Data Analysis Methods in IoT

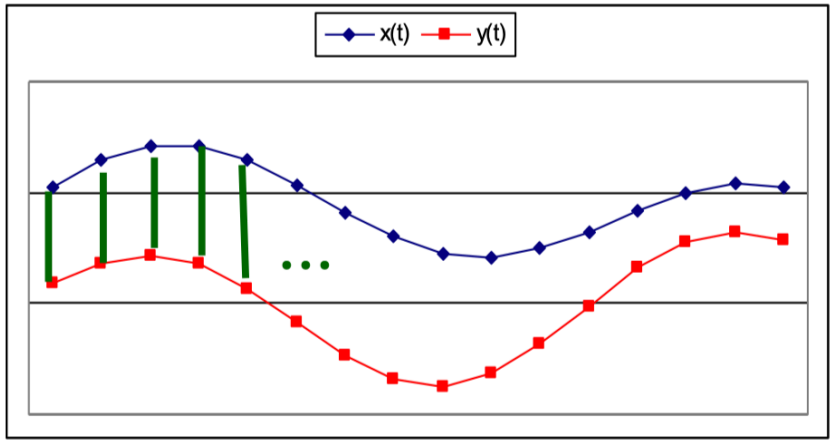
Similarity Measurements in IoT Time Series Data

Distance Functions

There are two major families we can use in similarity measurements.

- Euclidean and Lp norms
- Time warping and variations

Euclidean and Lp



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

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

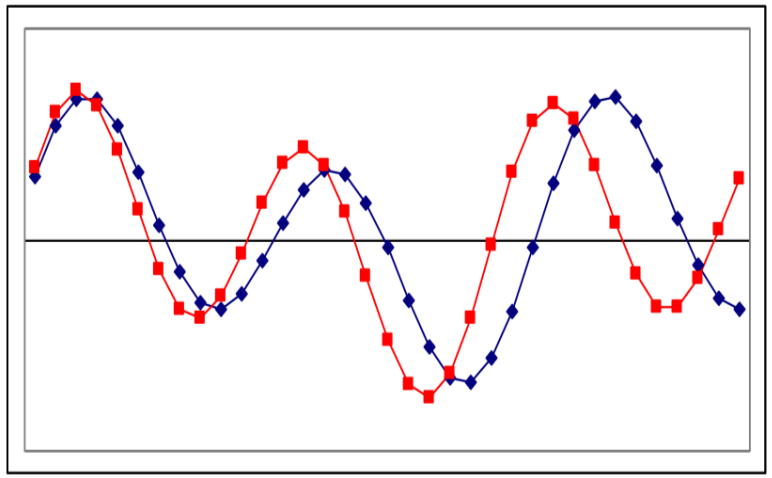
- L_1: city-block = Manhattan
- L_2 = Euclidean
- L_∞

Time Warping

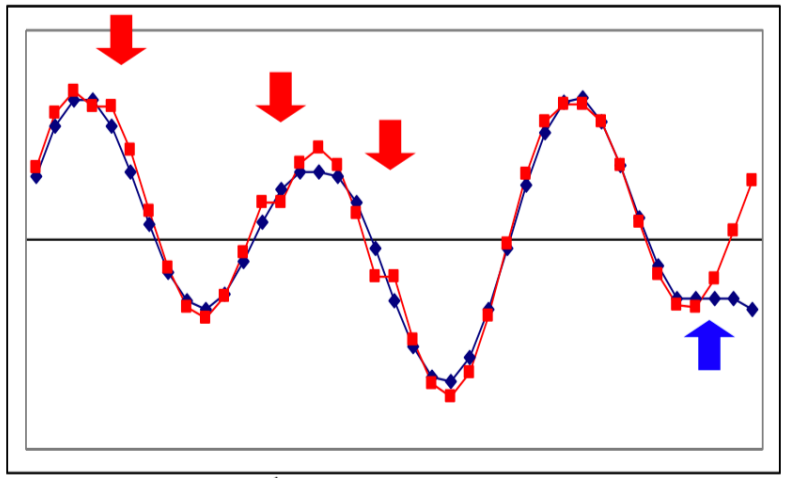
- allow accelerations - decelerations
  - –(with or w/o penalty)
- THEN compute the (Euclidean) distance (+ penalty)
- related to the string-editing distance
- fast search methods [Yi+98] [Keogh+02] [Sakurai+05]

Time Warping

- Allow sequences to be stretched along the time axis
  1. minimize the distance of sequences
  2. insert ‘stutters’ into a sequence
  3. THEN compute the (Euclidean) distance



original



‘stutters’:

Time Warping

Q: how to compute it?

A: dynamic programming

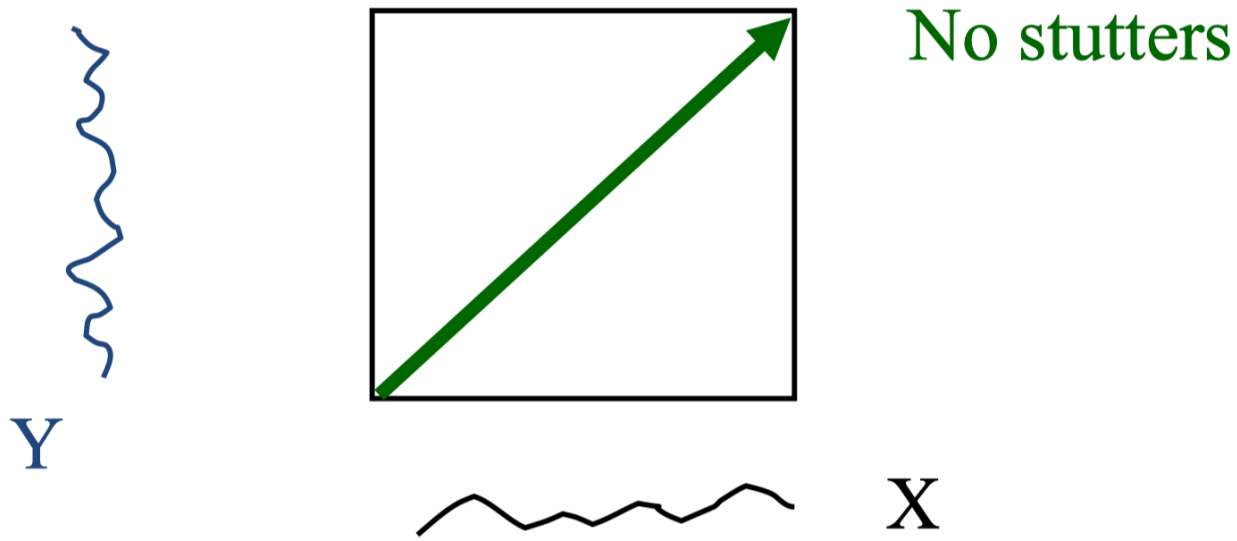
$$X=\{x_1, x_2, \ldots, x_i\},\ Y=\{y_1, y_2, \ldots, y_j\}$$

$$D_{dtw}(X,Y) = f(n,m)$$

$$f(i,j) = \left\| x_i - y_j \right\| + \min \begin{cases} f(i,j-1) & x\text{-stutter} \\ f(i-1,j) & y\text{-stutter} \\ f(i-1,j-1) & \text{no stutter} \end{cases}$$

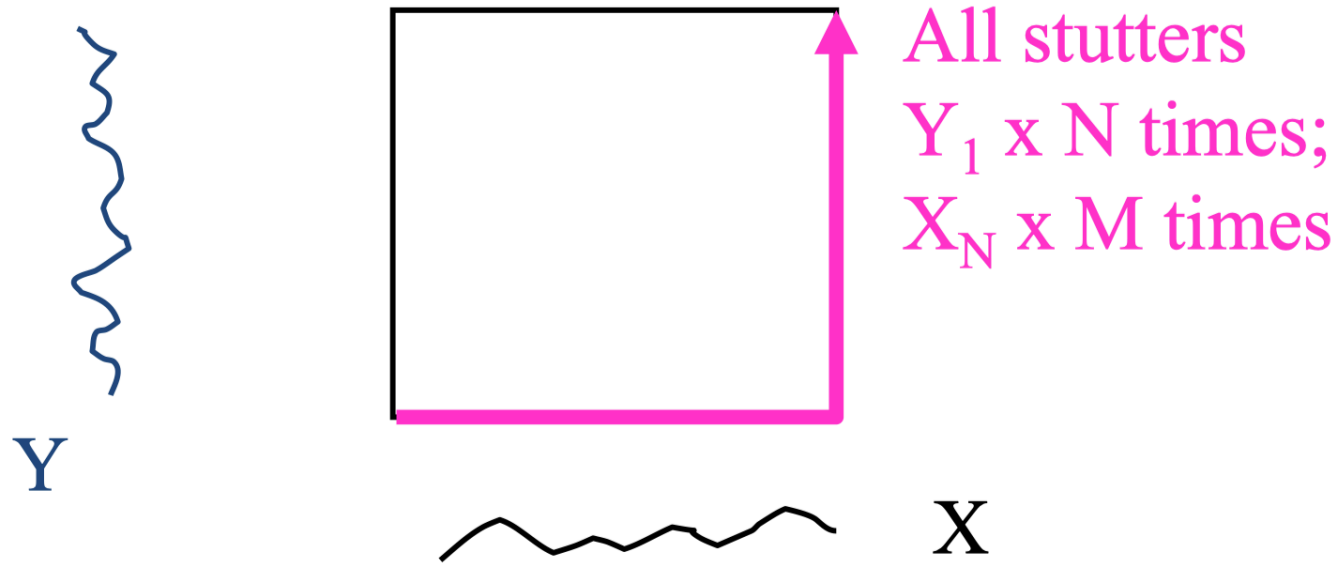
Time Warping

Time warping matrix & optimal path:



Time Warping

Time warping matrix & optimal path:



Time Warping

- Complexity: O(M*N) - quadratic on the length of the strings

- Many variations (penalty for stutters; limit on the number/percentage of stutters; …)

- popular in voice processing [Rabiner+Juang]

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]

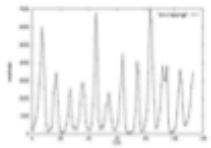
Classical Methods in DSP, Data Aggregation, Feature Extraction

Outline for the Classical Methods in DSP, Data Aggregation, Feature Extraction

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

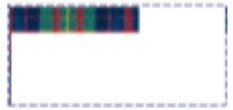
- DWT
  - Definition of DWT and properties
  - how to read the DWT scalogram
- SAX
  - Definition of SAX and properties
  - how to read the SAX output

Important observations



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

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

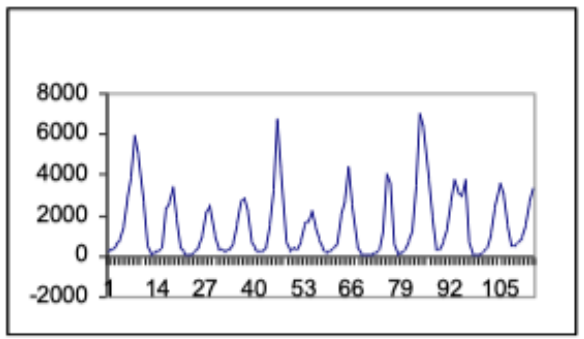


Introduction - Problem#1

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

Find: patterns and/or compress

count



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

year

What does DFT do?

A: highlights the periodicities

DFT: definition

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

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

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

inverse DFT

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

DFT: definition

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

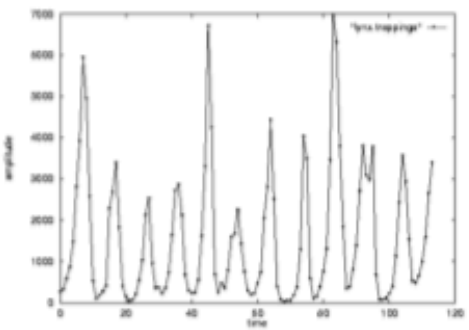
```
x = [1,2,1,2];
X = Fourier[x];
Plot[ Abs[X] ];
```

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)

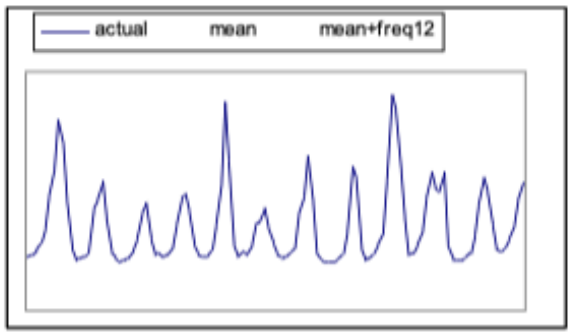
DFT: Amplitude spectrum

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

x

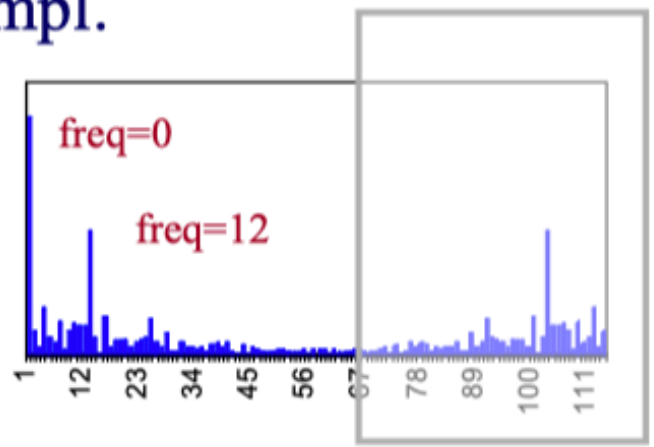
Plot[Abs[Fourier[x]]];

count



year

Ampl.



