Indexing and Mining Time Sequences

Christos Faloutsos and Lei Li

CMU

Outline

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

Problem definition

- Given: one or more sequences

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

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

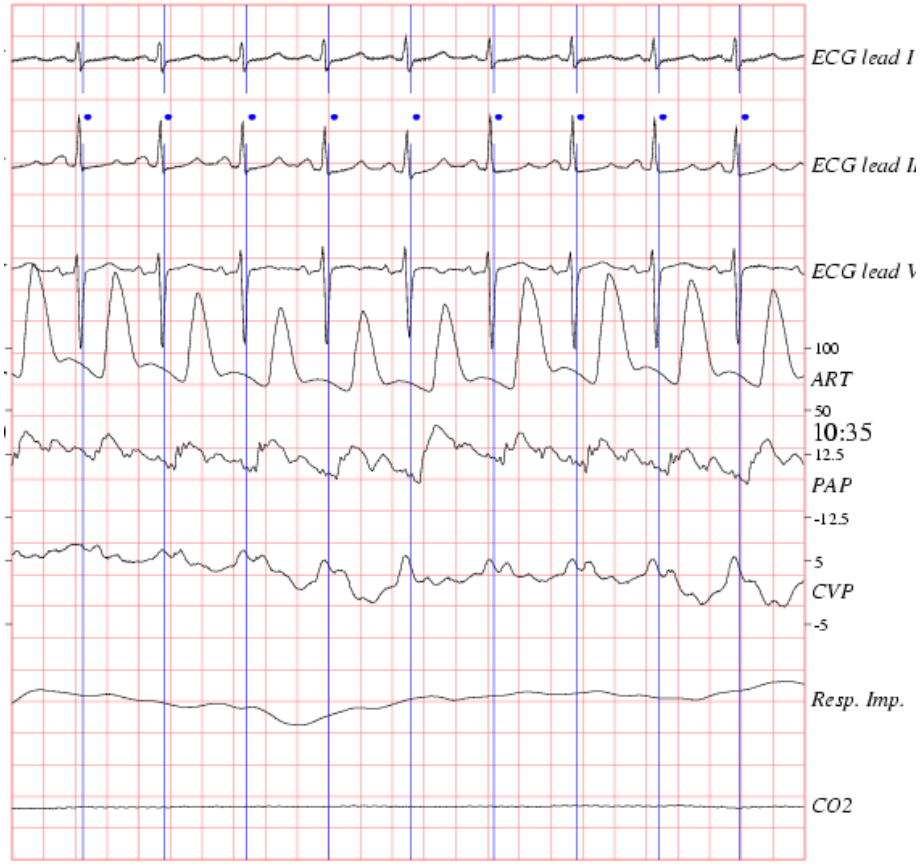
  $\ldots)$

- Find
  - similar sequences; forecasts
  - patterns; clusters; outliers

Motivation - Applications

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

ECG - physionet.org



EEG - epilepsy

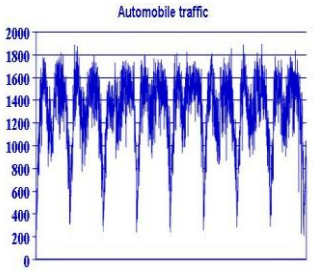
from wikipedia

Motivation - Applications (cont'd)

- 'Smart house'
  - sensors monitor temperature, humidity, air quality
- video surveillance

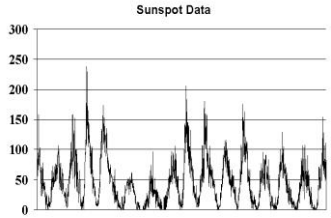
Motivation - Applications (cont'd)

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



Motivation - Applications (cont'd)

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

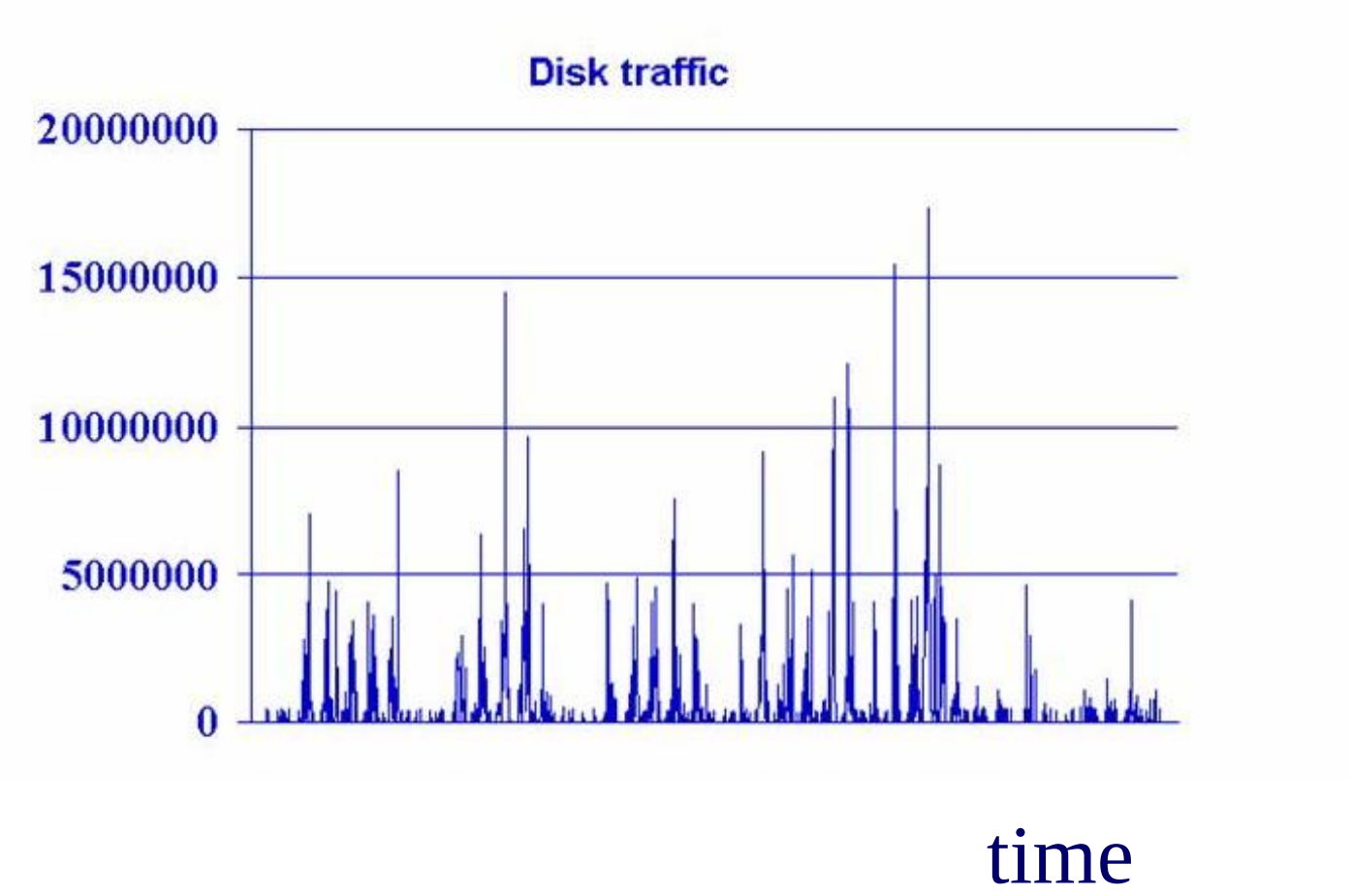


Motivation - Applications (cont'd)

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

Stream Data: Disk accesses

#bytes



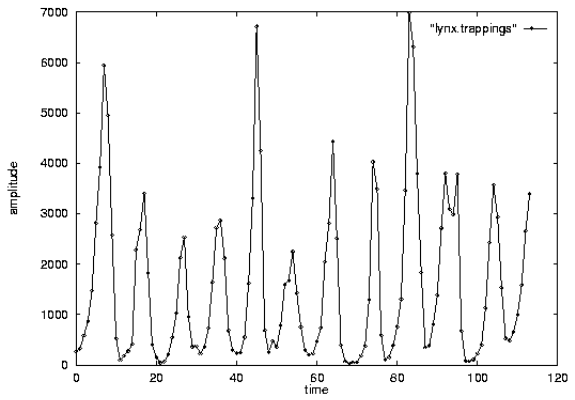
time

Problem #1:

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

Find: patterns, periodicities, and/or compress

count

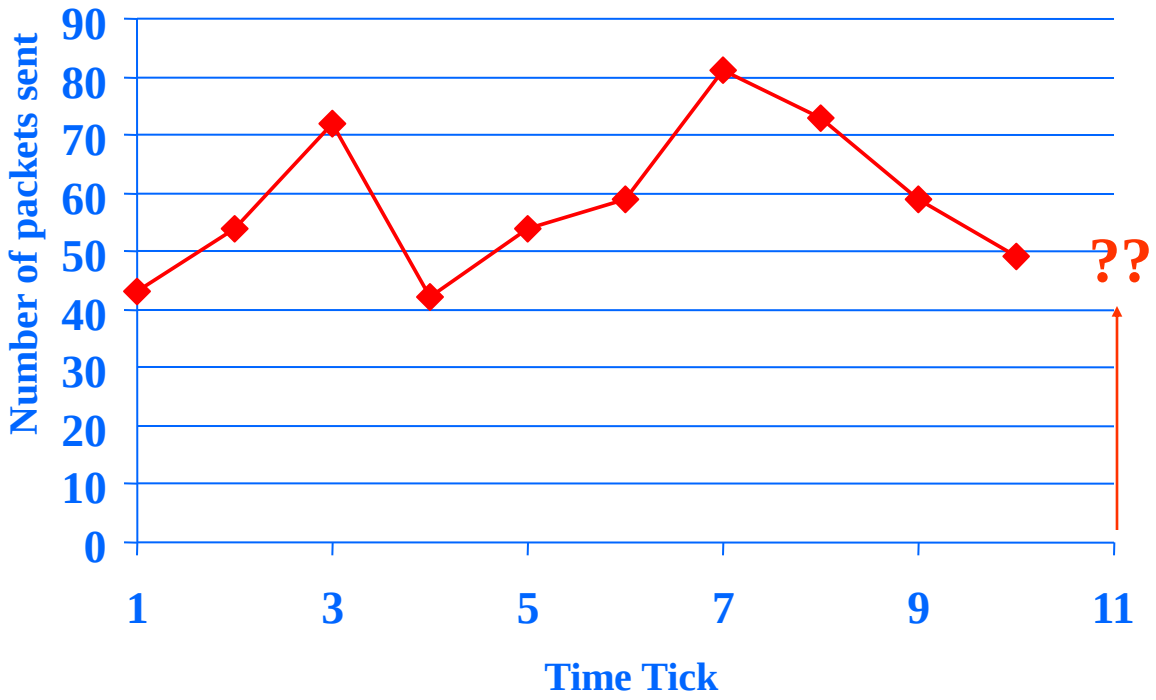


lynx caught per year
(packets per day;
temperature per day)

year

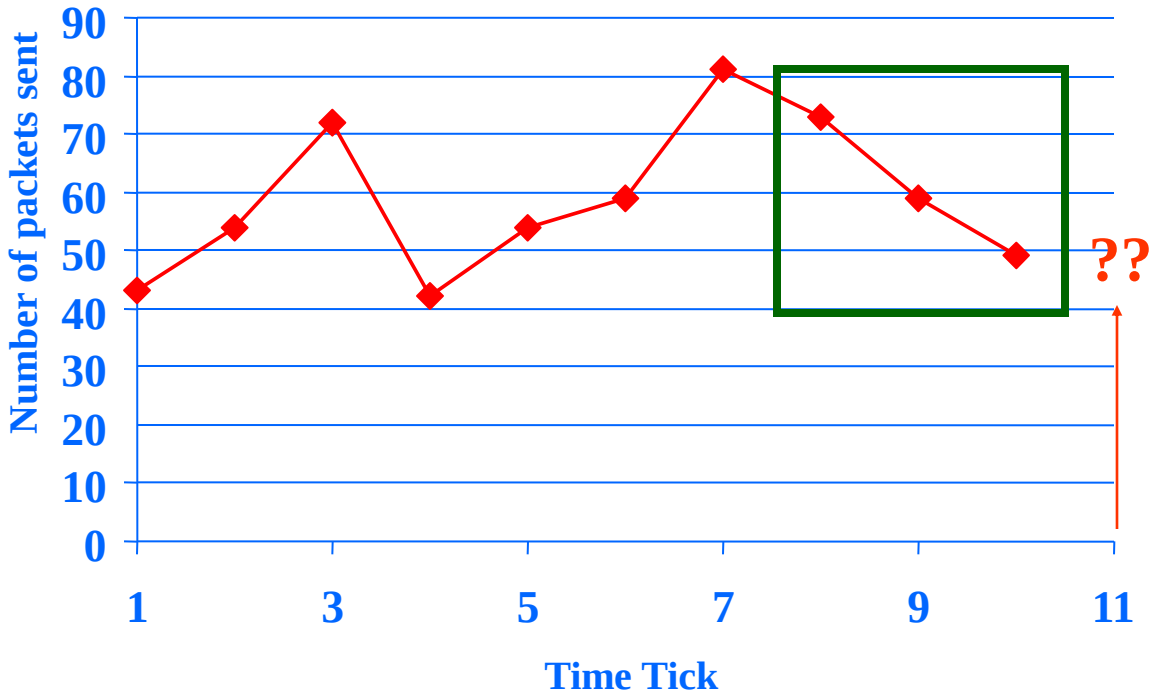
Problem#2: Forecast

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



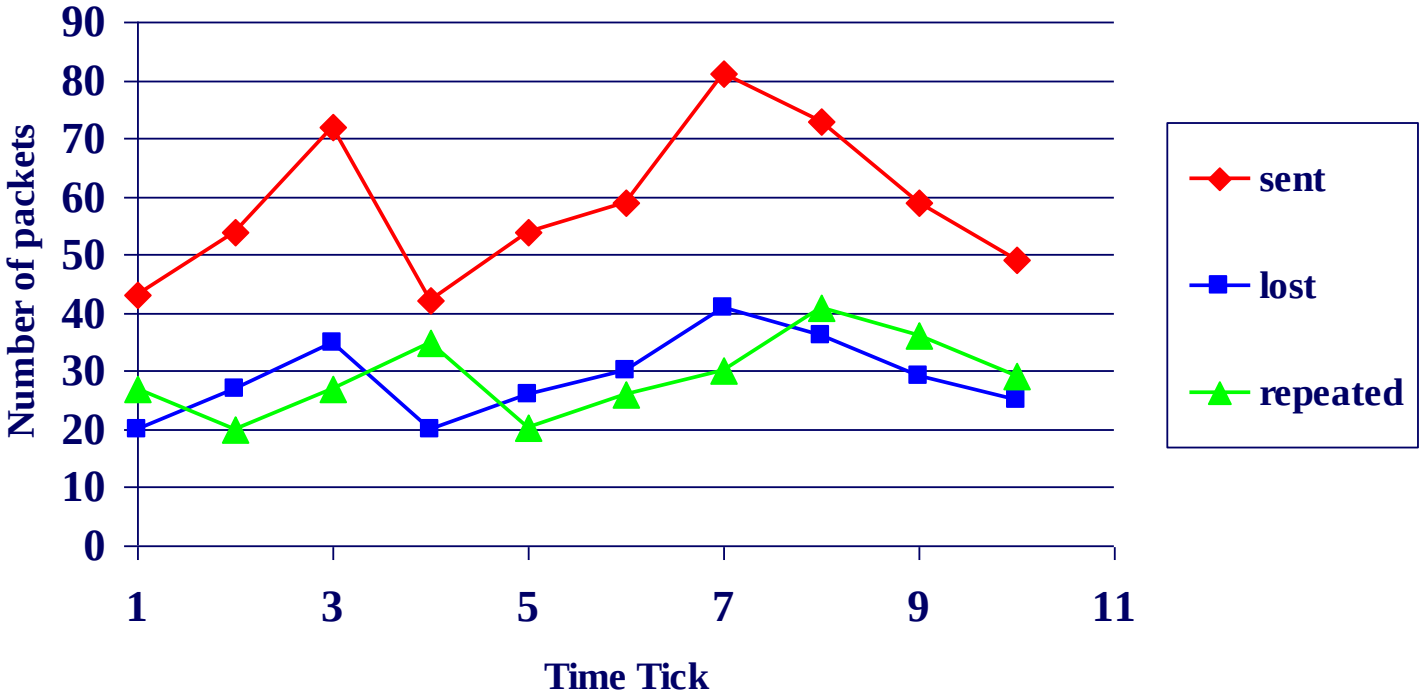
Problem#2': Similarity search

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
  - to find similar settings in the past
- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)

Important topics NOT in this tutorial:

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

Outline

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

Outline

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

Importance of distance functions

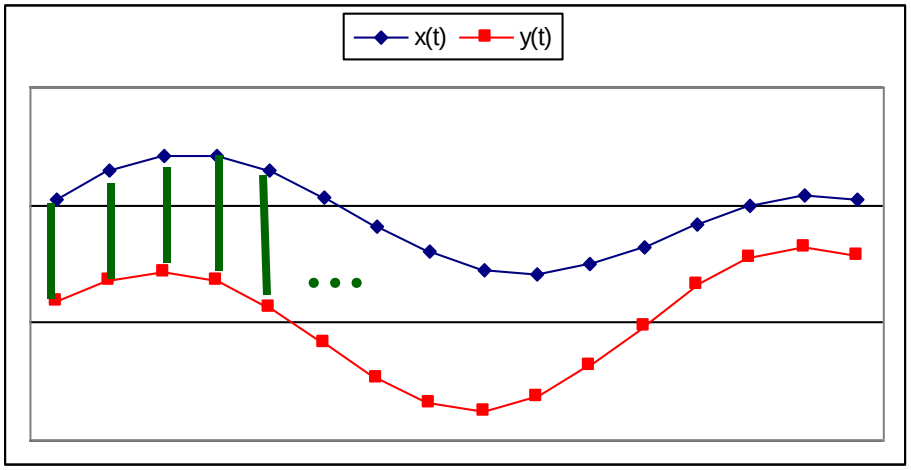
Subtle, but **absolutely necessary**:

- A ‘must’ for similarity indexing (-> forecasting)
- A ‘must’ for clustering

Two major families

- Euclidean and Lp norms
- Time warping and variations

Euclidean and Lp

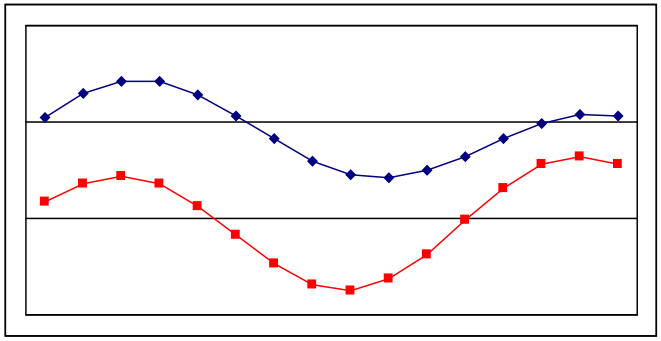


$$D(\vec{x},\vec{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$$

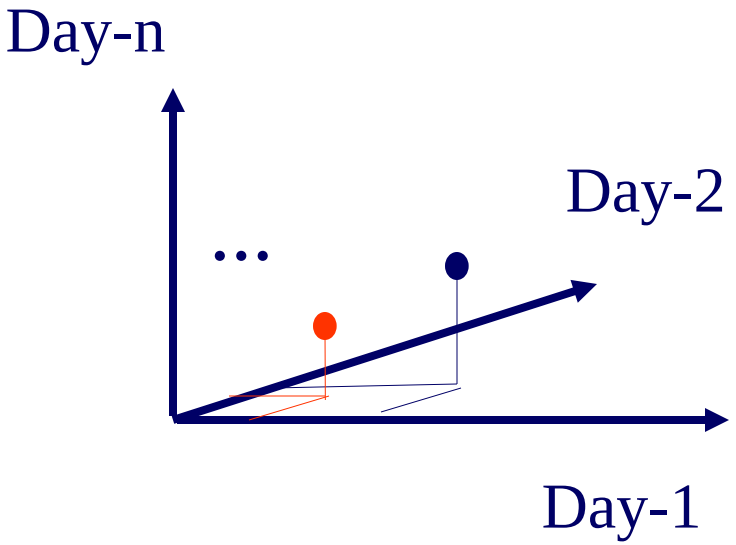
$$L_p(\vec{x},\vec{y}) = \sum_{i=1}^{n} | x_i - y_i |^p$$

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

Observation #1



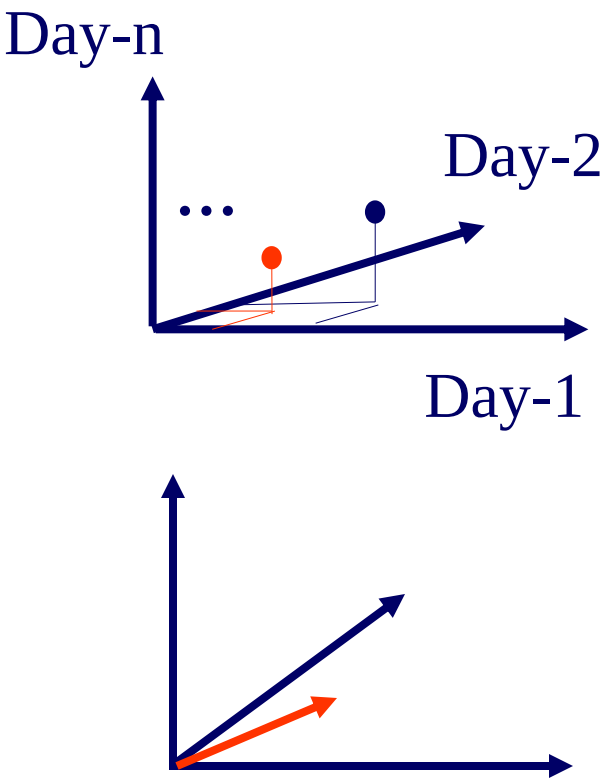
- **Time sequence -> n-d vector**



Observation #2

Euclidean distance is closely related to

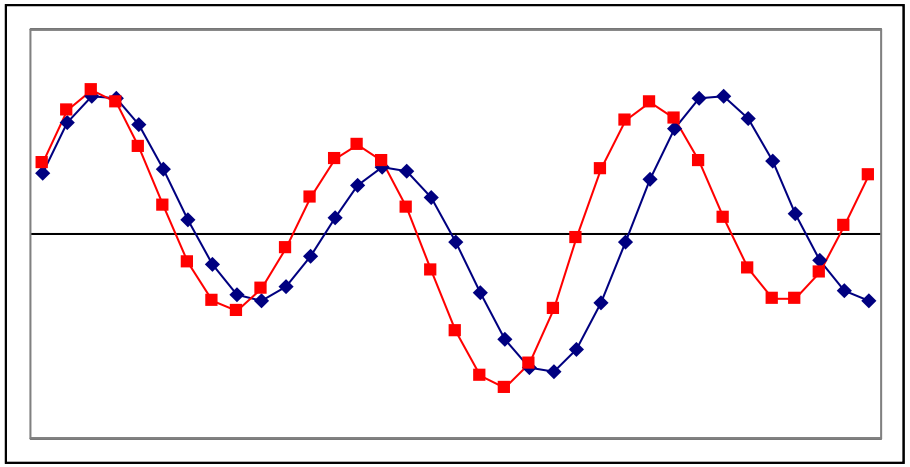
- cosine similarity
- dot product
- 'cross-correlation' function



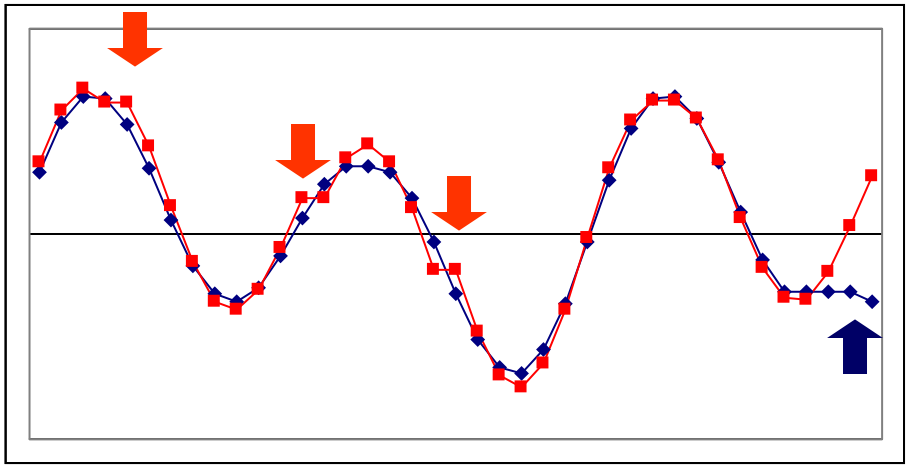
Time Warping

- allow accelerations - decelerations
  - (with or w/o penalty)
- THEN compute the (Euclidean) distance (+ penalty)
- related to the string-editing distance

Time Warping



‘stutters’:



Skip

Time Warping

Q: how to compute it?

A: dynamic programming

D(i, j) = cost to match

prefix of length *i* of first sequence *x* with prefix of length *j* of second sequence *y*

Skip

Time Warping

Thus, with no penalty for stutter, for sequences

$x_1, x_2, \ldots, x_{i,;}$ $\quad$ $y_1, y_2, \ldots, y_j$

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

Other Distance functions

- piece-wise linear/flat approx.; compare pieces [Keogh+01] [Faloutsos+97]
- 'cepstrum' (for voice [Rabiner+Juang])
  - do DFT; take log of amplitude; do DFT again!
- Allow for small gaps [Agrawal+95]

More distance functions.

- Chen + Ng [vldb'04]: ERP 'Edit distance with Real Penalty': give a penalty to stutters
- Keogh+ [kdd'04]: VERY NICE, based on information theory: compress each sequence (quantize + Lempel-Ziv), using the **other** sequences' LZ tables

On The Marriage of Lp-norms and Edit Distance, Lei Chen, Raymond T. Ng:, VLDB'04

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

Conclusions

Prevailing distances:

- Euclidean and
- time-warping

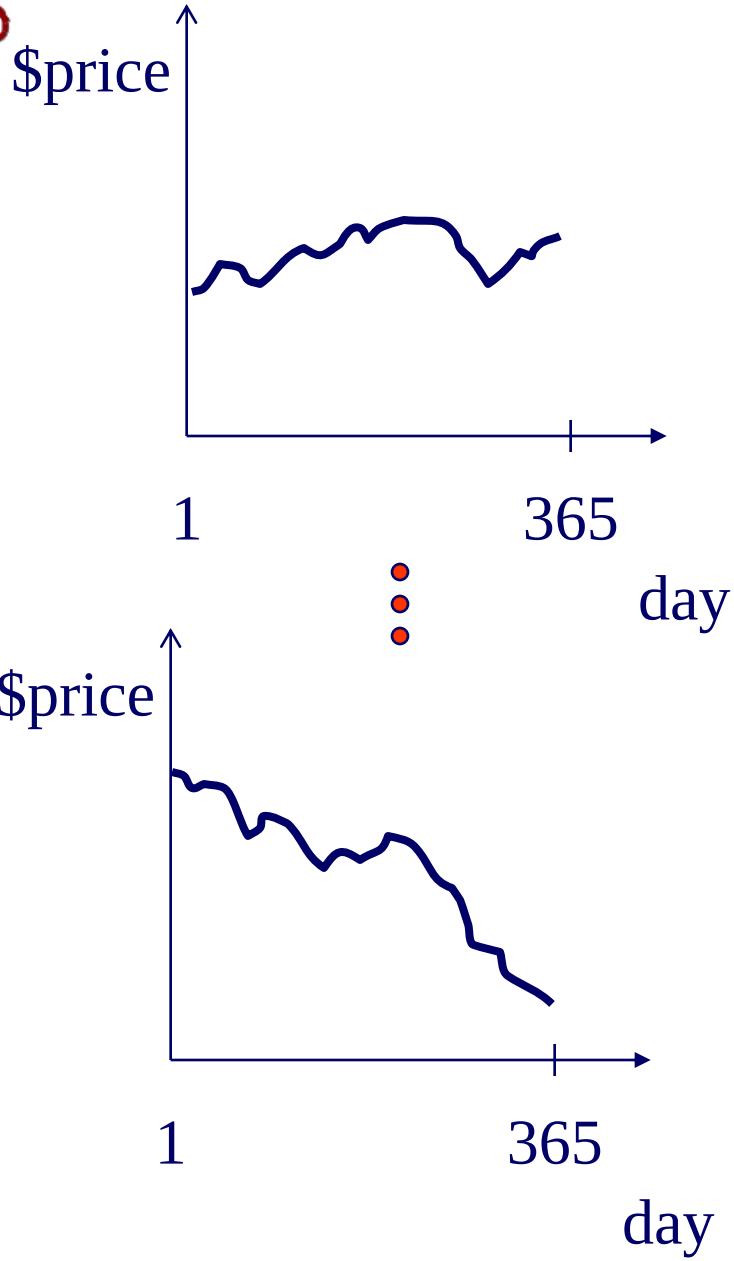
Outline

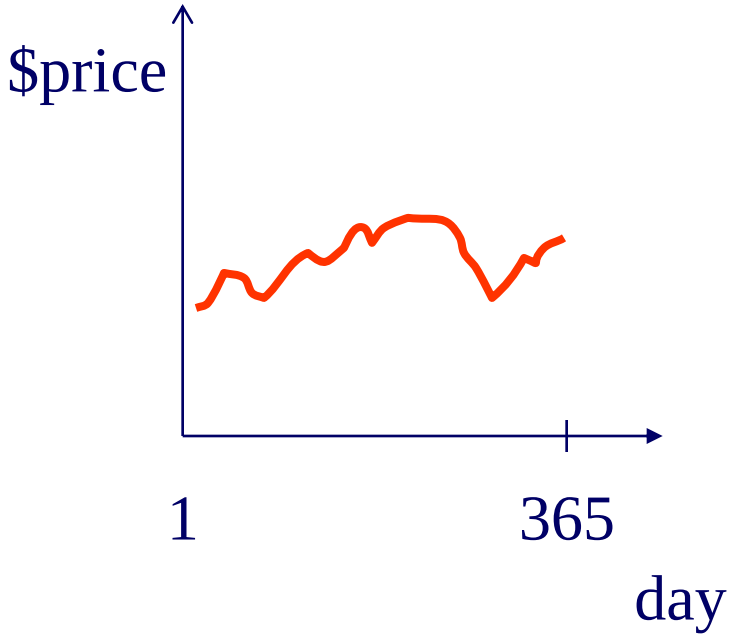
- Motivation
- Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
- DSP
- ...

Indexing

Problem:

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





distance function: by expert

Idea: 'GEMINI'

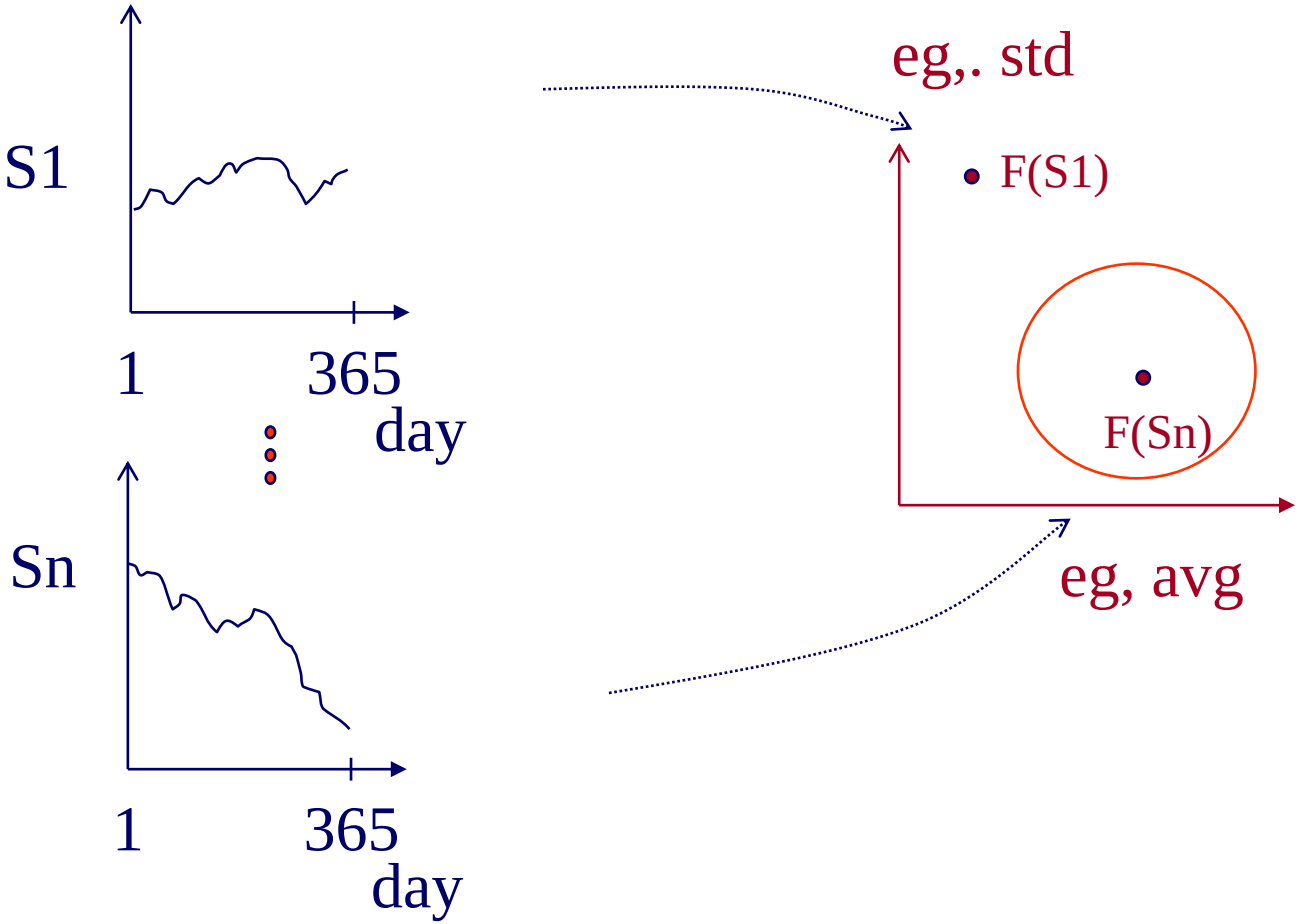
Eg., '*find stocks similar to MSFT*'

Seq. scanning: too slow

How to accelerate the search?

[Faloutsos96]

'GEMINI' - Pictorially



GEMINI

Solution: Quick-and-dirty' filter:

- extract *n* features (numbers, eg., avg., etc.)
- map into a point in *n*-d feature space
- organize points with off-the-shelf spatial access method ('SAM')
- discard false alarms

Examples of GEMINI

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

Indexing - SAMs

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

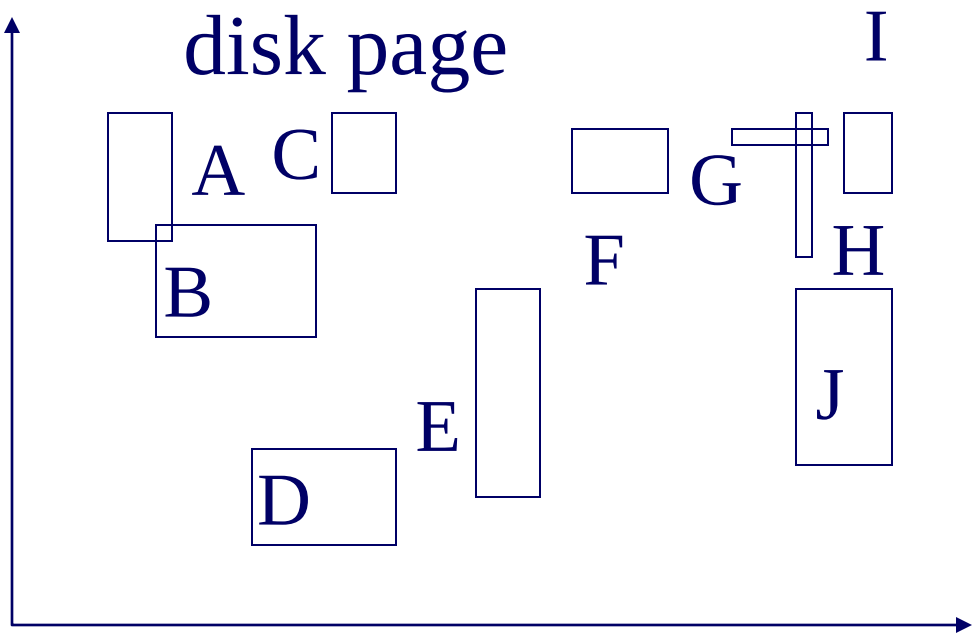
A: they group nearby points (or regions) together, on nearby disk pages, and answer spatial queries quickly (‘range queries’, ‘nearest neighbor’ queries etc)

For example:

Skip

R-trees

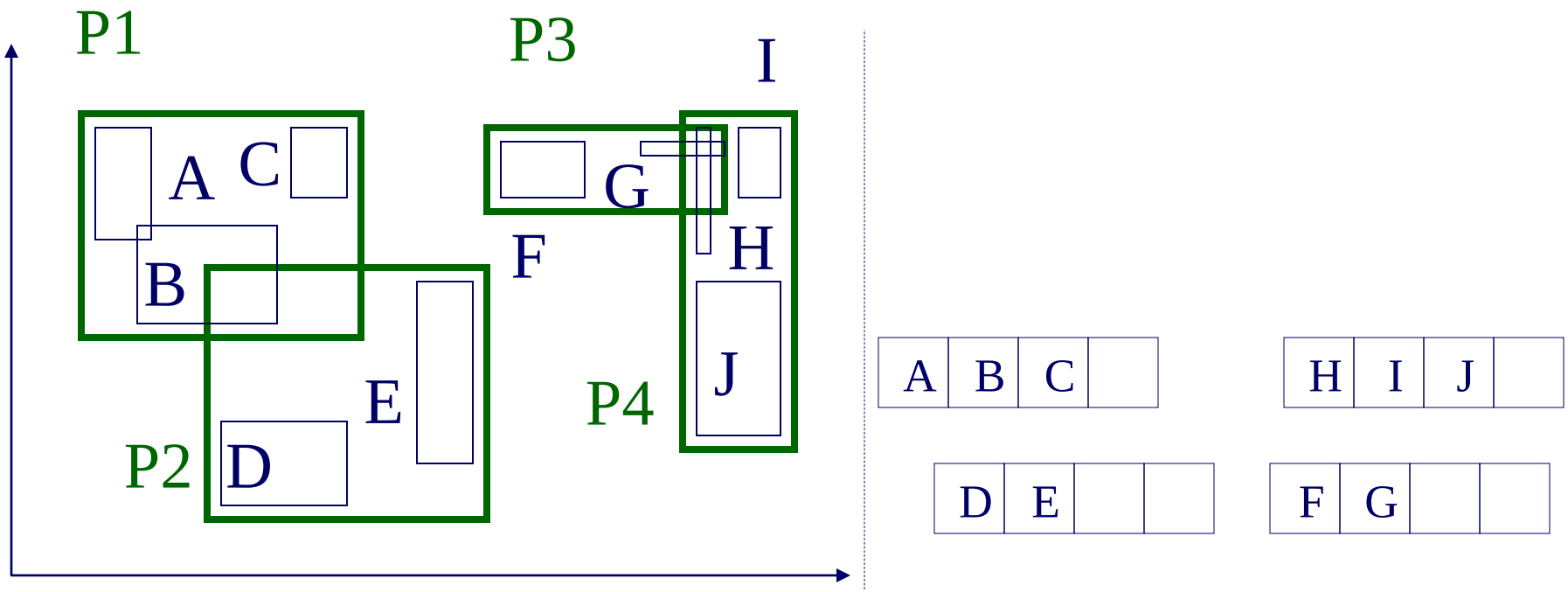
- [Guttman84] eg., w/ fanout 4: group nearby rectangles to parent MBRs; each group -> disk page



Skip

R-trees

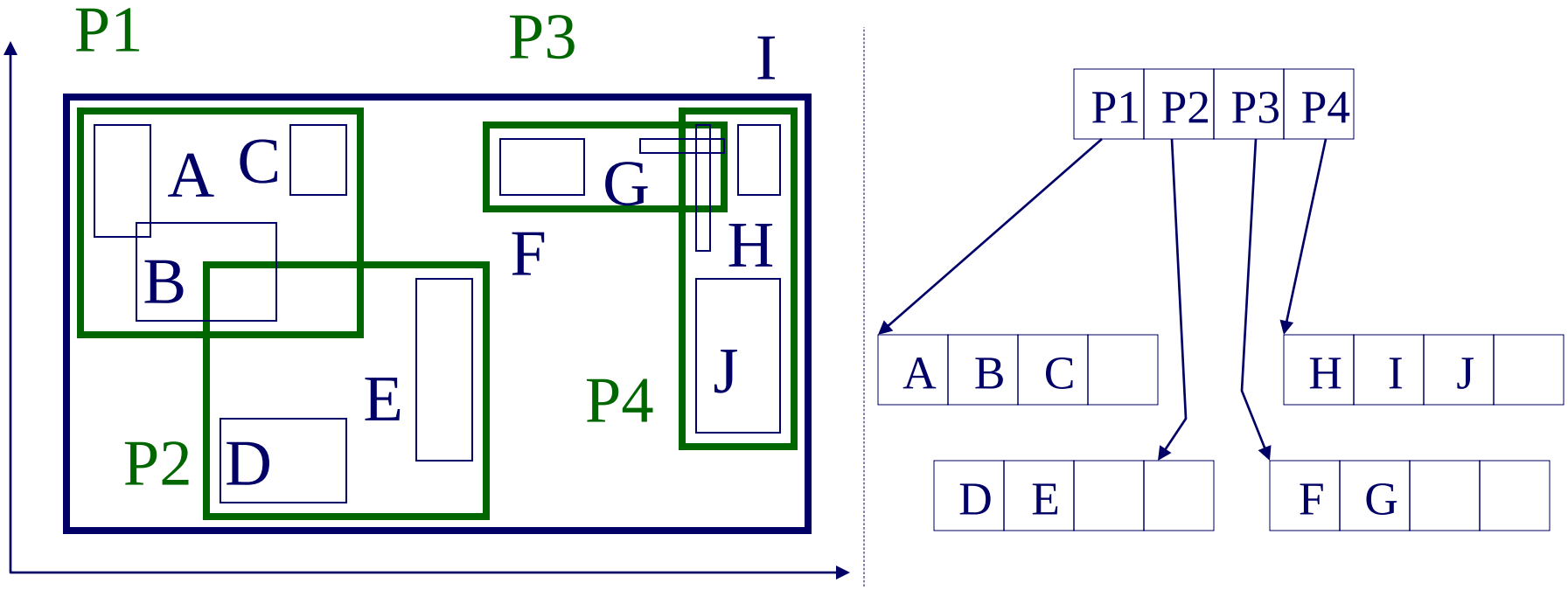
- eg., w/ fanout 4:



Skip

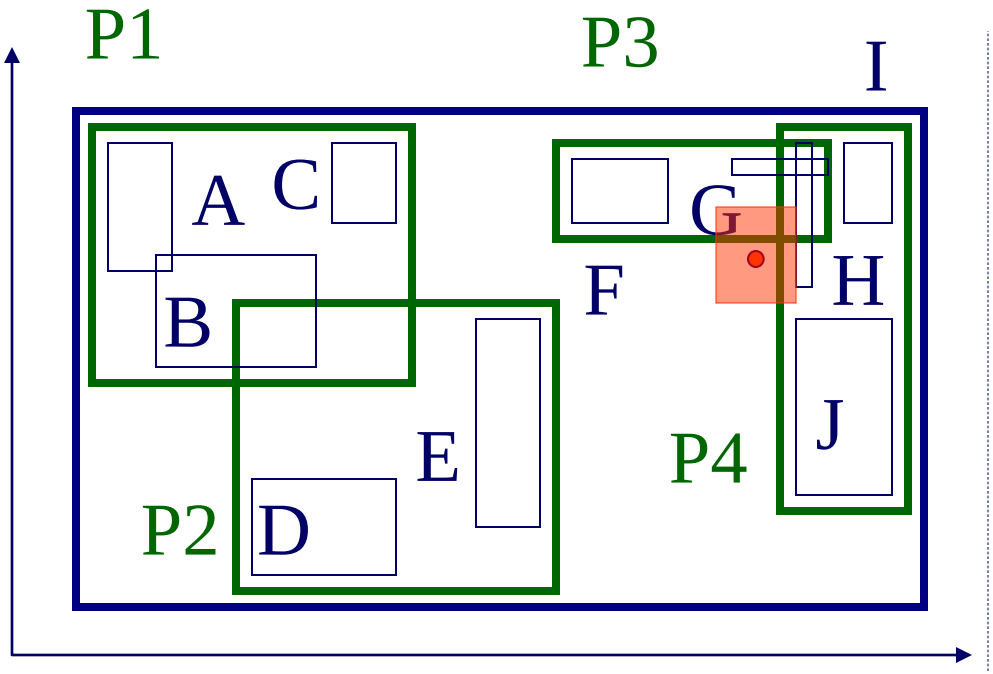
R-trees

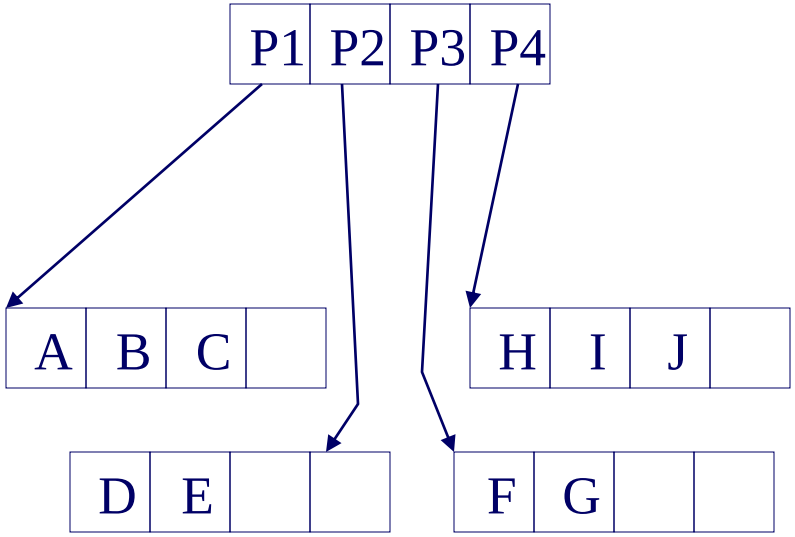
- eg., w/ fanout 4:



Skip

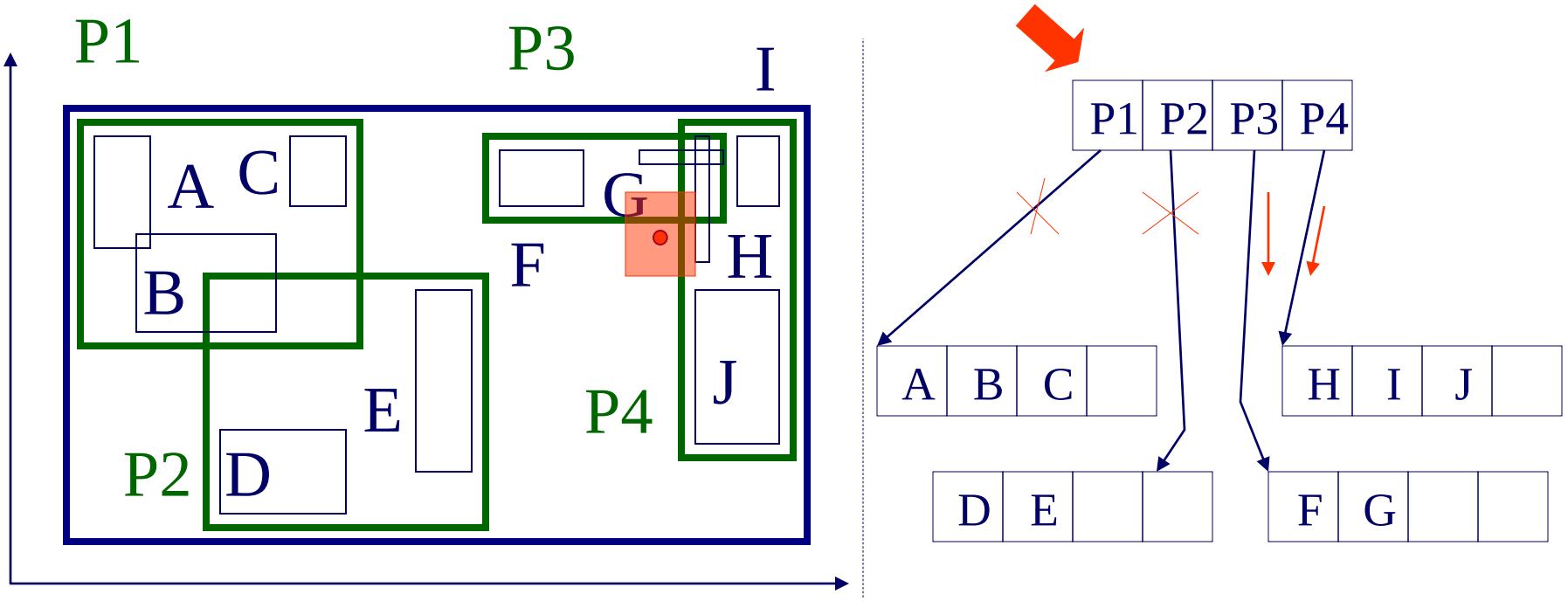
R-trees - range search?





Skip

R-trees - range search?



Conclusions

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

Outline

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

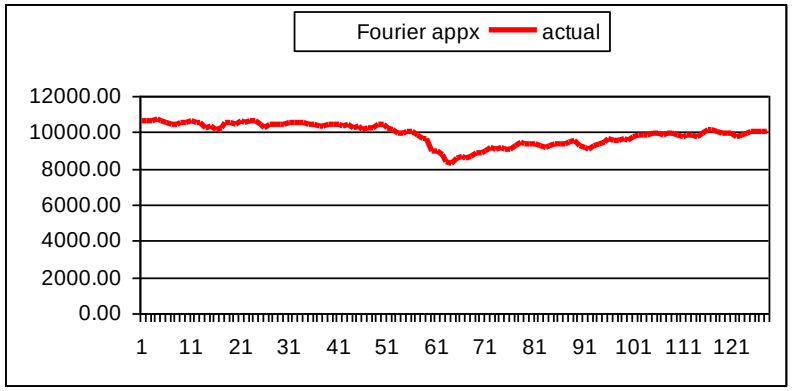
Outline

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

DFT and cousins

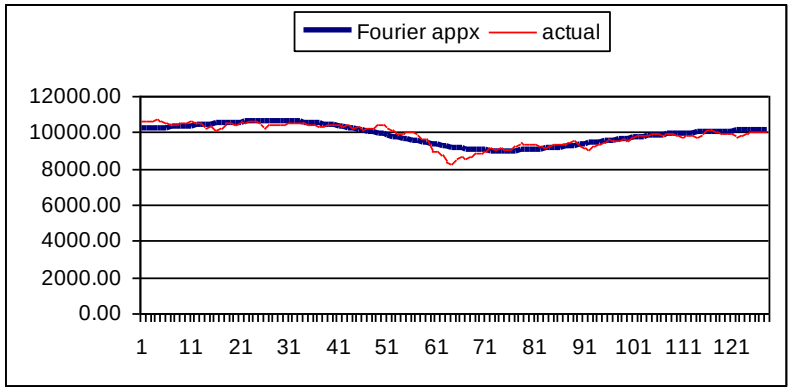
- very good for compressing real signals
- more details on DFT/DCT/DWT: later

DFT and stocks

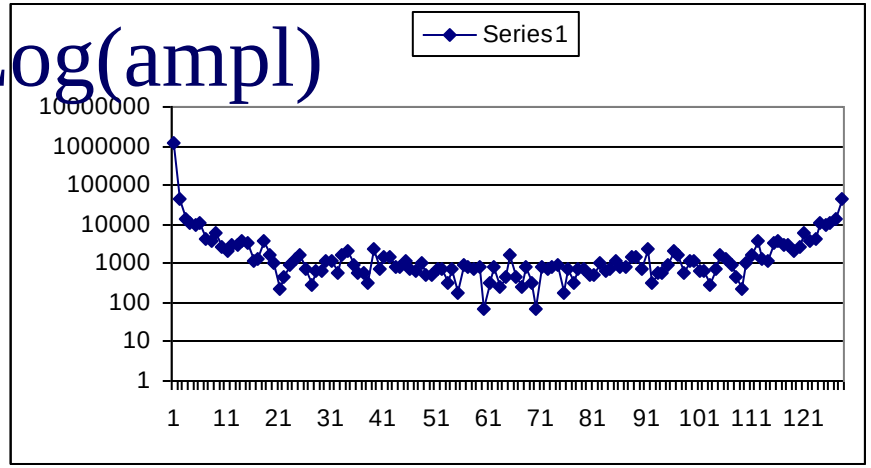


- Dow Jones Industrial index, 6/18/2001-12/21/2001

DFT and stocks



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



Outline

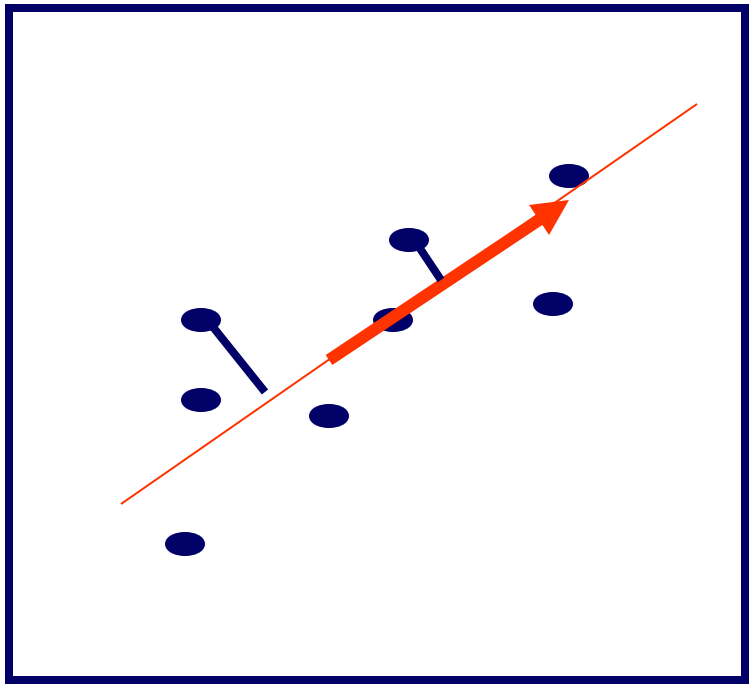
- Motivation
- Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
    - DFT, DWT, DCT (data independent)
    - SVD etc (data dependent)
    - MDS, FastMap

SVD

- THE optimal method for dimensionality reduction
  - (under the Euclidean metric)

Singular Value Decomposition (SVD)

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



LSI: S. Dumais; M. Berry

KL: eg, Duda+Hart

PCA: eg., Jolliffe

Details: [Press+],

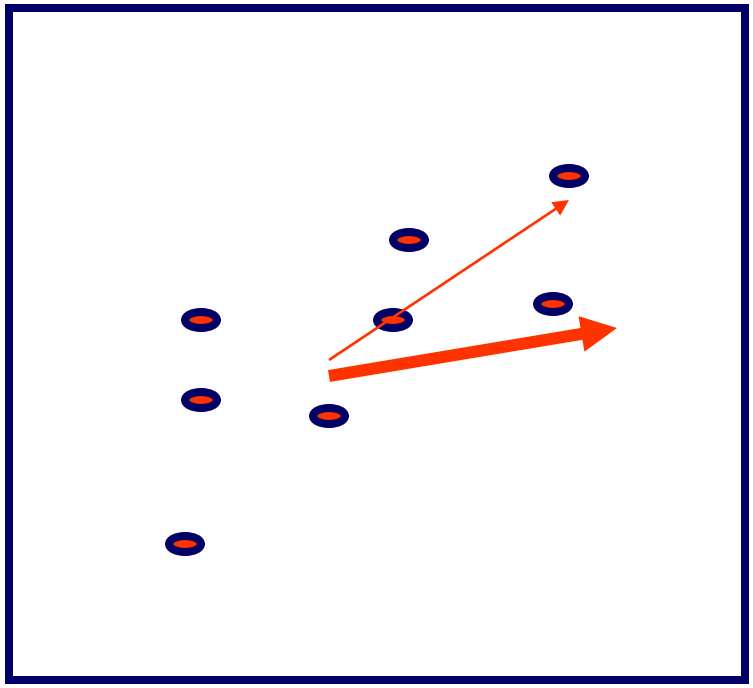
[Faloutsos96]

SVD

- **Extremely** useful tool
  - (also behind PageRank/google and Kleinberg's algorithm for hubs and authorities)
- But may be slow: O($N * M * M$) if $N>M$
- any approximate, faster method?

