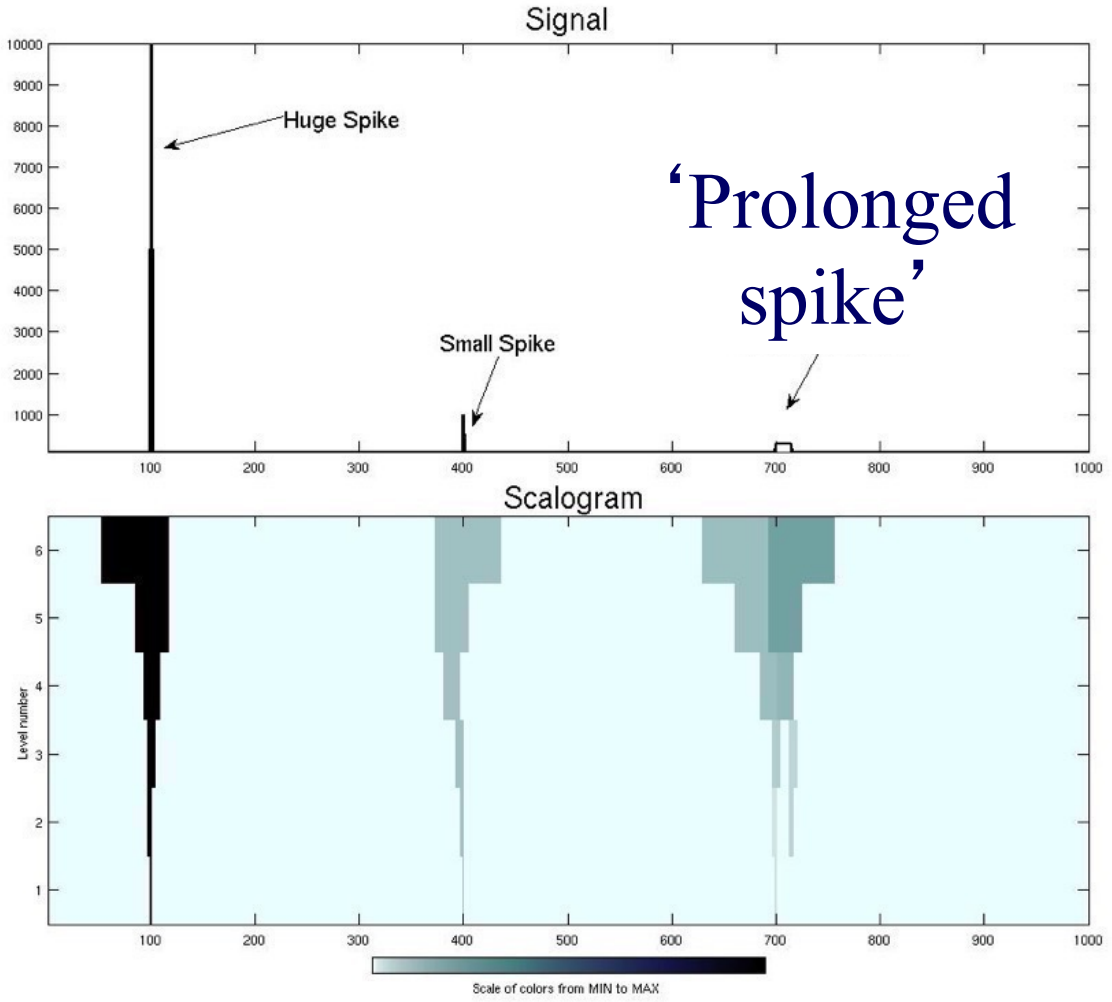


freq. ↓

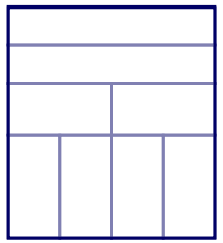


More examples (BGP updates)





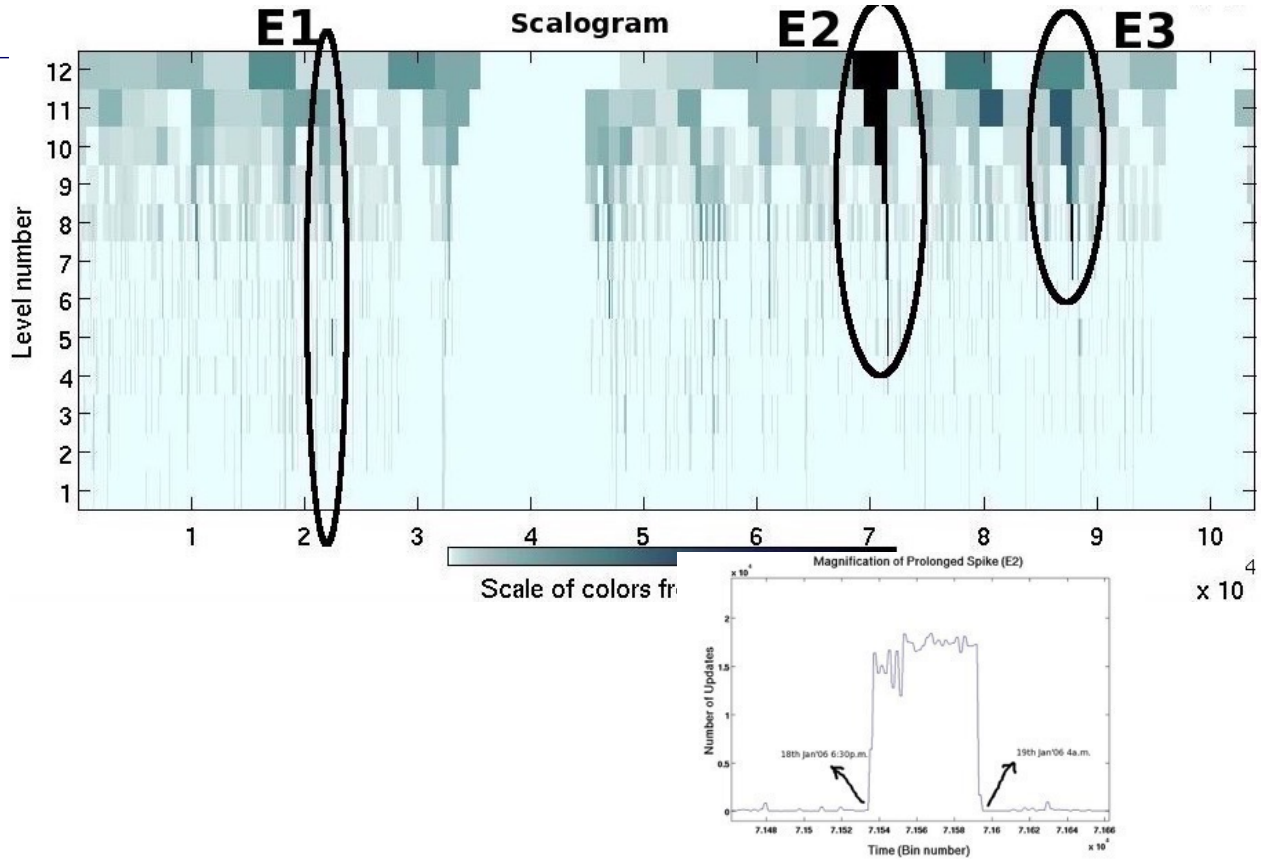
freq.



More examples (BGP updates)

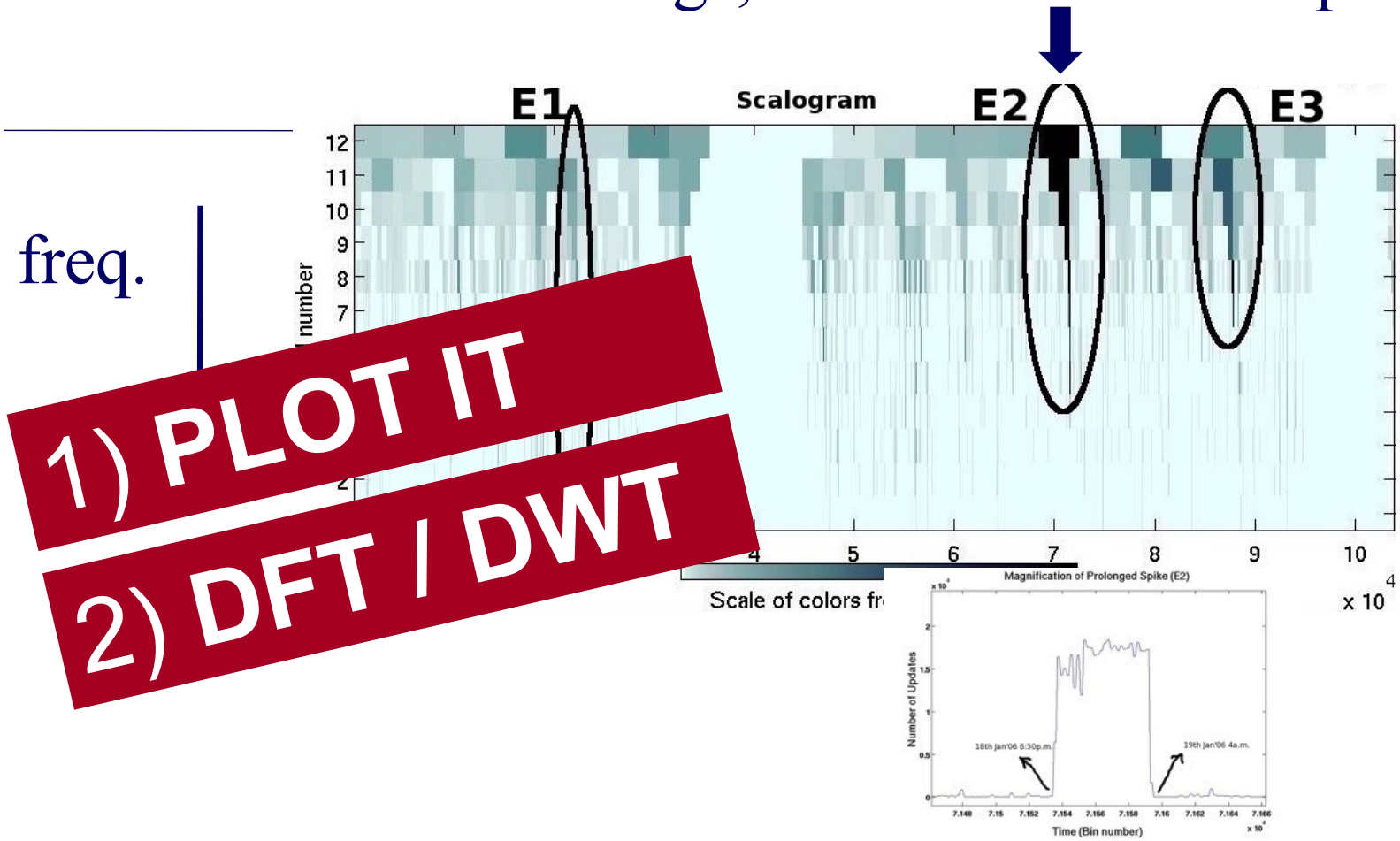
15K msgs, for several hours: 6pm-4am

freq.



More examples (BGP updates)

15K msgs, for several hours: 6pm-4am



1) PLOT IT

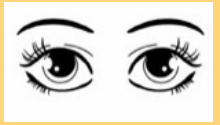
2) DFT / DWT

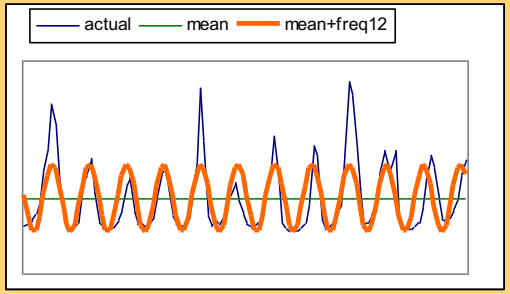
Advantages of Wavelets

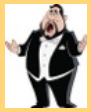
- Better compression (better RMSE with same number of coefficients - used in JPEG-2000)
- fast to compute (usually: O(n))
- good for 'spikes'
- good for de-noising
- mammalian eye and ear: Gabor wavelets

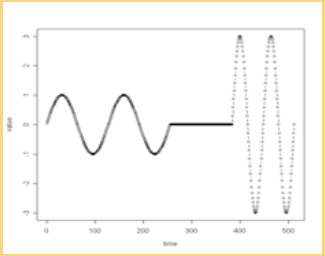
Part 1.2: Conclusions

- DFT spots periodicities
- **DWT** : multi-resolution - matches processing of mammalian ear/eye better
- Both: powerful tools for **compression**, **pattern detection** in real signals



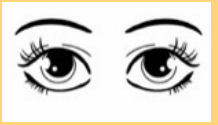






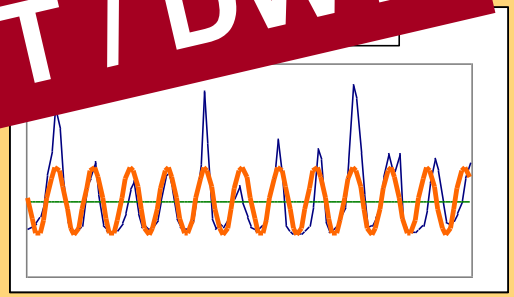
Part 1.2: Conclusions

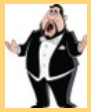
- DFT spots periodicities
- **DWT** : multi-resolution - matches processing of mammalian ear/eye better
- R tools for **compression**, **n** in real signals

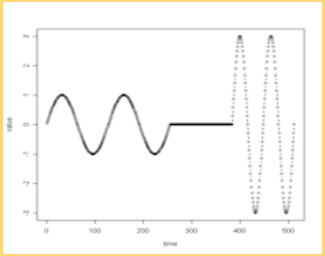


1) PLOT IT

2) DFT / DWT







Resources: software

- *xwpl:* open source wavelet package from Yale, with excellent GUI

- PyWavelets

- R, 'wavelets'

- Mathematica, matlab, etc

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

Part 1 - Outline

- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP
- P1.3. Linear Forecasting
- P1.4. Non-linear forecasting
- P1.5. Tensors
- Conclusions

Part 1.3:

Linear Forecasting

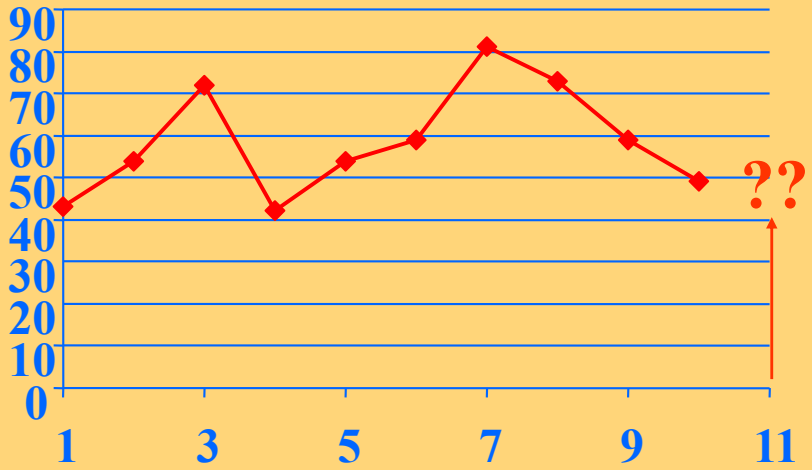
Forecasting

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

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

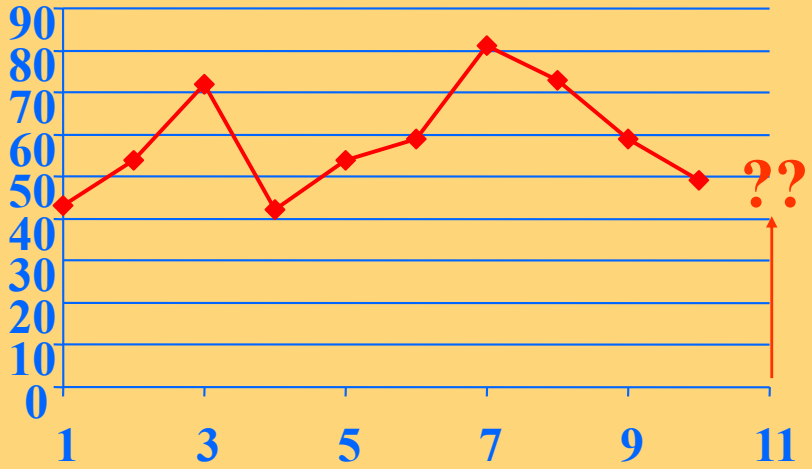
Problem#2: Forecast

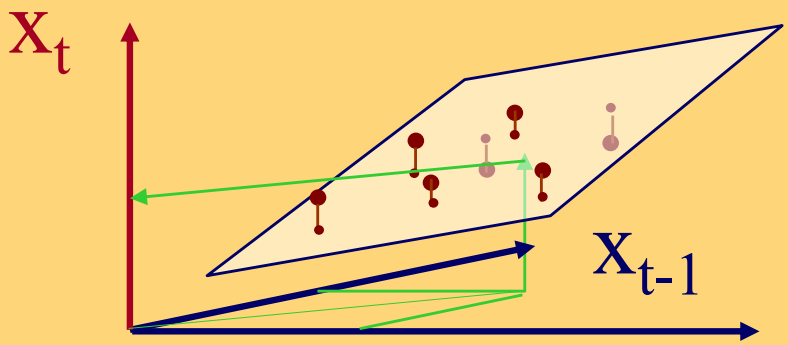
- given x_{t-1}, x_{t-2}, …,
- Q: forecast x_t



Solution: AR(IMA)

- given x_{t-1}, x_{t-2}, …,
- Q: forecast x_t
- A: AR(IMA) = Box-Jenkins (< Holt-Winters, Kalman)



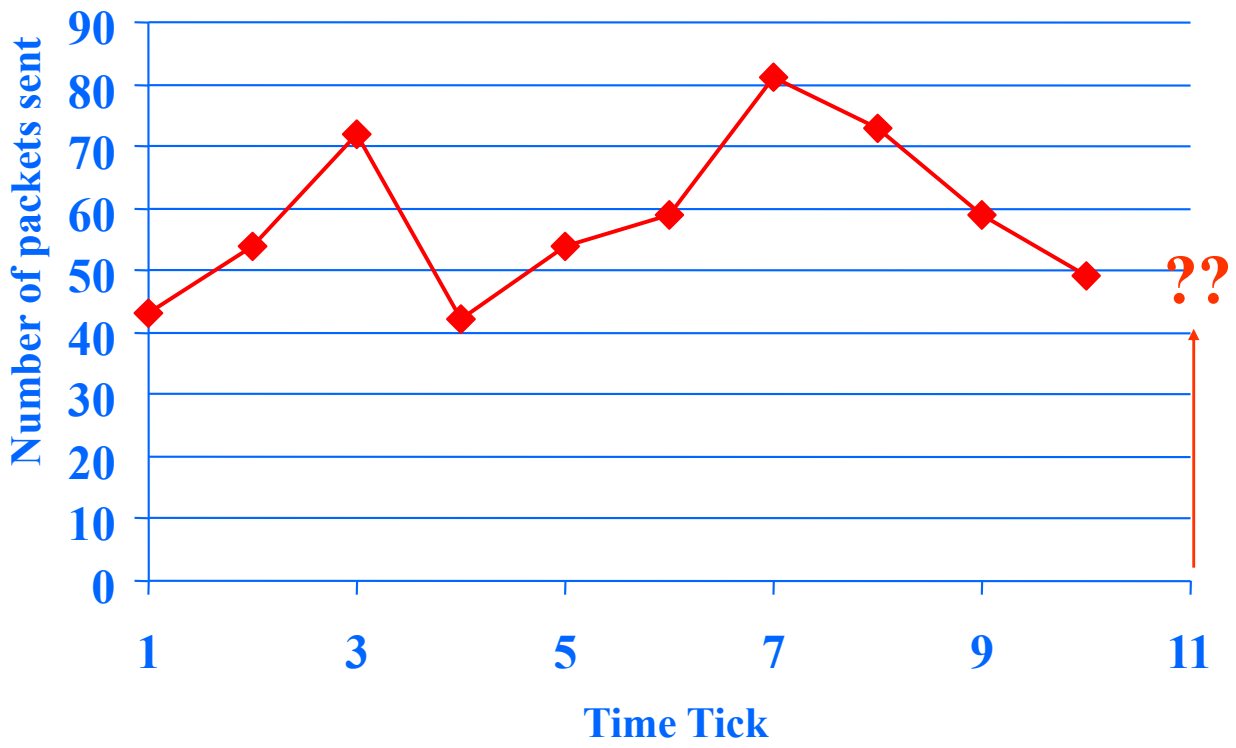


Part 1 - Outline

- Motivation
- ...
- P1.3. Linear Forecasting
  - Auto-regression: Least Squares; RLS
  - Co-evolving time sequences
  - Examples
  - Conclusions

Problem#2: Forecast

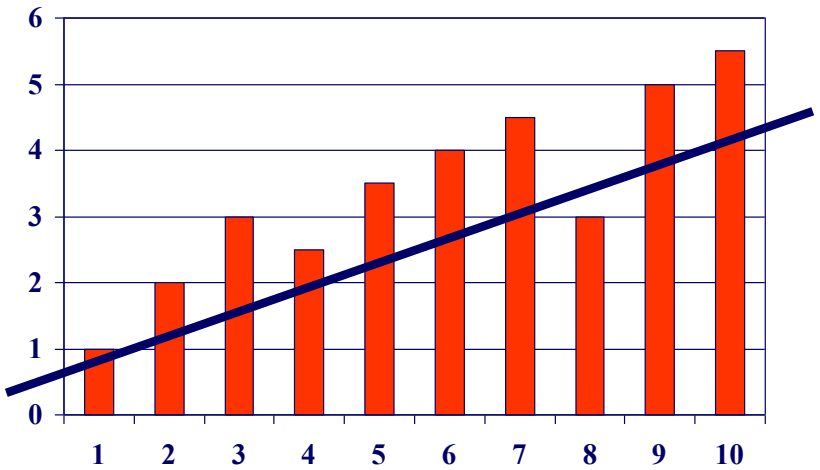
- Example: give x_{t-1}, x_{t-2}, …, forecast x_t



Forecasting: Preprocessing

MANUALLY:

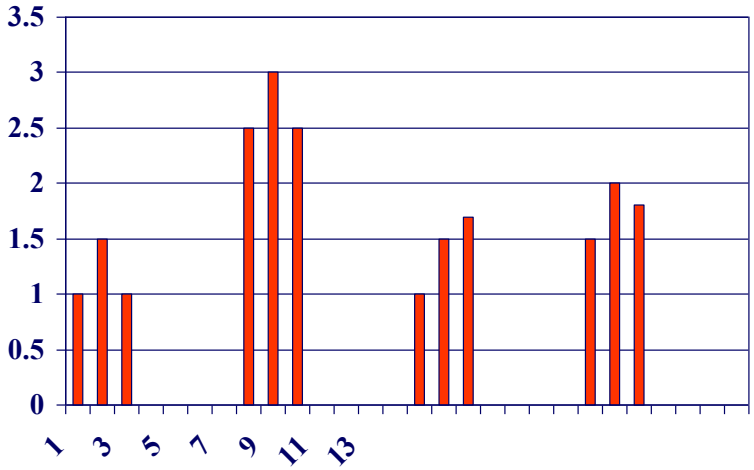
remove trends



time

spot periodicities

7 days



time

Problem#2: Forecast

- Solution: try to express

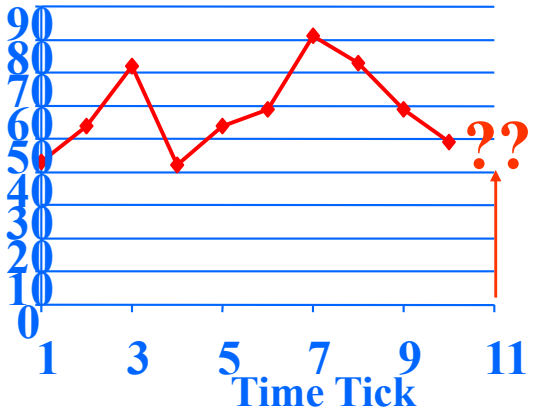
  x_t

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

  (up to a window of w)