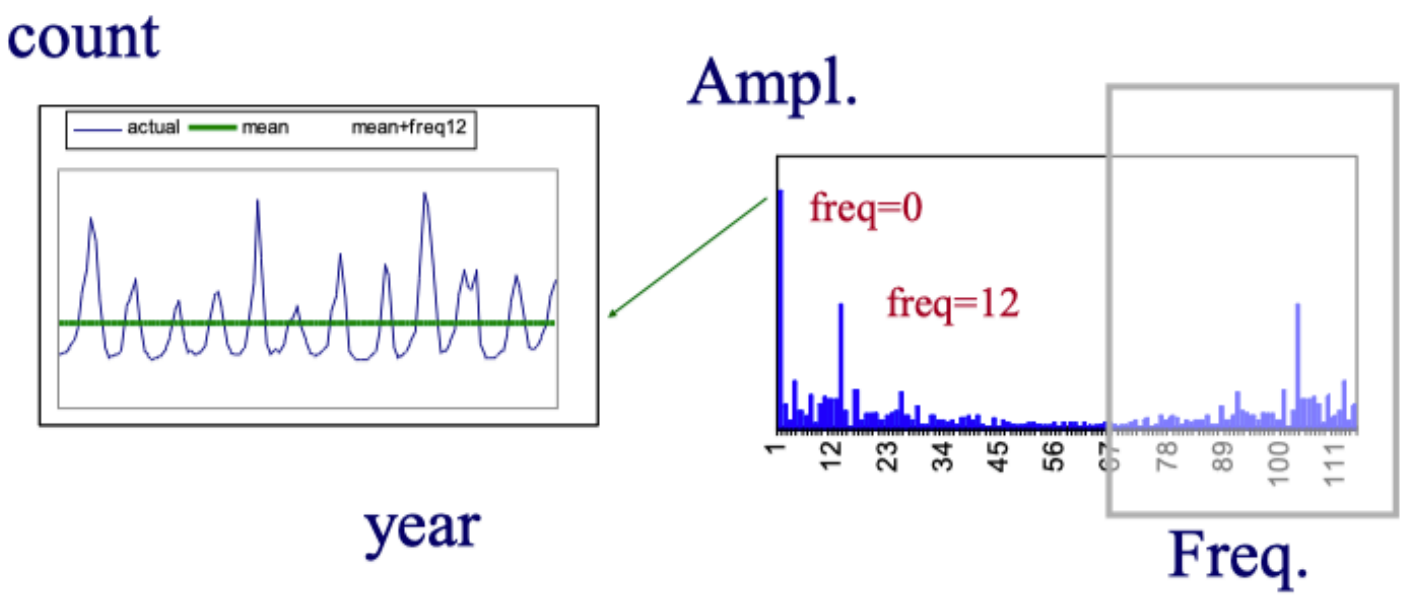
Freq.

The number of Canadian lynx for 120 years by Hudson company “Hunters”

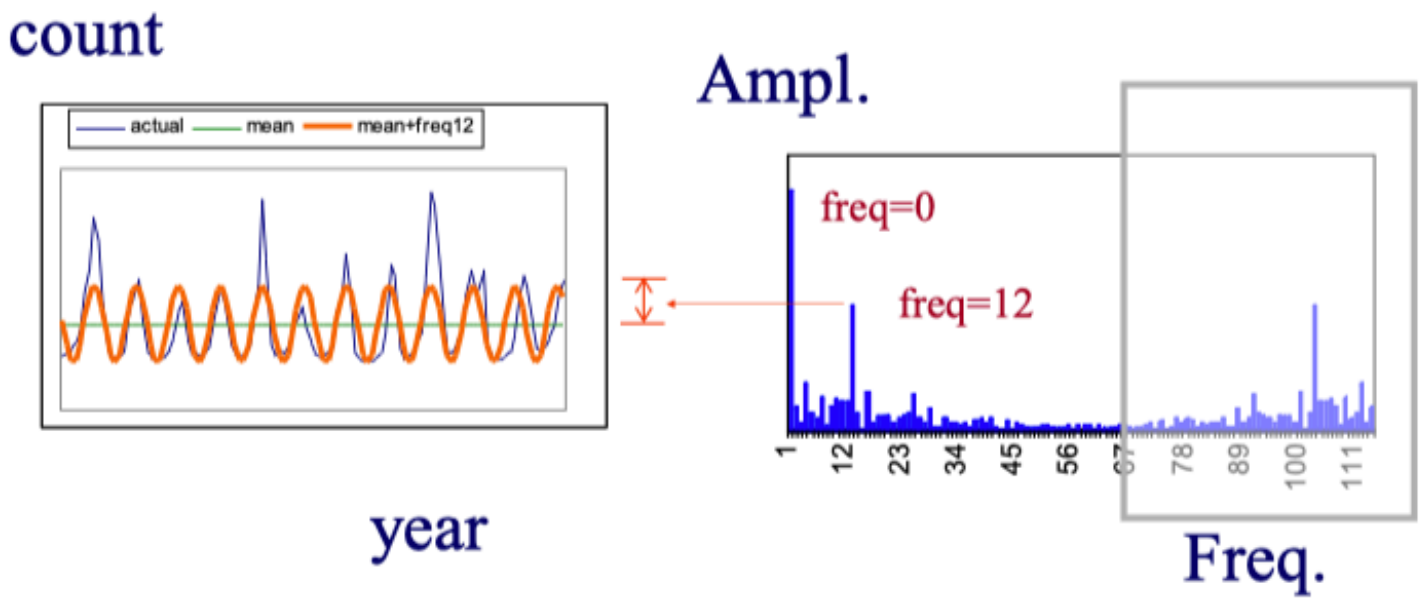
There is roughly 11 (roughly 12 spikes and they are 11 years apart from each other) years of periodicity. T

lynx in the nature (Average lynx).

DFT: Amplitude spectrum

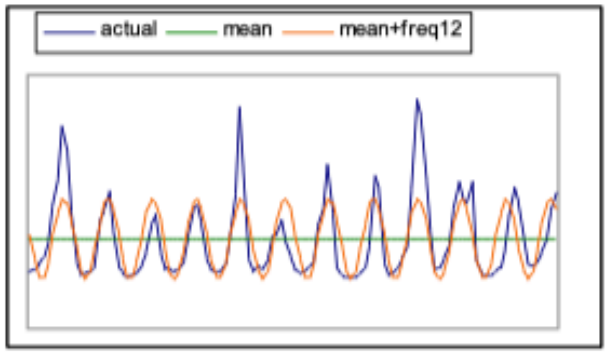


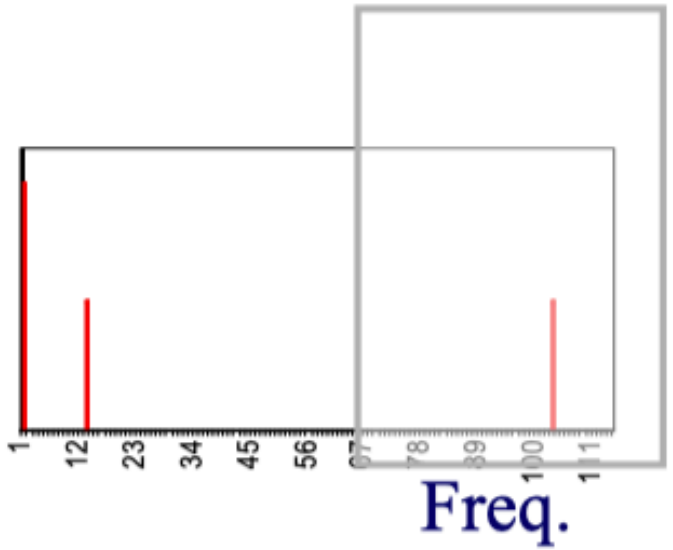
DFT: Amplitude spectrum



DFT: Amplitude spectrum

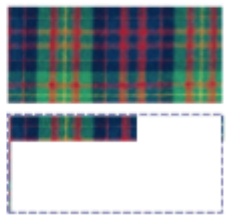
- excellent approximation, with only 2 frequencies!
- so what?

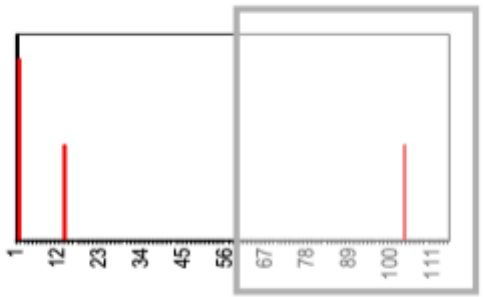




DFT: Amplitude spectrum

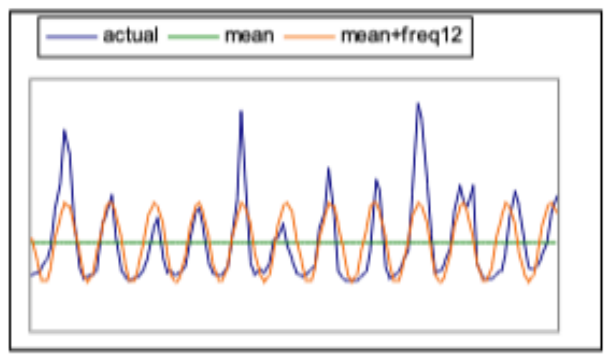
- excellent approximation, with only 2 frequencies!
- so what?
- A1: **(lossy) compression**
- A2: pattern discovery





DFT: Amplitude spectrum

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



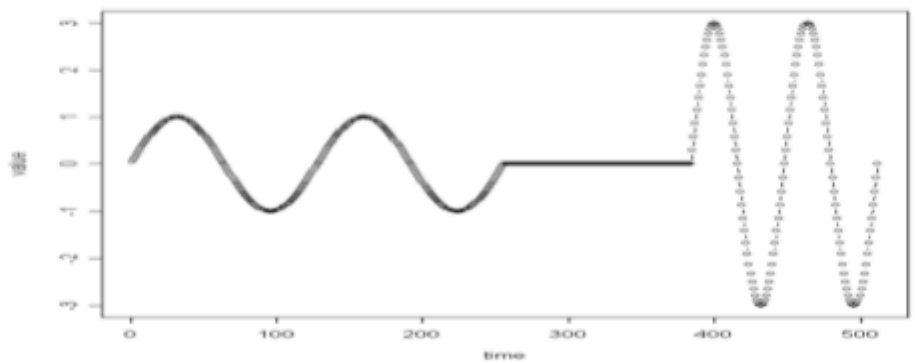
DFT - Conclusions

- 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)