SVD shorcuts

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



Random projections

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

Outline

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

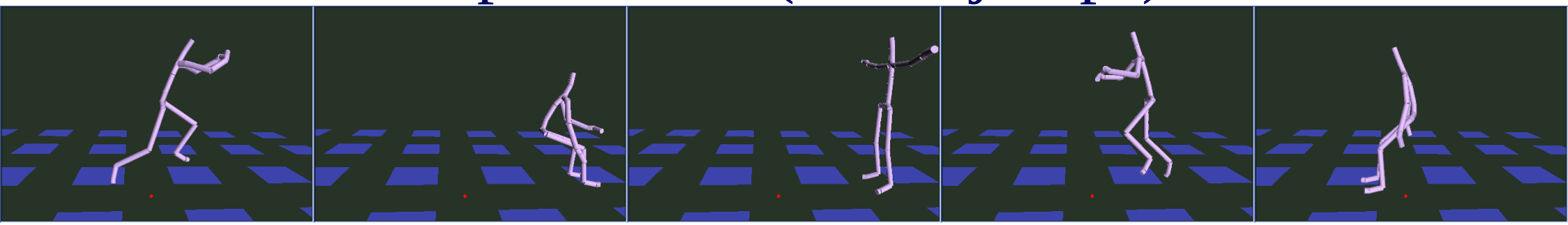
Citation

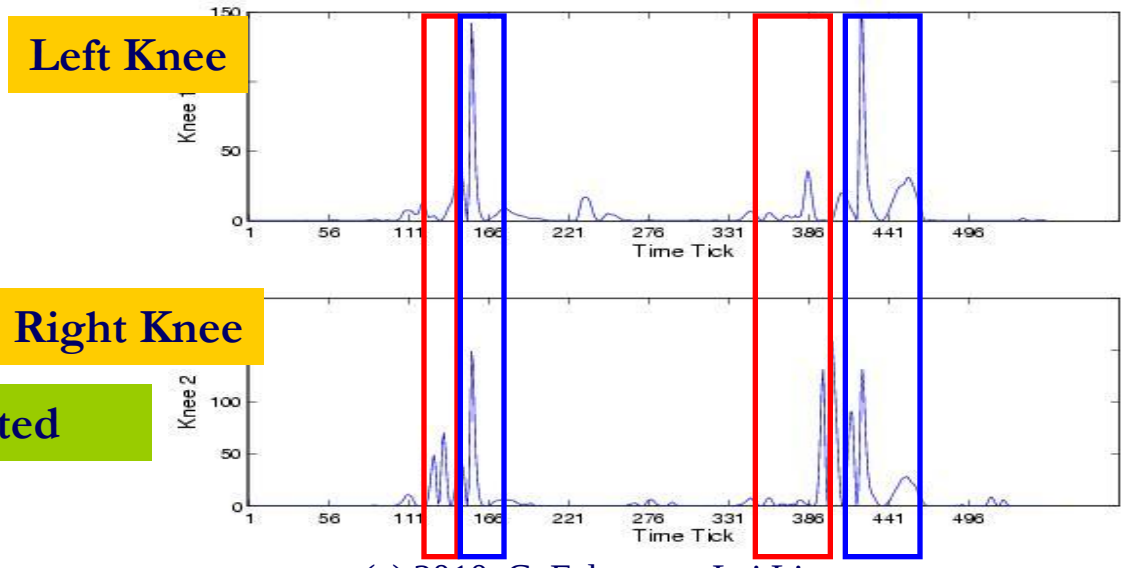
- *AutoSplit: Fast and Scalable Discovery of Hidden Variables in Stream and Multimedia Databases*, **Jia-Yu Pan**, Hiroyuki Kitagawa, Christos Faloutsos and Masafumi Hamamoto

PAKDD 2004, Sydney, Australia

Motivation: (Q1) Find patterns in data

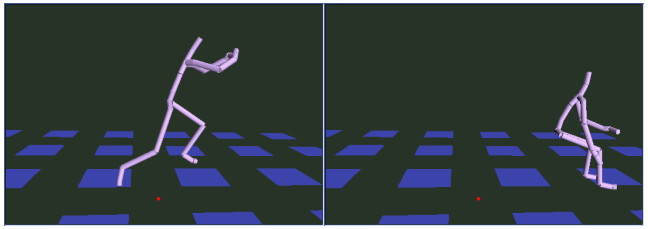
- Motion capture data (broad jumps)

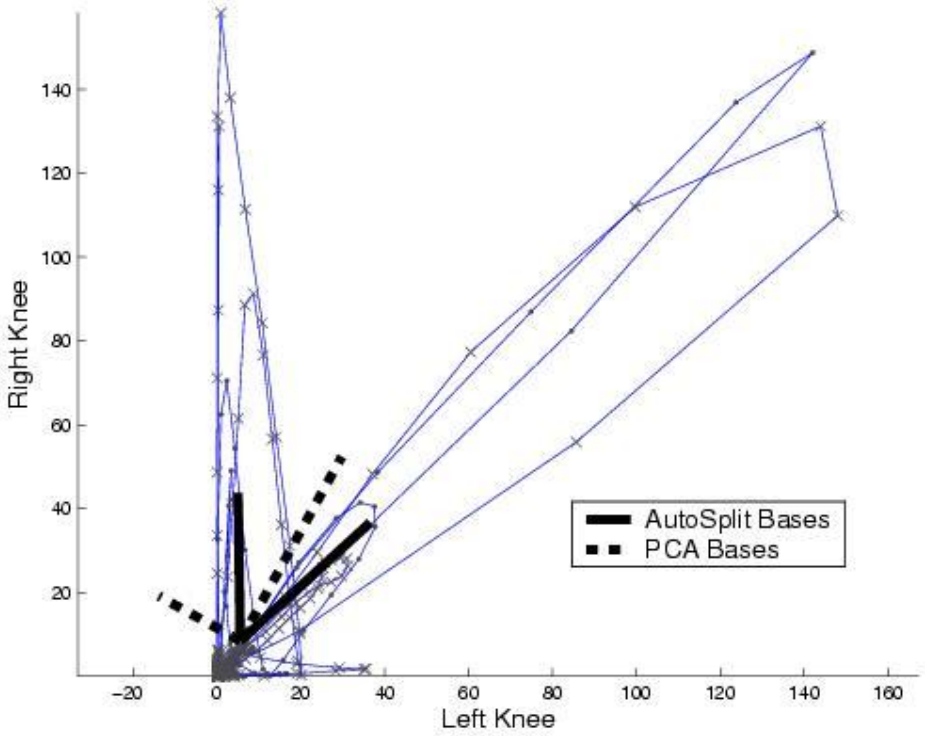




PCA sometimes misses essential features

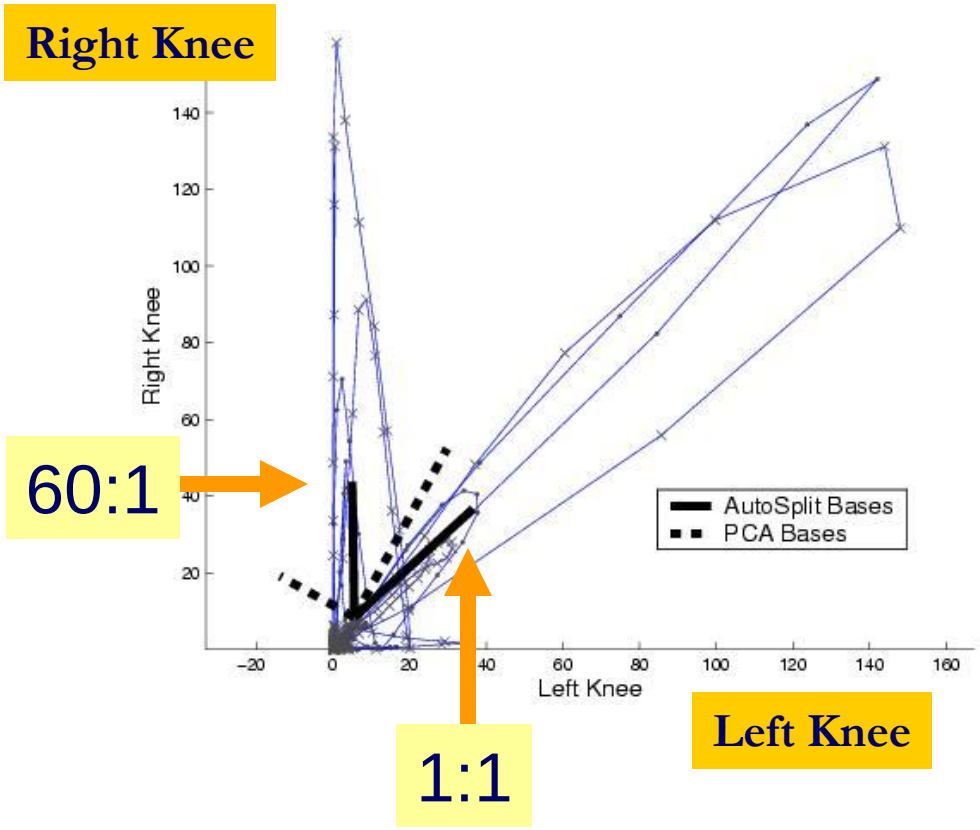
- Best SVD axis: not always meaningful!





Motivation: (Q1) Find patterns in data

- Human would say
  - Pattern 1: along diagonal
  - Pattern 2: along vertical axis
- How to find these automatically?



Motivation: (Q2) Find hidden variables

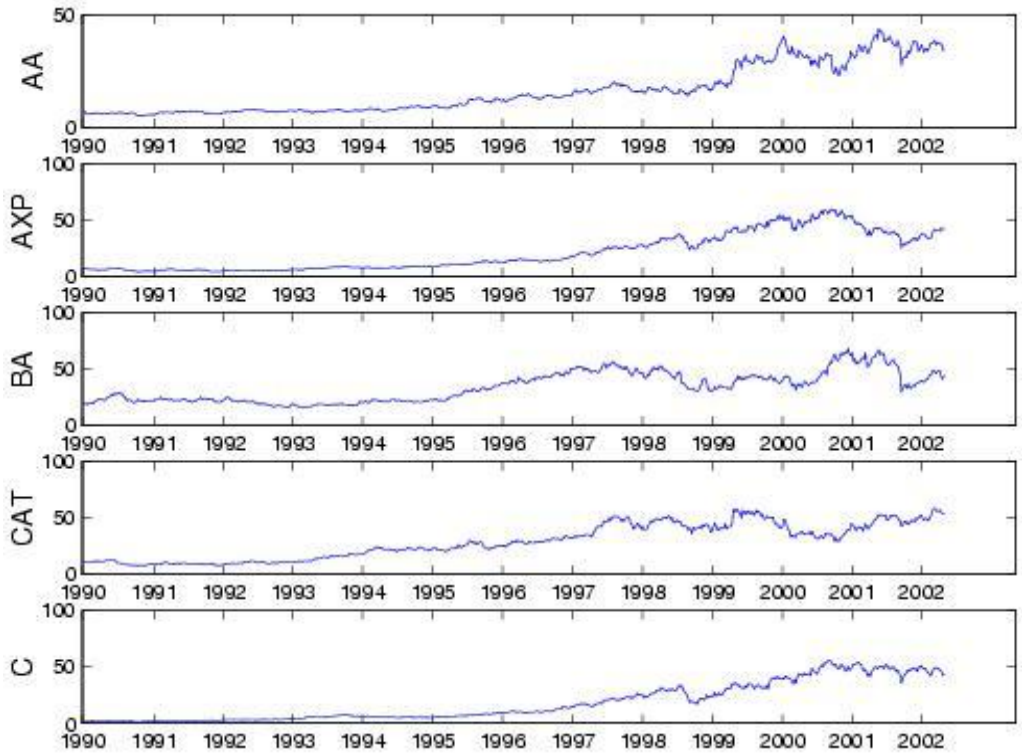
Alcoa

American Express

Boeing

Caterpillar

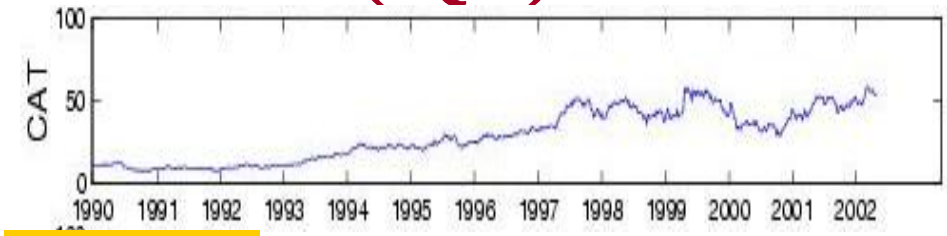
Citi Group

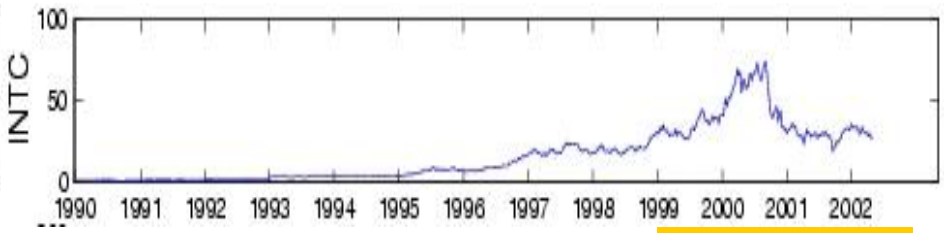


Dow Jones Industrial Average

Find common hidden variables, and weights.

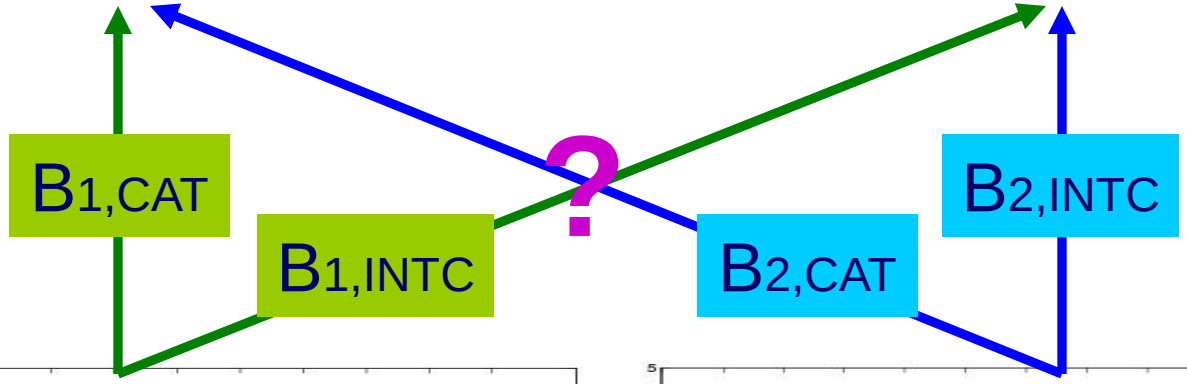
Motivation:
(Q2) Find hidden variables

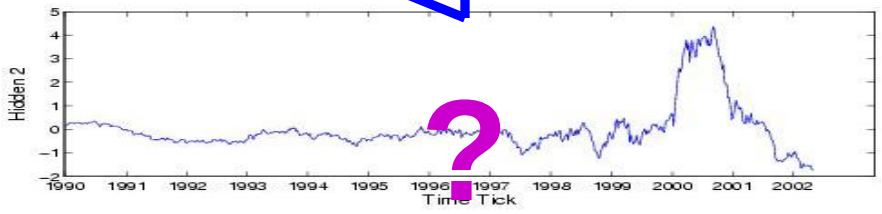




Caterpillar

Intel

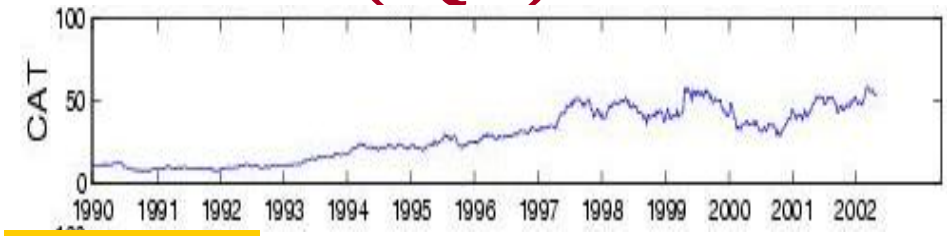


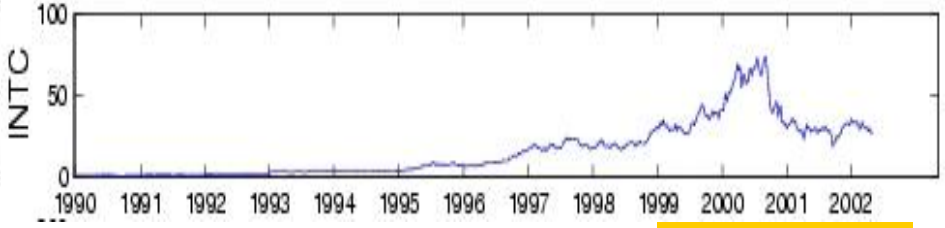


Hidden variable 1

Hidden variable 2

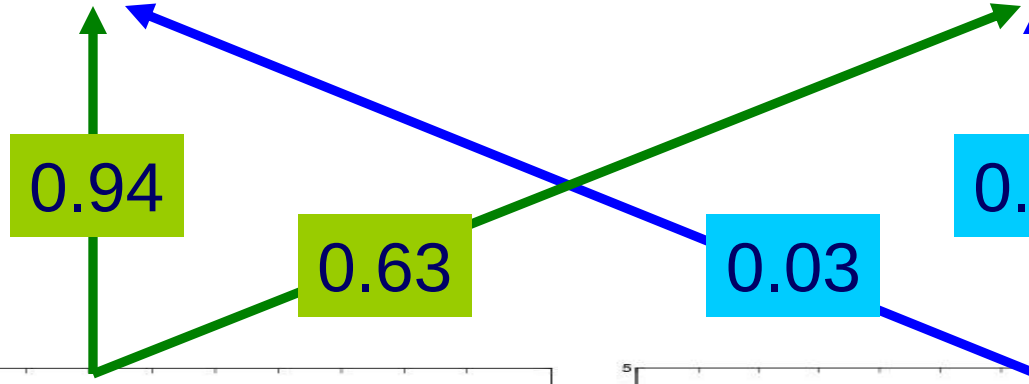
Motivation:
(Q2) Find hidden variables

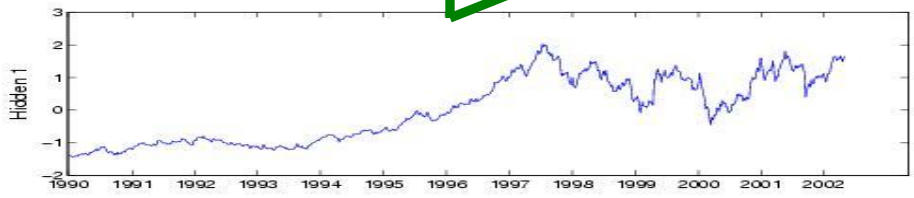


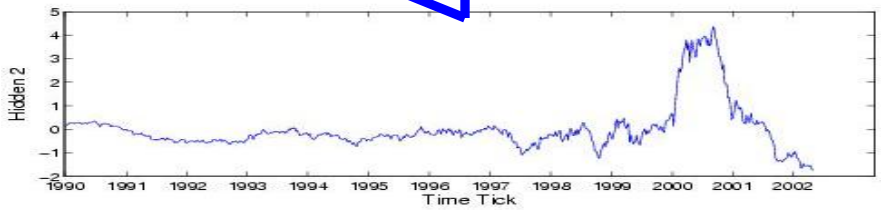


Caterpillar

Intel



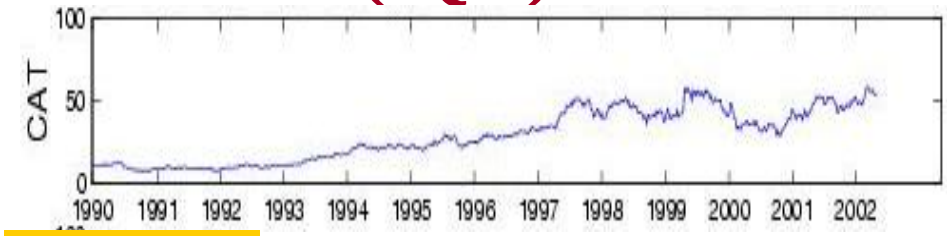


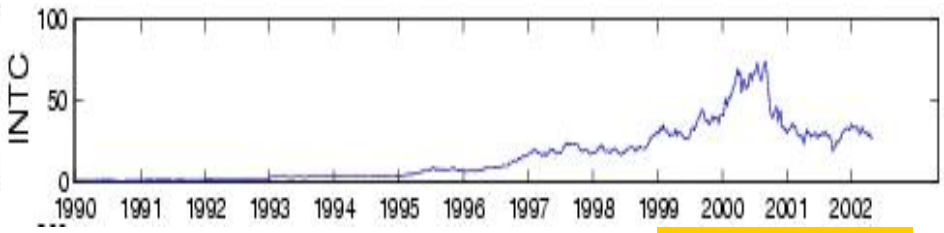


"Hidden variable 1"

"Hidden variable 2"

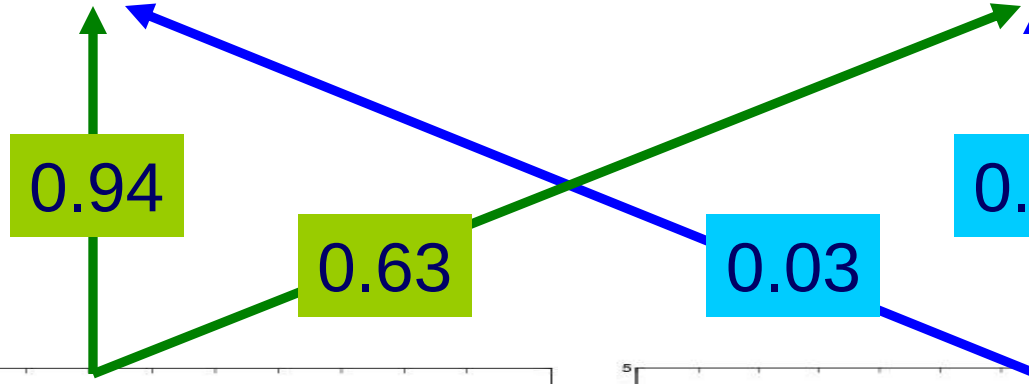
Motivation: (Q2) Find hidden variables

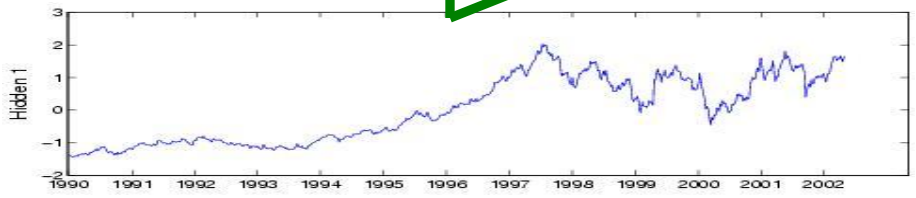


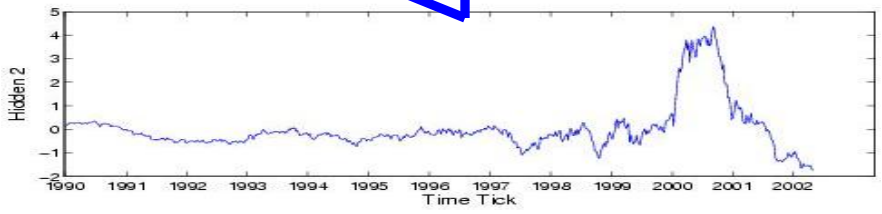


Caterpillar

Intel







"General trend"

"Internet bubble"

Outline

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

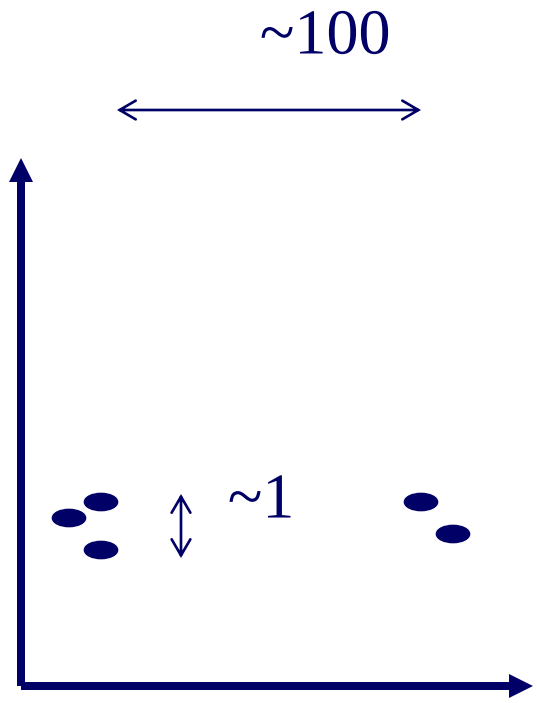
MDS / FastMap

- but, what if we have NO points to start with?

  (eg. Time-warping distance)
- A: Multi-dimensional Scaling (MDS) ; FastMap

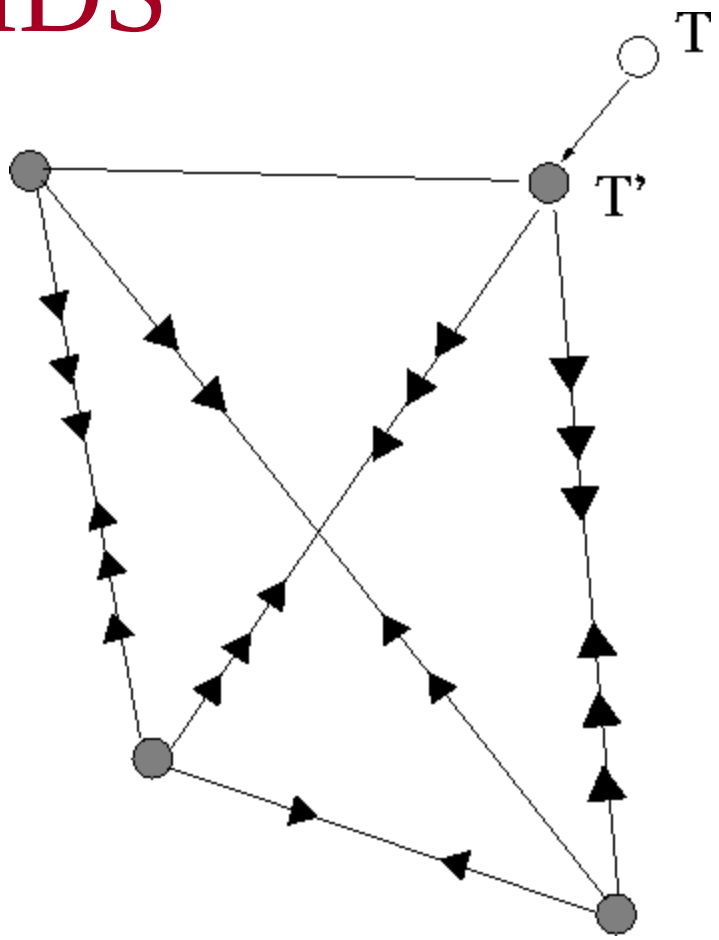
MDS/FastMap

	O1	O2	O3	O4	O5
O1	0	1	1	100	100
O2	1	0	1	100	100
O3	1	1	0	100	100
O4	100	100	100	0	1
O5	100	100	100	1	0



MDS

Multi Dimensional Scaling

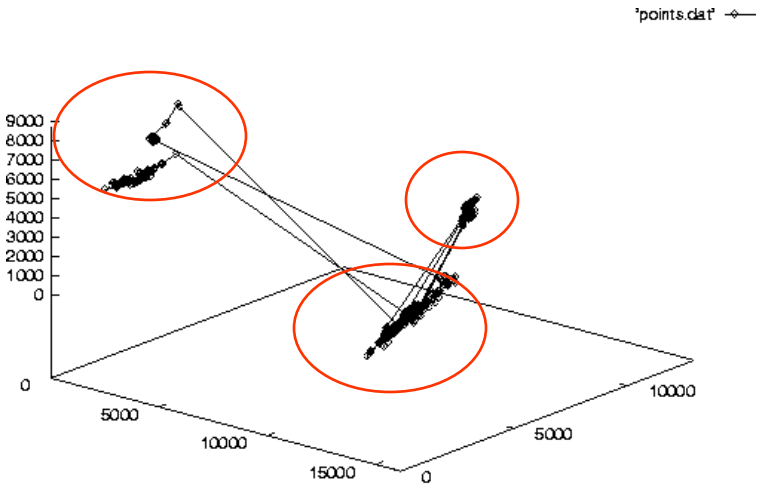


FastMap

- Multi-dimensional scaling (MDS) can do that, but in O(N**2) time
- FastMap [Faloutsos+95] takes O(N) time

FastMap: Application

VideoTrails [Kobla+97]



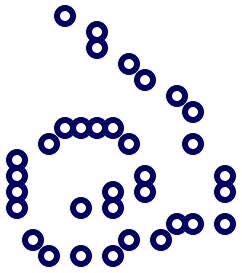
scene-cut detection (about 10% errors)

Outline

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

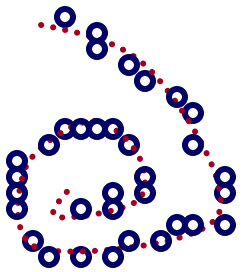
Variations

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



Variations

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



Outline

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

Conclusions - Practitioner's guide

Similarity search in time sequences

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

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

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

Books

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

References

- Agrawal, R., K.-I. Lin, et al. (Sept. 1995). Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases. Proc. of VLDB, Zurich, Switzerland.
- Babu, S. and J. Widom (2001). "Continuous Queries over Data Streams." SIGMOD Record 30(3): 109-120.
- Breunig, M. M., H.-P. Kriegel, et al. (2000). LOF: Identifying Density-Based Local Outliers. SIGMOD Conference, Dallas, TX.
- Berry, Michael: http://www.cs.utk.edu/~lsi/

References

- Ciaccia, P., M. Patella, et al. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. VLDB.
- 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.
- Gehrke, J. E., F. Korn, et al. (May 2001). On Computing Correlated Aggregates Over Continual Data Streams. ACM Sigmod, Santa Barbara, California.

References

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

References

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

References

- Keogh, E. J., K. Chakrabarti, et al. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. SIGMOD Conference, Santa Barbara, CA.
- Kobla, V., D. S. Doermann, et al. (Nov. 1997). VideoTrails: Representing and Visualizing Structure in Video Sequences. ACM Multimedia 97, Seattle, WA.

References

- Oppenheim, I. J., A. Jain, et al. (March 2002). A MEMS Ultrasonic Transducer for Resident Monitoring of Steel Structures. SPIE Smart Structures Conference SS05, San Diego.
- Papadimitriou, C. H., P. Raghavan, et al. (1998). Latent Semantic Indexing: A Probabilistic Analysis. PODS, Seattle, WA.
- Rabiner, L. and B.-H. Juang (1993). Fundamentals of Speech Recognition, Prentice Hall.

References

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

References

- Dennis Shasha and Yunyue Zhu *High Performance Discovery in Time Series: Techniques and Case Studies* Springer 2004
- Yunyue Zhu, Dennis Shasha ``*StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time*'' VLDB, August, 2002. pp. 358-369.
- Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. *The Design of an Acquisitional Query Processor for Sensor Networks.* SIGMOD, June 2003, San Diego, CA.

References

- Lawrence Saul & Sam Roweis. *An Introduction to Locally Linear Embedding* (draft)
- Sam Roweis & Lawrence Saul. *Nonlinear dimensionality reduction by locally linear embedding*. Science, v.290 no.5500 , Dec.22, 2000. pp.2323--2326.
- B. Shaw and T. Jebara. "*Minimum Volume Embedding*" . Artificial Intelligence and Statistics, AISTATS, March 2007.

References

- Josh Tenenbaum, Vin de Silva and John Langford. *A Global Geometric Framework for Nonlinear dimensionality Reduction*. Science 290, pp. 2319-2323, 2000.

Part 2: DSP (Digital Signal Processing)

Outline

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

Outline

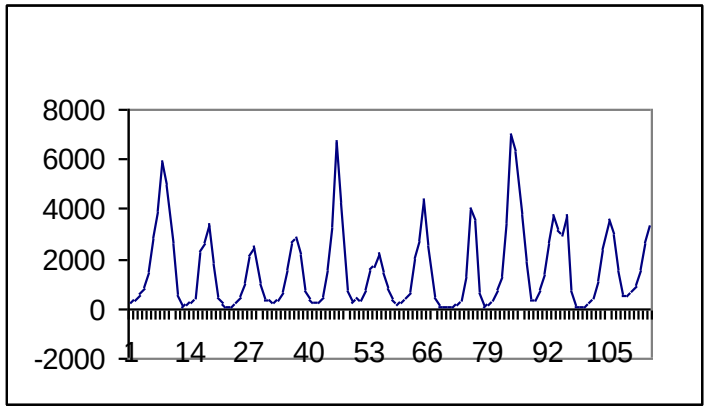
- DFT
  - Definition of DFT and properties
  - how to read the DFT spectrum
- DWT
  - Definition of DWT and properties
  - how to read the DWT scalogram

Introduction - Problem#1

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

Find: patterns and/or compress

count

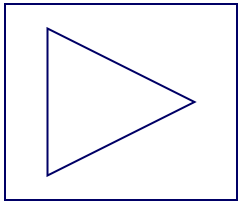


lynx caught per year
(packets per day;
automobiles per hour)

year

What does DFT do?

A: highlights the periodicities



Skip

DFT: definition

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

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

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

inverse DFT

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

DFT: definition

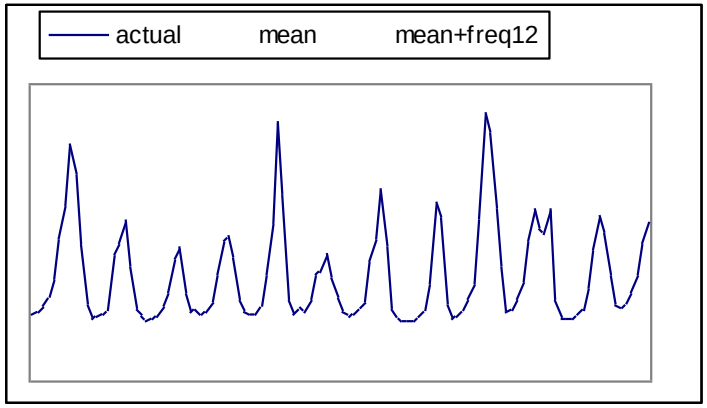
- **Good** news: Available in **all** symbolic math packages, eg., in ‘mathematica’

```
x = [1,2,1,2];
X = Fourier[x];
Plot[ Abs[X] ];
```

DFT: Amplitude spectrum

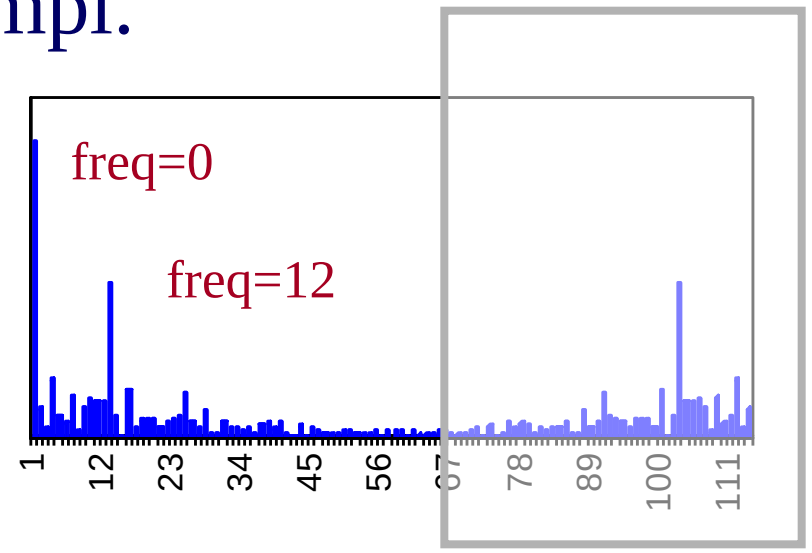
Amplitude: $A_f^{\ 2} = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$

count



year

Ampl.



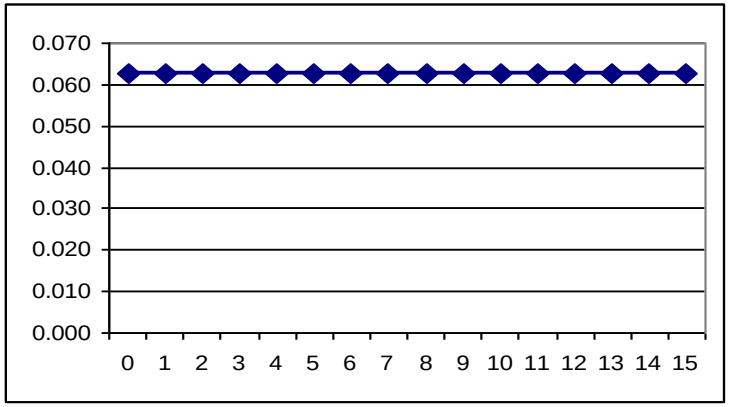
Freq.

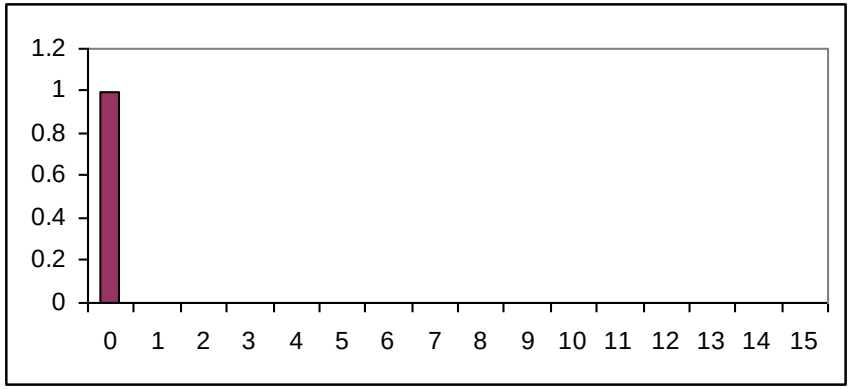
Skip

DFT: examples

flat

Amplitude





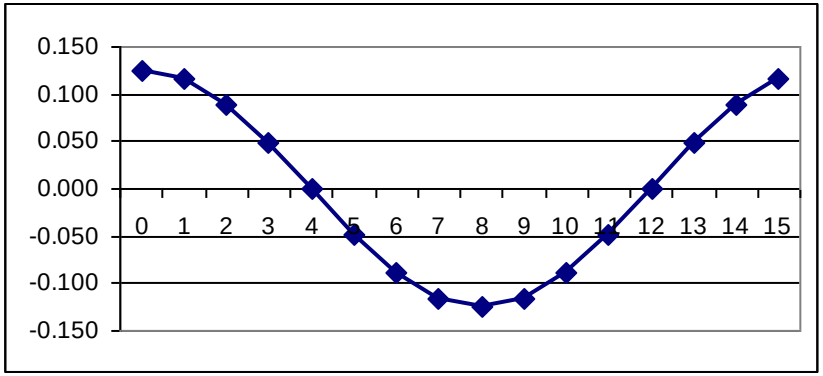
time

freq

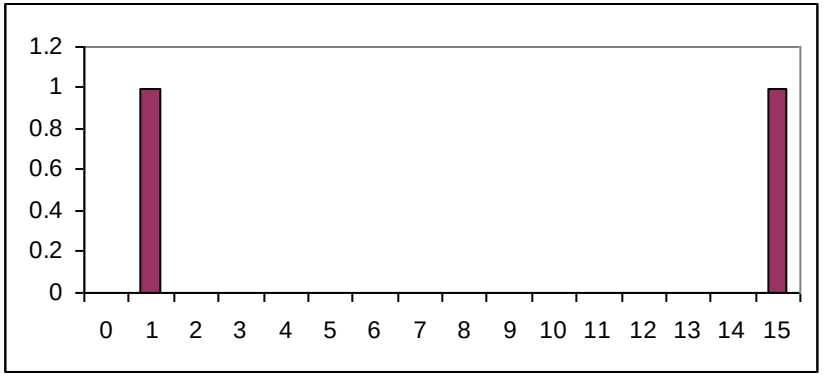
Skip

DFT: examples

Low frequency sinusoid



time

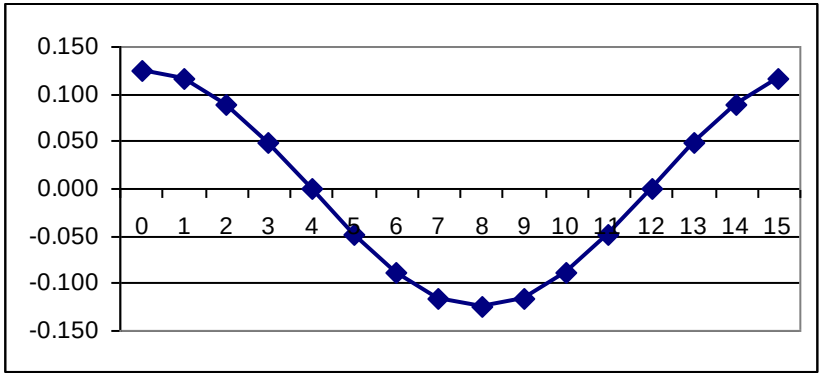


freq

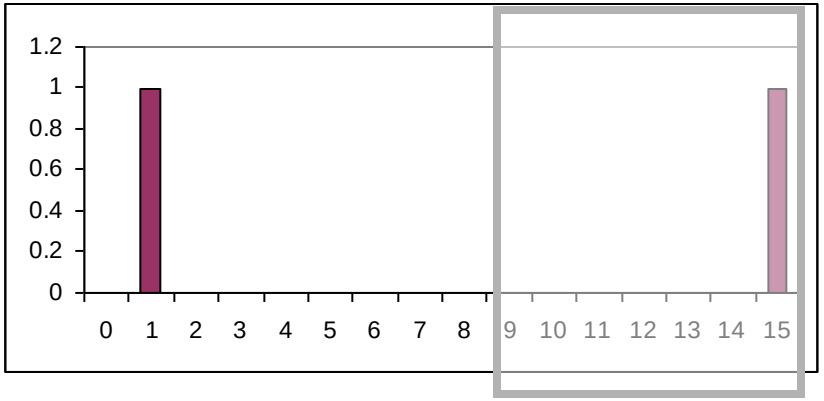
Skip

DFT: examples

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



time

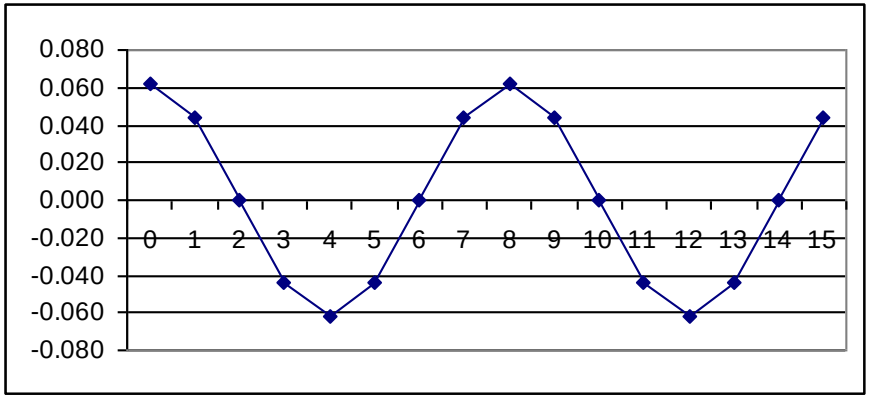


freq

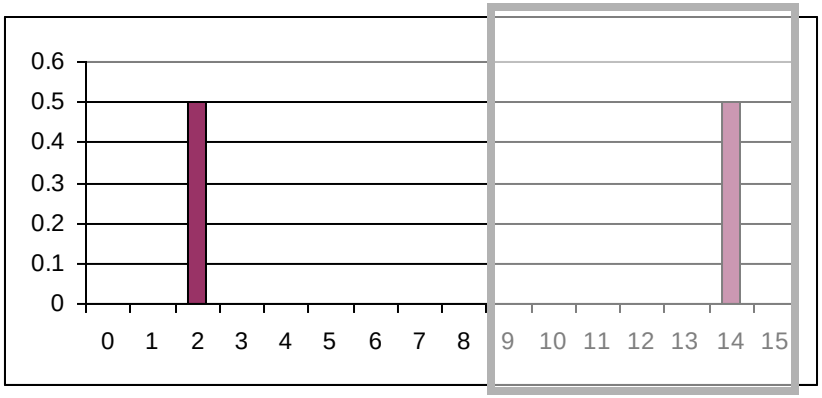
Skip

DFT: examples

- Higher freq. sinusoid



time

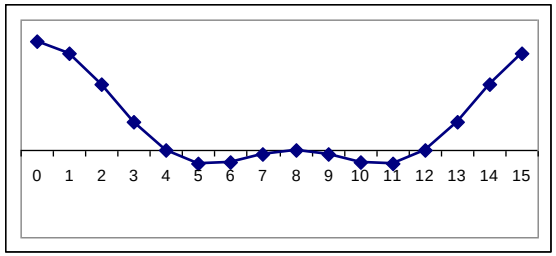


freq

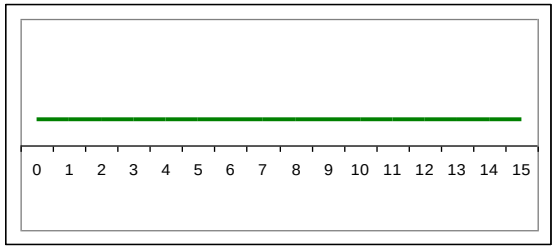
Skip

DFT: examples

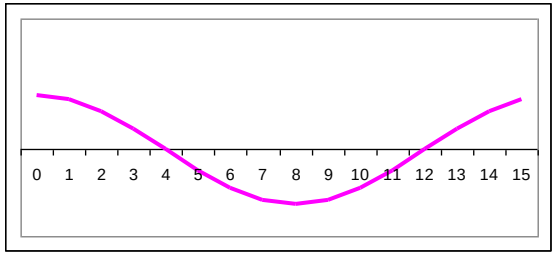
examples



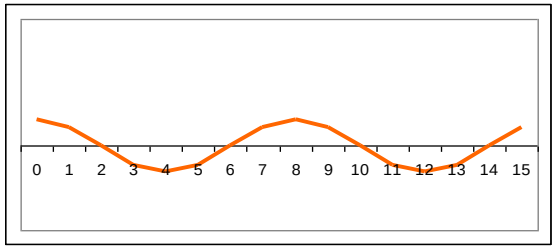
=



+



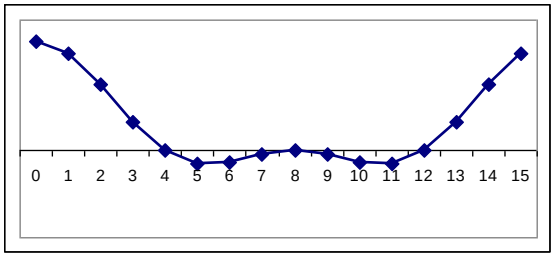
+

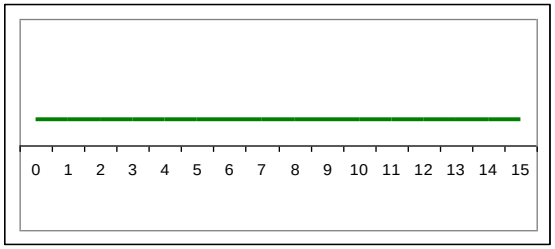


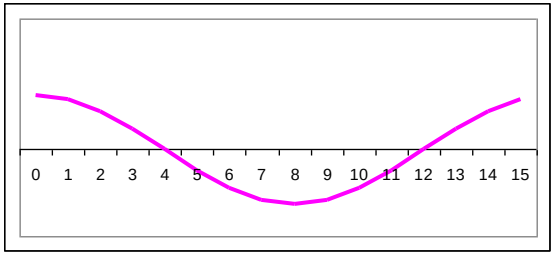
Skip

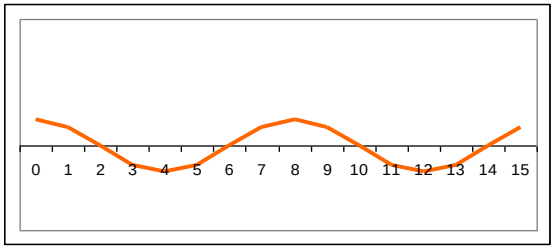
DFT: examples

examples

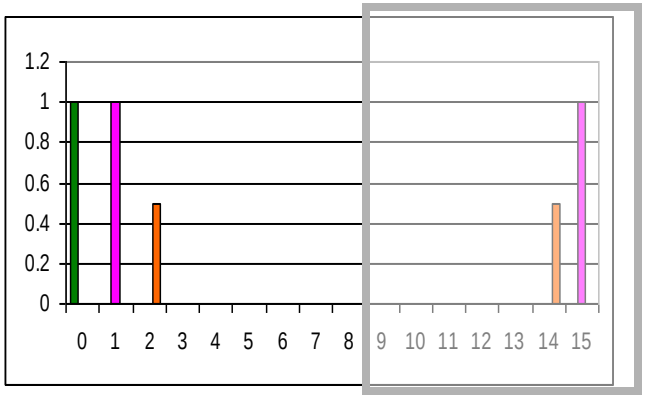




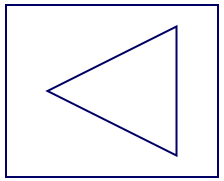




Ampl.



Freq.