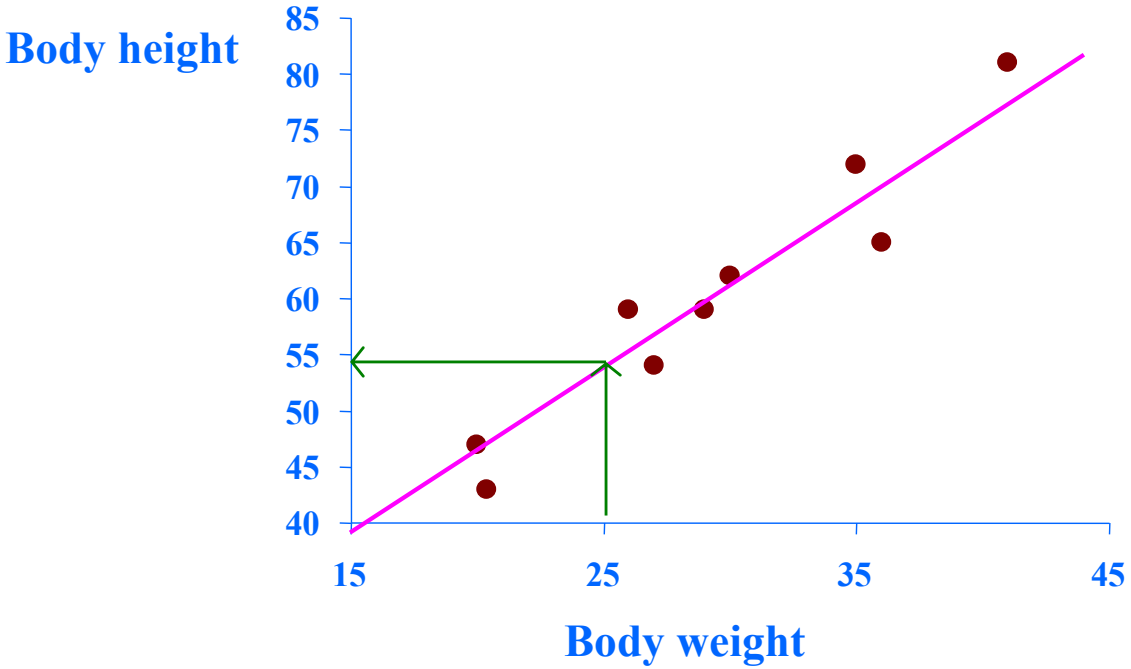
Formally:

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



Linear Regression: idea

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



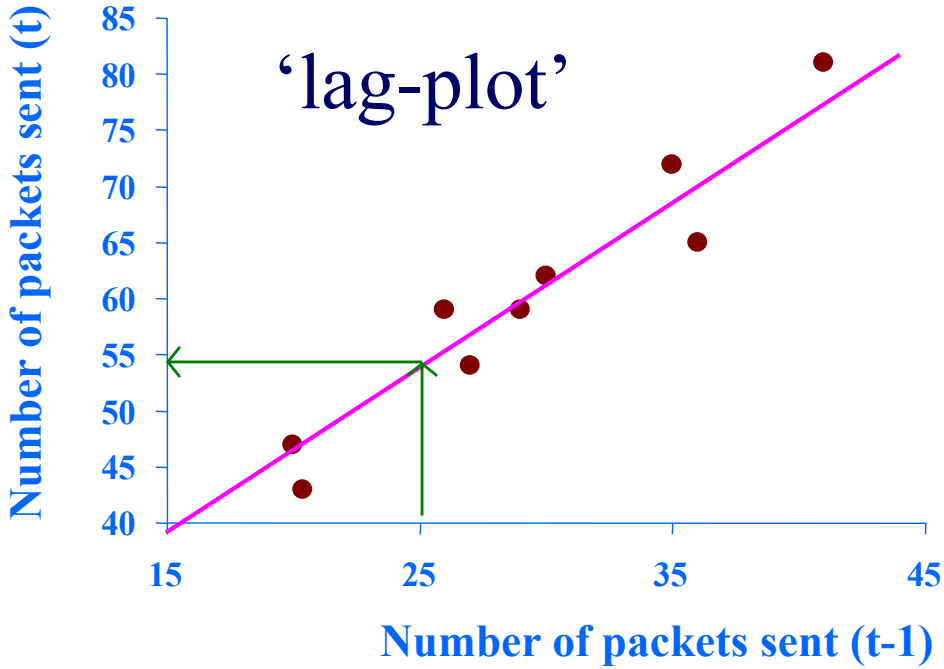
- express what we don't know (= 'dependent variable')
- as a linear function of what we know (= 'indep. variable(s)')

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	??



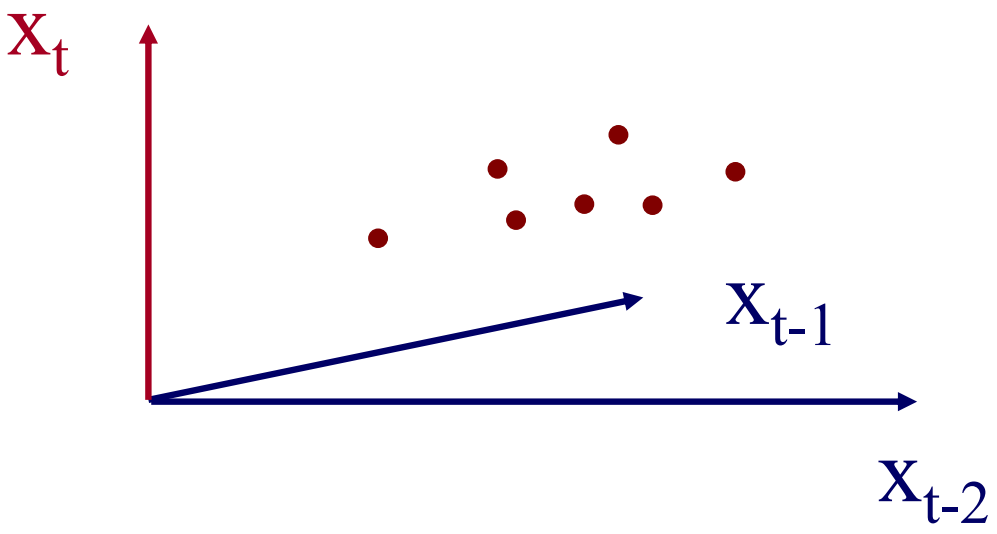
- lag w=1
- Dependent variable = # of packets sent (S [t])
- Independent variable = # of packets sent (S[t-1])

Part 1 - Outline

- Motivation
- ...
- P1.3. Linear Forecasting
  - Auto-regression: Least Squares; **RLS**
  - Co-evolving time sequences
  - Conclusions

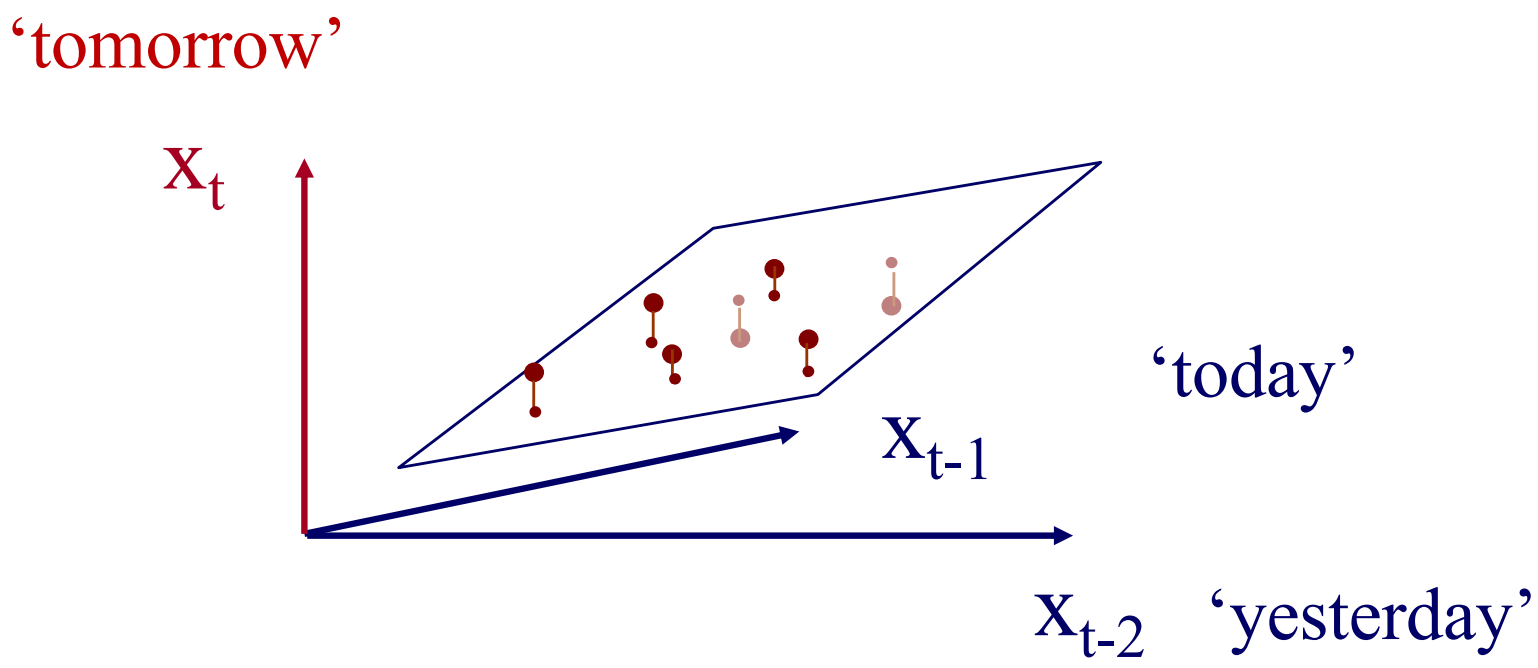
More details:

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



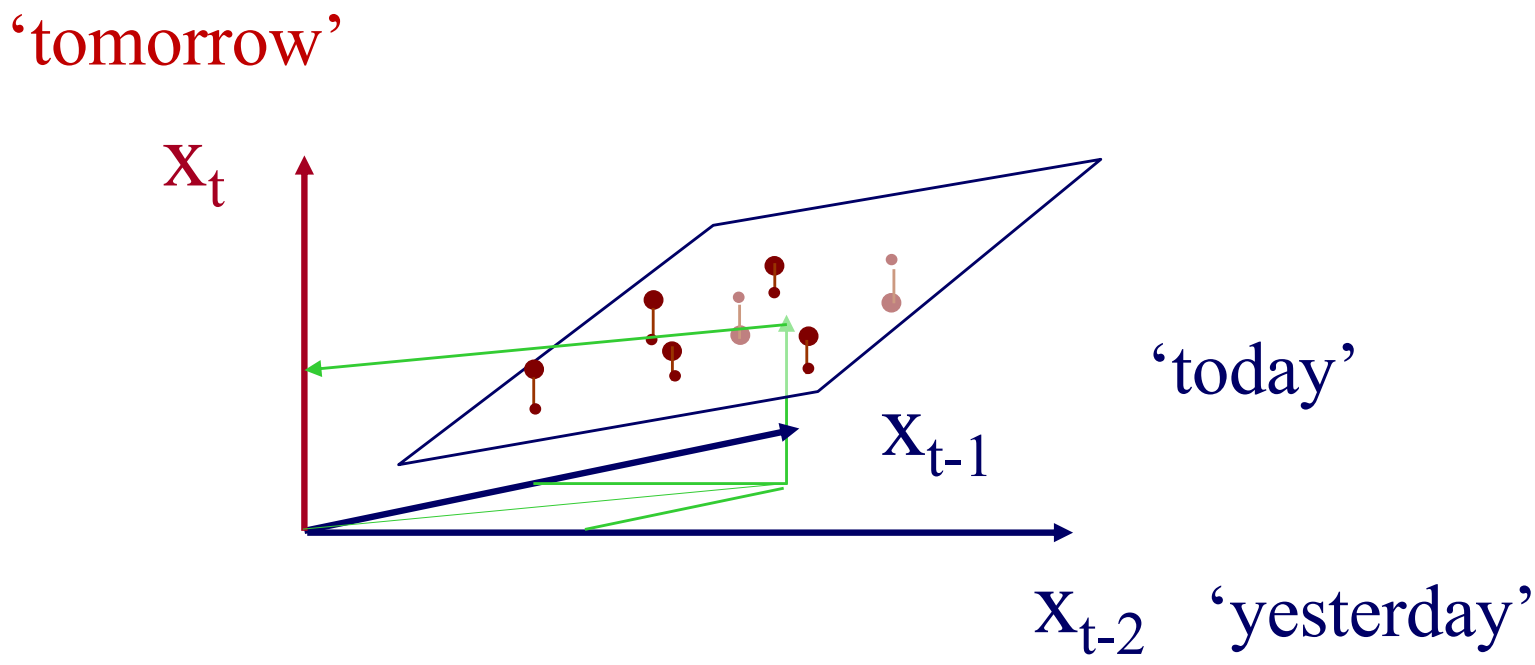
More details:

- 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!)



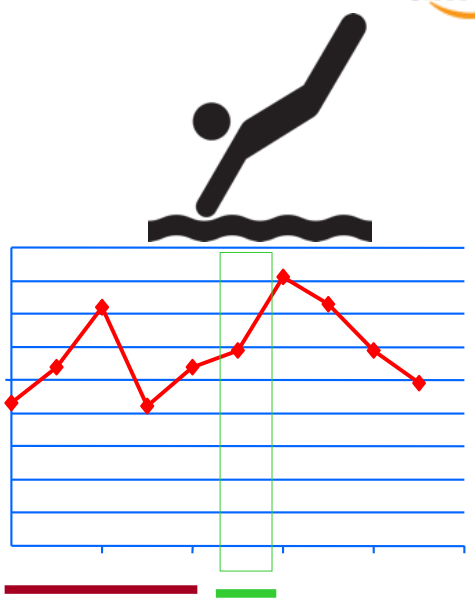
More details:

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

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

- 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

More details:



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

Ind-var1 Ind-var-w

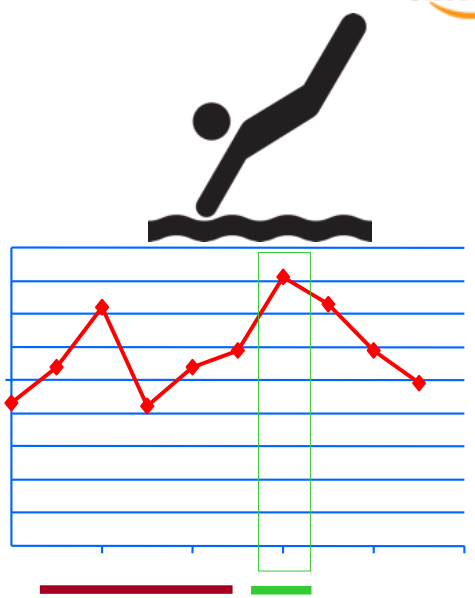
time

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

Carnegie Mellon

amazon

More details:



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

Ind-var1 Ind-var-w

time

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

More details

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

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

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

Carnegie Mellon

amazon

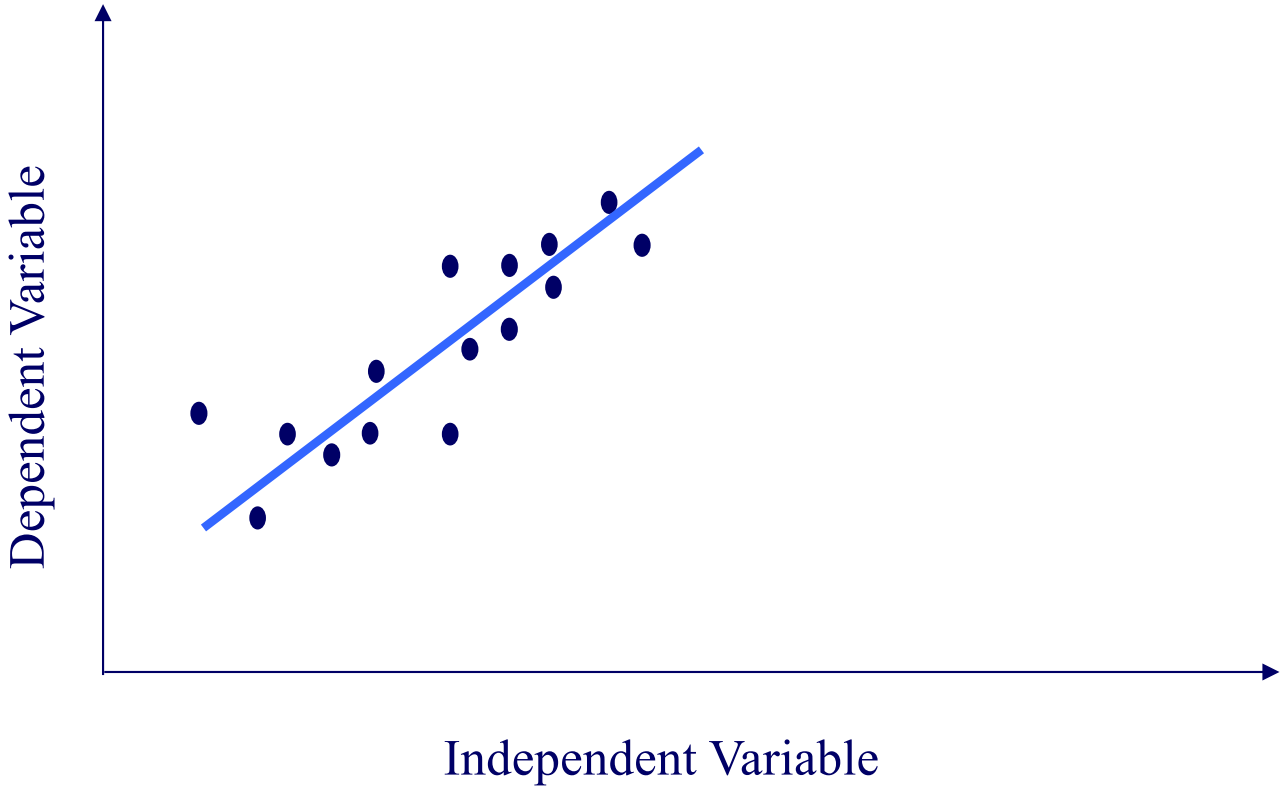
Even more details

- 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:

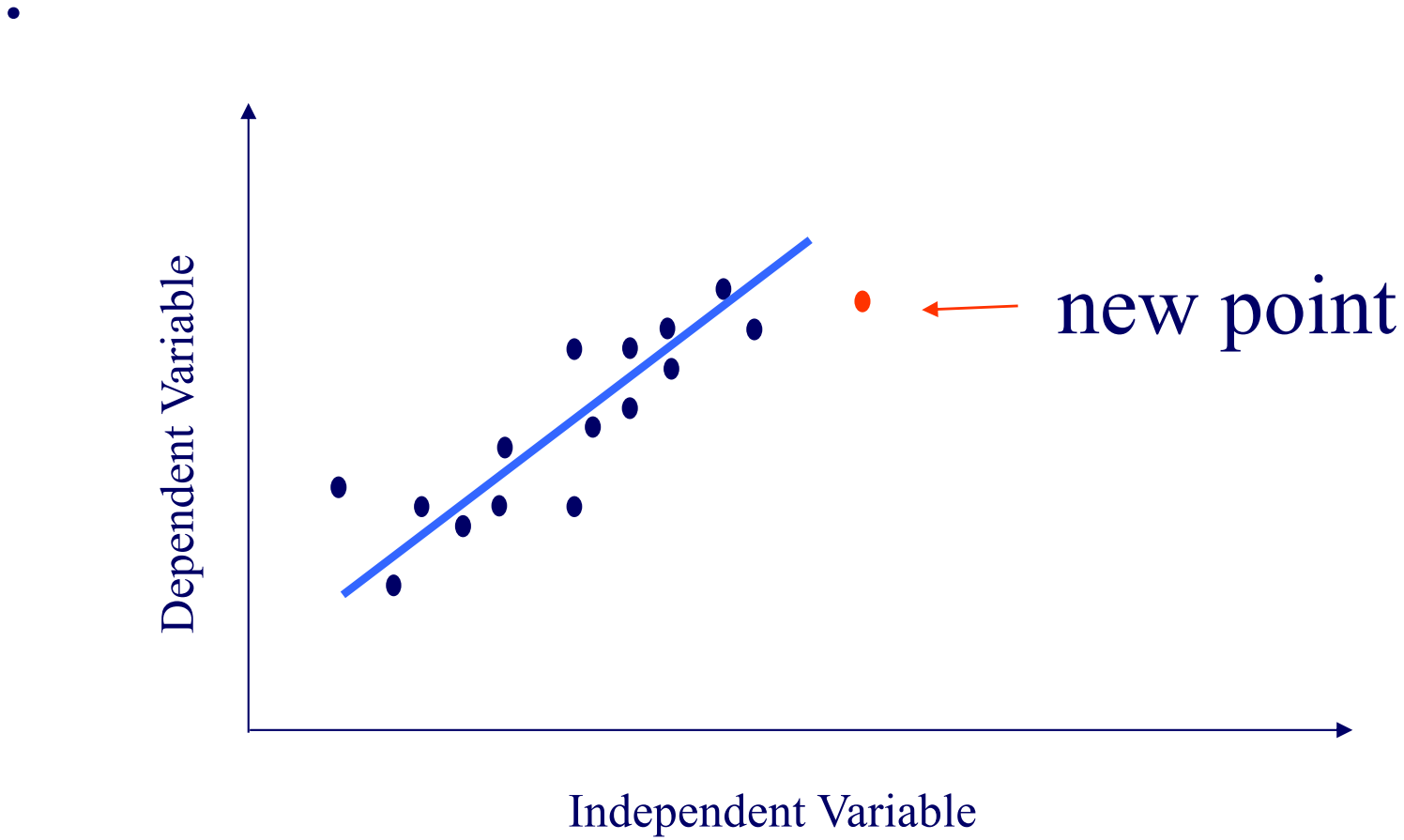
[Yi+00] Byoung-Kee Yi et al.: *Online Data Mining for Co-Evolving Time Sequences*, ICDE 2000

Even more details

- Given:

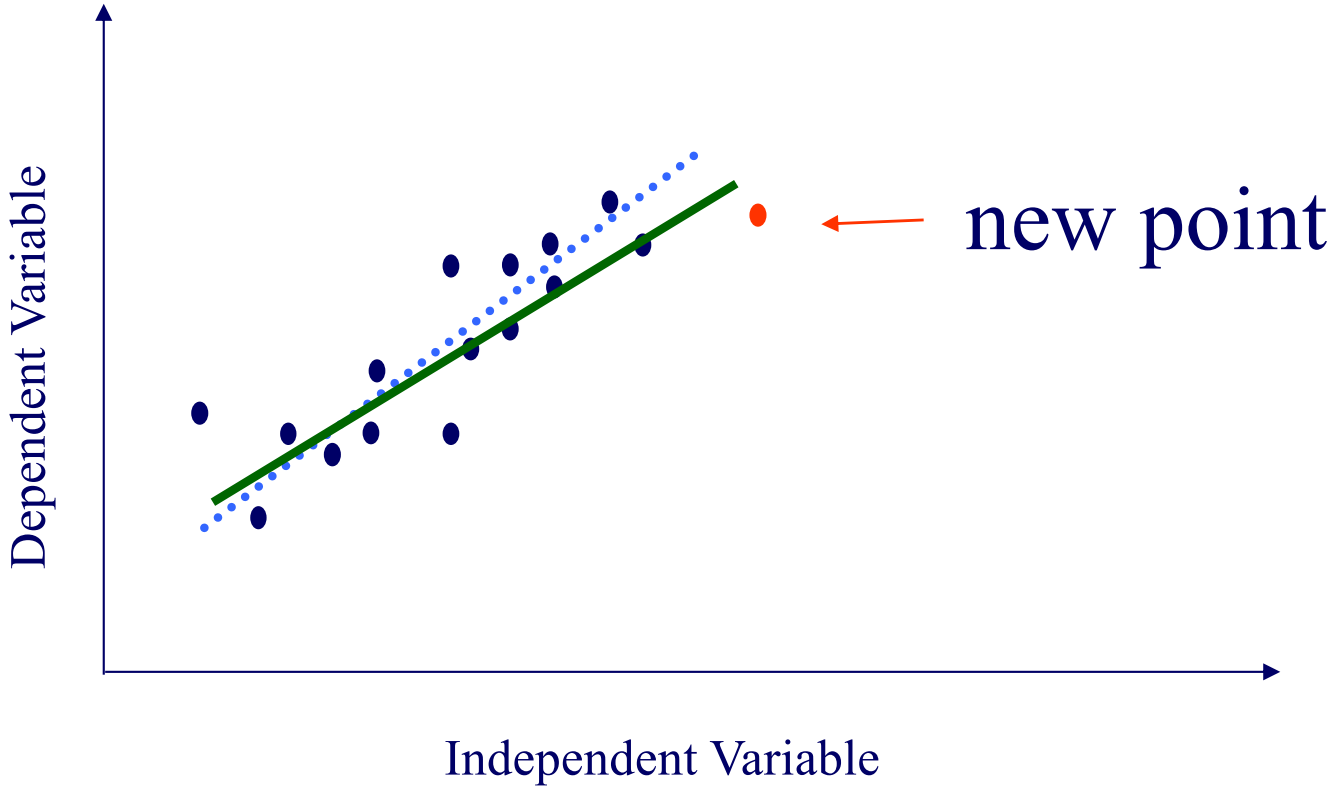


Even more details



Even more details

RLS: quickly compute new best fit

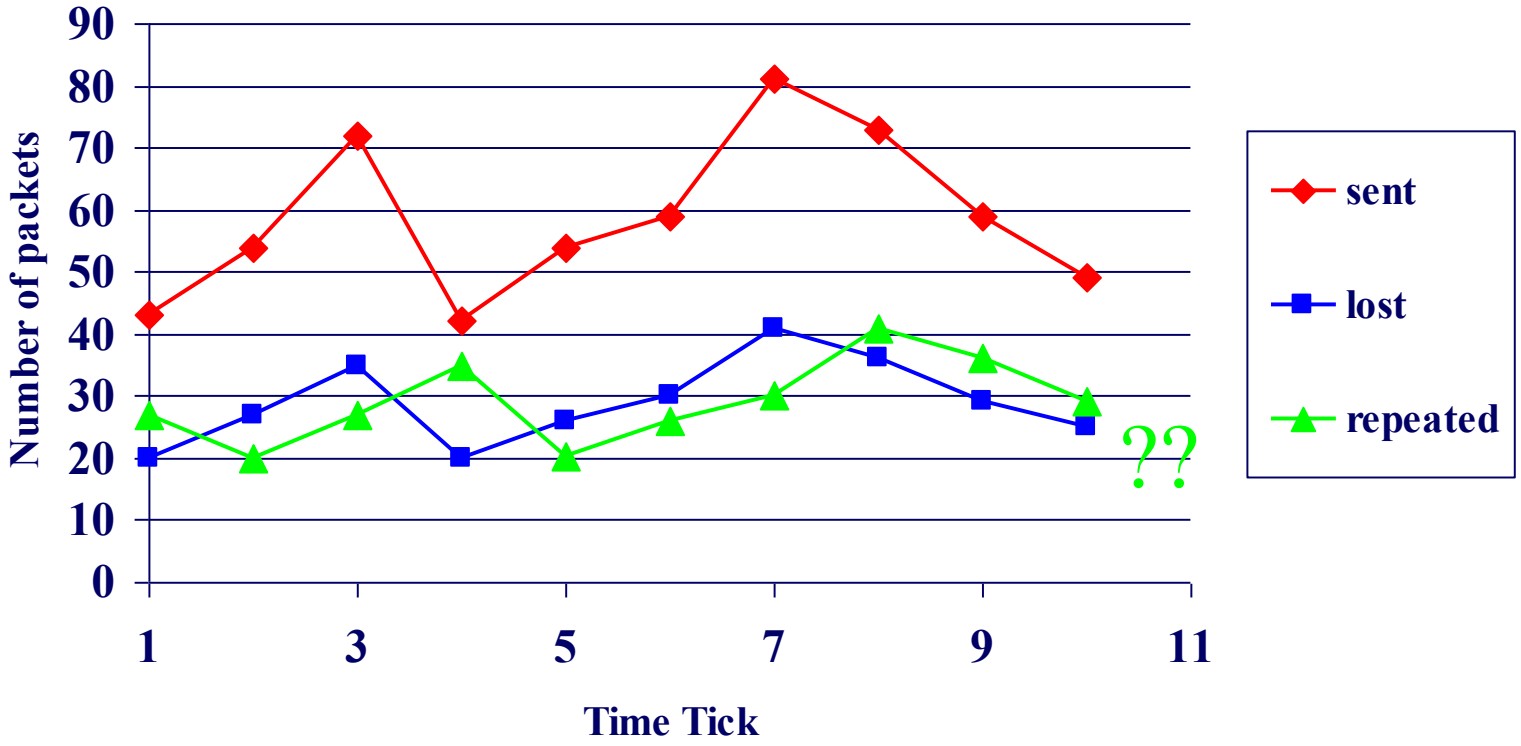


Part 1 - Outline

- Motivation
- ...
- P1.3. Linear Forecasting
  - Auto-regression: Least Squares; **RLS**
  - Co-evolving time sequences
  - Conclusions

Co-Evolving Time Sequences

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



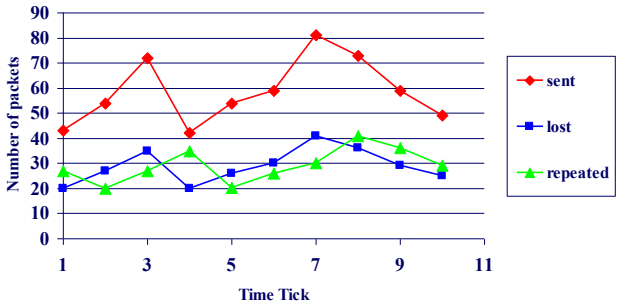
Solution:

Q: what should we do?

Solution: Vector ARIMA

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])

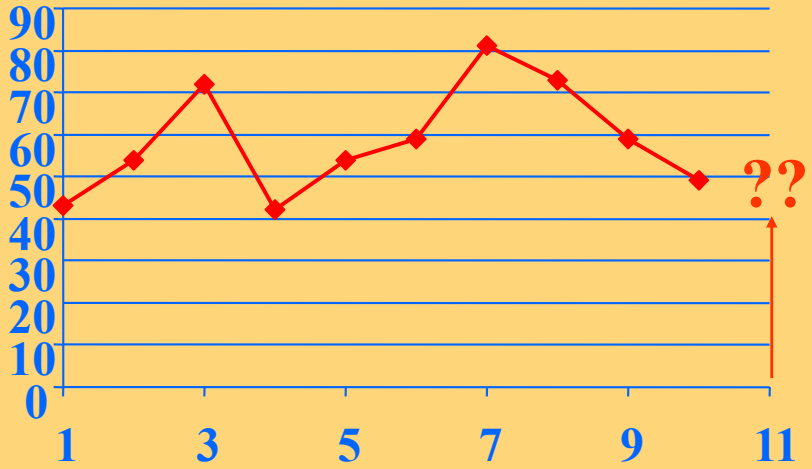


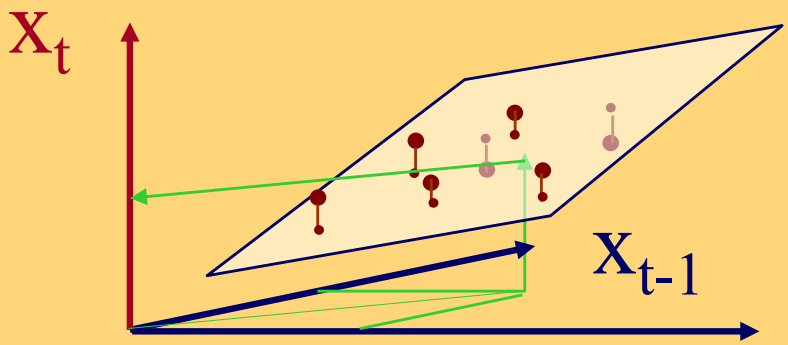
Part 1 - Outline

- Motivation
- ...
- P1.3. Linear Forecasting
  - Auto-regression: Least Squares; **RLS**
  - Co-evolving time sequences
  - Conclusions

Solution: AR(IMA)

- given x_{t-1}, x_{t-2}, …,
- Q: forecast x_t
- A: AR(IMA) = Box-Jenkins (< Holt-Winters, Kalman)





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)
- Kalman filters: KDD'10 tutorial

  https://lileicc.github.io/TALKS/2010-KDD/kdd2010-tut-partB.pdf

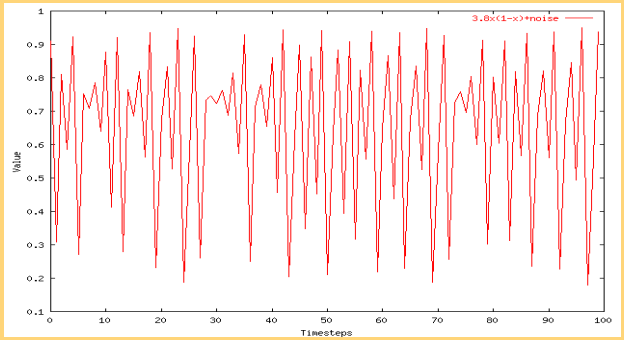
Part 1.4: chaos and non-linear forecasting

Part 1 - Outline

- …
- P1.3. Linear forecasting
- P1.4. Non-linear forecasting
  - Problem
  - Idea
  - How-to
  - Experiments
  - (Lotka-Volterra equations)
  - Conclusions
- P1.5. Tensors

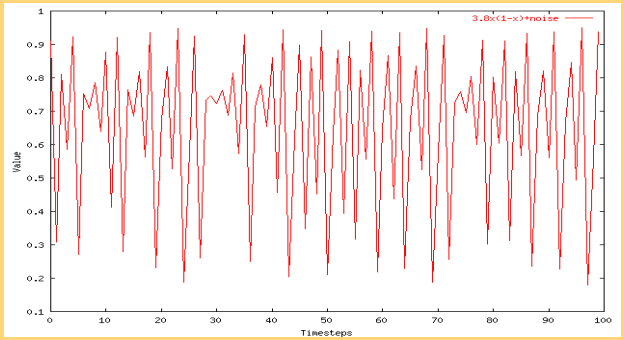
Problem: Forecast

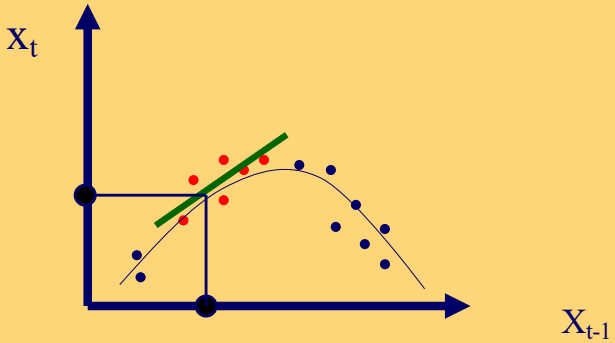
- given x_{t-1}, x_{t-2}, …, ('chaotic'/non-linear)
- Q: forecast x_t



Solution

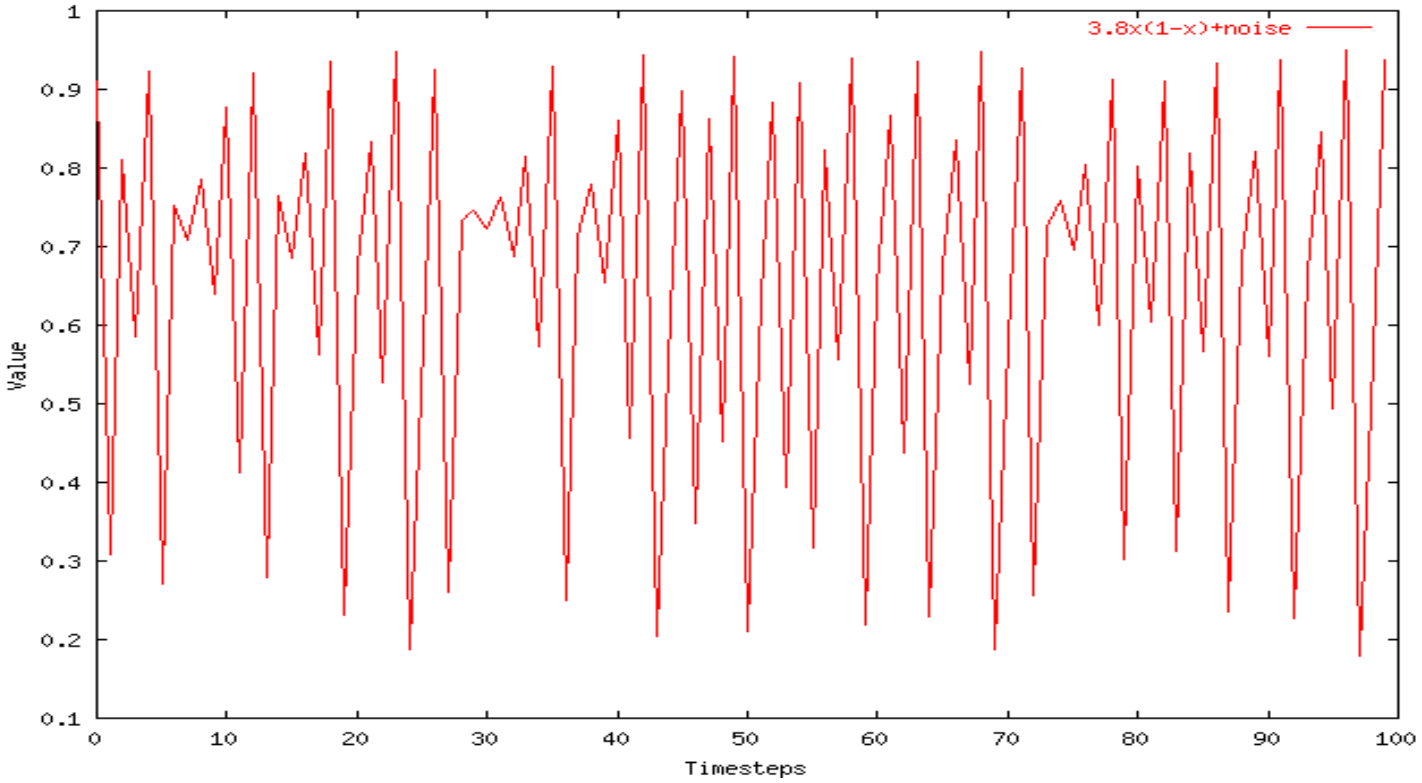
- given x_{t-1}, x_{t-2}, …, ('chaotic'/non-linear)
- Q: forecast x_t
- A: lag-plots + sim. search (= 'Delayed Coordinate Embedding')





Recall: Problem #1

Value



Time

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

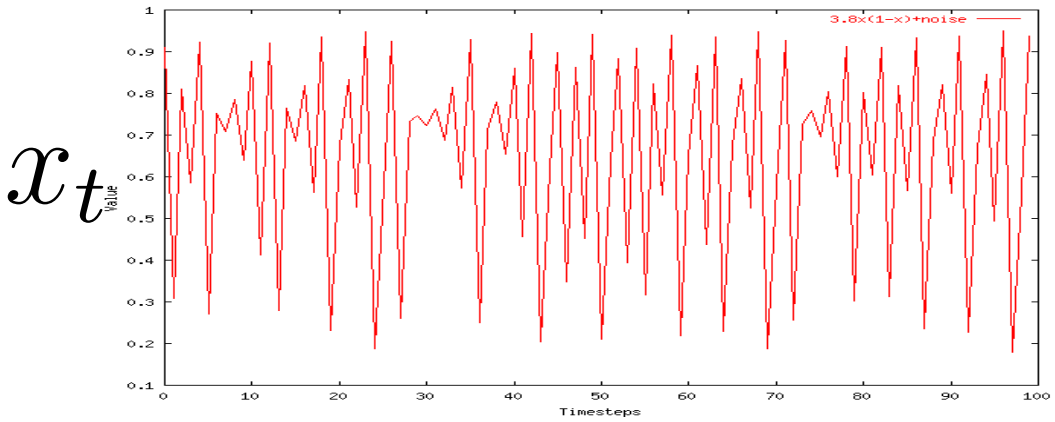
ARIMA pitfall

Example: logistic parabola

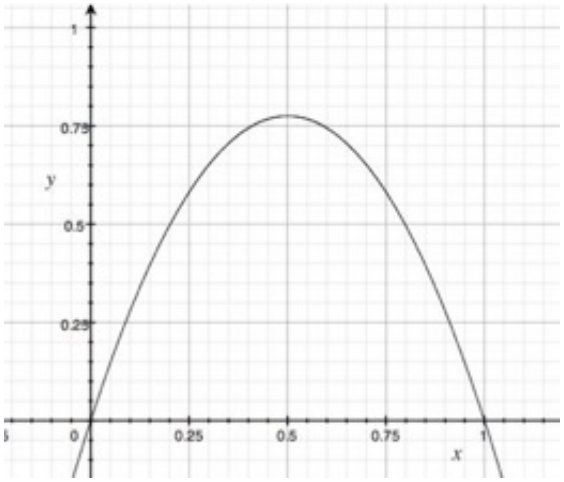
Models population of flies [R. May/1976]

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

Logistic map



Time t



ARIMA pitfall

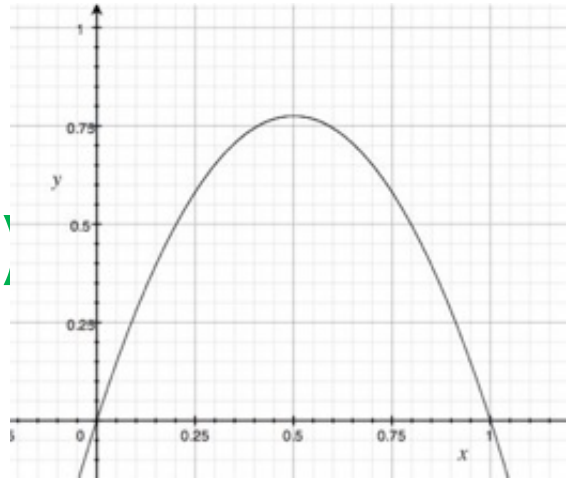
Example: logistic parabola

Models population of flies [R. May/1976]

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

Logistic map

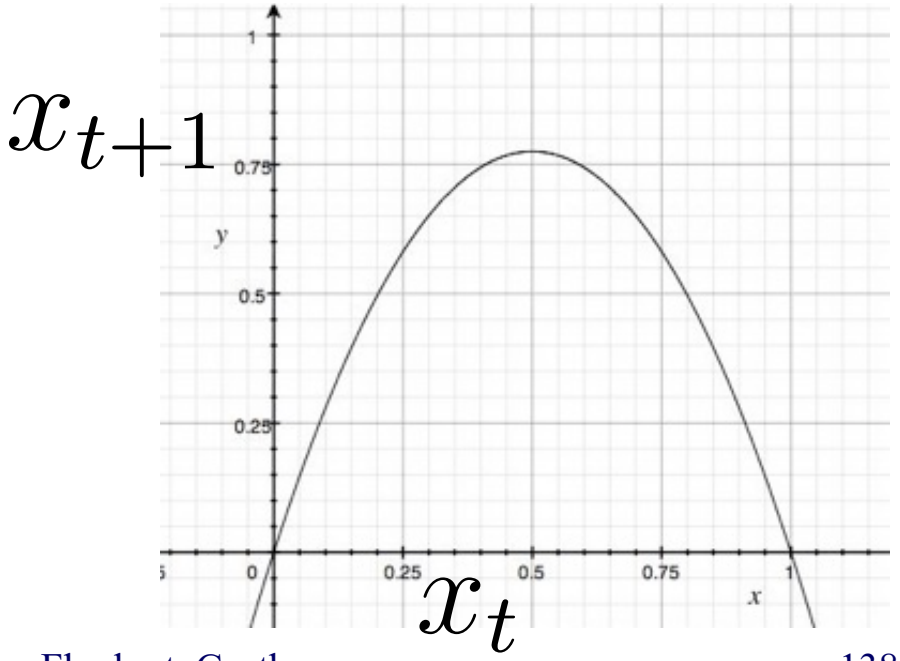
- = SI virus prop. model
- ~ Bass equation (market penetration)
- Special case of Lotka-Volterra



x_t

ARIMA pitfall

Linear equations, e.g., AR, ARIMA, …

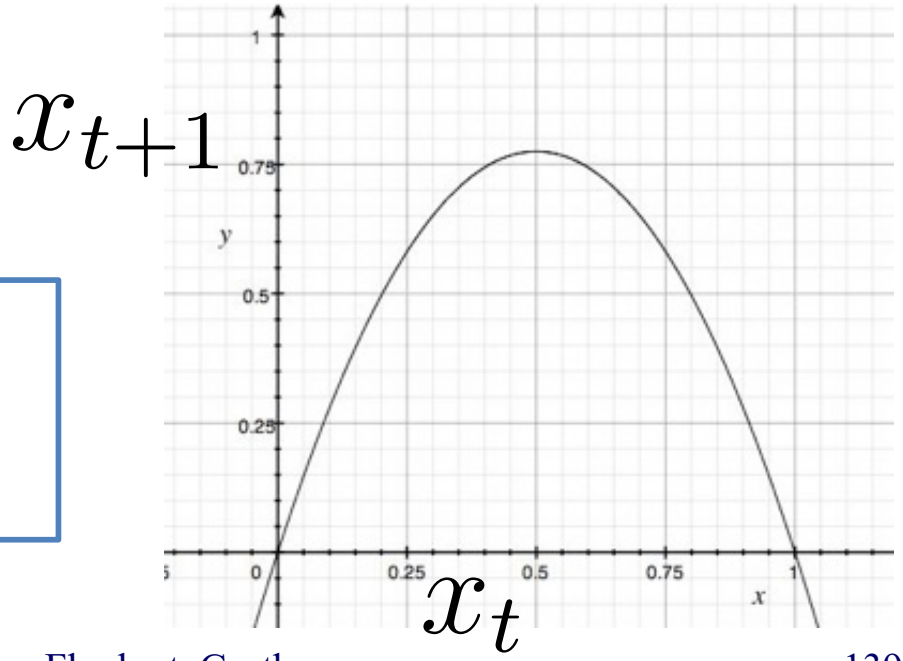


ARIMA pitfall

Linear equations, e.g., AR, ARIMA, …

e.g., AR(1)