Wavelets - DWT

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

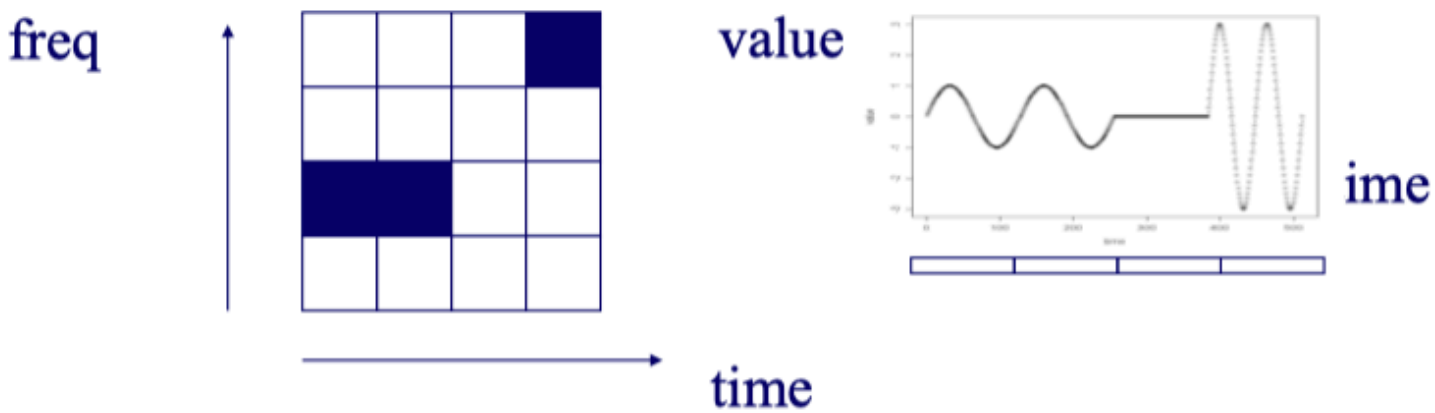
value



time

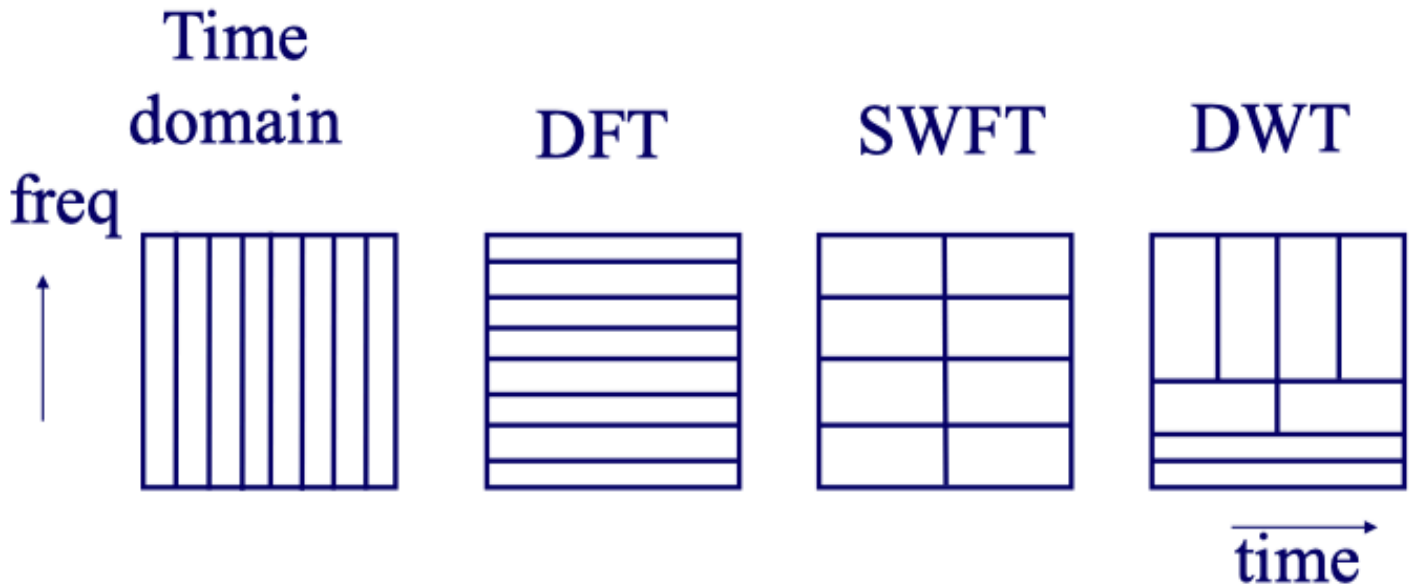
Wavelets - DWT

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



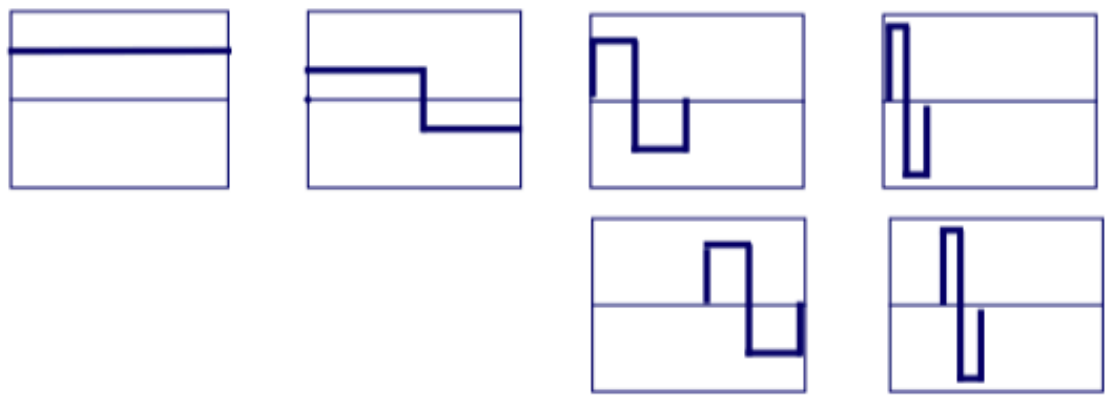
Wavelets - DWT

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



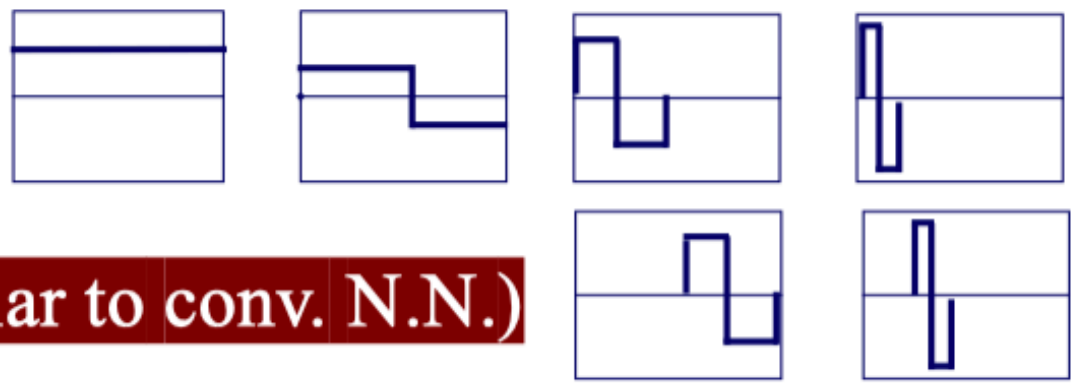
Haar Wavelets

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



Haar Wavelets

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

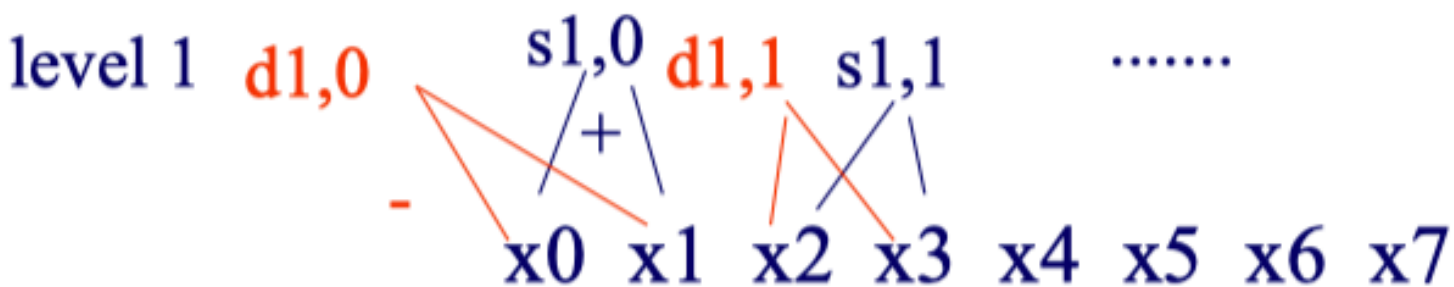


(Similar to conv. N.N.)

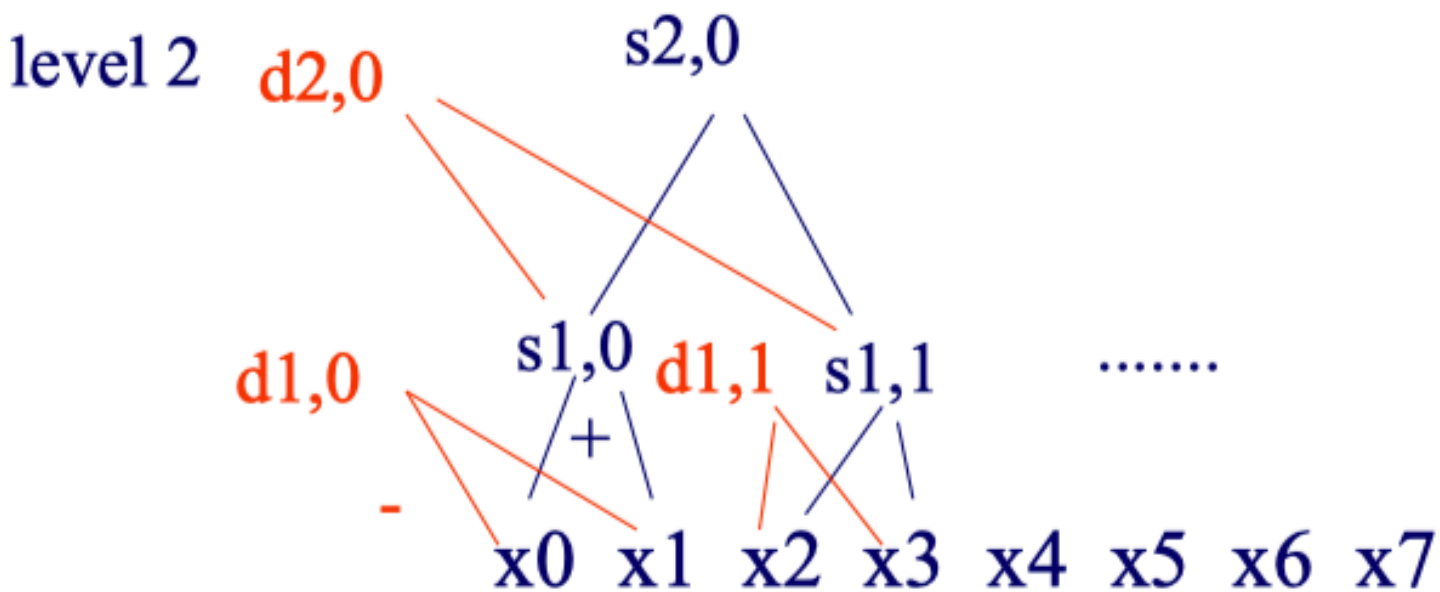
Wavelets - construction

x0 x1 x2 x3 x4 x5 x6 x7

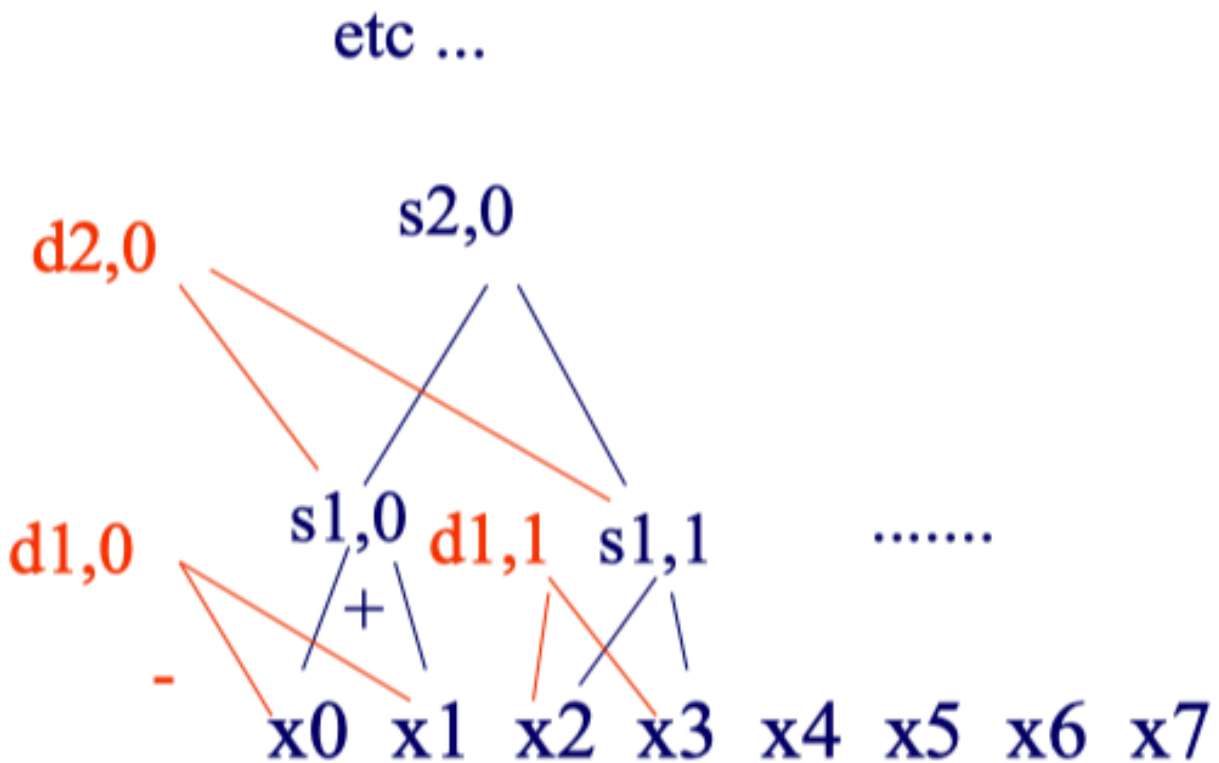
Wavelets - construction



Wavelets - construction



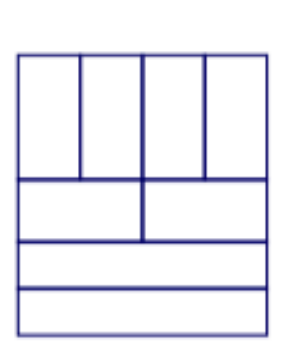
Wavelets - construction

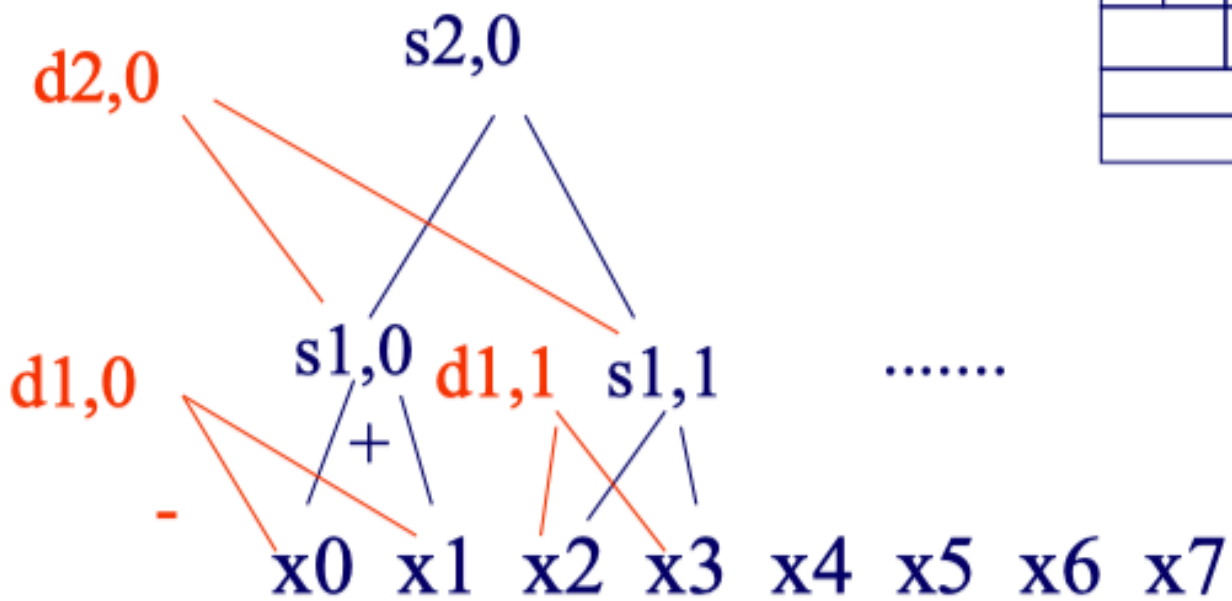


Wavelets - construction

Q: map each coefficient

on the time-freq. plane

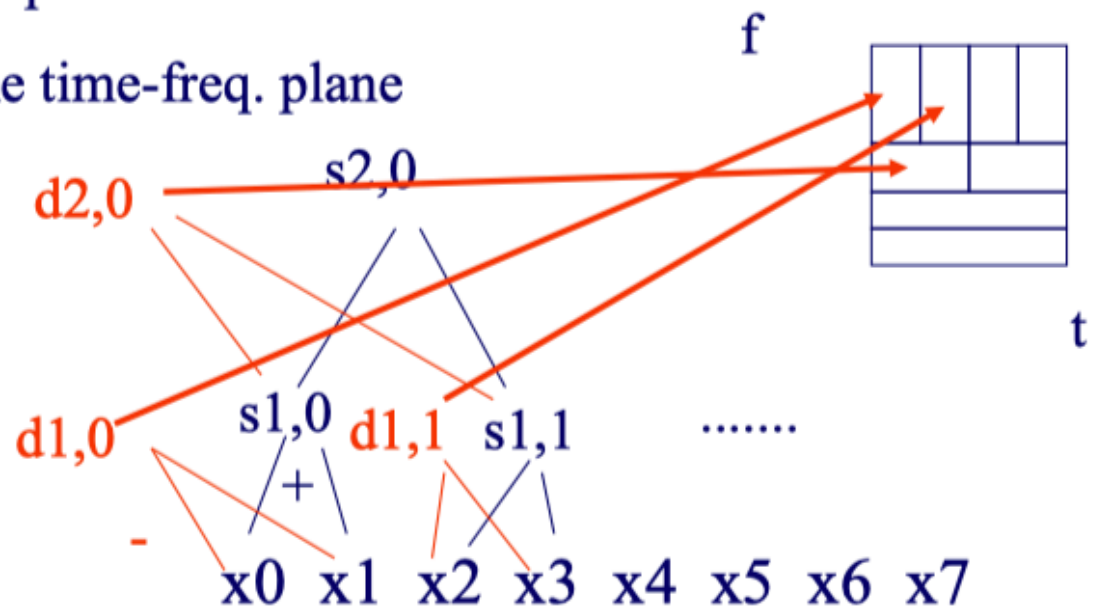


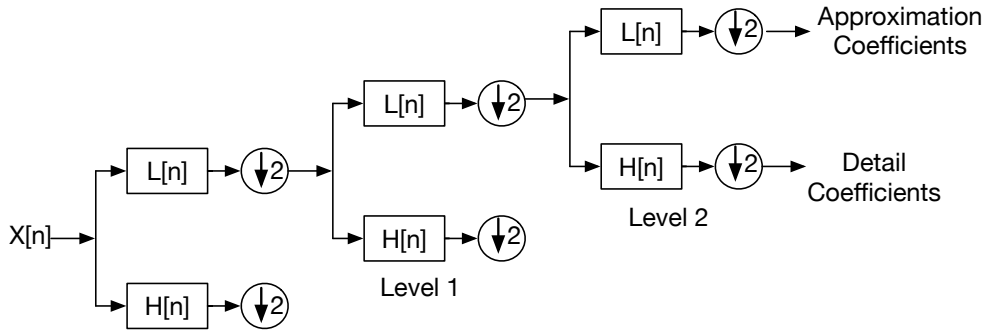


Wavelets - construction

Q: map each coefficient

on the time-freq. plane





Haar wavelets - code

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

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

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

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

Wavelets - construction

Observation1:

‘+’ can be some weighted addition

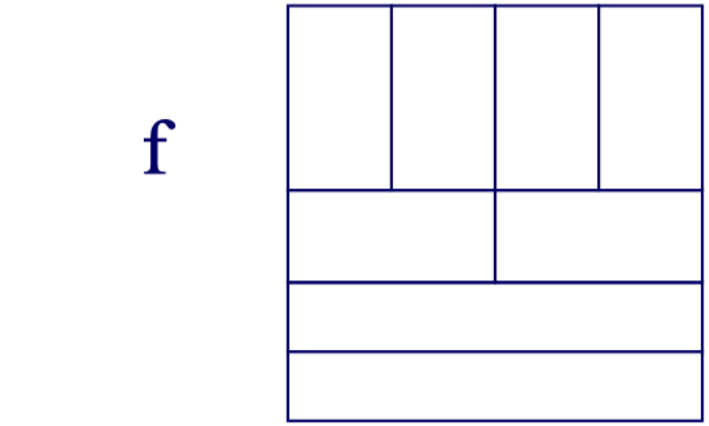
‘-’ is the corresponding weighted difference
(‘Quadrature mirror filters’)

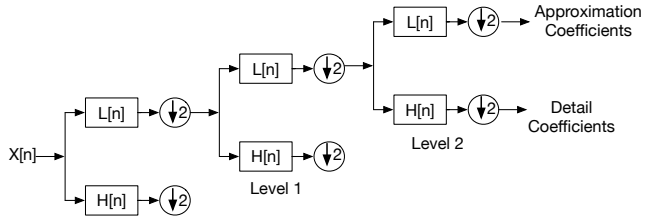
Observation2: unlike DFT/DCT,

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

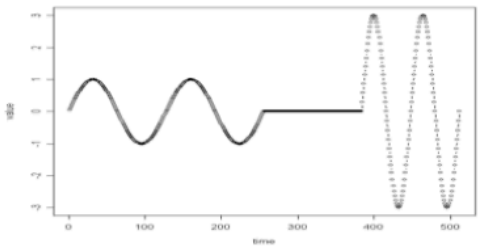
Wave

- Q: baritone/silence/soprano - DWT?





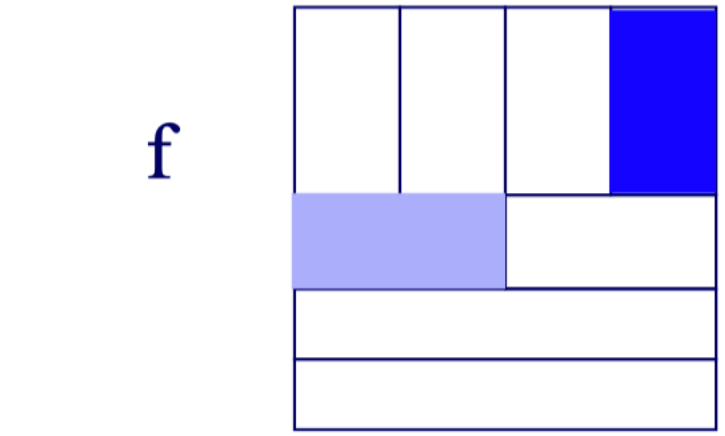
value

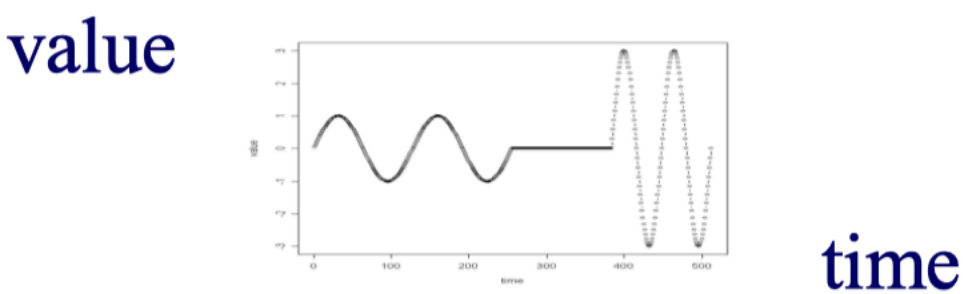


time

Wavelets - Example#1:

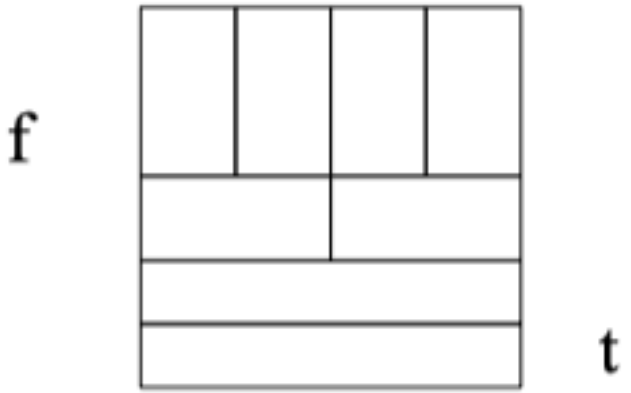
- Q: baritone/silence/soprano - DWT?

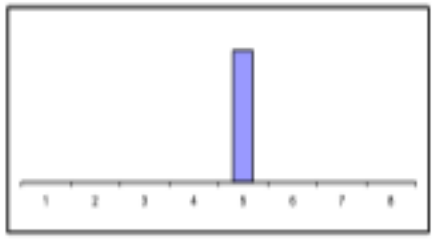




Wavelets – Example#2

- Q: spike - DWT?

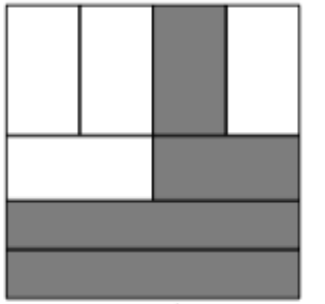


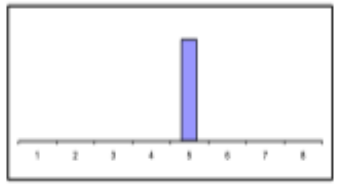


Wavelets - Drill#2:

- Q: spike - DWT?

f





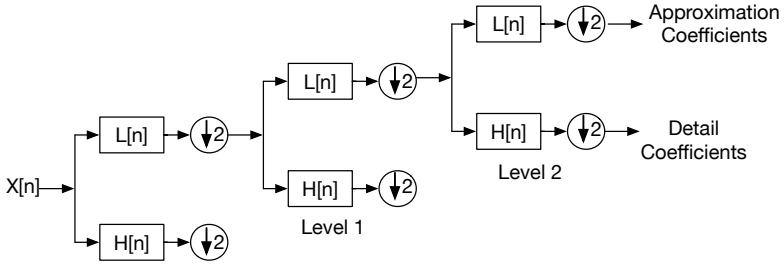
t

0.00 0.00 **0.71** 0.00

0.00 **0.50**

-0.35

0.35



X=[0,0,0,0,1,0,0,0]

Take average of the summation for approximations and subtraction for details of each pair, and multiply by √2. This example has got 3 levels.

A1=((0+0)/2) * √2, ((0+0)/2) * √2, ((1+0)/2) * √2, ((0+0)/2) * √2 = [0,0, 0.71,0]

D1= ((0-0)/2) * √2, ((0-0)/2) * √2, ((1-0)/2) * √2, ((0-0)/2) * √2 = [0, 0, 0.71, 0]

A2= ((0+0)/2) * √2, ((0.71+0)/2) * √2= [0, 0.5]

D2= ((0-0)/2) * √2, ((0.71-0)/2) * √2= [0, 0.5]

A3= ((0+0.5)/2) * √2= [0.35]

D3= ((0-0.5)/2) * √2= [-0.35]

Advantages of Wavelets

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


Overall Conclusions

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

Overall Conclusions

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

Resources: software and urls

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

Books

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

Additional Reading

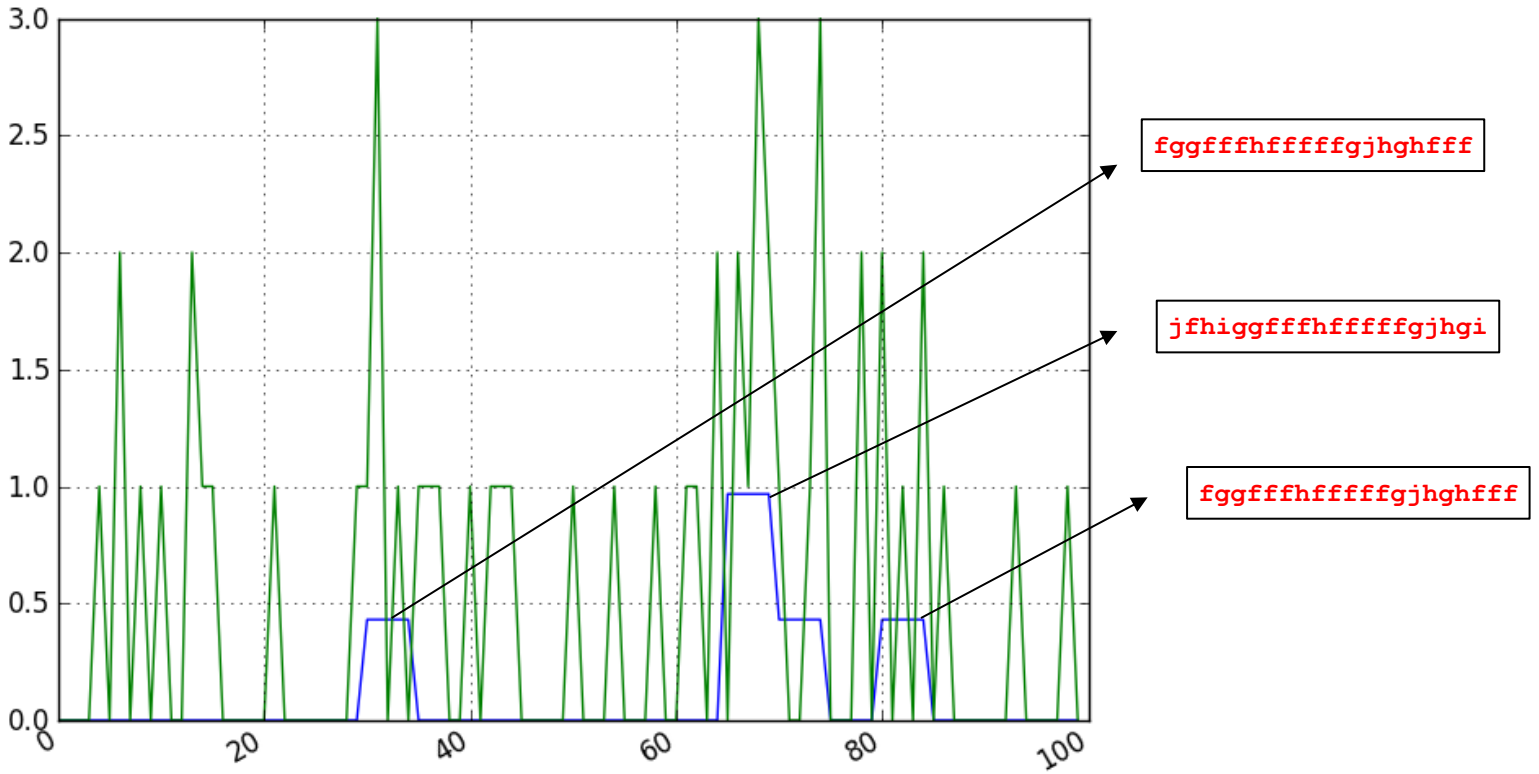
- [Gilbert+01] Anna C. Gilbert, Yannis Kotidis and S. Muthukrishnan and Martin Strauss, *Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries*, VLDB 2001

Symbolic Aggregate Approximation

- Different representation method that introduced for time-series data can be applied.
- The goal is to reduce the dimensionality (and size) of the data, to find patterns, detect anomalies, to query similar data;
- Dimensionality reduction techniques transform a data series with n items to a representation with w items where $w < n$.
  - *This functions are often lossy in comparison with solutions like normal compression that preserve all the data.*
- One of these techniques is called Symbolic Aggregation Approximation (SAX).
- SAX was originally proposed for symbolic representation of time-series data; it can be also used for symbolic representation of time-series sensor measurements.
- The computational foot-print of SAX is low; so it can be also used as a an in-network processing technique.

In-network processing

Using Symbolic Aggregate Approximation (SAX)



SAX Pattern (blue) with word length of 20 and a alphabet size of 10 over the original sensor time-series data (green)

Source: P. Barnaghi, F. Ganz, C. Henson, A. Sheth, "Computing Perception from Sensor Data", in Proc. of the IEEE Sensors 2012, Oct. 2012.