Outline

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

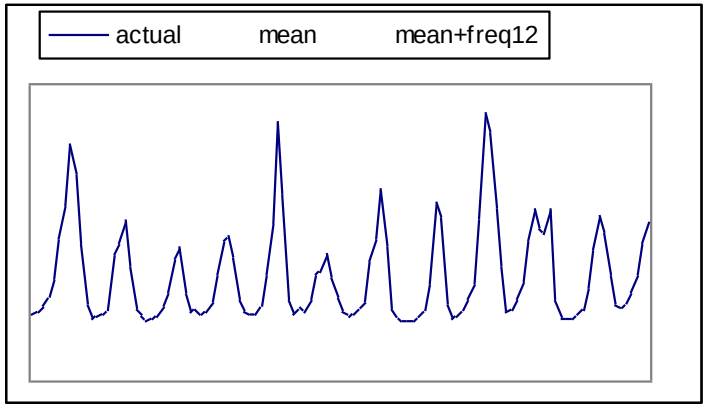
Outline

- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
    - Definition of DFT and properties
    - how to read the DFT spectrum
  - DWT

DFT: Amplitude spectrum

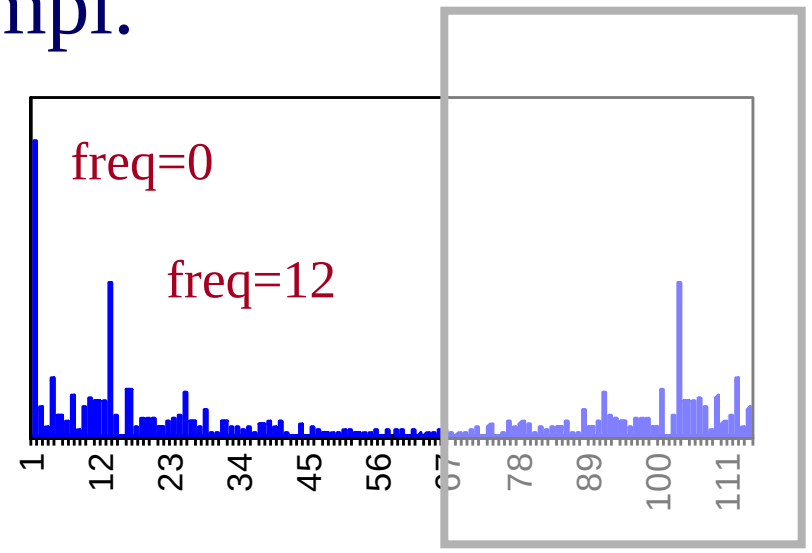
Amplitude: $A_f^{\ 2} = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$

count



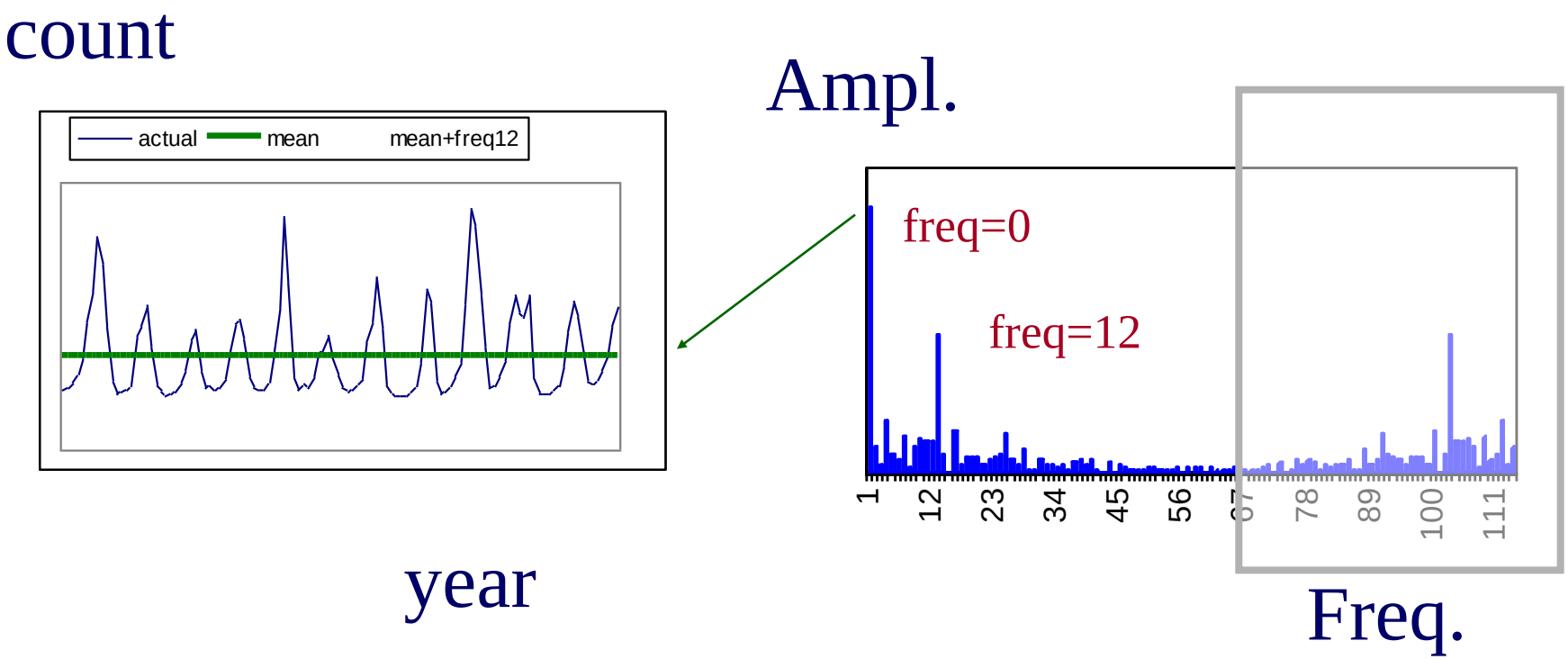
year

Ampl.

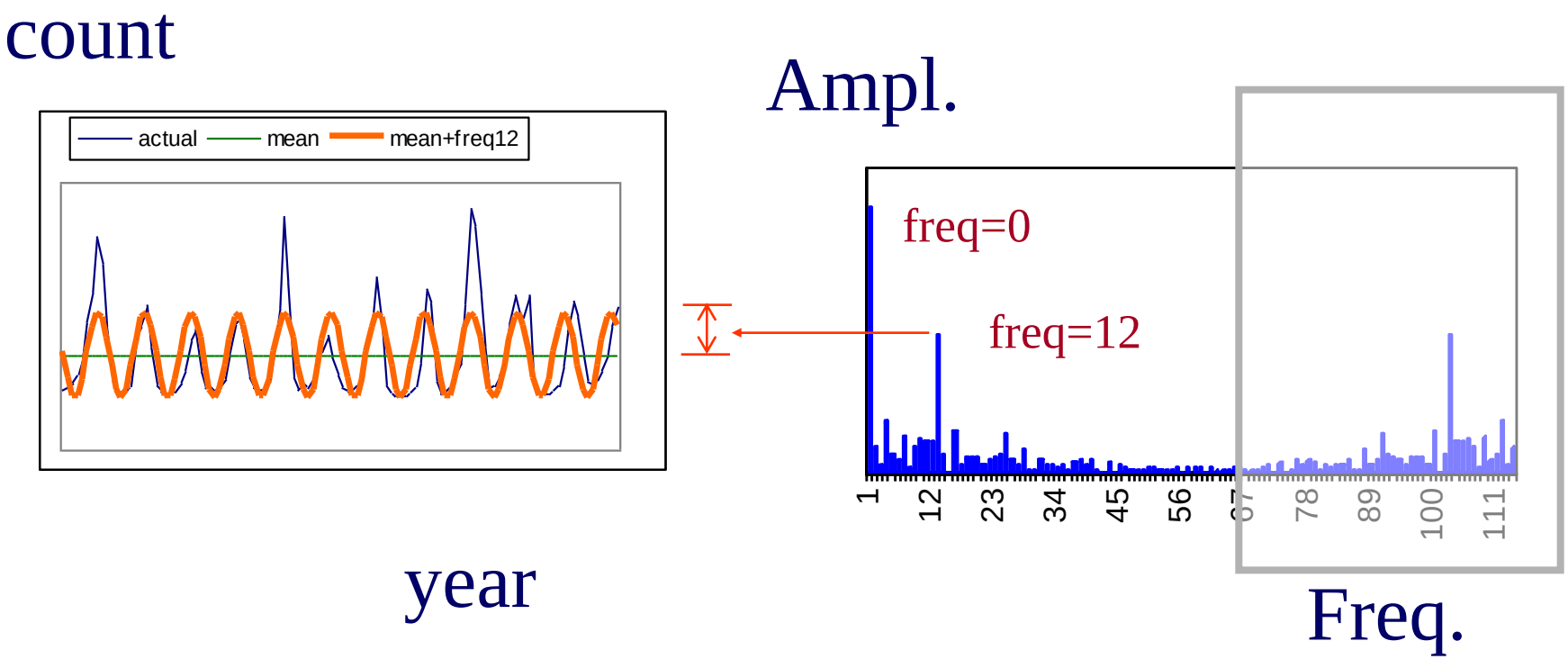


Freq.

DFT: Amplitude spectrum

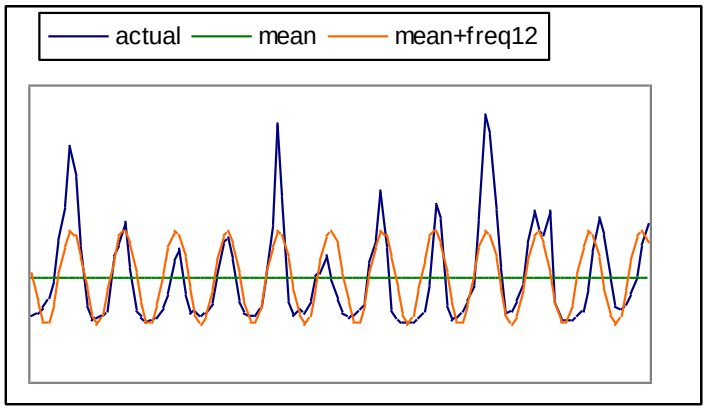


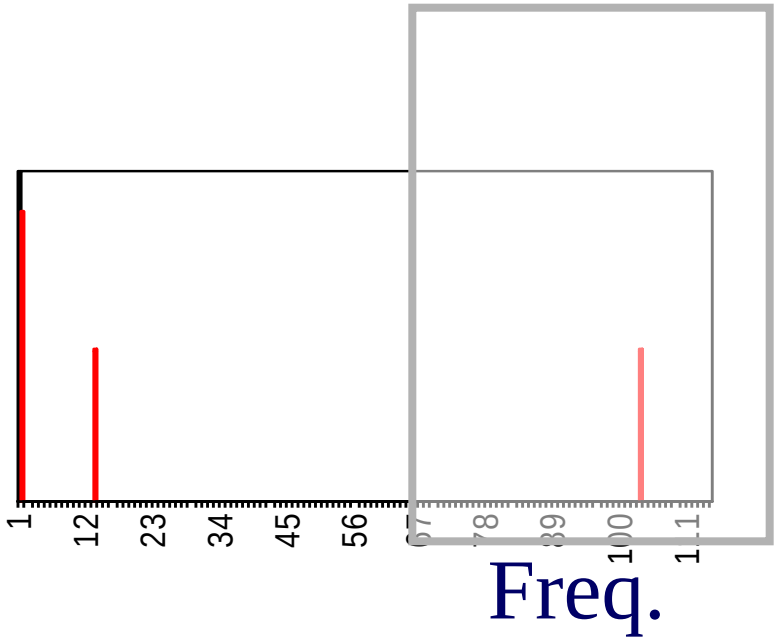
DFT: Amplitude spectrum



DFT: Amplitude spectrum

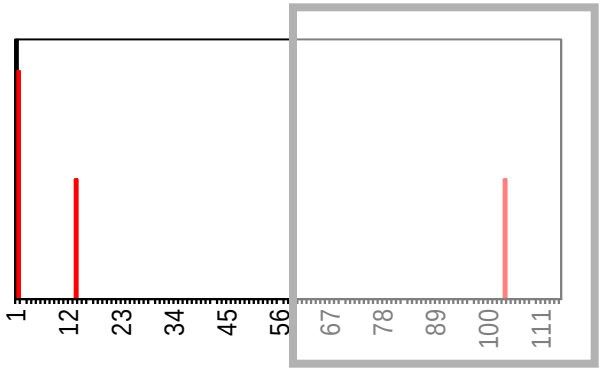
- excellent approximation, with only 2 frequencies!
- so what?





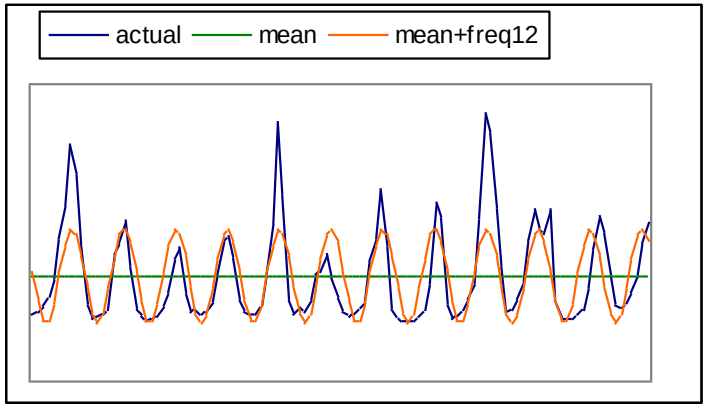
DFT: Amplitude spectrum

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



DFT: Amplitude spectrum

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



DFT - Conclusions

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

Outline

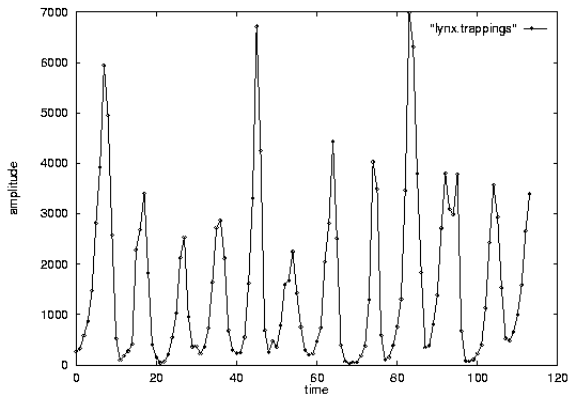
- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
  - DWT
    - Definition of DWT and properties
    - how to read the DWT scalogram

Problem #1:

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

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

count



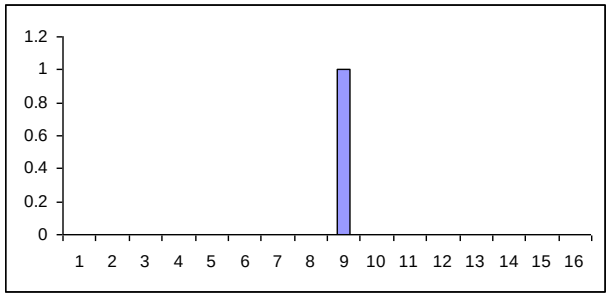
year

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

Wavelets - DWT

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

value

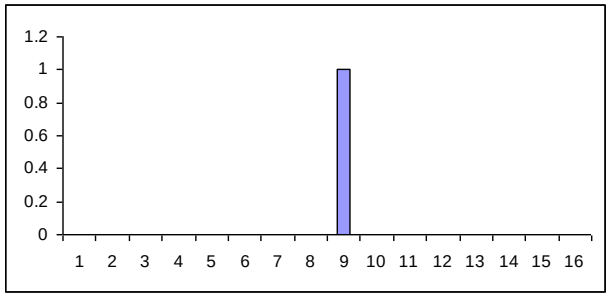


time

Wavelets - DWT

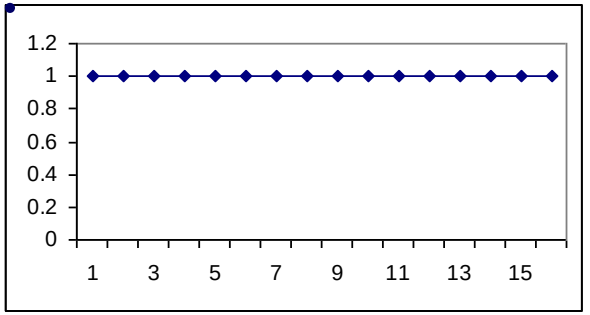
- DFT is great - but, how about compressing a spike?
- A: Terrible - all DFT coefficients needed!

value



time

Ampl.

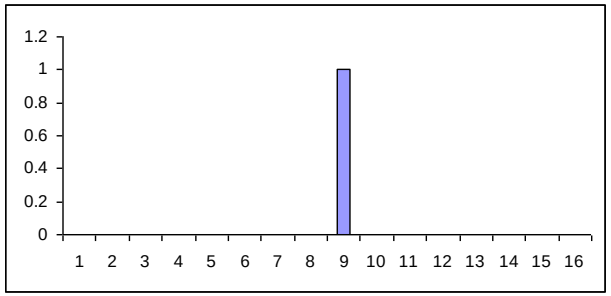


Freq.