$$x_{t+1} = ax_t + \epsilon$$

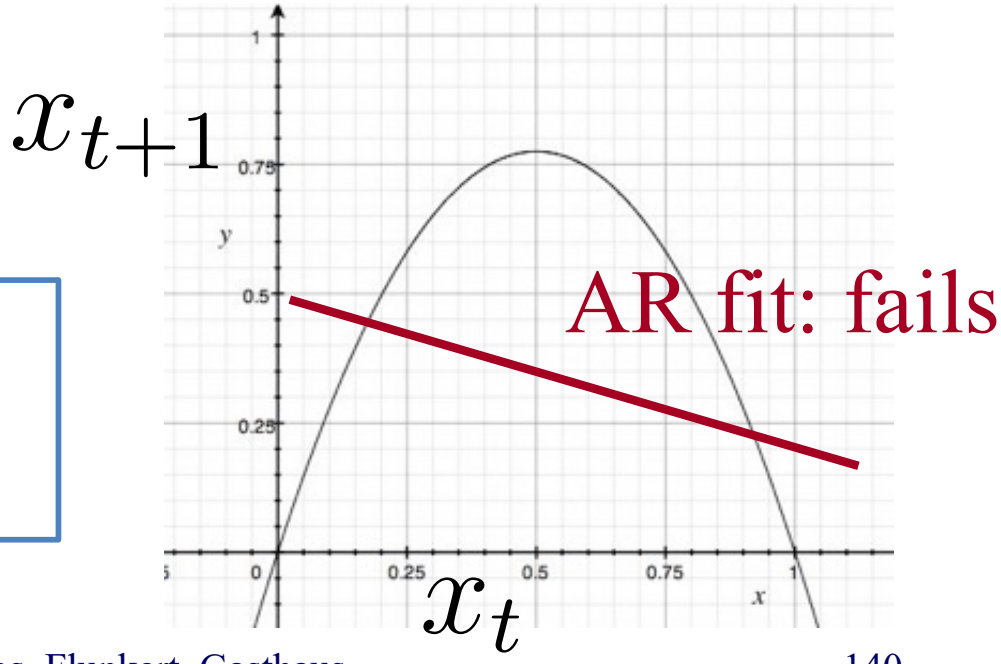


ARIMA pitfall

Linear equations, e.g., AR, ARIMA, …

e.g., AR(1)

$$x_{t+1} = ax_t + \epsilon$$



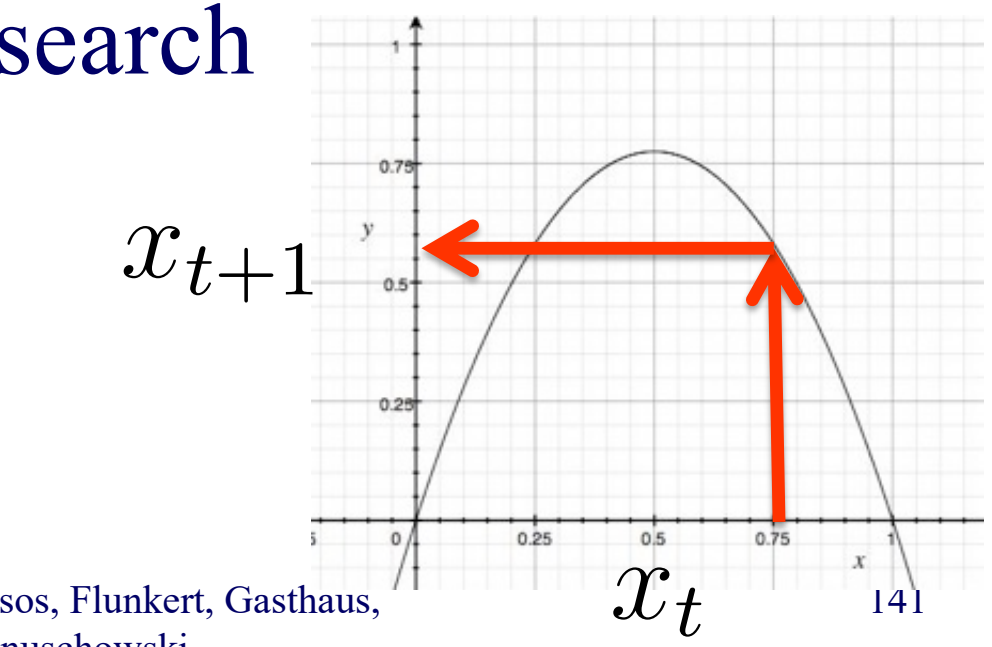
Solution?

"Delayed Coordinate Embedding"

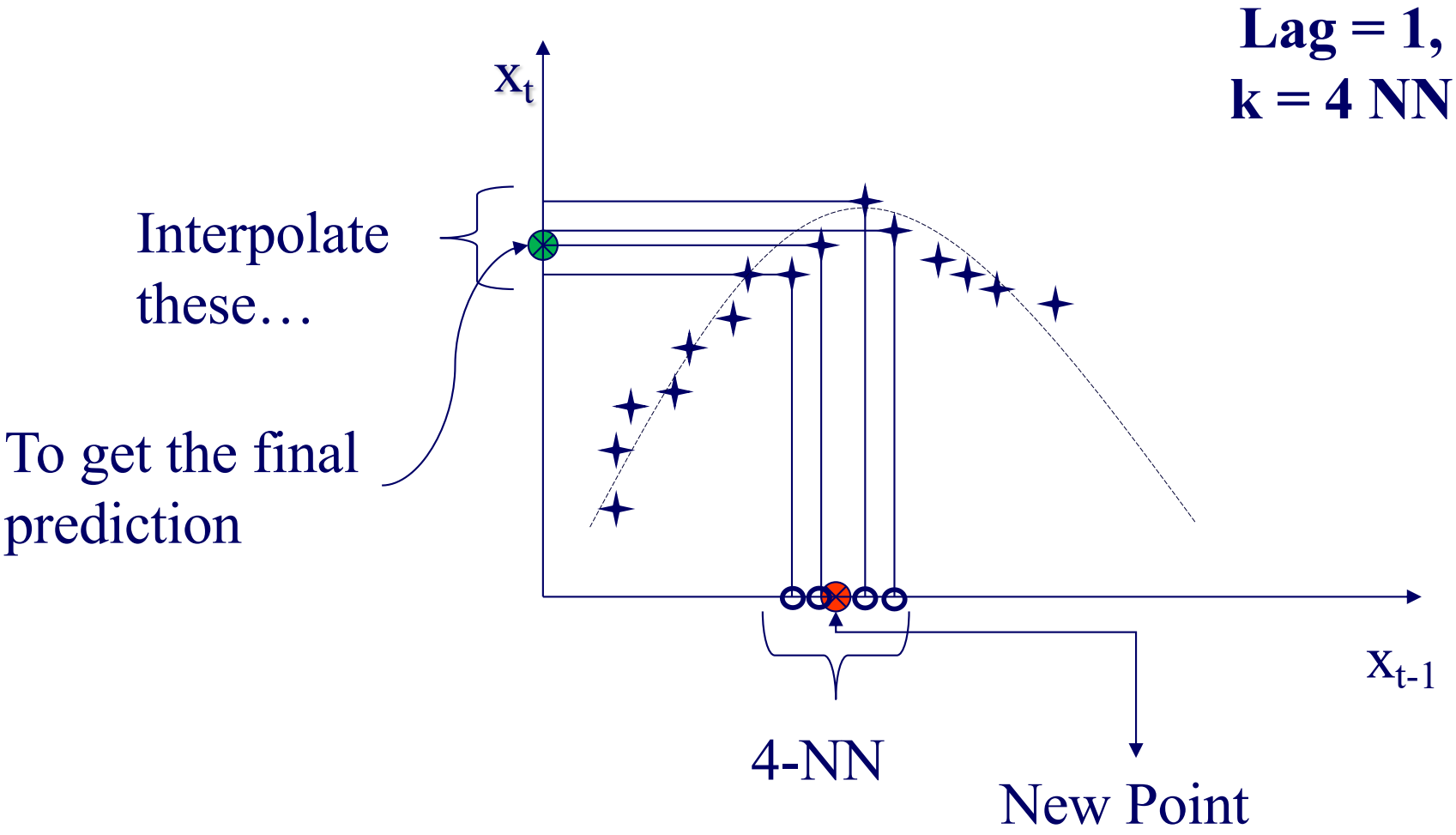
= Lag Plots

[Sauer94]

k-nearest neighbor search

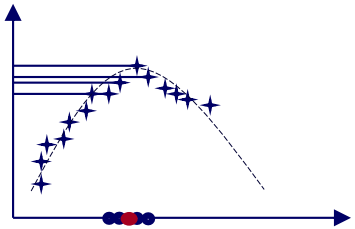


General Intuition (Lag Plot)



Q: How to interpolate?

How do we interpolate between the k nearest neighbors?

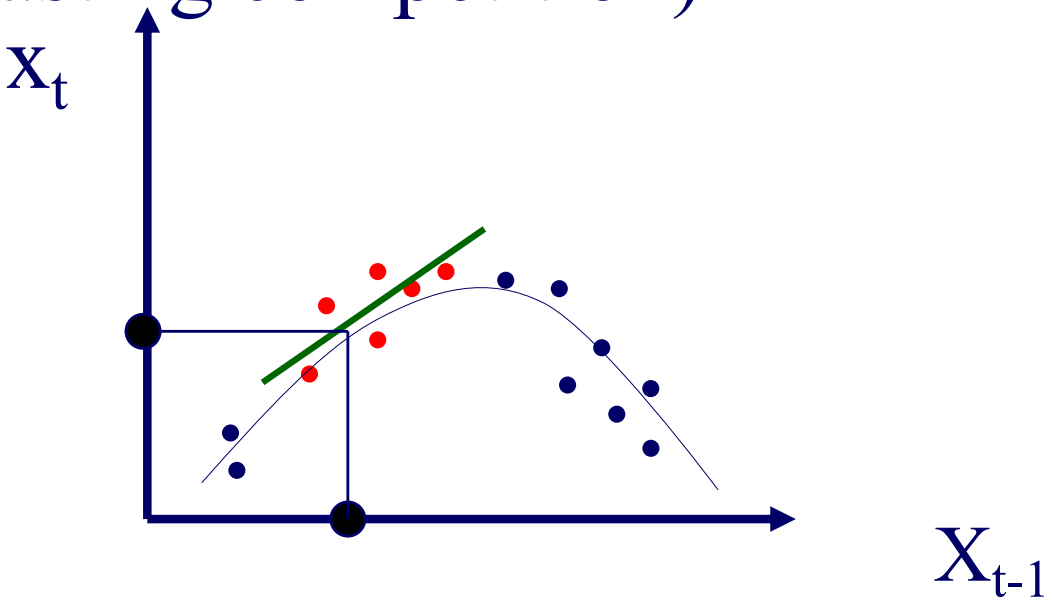


A1: Average

A2: Weighted average (weights drop with distance - how?)

Q: How to interpolate?

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



Solution?

"Delayed Coordinate Embedding"

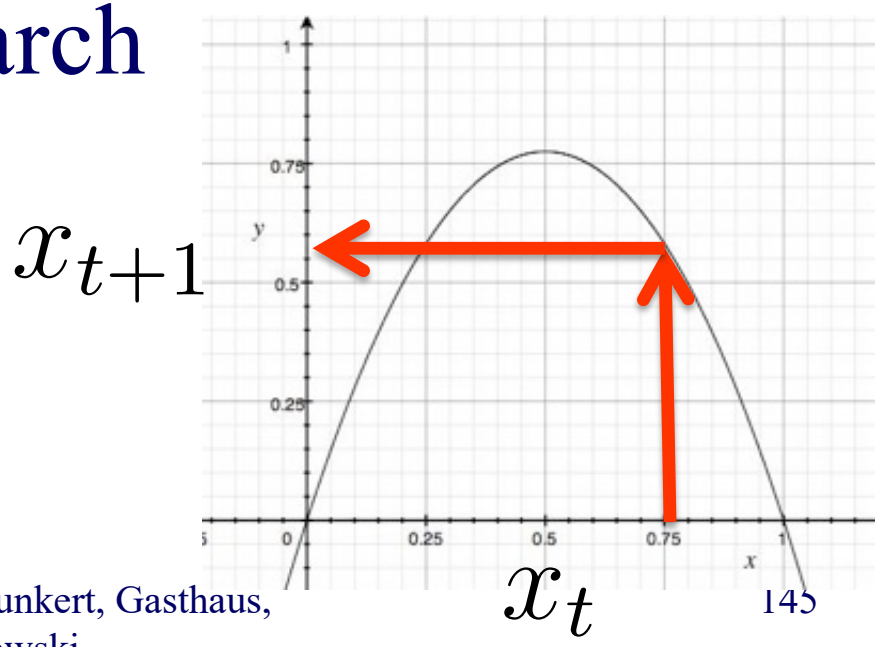
= Lag Plots

[Sauer94]

k-nearest neighbor search

k= ??

Lag = ??



Solution?

"Delayed Coordinate Embedding"

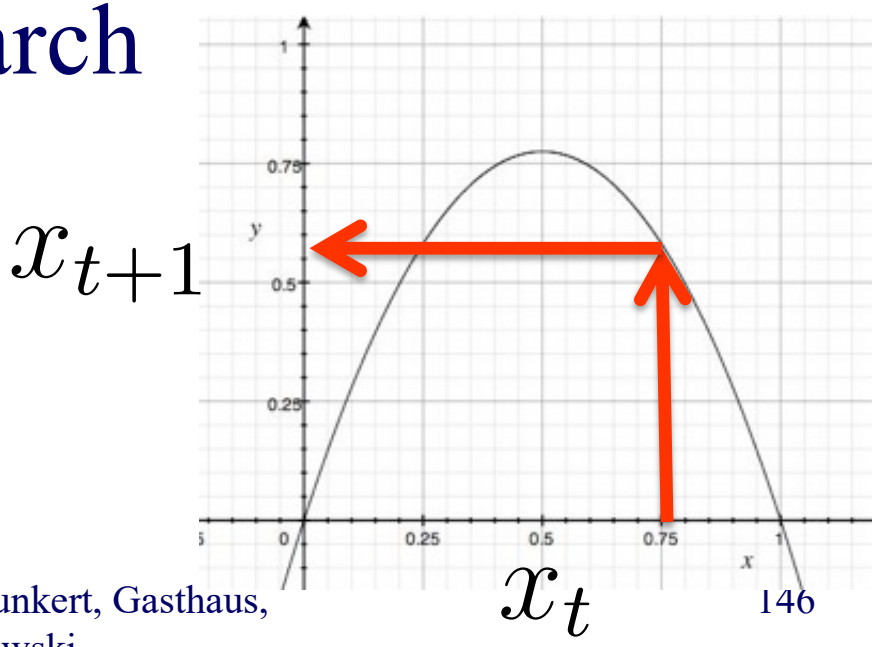
= Lag Plots

[Sauer94]

k-nearest neighbor search

k= ~5

Lag ~ 16



Carnegie Mellon

amazon

Q: Any theory behind it?

A: YES!

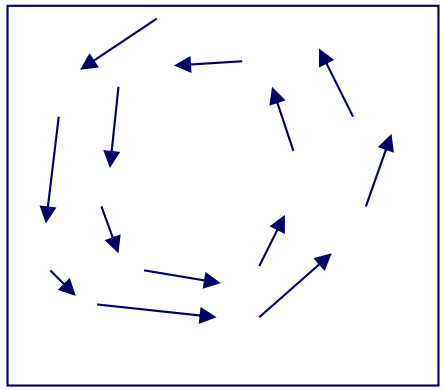
Theoretical foundation

- 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

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

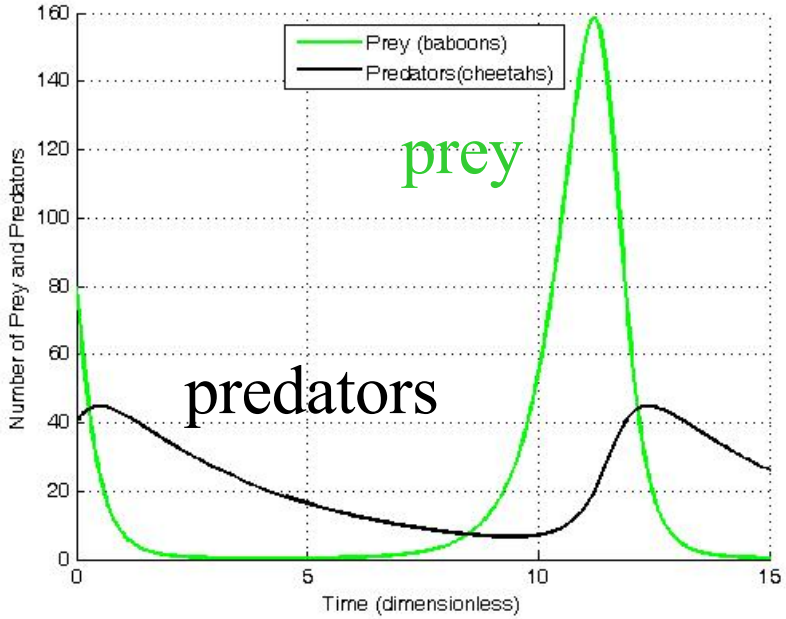


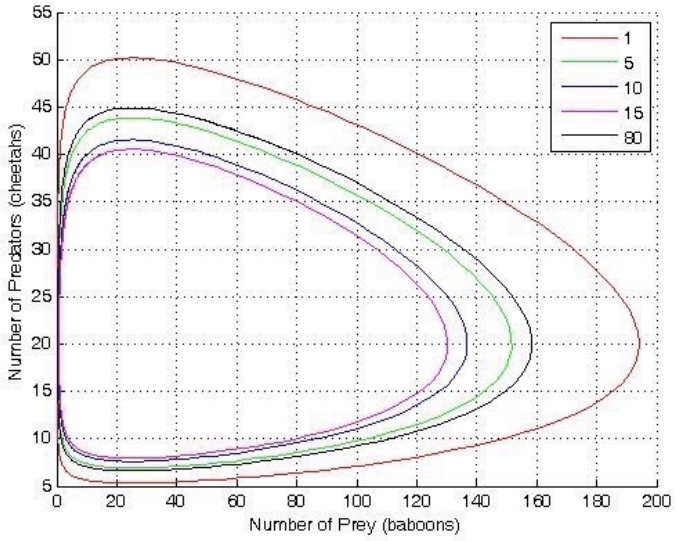
H is count of prey (e.g., hare)
P is count of predators (e.g., lynx)

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

Solution to Volterra-Lotka eq.

predators





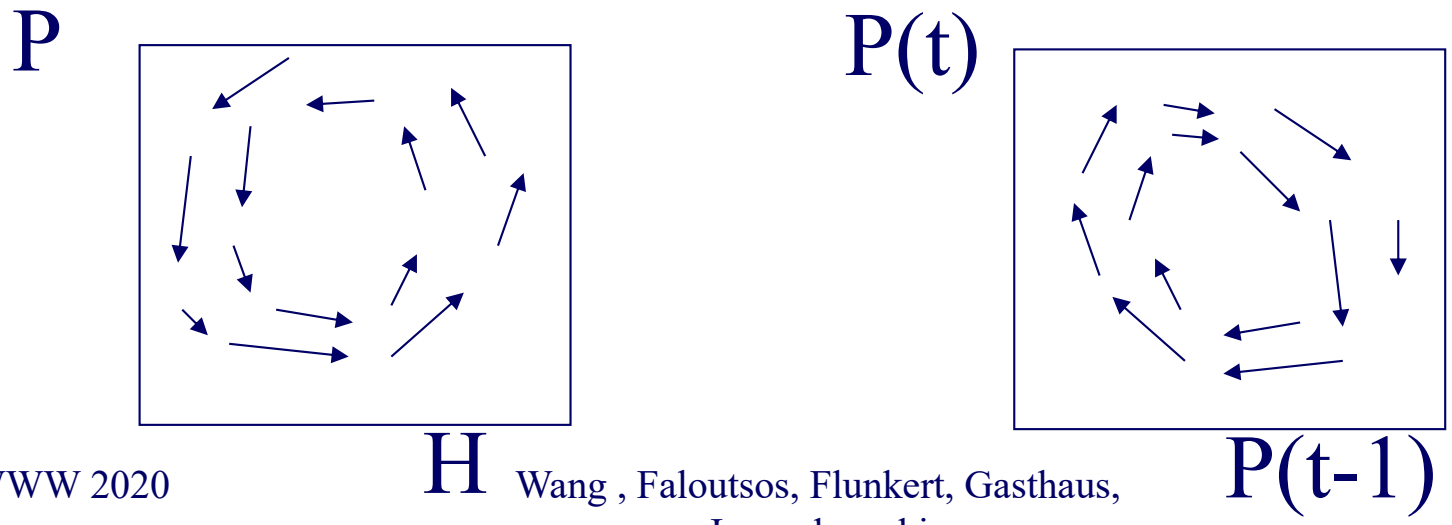
time

prey

from wikipedia

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**

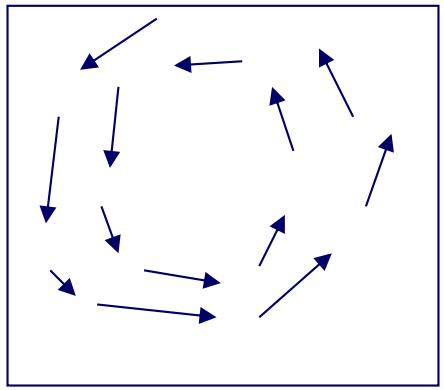


Theoretical foundation

Example: Lotka-Volterra equations

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

P



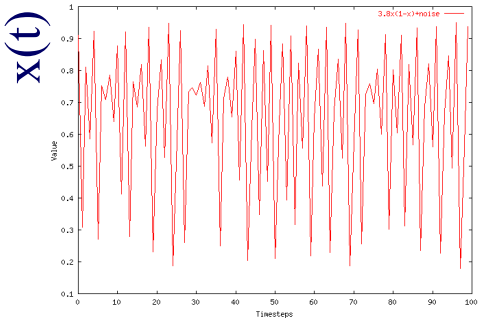
H

More on LV: later

Part 1 - Outline

- …
- P1.3. Linear forecasting
- P1.4. Non-linear forecasting
  - Problem
  - Idea
  - How-to
  - Experiments
  - (Lotka-Volterra equations)
  - Conclusions
- P1.5. Tensors

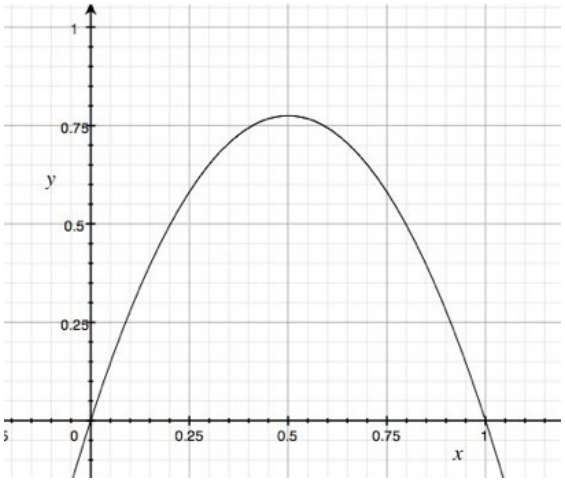
Datasets



Logistic Parabola:

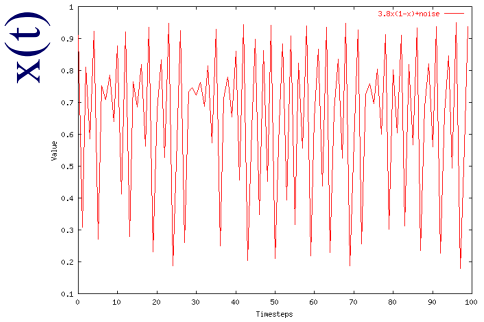
$x_t = ax_{t-1}(1-x_{t-1}) + noise$

Models population of flies [R. May/1976]



Lag-plot

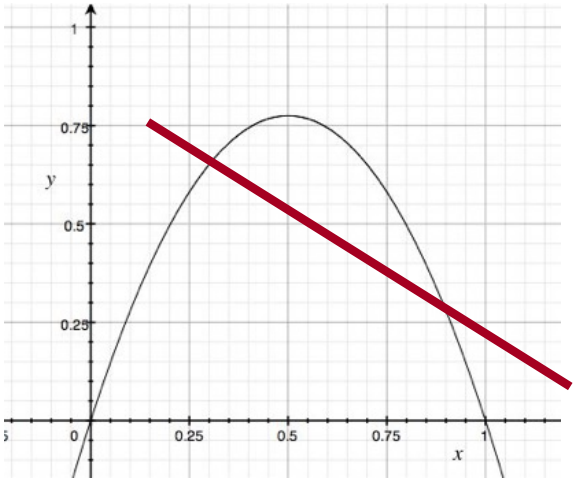
Datasets



Logistic Parabola:

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

Models population of flies [R. May/1976]



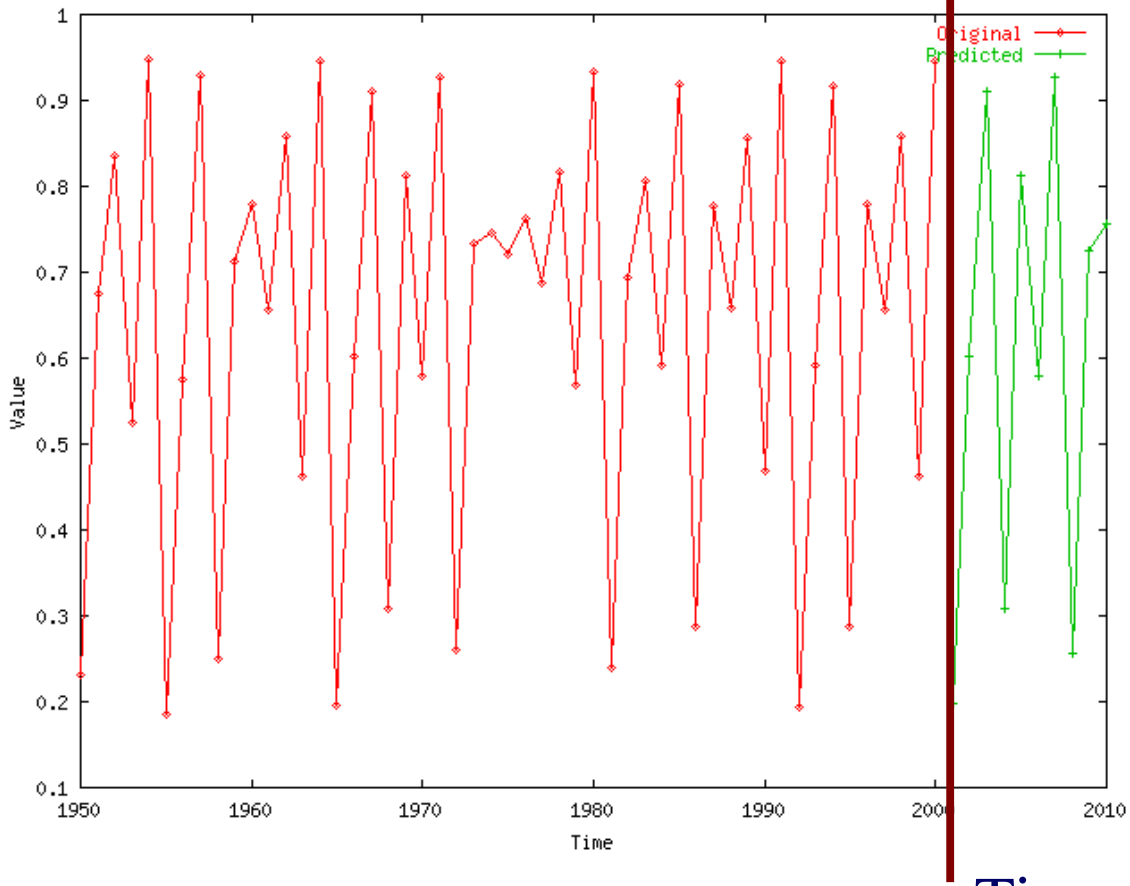
Lag-plot

ARIMA: fails

Logistic Parabola

Our Prediction from here

Value



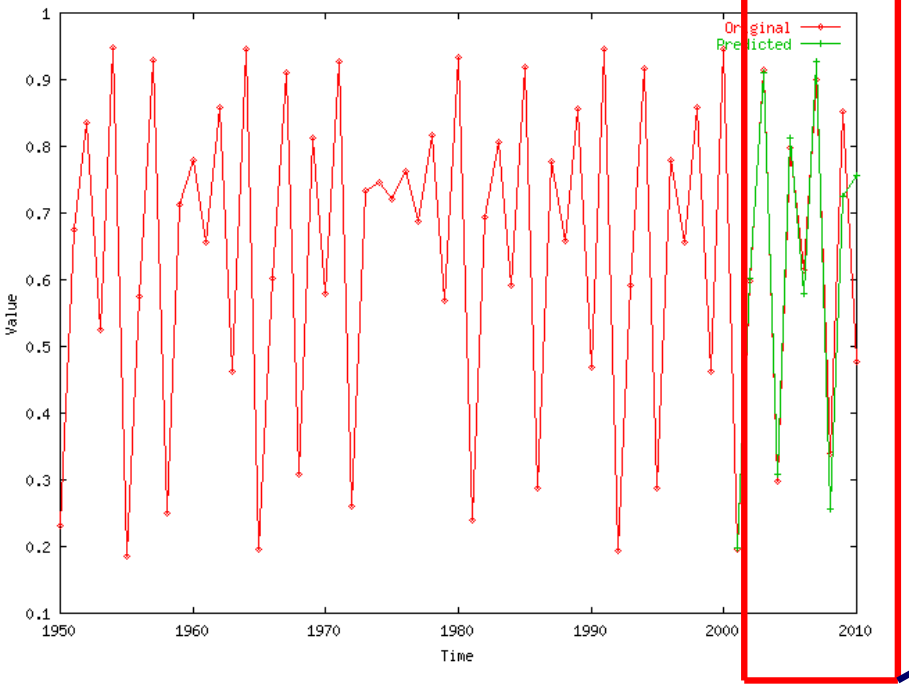
Timesteps

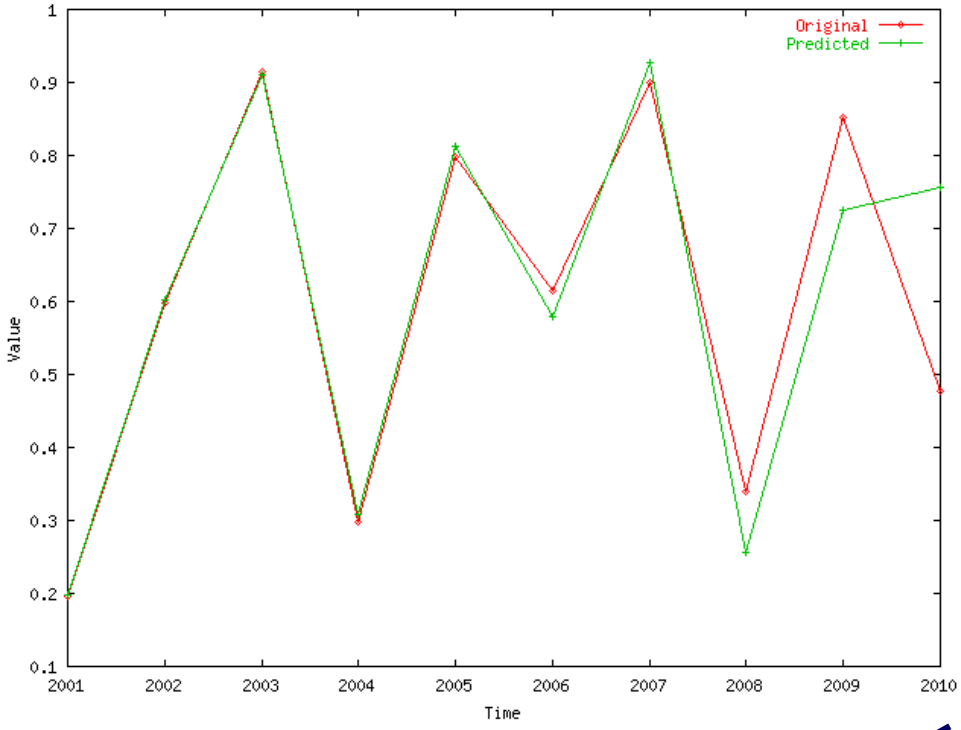
Carnegie Mellon

amazon

Logistic Parabola

Comparison of prediction to correct values





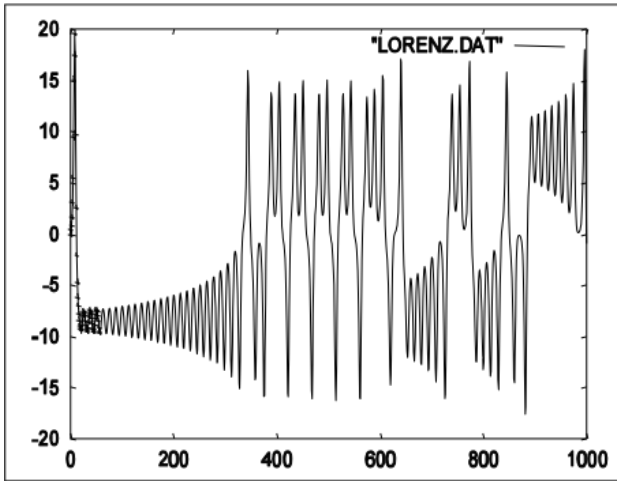
Datasets

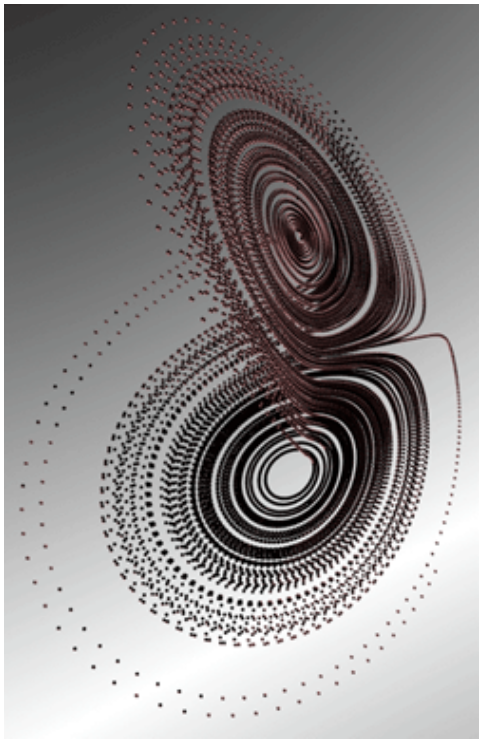
LORENZ: Models convection currents in the air

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

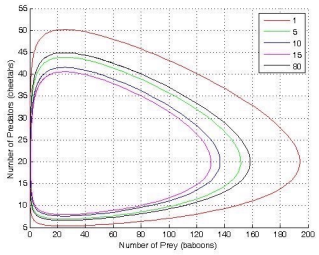
$dy / dt = x (b - z) - y$

$dz / dt = xy - c z$





Datasets



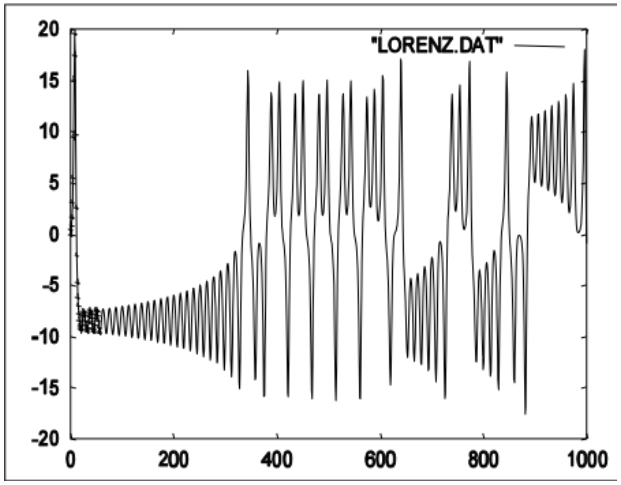
LORENZ: Models convection currents in the air

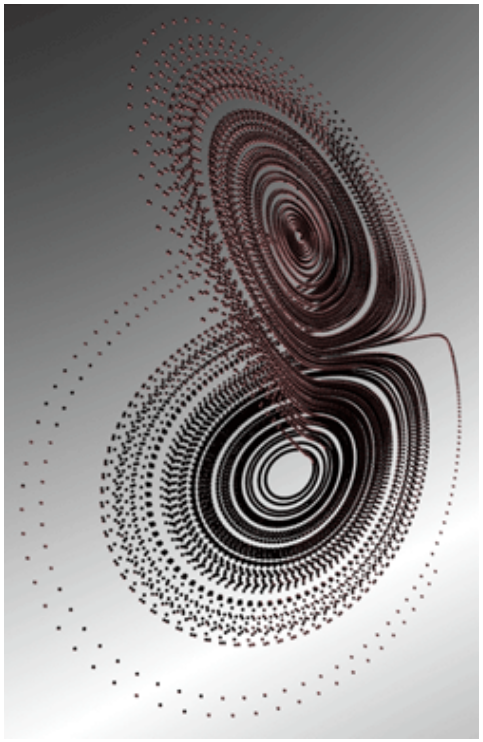
$dx / dt = a (y - x)$

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

$dz / dt = xy - c z$

'butterfly effect'



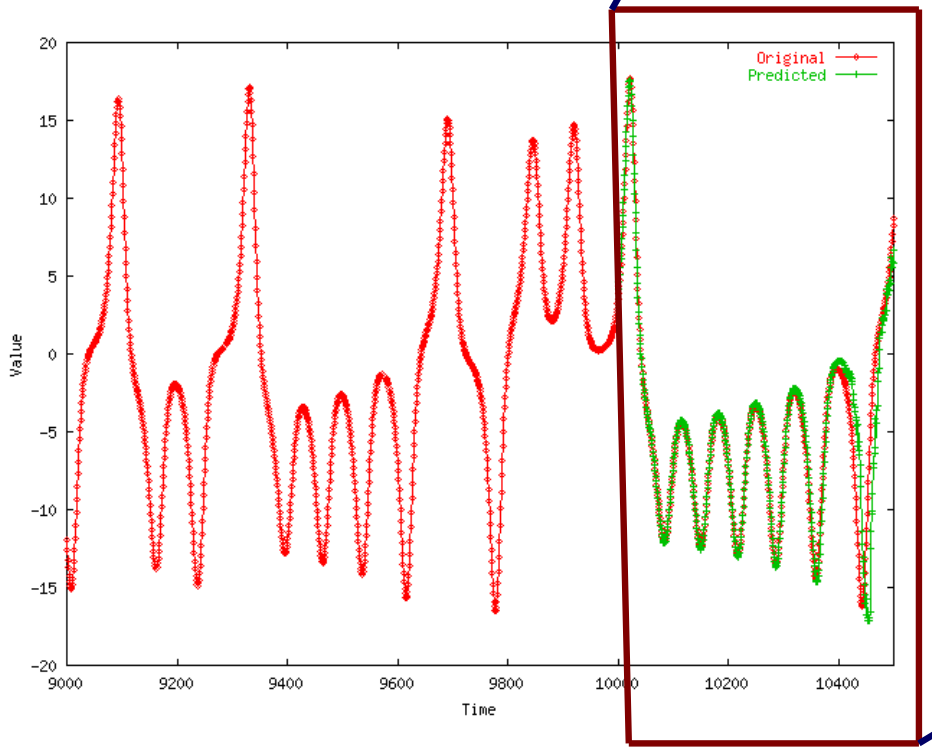


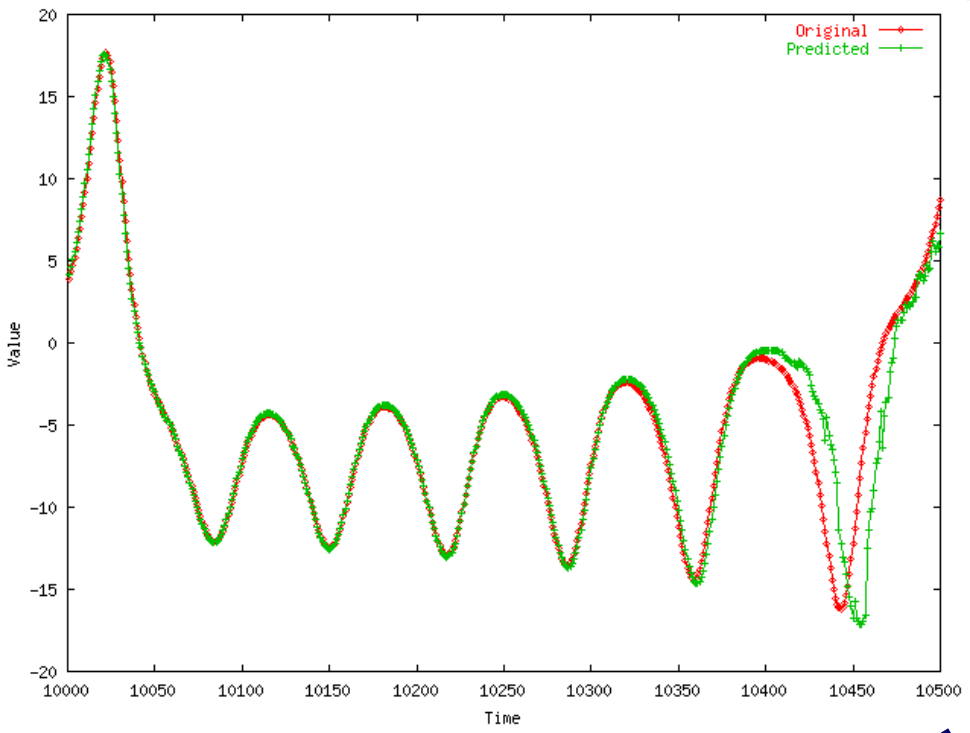
Carnegie Mellon

amazon

LORENZ

Comparison of prediction to correct values





Value

Timesteps

Part 1 - Outline

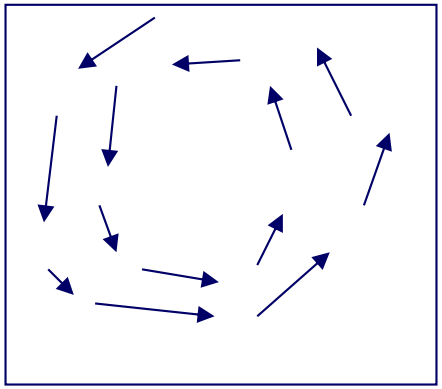
- …
- P1.3. Linear forecasting
- P1.4. Non-linear forecasting
  - Problem
  - Idea
  - How-to
  - Experiments
  - (Gray box modeling - Lotka-Volterra equations)
  - Conclusions
- P1.5. Tensors

Notice: LV are vital!

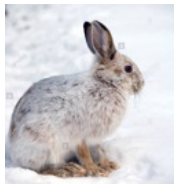
Example: Lotka-Volterra equations

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

P



H



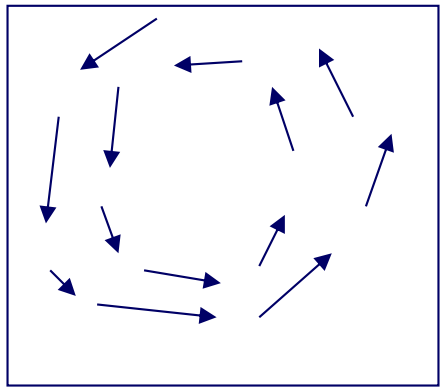
- Prey-predator
- Competing animals (rabbits/goats)
- " products (stocks/bonds)
- Self-competition (Bass model)
- Virus propagation (SI model)
- …

Notice: LV are vital!