Symbolic Aggregate Approximation (SAX)

- SAX transforms time-series data into symbolic string representations.
- Symbolic Aggregate approXimation was proposed by Jessica Lin et al at the University of California –Riverside;
  - http://www.cs.ucr.edu/~eamonn/SAX.htm .
- It extends Piecewise Aggregate Approximation (PAA) symbolic representation approach.
- SAX algorithm is interesting for in-network processing in WSN because of its simplicity and low computational complexity.
- SAX provides reasonable sensitivity and selectivity in representing the data.
- The use of a symbolic representation makes it possible to use several other algorithms and techniques to process/utilise SAX representations such as hashing, pattern matching, suffix trees etc.

Processing Steps in SAX

- SAX transforms a time-series X of length n into the string of arbitrary length, where typically, using an alphabet A of size $a > 2$.
- The SAX algorithm has two main steps:
  - Transforming the original time-series into a PAA representation
  - Converting the PAA intermediate representation into a string.
- The string representations can be used for pattern matching, distance measurements, outlier detection, etc.

Piecewise Aggregate Approximation

- In PAA, to reduce the time series from *n* dimensions to *w* dimensions, the data is divided into *w* equal sized "*frames*."
- The mean value of the data falling within a frame is calculated and a vector of these values becomes the reduced representation.
- Before applying PAA, each time series needs to be normalised to achieve a mean of zero and a standard deviation of one.
  - The reason is to avoid comparing time series with different offsets and amplitudes;

Source: Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (DMKD '03). ACM, New York, NY, USA, 2-11.

SAX- normalisation before PAA

Timeseries (c): 2, 3, 4.5, 7.6, 4, 2, 2, 2, 3, 1

Mean (μ): $\mu = (2+3+4.5+7.6+4+2+2+2+3+1)/10 = 3.11$

Standard deviation (σ):

$(2-3.11)^2 = 1.2321$
$(3-3.11)^2 = 0.0121$
$(4.5-3.11)^2 = 1.9321$
$(7.6-3.11)^2 = 20.1601$
$(4-3.11)^2 = 0.7921$
$(2-3.11)^2 = 1.2321$
$(2-3.11)^2 = 1.2321$
$(2-3.11)^2 = 1.2321$
$(3-3.11)^2 = 0.0121$
$(1-3.11)^2 = 4.4521$

$1.2321+0.0121+ 1.9321+ 20.1601+ 0.7921+ 1.2321+ 1.2321+ 1.2321+ 0.0121+4.4521 = 33.5211$

$\sigma = \sqrt{(33.5211/10)} = 1.83087683911$

Normalisation

Timeseries (c): 2, 3, 4.5, 7.6, 4, 2, 2, 2, 3, 1

Normalised: $z_i = (c_i - \mu)/\sigma$

$\sigma = 1.83087683911$

$\mu = 3.11$

z1 = (2- 3.11)/1.83087683911 = -0.606
z2 = (3-3.11)/ 1.83087683911= -0.600
z3 = (4.5-3.11)/ 1.83087683911= 2.452
z4 = (7.6-3.11)/ 1.83087683911= -0.600
z5 = (4-3.11)/ 1.83087683911= 0.486
z6 = (2-3.11)/ 1.83087683911= -0.606
z7 = (2-3.11)/ 1.83087683911= -0.606
z8 = (2-3.11)/ 1.83087683911= -0.606
z9 = (3-3.11)/ 1.83087683911= -0.600
z10 = (1-3.11)/ 1.83087683911= -1.152

Normalised Timeseries (z): -0.606, -0.600, 2.452, -0.600, 0.486, -0.606, -0.606, -0.606 , -0.600, -1.152

PAA calculation

Timeseries (c): 2, 3, 4.5, 7.6, 4, 2, 2, 2, 3, 1

Normalised Timeseries (z): -0.606, -0.600, 2.452, -0.600, 0.486, -0.606, -0.606, -0.606 , -0.600, -1.152

PAA (w=5): -0.603, 0.926, -0.06, -0.606, 0.273

PAA to SAX Conversion

- Conversion of the PAA representation of a time-series into SAX is based on producing symbols that correspond to the time-series features with equal probability.
- The SAX developers have shown that time-series which are normalised (zero mean and standard deviation of 1) follow a Normal distribution (Gaussian distribution).
- The SAX method introduces breakpoints that divides the PAA representation to equal sections and assigns a letter from a given alphabet for each section.
    - For defining breakpoints, Normal inverse cumulative distribution function

Breakpoints in SAX

- "Breakpoints: breakpoints are a sorted list of numbers $B = \beta_1, \ldots, \beta_{a-1}$ such that the area under a N(0,1) Gaussian curve from β_i to $\beta_{i+1} = 1/a$".

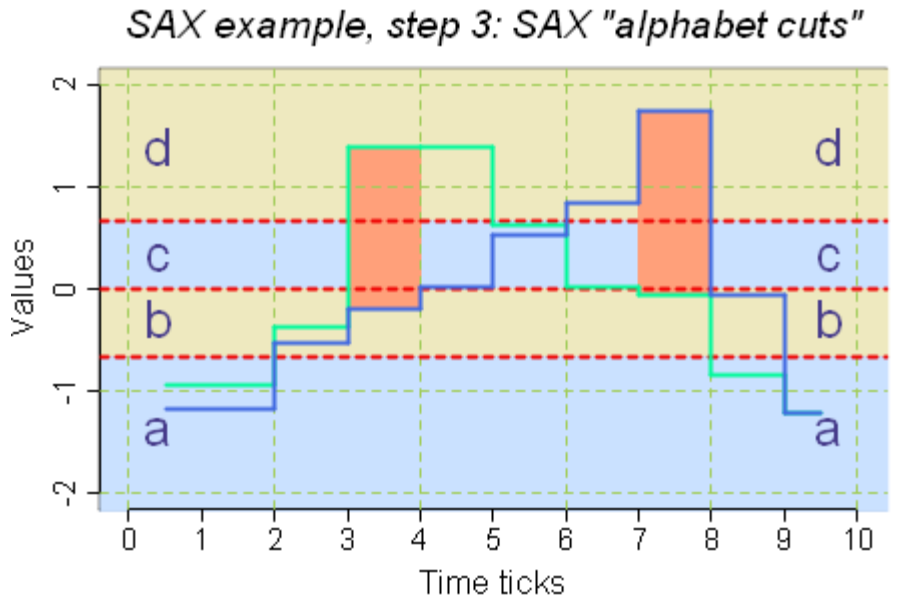
β_i \ a	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Table 3: A lookup table that contains the breakpoints that divide a Gaussian distribution in an arbitrary number (from 3 to 10) of equiprobable regions

Source: Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (DMKD '03). ACM, New York, NY, USA, 2-11.

Alphabet representation in SAX

- Let's assume that we will have 4 symbols alphabet: a,b,c,d
- As shown in the table in the previous slide, the cut lines for this alphabet (also shown as the thin red lines on the plot below) will be { -0.67, 0, 0.67 }



Source: JMOTIF Time series mining, http://code.google.com/p/jmotif/wiki/SAX

SAX Represetantion

Timeseries (c): 2, 3, 4.5, 7.6, 4, 2, 2, 2, 3, 1

Normalised Timeseries (z): -0.606, -0.600, 2.452, -0.600, 0.486, -0.606, -0.606, -0.606 , -0.600, -1.152

PAA (w=5): -0.603, 0.926, -0.06, -0.606, 0.273

Cut off ranges: *{-0.67, 0, 0.67}*

Alphabet: a ,b ,c, d

What will be the SAX representation of the given time series data?

SAX Represetantion

Timeseries (c): 2, 3, 4.5, 7.6, 4, 2, 2, 2, 3, 1

Normalised Timeseries (z): -0.606, -0.600, 2.452, -0.600, 0.486, -0.606, -0.606, -0.606 , -0.600, -1.152

PAA (w=5): -0.603, 0.926, -0.06, -0.606, 0.273

Cut off ranges: *{-0.67, 0, 0.67}*

Alphabet: a ,b ,c, d

SAX representation: *bdbbc*

Features of the SAX technique

- SAX divides a time series data into equal segments and then creates a string representation for each segment.
- The SAX patterns create the lower-level abstractions that are used to create the higher-level interpretation of the underlying data.
- The string representation of the SAX mechanism enables to compare the patterns using a specific type of string similarity function.

Control Charts

Outline

- Cumulative Sum
  - What is CUSUM?
  - How can we use it?
- Moving Average Charts
  - What is Moving Average Charts?
  - How can we use it?

Tabular CUSUM

Choose a target value μ_0 and a **reference value** K, and let

$$C_i^+ = \max\left[0, x_i - (\mu_0 + K) + C_{i-1}^+\right]$$
$$C_i^- = \max\left[0, (\mu_0 - K) - x_i + C_{i-1}^-\right]$$

starting with $C_0^+ = C_0^- = 0$.

The process is declared out of control if either C_i^+ or C_i^- exceeds a **decision interval** H.

Tabular CUSUM

- Screensh

Notes

If a change of process mean to μ_1 needs to be detected quickly, choose

$$K = \frac{|\mu_1 - \mu_0|}{2}.$$

Often $\mu_1 = \mu_0 \pm 1\sigma$ (a "one-sigma shift"), so $K = \sigma/2$.

Also, a decision interval $H = 5\sigma$ is often used.

Standardized cusum

Cusum charts are often based on *standardized* variables:

$$z_i = \frac{x_i - \mu_0}{\sigma}.$$

Tabular CUSUM – Example#1

- The first 20 of these observations were drawn at random from a normal distribution with mean μ_0 : 10 and standard deviation σ=1.

- suppose that the magnitude of the shift we are interested in detecting is 1.0 σ = 1.0(1.0) = 1.0. Therefore, the out-of-control value of the process mean is m 1 = 10 + 1 = 11. We will use a tabular CUSUM with K = 2 – 1 (because the shift size is 1.0 and σ = 1)

- H = 5 (because the recommended value of the decision interval is H = 5 σ = 5(1) = 5).

Sample, i	(a) x_i	(b) $x_i - 10$	(c) $C_i = (x_i - 10) + C_{i-1}$
1	9.45	–0.55	–0.55
2	7.99	–2.01	–2.56
3	9.29	–0.71	–3.27
4	11.66	1.66	–1.61
5	12.16	2.16	0.55
6	10.18	0.18	0.73
7	8.04	–1.96	–1.23
8	11.46	1.46	0.23
9	9.20	–0.80	–0.57
10	10.34	0.34	–0.23
11	9.03	–0.97	–1.20
12	11.47	1.47	0.27
13	10.51	0.51	0.78
14	9.40	–0.60	0.18
15	10.08	0.08	0.26
16	9.37	–0.63	–0.37
17	10.62	0.62	0.25
18	10.31	0.31	0.56
19	8.52	–1.48	–0.92
20	10.84	0.84	–0.08
21	10.90	0.90	0.82
22	9.33	–0.67	0.15
23	12.29	2.29	2.44
24	11.50	1.50	3.94
25	10.60	0.60	4.54
26	11.08	1.08	5.62
27	10.38	0.38	6.00
28	11.62	1.62	7.62
29	11.31	1.31	8.93
30	10.52	0.52	9.45

Tabular CUSUM – Example#1

- For first period:

$$C_1^+ = \max\left[0, x_1 - 10.5 + C_0^+\right]$$

$$C_1^- = \max\left[0, 9.5 - x_1 + C_0^-\right]$$

since $K = 0.5$ and $\mu_0 = 10$. Now $x_1 = 9.45$, so since $C_0^+ = C_0^- = 0$,

$$C_1^+ = \max[0, 9.45 - 10.5 + 0] = 0$$

and

$$C_1^- = \max[0, 9.5 - 9.45 + 0] = 0.05$$

Tabular CUSUM – Example#1

- For second period we would use:

$$C_2^+ = \max\left[0, x_2 - 10.5 + C_1^+\right]$$
$$= \max\left[0, x_2 - 10.5 + 0\right]$$

$$C_2^- = \max\left[0, 9.5 - x_2 + C_1^-\right]$$
$$= \max\left[0, 9.5 - x_2 + 0.05\right]$$