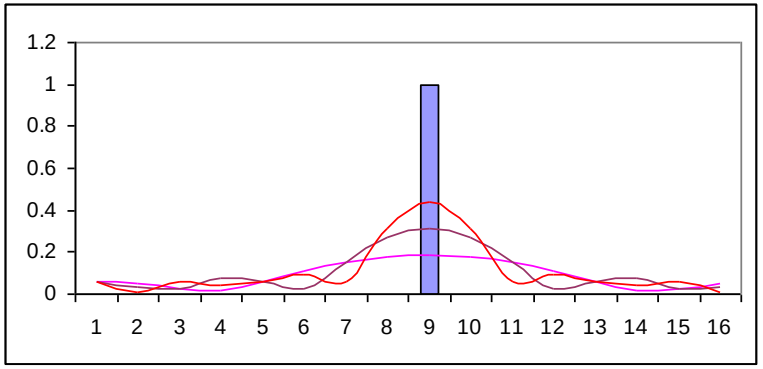
Wavelets - DWT

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

value



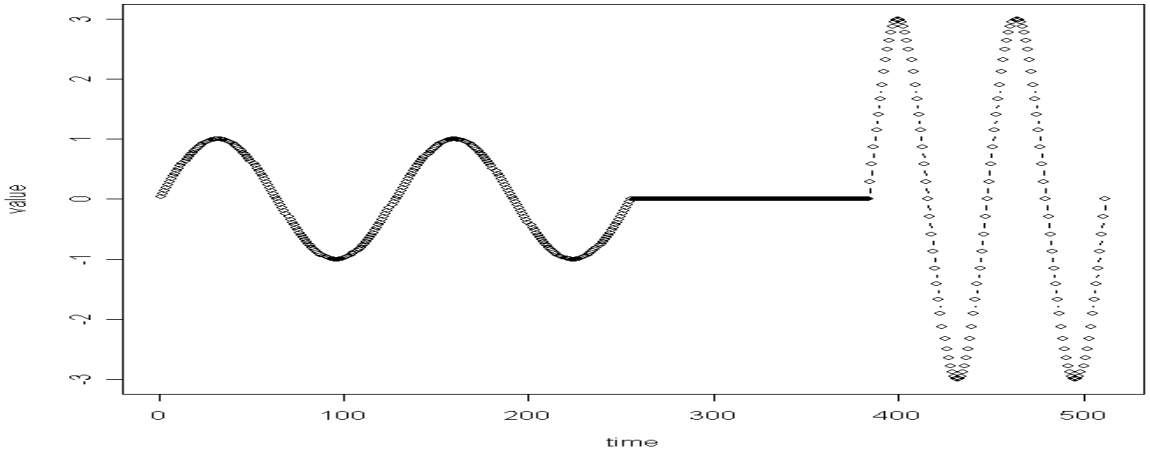
time



Wavelets - DWT

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

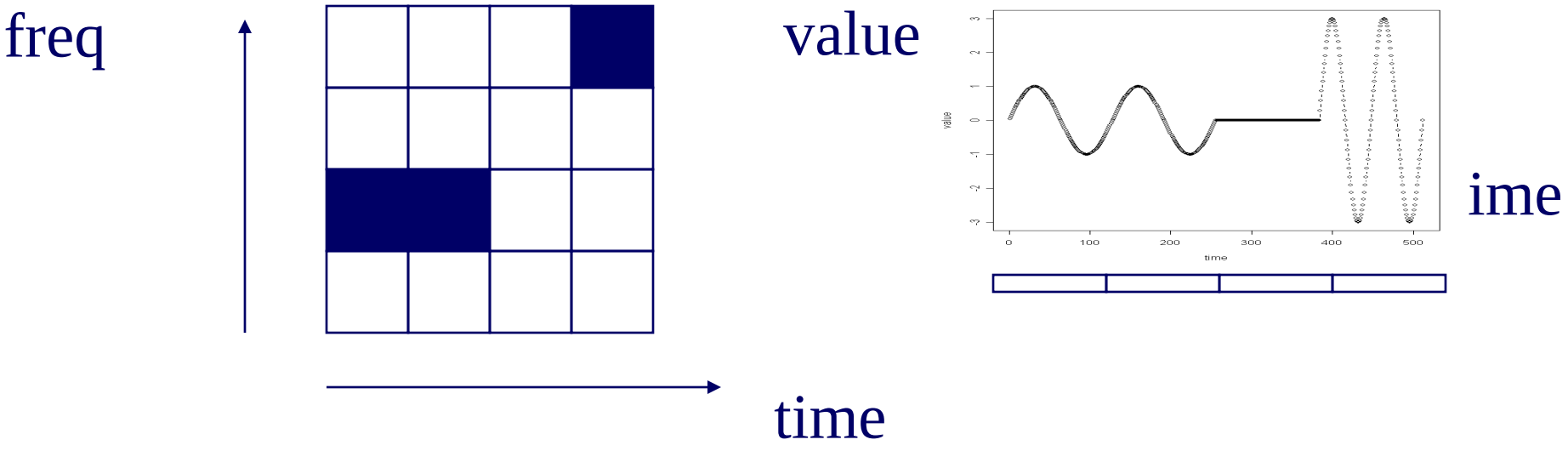
value



time

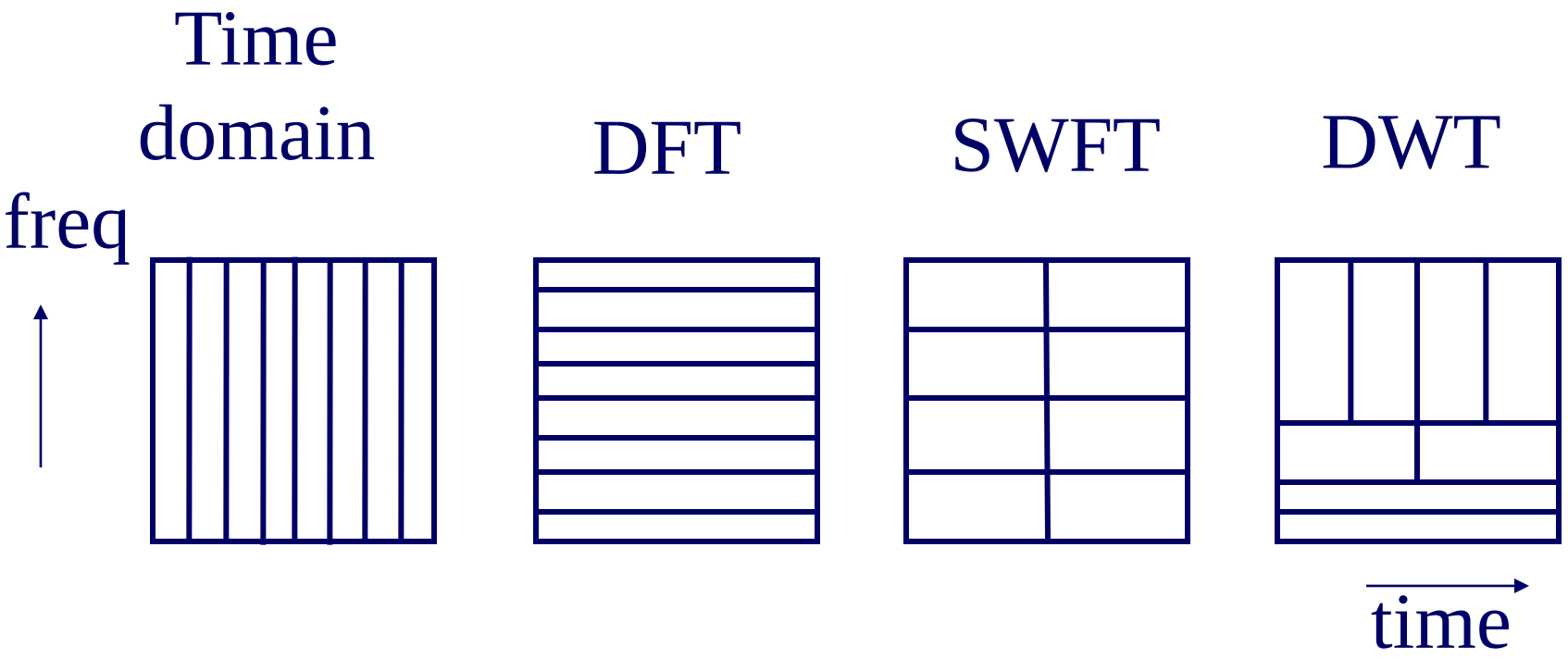
Wavelets - DWT

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



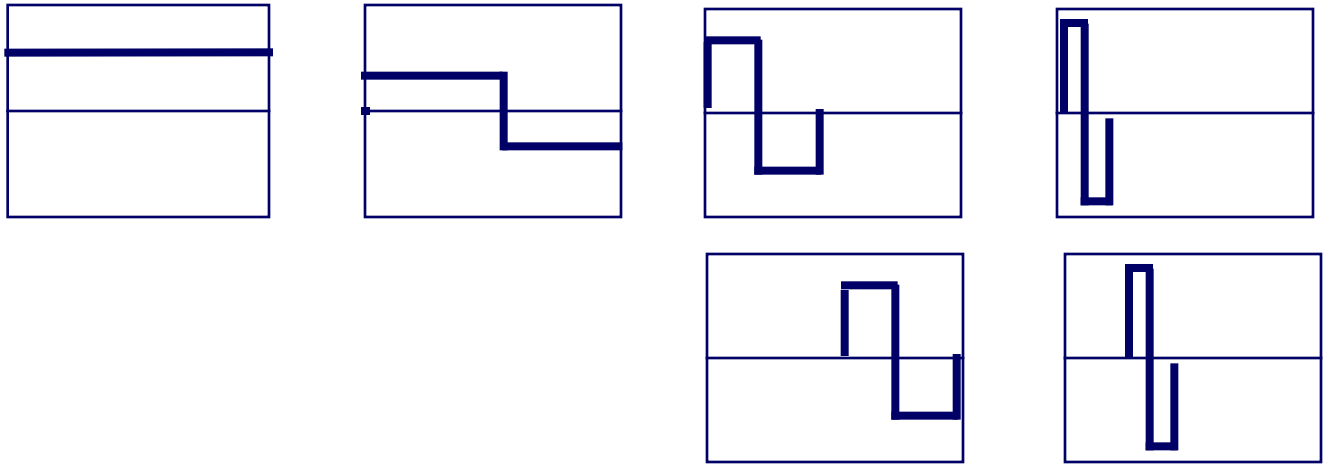
Wavelets - DWT

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



Haar Wavelets

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



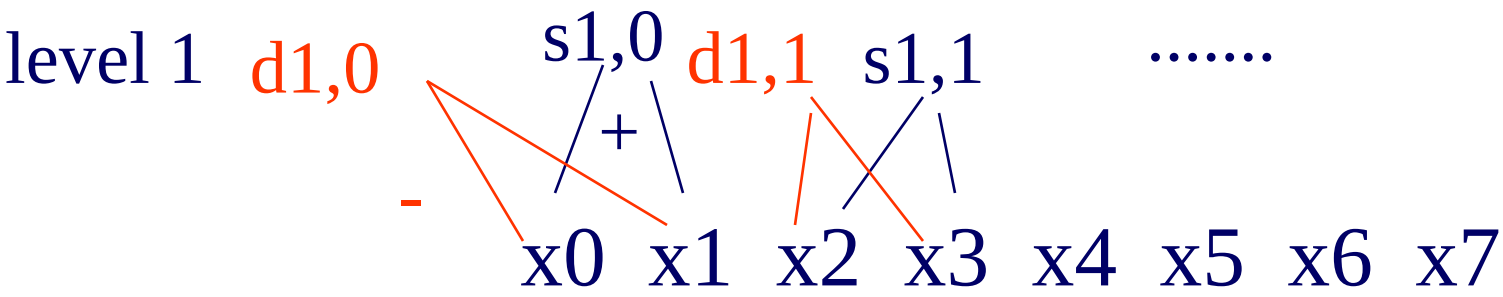
Skip

Wavelets - construction

x0 x1 x2 x3 x4 x5 x6 x7

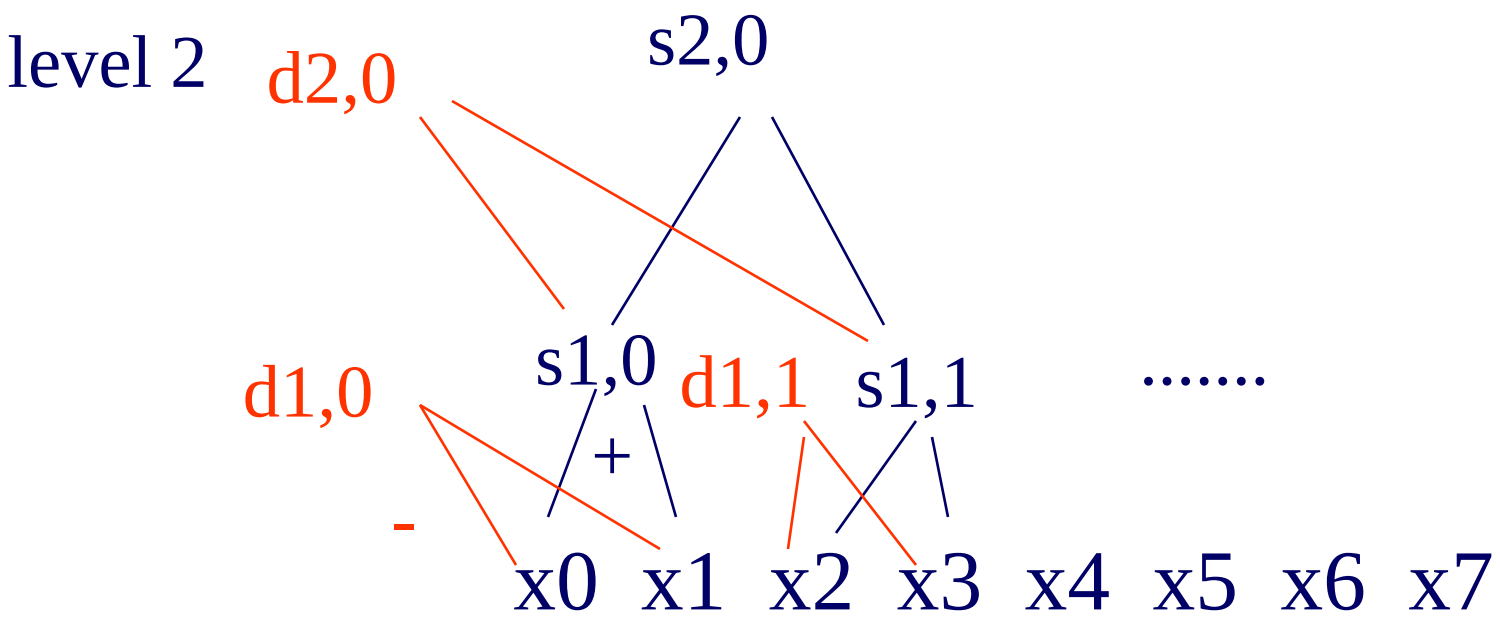
Skip

Wavelets - construction



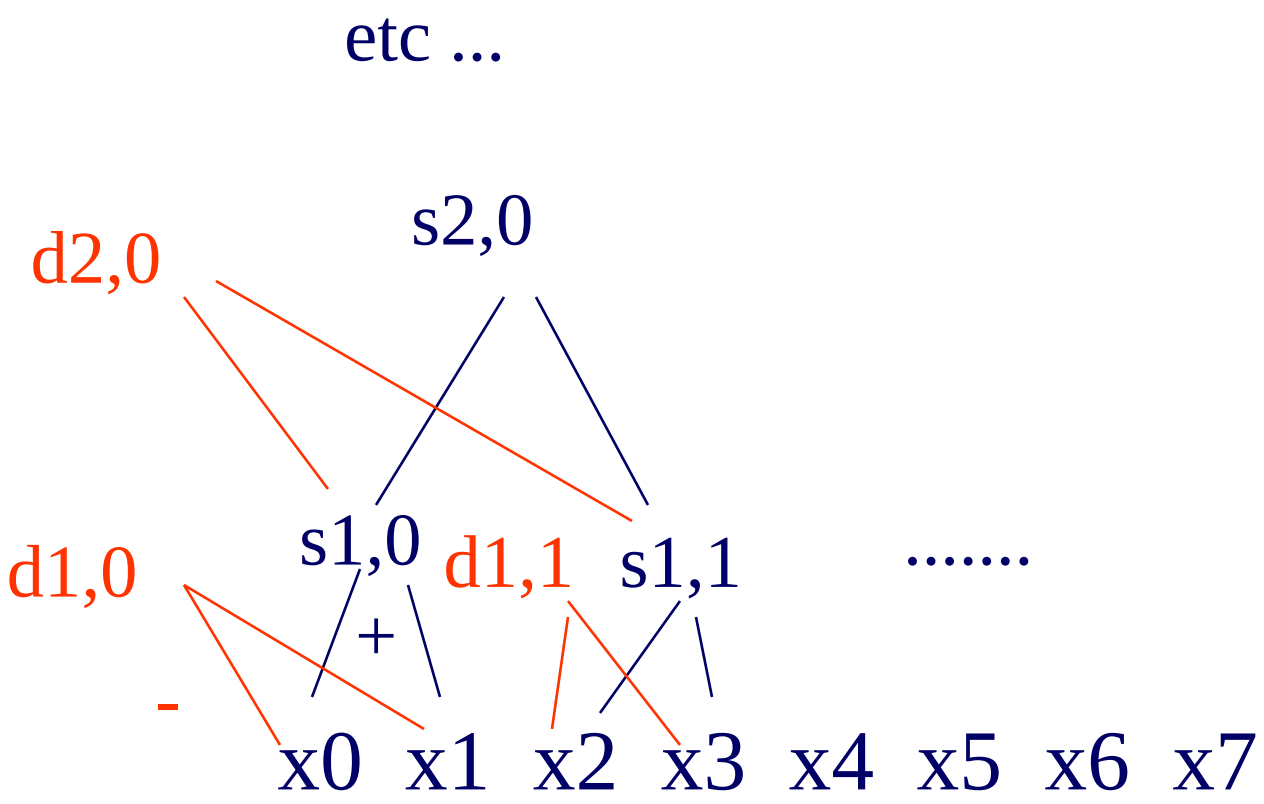
Skip

Wavelets - construction



Skip

Wavelets - construction

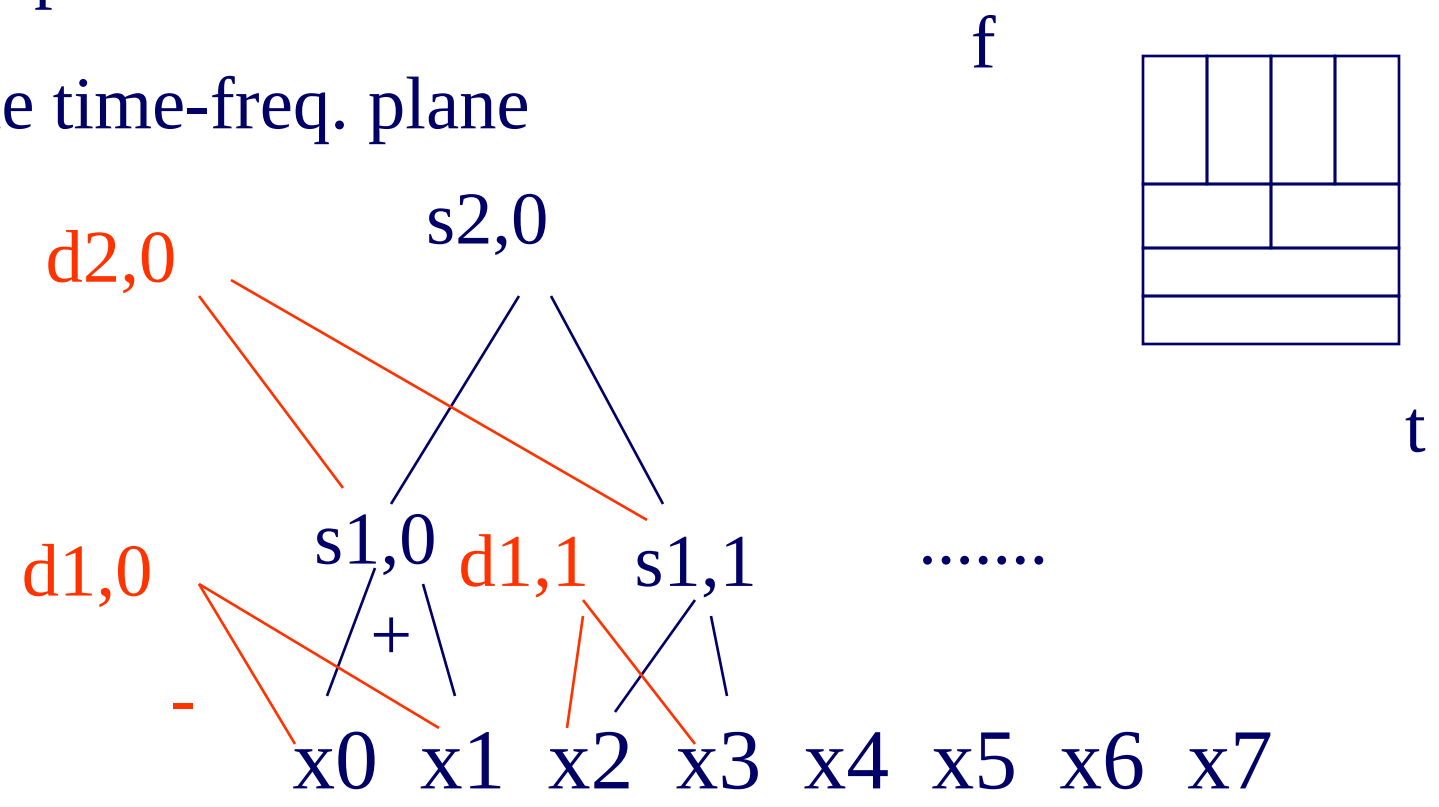


Skip

Wavelets - construction

Q: map each coefficient

on the time-freq. plane

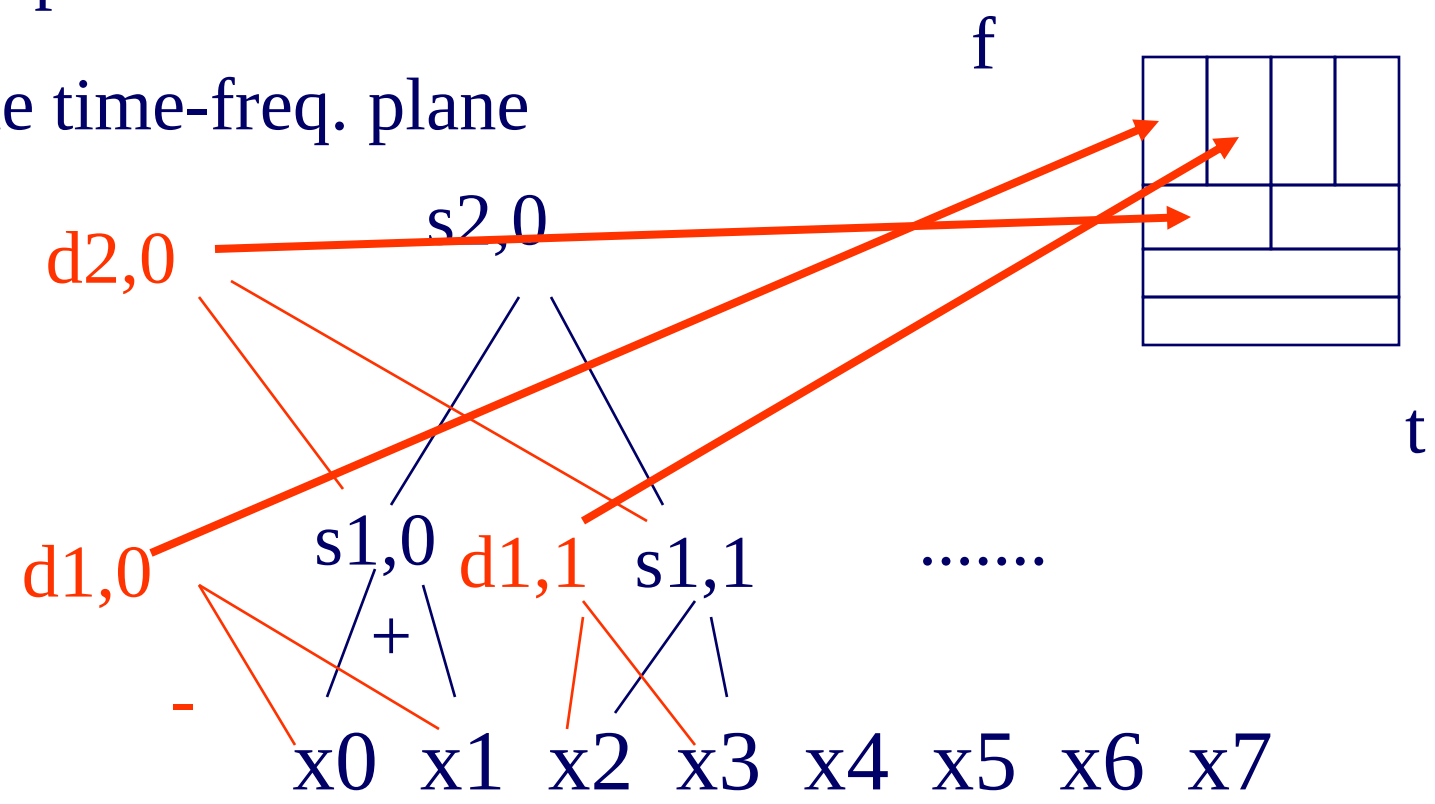


Skip

Wavelets - construction

Q: map each coefficient

on the time-freq. plane



Haar wavelets - code

```
#!/usr/bin/perl5
# expects a file with numbers
# and prints the dwt transform
# The number of time-ticks should be a power of 2
# USAGE
#    haar.pl <fname>

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

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

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

Wavelets - construction

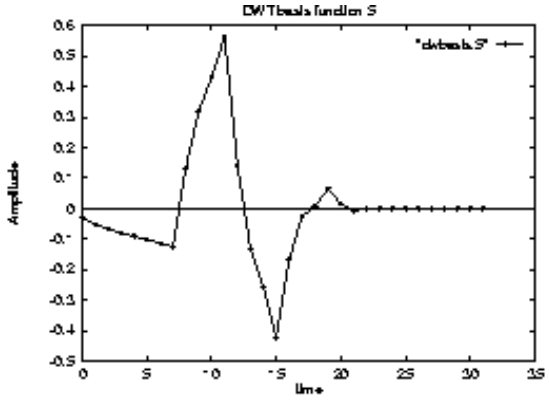
Observation1:

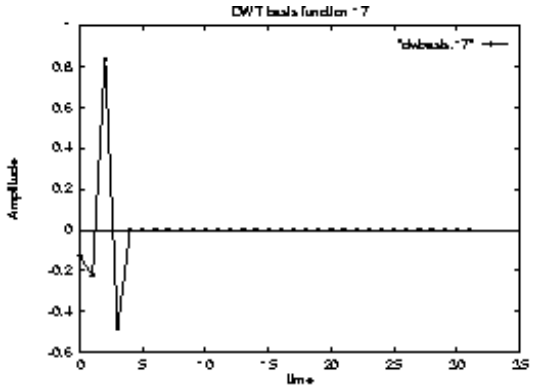
- '+' can be some weighted addition
- '-' is the corresponding weighted difference ('Quadrature mirror filters')

Observation2: unlike DFT/DCT,

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

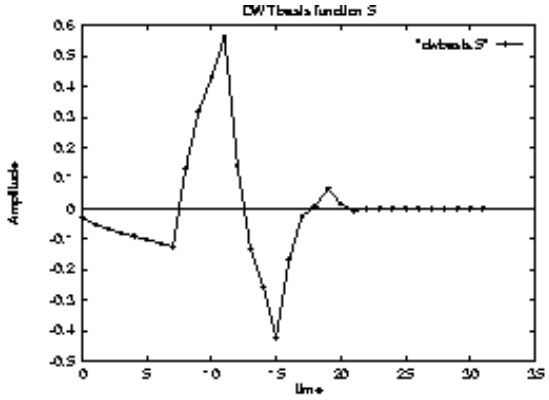
Wavelets - how do they look like?





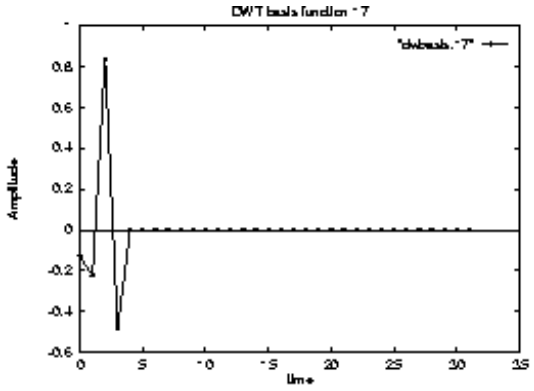
- E.g., Daubechies-4

Wavelets - how do they look like?

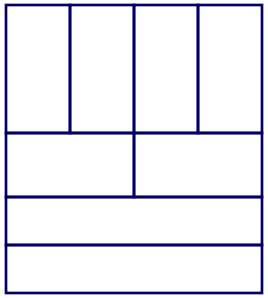


- E.g., Daubechies-4

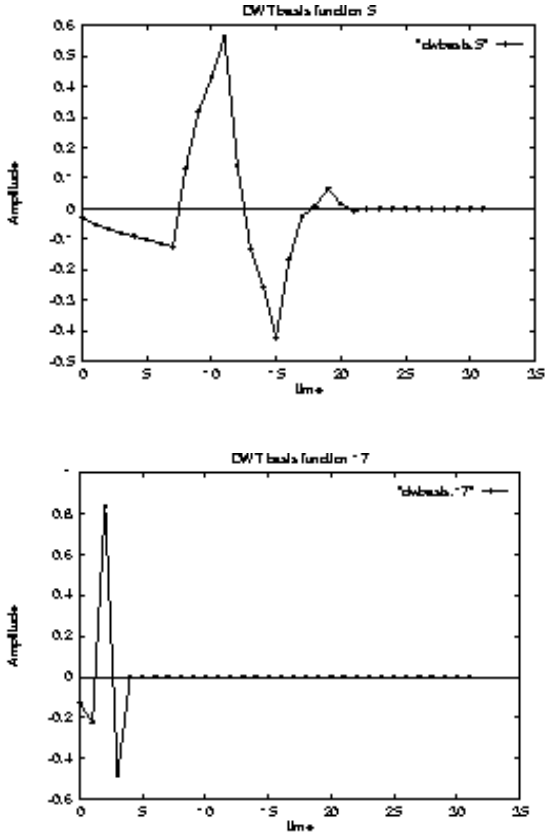
?



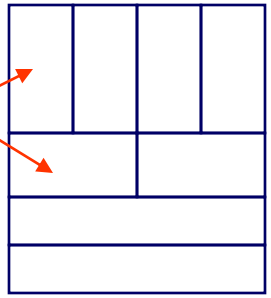
?



Wavelets - how do they look like?



- E.g., Daubechies-4

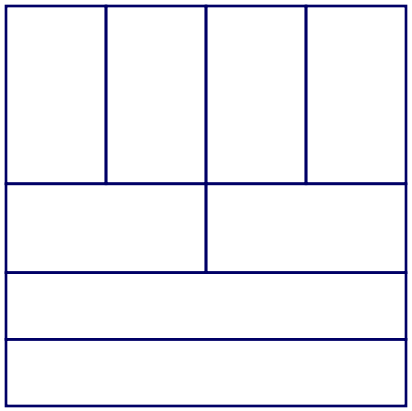


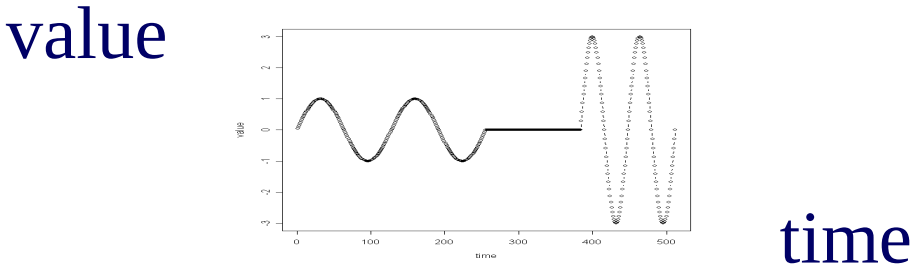
Outline

- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
  - DWT
    - Definition of DWT and properties
    - how to read the DWT scalogram

Wavelets - Drill#1:

- Q: baritone/silence/soprano - DWT?

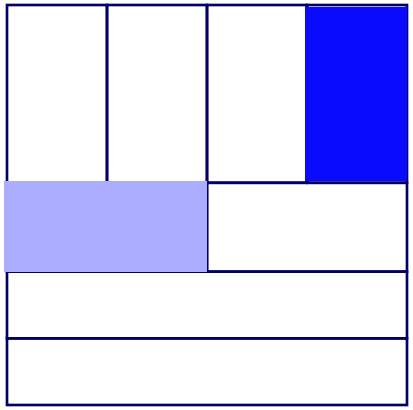




Wavelets - Drill#1:

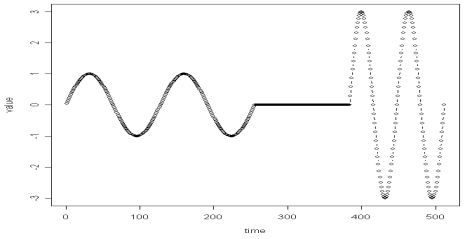
- Q: baritone/silence/soprano - DWT?

f



t

value

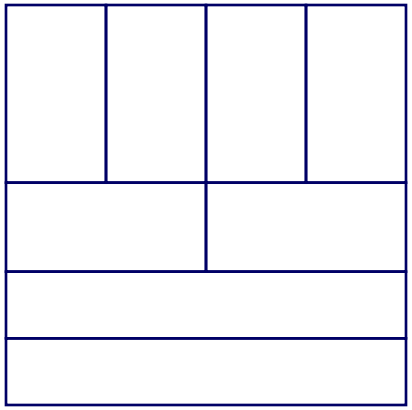


time

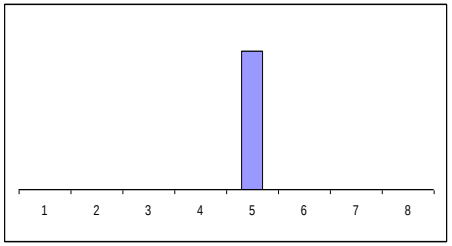
Wavelets - Drill#2:

- Q: spike - DWT?

f



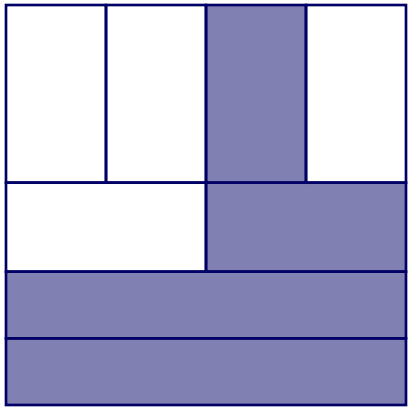
t



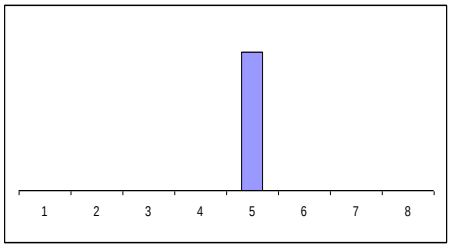
Wavelets - Drill#2:

- Q: spike - DWT?

f



t



0.00 0.00 **0.71** 0.00

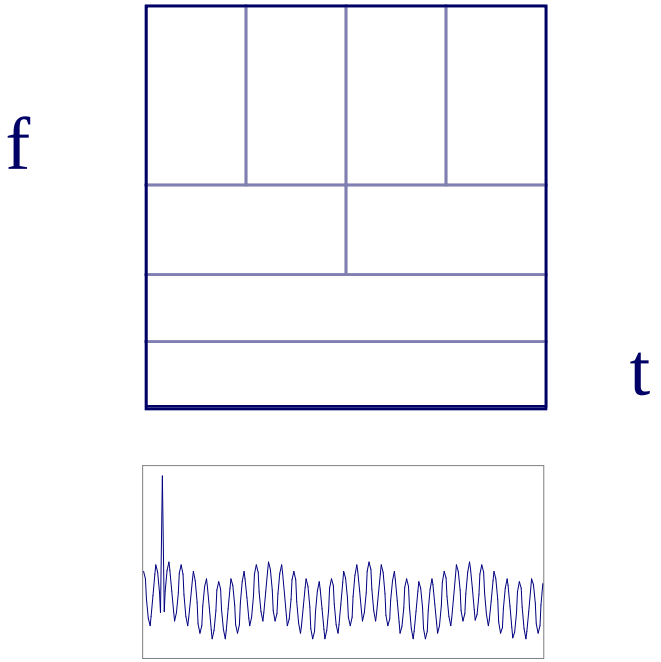
0.00 **0.50**

-0.35

0.35

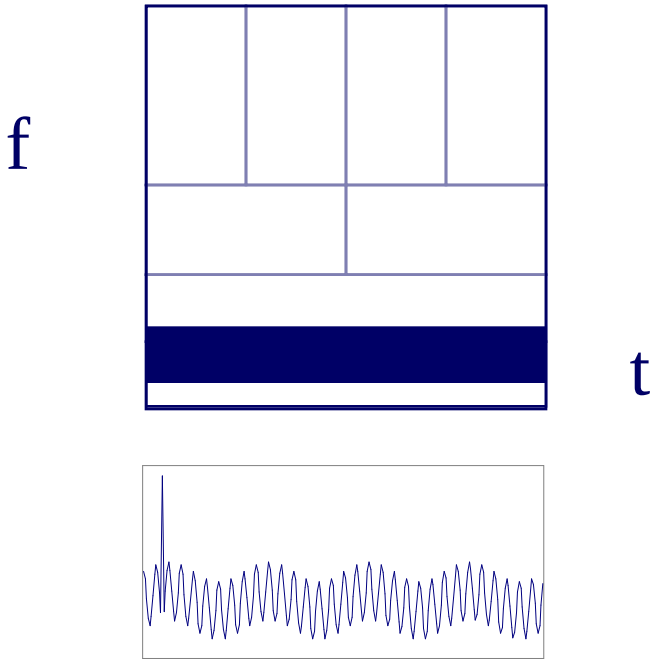
Wavelets - Drill#3:

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



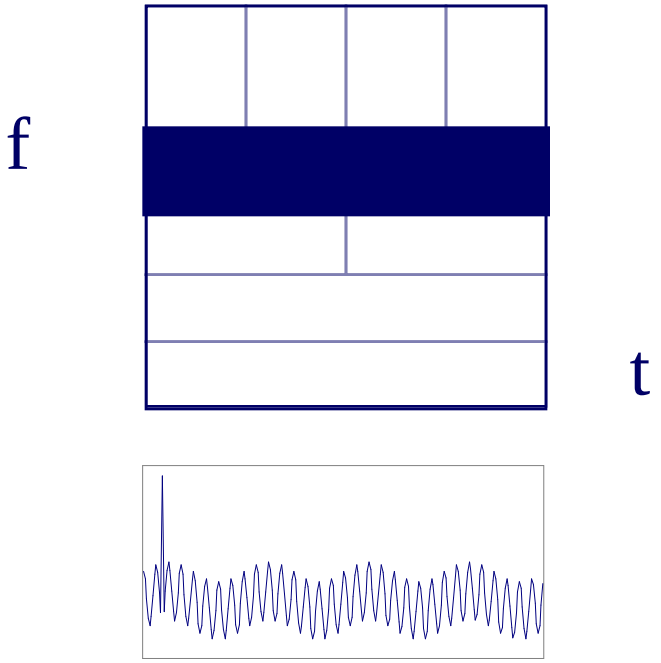
Wavelets - Drill#3:

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



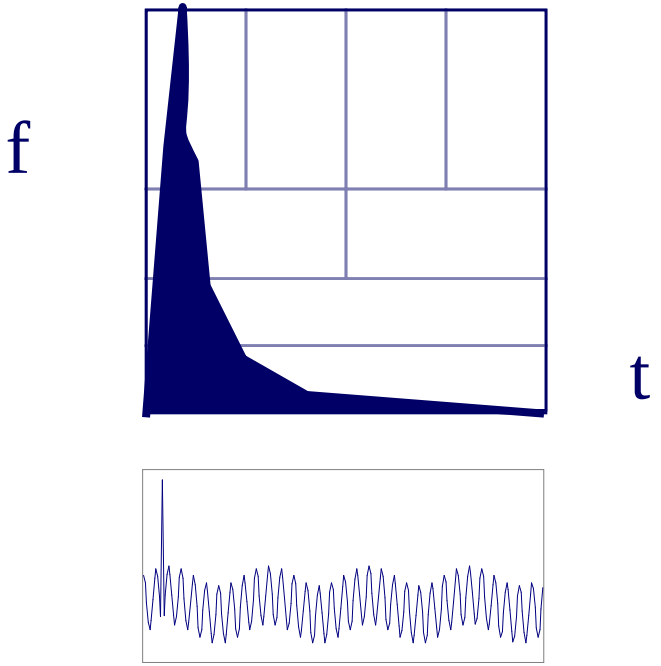
Wavelets - Drill#3:

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



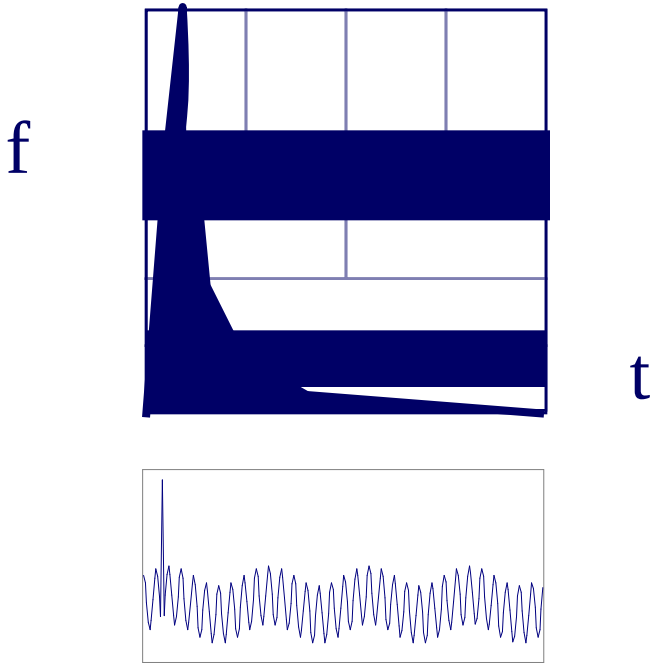
Wavelets - Drill#3:

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



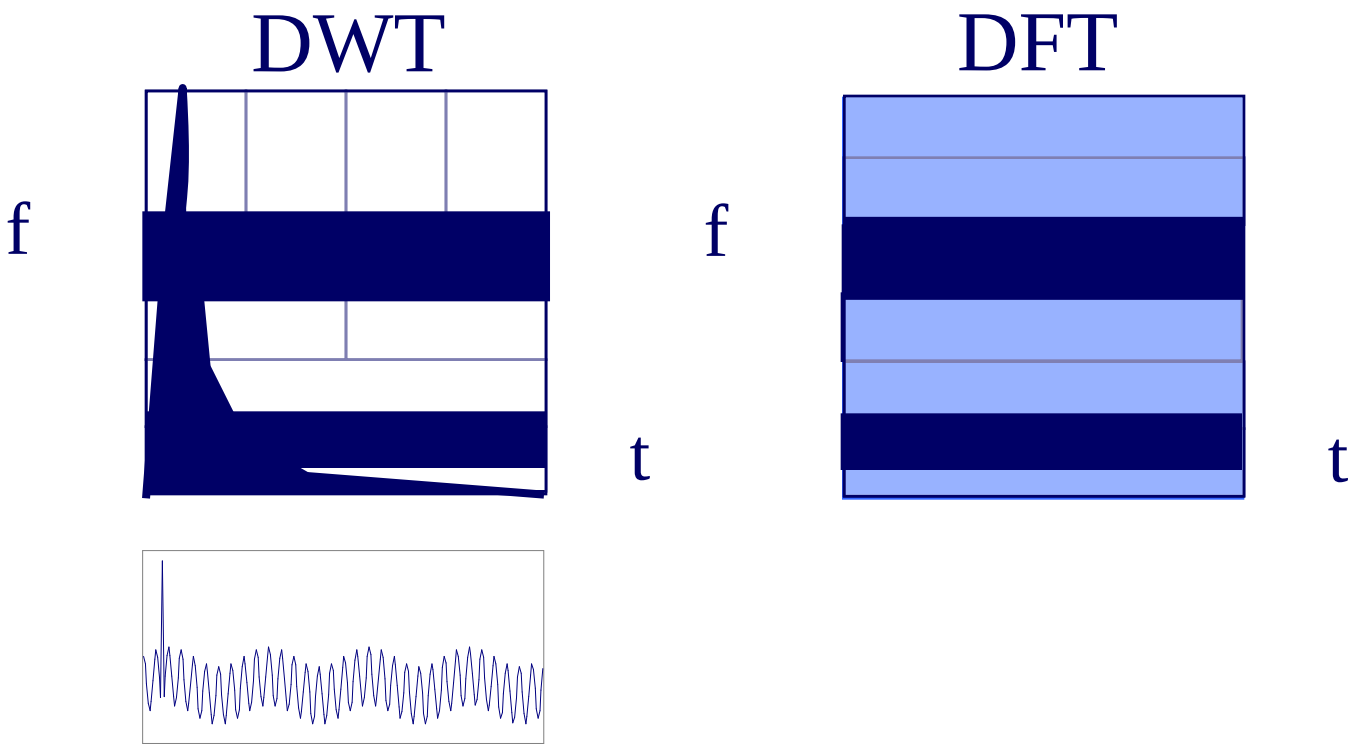
Wavelets - Drill#3:

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



Wavelets - Drill#3:

- Q: DFT?



Advantages of Wavelets

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

Overall Conclusions

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

Overall Conclusions

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

Resources - software and urls

- http://www.dsptutor.freeuk.com/jsanalyser/FFTSpectrumAnalyser.html : Nice java applets for FFT
- http://www.relisoft.com/freeware/freq.html voice frequency analyzer (needs microphone)

Resources: software and urls

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

Books

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

Additional Reading

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

Part 3:

Linear Forecasting

Outline

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

Forecasting

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

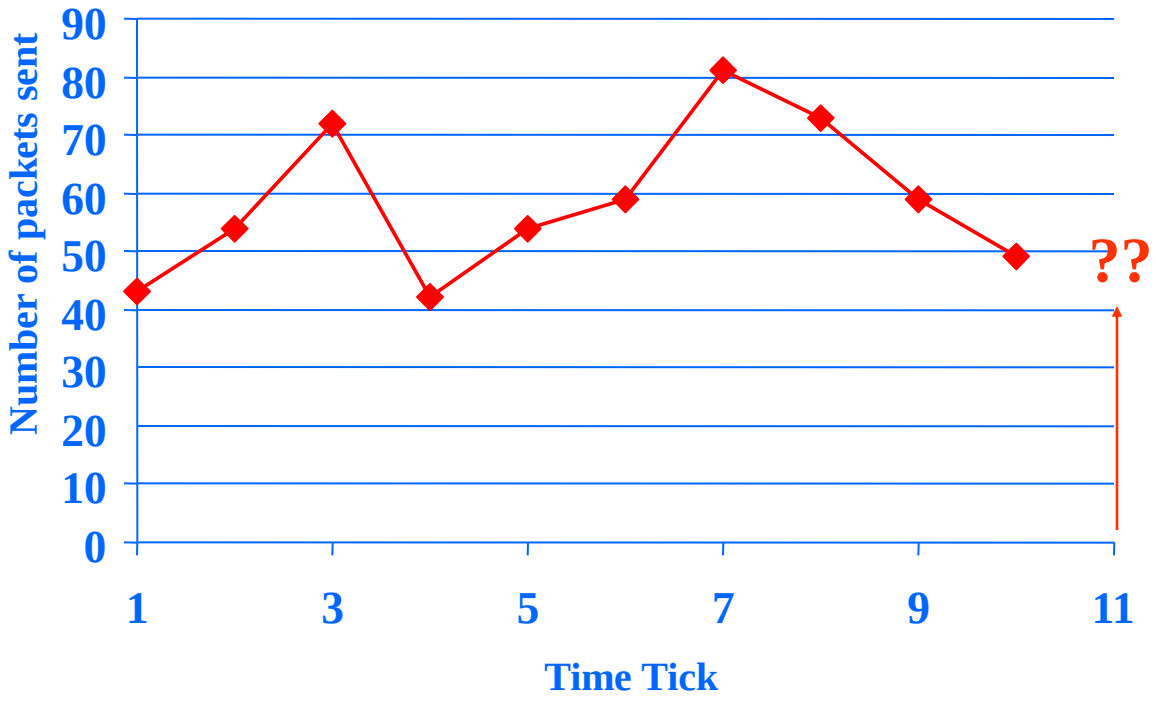
http://www.hfac.uh.edu/MediaFutures/thoughts.html

Outline

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

Problem#2: Forecast

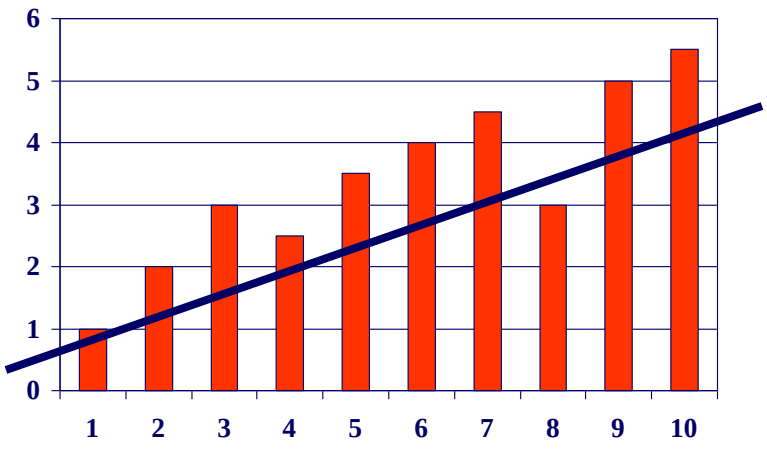
- Example: give x_{t-1}, x_{t-2}, …, forecast x_t



Forecasting: Preprocessing

MANUALLY:

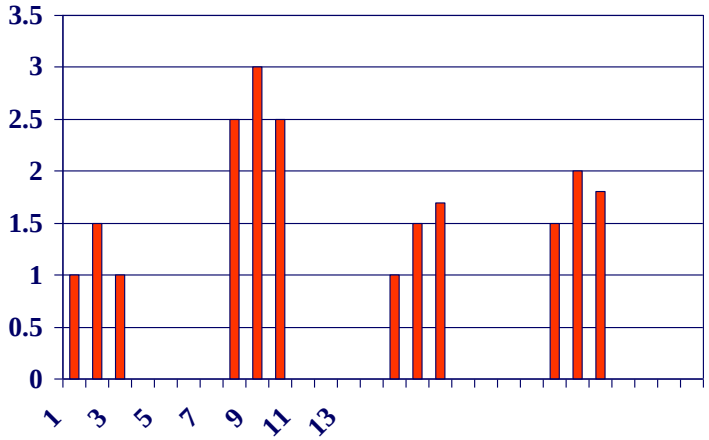
remove trends



time

spot periodicities

7 days



time

Problem#2: Forecast

- Solution: try to express

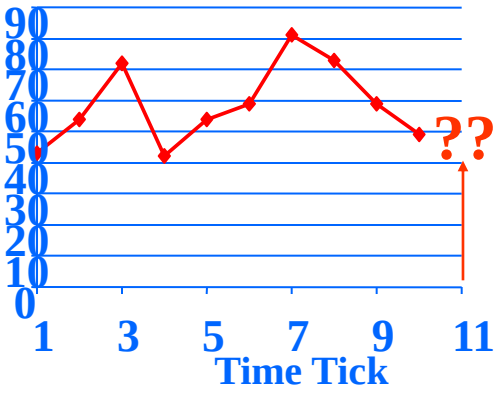
  x_t

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

  (up to a window of w)

Formally:

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



(Problem: Back-cast; interpolate)

- Solution - interpolate: try to express

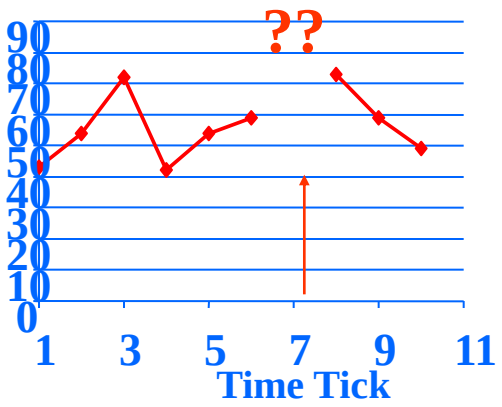
  x_t

  as a linear function of the past AND the future:

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

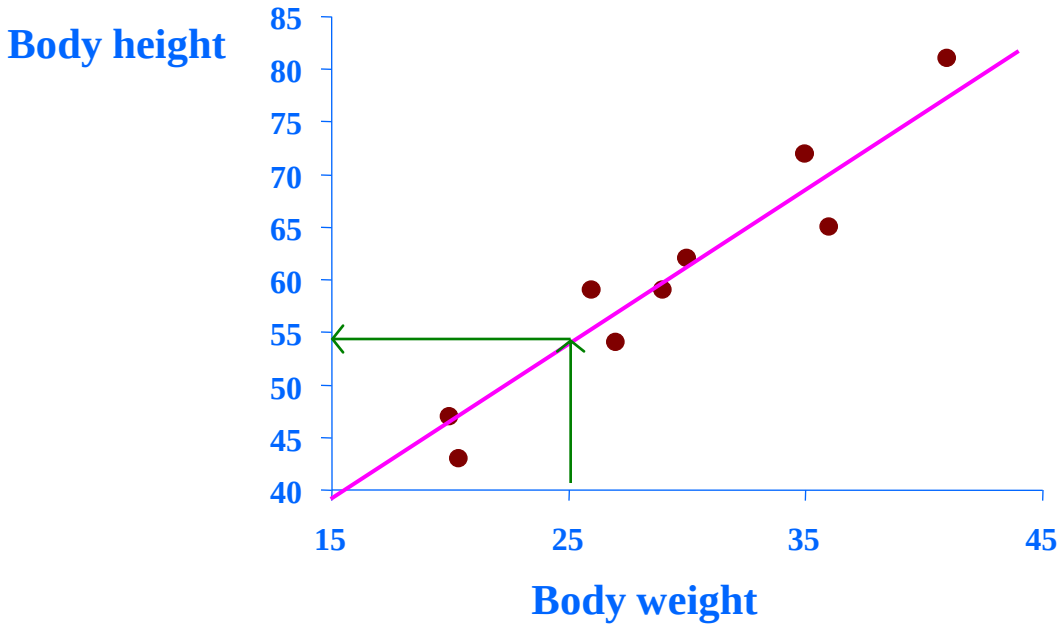
  (up to windows of w_{past} , w_{future})

- EXACTLY the same algo's



Linear Regression: idea

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



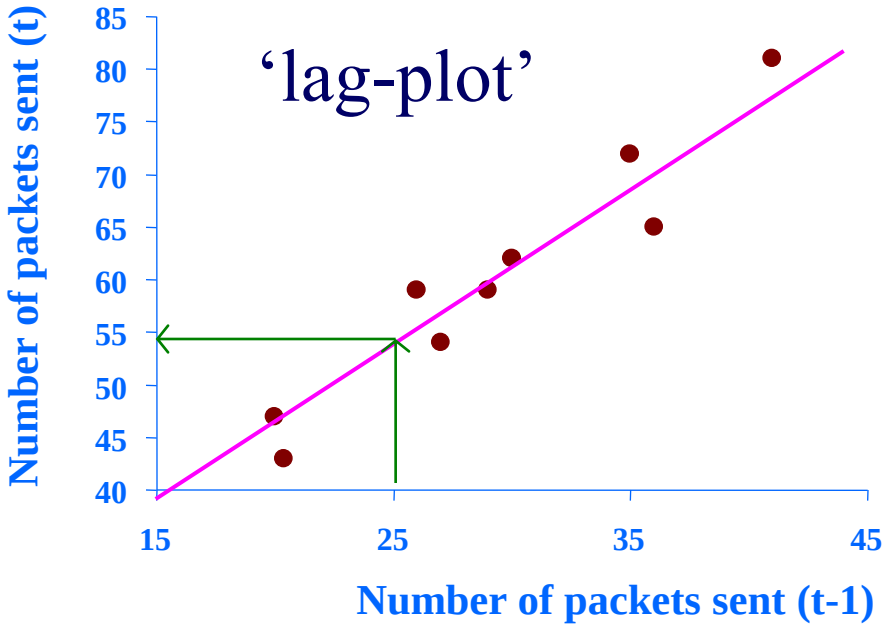
- express what we don't know (= 'dependent variable')
- as a linear function of what we know (= 'indep. variable(s)')

Linear Auto Regression:

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

Linear Auto Regression:

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



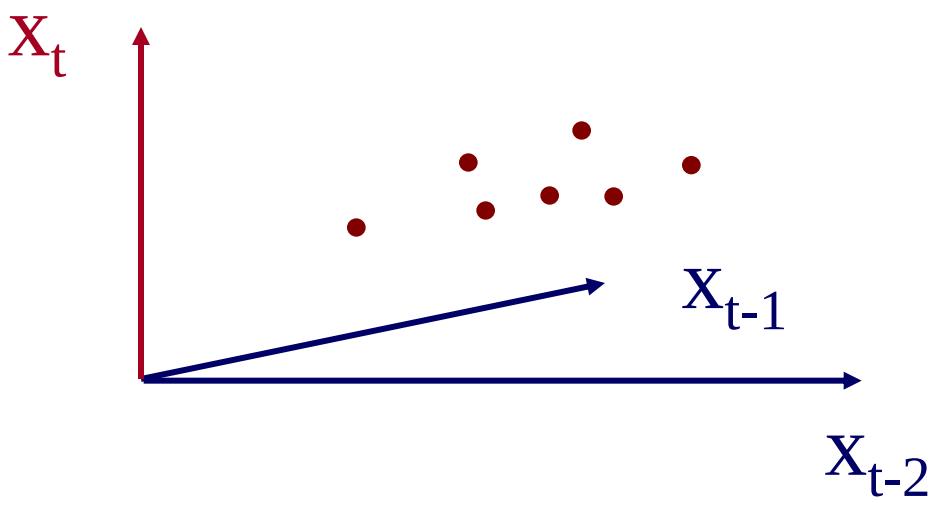
- lag w=1
- Dependent variable = # of packets sent (S [t])
- Independent variable = # of packets sent (S[t-1])

Outline

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

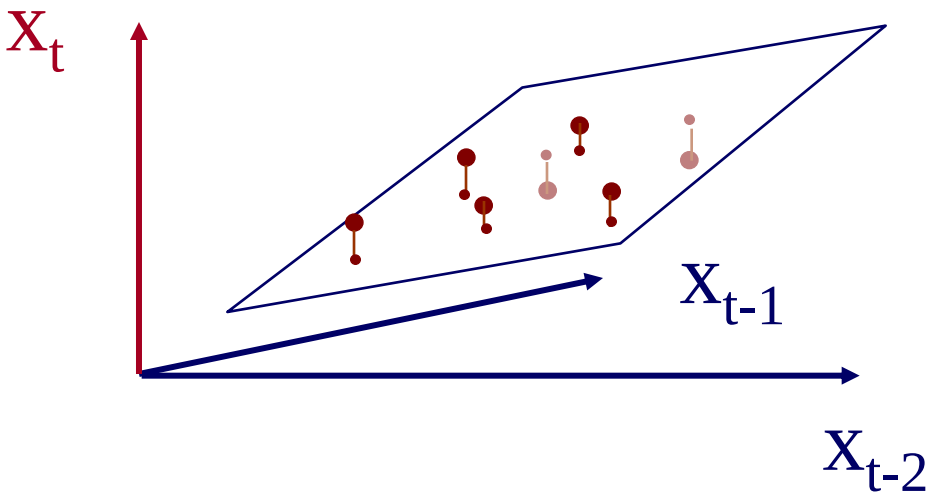
More details:

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



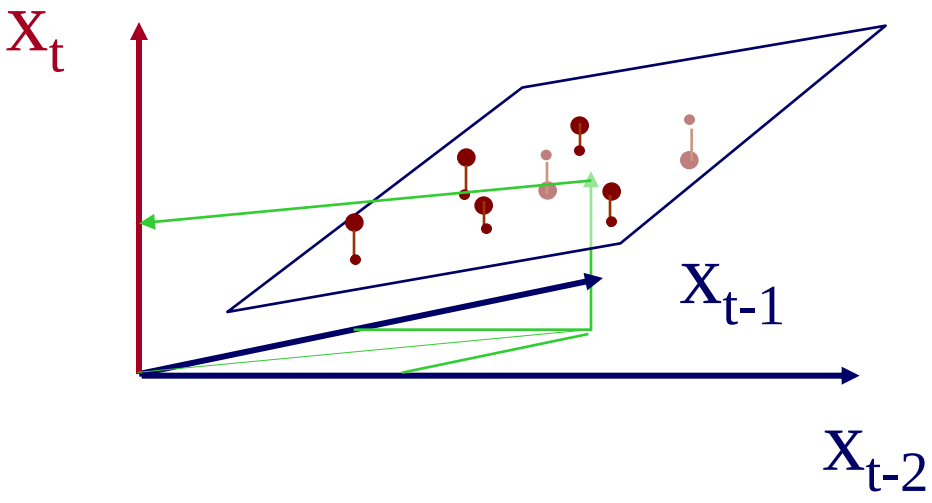
More details:

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



More details:

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



Skip

More details:

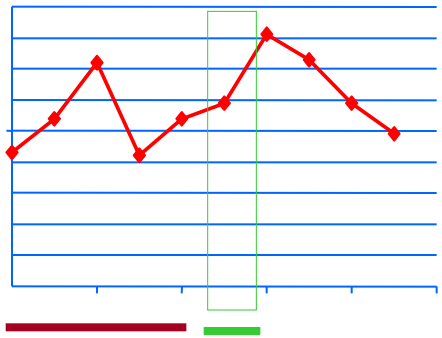
- Q1: Can it work with window $w>1$?
- A1: YES! The problem becomes:

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

- OVER-CONSTRAINED
  - **a** is the vector of the regression coefficients
  - **X** has the N values of the w indep. variables
  - **y** has the N values of the dependent variable

Skip

More details:



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

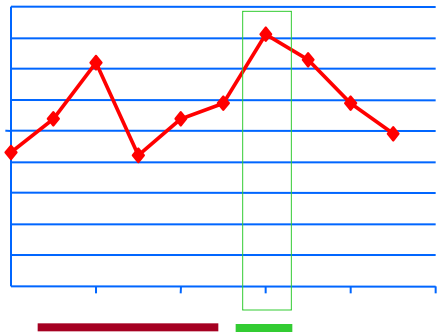
Ind-var1 Ind-var-w

time

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

Skip

More details:



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

Ind-var1 Ind-var-w

time

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

Skip

More details

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

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

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

Skip

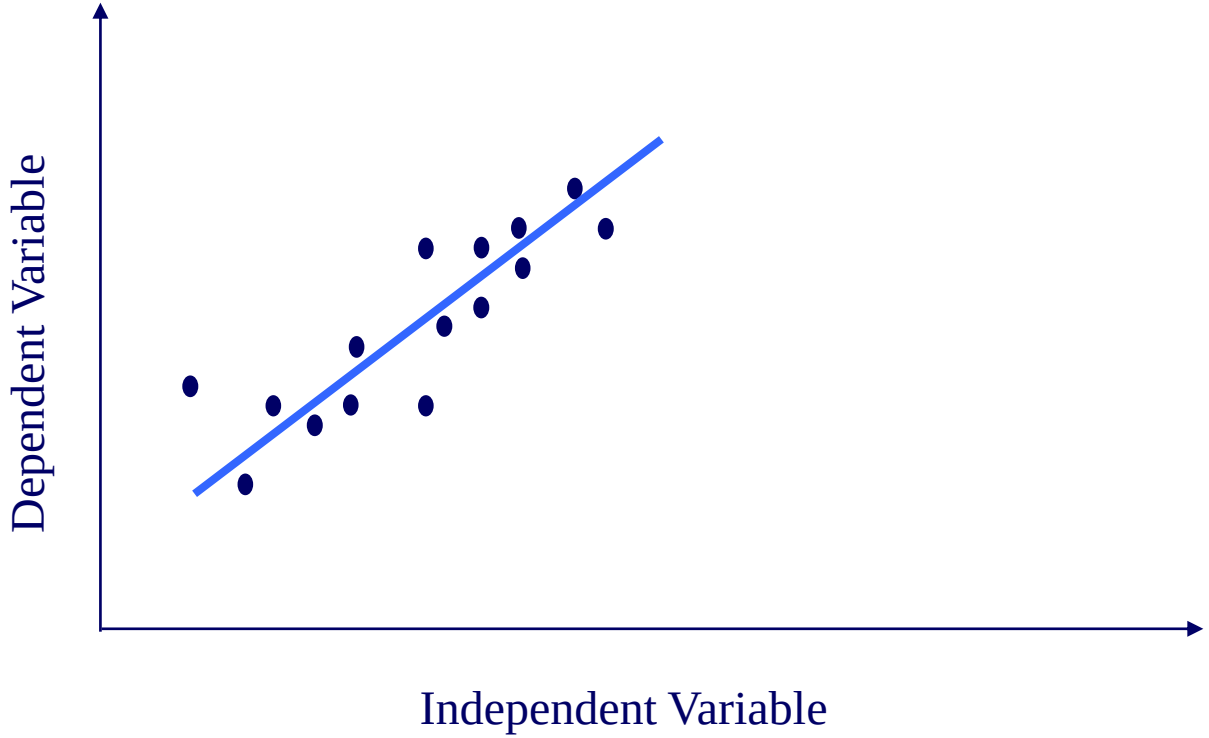