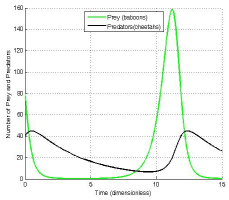
Example: Lotka-Volterra equations

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

P



H



Gray-box modeling:

- Given historical data,
- Assume LV
- Estimate highlighted coefficients
- and forecast

Carnegie Mellon

amazon

The Web as a Jungle: Non-Linear Dynamical Systems for Co-evolving Online Activities

Yasuko Matsubara (Osaka University)

Yasushi Sakurai (Osaka University)

Christos Faloutsos (CMU)

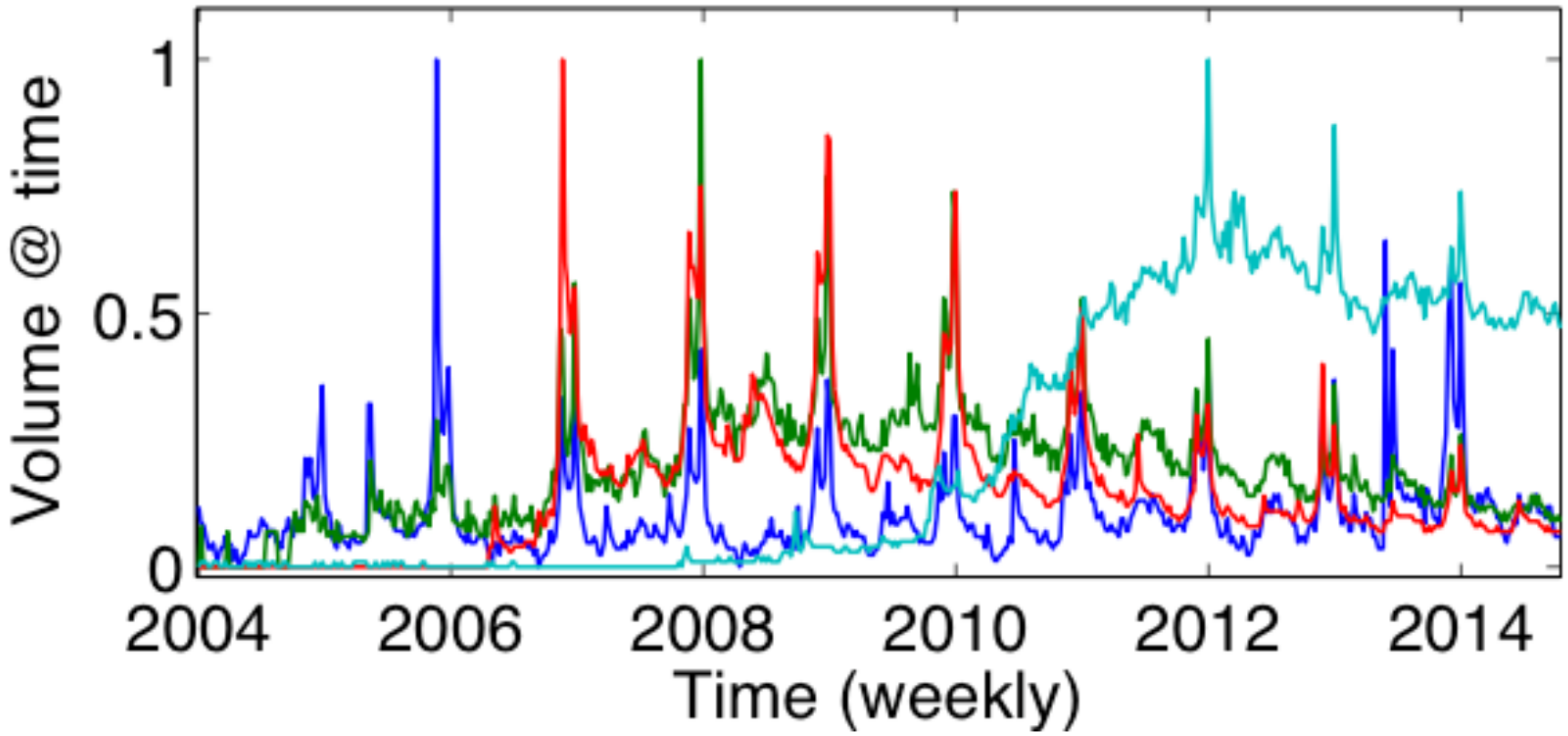
WWW 2015

Open-source code: here

Carnegie Mellon

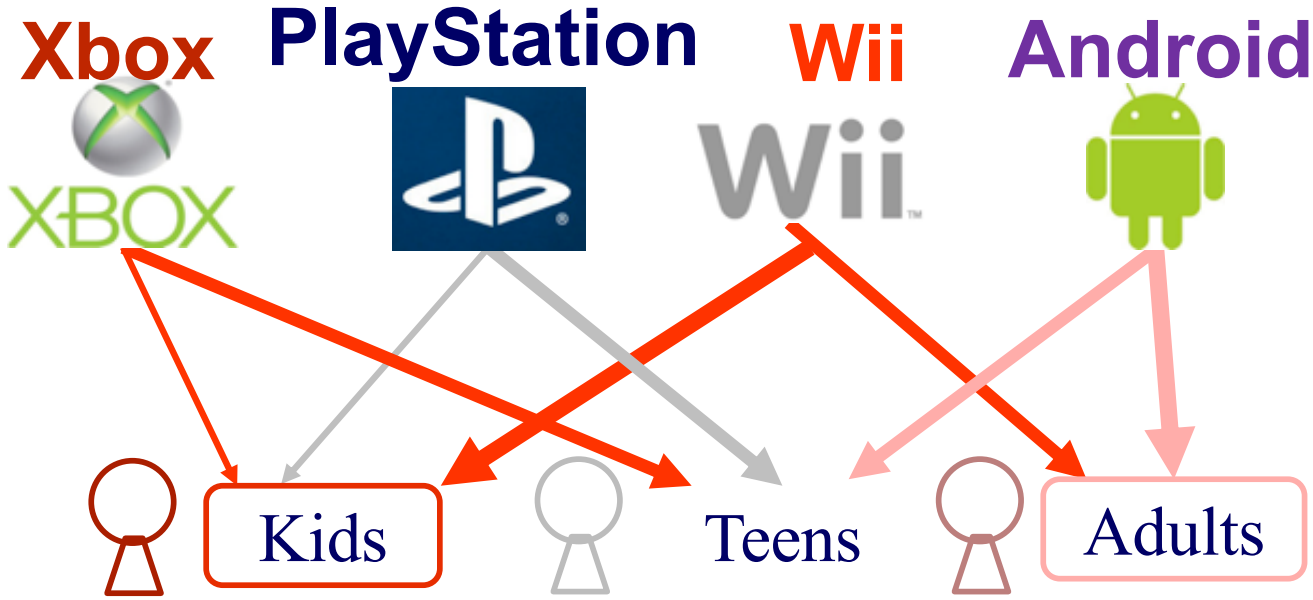
Given: online user activities

Xbox, PlayStation, Wii, Android

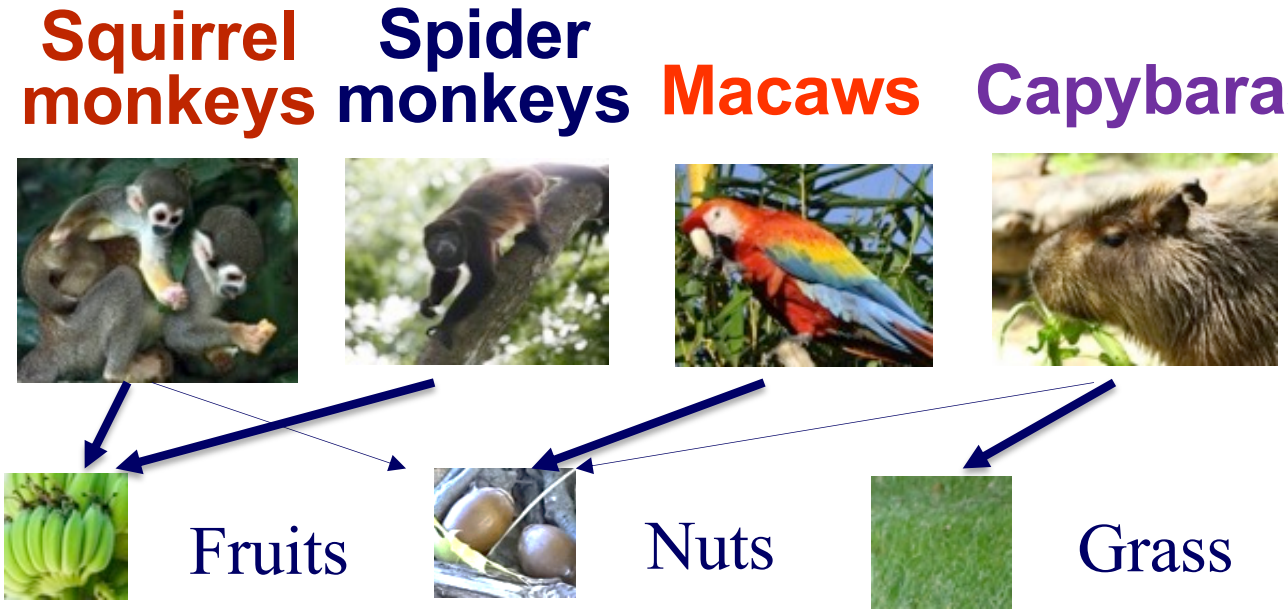


Carnegie Mellon

The Web as a jungle

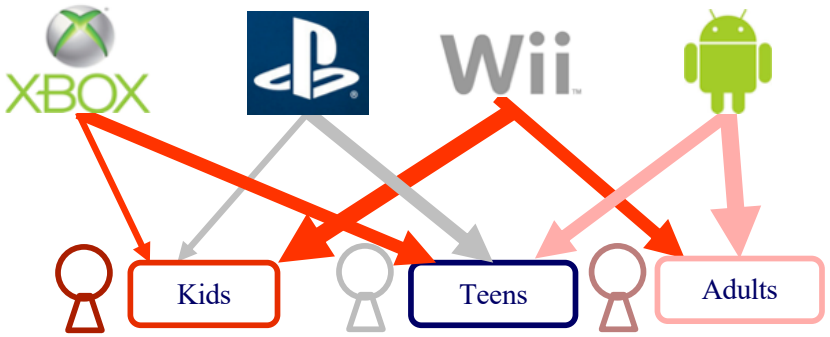


The Web as a jungle



Ecosystem in the **Jungle**

Ecosystem on the **Web**



Carnegie Mellon

LV equations

Interaction between multiple ('*d*') species/products/viruses


$$P_i(t+1) = P_i(t)\left[1 + r_i\left(1 - \frac{\sum_{j=1}^{d} a_{ij} P_j(t)}{K_i}\right)\right],$$
$$(i = 1, \cdots, d),$$

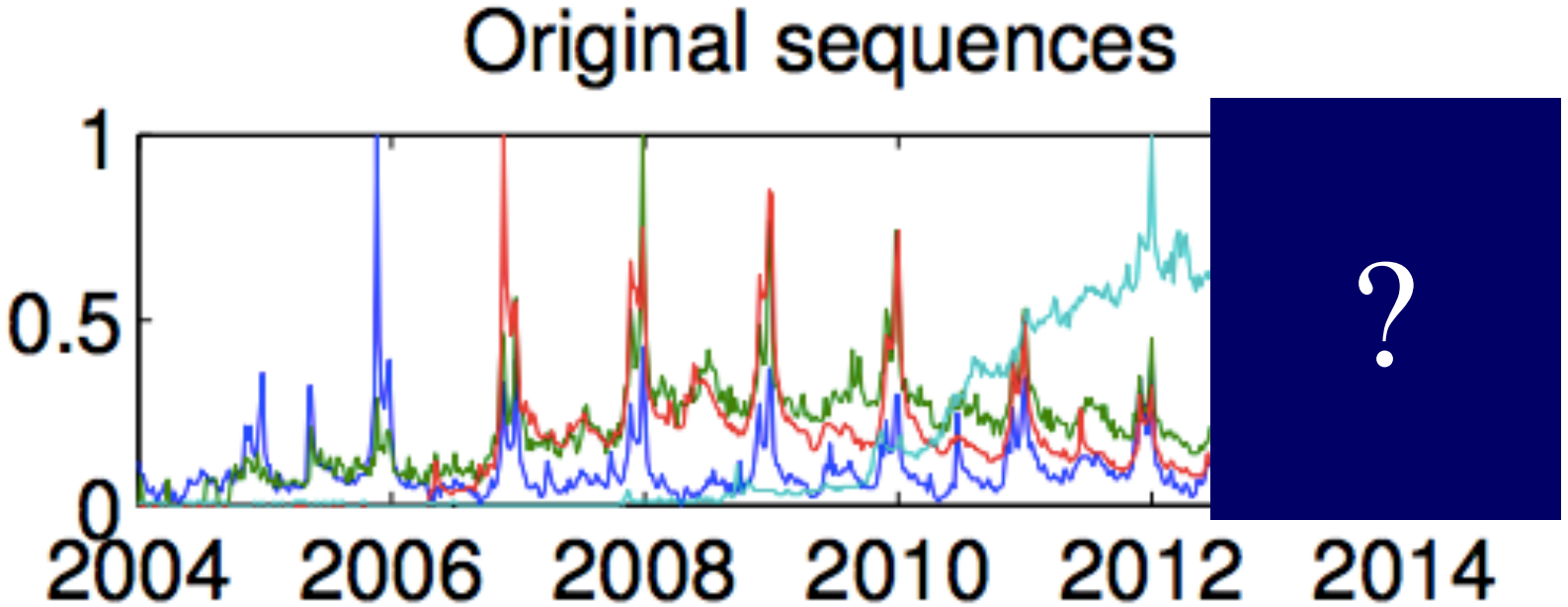
$a_{i,j}$: effect (harm) of species j on species i (positive: hurts)

$a_{i,i}$: always positive

$a_{i,j}, a_{j,i}$: if both negative: symbiosis (eg., sharks/pilot-fish)

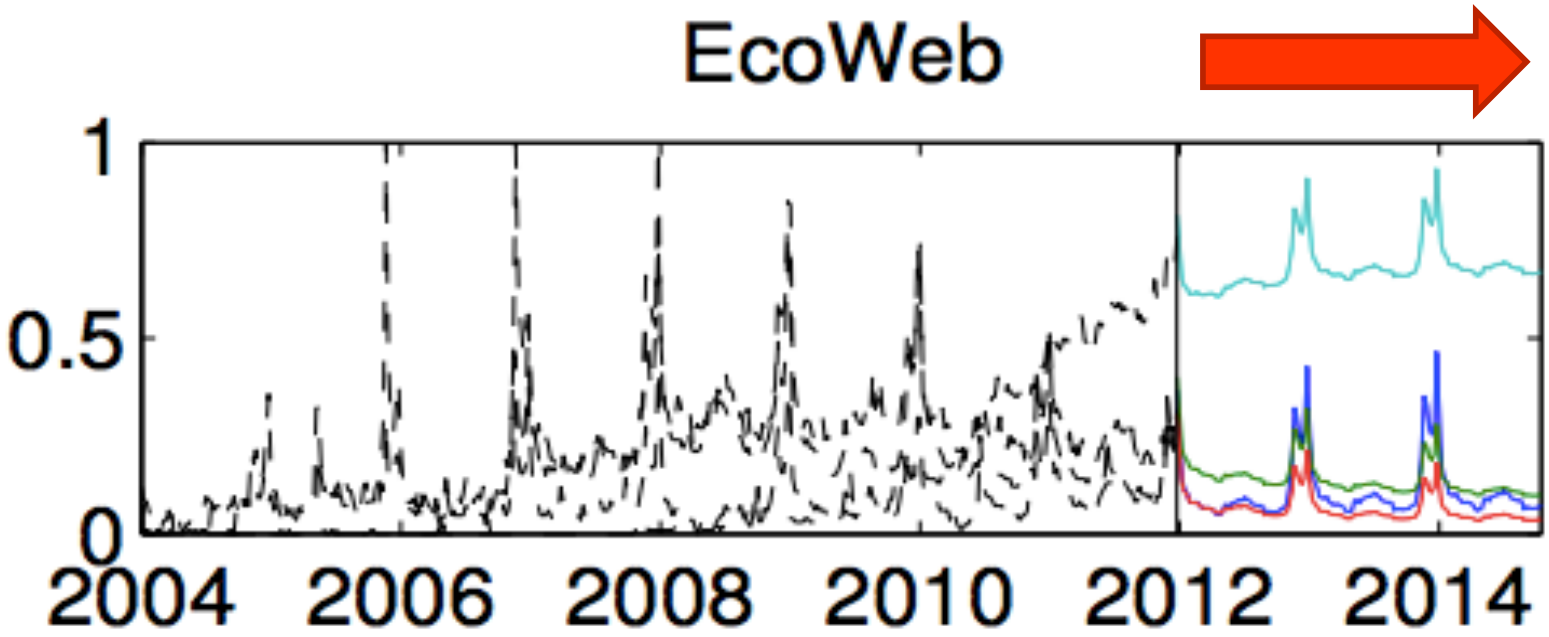
EcoWeb at work - forecasting

Train:	Forecast:
2/3 sequences	1/3 following years



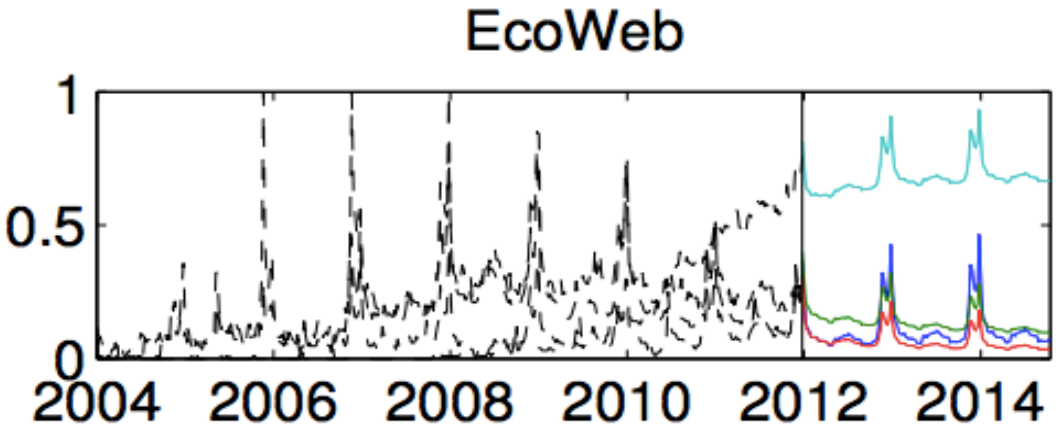
EcoWeb at work - forecasting

Train:	Forecast:
2/3 sequences	1/3 following years



EcoWeb can capture future patterns

EcoWeb at work - forecasting



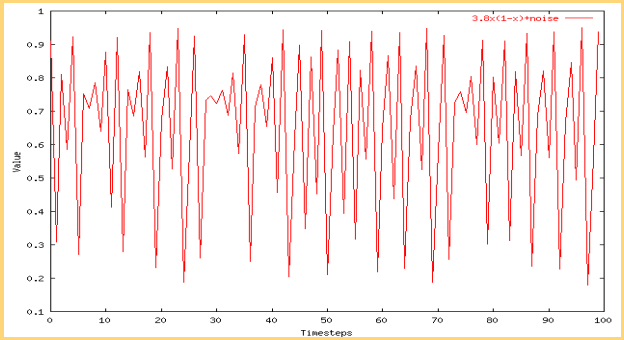
Again: Open-source code: here

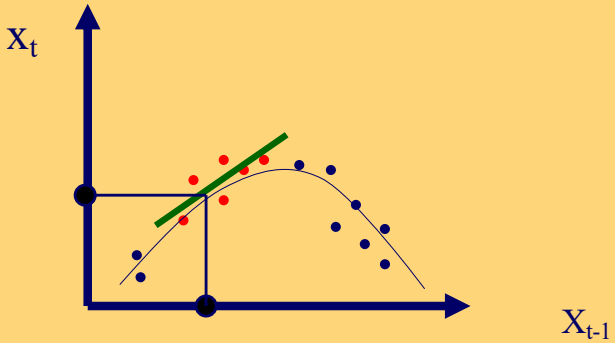
Part 1 - Outline

- …
- P1.3. Linear forecasting
- P1.4. Non-linear forecasting
  - Problem
  - Idea
  - How-to
  - Experiments
  - (Gray box modeling - Lotka-Volterra equations)
  - Conclusions
- P1.5. Tensors

Solution

- given x_{t-1}, x_{t-2}, …, ('chaotic'/non-linear)
- Q: forecast x_t
- A: lag-plots + sim. search (= 'Delayed Coordinate Embedding')





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 - Outline

- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP
- P1.3. Linear Forecasting
- P1.4. Non-linear forecasting
- P1.5. Tensors
- Conclusions

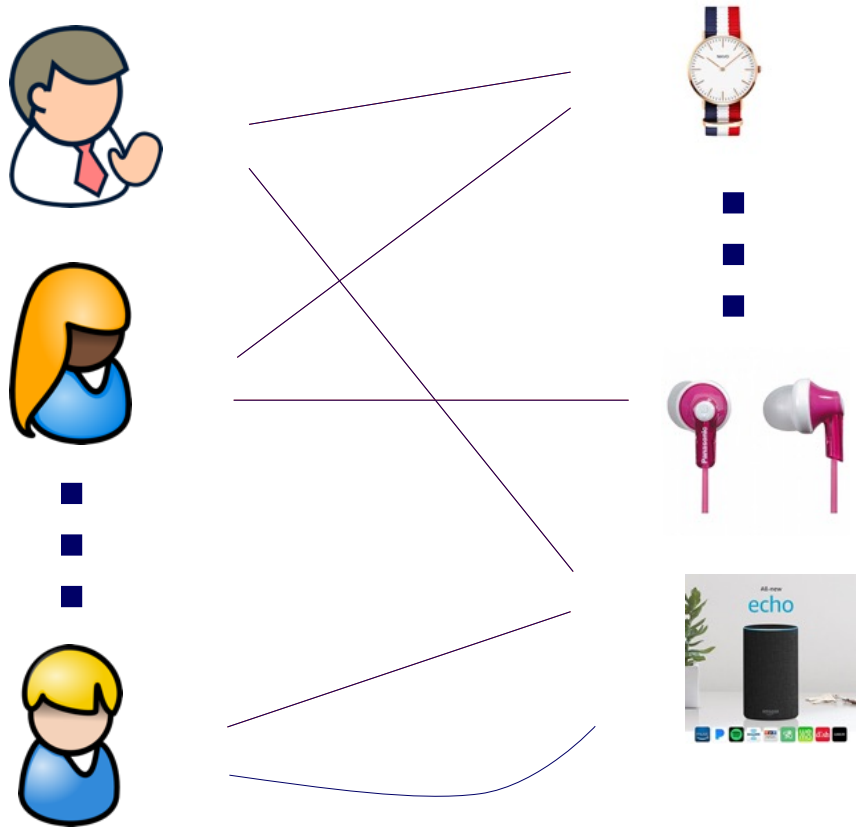
Carnegie Mellon

amazon

Part 1.5: Tensors – time evolving graphs

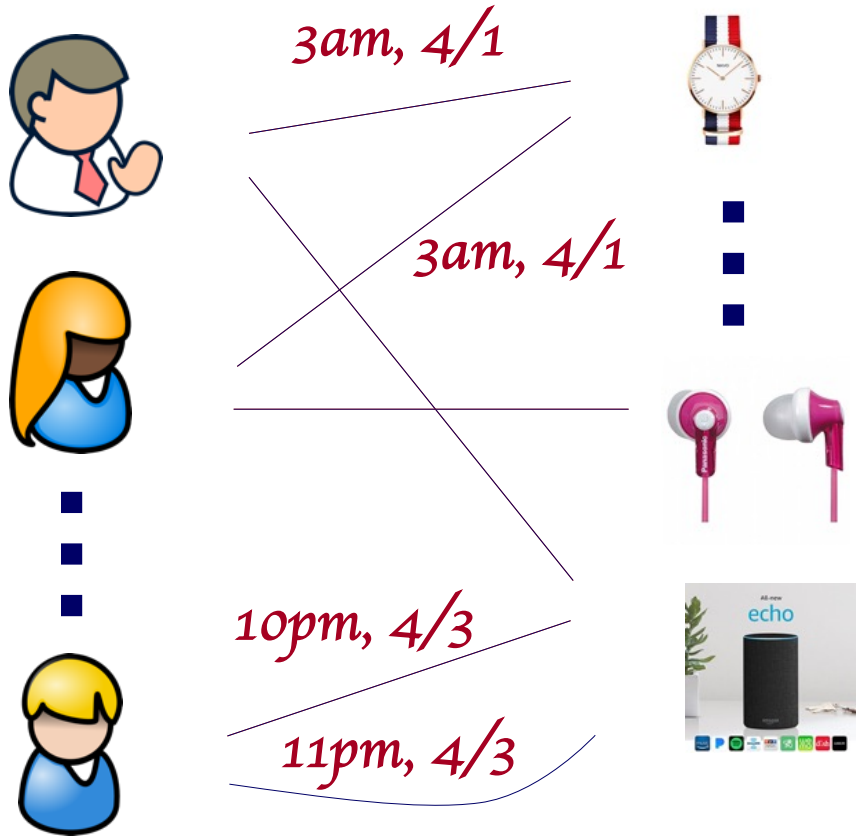
Binary relationships: graph

- Who – buys - what



Time evolving graphs - model?