Since x 2 = 7.99, we obtain

$$C_2^+ = \max[0, 7.99 - 10.5 + 0] = 0$$

$$C_2^- = \max[0, 9.5 - 7.99 + 0.05] = 1.56$$

Tabular CUSUM – Example#1

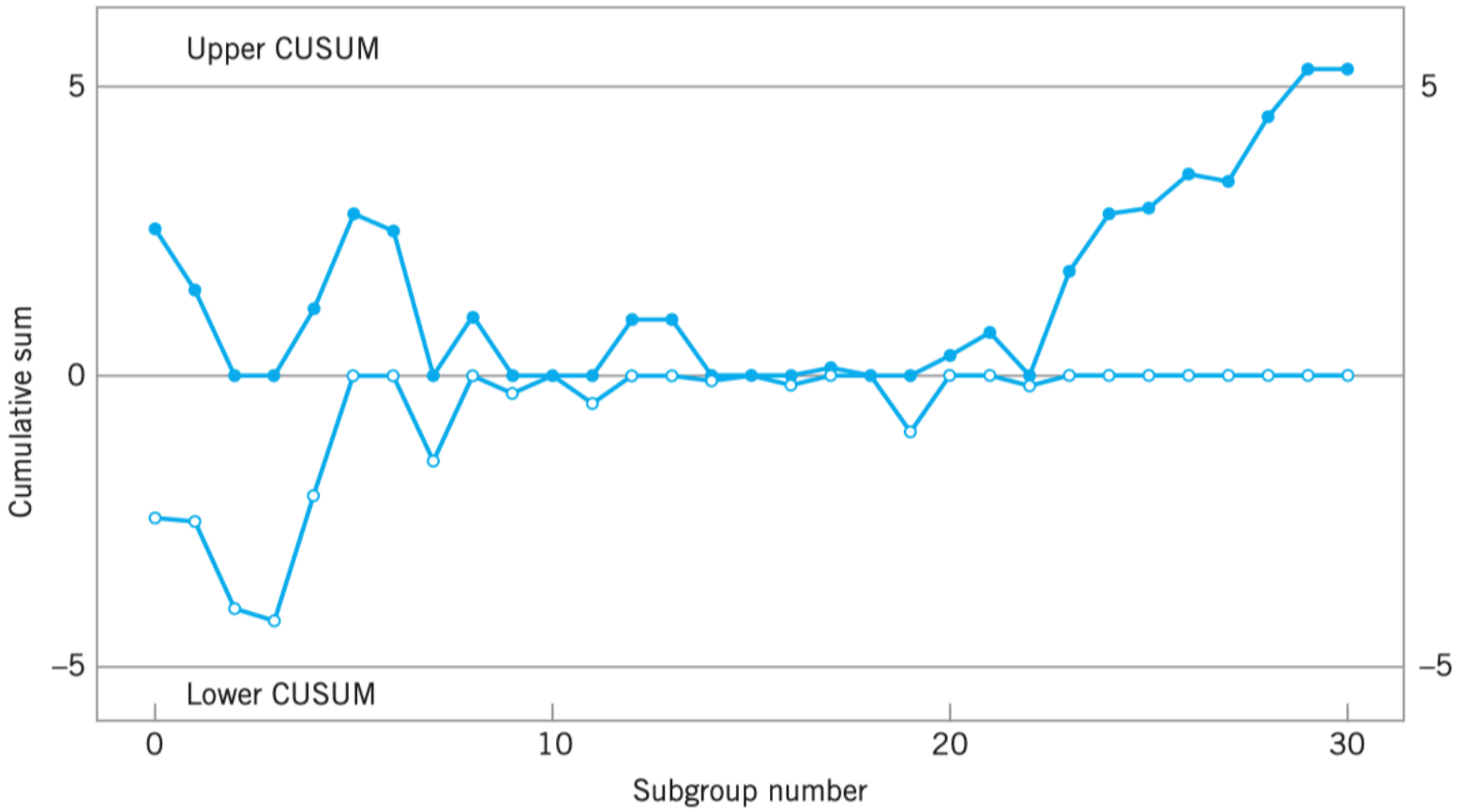
		(a)			(b)		
Period i	x_i	$x_i - 10.5$	C_i^+	N^+	$9.5 - x_i$	C_i^-	N^-
1	9.45	–1.05	0	0	0.05	0.05	1
2	7.99	–2.51	0	0	1.51	1.56	2
3	9.29	–1.21	0	0	0.21	1.77	3
4	11.66	1.16	1.16	1	–2.16	0	0
5	12.16	1.66	2.82	2	–2.66	0	0
6	10.18	–0.32	2.50	3	–0.68	0	0
7	8.04	–2.46	0.04	4	1.46	1.46	1
8	11.46	0.96	1.00	5	–1.96	0	0
9	9.20	–1.3	0	0	0.30	0.30	1
10	10.34	–0.16	0	0	–0.84	0	0
11	9.03	–1.47	0	0	0.47	0.47	1
12	11.47	0.97	0.97	1	–1.97	0	0
13	10.51	0.01	0.98	2	–1.01	0	0
14	9.40	–1.10	0	0	0.10	0.10	1
15	10.08	–0.42	0	0	–0.58	0	0
16	9.37	–1.13	0	0	0.13	0.13	1
17	10.62	0.12	0.12	1	–1.12	0	0
18	10.31	–0.19	0	0	–0.81	0	0
19	8.52	–1.98	0	0	0.98	0.98	1
20	10.84	0.34	0.34	1	–1.34	0	0
21	10.90	0.40	0.74	2	–1.40	0	0
22	9.33	–1.17	0	0	0.17	0.17	1
23	12.29	1.79	1.79	1	–2.79	0	0
24	11.50	1.00	2.79	2	–2.00	0	0
25	10.60	0.10	2.89	3	–1.10	0	0
26	11.08	0.58	3.47	4	–1.58	0	0
27	10.38	–0.12	3.35	5	–0.88	0	0
28	11.62	1.12	4.47	6	–2.12	0	0
29	11.31	0.81	5.28	7	–1.81	0	0
30	10.52	0.02	5.30	8	–1.02	0	0

The quantities N + and N − indicate the + number of consecutive periods that the CUSUMs C i or C i have been nonzero.

- Table show that the upper+ side CUSUM at period 29 is C 29 = 5.28. Since this is the first period at which C > H = 5, we would conclude that the process is out of control at that point.

- The tabular CUSUM also indicates when the shift probably occurred. The counter N + records the number of consecutive periods since the upper-side CUSUM C i + rose above the value of zero. Since N + = 7 at period 29, we would conclude that the process was last in control at period 29 − 7 = 22, so the shift likely occurred between periods 22 and 23.

Tabular CUSUM Graphical Representation – Example#



The Exponentially Weighted Moving Average Control Chart for Monitoring the Process Mean

- The exponentially weighted moving average (EWMA) control chart is also a good alternative to the CUSUM control chart when we are interested in detecting small shifts.

- The performance of the EWMA control chart is approximately equivalent to that of the cumulative sum control chart, and in some ways it is easier to set up and operate.

- As with the CUSUM, the EWMA is typically used with individual observations.

EWMA

- The exponentially weighted moving average is defined as

$$z_i = \lambda x_i + (1-\lambda) z_{i-1}$$

- where $0 < \lambda \leq 1$ is a constant and the starting value (required with the first sample at i = 1) is the process targ $z_0 = \mu_0$ at

- Sometimes the average of preliminary data is used as the starting value of the EWMA, that $z_0 = \bar{x}$.

$$\text{UCL} = \mu_0 + L\sigma\sqrt{\frac{\lambda}{(2-\lambda)}\left[1-(1-\lambda)^{2i}\right]}$$

$$\text{Center line} = \mu_0$$

$$\text{LCL} = \mu_0 - L\sigma\sqrt{\frac{\lambda}{(2-\lambda)}\left[1-(1-\lambda)^{2i}\right]}$$

- the factor L is the width of the control limits.

EWMA – Example#1

- Let's set up an EWMA control chart with λ = 0.10 and L = 2.7 for the time series data we used for CUSUM. The mean μ_0 = 10 and standard deviation σ = 1
- To illustrate the calculations, consider the first observation, x 1 = 9.45. The first value of the EWMA is

$$\begin{aligned} z_1 &= \lambda x_1 + (1-\lambda) z_0 \\ &= 0.1(9.45) + 0.9(10) \\ &= 9.945 \end{aligned}$$

- Therefore, z 1 = 9.945 is the f ıart. The second value of the EWMA is

$$\begin{aligned} z_2 &= \lambda x_2 + (1-\lambda) z_1 \\ &= 0.1(7.99) + 0.9(9.945) \\ &= 9.7495 \end{aligned}$$

EWMA – Example#1

- The other values of the EWMA statistic are computed similarly. For period i=1,

$$\text{UCL} = \mu_0 + L\sigma\sqrt{\frac{\lambda}{(2-\lambda)}\left[1-(1-\lambda)^{2i}\right]}$$

$$= 10 + 2.7(1)\sqrt{\frac{0.1}{(2-0.1)}\left[1-(1-0.1)^{2(1)}\right]}$$

$$= 10.27$$

$$\text{LCL} = \mu_0 - L\sigma\sqrt{\frac{\lambda}{(2-\lambda)}\left[1-(1-\lambda)^{2i}\right]}$$

$$= 10 - 2.7(1)\sqrt{\frac{0.1}{(2-0.1)}\left[1-(1-0.1)^{2(1)}\right]}$$

$$= 9.73$$

- For period

$$\text{UCL} = \mu_0 + L\sigma\sqrt{\frac{\lambda}{(2-\lambda)}\left[1-(1-\lambda)^{2i}\right]}$$

$$= 10 + 2.7(1)\sqrt{\frac{0.1}{(2-0.1)}\left[1-(1-0.1)^{2(2)}\right]}$$

$$= 10.36$$

$$\text{LCL} = \mu_0 - L\sigma\sqrt{\frac{\lambda}{(2-\lambda)}\left[1-(1-\lambda)^{2i}\right]}$$

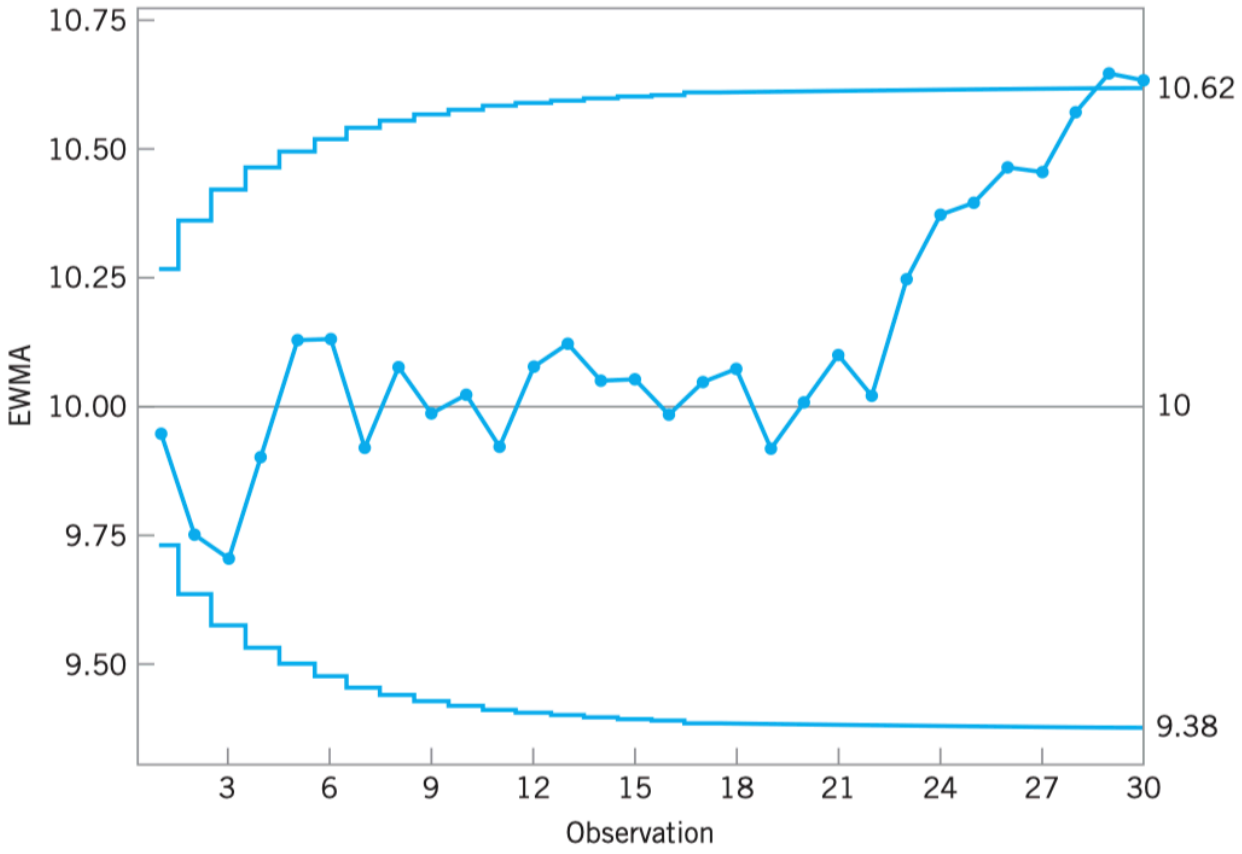
$$= 10 - 2.7(1)\sqrt{\frac{0.1}{(2-0.1)}\left[1-(1-0.1)^{2(2)}\right]}$$

$$= 9.64$$

EWMA – Example#1

Subgroup, i	* = Beyond Limits x_i	EWMA, z_i	Subgroup, i	* = Beyond Limits x_i	EWMA, z_i
1	9.45	9.945	16	9.37	9.98426
2	7.99	9.7495	17	10.62	10.0478
3	9.29	9.70355	18	10.31	10.074
4	11.66	9.8992	19	8.52	9.91864
5	12.16	10.1253	20	10.84	10.0108
6	10.18	10.1307	21	10.9	10.0997
7	8.04	9.92167	22	9.33	10.0227
8	11.46	10.0755	23	12.29	10.2495
9	9.2	9.98796	24	11.5	10.3745
10	10.34	10.0232	25	10.6	10.3971
11	9.03	9.92384	26	11.08	10.4654
12	11.47	10.0785	27	10.38	10.4568
13	10.51	10.1216	28	11.62	10.5731
14	9.4	10.0495	29	11.31	10.6468*
15	10.08	10.0525	30	10.52	10.6341*

EWMA – Example#1



Linear Dynamic Systems

Outline for LDS

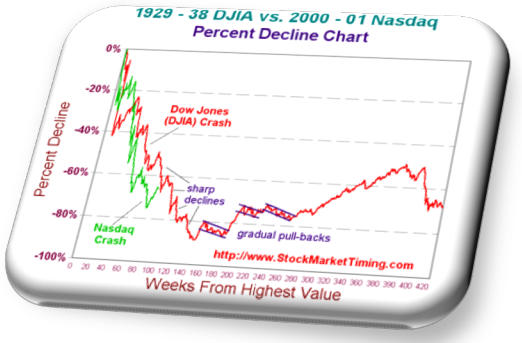
- Kalman Filter
  - How to use it?
- Kalman Smoother
  - How to use it?