Even more details

- Q3: Can we estimate **a** incrementally?
- A3: Yes, with the brilliant, classic method of 'Recursive Least Squares' (RLS) (see, e.g., [Yi+00], for details) - pictorially:

Skip

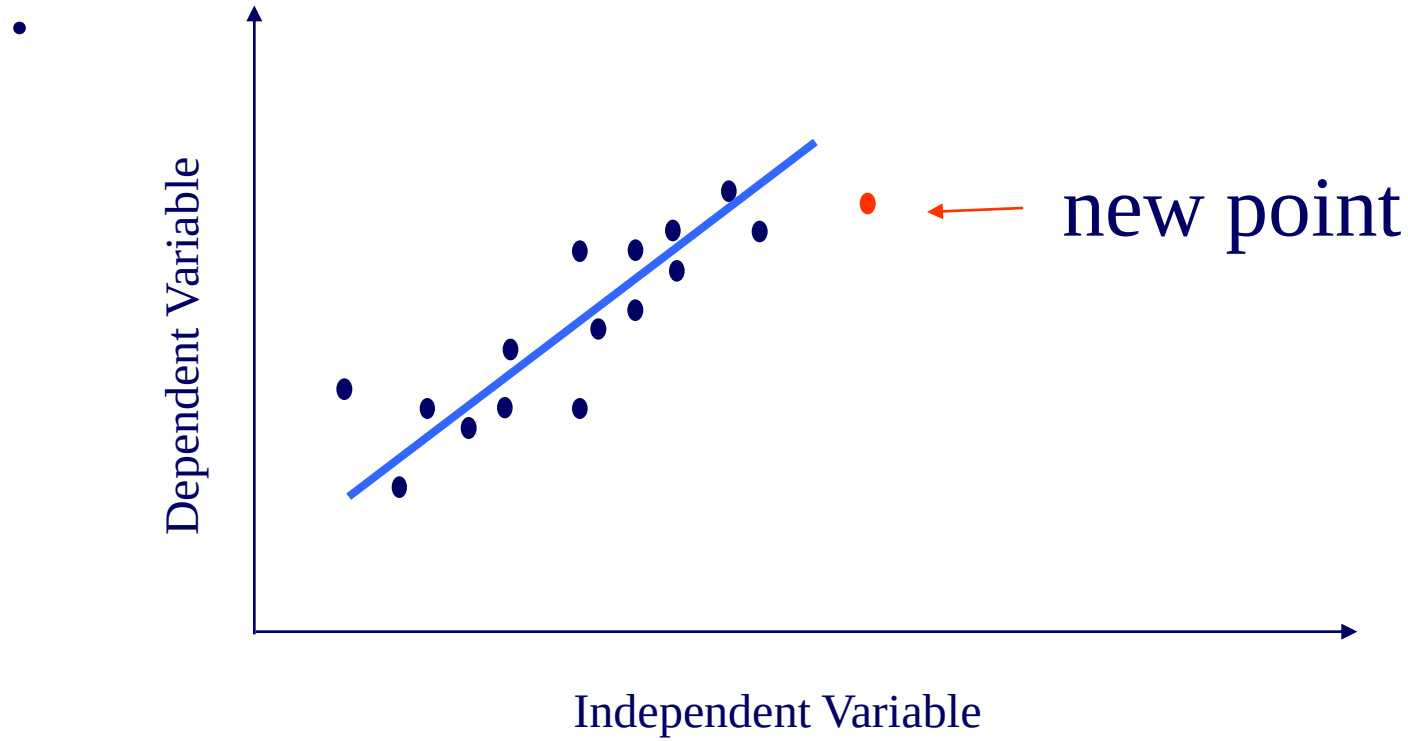
Even more details

- Given:



Skip

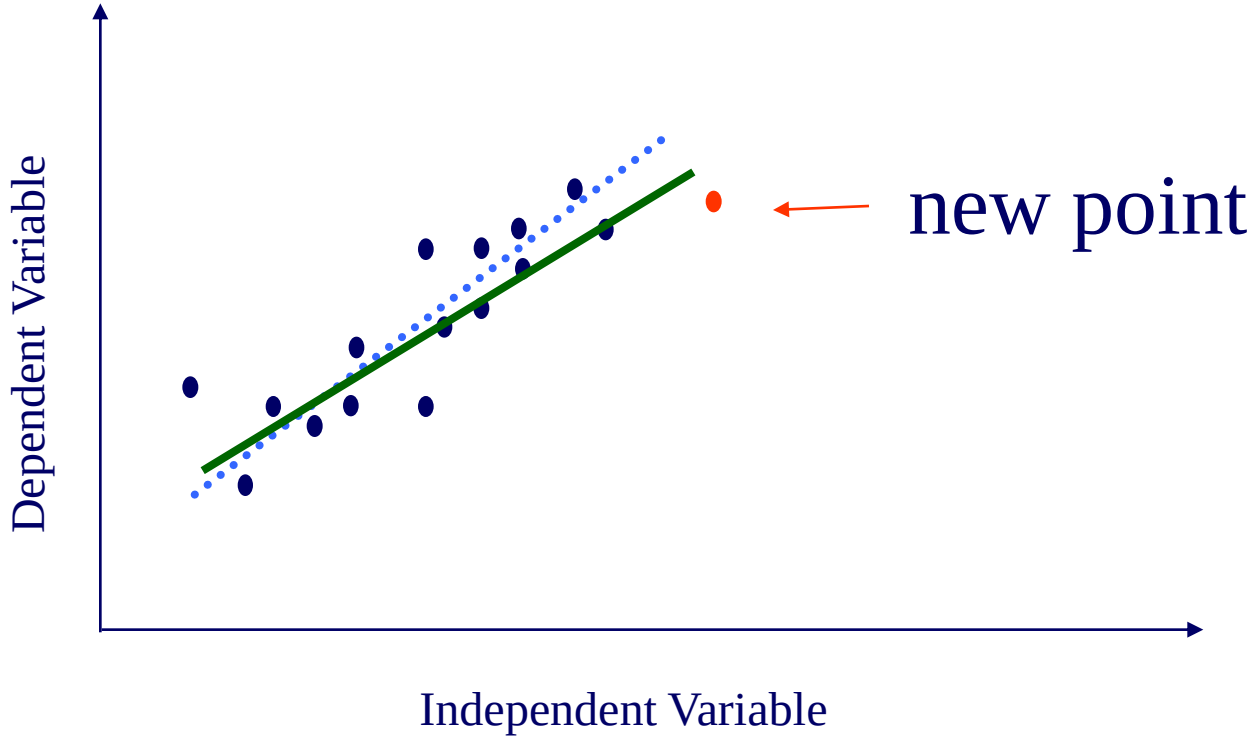
Even more details



Skip

Even more details

RLS: quickly compute new best fit



Skip

Even more details

- Straightforward Least Squares
  - Needs huge matrix (**growing** in size) $O(N \times w)$
  - Costly matrix operation $O(N \times w^2)$
- Recursive LS
  - Need much smaller, fixed size matrix $O(w \times w)$
  - Fast, incremental computation $O(1 \times w^2)$

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

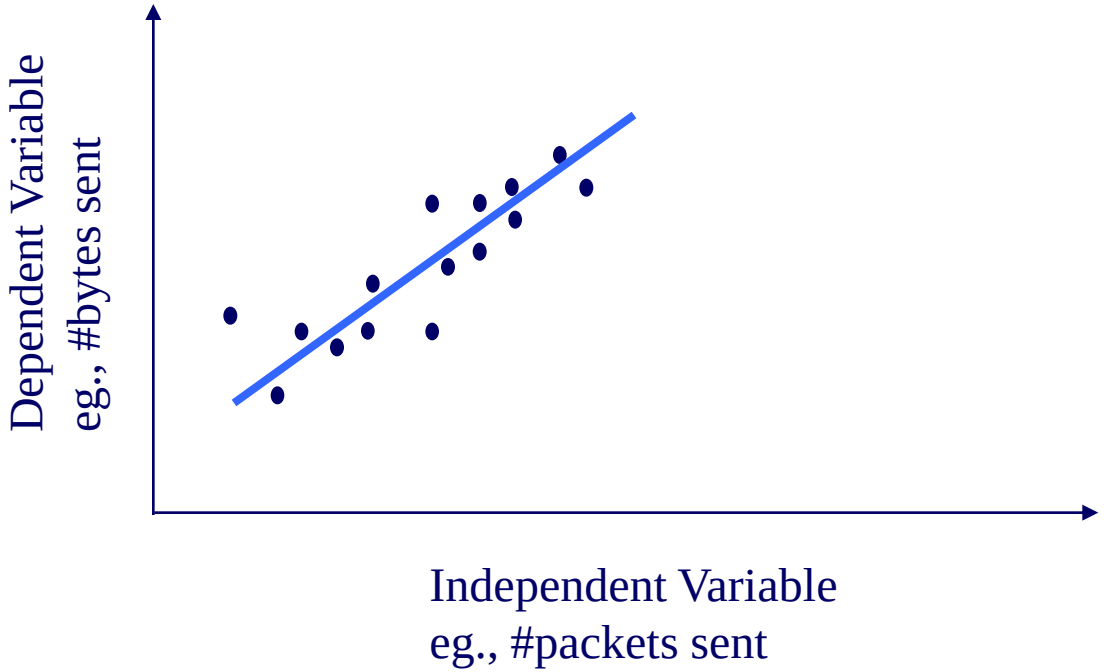
Skip

Even more details

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

Skip

Adaptability - ‘forgetting’

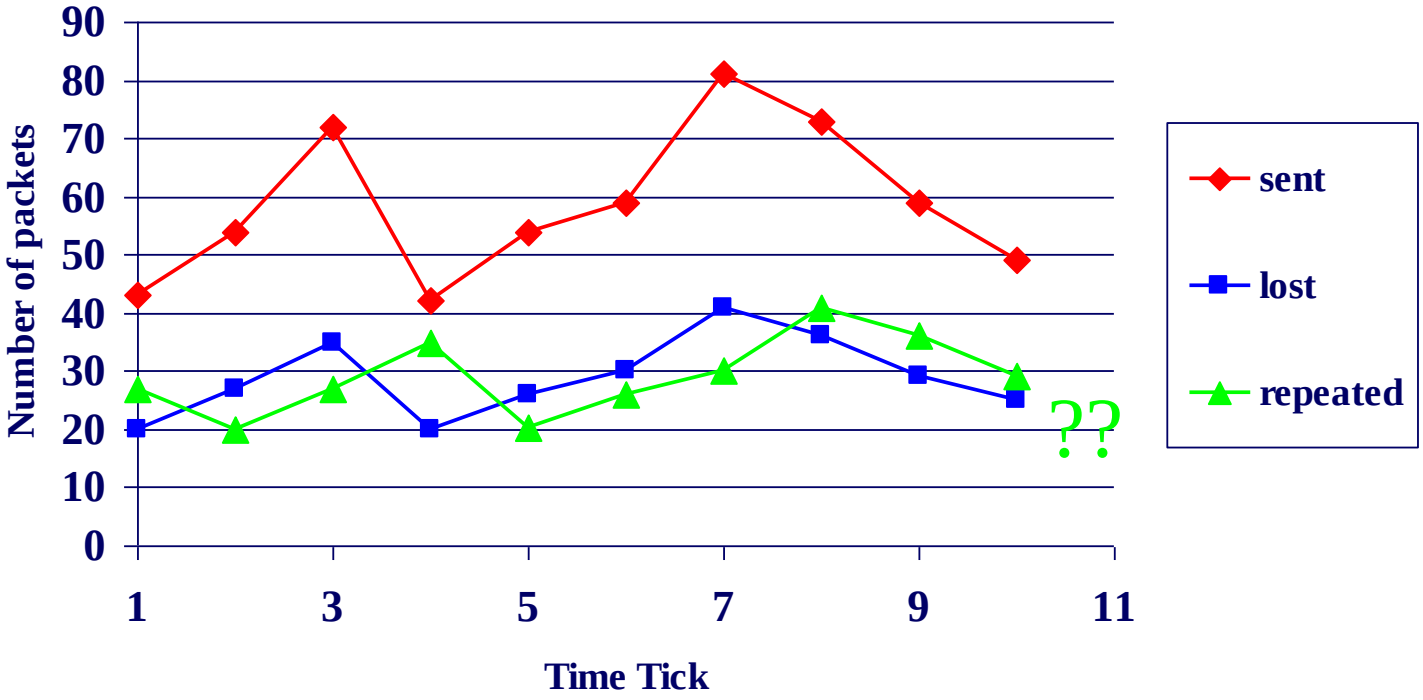


Outline

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

Co-Evolving Time Sequences

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



Solution:

Q: what should we do?

Solution:

Least Squares, with

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

Time Series Analysis - Outline

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

Conclusions - Practitioner's guide

- AR(IMA) methodology: prevailing method for linear forecasting
- Brilliant method of Recursive Least Squares for fast, incremental estimation.
- See [Box-Jenkins]

Resources: software and urls

- MUSCLES: Prof. Byoung-Kee Yi:
  http://www.postech.ac.kr/~bkyi/
  or christos@cs.cmu.edu
- free-ware: 'R' for stat. analysis
  (clone of Splus)
  http://cran.r-project.org/

Books

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

Additional Reading

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

BREAK!

Next: Kalman filters

Indexing and Mining Time Sequences
Part 4
Kalman Filters

Christos Faloutsos

Lei Li

CMU

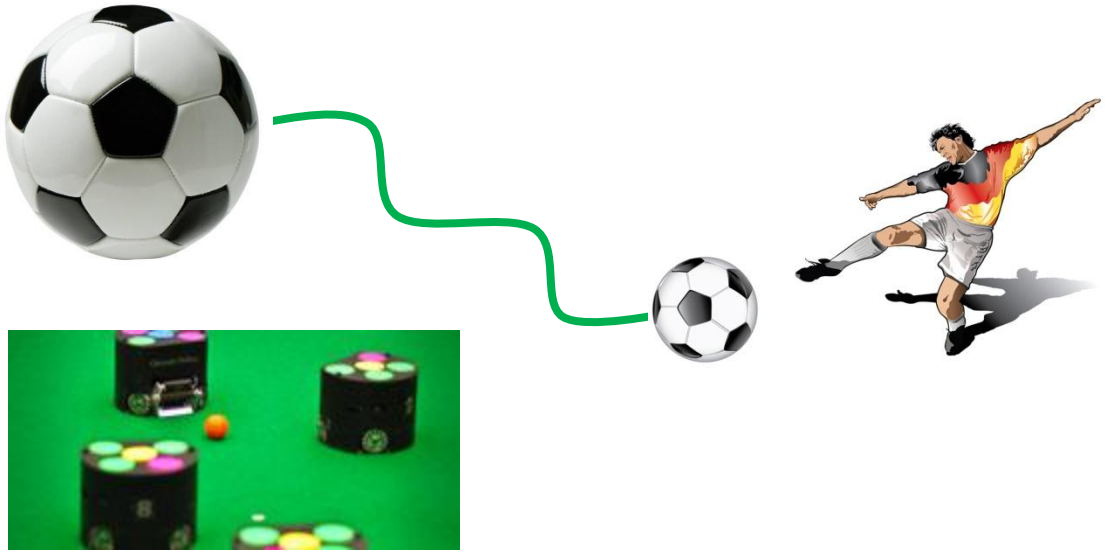
Outline

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

Intuition

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

from FIFA 2010



RoboCup 2010

Linear Dynamical System

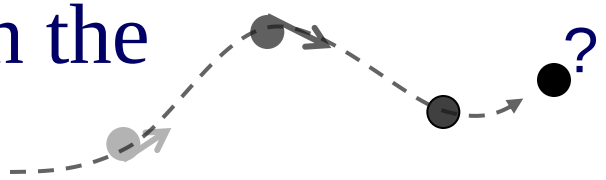
- Known parameters
  - Original Kalman Filters [Kalman 1960, Rauch 1965]
- Unknown parameters
  - Parameter estimation through EM algorithm [Shumway et al 1982, Ghahramani 1996]

Kalman Filters

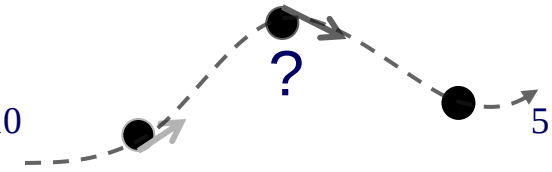
Given observations of the soccer ball position t=1..T, "Model parameters"

Goal: two types of prediction

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



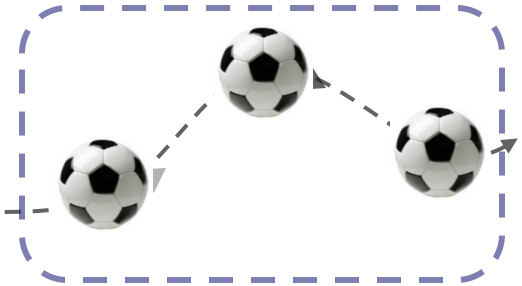
Kalman smoothing: Estimate for every time tick, based on all observations



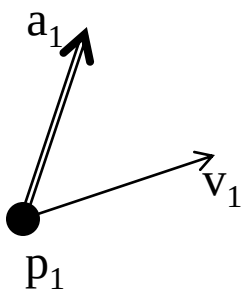
Kalman Filters (intuition)

t=1, soccer with initial pos, vel and acc.

To estimate the future

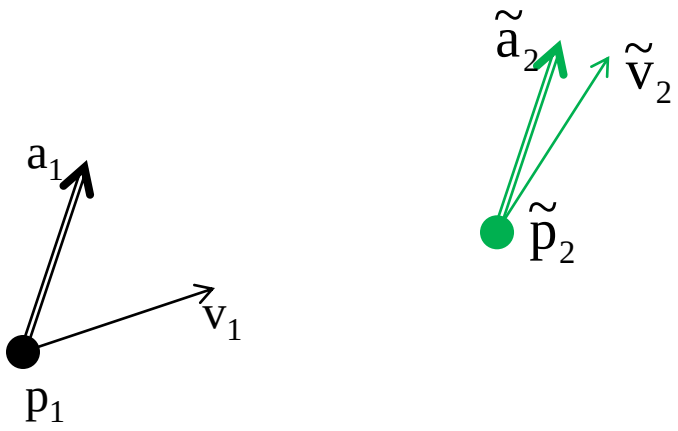


past observations



Kalman Filters (intuition)

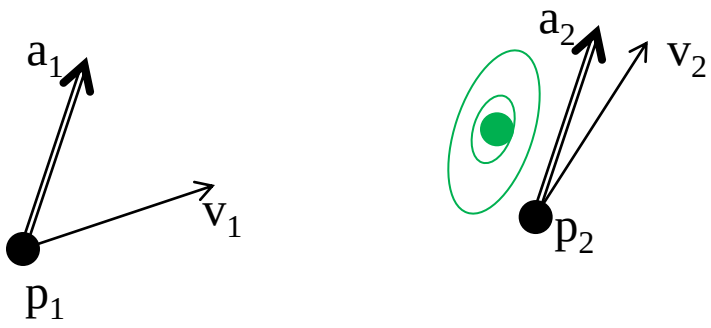
t=2, according to Newton's law, it should be...



Kalman Filters (intuition)

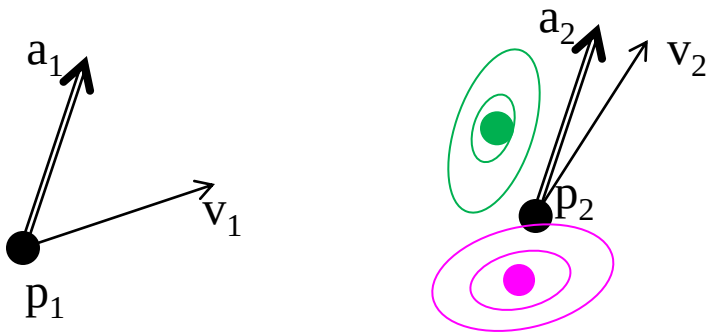
t=2, according to Newton's law, it should be...

however, imperfect soccer/kick movement…



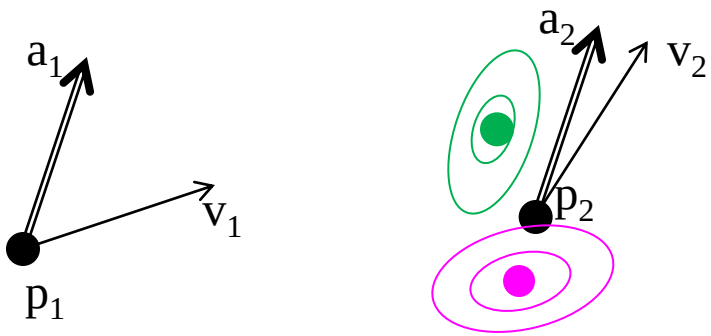
Kalman Filters (intuition)

Now take a photo, due to imperfect camera…



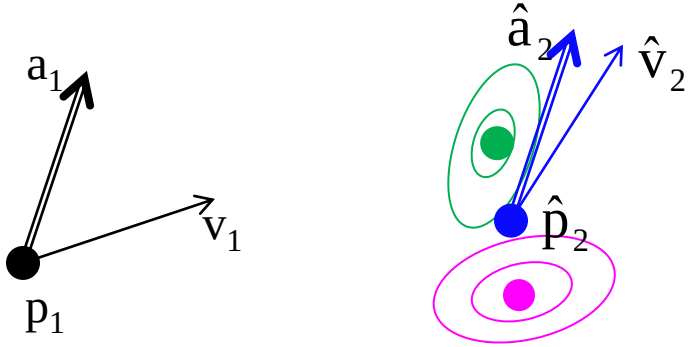
Kalman Filters (intuition)

What is the best estimate for next time tick?



Kalman Filters (intuition)

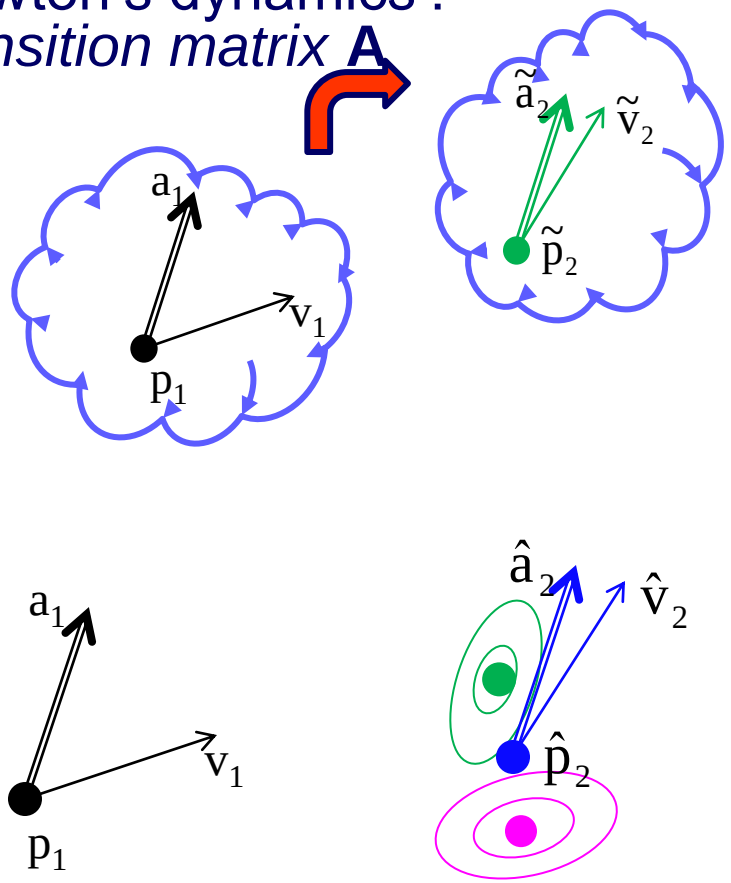
What is the best estimate for next time tick?



Some math notation

	name
A	transition matrix
C	transmission/ projection/ output matrix
Q	transition covariance
R	transmission/ projection/ output covariance

‘Newton’s dynamics’:
Transition matrix **A**



(details)

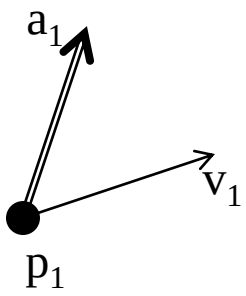
Example

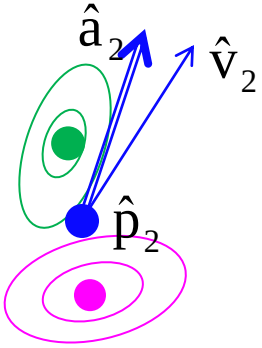
hidden states $z_1=(p_1, v_1, a_1)^T$

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

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

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





transition covariance Q

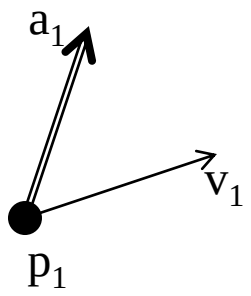
output covariance R

(details)

Example

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

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



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

$p_2=p_1+ v_1*\Delta t + 0.5*a_1*\Delta t^2$

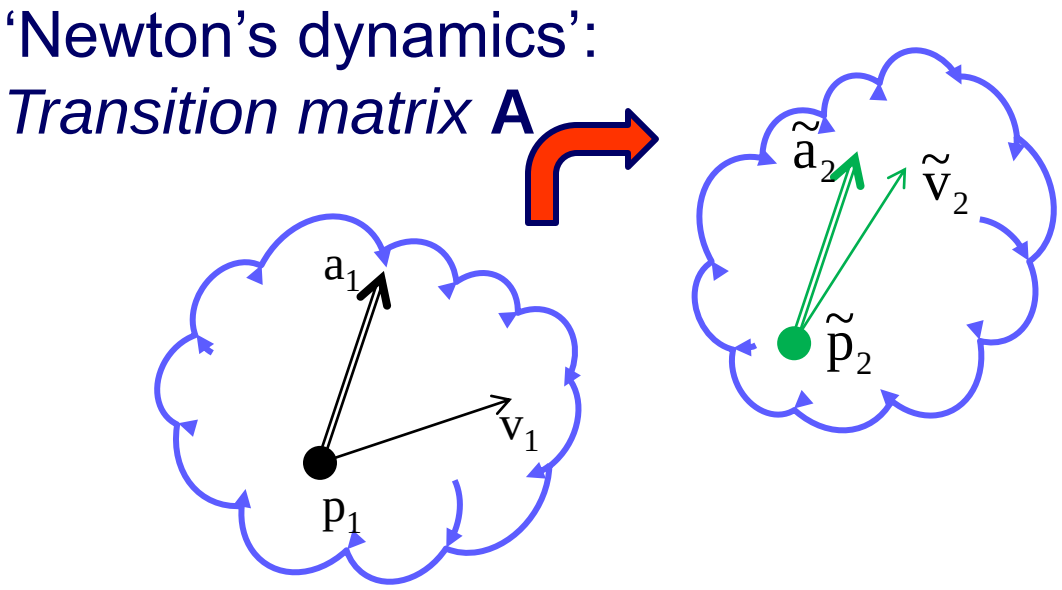
$v_2=v_1+a_1*\Delta t$

$a_2=a_1$

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

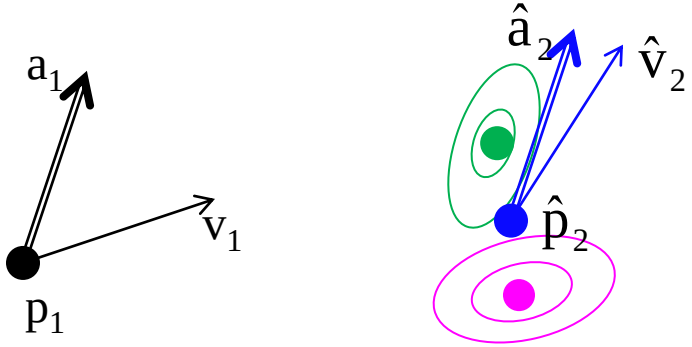
Kalman Filtering (intuition)

Step1, forecast next time tick before observation



Kalman Filtering (intuition)

Step 2: adjust estimation after observation



Example: Kalman filtering (forward)

Given:

a sequence of observations, Model parameters (A, C …)

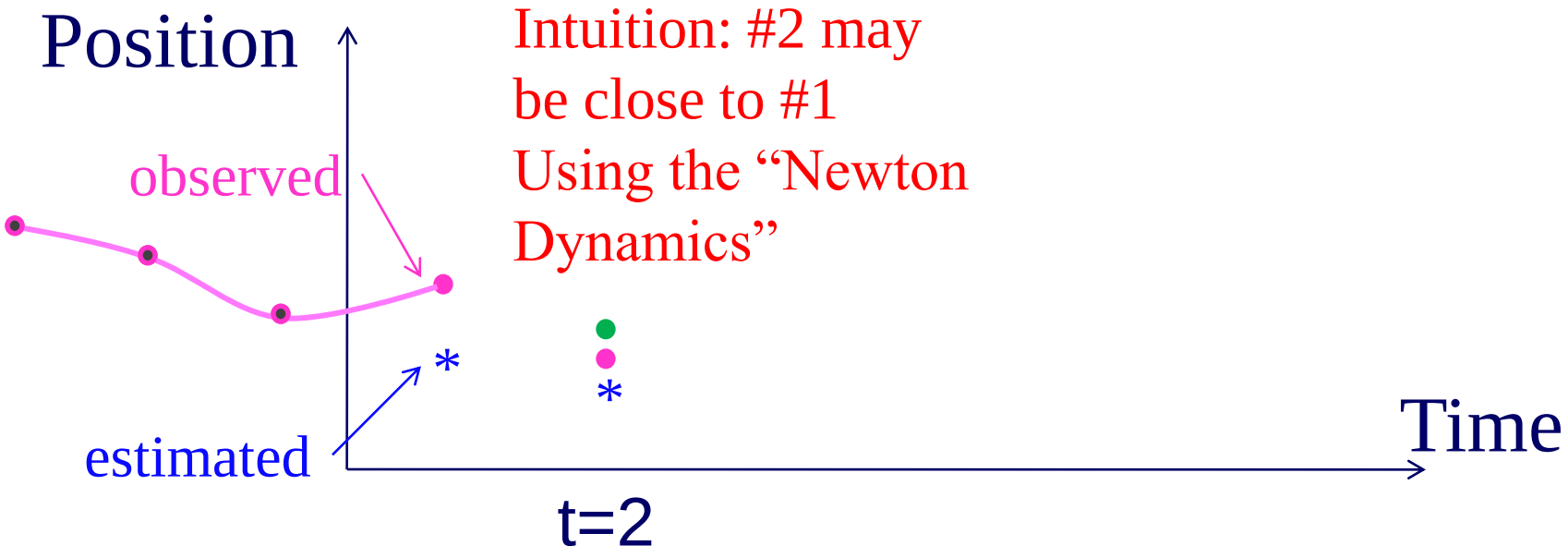
Goal: remove noise and forecast real position

Example: Kalman filtering (forward)

[R. E. Kalman. A new approach to linear filtering and prediction problems. 1960]

Example: Kalman filtering

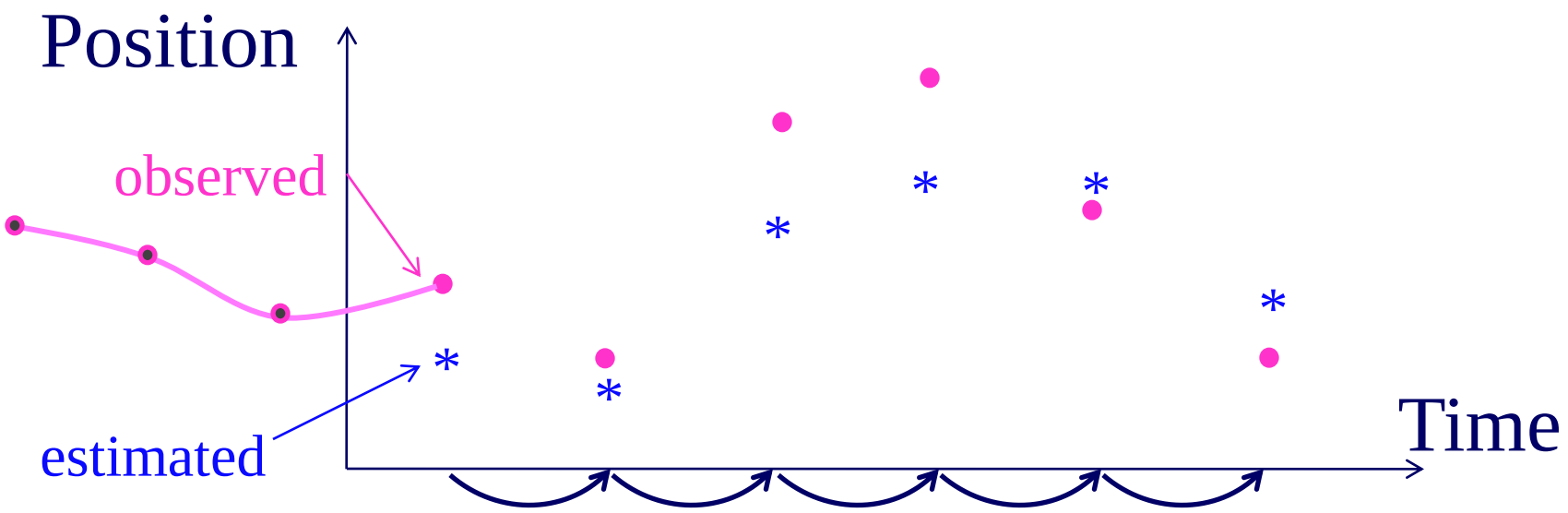
$$\hat{z}_n = A \cdot \hat{z}_{n-1} + K_n \cdot (x_n - C \cdot A \cdot \hat{z}_{n-1})$$
$$\hat{V}_n = (I - K_n) \cdot P_{n-1}$$
$$K_n = P_{n-1} \cdot C^T \cdot (C \cdot P_{n-1} \cdot C^T + R)^{-1}$$
$$P_{n-1} = A \cdot \hat{V}_{n-1} \cdot A^T + Q$$



Intuition: #2 may be close to #1
Using the "Newton Dynamics"

Example: Kalman filtering

$$\hat{z}_n = A \cdot \hat{z}_{n-1} + K_n \cdot (x_n - C \cdot A \cdot \hat{z}_{n-1})$$
$$\hat{V}_n = (I - K_n) \cdot P_{n-1}$$
$$K_n = P_{n-1} \cdot C^T \cdot (C \cdot P_{n-1} \cdot C^T + R)^{-1}$$
$$P_{n-1} = A \cdot \hat{V}_{n-1} \cdot A^T + Q$$

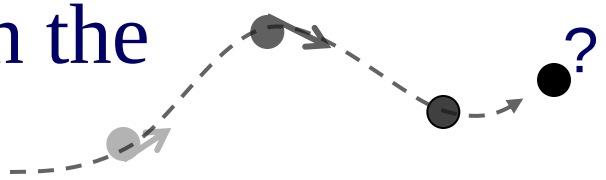


Kalman Filters

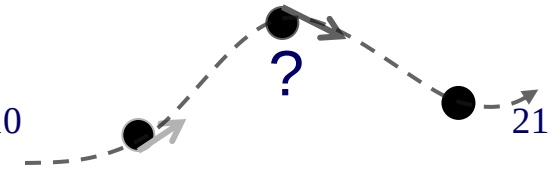
Given observations of the soccer position t=1..T, and model parameters (A, C …)

Goal: two types of prediction

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



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



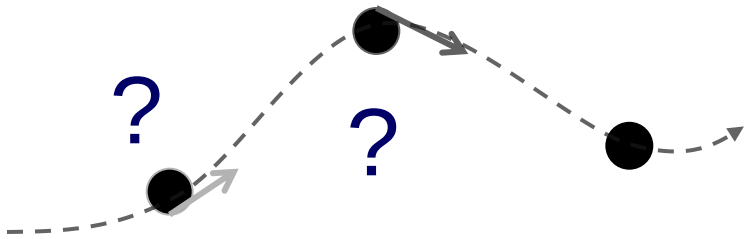
Kalman Smoothing

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

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

Difference from Kalman filtering:

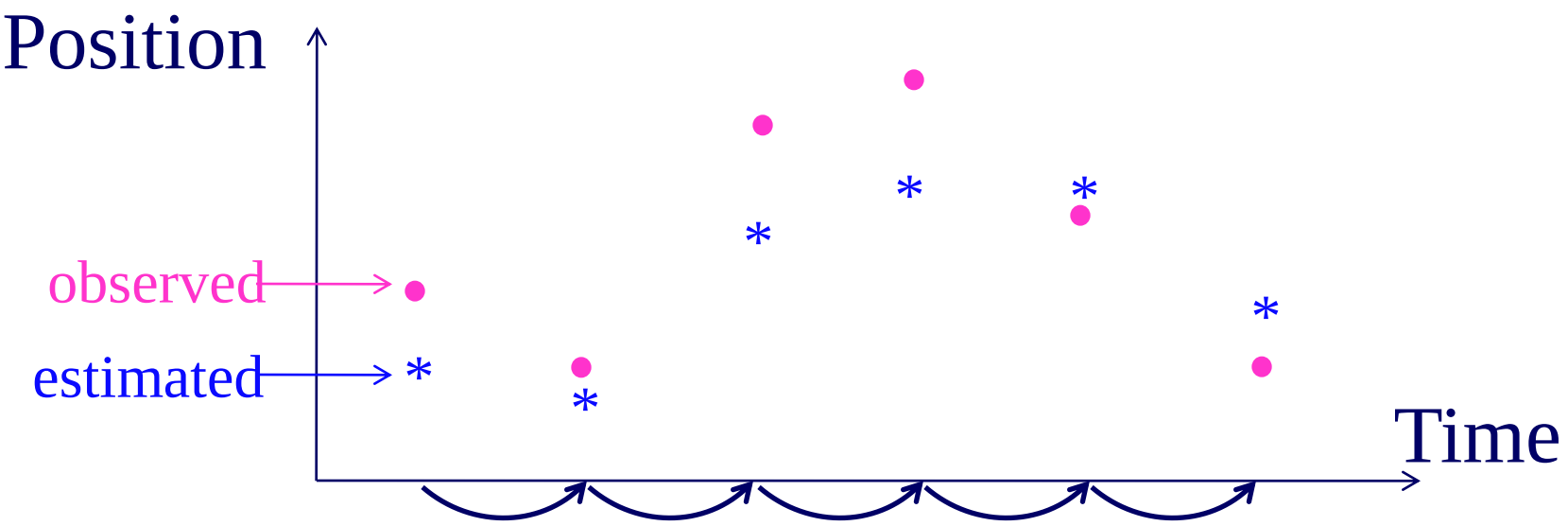
bring future observation back in history estimate



Recap: Kalman filtering

$$\hat{z}_n = A \cdot \hat{z}_{n-1} + K_n \cdot (x_n - C \cdot A \cdot \hat{z}_{n-1})$$
$$\hat{V}_n = (I - K_n) \cdot P_{n-1}$$
$$K_n = P_{n-1} \cdot C^T \cdot (C \cdot P_{n-1} \cdot C^T + R)^{-1}$$
$$P_{n-1} = A \cdot \hat{V}_{n-1} \cdot A^T + Q$$

Forward



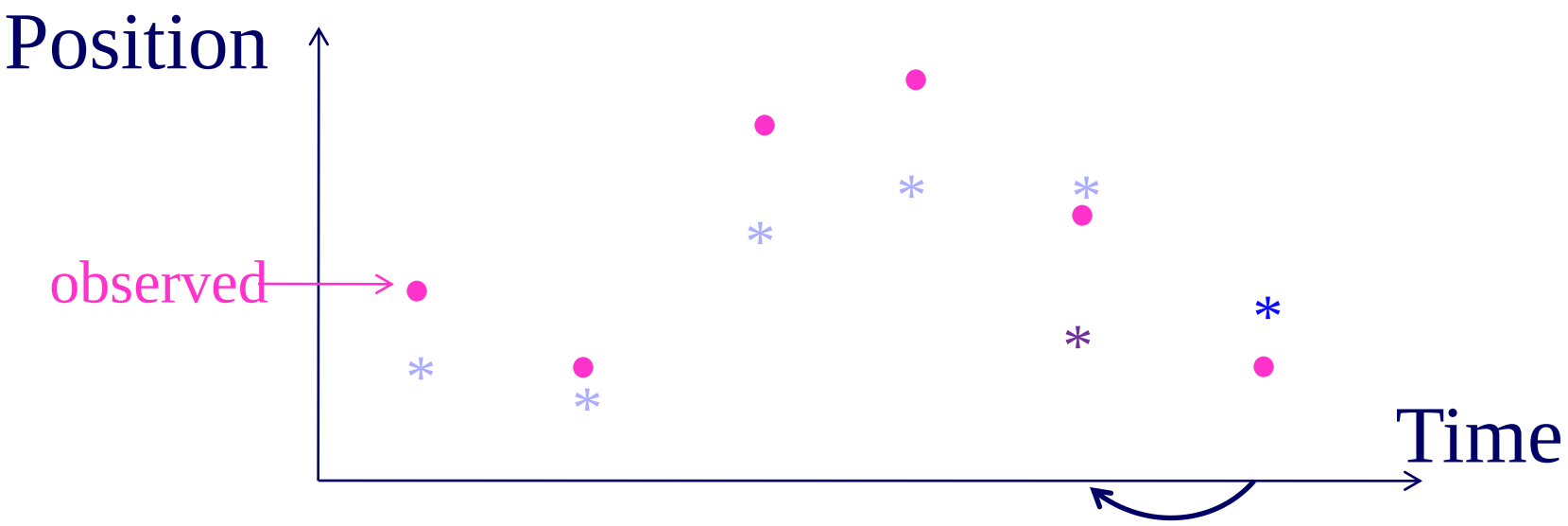
Example: Kalman Smoothing

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

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

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

Backward 1



[Rauch et al, Maximum likelihood estimates of linear dynamic systems. 1965]

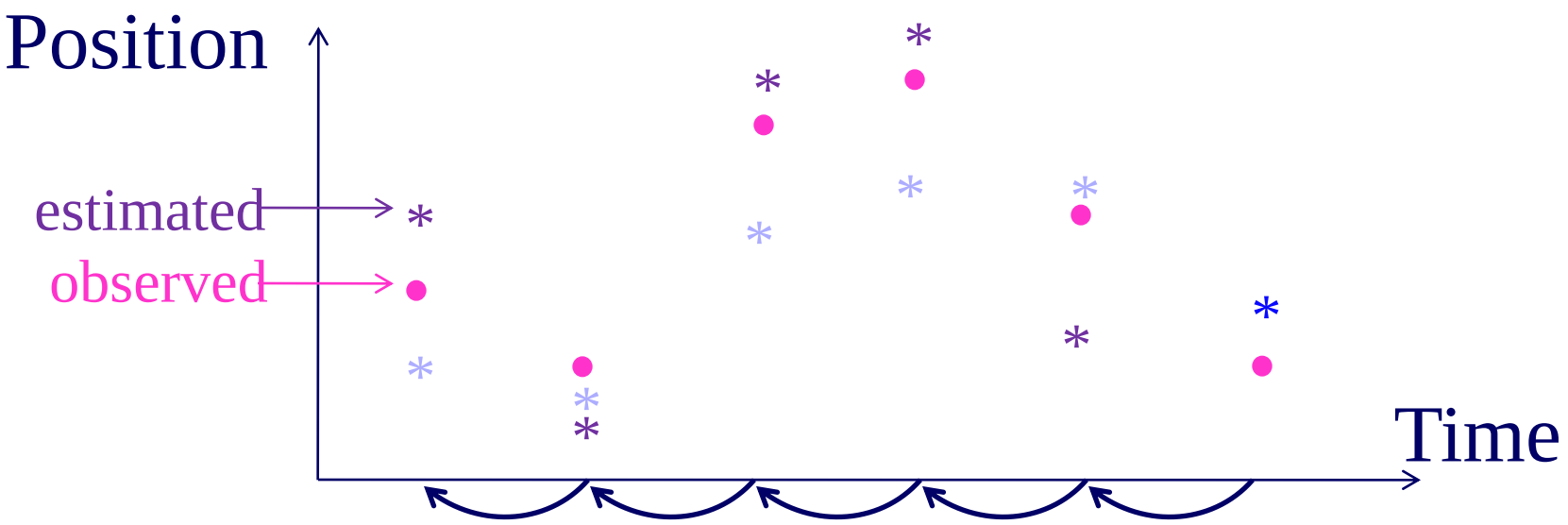
Example: Kalman Smoothing

Backward 2

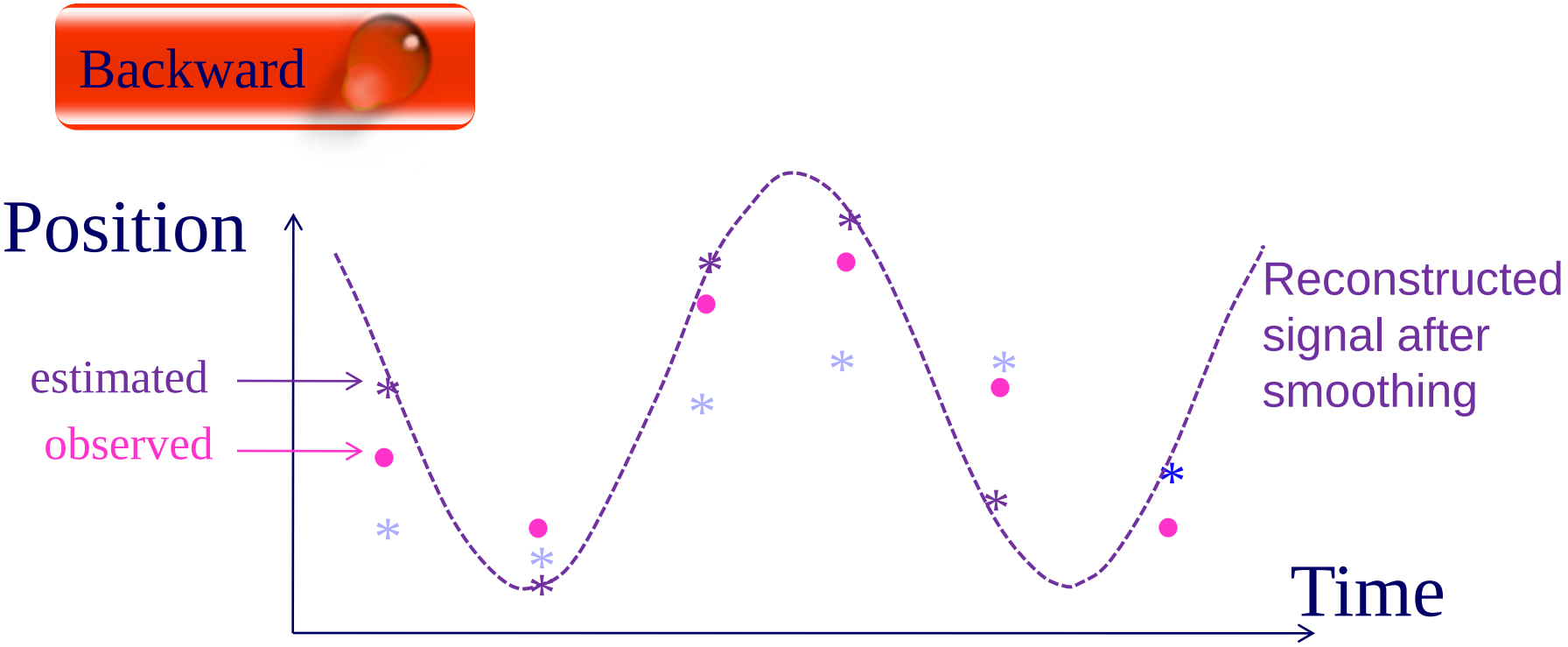
$$\hat{\hat{z}}_n = \hat{z}_n + J_n \cdot (\hat{\hat{z}}_{n+1} - A \cdot \hat{z}_n)$$

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

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



Example: Kalman Smoothing

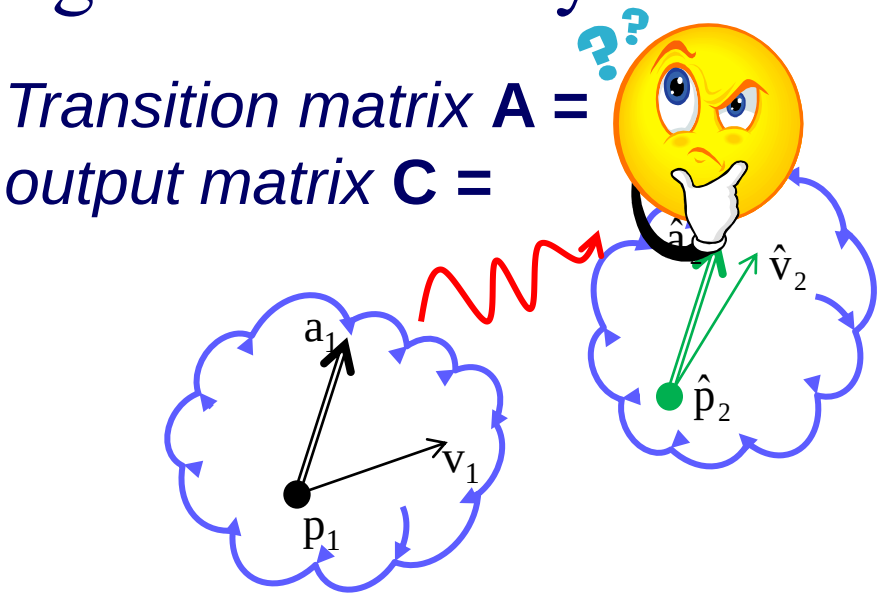