- Who – buys – what - when



A: tensors

- … for time evolving graphs (and higher-order relationships)

Tensor examples

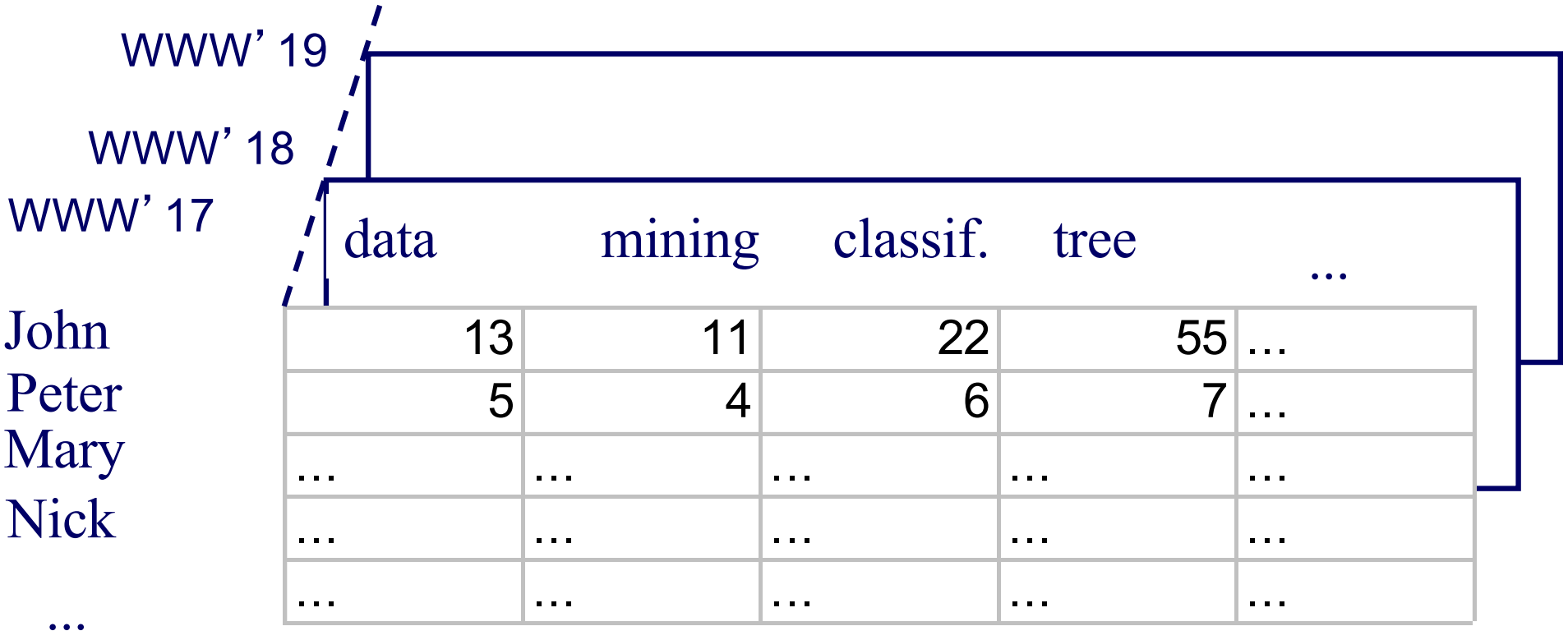
- A: N-D generalization of matrix:

WWW' 17

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

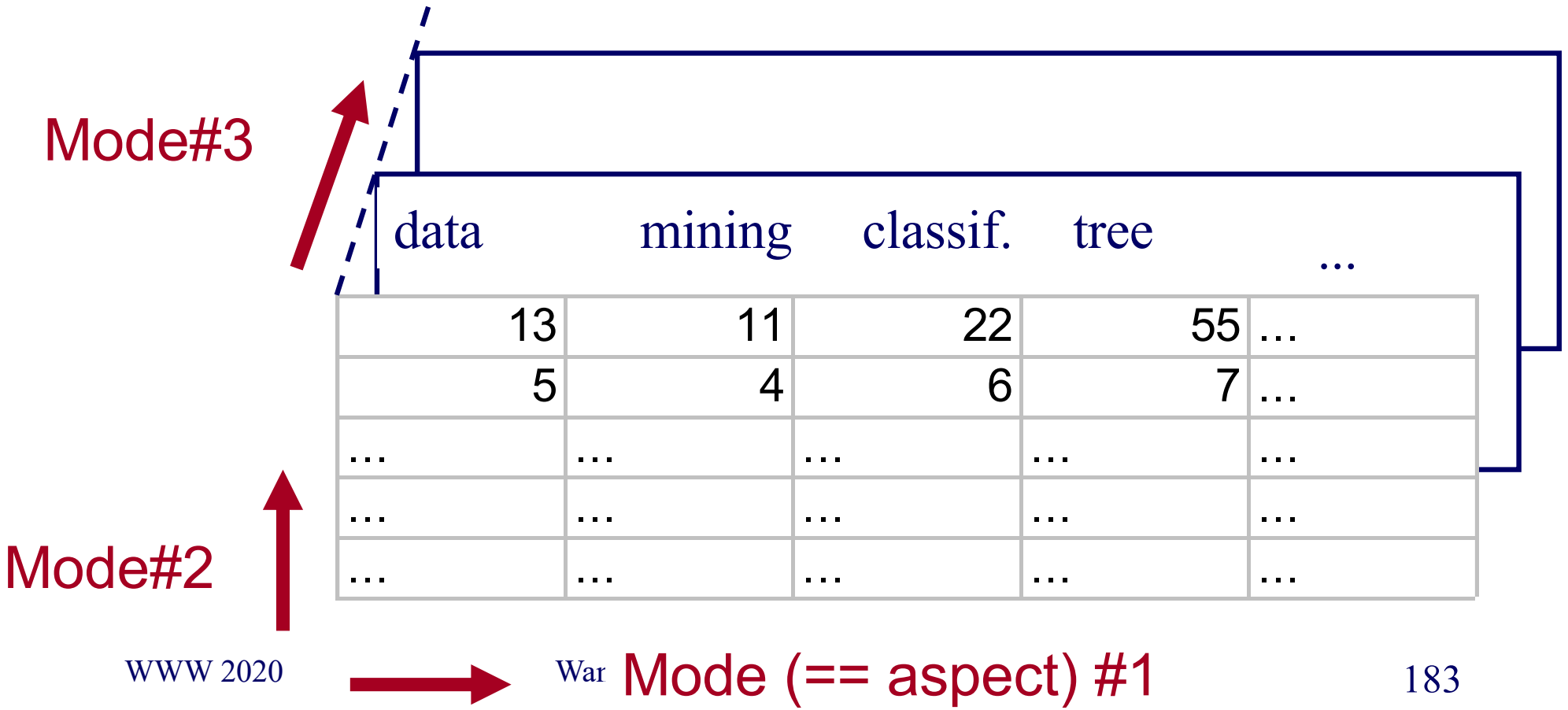
Tensor examples

- A: N-D generalization of matrix:



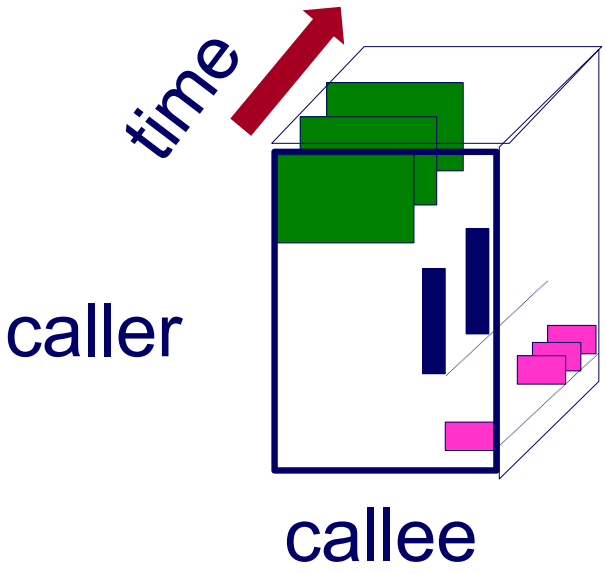
Tensors are useful for 3 or more modes

Terminology: 'mode' (or 'aspect'):



Problem: co-evolving graphs

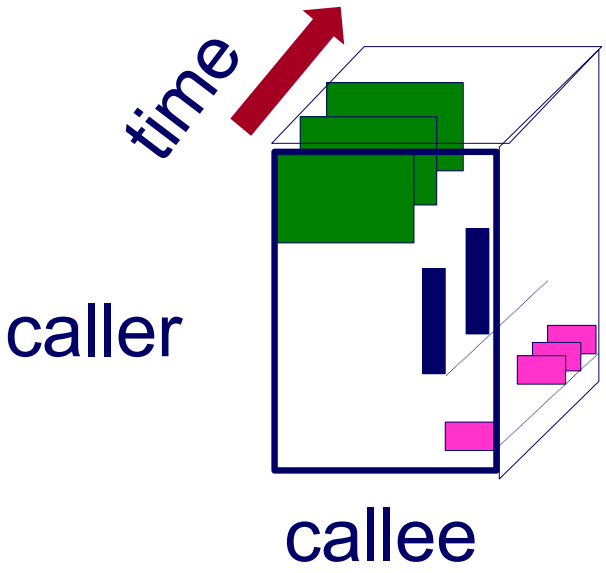
- How to forecast?
  - 4M x 4M x 15 days



4M x 4M AR models?

Problem: co-evolving graphs

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

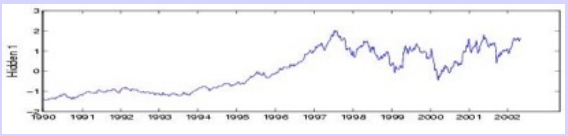


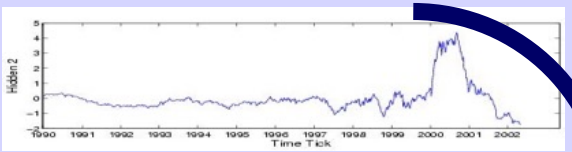
Carnegie Mellon

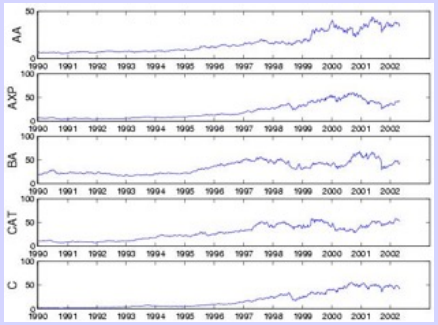
amazon

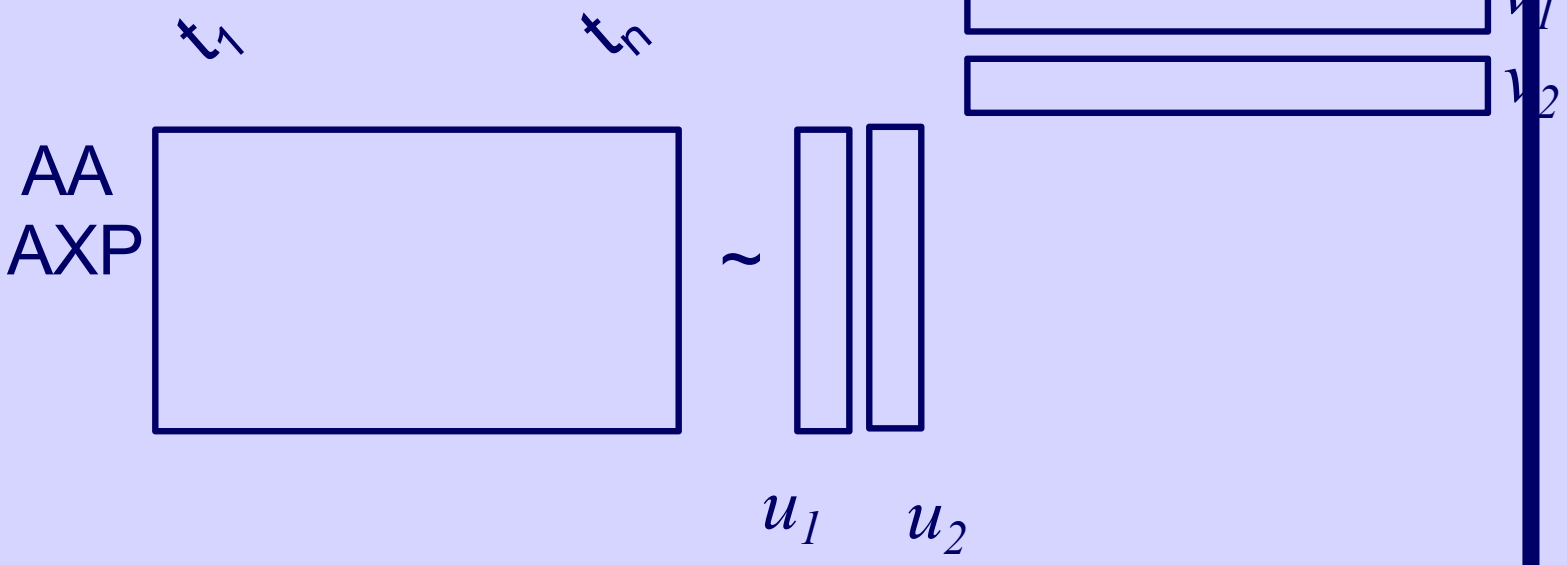
Tensor Basics: PARAFAC decomposition

SVD





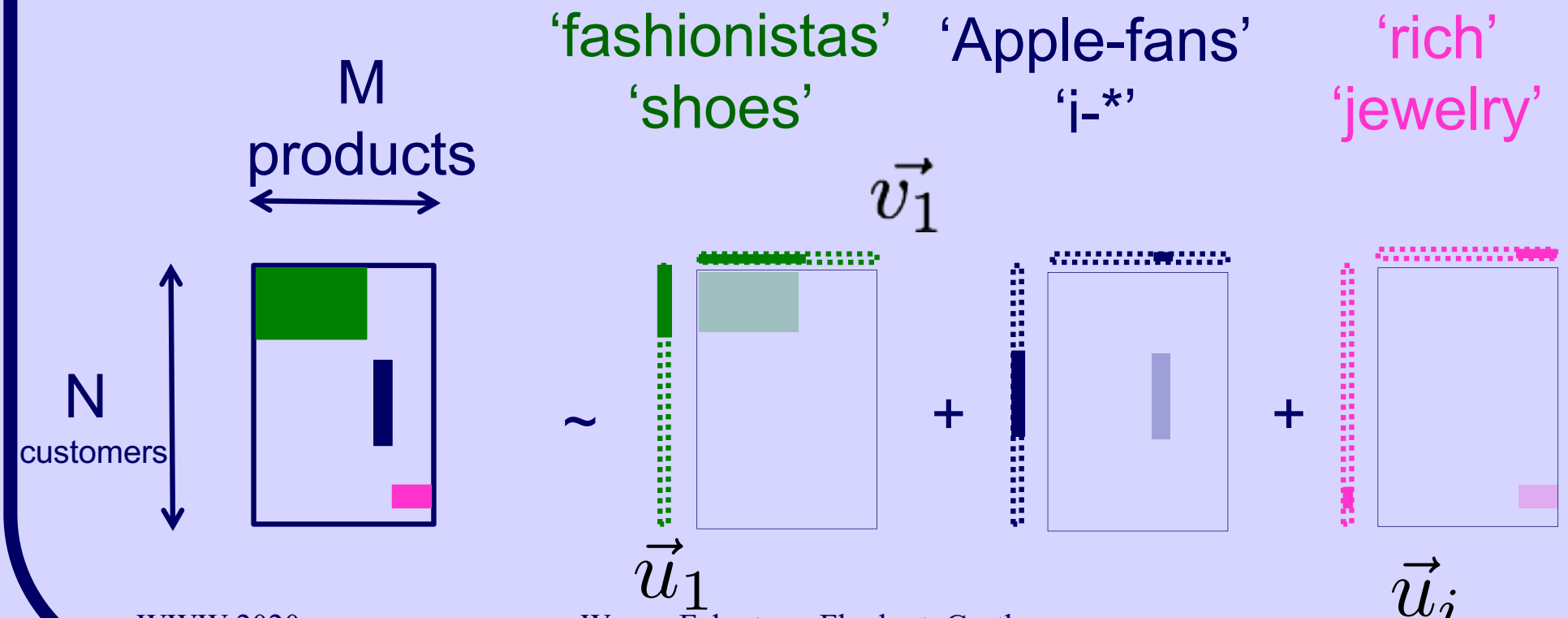





```
# svd code
import numpy as np
u, s, vh = np.linalg.svd(b)
```

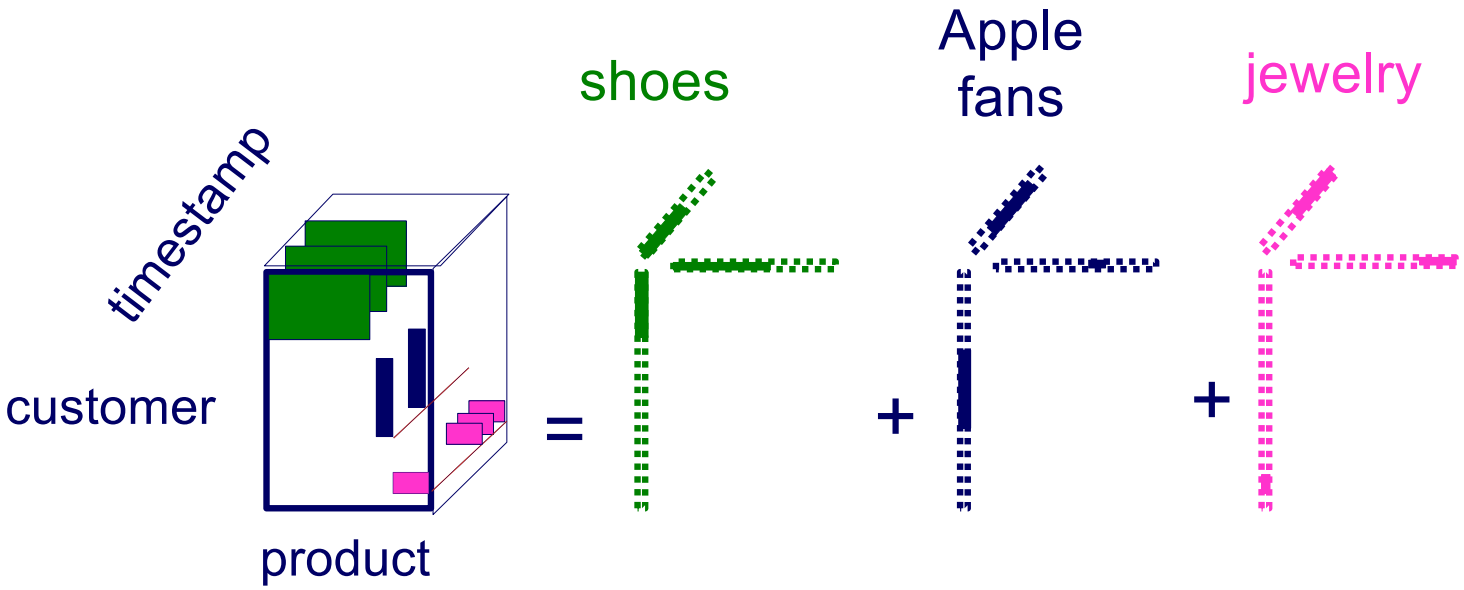
Tensor factorization

- (SVD) matrix factorization: finds blocks



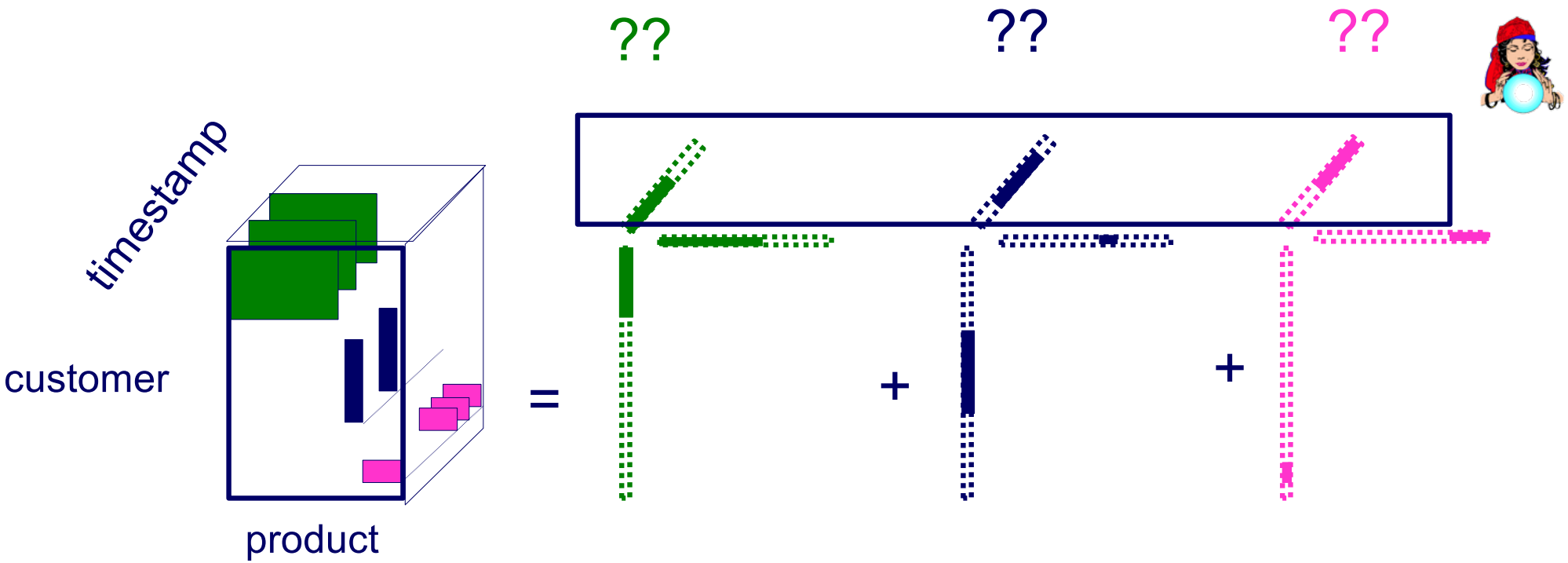
Tensor factorization

- PARAFAC decomposition



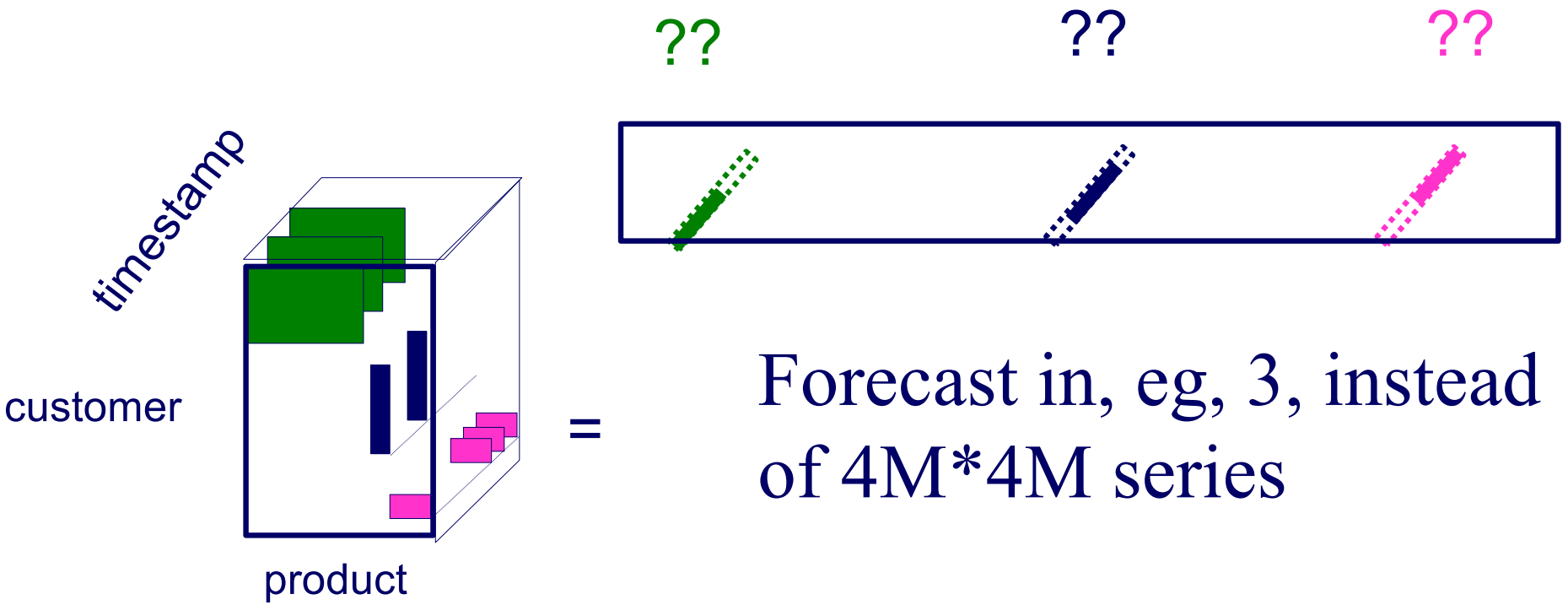
Tensor factorization

- PARAFAC decomposition

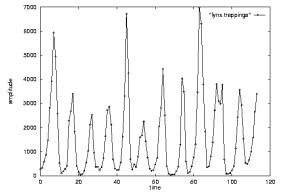


Tensor factorization

- PARAFAC decomposition



Important observations



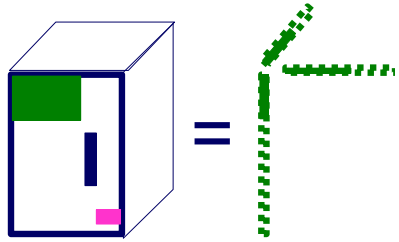
Patterns, rules, forecasting and similarity indexing are closely related:

- 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)

Applications

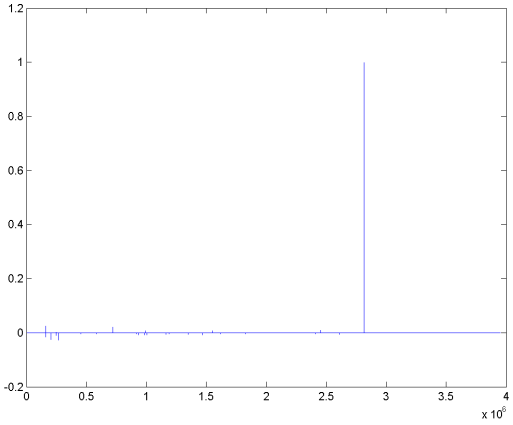
- TA1: Phonecall
- TA2: Network traffic
- TA3: FaceBook

TA1: Anomaly detection in time-evolving graphs

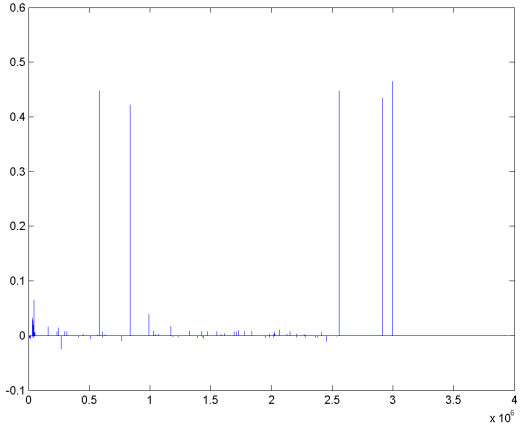


- Anomalous communities in phone call data:
  - European country, 4M clients, data over 2 weeks

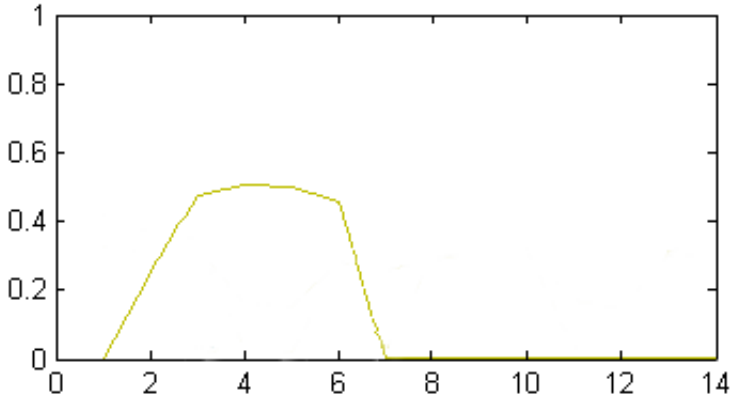
1 caller



5 receivers

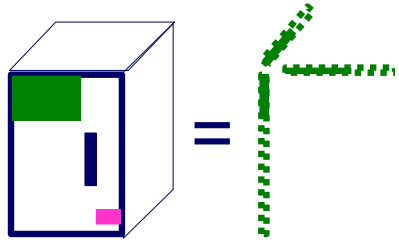


4 days of activity



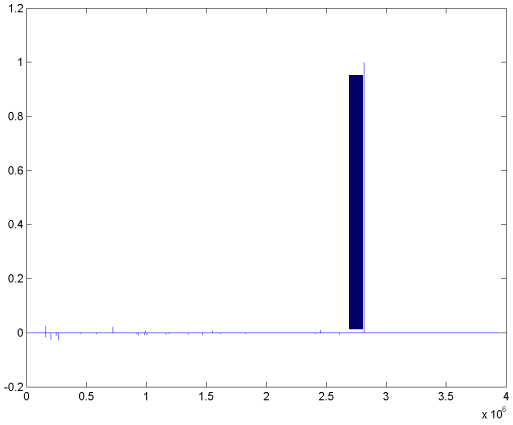
~200 calls to EACH receiver on EACH day!

TA1: Anomaly detection in time-evolving graphs

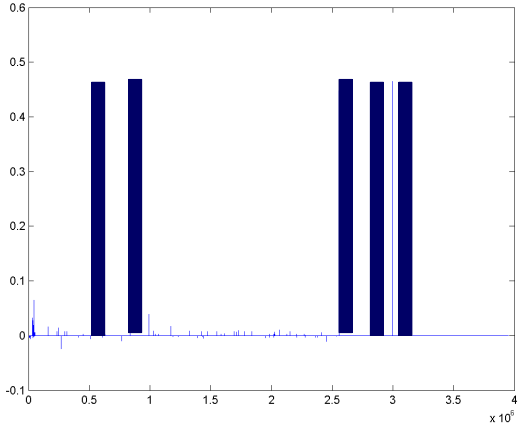


- Anomalous communities in phone call data:
  - European country, 4M clients, data over 2 weeks

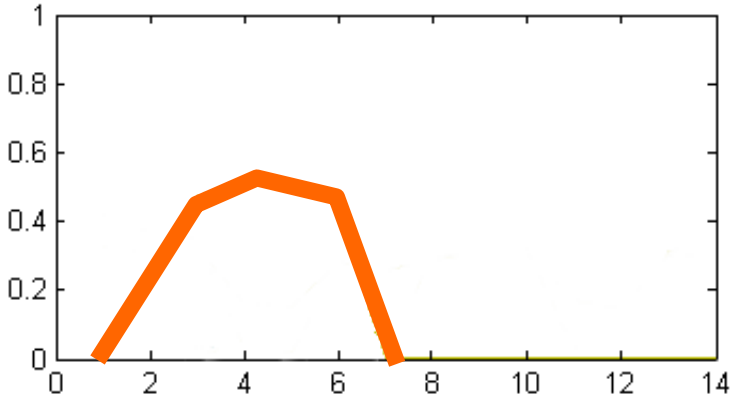
1 caller



5 receivers

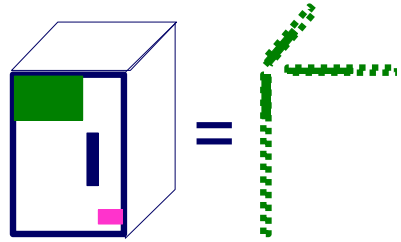


4 days of activity



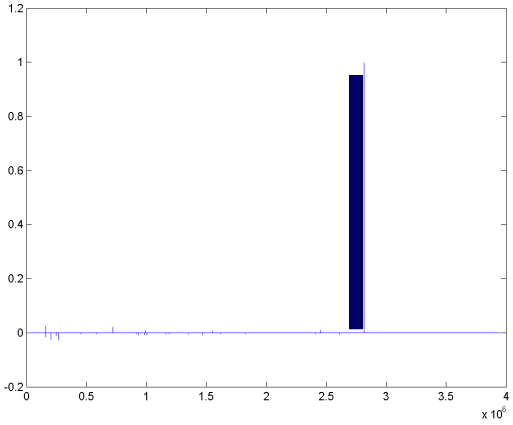
~200 calls to EACH receiver on EACH day!

TA1: Anomaly detection in time-evolving graphs

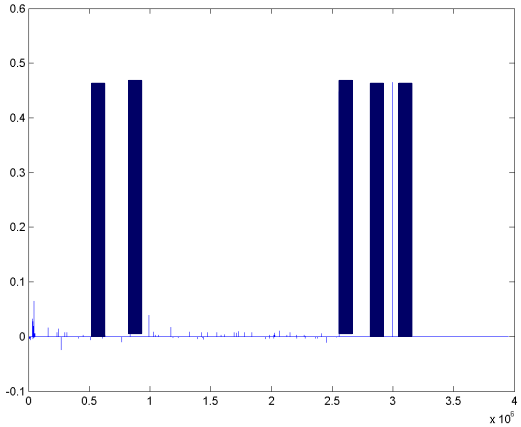


- Anomalous communities in phone call data:
  - European country, 4M clients, data over 2 weeks

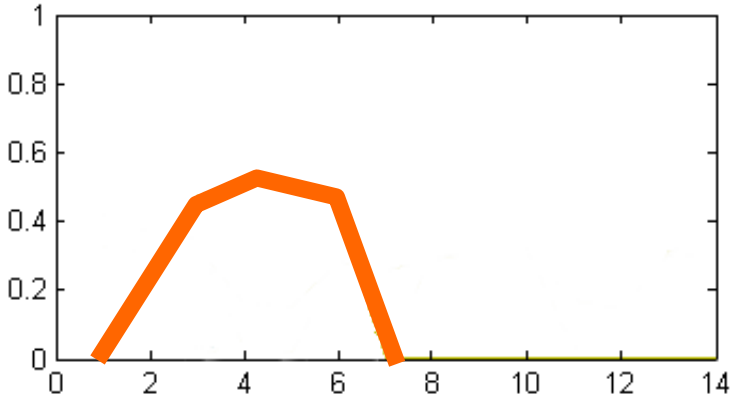
1 caller



5 receivers



4 days of activity

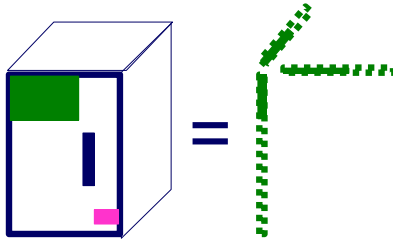


~200 calls to EACH receiver on EACH day!

Carnegie Mellon

amazon

TA1: Anomaly detection in time-evolving graphs



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

Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos Papalexakis, Danai Koutra. *Com2: Fast Automatic Discovery of Temporal (Comet) Communities*. PAKDD 2014, Tainan, Taiwan.

Applications

- TA1: Phonecall
- TA2: Network traffic
- TA3: FaceBook

Carnegie Mellon

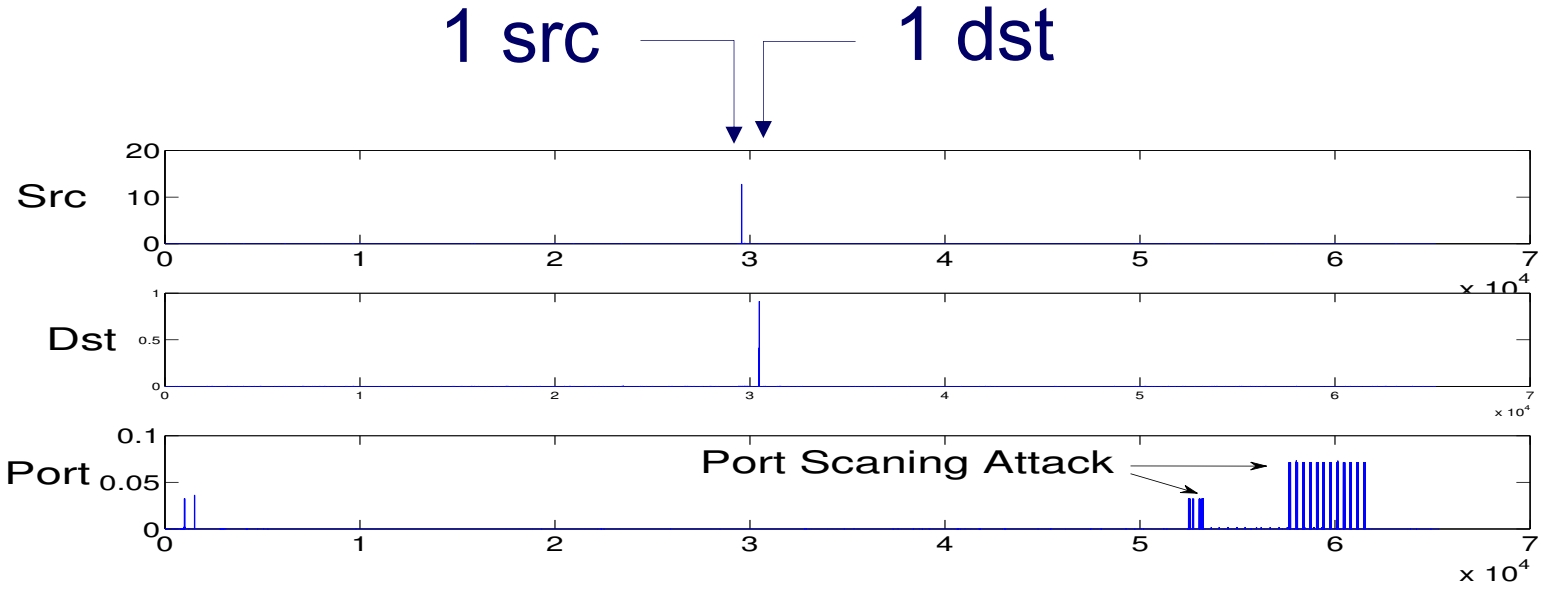
amazon

ParCube: Sparse Parallelizable Tensor Decompositions

Evangelos E. Papalexakis, Christos Faloutsos,
Nikos Sidiropoulos, ECML/PKDD 2012

http://www.cs.ucr.edu/~epapalex

TA2: LBNL Network Data

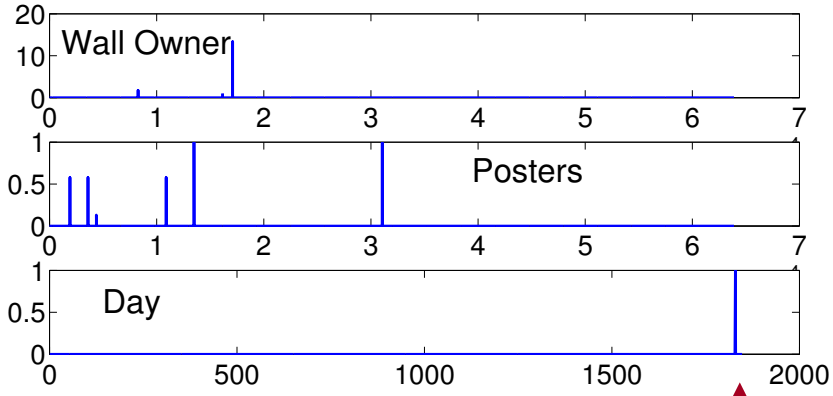


- Modes: src IP, dst IP, port #
- **~ Port Scanning Attack**

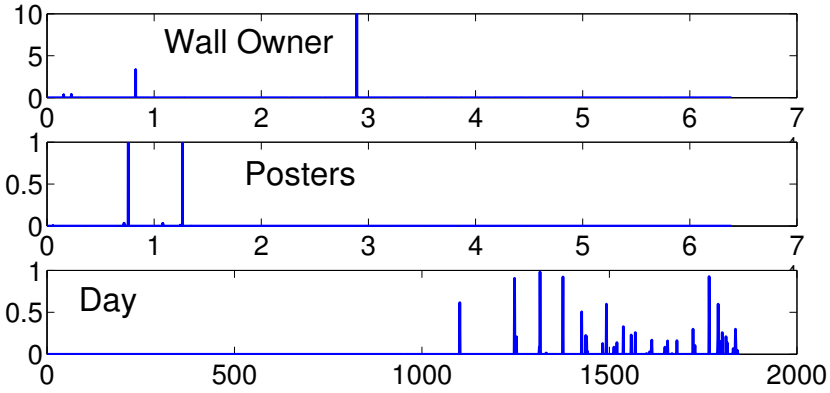
Carnegie Mellon

TA3: FACEBOOK Wall posts

1 Wall



(a) FACEBOOK anomaly

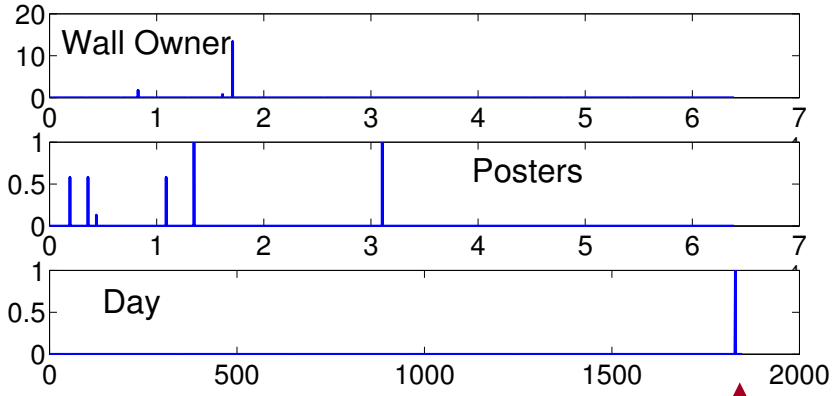


(b) FACEBOOK normal activity

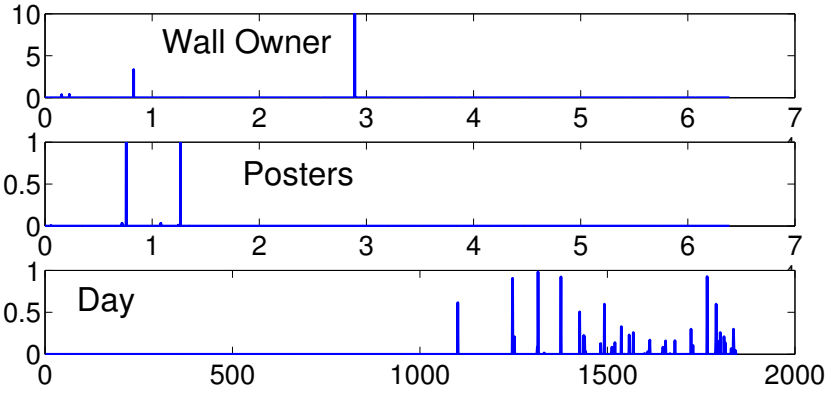
- Modes: wall-owner, poster, timestamp

TA3: FACEBOOK Wall posts

1 Wall



(a) FACEBOOK anomaly (Wall owner's birthday)



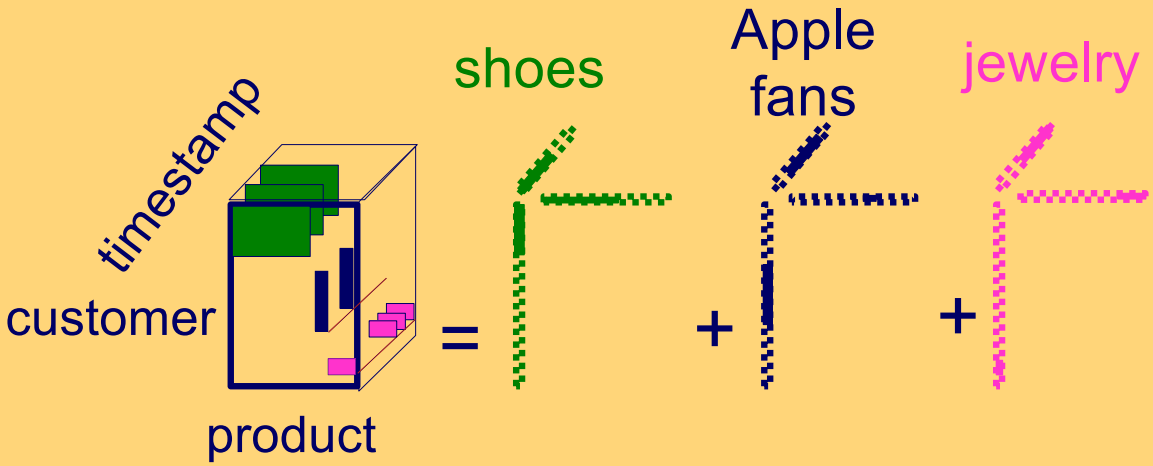
(b) FACEBOOK normal activity

1 day

- Modes: wall-owner, poster, timestamp
- Discovery: birthday-like event.

Conclusions (P1.5)

- Tensor analysis finds latent variables (market-segments, lockstep-groups, etc
  - Deviations → anomalies
- Extends SVD/factorization, to higher-modes



Tensors - More references

Tensor survey

- Tamara G. Kolda and Brett W. Bader
  Tensor Decompositions and Applications
  SIAM Rev., 51(3), pp 455–500, 2009.

Tensors - More references

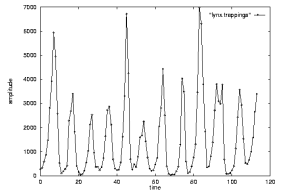
Tensor survey #2

- Nicholas D. Sidiropoulos, Lieven De Lathauwer,, Xiao Fu,, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos, *Tensor Decomposition for Signal Processing and Machine Learning*, IEEE TSP, 65(13), July 1, 2017

Part 1 - Outline

- Motivation
- P1.1. Similarity Search and Indexing
- P1.2. DSP
- P1.3. Linear Forecasting
- P1.4. Non-linear forecasting
- P1.5. Tensors
- Conclusions

Important observations



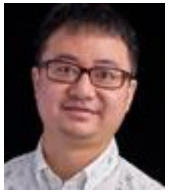
Patterns, rules, forecasting and similarity indexing are closely related:

- 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)

Carnegie Mellon

amazon

Forecasting Big Time Series: Theory and Practice

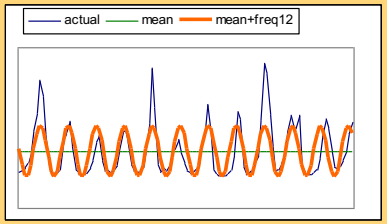


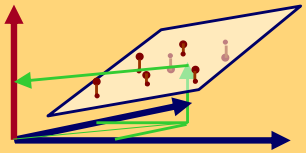
Yuyang (Bernie) Wang, ***Christos Faloutsos****, Valentin Flunkert, Jan Gasthaus and Tim Januschowski*

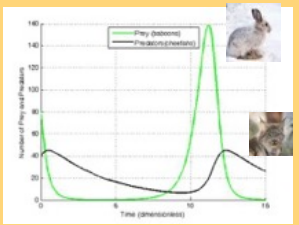

https://lovvge.github.io/Forecasting-Tutorial-WWW-2020/

Part 1 - conclusions

- P1.1. Similarity search: **Euclidean**/time-warping; **feature extraction** and **SAMs**
- P1.2. Periodicities: **DFT/DWT**
- P1.3. Linear Forecasting: **AR** (Box-Jenkins)
- P1.4 Non-linear forecasting: **lag-plots**
  - Gray-box modeling: **Lotka-Volterra**
- P1.5. Tensors: **PARAFAC**







Part 1 - conclusions

- P1.1. Similarity search: **Euclidean**/time-warping; **feature extraction** and **SAMs**
- P1.2. [illegible]: **DFT/DWT**
- [illegible]asting: **AR** (Box-Jenkins)
- [illegible] forecasting: **lag-plots**
- Gray-box modeling: **Lotka-Volterra**
- P1.5. Tensors: **PARAFAC**

1) PLOT IT
2) DFT / DWT