State space

- **The state vector** contains all available information to describe the investigated system
  - usually multidimensional: $X(k) \in R^{N_x}$
- **The measurement vector** represents observations related to the state vector
  - Generally (but not necessarily) of lower dimension than the state vector

$Z(k) \in R^{N_z}$

State space



- Tracking:

$N_x = 3$

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

$N_x = 4$

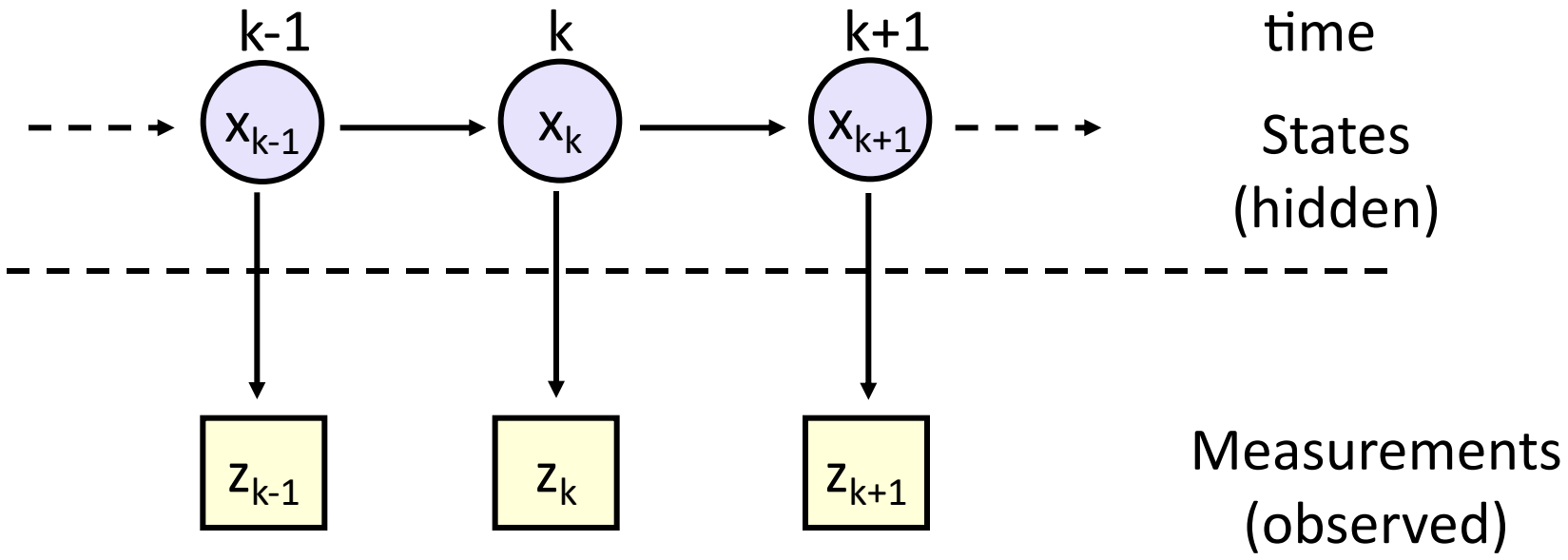
$$\begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$$

Econometrics:

- Monetary flow
- Interest rates
- Inflation
- …

Hidden Markov Model (HMM)

- the state is not directly visible, but output dependent on the state is visible



- A **state space model** or **SSM** is just like an HMM, except the hidden states are continuous. The model can be written in the following generic form:

$$\begin{aligned} \mathbf{z}_t &= g(\mathbf{u}_t, \mathbf{z}_{t-1}, \boldsymbol{\epsilon}_t) \\ \mathbf{y}_t &= h(\mathbf{z}_t, \mathbf{u}_t, \boldsymbol{\delta}_t) \end{aligned}$$

- Where *z(t)* is the hidden [illegible] optional input or control signal, *y(t)* is the observation, *g* **transition model**, *h* is the **observation model**, *ε(t)* is the system noise at time *t*, and *δ(t)* observation noise at time t.

- We assume that all parameters of the model, *ϑ*, are known; if not, they can be included into the hidden state.

- An important special case of an SSM is where all the Conditional Probability Distributions (CPDs) are linear-Gaussian. In other words, we assume
  - The transition model is a linear function

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \boldsymbol{\epsilon}_t$$

  - The observation model is a linear function

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \boldsymbol{\delta}_t$$

  - The system noise is Gaussian

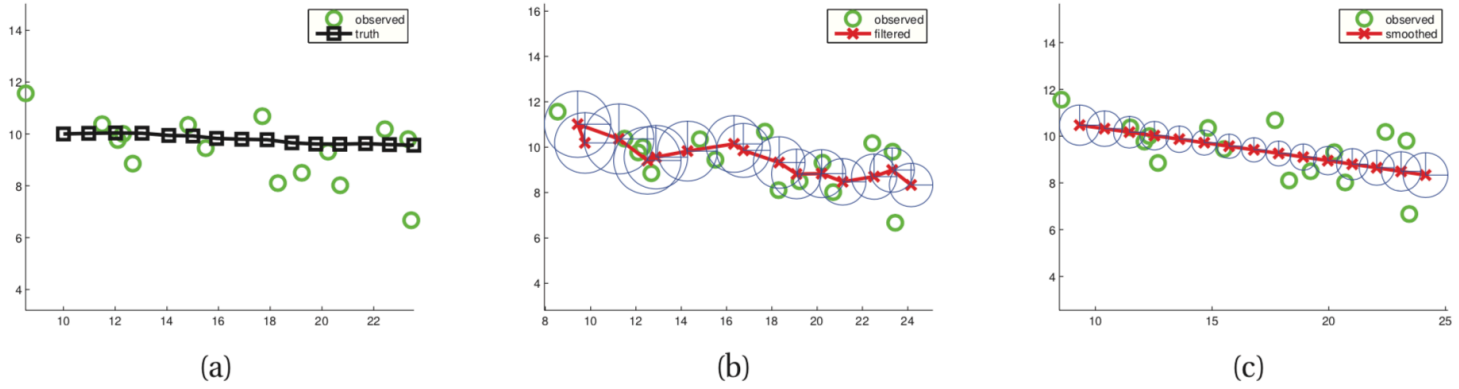
$$\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$$

  - The observation noise is Gaussian

$$\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$$

  - This model is called a **linear-Gaussian SSM** (**LG-SSM**) or a **linear dynamical system** (**LDS**). If the parameters θt = (At, Bt, Ct, Dt, Qt, Rt) are independent of time, the model is called **stationary**.

- Illustration of Kalman filtering and smoothing. (a) Observations (green cirles) are generated by an object moving to the right (true location denoted by black squares). (b) Filtered estimated is shown by dotted red line. Red cross is the posterior mean, blue circles are 95% confidence ellipses derived from the posterior covariance. For clarity, we only plot the ellipses every other time step. (c) Same as (b), but using offline Kalman smoothing.



Kevin P. Murphy, **State Space Models,** Machine Learning A Probabilistic Perspective, 2012

- Consider an object moving in a 2D plane. Let z(1t) and z(2t) be the horizontal and vertical locations of the object, and $\dot{z}(1t)$ and $(\dot{z}2t)$ be the corresponding velocity. We can represent this as a state vector $z(t) \in$ R4 as follows:

$$\mathbf{z}_t^T = \begin{pmatrix} z_{1t} & z_{2t} & \dot{z}_{1t} & \dot{z}_{2t} \end{pmatrix}$$

- Let us assume that the object is moving at constant velocity, but is "perturbed" by random Gaussian noise (e.g., due to the wind). Thus we can model the system dynamics as follows:

$$\begin{aligned} \mathbf{z}_t &= \mathbf{A}_t \mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t \\ \begin{pmatrix} z_{1t} \\ z_{2t} \\ \dot{z}_{1t} \\ \dot{z}_{2t} \end{pmatrix} &= \begin{pmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z_{1,t-1} \\ z_{2,t-1} \\ \dot{z}_{1,t-1} \\ \dot{z}_{2,t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \\ \epsilon_{3t} \\ \epsilon_{4t} \end{pmatrix} \end{aligned}$$

- where $\varepsilon t \sim N(0, Q)$ is the system noise, and Δ is the **sampling period**. This says that the new location zj,t is the old location $zj,t-1$ plus Δ times the old velocity $\dot{z}$ j,t−1, plus random noise, εjt, for j = 1 : 2. Also, the new velocity $\dot{z}j,t$ is the old velocity $\dot{z}$ j,t−1 plus random noise, εjt, for j = 3 : 4. This is called a **random accelerations model**, since the object moves according to Newton's laws, but is subject to random changes in velocity.

- Now suppose that we can observe the location of the object but not its velocity. Let yt ∈ R2 represent our observation, which we assume is subject to Gaussian noise. We can model this as follows:

$$\mathbf{y}_t = \mathbf{C}_t\mathbf{z}_t + \boldsymbol{\delta}_t$$

- where $\delta t \sim N(0, R)$ is th

$$\begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} z_{1t} \\ z_{2t} \\ \dot{z}_{1t} \\ \dot{z}_{2t} \end{pmatrix} + \begin{pmatrix} \delta_{1t} \\ \delta_{2t} \\ \delta_{3t} \\ \delta_{4t} \end{pmatrix}$$

Inference in LG-SSM

- **The Kalman filtering algorithm** is an algorithm for exact Bayesian filtering for linear-Gaussian state space models.
- The prediction step is straightforward to derive:

$$\begin{aligned}\boldsymbol{\mu}_{t|t-1} &\triangleq \mathbf{A}_t\boldsymbol{\mu}_{t-1} + \mathbf{B}_t\mathbf{u}_t \\ \boldsymbol{\Sigma}_{t|t-1} &\triangleq \mathbf{A}_t\boldsymbol{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{Q}_t\end{aligned}$$

- The measurement step can be computed using Bayes rule, as follows

$$\begin{aligned}\boldsymbol{\mu}_t &= \boldsymbol{\mu}_{t|t-1} + \mathbf{K}_t\mathbf{r}_t \\ \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\boldsymbol{\Sigma}_{t|t-1}\end{aligned}$$

- where rt is th ıe difference between our predicted observation and the actual observation:

$$\begin{aligned}\mathbf{r}_t &\triangleq \mathbf{y}_t - \hat{\mathbf{y}}_t \\ \hat{\mathbf{y}}_t &\triangleq \mathbb{E}\left[\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}\right] = \mathbf{C}_t\boldsymbol{\mu}_{t|t-1} + \mathbf{D}_t\mathbf{u}_t\end{aligned}$$

- and *K(t)* is the

- where *δt* - $\mathbf{K}_t \triangleq \boldsymbol{\Sigma}_{t|t-1}\mathbf{C}_t^T\mathbf{S}_t^{-1}$ rm $\mathbf{S}_t \triangleq \mathbf{C}_t\boldsymbol{\Sigma}_{t|t-1}\mathbf{C}_t^T + \mathbf{R}_t$ noise sources.

- **Kalman Smoothing Algorithm**.
  - Kalman Filter is useful for online inference problems, such as tracking. However, in an offline setting, we can wait until all the data has arrived, and then compute *p(zt|y1:T).* By conditioning on past and future data, our uncertainty will be significantly reduced.
  - Kalman filtering can be regarded as message passing on a graph, from left to right. When the messages have reached the end of the graph, we have successfully computed *p(zT |y1:T)*. Now we work backwards, from right to left, sending information from the future back to the past, and them combining the two information sources.

$$
\begin{aligned}
\boldsymbol{\mu}_{t|T} &= \boldsymbol{\mu}_{t|t} + \mathbf{J}_t(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t}) \\
\boldsymbol{\Sigma}_{t|T} &= \boldsymbol{\Sigma}_{t|t} + \mathbf{J}_t(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{t+1|t})\mathbf{J}_t^T \\
\mathbf{J}_t &\triangleq \boldsymbol{\Sigma}_{t|t}\mathbf{A}_{t+1}^T\boldsymbol{\Sigma}_{t+1|t}^{-1}
\end{aligned}
$$

- where *J(t)* is the backwa e initialized from *μ(T|T)* and *Σ(T|T)* from the Kalman filter.

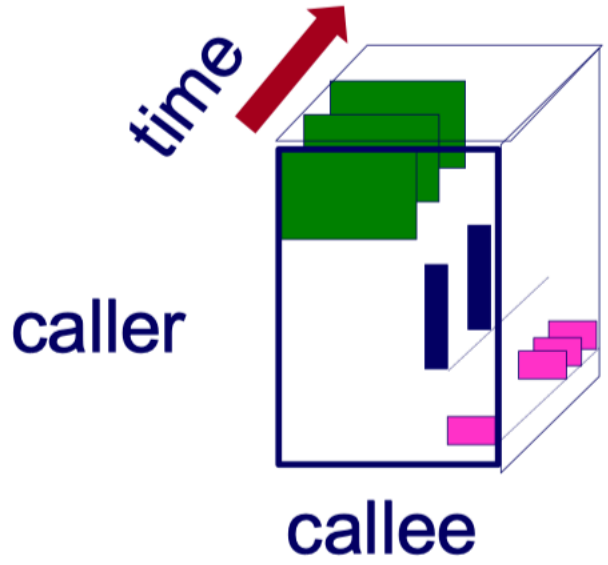
Tensor Decompositions – Time Evolving Graphs

Outline for Tensor Decompositions

- Tucker Decomposition
  - What is it?
- CP Decomposition
  - What is it?

Problem: co-evolving graphs

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



- Q: what is a tensor?

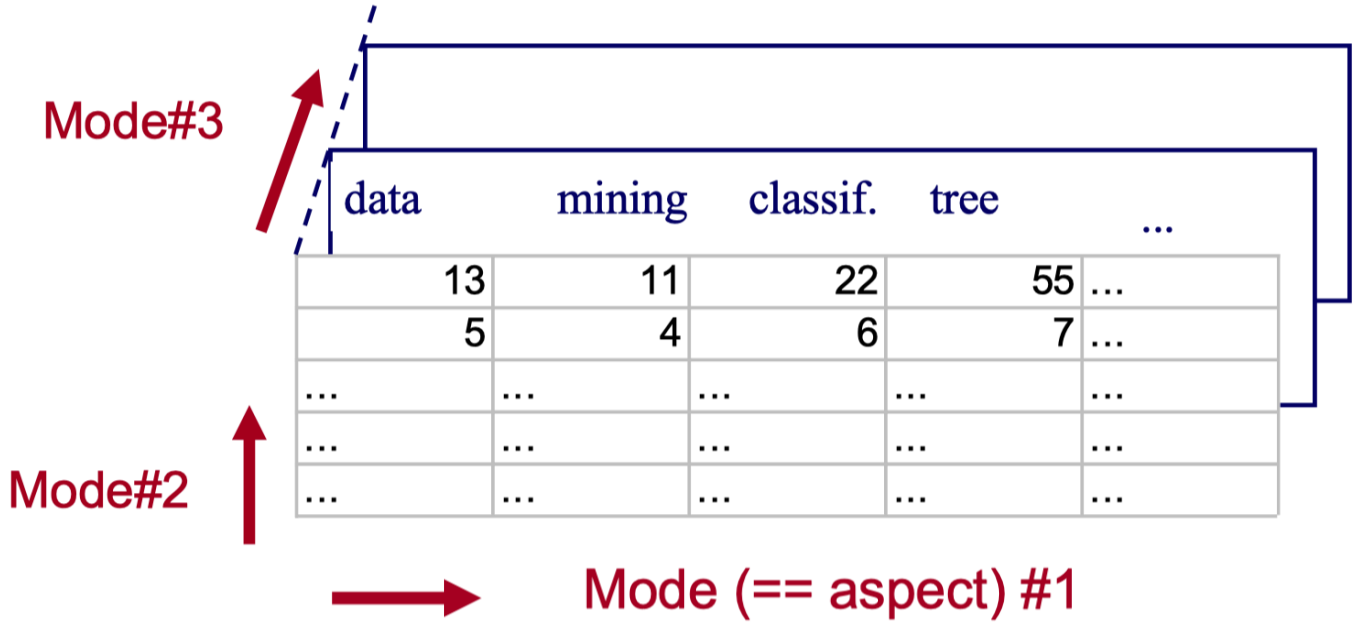
Tensor examples

- A: N-D generalization of matrix:

arxiv' 17	data	mining	classif.	tree	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