Outline

- Intuition, example, and definition
  - Original Kalman
  - Kalman filters with parameter estimation
- Extensions
- Kalman filters at work

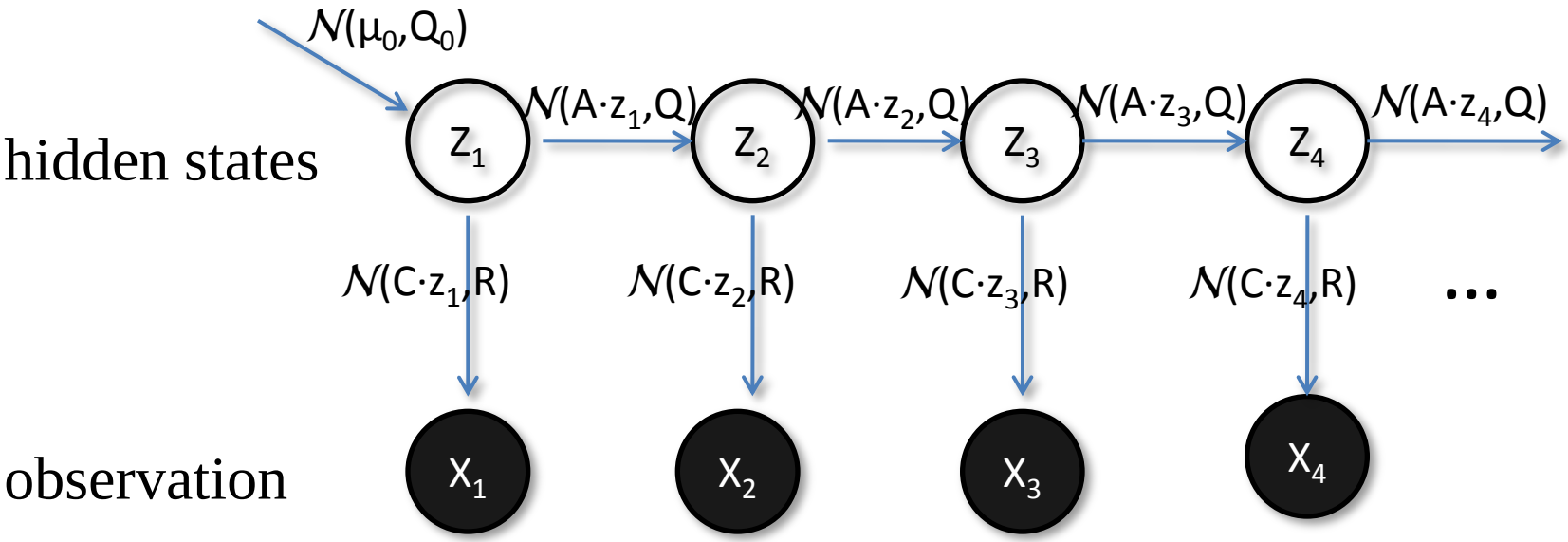
What if not know the model parameters?

- E.g. Datacenter sensor temperatures
- no longer "Newton dynamics"



(details)

Graphical Model Representation



Model parameters:

$$\theta=\{\mu_0, Q_0, A, Q, C, R\}$$

$$z_1 = \mu_0 + \omega_0$$
$$z_{n+1} = A\cdot z_n + \omega_n$$
$$x_n = C\cdot z_n + \varepsilon_n$$

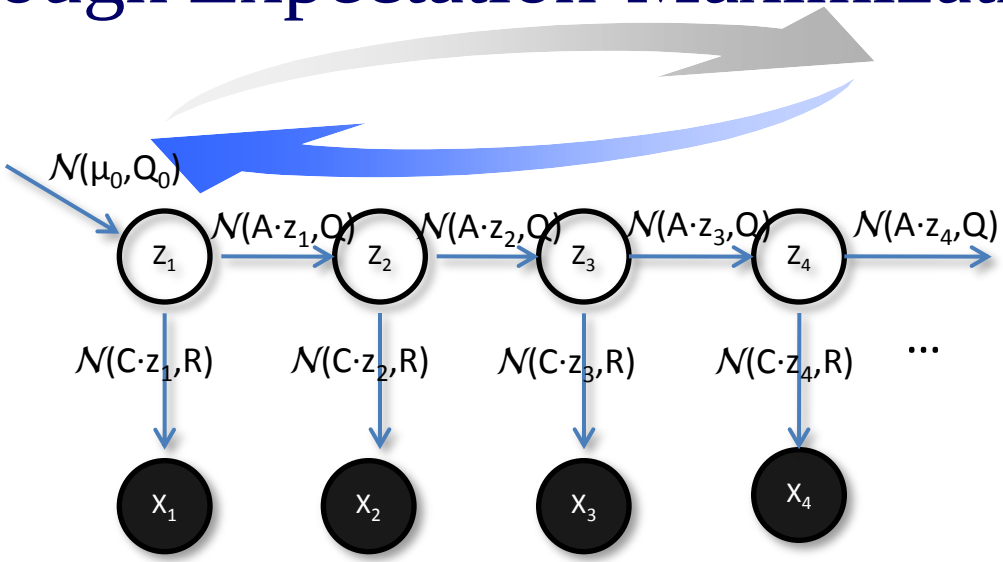
Gaussian noise

Linear Dynamical Systems: parameters

	name	meaning & example
μ_0	initial state for hidden variable	e.g. initial position, velocity & acceleration
A	transition matrix	how the states move forward, e.g. soccer flying in the air
C	transmission/ projection/ output matrix	hidden state ➔ observation, e.g. camera taking picture of the soccer
Q_0	Initial covariance	
Q	transition covariance	how precision is the soccer motion
R	transmission/ projection covariance	i.e. observation noise; e.g. how accurate is the camera

Estimating parameters of LDS

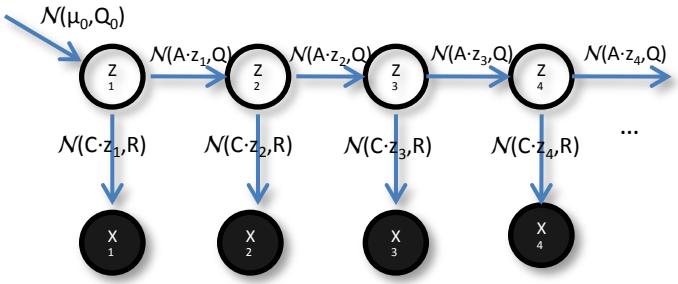
- Given: a sequence of observations (e.g. car positions)
- Find: best-fit parameters
- Basic principle: maximum likelihood
- Through Expectation-Maximization Alg.



Learning LDS: EM alg.

- E-step: Kalman filtering-smoothing
  - Estimate the hidden variables based on observation and current parameters
- M-step:
  - Update parameters (A, C…) to maximize

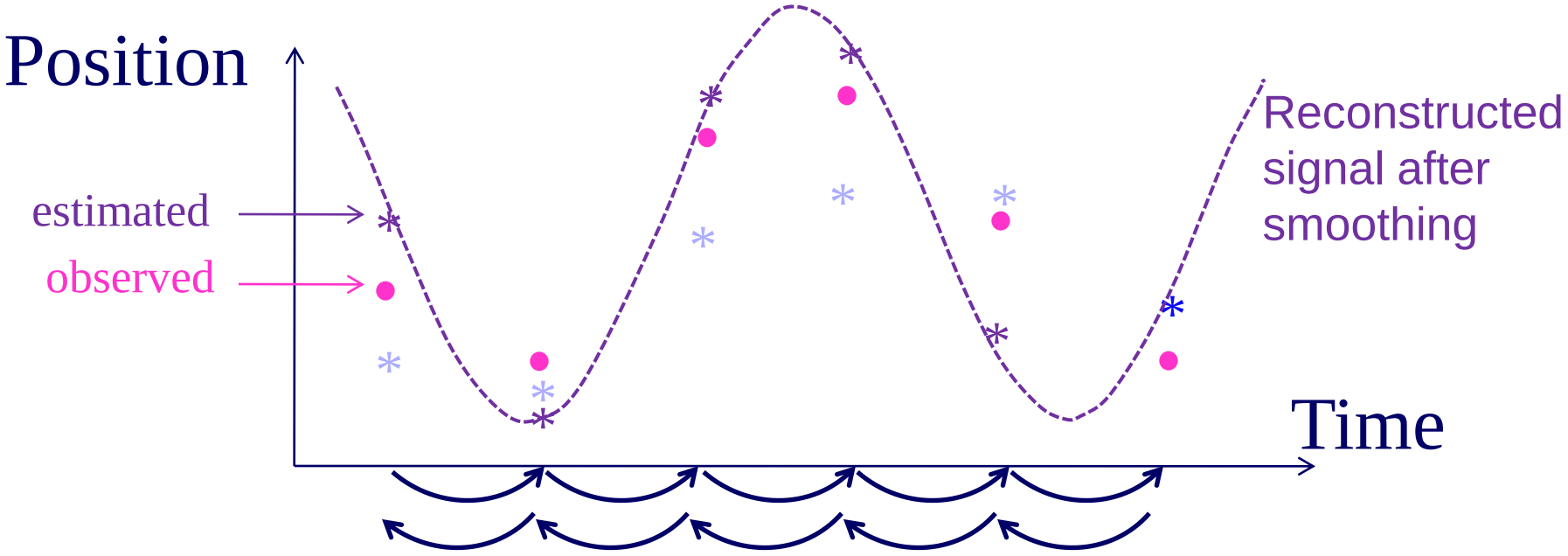
$$L(\theta;\mathcal{X}) = \mathbb{E}_{\mathcal{X},\mathcal{Z}|\theta}[-D(\vec{z}_1,\vec{\mu}_0,\mathbf{Q}_0) - \sum_{t=2}^{\mathrm{T}} D(\vec{z}_t,\mathbf{A}\vec{z}_{t-1},\mathbf{Q}) - \sum_{t=1}^{\mathrm{T}} D(\vec{x}_t,\mathbf{C}\vec{z}_t,\mathbf{R}) - \frac{1}{2}\log|\mathbf{Q}_0| - \frac{\mathrm{T}-1}{2}\log|\mathbf{Q}|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



[Shumway et al 1982, Ghahramani 1996]

EM alg. Intuition

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



EM alg.

M-step: Maximizing the log-likelihood

$$
\begin{aligned}
L(\theta;\mathcal{X}) \quad = \quad & \mathbb{E}_{\mathcal{X},\mathcal{Z}|\theta}[-D(\vec{z}_1,\vec{\mu}_0,\mathbf{Q}_0) \\
& -\sum_{t=2}^{\mathrm{T}} D(\vec{z}_t,\mathbf{A}\vec{z}_{t-1},\mathbf{Q}) - \sum_{t=1}^{\mathrm{T}} D(\vec{x}_t,\mathbf{C}\vec{z}_t,\mathbf{R}) \\
& -\frac{1}{2}\log|\mathbf{Q}_0| - \frac{\mathrm{T}-1}{2}\log|\mathbf{Q}| - \frac{T}{2}\log|\mathbf{R}|)
\end{aligned}
$$

by solving

$$\frac{\partial L}{\partial \mathrm{A}} = 0$$

$$\frac{\partial L}{\partial \mathrm{C}} = 0$$

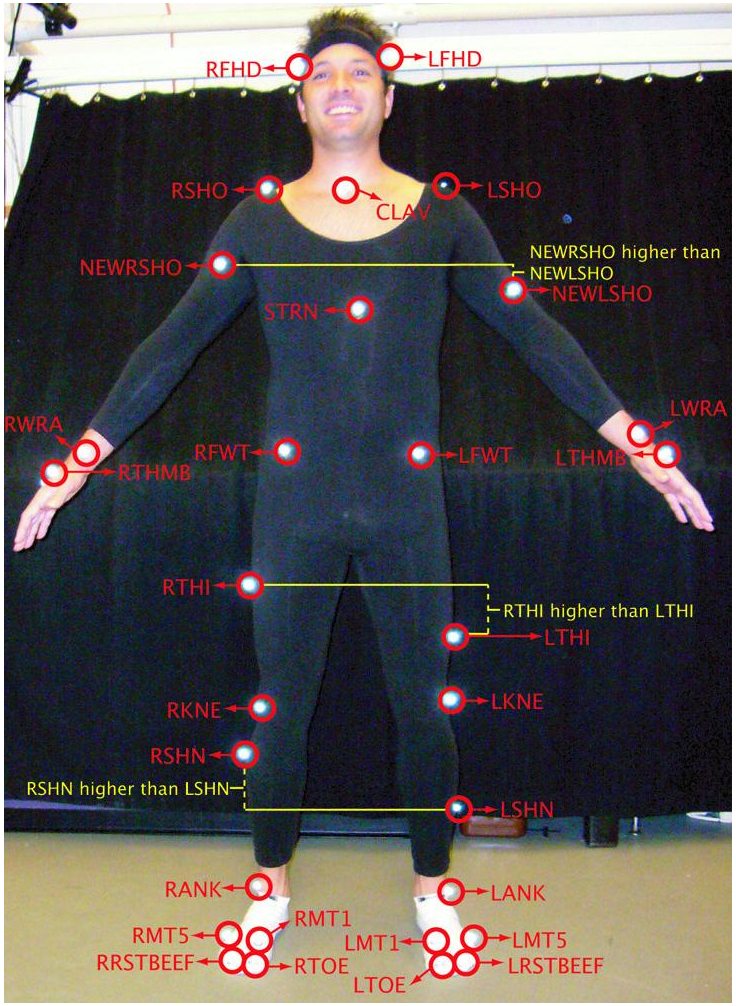
. . .

Outline

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

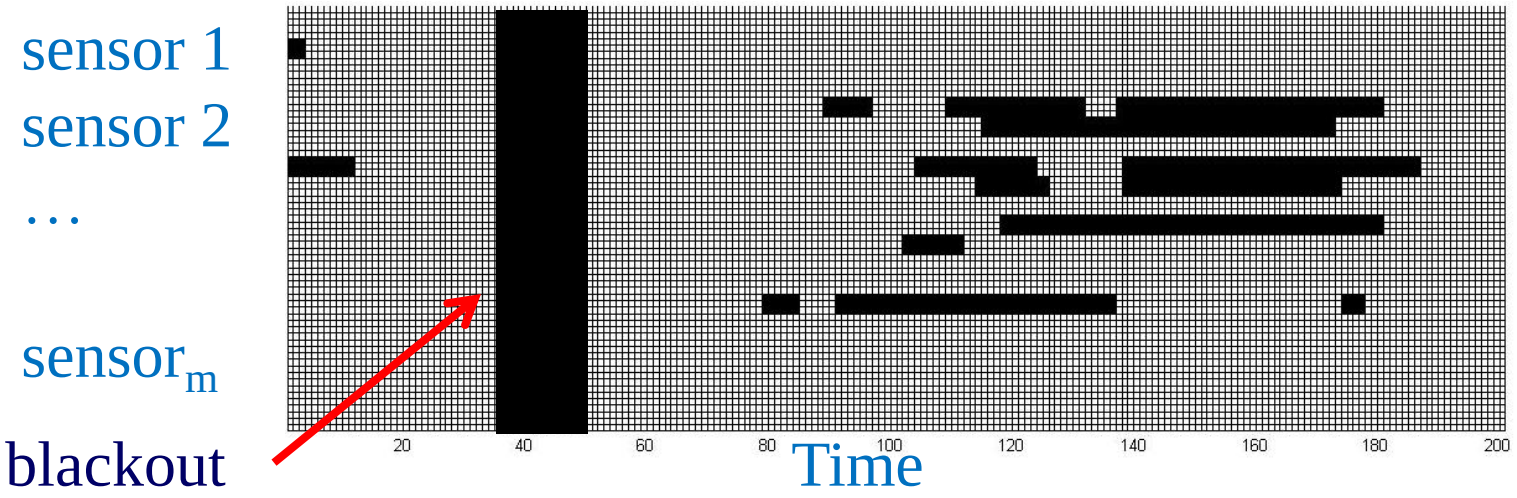
Motivation Examples

- Motion Capture:
  - Markers on human actors
  - Cameras used to track the 3D positions
  - Duration: 100-500
  - 93 dimensional body-local coordinates after preprocessing (31-bones)
- Sensor data missing due to:
  - Low battery
  - RF error



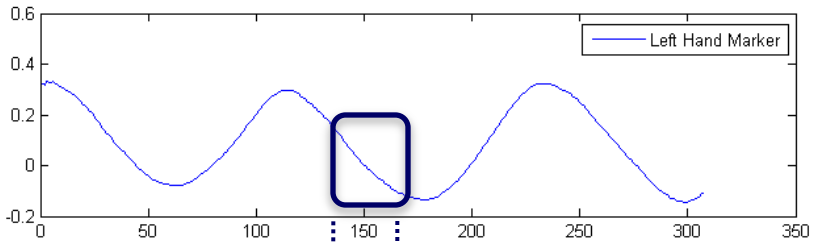
From mocap.cs.cmu.edu

Illustration of data

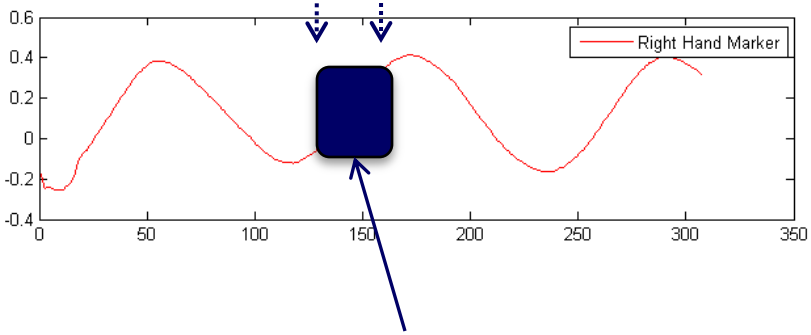


DynaMMo: Intuition

Position of Left hand marker

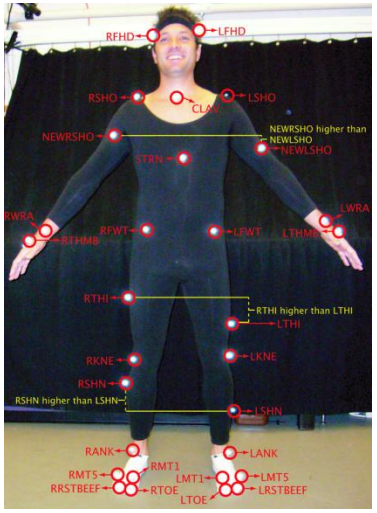


Position of right hand marker



missing

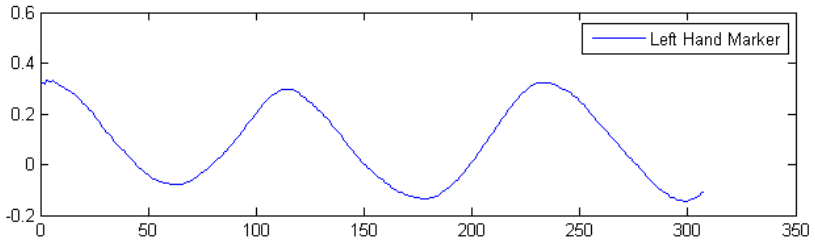
Recover using Correlation among multiple sequences



[Lei Li et al DynaMMo. KDD 2009]

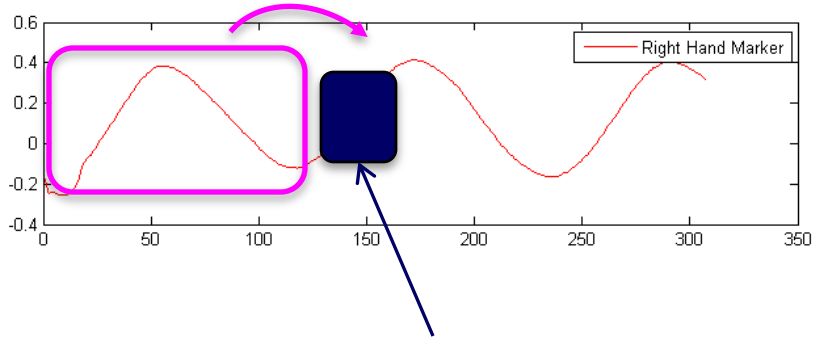
DynaMMo: Intuition

Position of Left hand marker



Recover using Dynamics temporal moving pattern

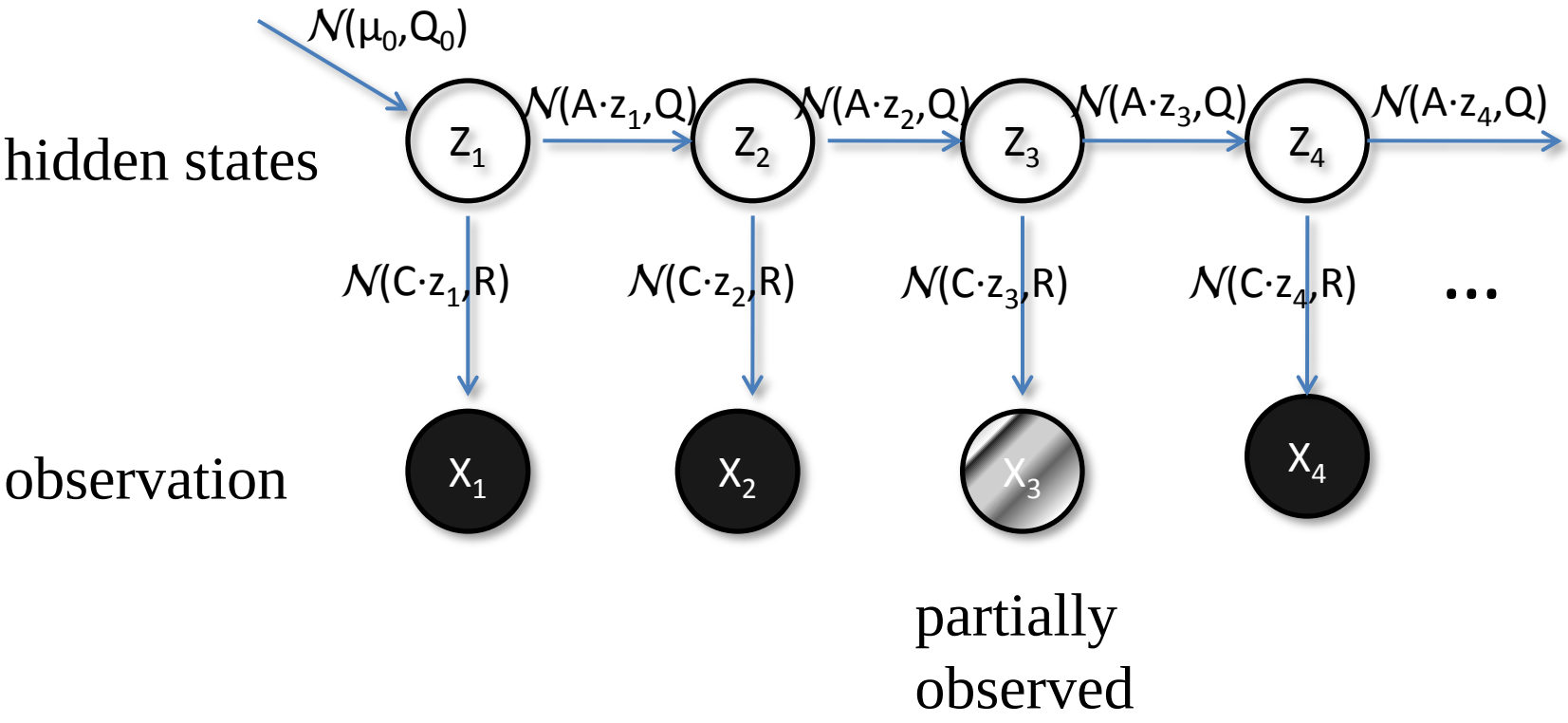
Position of right hand marker



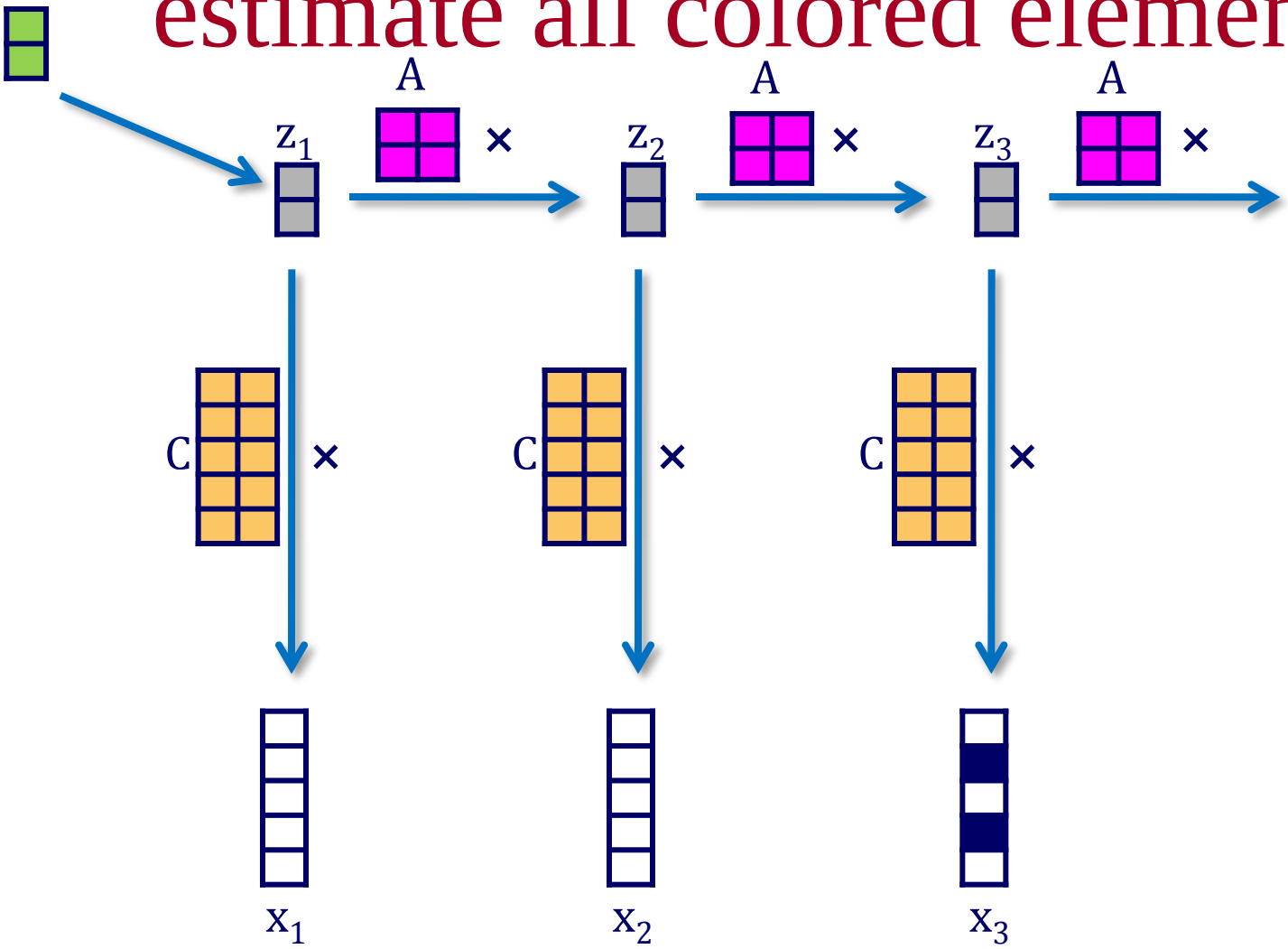
missing

[Lei Li et al DynaMMo. KDD 2009]

LDS with Missing Value

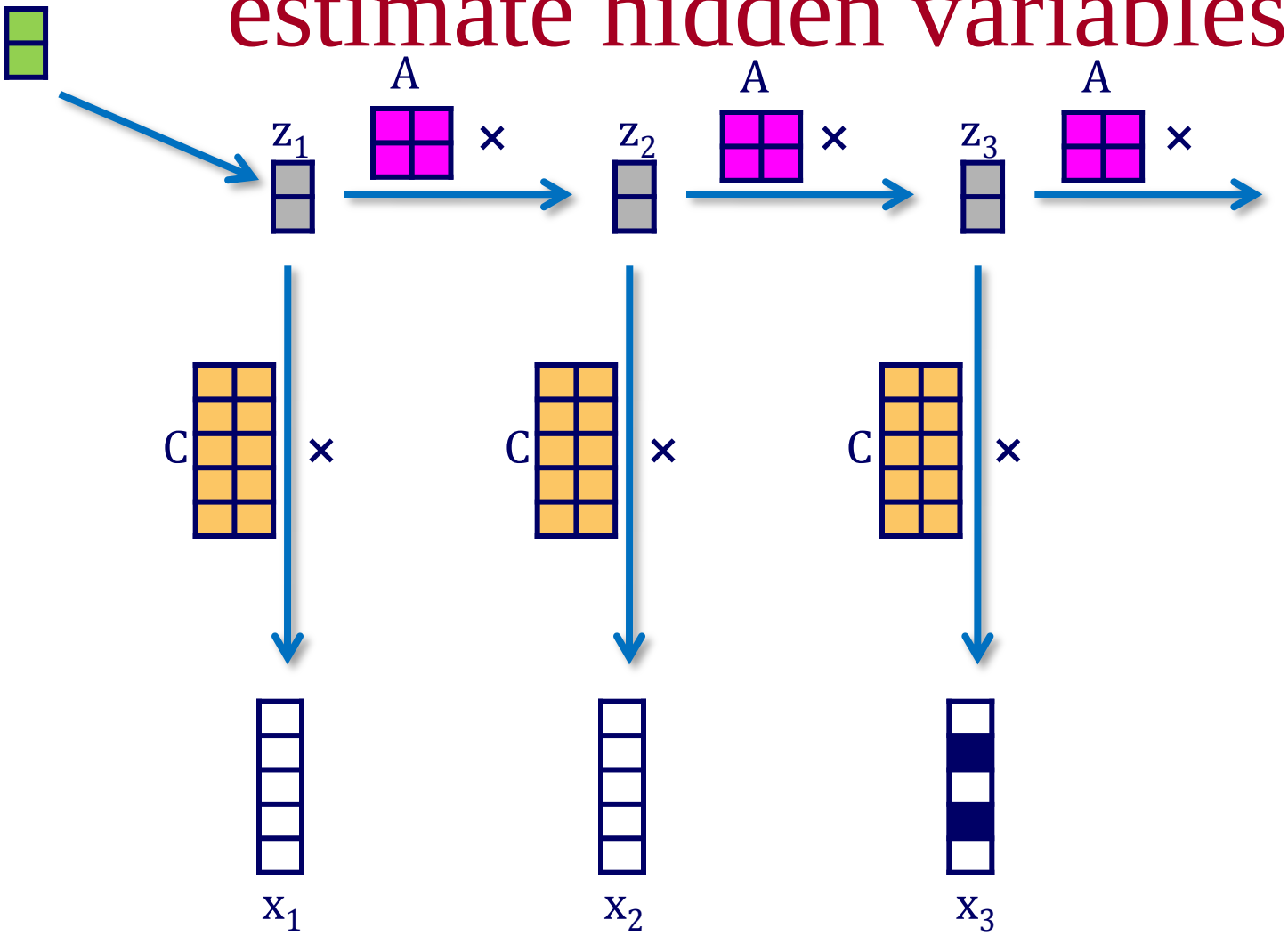


DynaMMo Illustration: estimate all colored elements



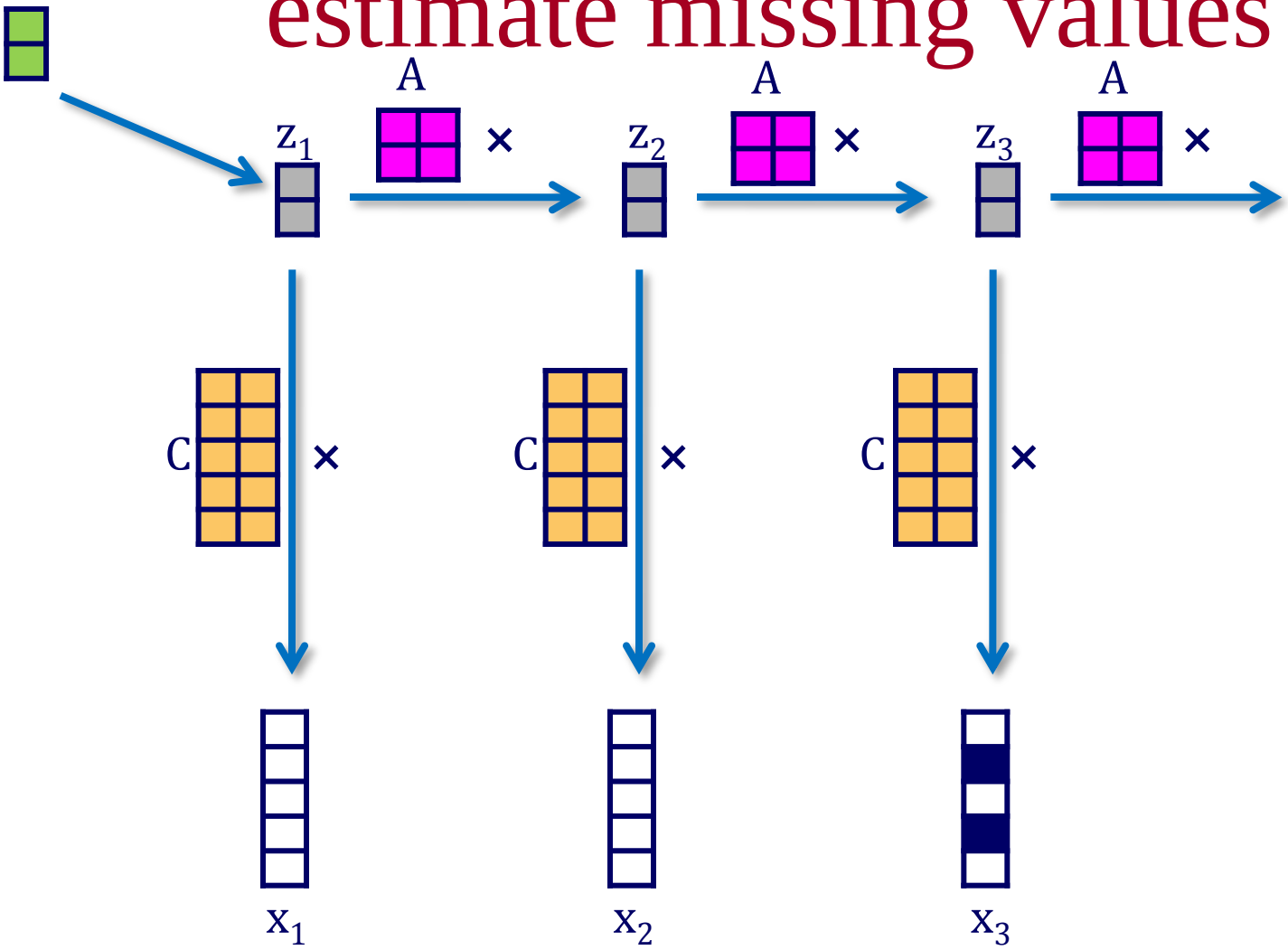
Details in [Li+2009]

DynaMMo Illustration: step 1 estimate hidden variables



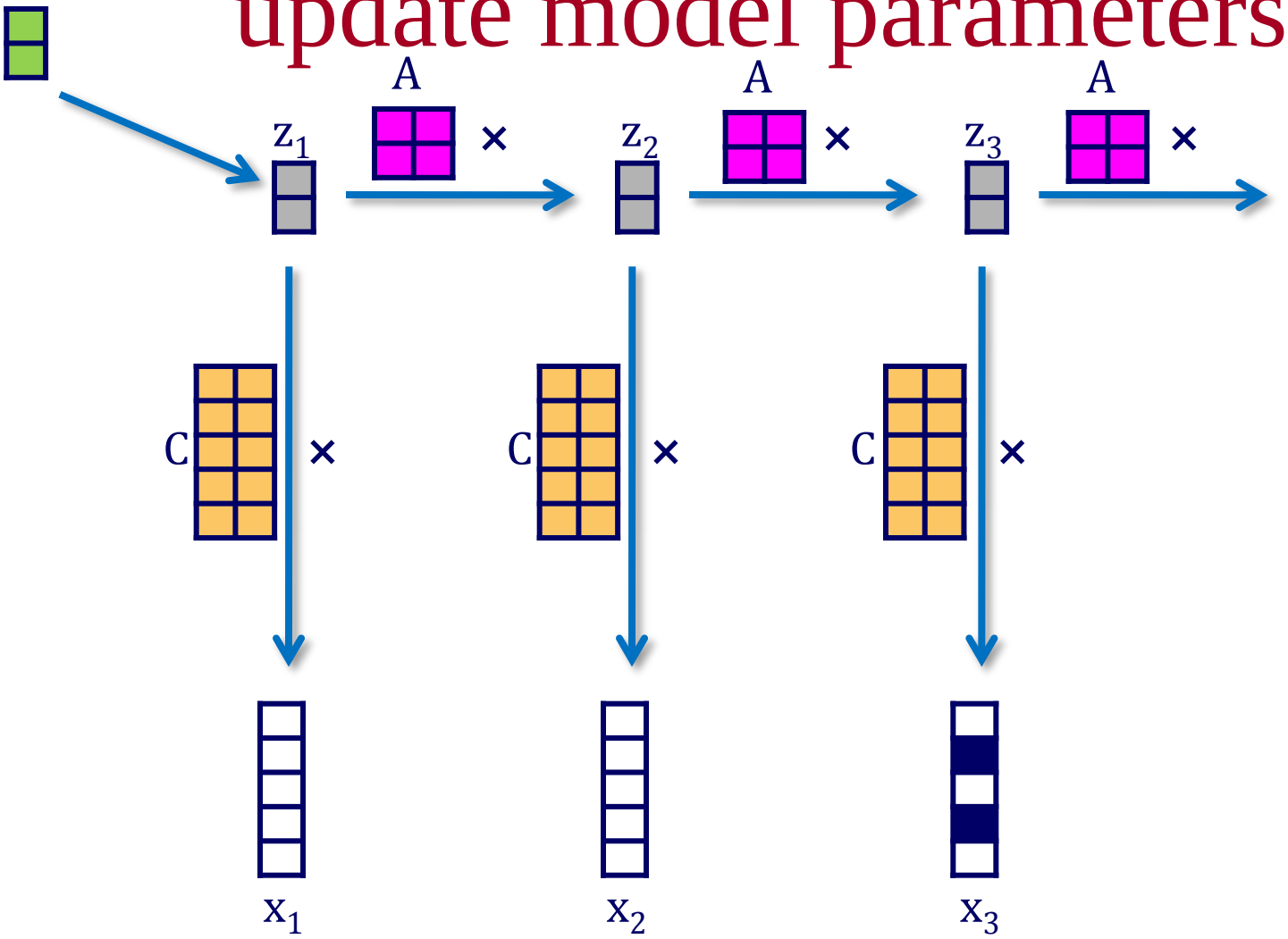
Details in [Li+2009]

DynaMMo Illustration: step 2 estimate missing values



Details in [Li+2009]

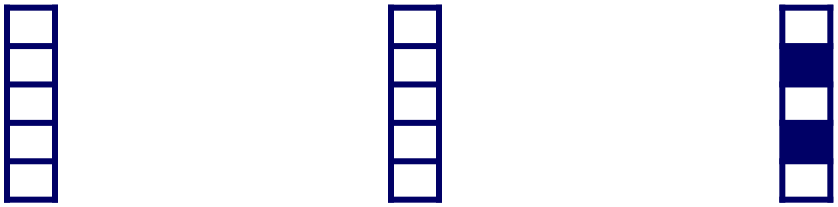
DynaMMo Illustration: step 3 update model parameters



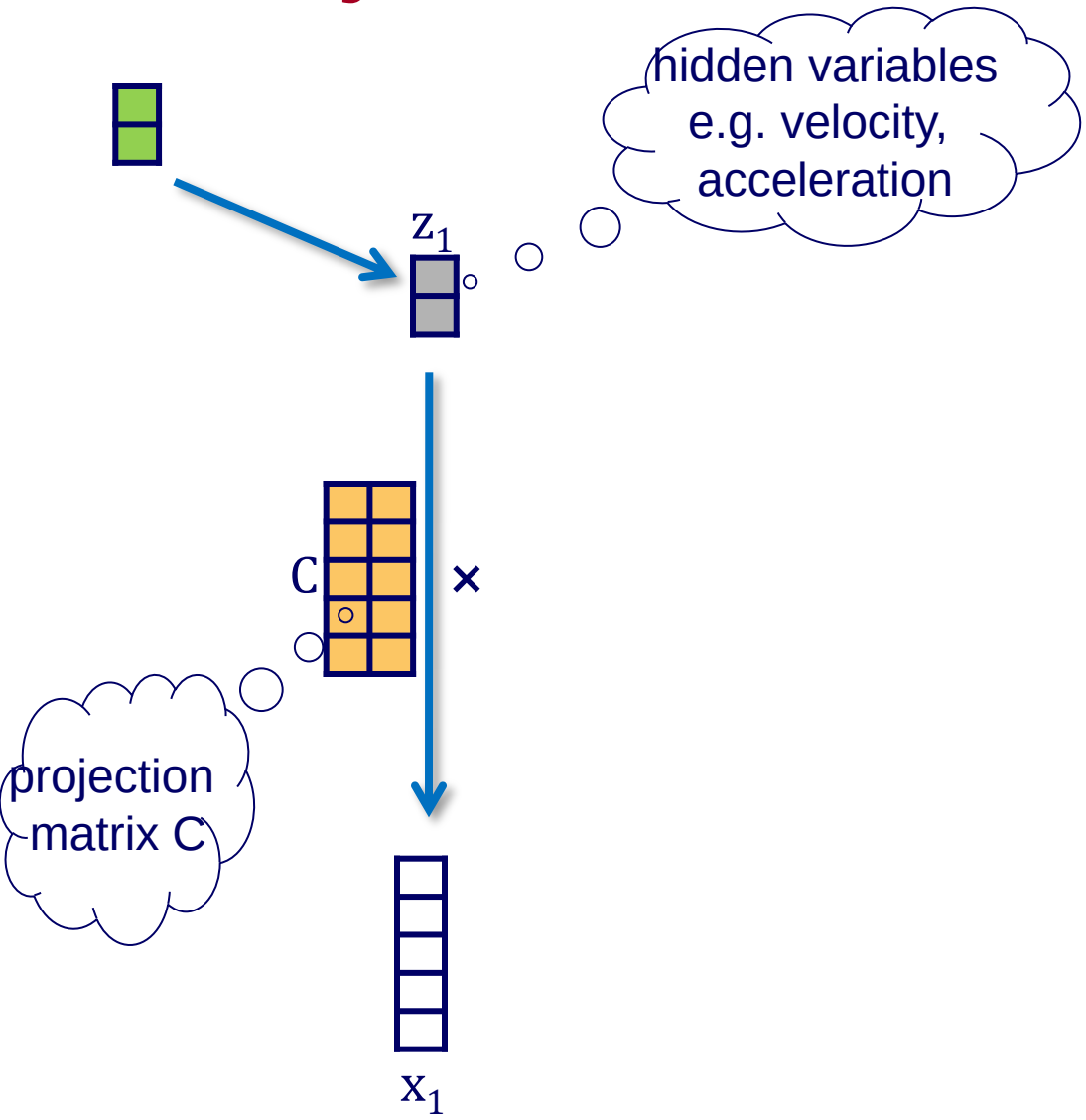
Details in [Li+2009]

DynaMMo Intuition, forward-backward algorithm

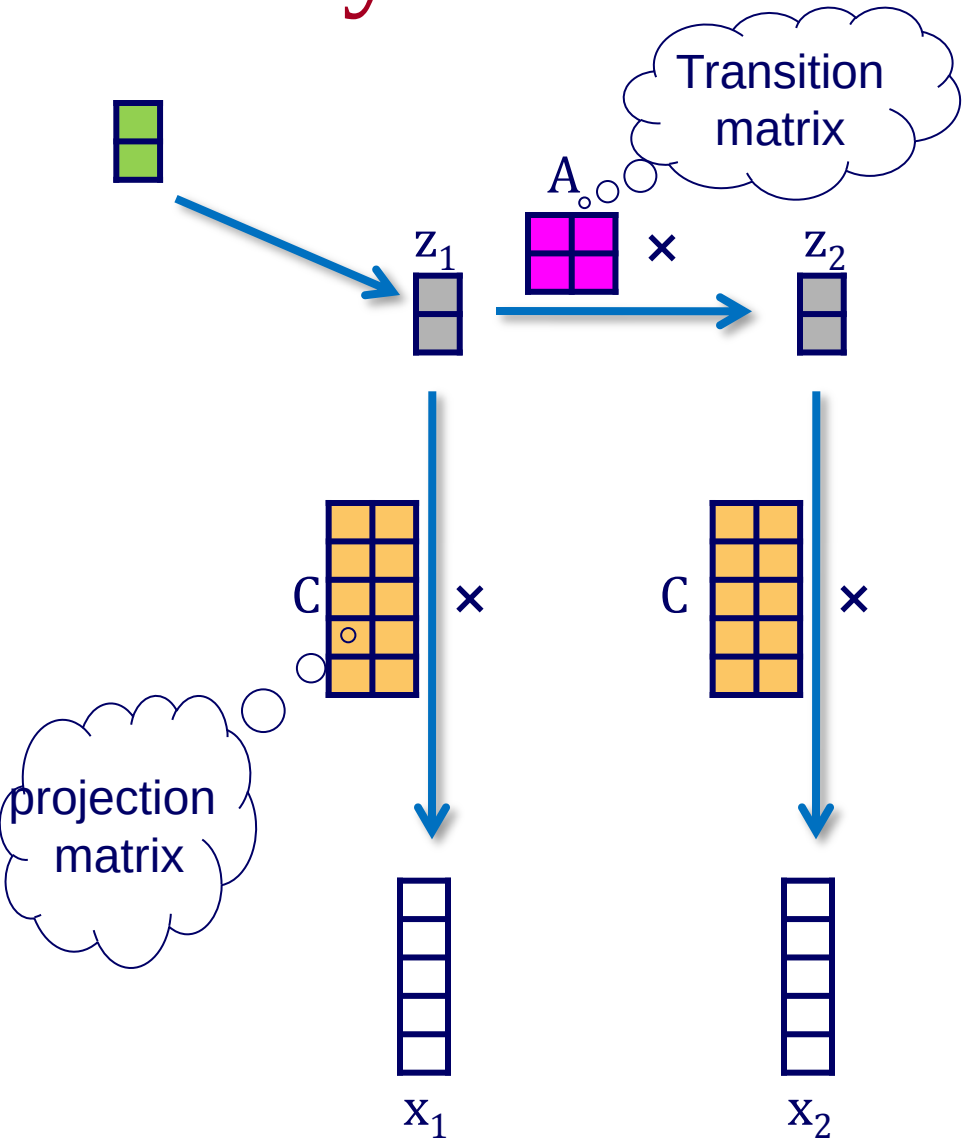
- How to recover the missing values?



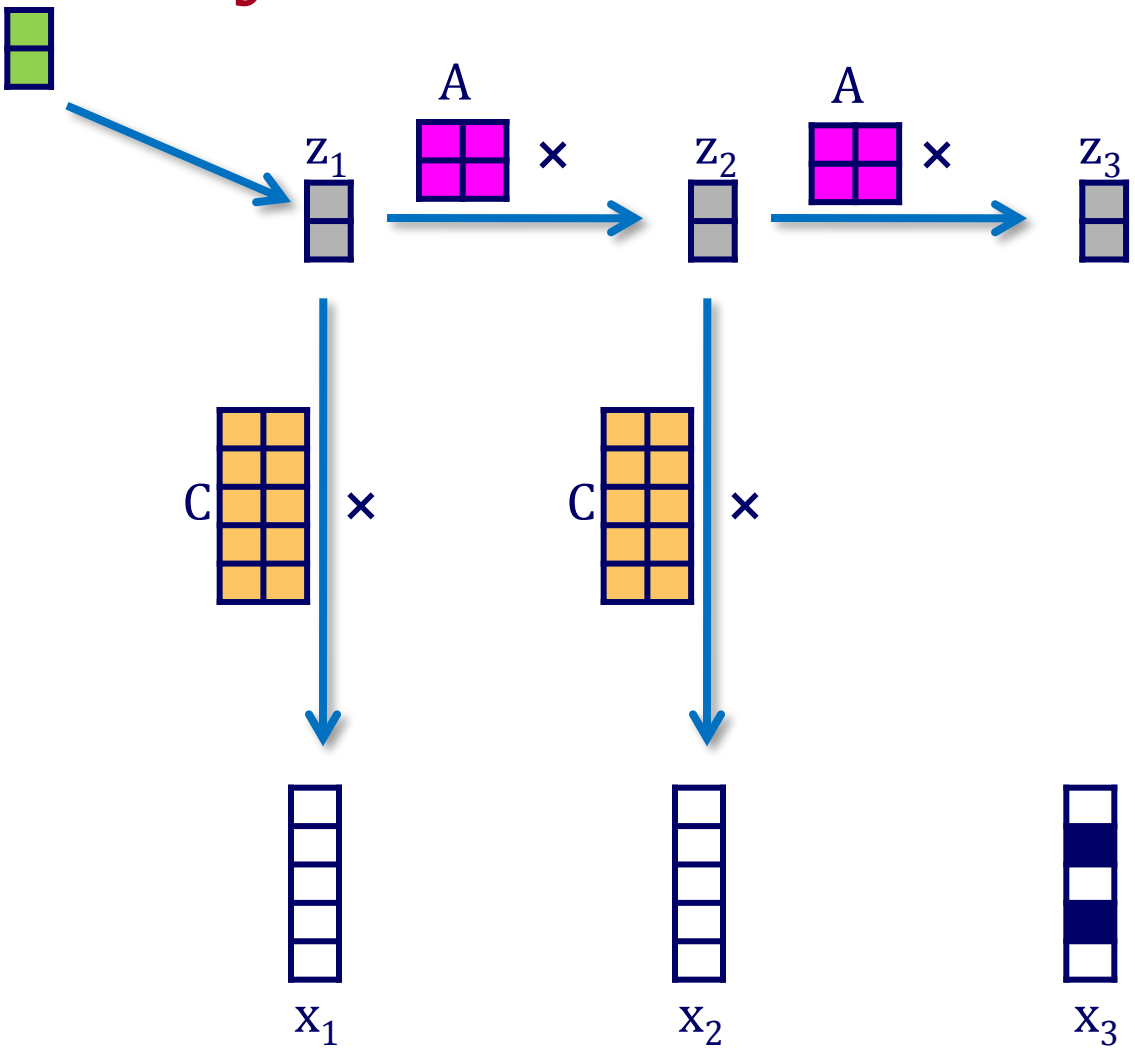
DynaMMo: How to Recover?



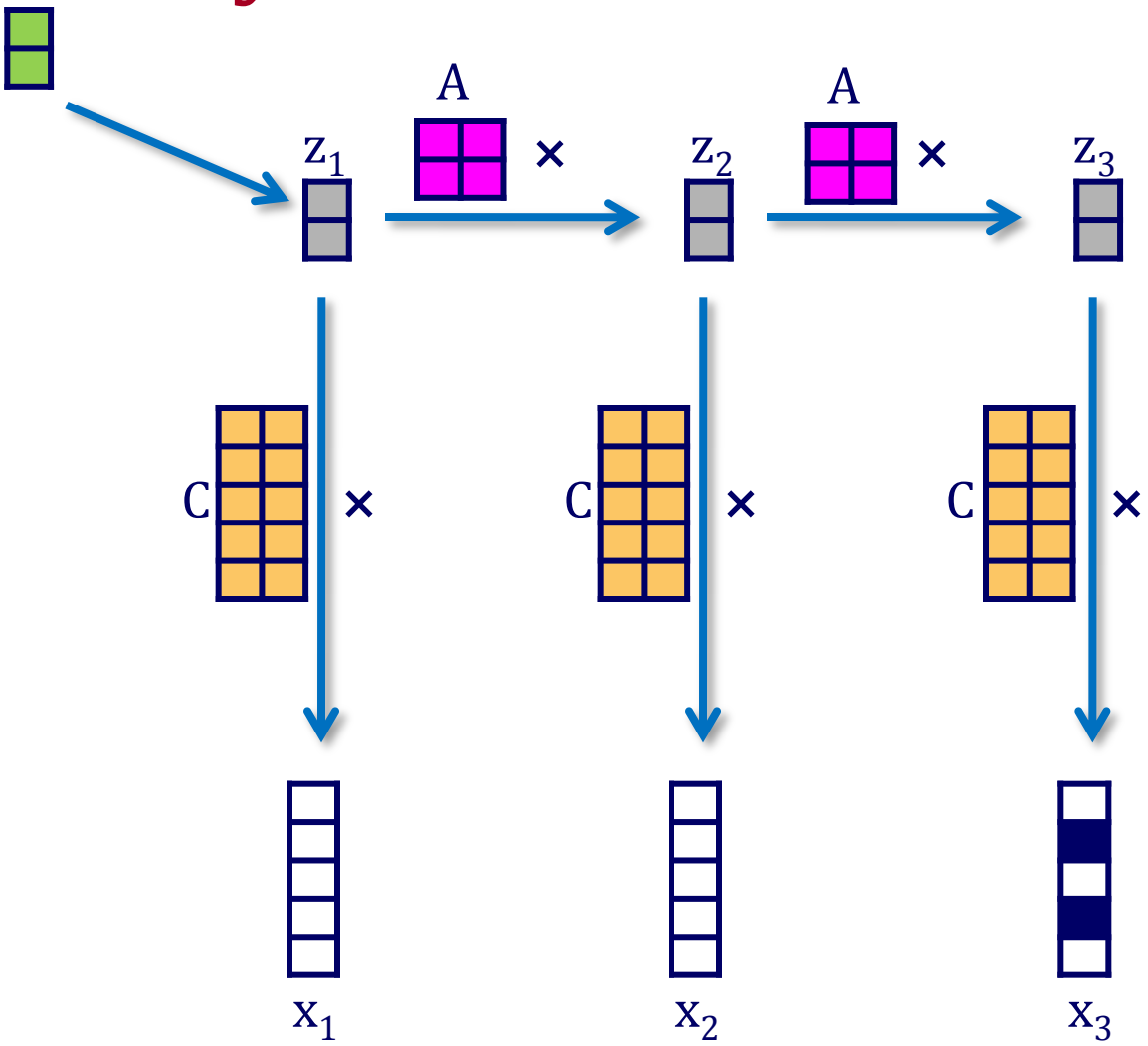
DynaMMo: How to Recover?



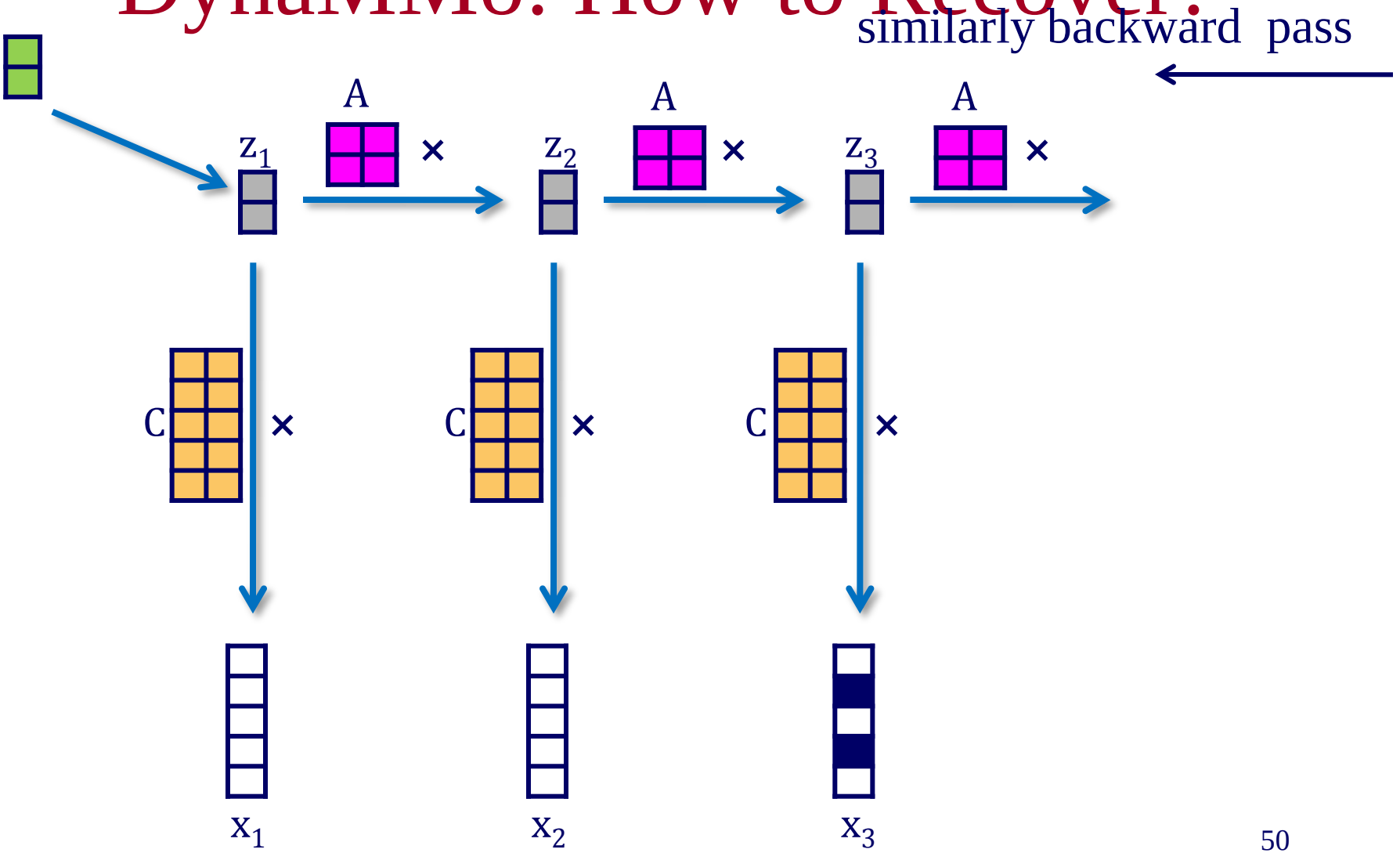
DynaMMo: How to Recover?



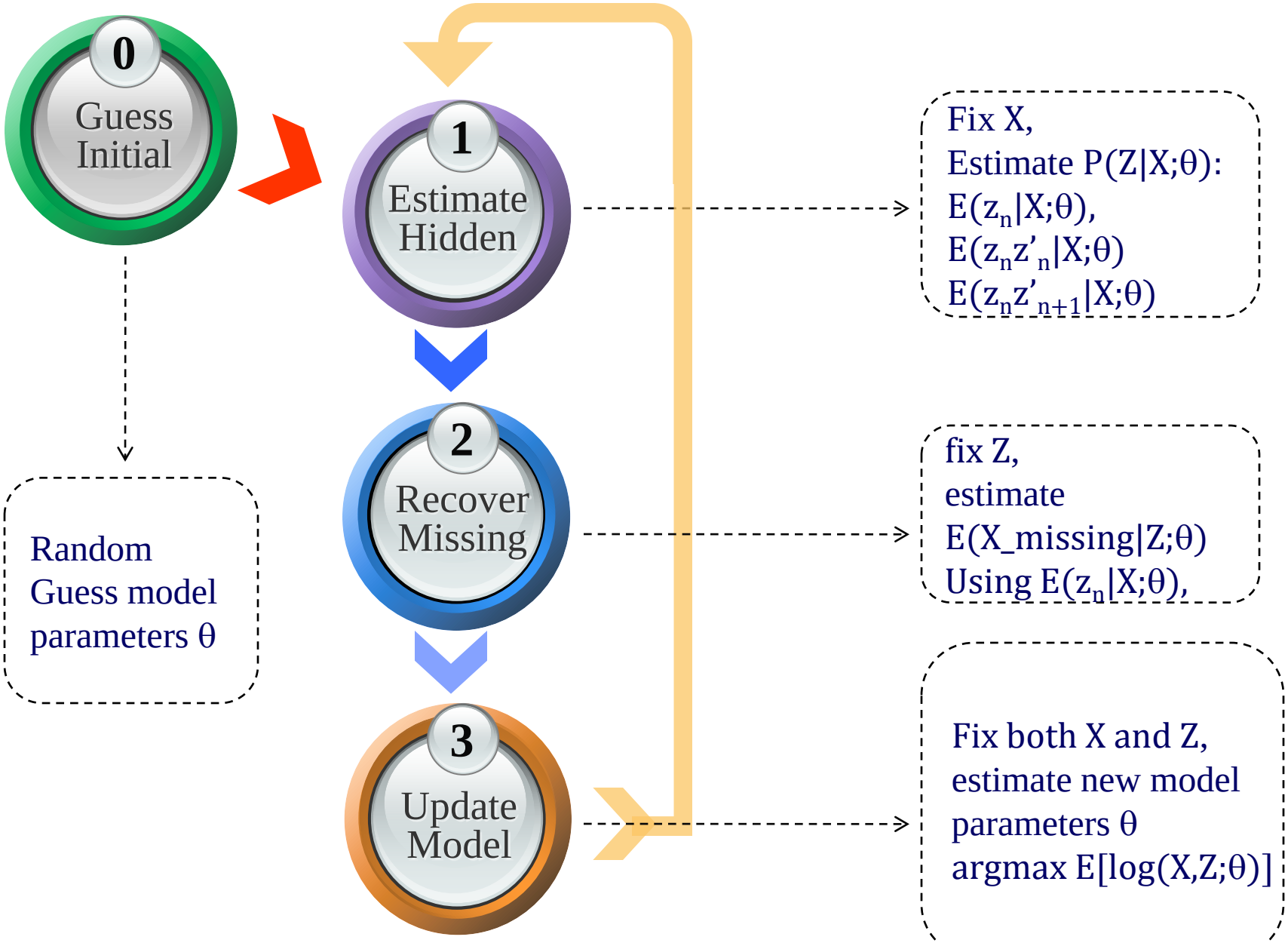
DynaMMo: How to Recover?



DynaMMo: How to Recover?

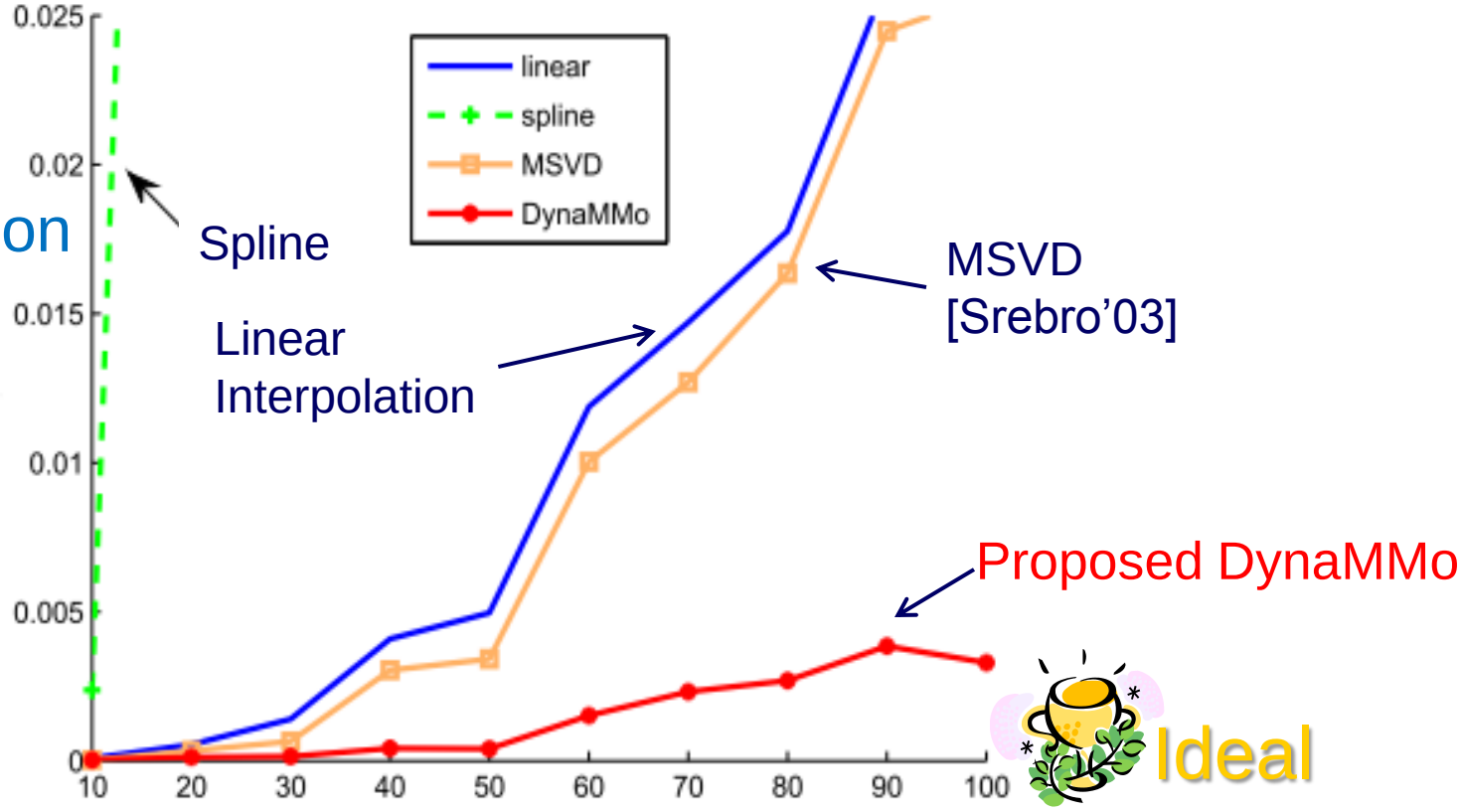


DynaMMo Learning(details)



Result – Better Missing Value Recovery

Reconstruction error



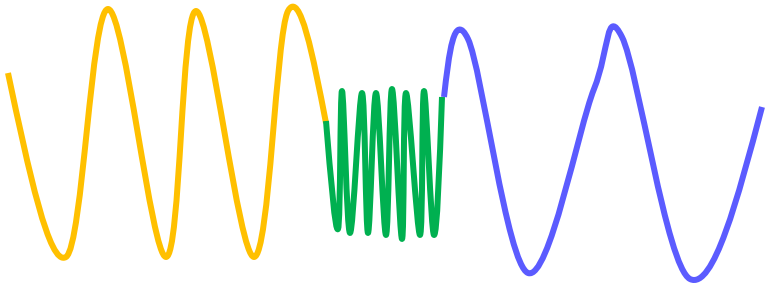
Dataset:
CMU Mocap #16
mocap.cs.cmu.edu

more results in [Li+2009]

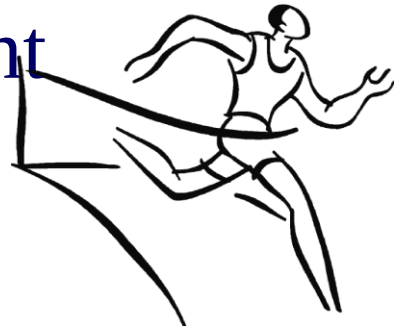
Outline

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

Switching LDS: Intuition

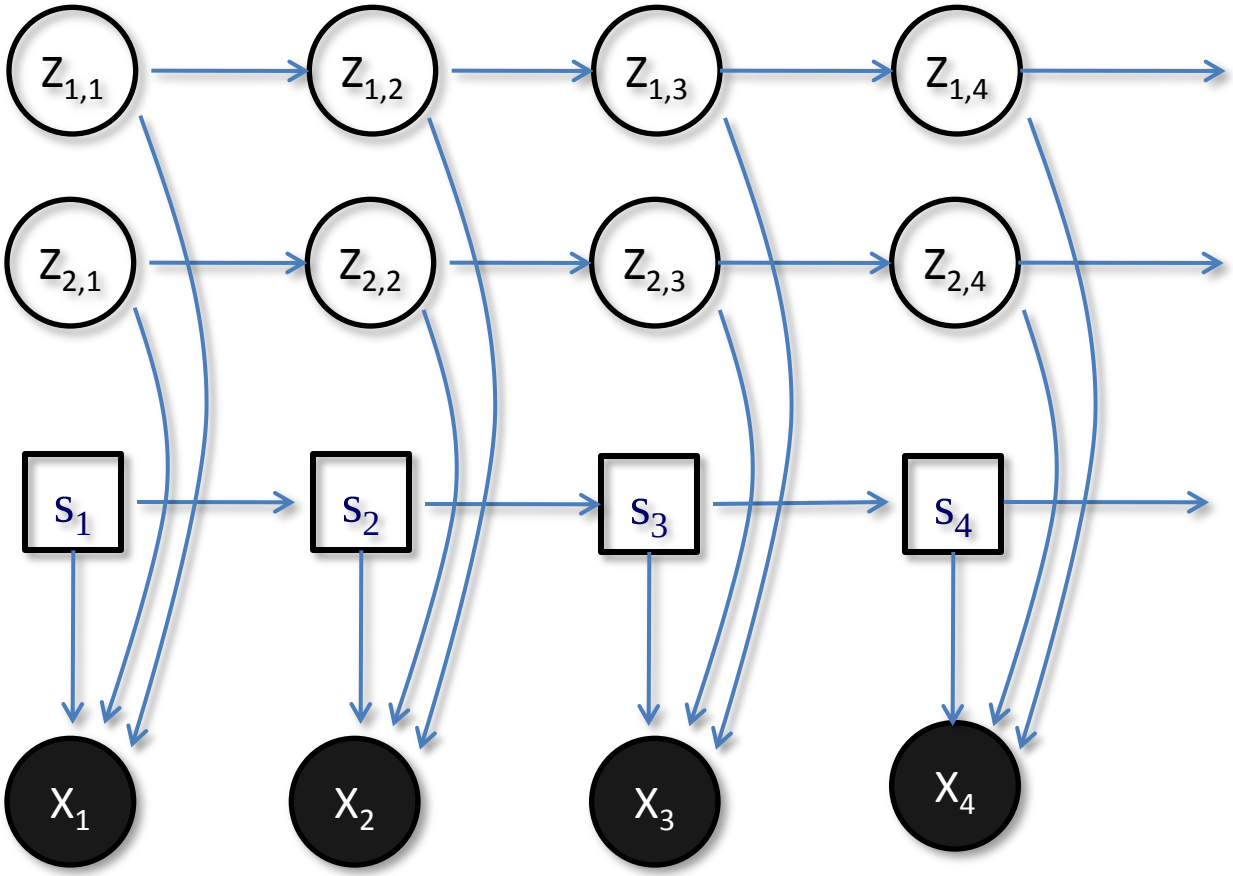


- Tracking human motion
  - Start from slow walking
  - Transition to running for a while
  - Gradually stop
- Each part corresponds to a different state & dynamics



Switching LDS

S={running, walking}



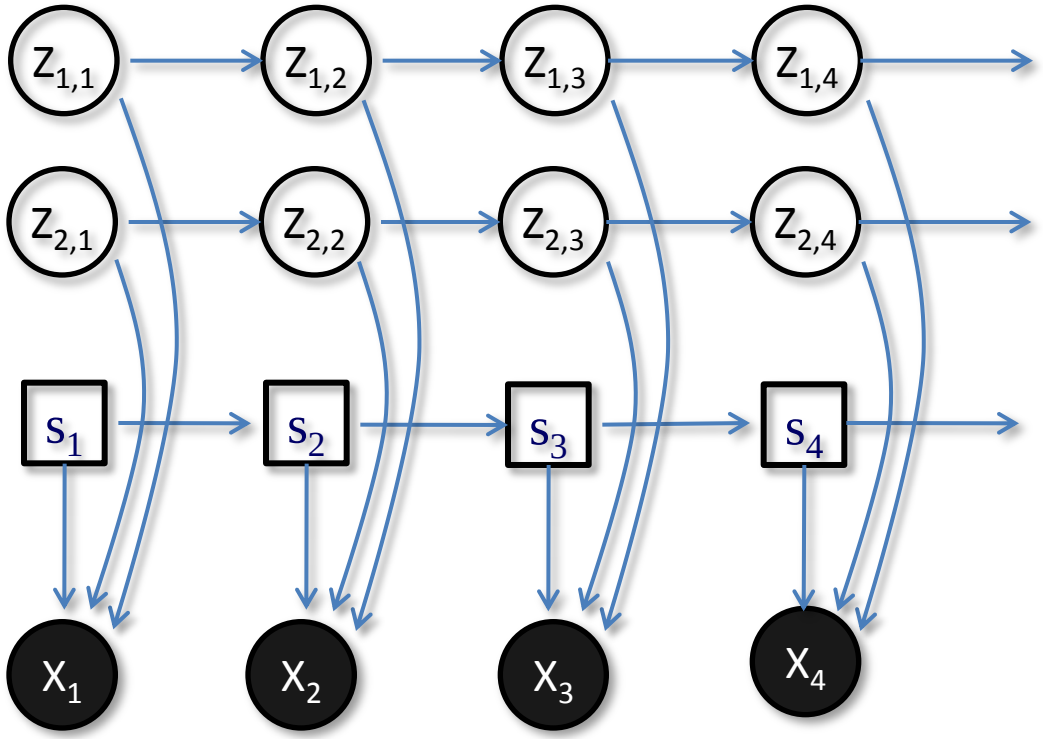
Switching LDS: example

S=1, walking

$A=A_1$, $C=C_1$

S=2, running

$A=A_2$, $C=C_2$



Switching LDS in Penalty Kick

Before touching ball

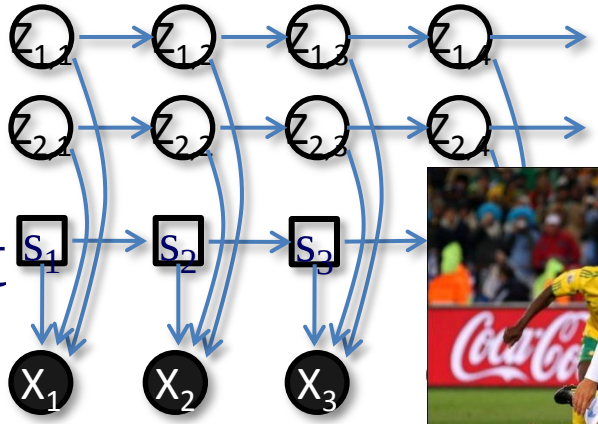
S=1, running towards left

$A=A_1$, $C=C_1$

After hitting ball

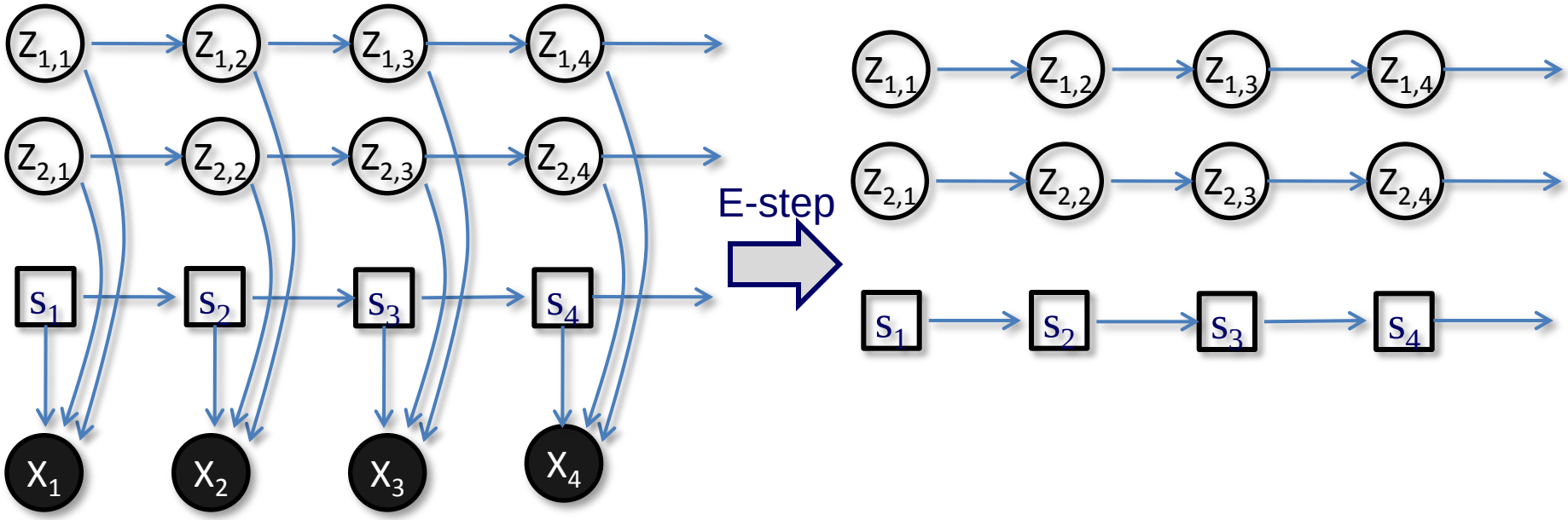
S=2, kicking towards right

$A=A_2$, $C=C_2$



From FIFA 2010

Estimating Parameters for SLDS



Approximate inference: Variational EM

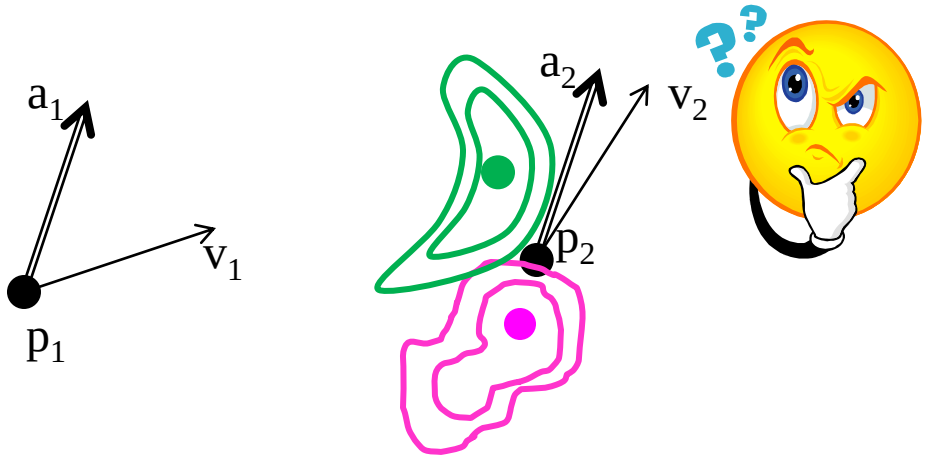
[Ghahramani&Hinton. Variational learning for switching state-space models.2000]

Outline

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

Particle Filters

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



[Gordon et al, Nonlinear/non-gaussian bayesian state estimation, 1993]

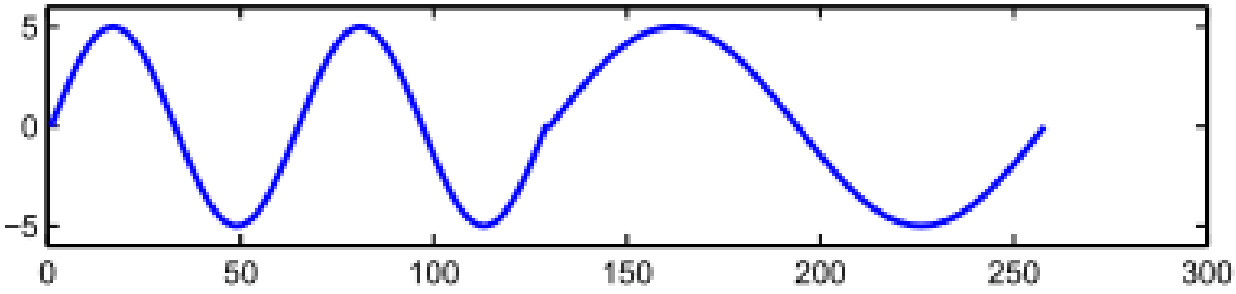
Outline

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

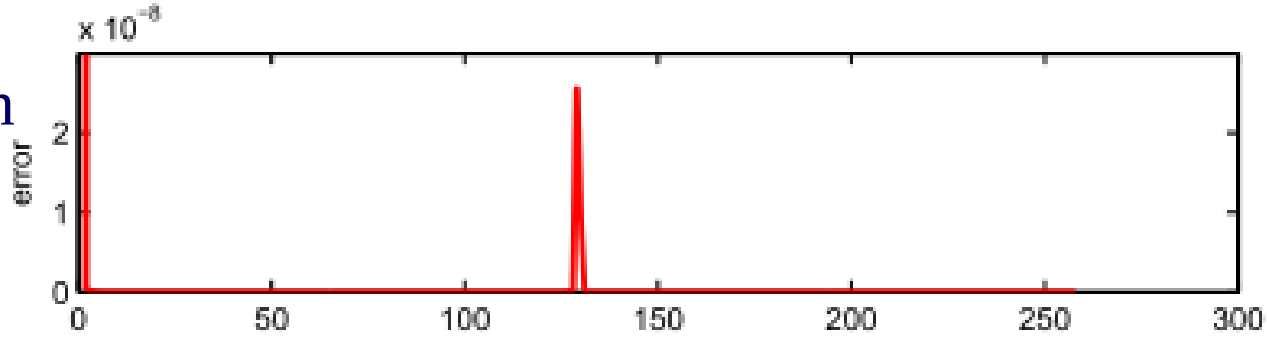
How to Segment

- Segment by threshold on reconstruction error

original data



reconstruction error



[Lei Li et al DynaMMo. KDD 2009]

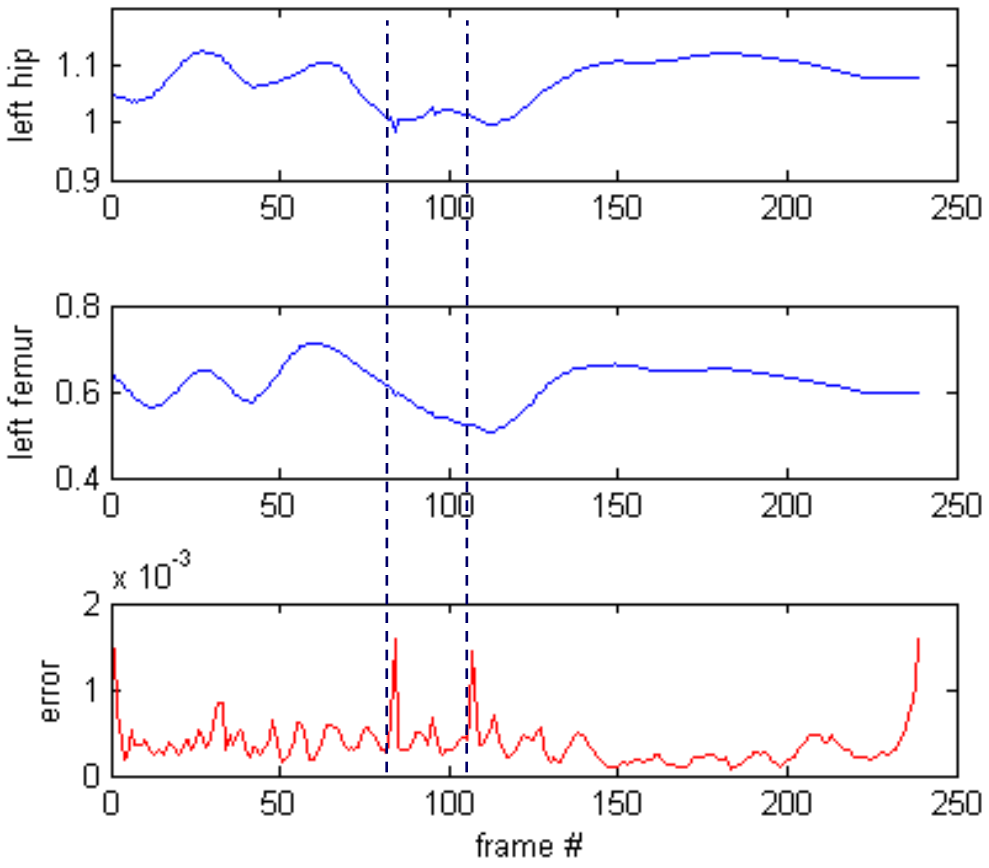
Results – Segmentation

- Find the *transition* during "running" to "stop".

left hip

left femur

reconstruction error



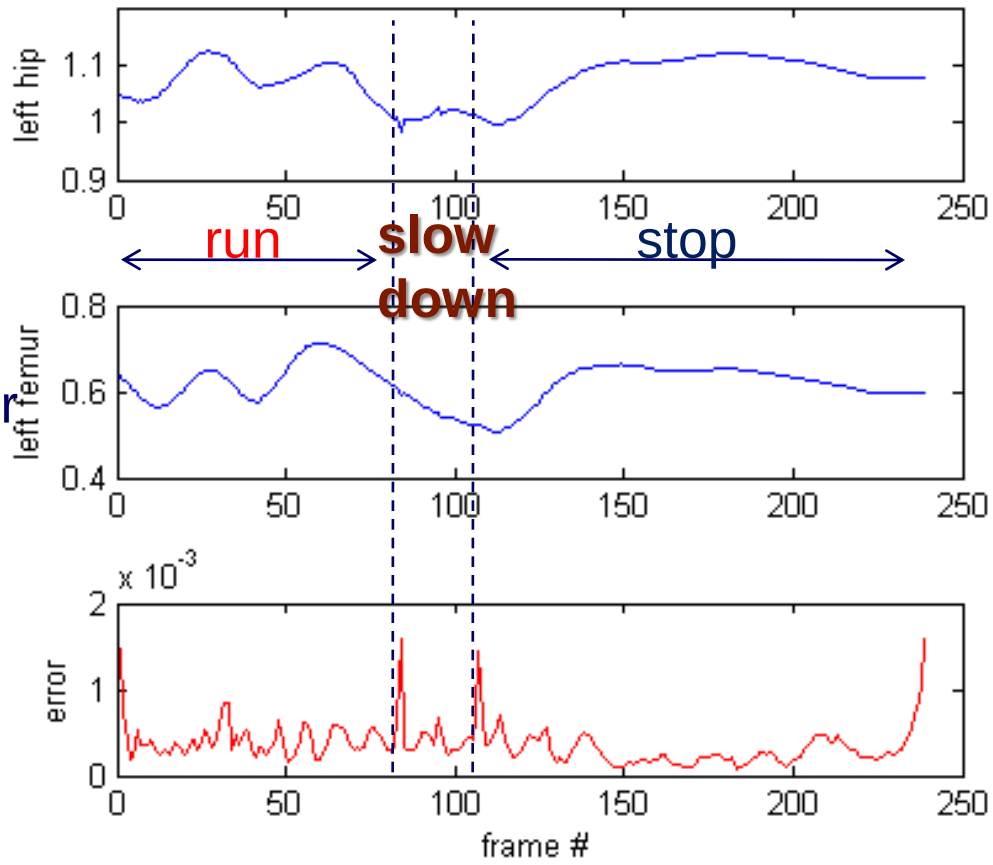
Results – Segmentation

- Find the *transition* during "running" to "stop".

left hip

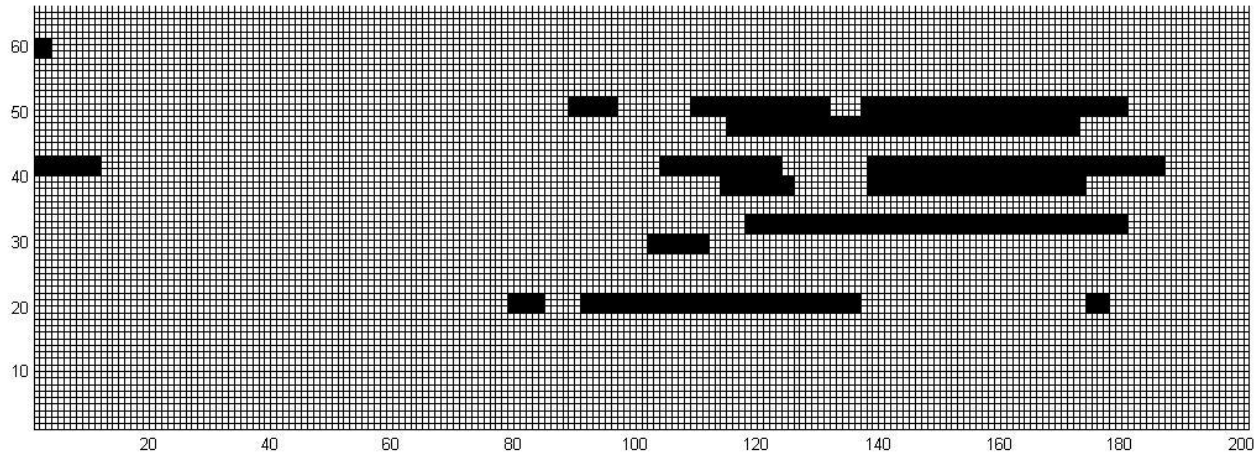
left femur

reconstruction error



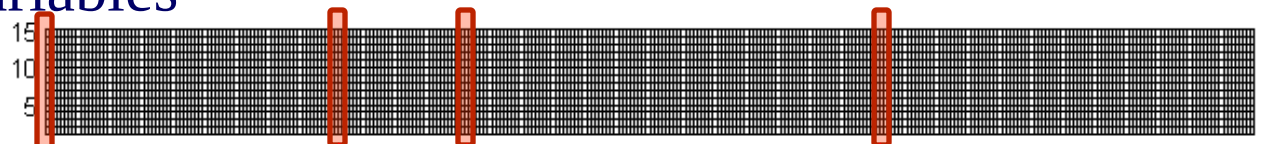
How to Compress (DynaMMo)

Original data w/ missing values

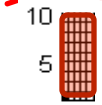


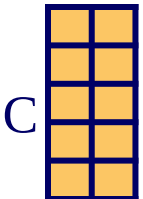
DynaMMo

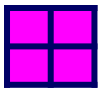
hidden variables



keep only a portion
(optimal samples)







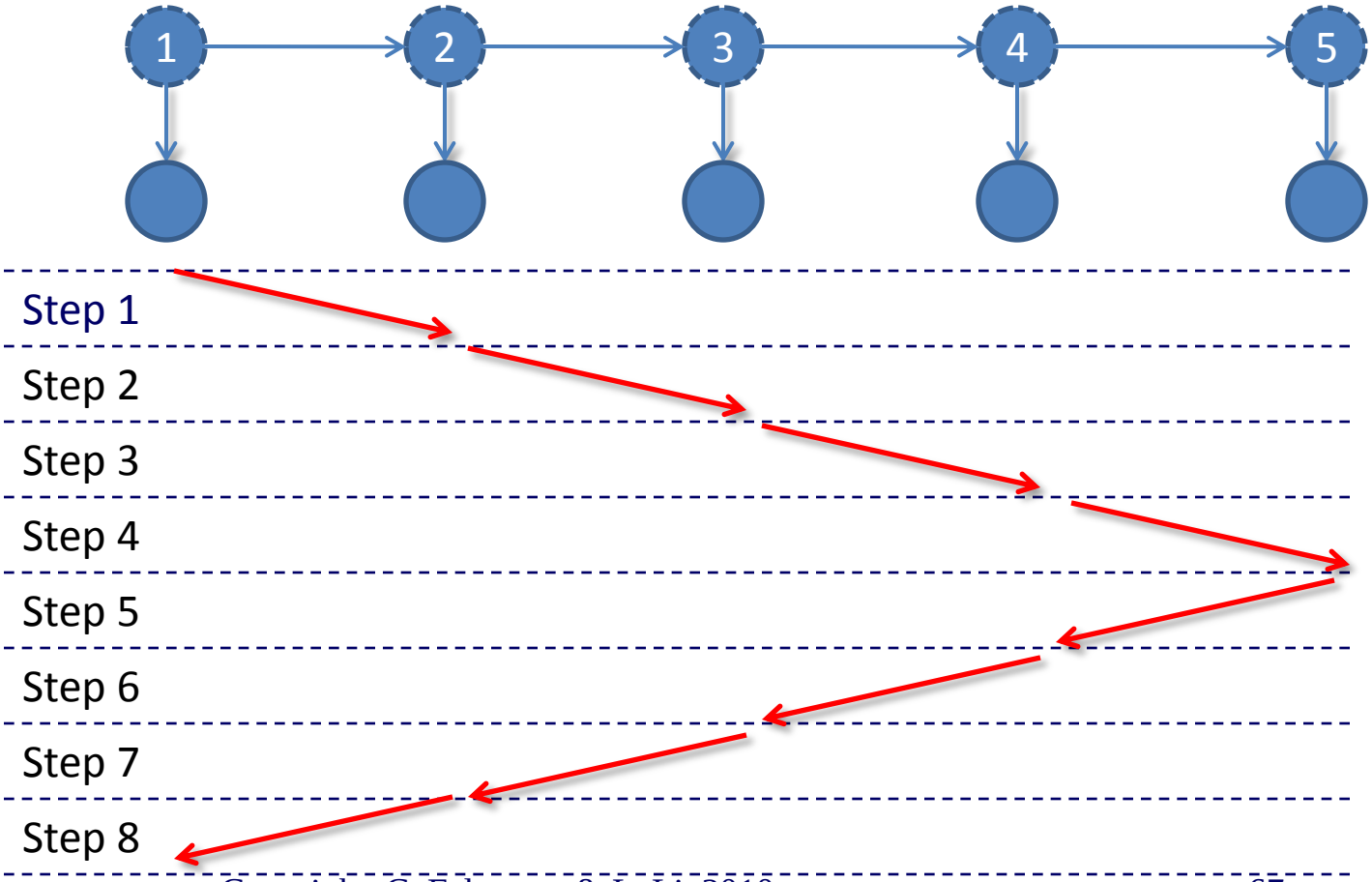
Outline

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

Challenge illustration Expectation-Maximization Alg.

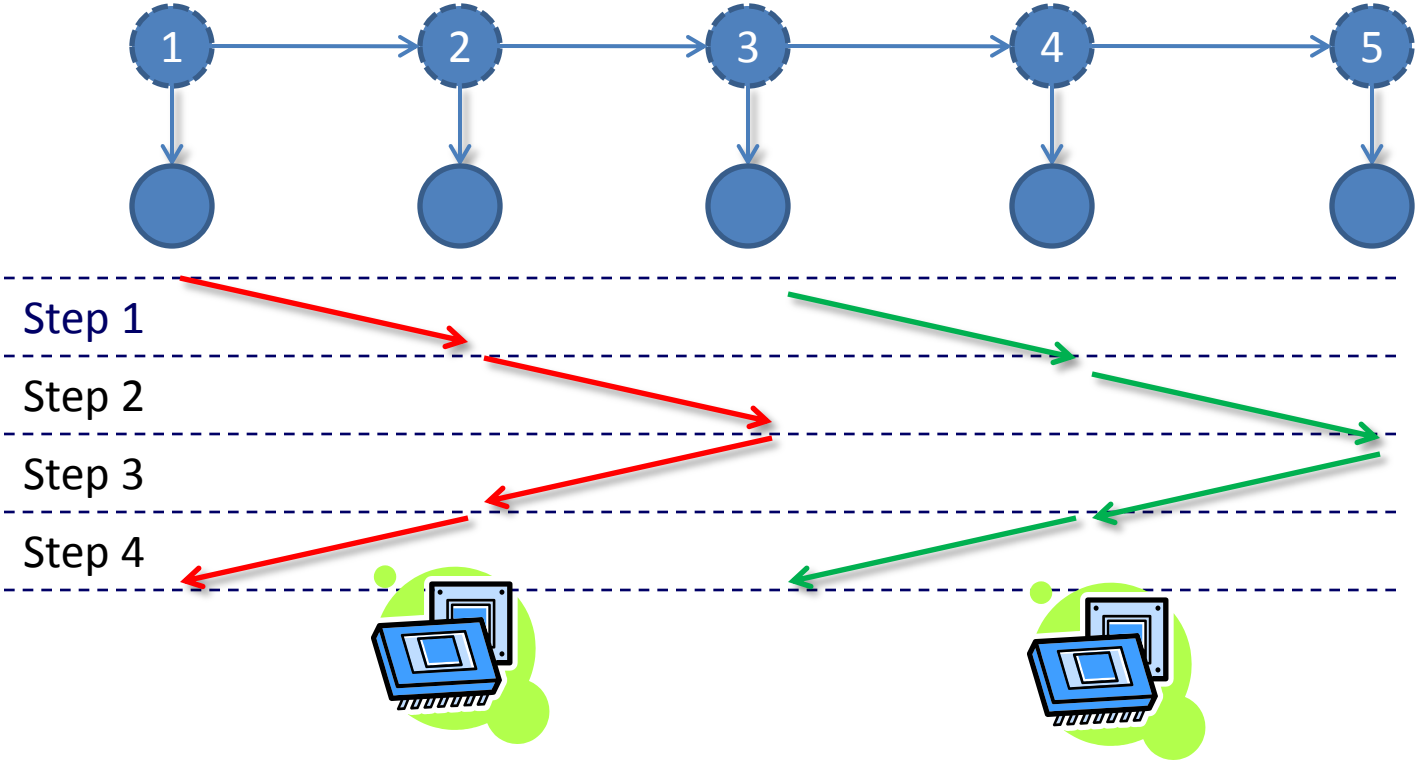
Timeline for E-step (forward-backward) in learning LDS