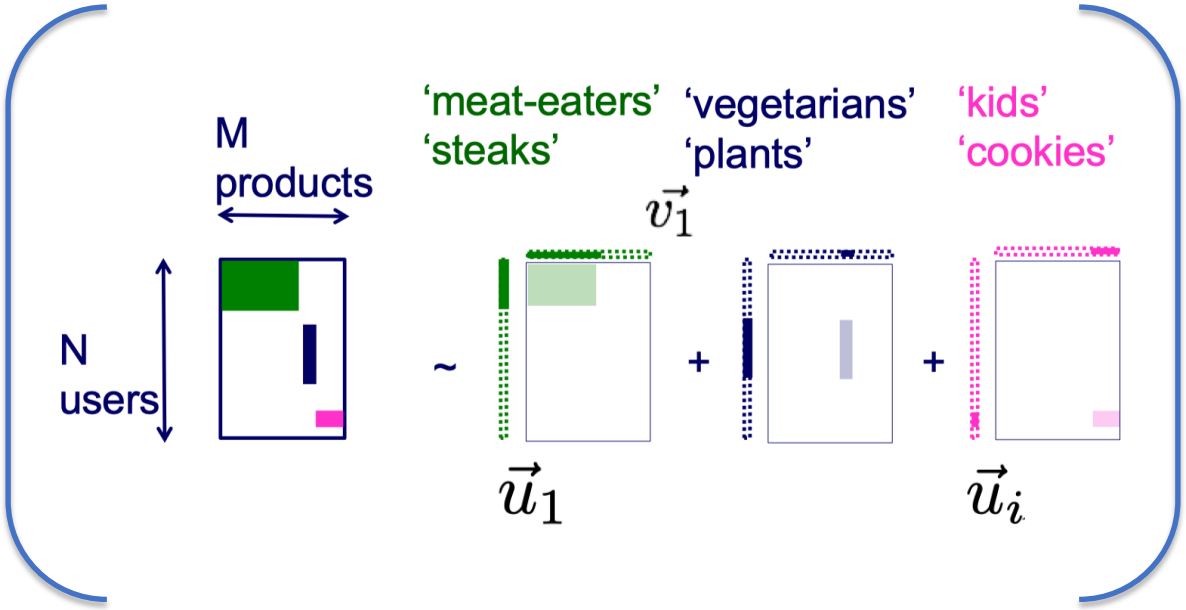
Tensors are useful for 3 or more modes

- Terminology: ‘mode’ (or ‘aspect’):



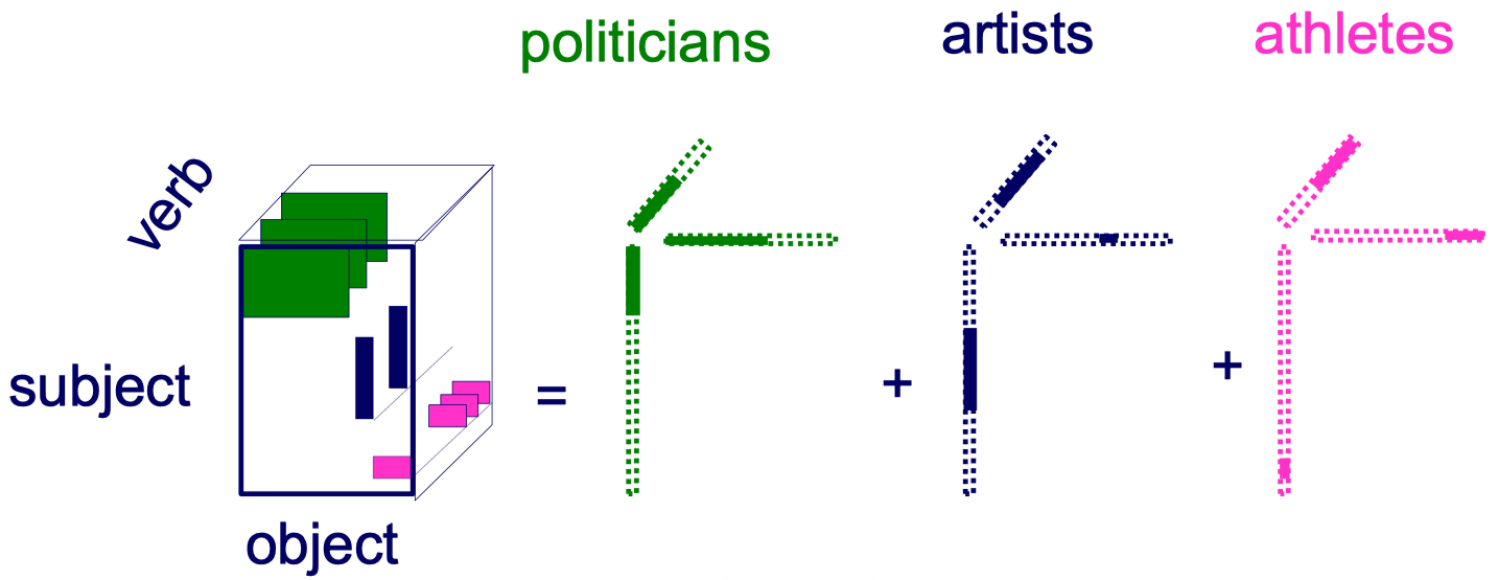
Tensor Basics

- Recall (SVD) matrix factorization finds blocks:



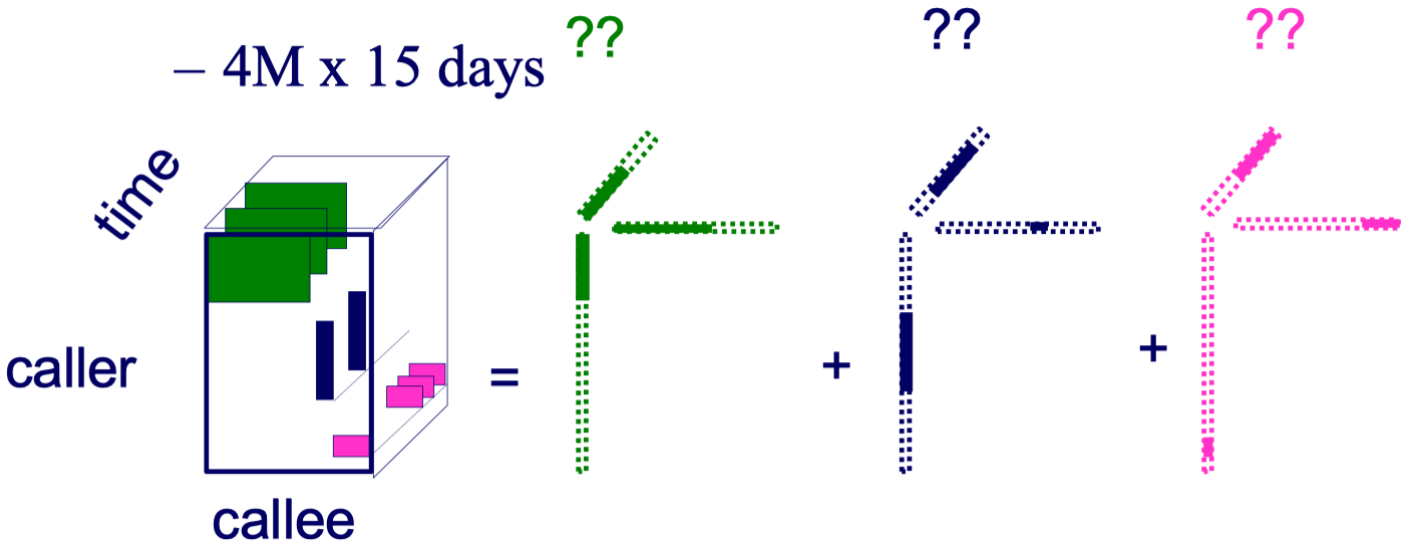
Tensor Factorisation

- PARAFAC decomposition



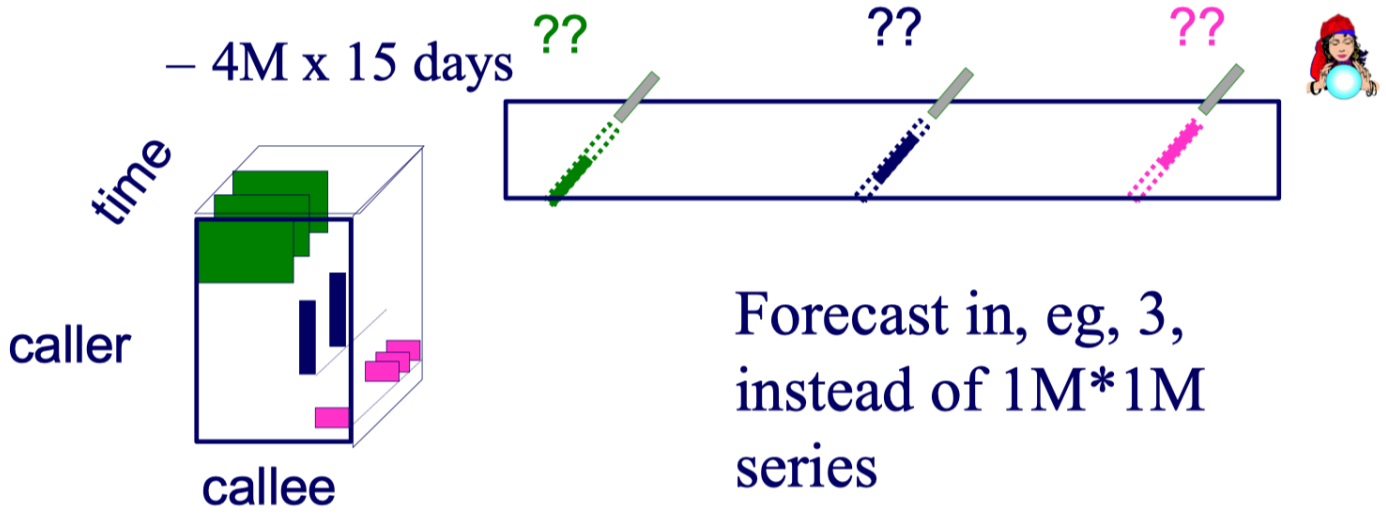
Tensor Factorisation

- PARAFAC decomposition
- Results for who-calls-whom-when



Tensor Factorisation

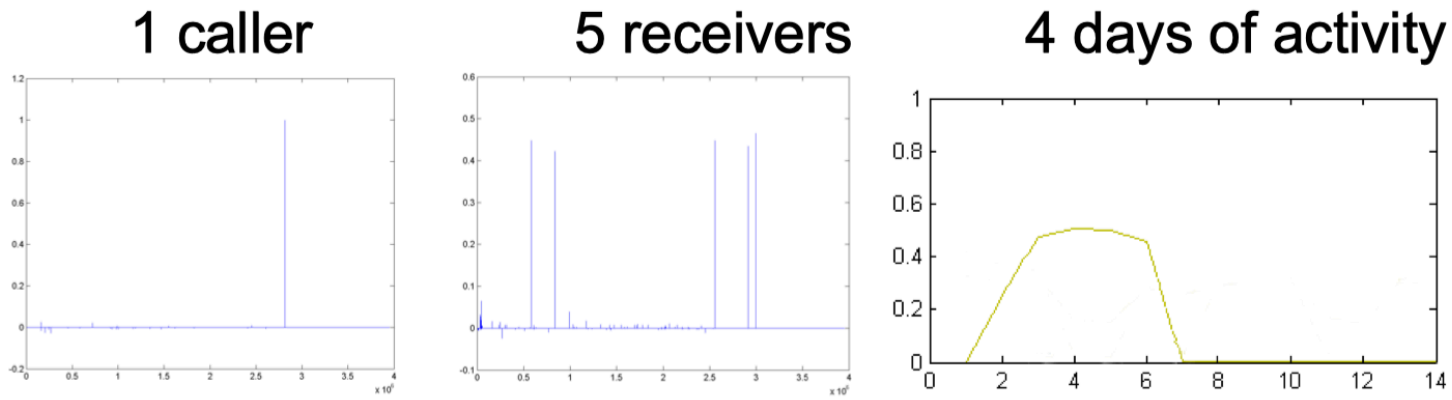
- PARAFAC decomposition
- Results for who-calls-whom-when



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

Detection in time-evolving graphs

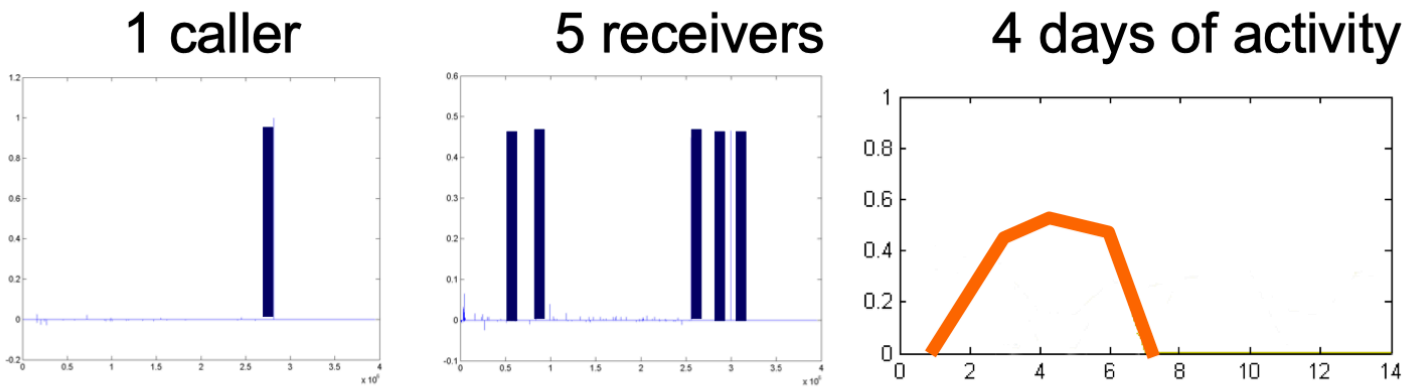
- Strange communities in phone call data:
  - European country, 4M clients, data over 2 weeks



~200 calls to EACH receiver on EACH day!

Detection in time-evolving graphs

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



~200 calls to EACH receiver on EACH day!

- Overall conclusions
  - P1.1. Similarity search: Euclidean/timewarping; feature extraction and SAMs
  - P1.2. Signal processing: DFT, DWT are powerful tools
  - P1.3. Control Charts: CUSUM, EWMA
  - P1.4. Linear Dynamic Systems P1.5. Tensors: PARAFAC etc

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

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

- To find outliers, we need to have either model based forecasts or (model-free) statistical change detection systems
  - (outlier = too far away from our forecast)

Thank you!