EM can only use single CPU due to data dependency



Parallel learning by Cut-And-Stitch Method

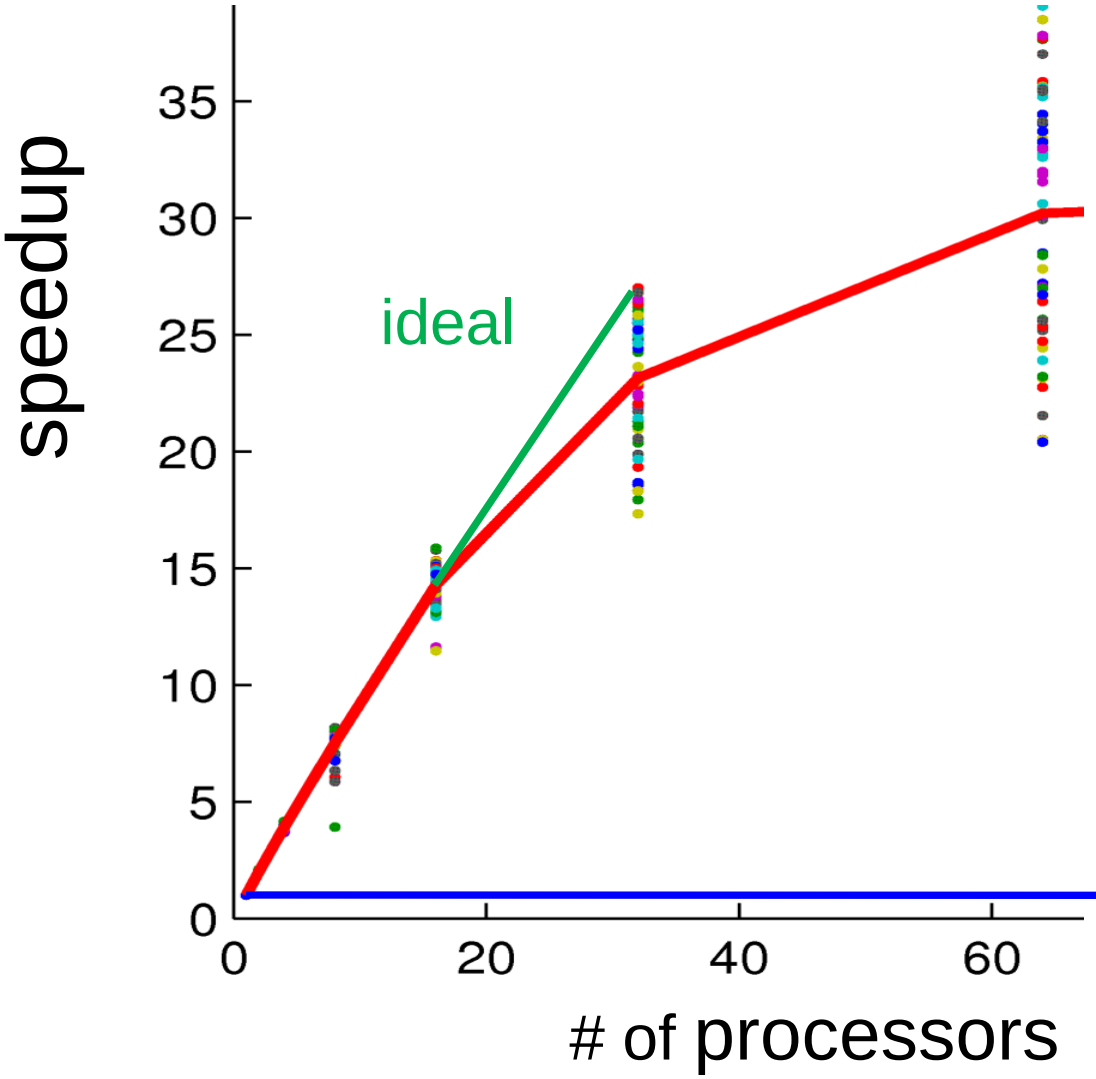
Goal:
with 2 CPUs



[Li et al 2008b]

[Li et al Cut-And-Stitch. KDD 2008]

It works!



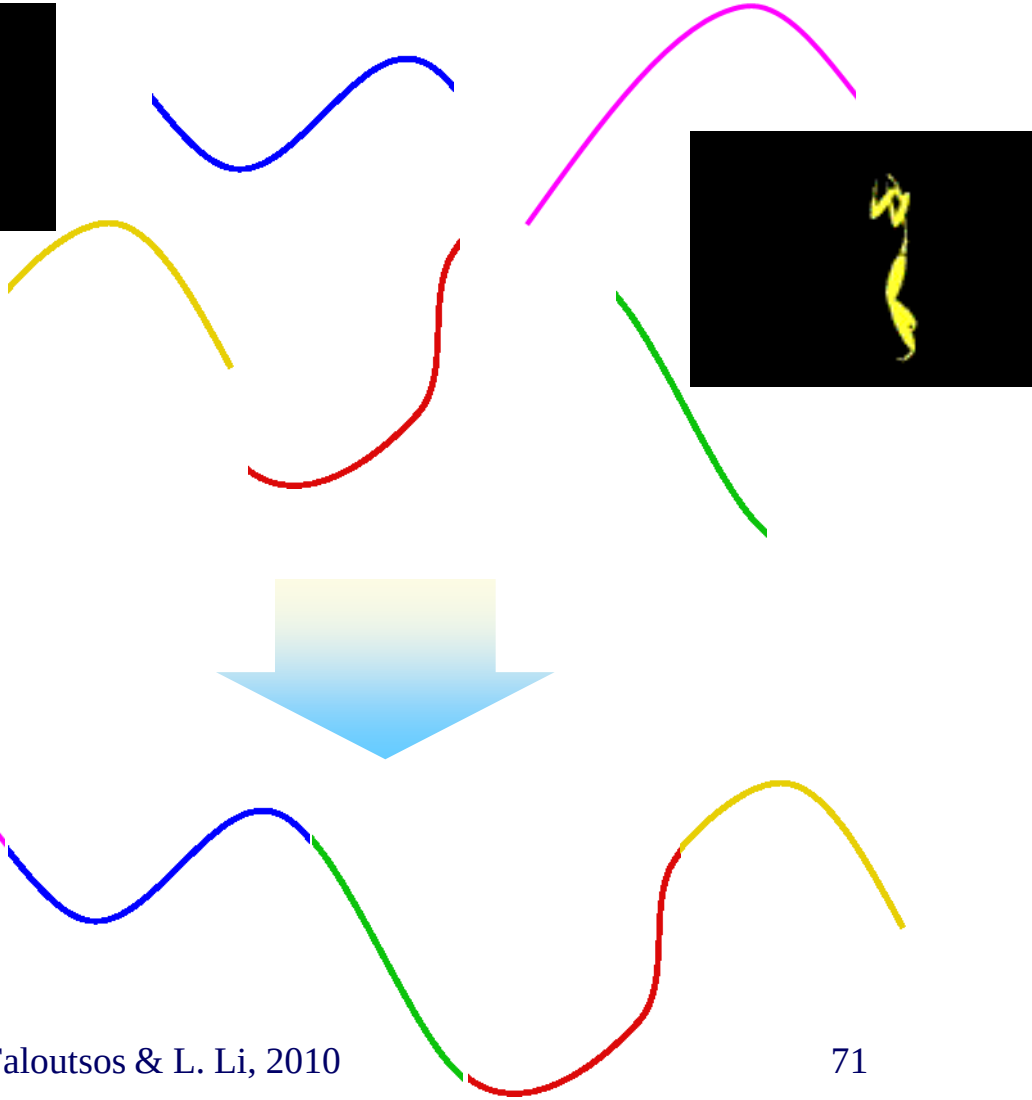
Dataset:
58 motion sequences
CMU Mocap #16
mocap.cs.cmu.edu,
tested on NCSA super computer,

Outline

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

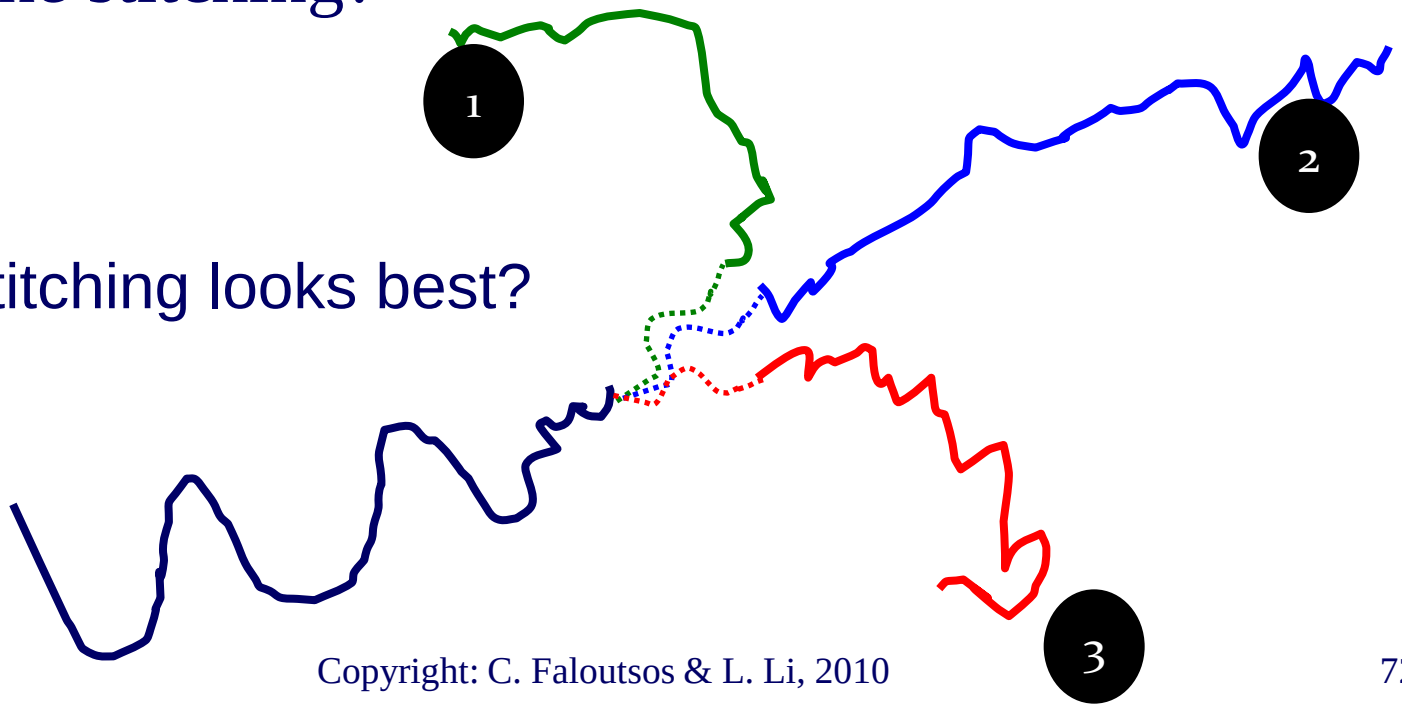
Motion Stitching
A Database Approach

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



Problem Definition

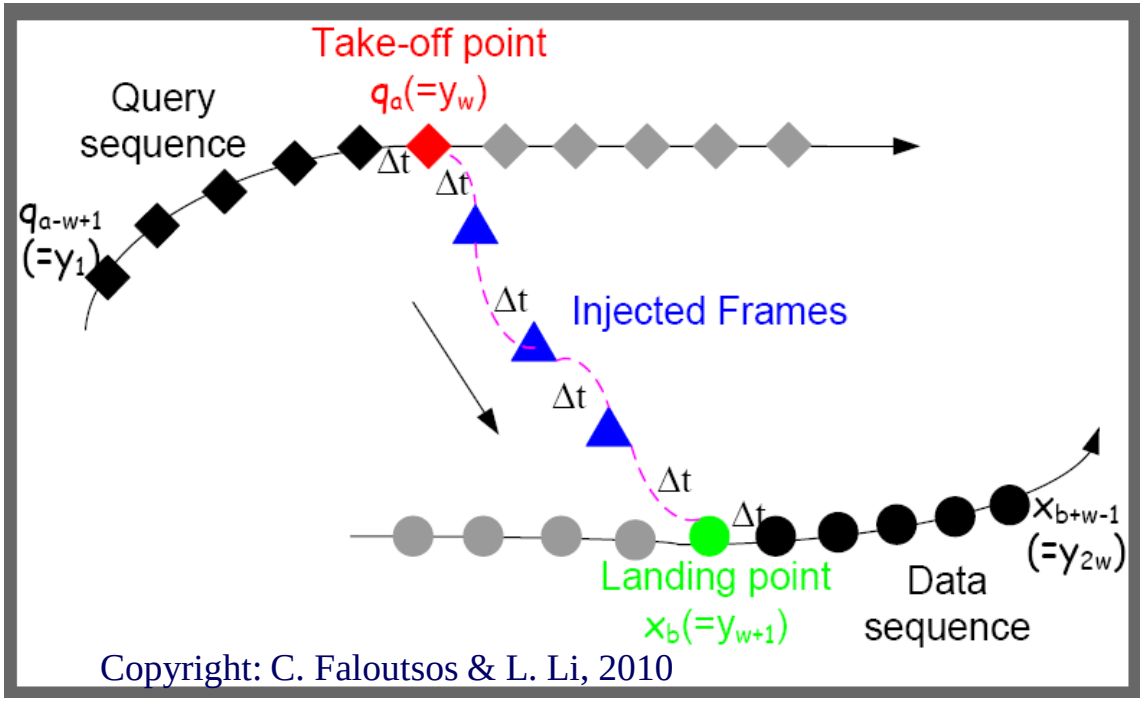
- Given two motion-capture sequences that are to be stitched together, how can we assess the goodness of the stitching?



Which stitching looks best?

Proposed Method: Laziness Score [Li+2008a]

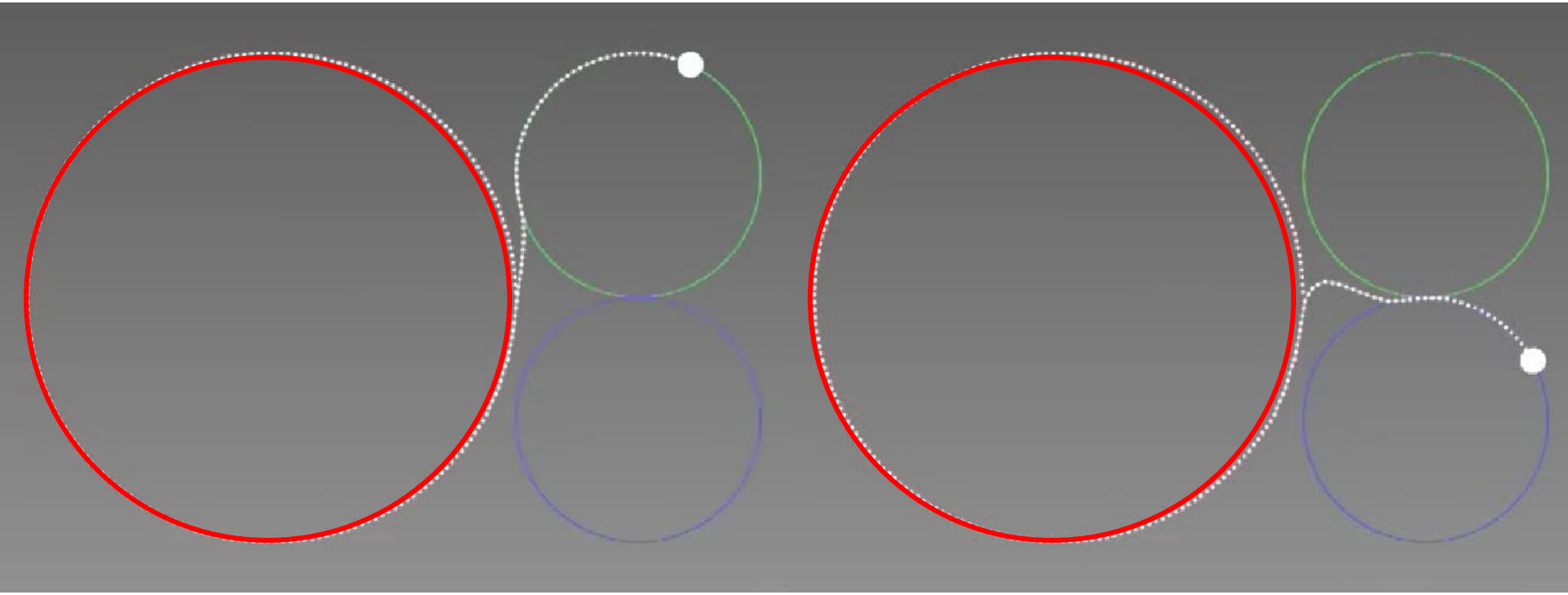
- Conjecture: *less human effort → more natural*
- Proposed: use Kalman filters to estimate position, velocity, acceleration → Compute effort/ energy



Which continues to? Green or Blue?

straight moving

U-Turn



Result – Laziness-score prefers straightforward moving

straight moving

U-Turn

[Li et al Laziness Score. Eurographics 2008]

Conclusion

- Intuition, example, and definition
  - Original kalman filter (known parameters)
    - Kalman filtering
    - Kalman smoothing
  - Kalman filters with parameter estimation (EM)
- Extensions
  - Handling missing values
  - Switching linear dynamical
  - Particle filters (MCMC sampling)
- Kalman filters at work
  - Segmentation & compression
  - Parallel learning
  - Motion stitching

References

- Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for linear dynamical systems. 1996
- Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. Neural Computation, 12(4):831–864, 2000.
- N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. IEE Proceedings F Radar and Signal Processing, 140(2):107–113, 1993.
- R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the em algorithm. Journal of Time Series Analysis, 3:253–264, 1982.
- R. E. Kalman. A new approach to linear filtering and prediction problems. Transactions of the ASME ÂšC Journal of Basic Engineering, (82 (Series D)):35–45, 1960.

References (cont')

- H. Rauch, F Tung, and C T Striebel. Maximum likelihood estimates of linear dynamic systems. AIAA Journal, pages 1445–1450, 1965.
- Lei Li, Wenjie Fu, Fan Guo, Todd C Mowry, and Christos Faloutsos. Cut-and-stitch: efficient parallel learning of linear dynamical systems on smps. In Proceedings of the 14th ACM SIGKDD, pages 471–479, 2008.
- Lei Li, James McCann, Christos Faloutsos, and Nancy Pollard. Laziness is a virtue: Motion stitching using effort minimization. In Short Papers Proceedings of EUROGRAPHICS, 2008.
- Lei Li, James McCann, Nancy S Pollard, and Christos Faloutsos. DynaMMo: mining and summarization of coevolving sequences with missing values. In Proceedings of the 15th ACM SIGKDD,, pages 507–516, 2009.

Software

- DynaMMo code (matlab) for missing value, compression & segmentation.
- Parallel learning (in C) for LDS
- http://www.cs.cmu.edu/~leili/
- http://www.cs.cmu.edu/~leili/pubs/dynammo.2.1.2.zip
- http://www.cs.cmu.edu/~leili/paralearn/paralearn.0.1.zip (running on gcc 4.2.0 above)

Part 5: Bursty traffic and multifractals

Outline

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

Outline

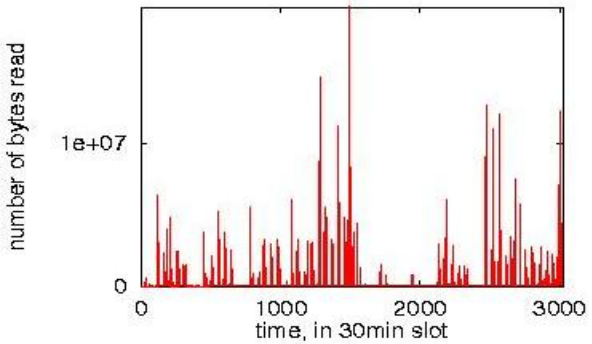
- Motivation
- ...
- Linear Forecasting
- Bursty traffic - fractals and multifractals
  - Problem
  - Main idea (80/20, Hurst exponent)
  - Results

Recall: Problem #1:

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

Find: patterns, periodicities, and/or compress

#bytes

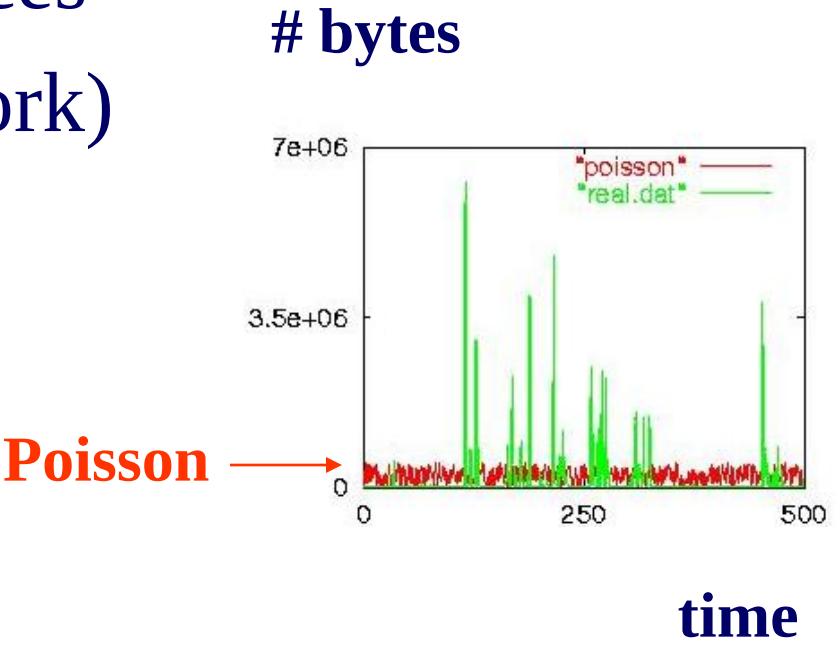


Bytes per 30'
(packets per day;
earthquakes per year)

time

Problem #1

- model bursty traffic
- generate realistic traces
- (Poisson does not work)



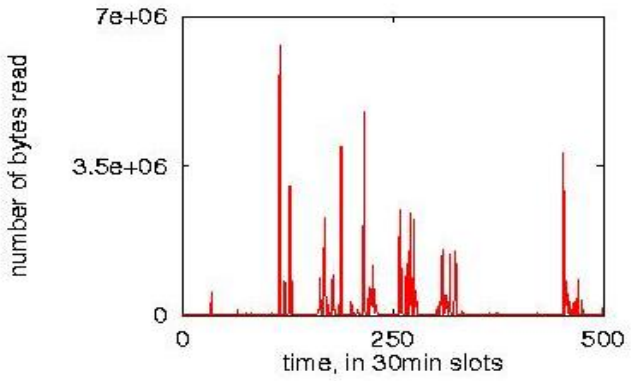
Motivation

- predict queue length distributions (e.g., to give probabilistic guarantees)
- “learn” traffic, for buffering, prefetching, ‘active disks’, web servers

Q: any 'pattern'?

- Not Poisson
- spike; silence; more spikes; more silence…
- any rules?

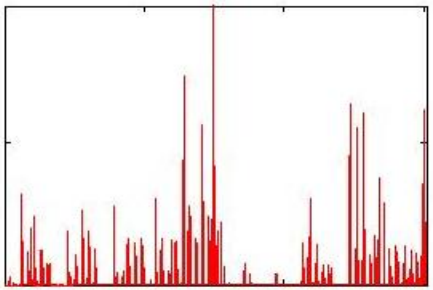
 # bytes



time

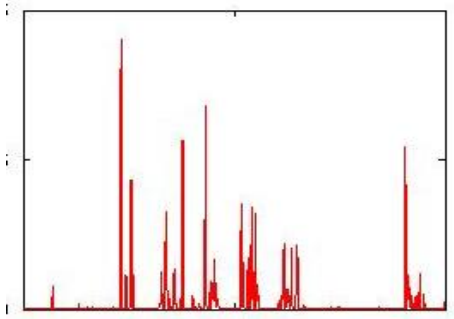
solution: self-similarity

 # bytes



time

 # bytes

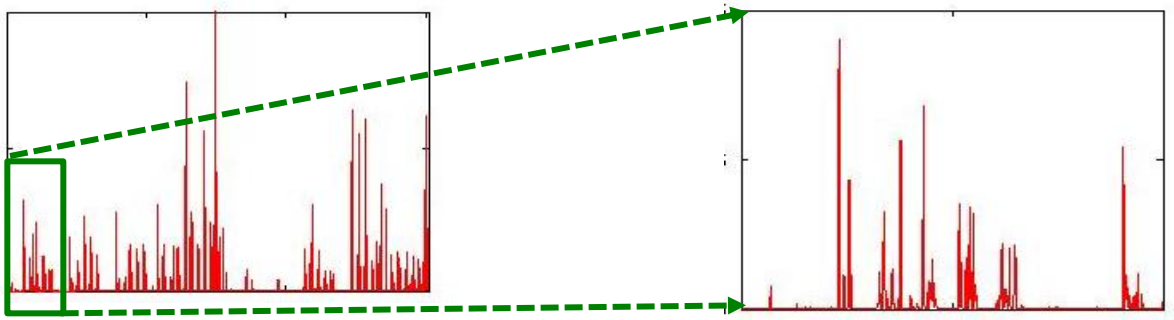


time

solution: self-similarity

 # bytes

 # bytes



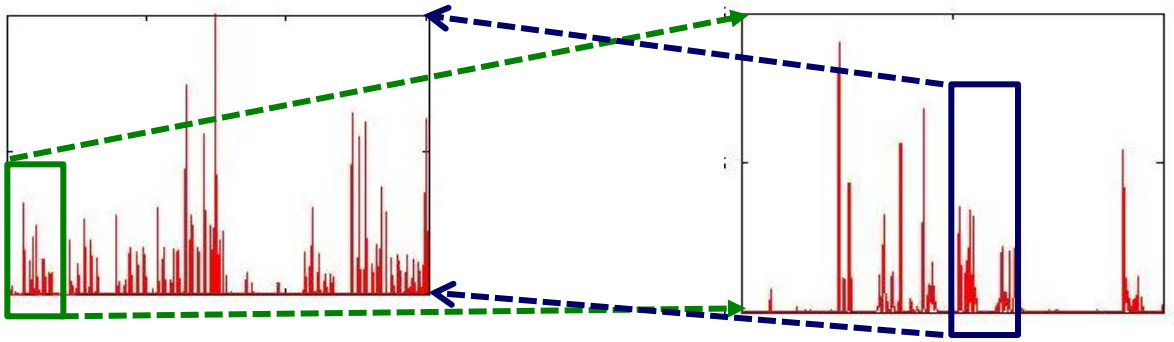
time

time

solution: self-similarity

 # bytes

 # bytes



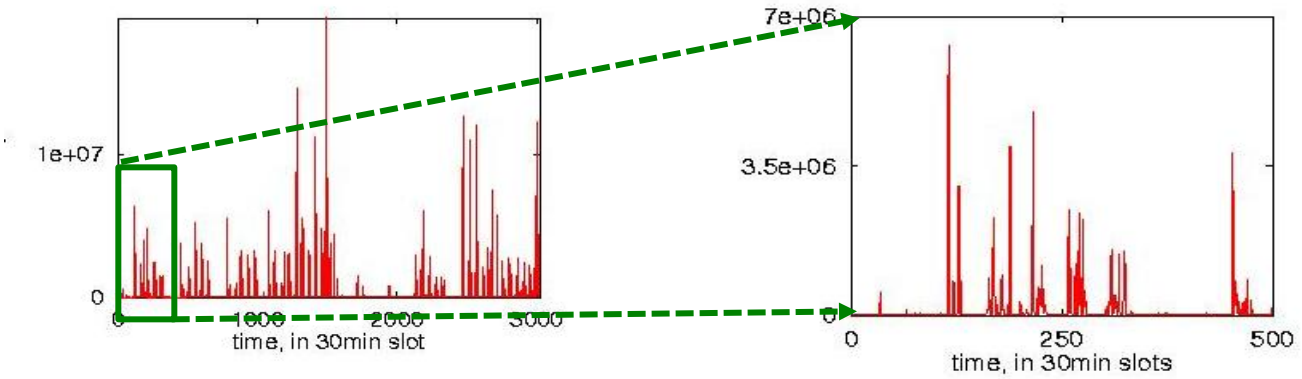
time

time

solution: self-similarity

 # bytes

 # bytes



time

time

But:

- Q1: How to generate realistic traces; extrapolate; give guarantees?
- Q2: How to estimate the model parameters?

Outline

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

Approach

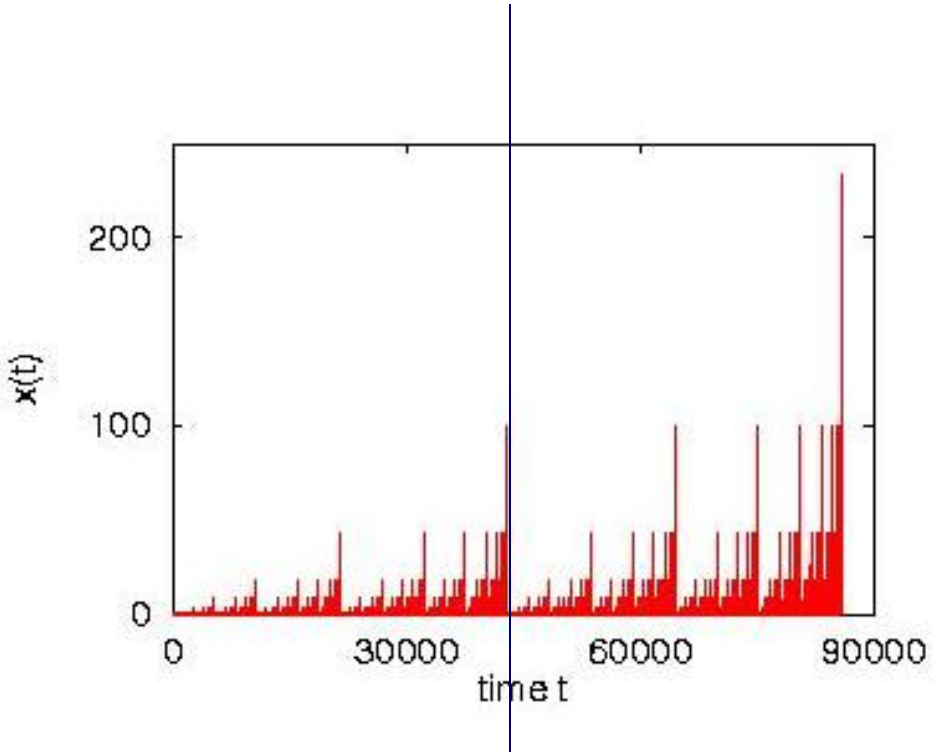
- Q1: How to generate a sequence, that is
  - bursty
  - self-similar
  - and has similar queue length distributions

Approach

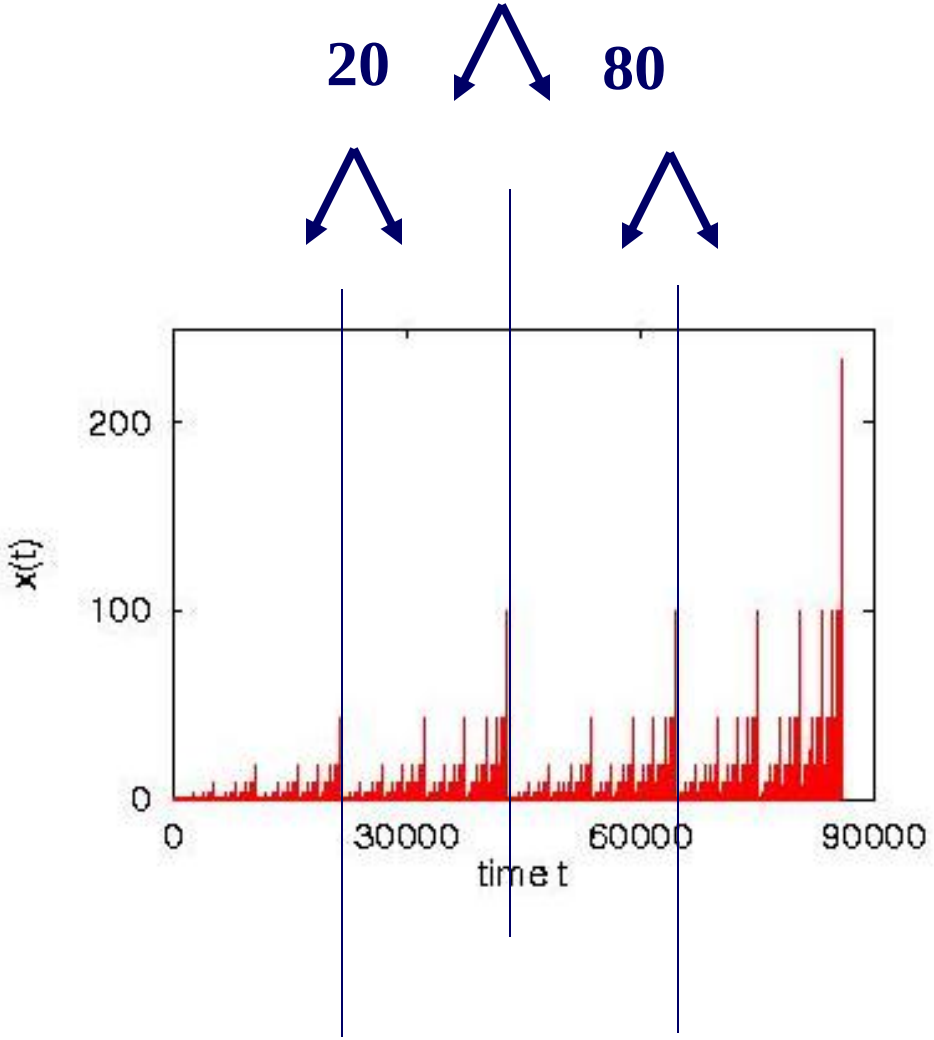
- A: 'binomial multifractal' [Wang+02]
- ~ 80-20 'law':
  - 80% of bytes/queries etc on first half
  - repeat recursively
- *b*: bias factor (eg., 80%)

binary multifractals

20 **80**



binary multifractals



Parameter estimation

- Q2: How to estimate the bias factor *b*?

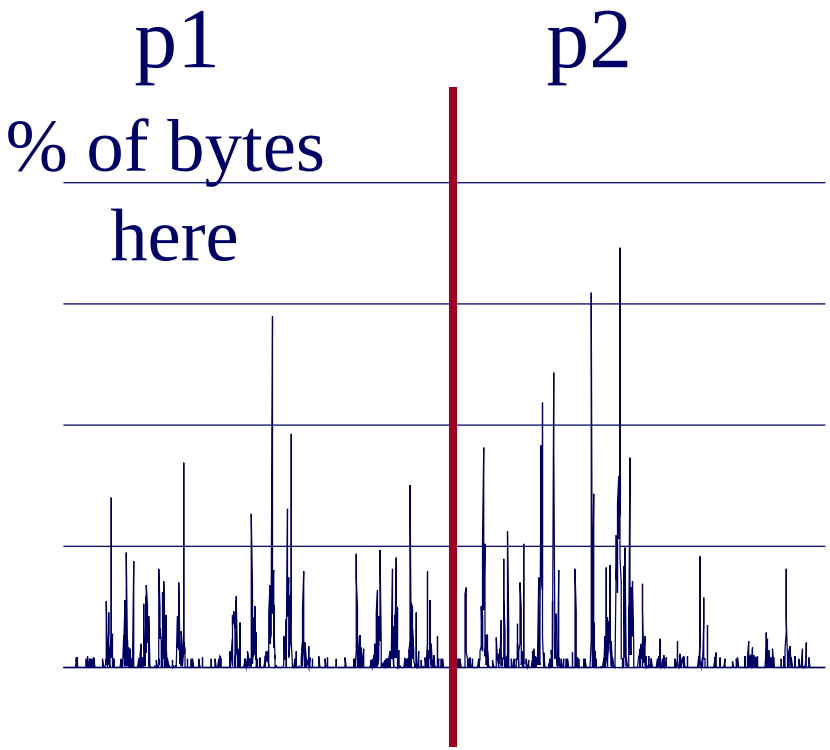
Parameter estimation

- Q2: How to estimate the bias factor *b*?
- A: MANY ways [Crovella+96]
  - Hurst exponent
  - variance plot
  - even DFT amplitude spectrum! ('periodogram')
  - More robust: 'entropy plot' [Wang+02]

Entropy plot

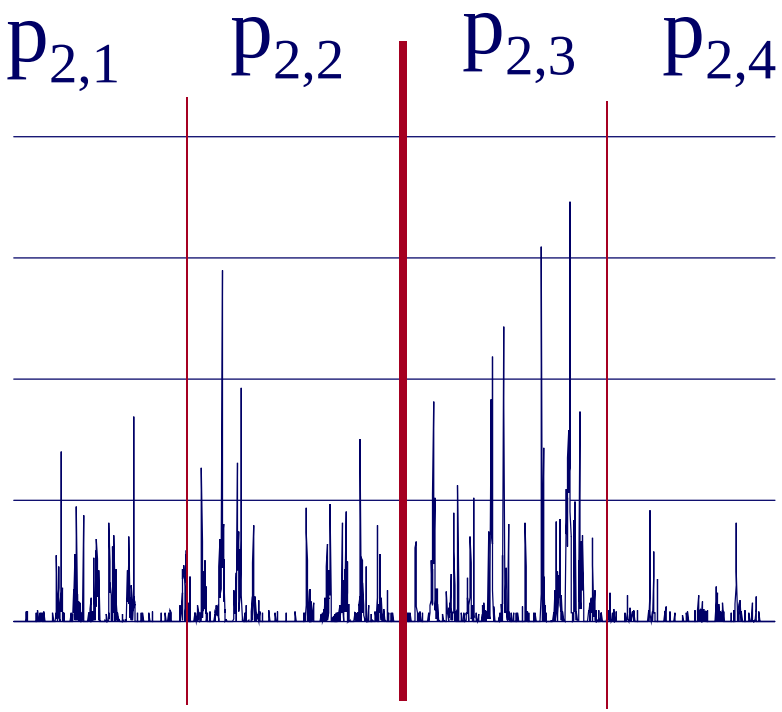
- Rationale:
  - burstiness: inverse of uniformity
  - entropy measures uniformity of a distribution
  - find entropy at several granularities, to see whether/how our distribution is close to uniform.

Entropy plot



- Entropy *E(n)* after *n* levels of splits
- n=1: E(1)= - p1 $\log_2$(p1)- p2 $\log_2$(p2)

Entropy plot

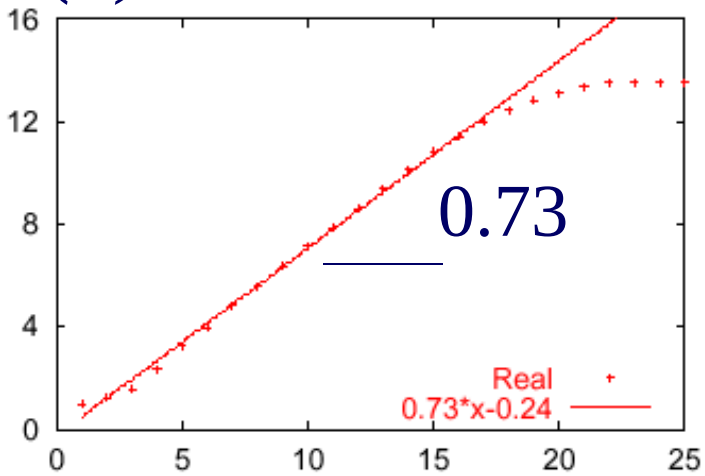


- Entropy *E(n)* after *n* levels of splits
- n=1: E(1)= - p1 log(p1)- p2 log(p2)
- n=2: E(2) = - Σ_i $p_{2,i}$ * $\log_2$ ($p_{2,i}$)

Real traffic

Entropy

E(n)



# of levels (*n*)

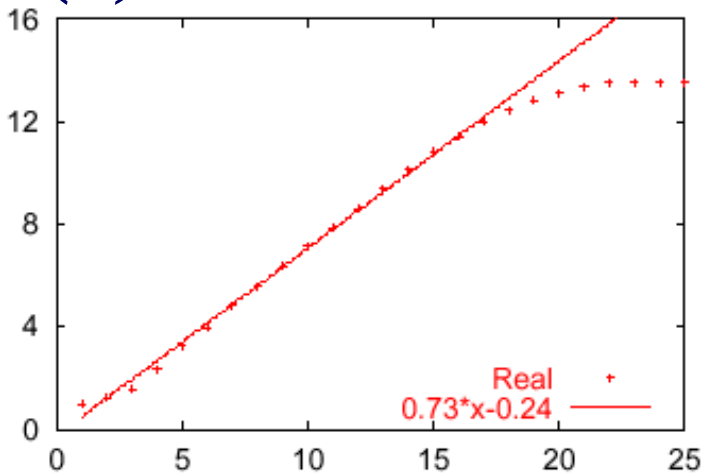
- Has linear entropy plot (-> self-similar)

Skip

Observation - intuition:

Entropy

$E(n)$



# of levels (n)

intuition: slope =

intrinsic dimensionality =

info-bits per coordinate-bit

- unif. Dataset: slope =1
- multi-point: slope = 0

Skip

Entropy plot - Intuition

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

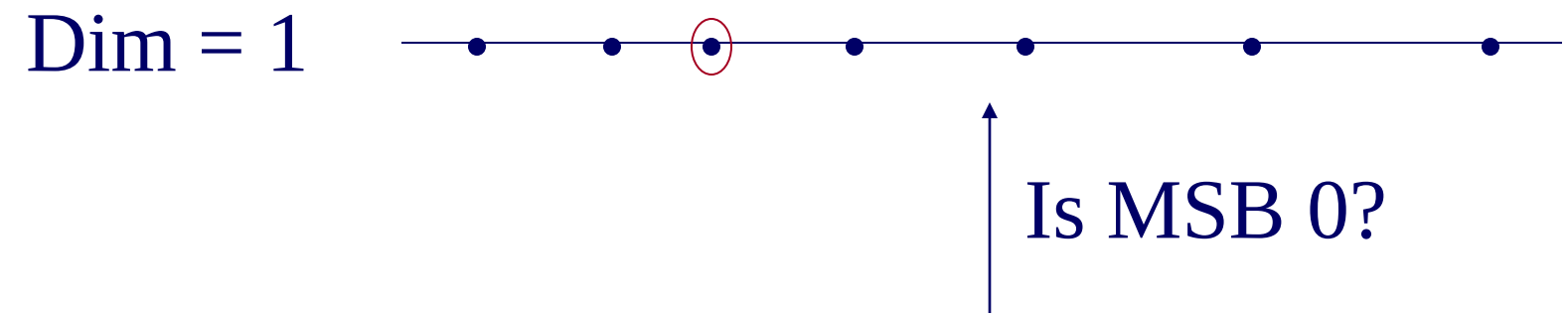
Dim = 1

Pick a point;
reveal its coordinate bit-by-bit -
how much info is each bit worth to me?

Skip

Entropy plot

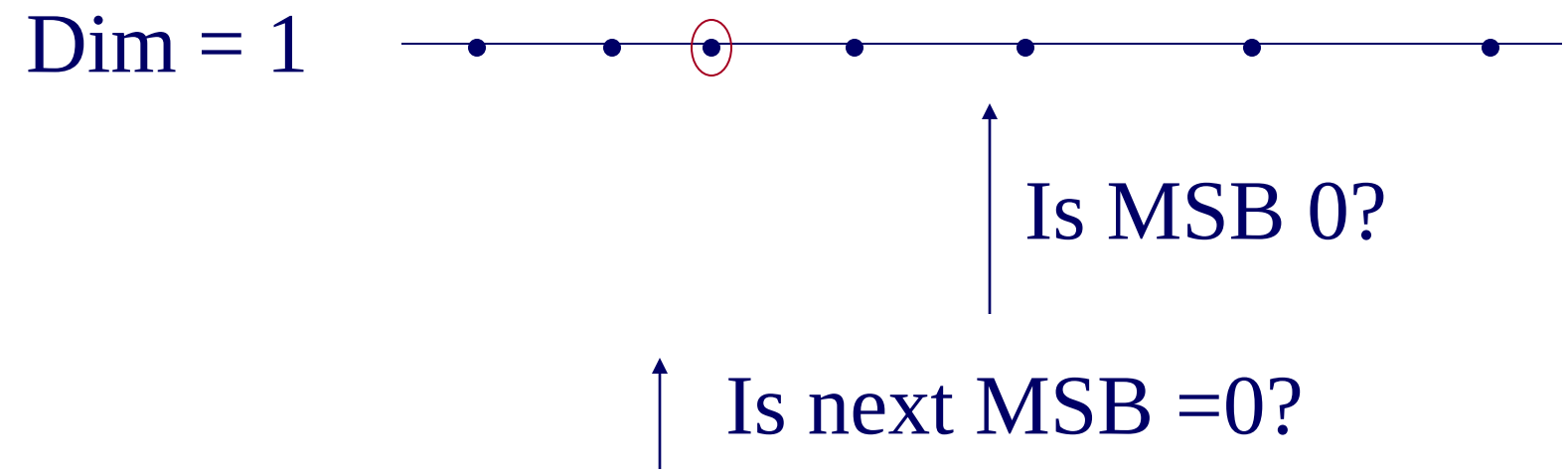
- Slope ~ intrinsic dimensionality (in fact, 'Information fractal dimension')
- = info bit per coordinate bit - eg



Skip

Entropy plot

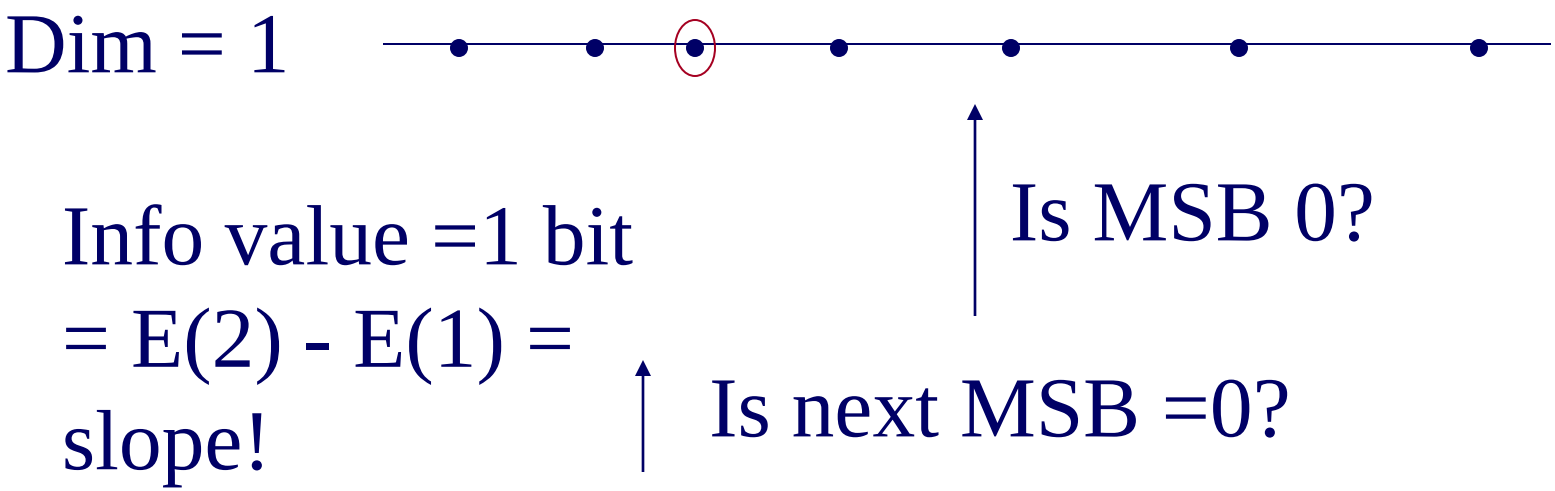
- Slope ~ intrinsic dimensionality (in fact, 'Information fractal dimension')
- = info bit per coordinate bit - eg



Skip

Entropy plot

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



Skip

Entropy plot

- Repeat, for all points at same position:

Dim=0

Skip

Entropy plot

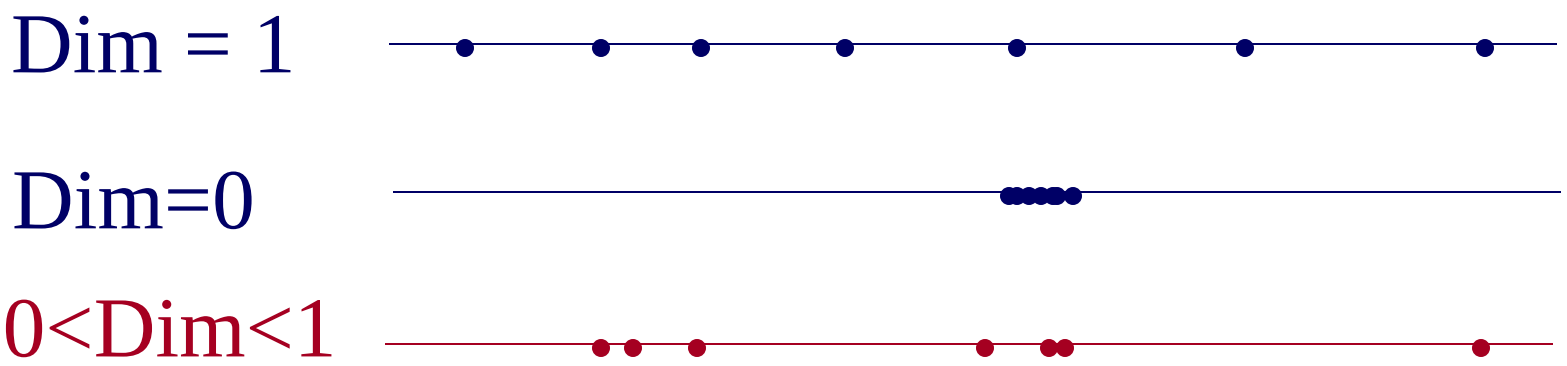
- Repeat, for all points at same position:
- we need 0 bits of info, to determine position
- -> slope = 0 = intrinsic dimensionality

Dim=0

Skip

Entropy plot

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



(Fractals)

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

Cantor dust

- Eliminate the middle third
- Recursively!

Cantor dust

Cantor dust

Cantor dust

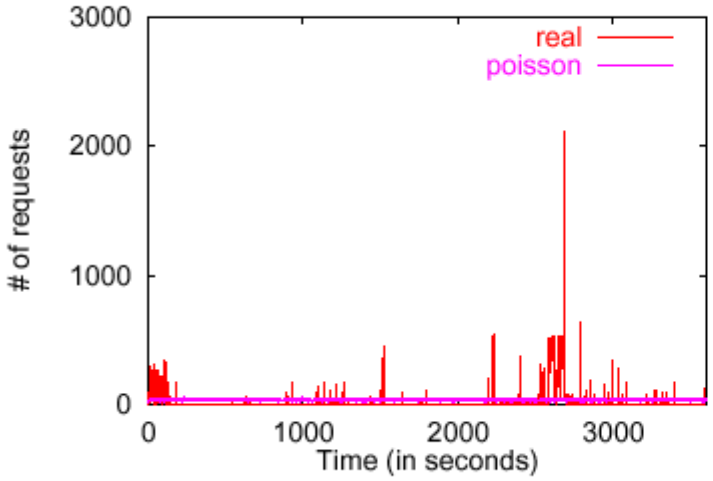
Cantor dust

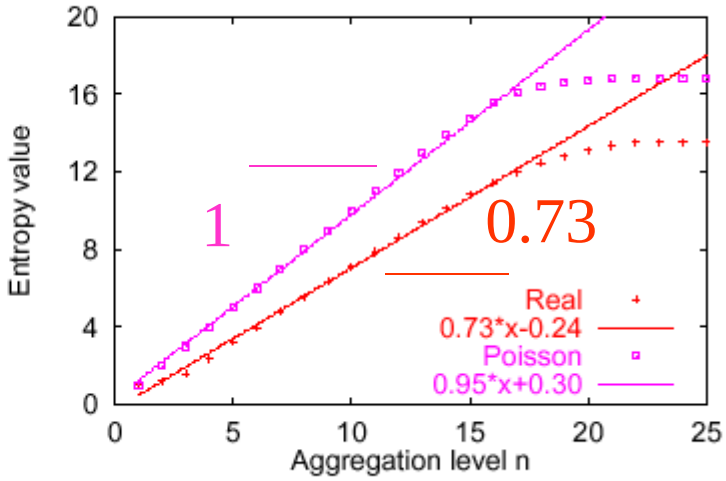
Cantor dust

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

Some more entropy plots:

- Poisson vs real

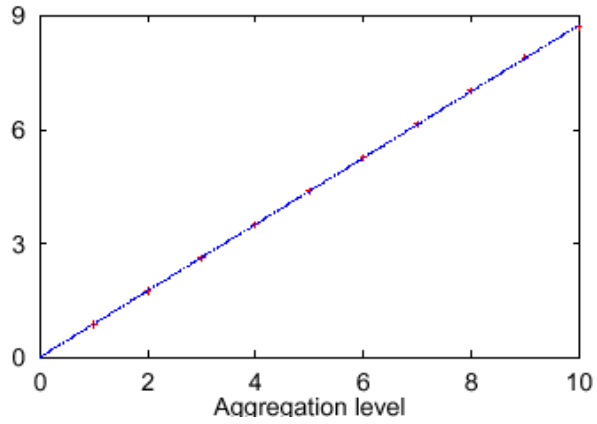




Poisson: slope = ~1 -> uniformly distributed

B-model

$E(n)$



n

- b-model traffic gives perfectly linear plot
- Lemma: its slope is

  $slope = -b \log_2 b - (1-b) \log_2 (1-b)$
- Fitting: do entropy plot; get slope; solve for b

Outline

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

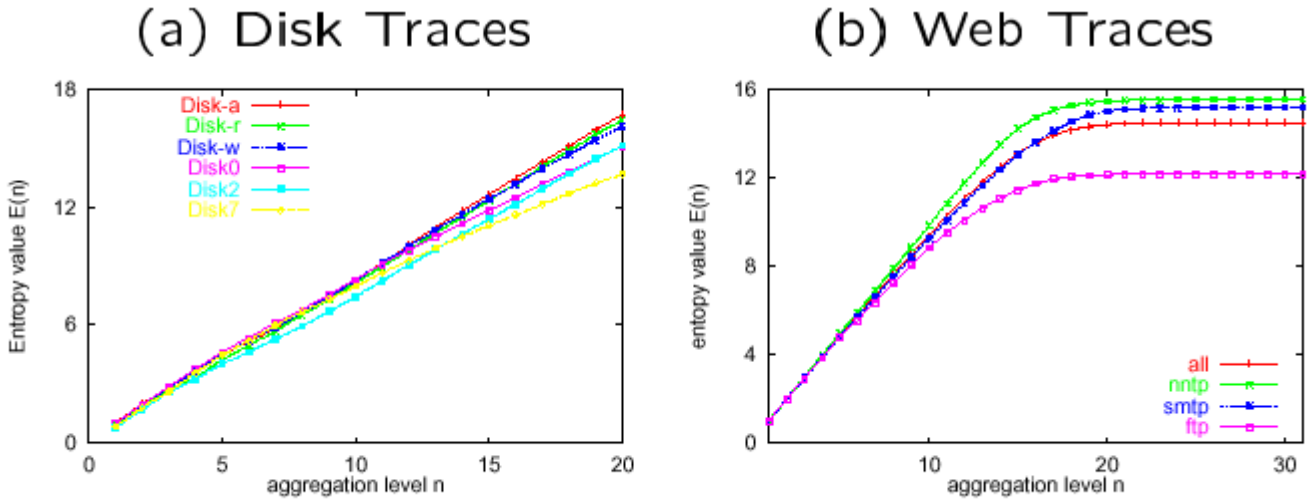
Experimental setup

- Disk traces (from HP [Wilkes 93])
- web traces from LBL

  http://repository.cs.vt.edu/lbl-conn-7.tar.Z

Model validation

- Linear entropy plots



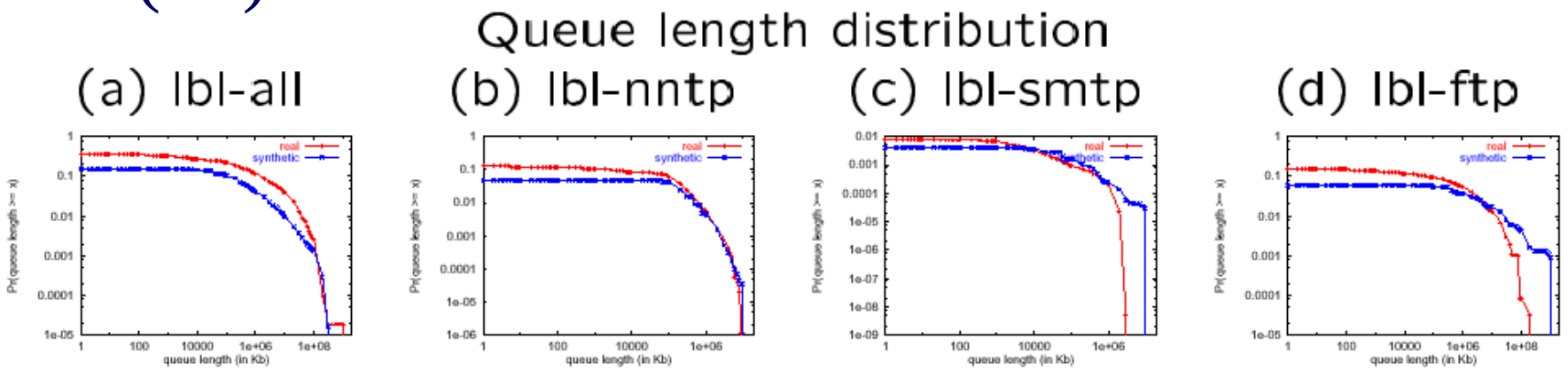
Bias factors *b*: 0.6-0.8

smallest *b* / smoothest: nntp traffic

Web traffic - results

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

Prob(>*l*)

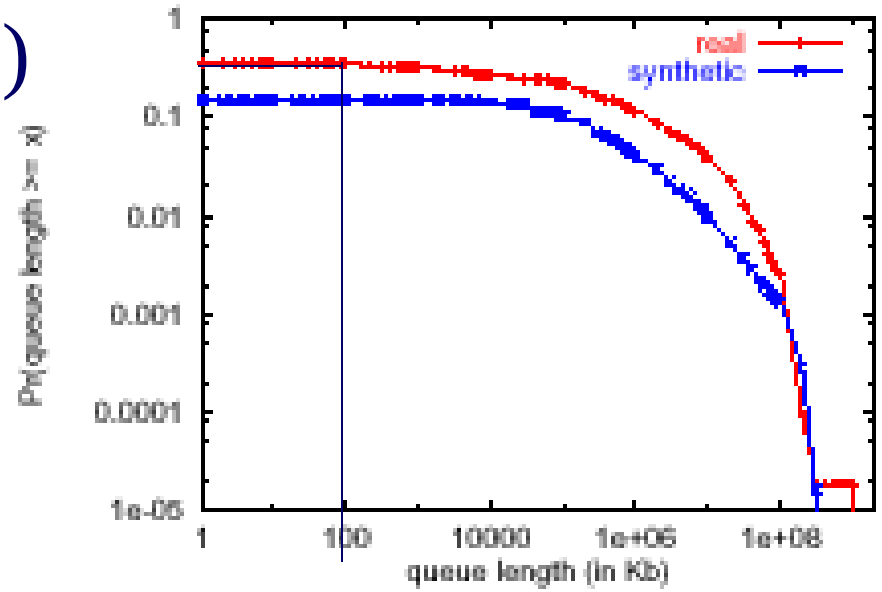


How to give guarantees? (queue length *l*)

Web traffic - results

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

Prob(>*l*)



20% of the requests will see queue lengths <100

(queue length *l*)

Conclusions

- Multifractals (80/20, ‘b-model’, Multiplicative Wavelet Model (MWM)) for analysis and synthesis of bursty traffic
- can give (probabilistic) guarantees

Books

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

Further reading:

- Crovella, M. and A. Bestavros (1996). Self-Similarity in World Wide Web Traffic, Evidence and Possible Causes. Sigmetrics.
- [ieeeTN94] W. E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, *On the Self-Similar Nature of Ethernet Traffic*, IEEE Transactions on Networking, 2, 1, pp 1-15, Feb. 1994.

Further reading

- [Riedi+99] R. H. Riedi, M. S. Crouse, V. J. Ribeiro, and R. G. Baraniuk, *A Multifractal Wavelet Model with Application to Network Traffic*, IEEE Special Issue on Information Theory, 45. (April 1999), 992-1018.
- [Wang+02] Mengzhi Wang, Tara Madhyastha, Ngai Hang Chang, Spiros Papadimitriou and Christos Faloutsos, *Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic*, ICDE 2002, San Jose, CA, 2/26/2002 - 3/1/2002.

Part 6: chaos and non-linear forecasting

Outline

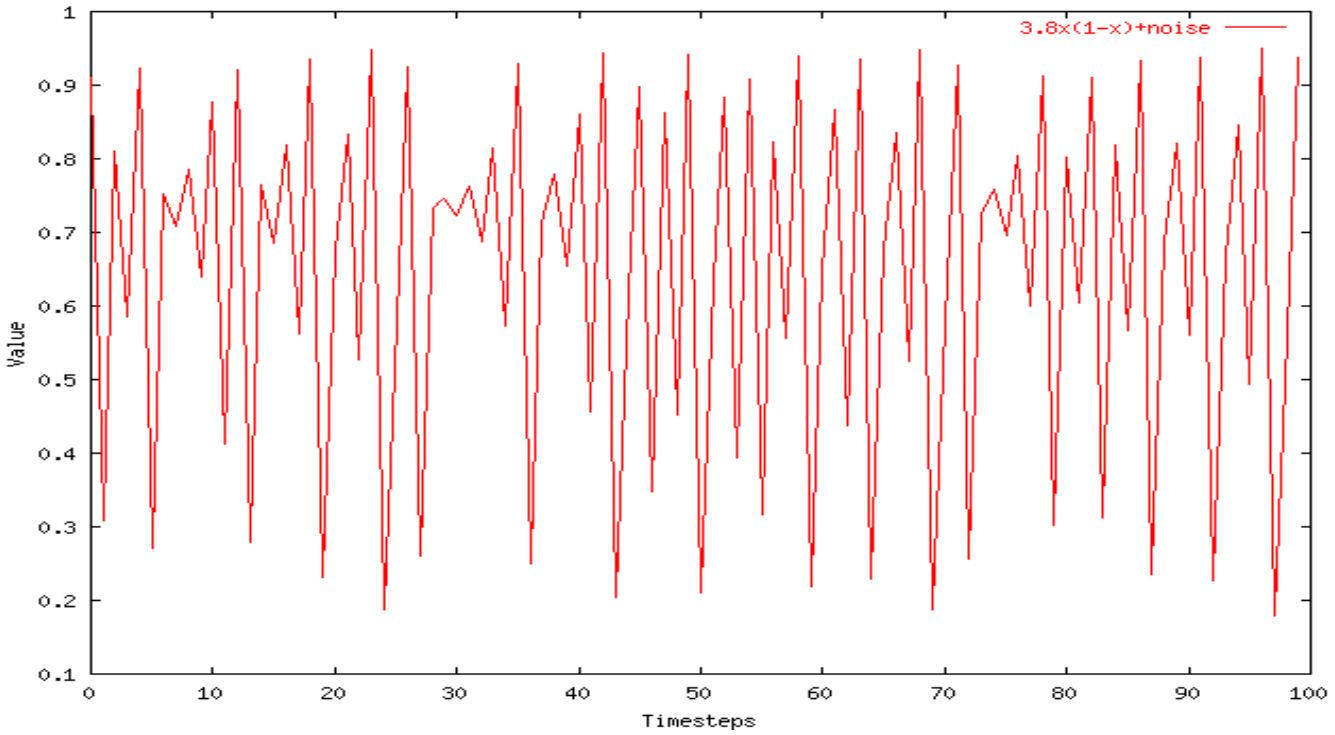
- Motivation
- Similarity Search and Indexing
- DSP
- Linear Forecasting
- Kalman filters
- Bursty traffic - fractals and multifractals
- Non-linear forecasting
- Conclusions

Detailed Outline

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

Recall: Problem #1

Value



Time

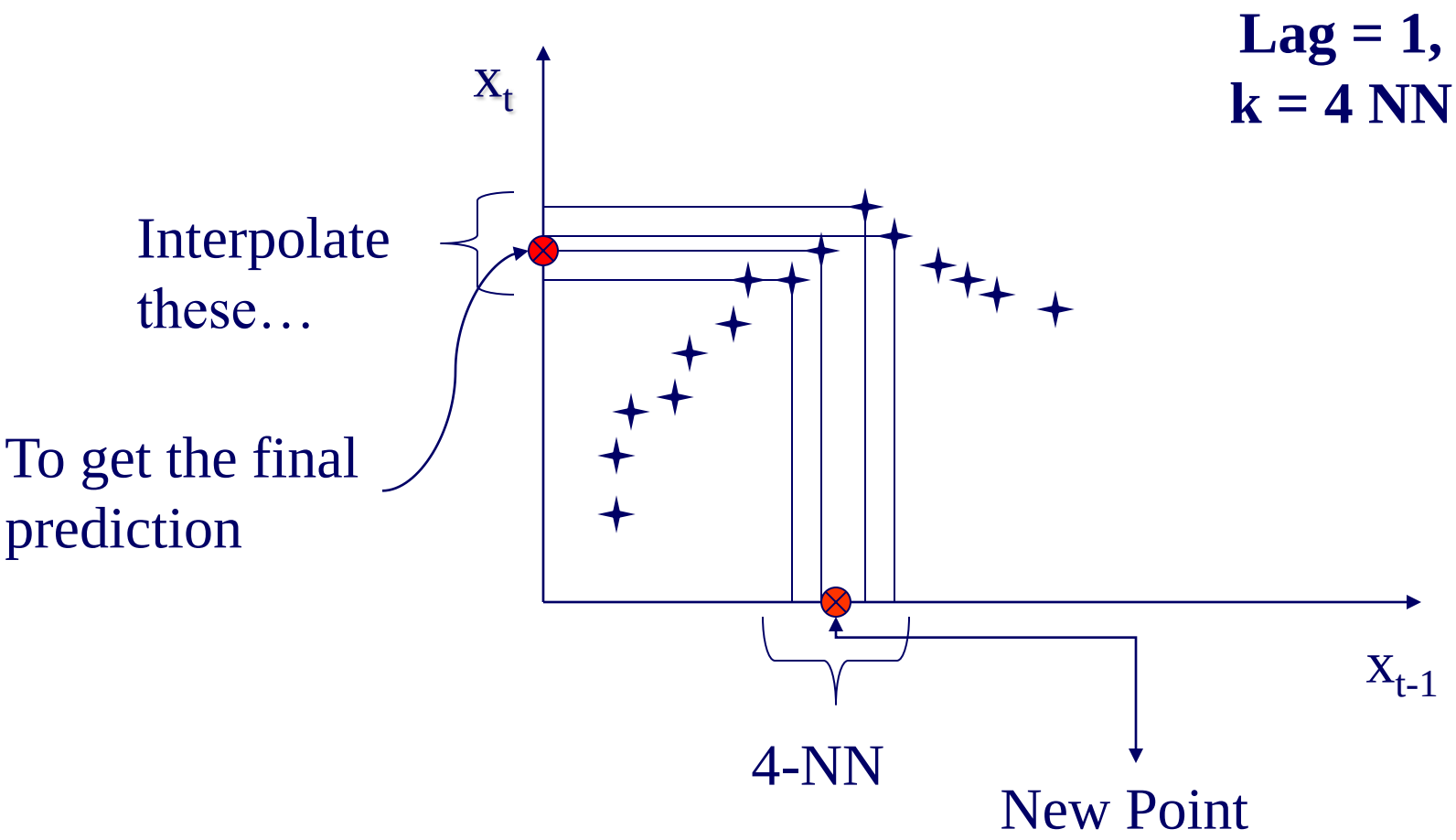
Given a time series $\{x_t\}$, predict its future course, that is, x_{t+1}, x_{t+2}, ...

How to forecast?

- ARIMA - but: linearity assumption

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

General Intuition (Lag Plot)



Questions:

- Q1: How to choose lag *L*?
- Q2: How to choose *k* (the # of NN)?
- Q3: How to interpolate?
- Q4: why should this work at all?

Q1: Choosing lag *L*

- Manually (16, in award winning system by [Sauer94])

Q2: Choosing number of neighbors *k*

- Manually (typically ~ 1-10)

Q3: How to interpolate?

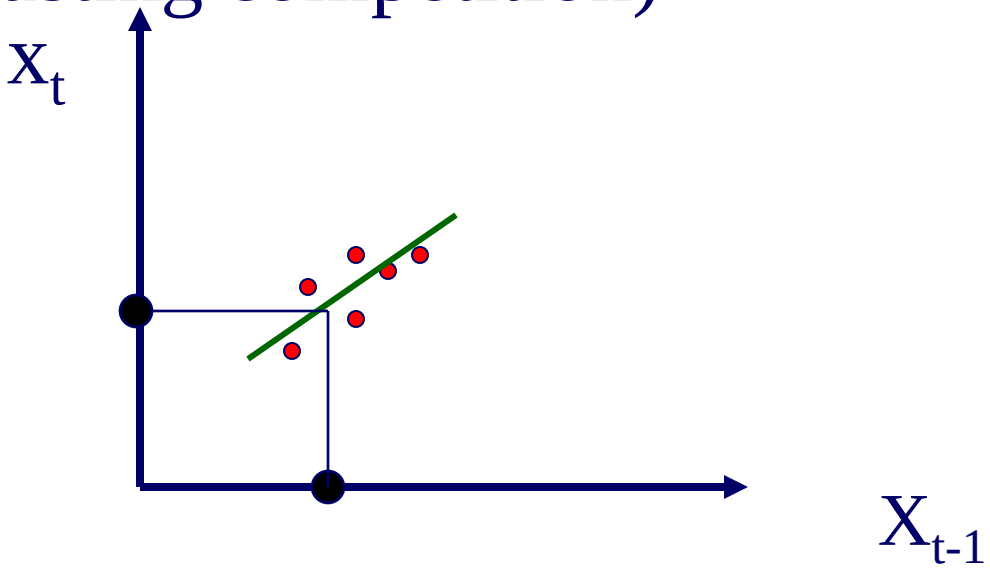
How do we interpolate between the *k* nearest neighbors?

A3.1: Average

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

Q3: How to interpolate?

A3.3: Using SVD - seems to perform best ([Sauer94] - first place in the Santa Fe forecasting competition)



Q4: Any theory behind it?

A4: YES!

Theoretical foundation

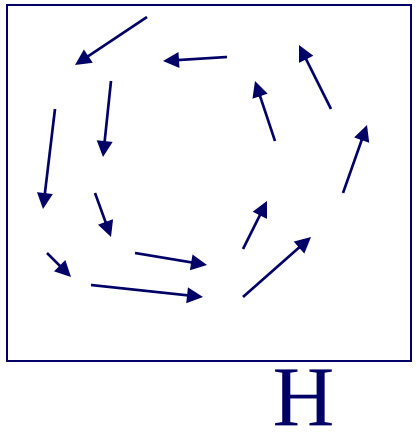
- Based on the "Takens' Theorem" [Takens81]
- which says that long enough delay vectors can do prediction, even if there are unobserved variables in the dynamical system (= diff. equations)

Skip

Theoretical foundation

Example: Lotka-Volterra equations

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



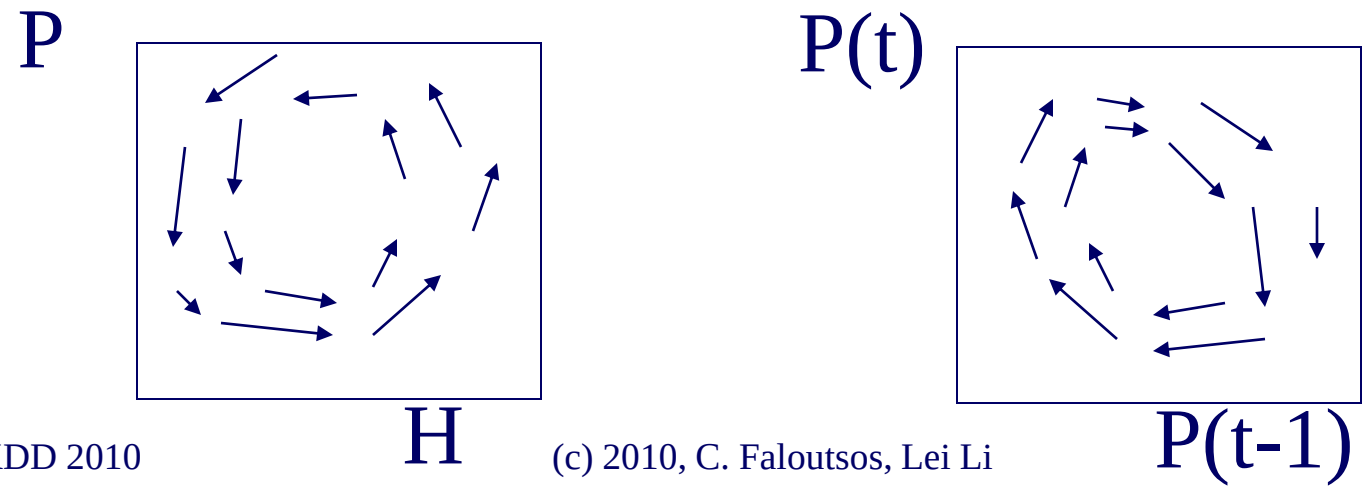
H is count of prey (e.g., hare)
P is count of predators (e.g., lynx)

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

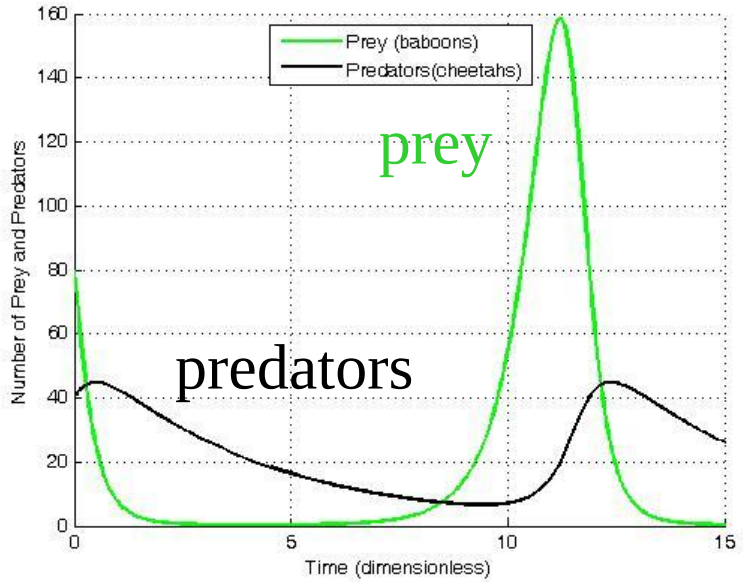
Skip

Theoretical foundation

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

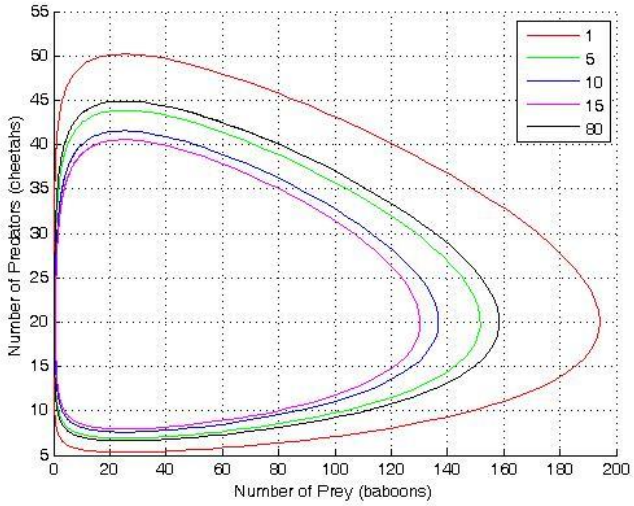


Solution to Volterra-Lotka eq.



time

predators



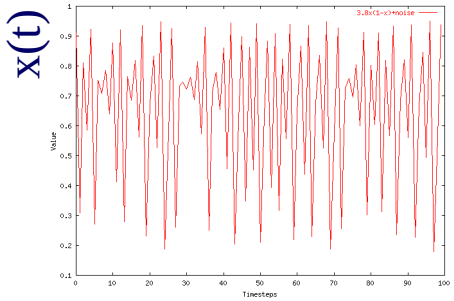
prey

from wikipedia

Detailed Outline

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

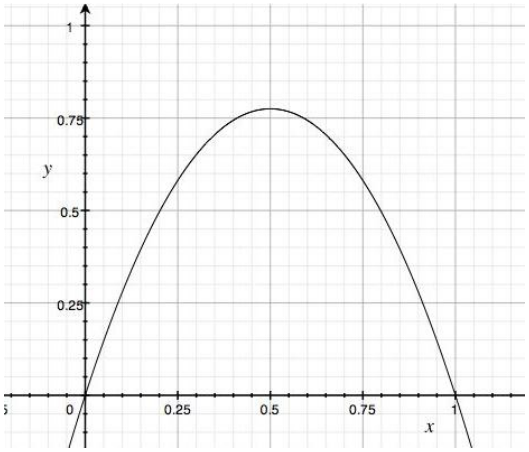
Datasets



Logistic Parabola:

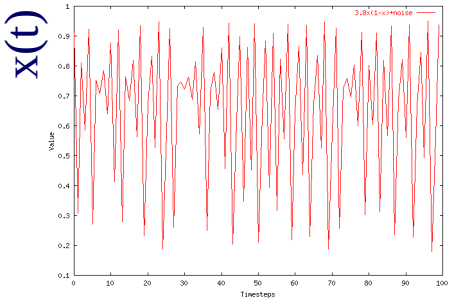
$x_t = a x_{t-1}(1-x_{t-1}) + noise$

Models population of flies [R. May/1976]



Lag-plot

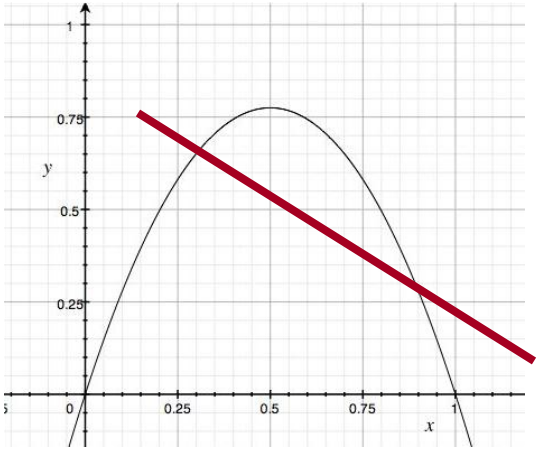
Datasets



Logistic Parabola:

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

Models population of flies [R. May/1976]



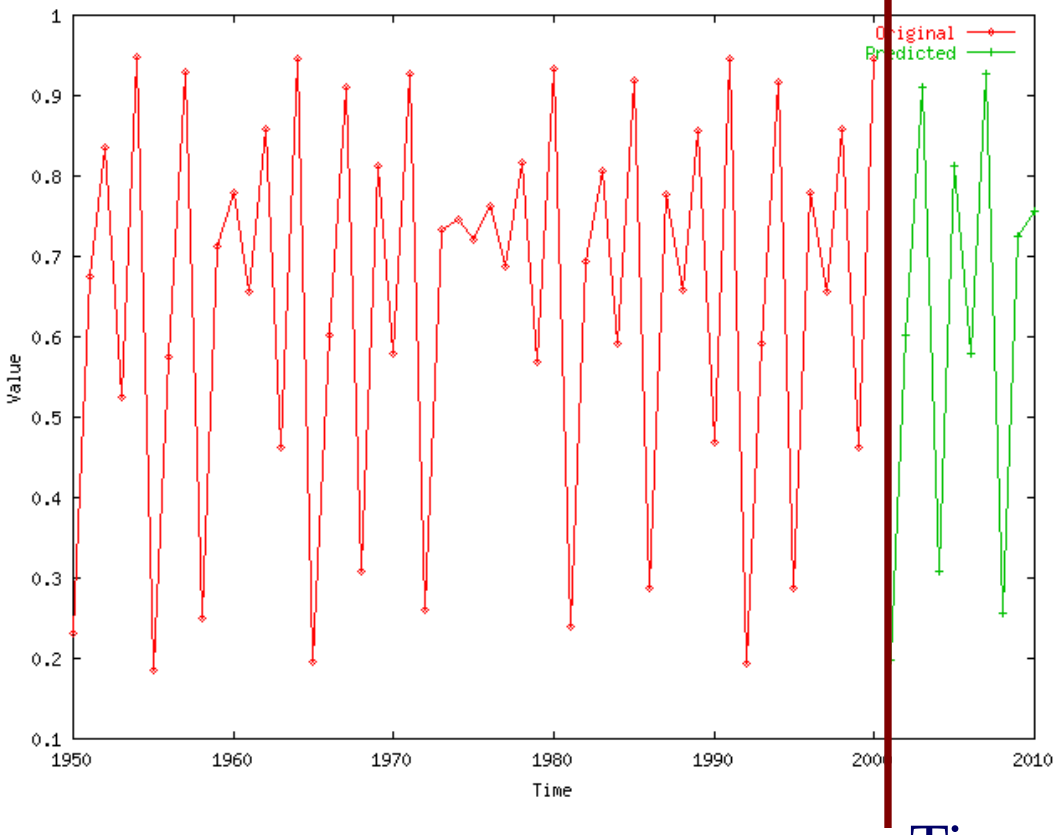
Lag-plot

ARIMA: fails

Logistic Parabola

Our Prediction from here

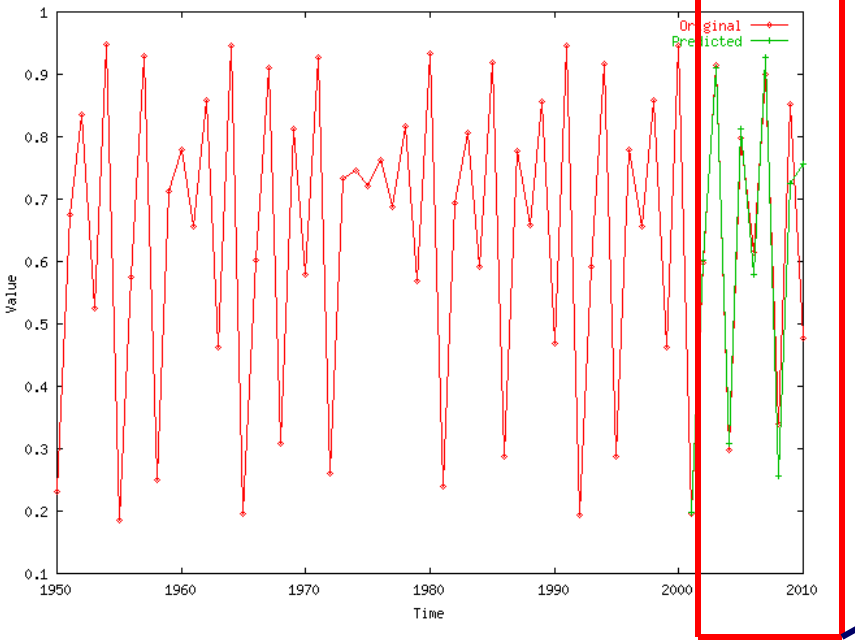
Value



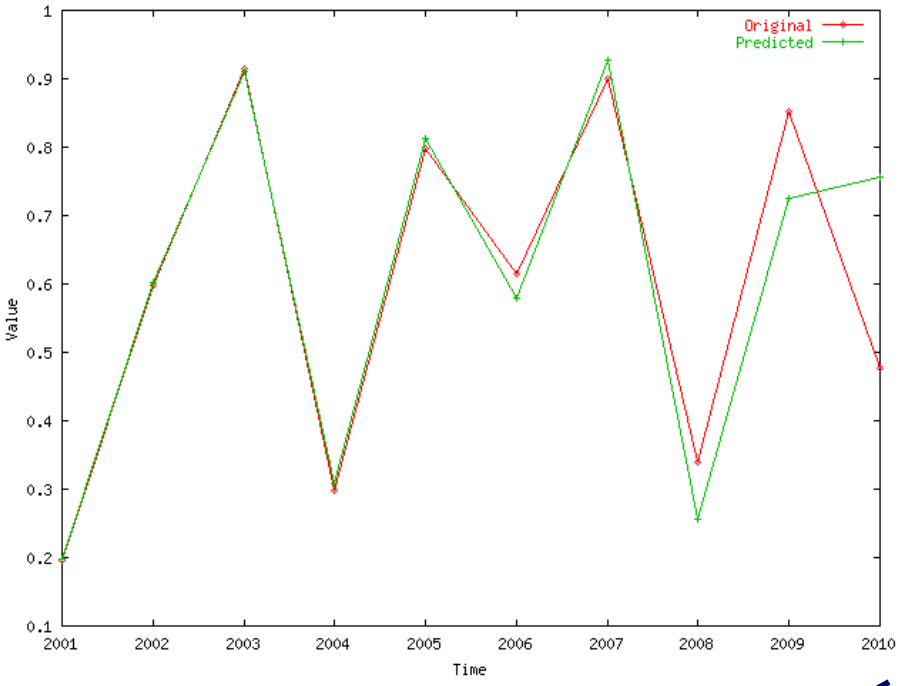
Timesteps

Logistic Parabola

Comparison of prediction to correct values



Value



Timesteps

Skip

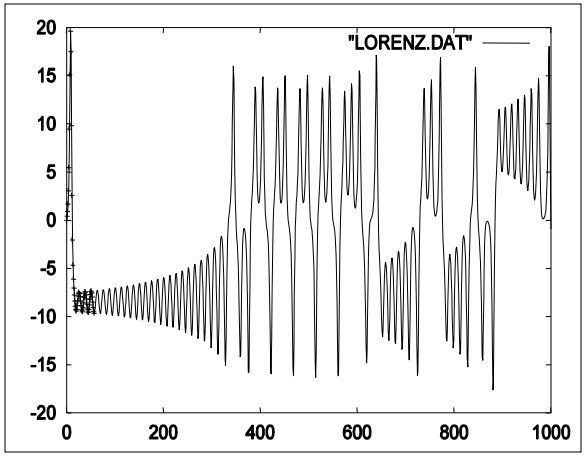
Datasets

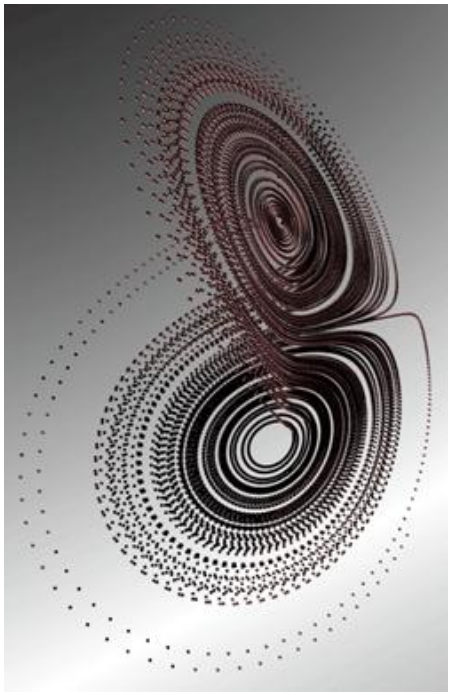
LORENZ: Models convection currents in the air

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

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

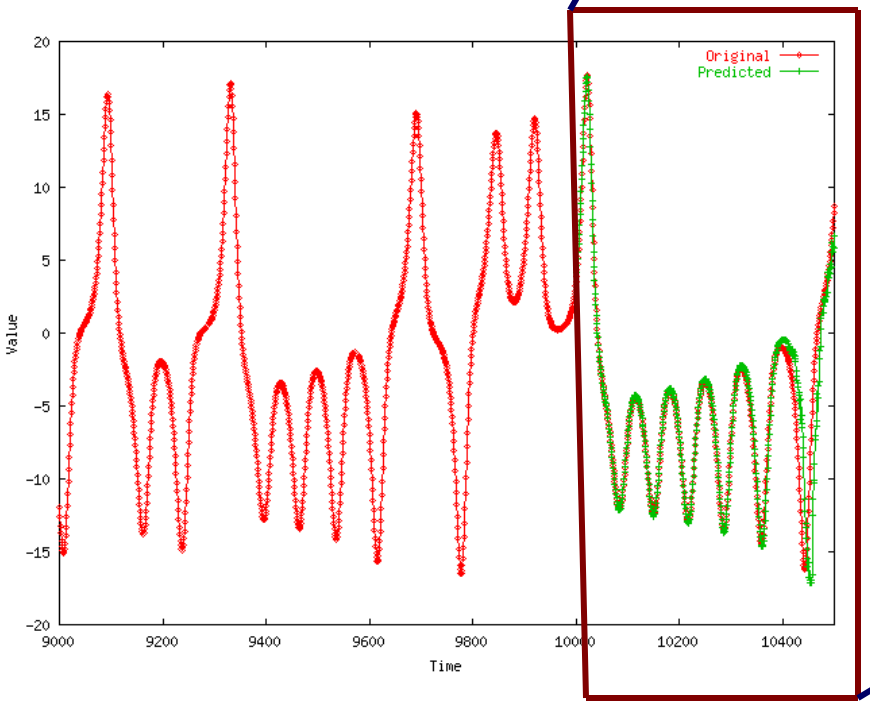
$dz / dt = xy - c z$





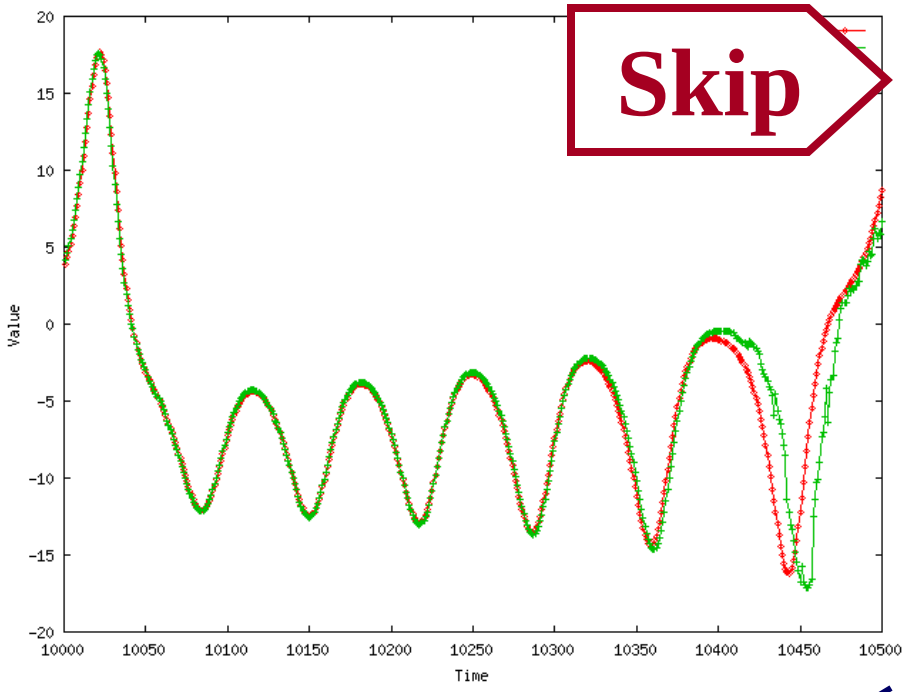
LORENZ

Comparison of prediction to correct values



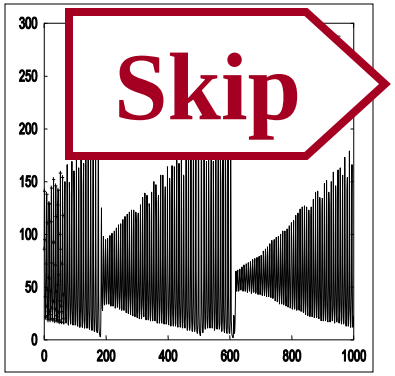
Value

Timesteps



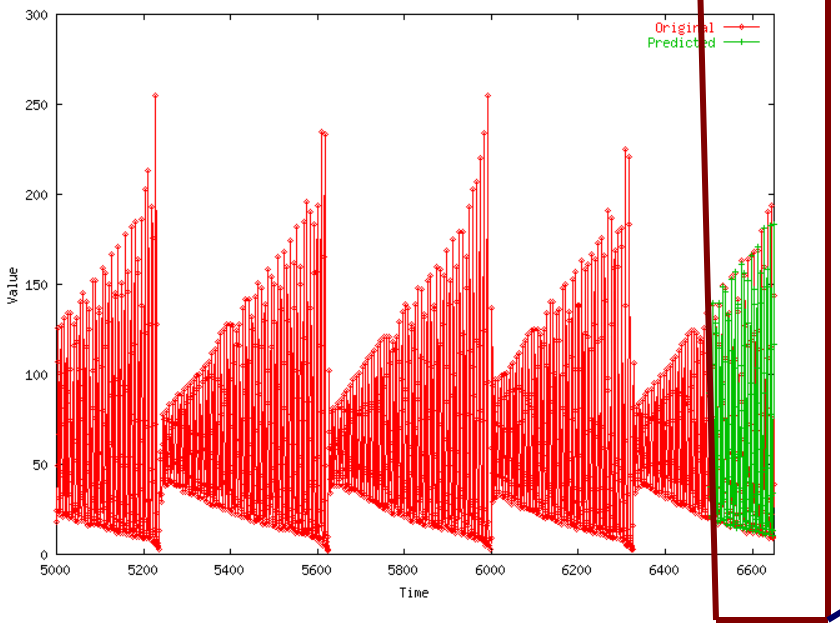
Datasets

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

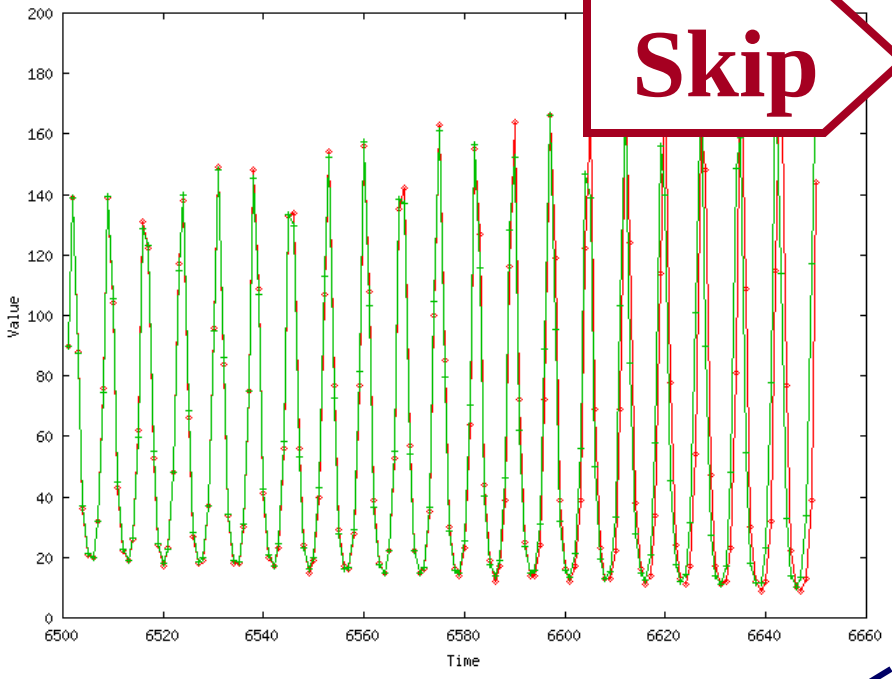


Laser

Comparison of prediction to correct values



Value



Timesteps

Conclusions

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

References

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

References

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

Overall conclusions

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

Overall conclusions

- Similarity search: **Euclidean**/time-warping; **feature extraction** and **SAMs**
- Signal processing: **DWT** is a powerful tool

Overall conclusions

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

Overall conclusions

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

Overall conclusions

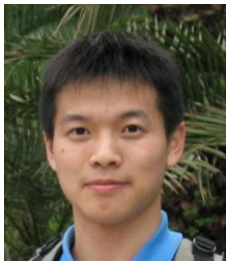
- Similarity search: **Euclidean**/time-warping; **feature extraction** and **SAMs**
- Signal processing: **DWT** is a powerful tool
- Linear Forecasting: **AR** (Box-Jenkins)
- **Kalman** filters & extensions: forecasting, pattern discovery, segmentation
- Bursty traffic: **multifractals** (80-20 ‘law’)

Overall conclusions

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

THANK YOU!

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



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

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