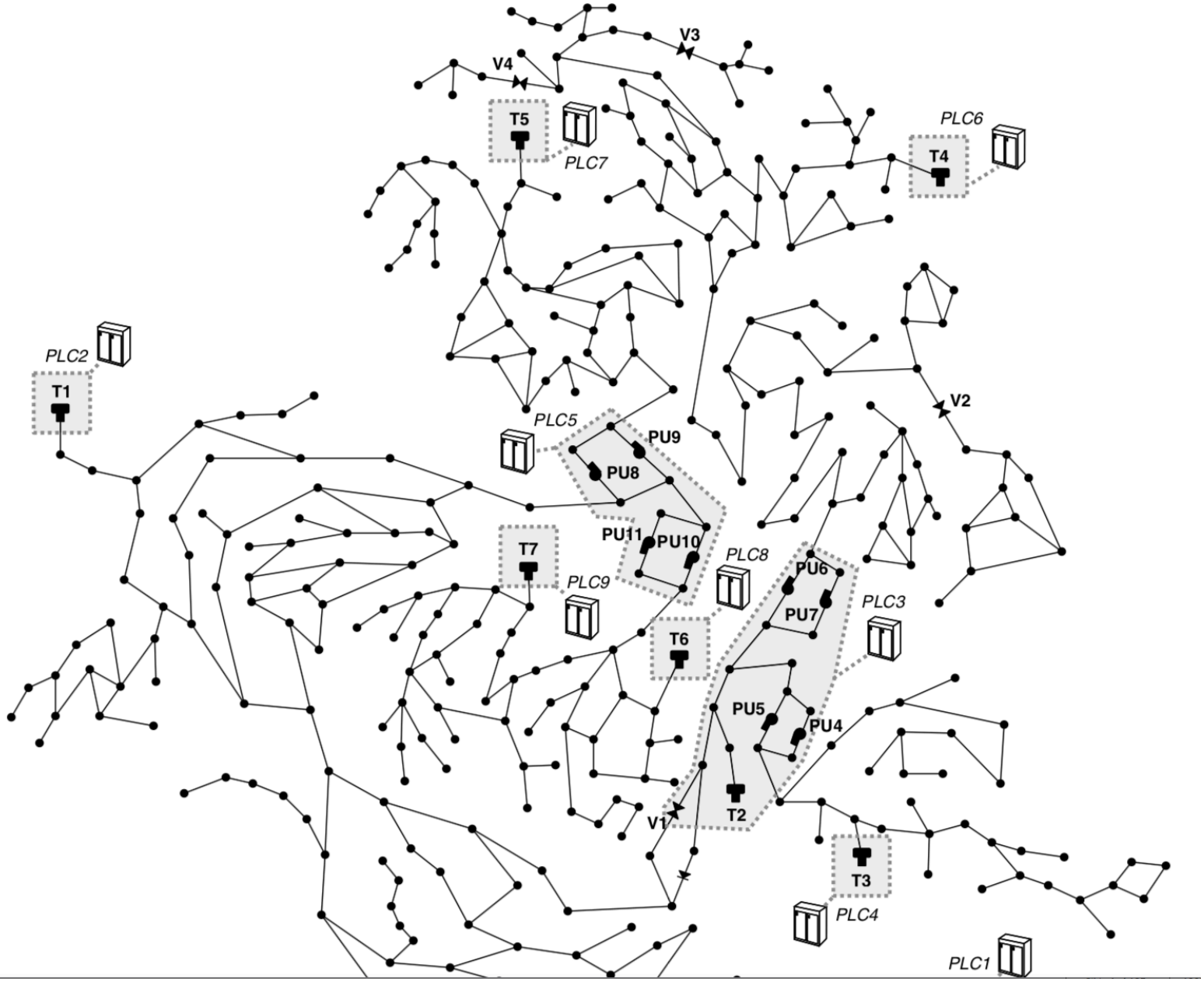
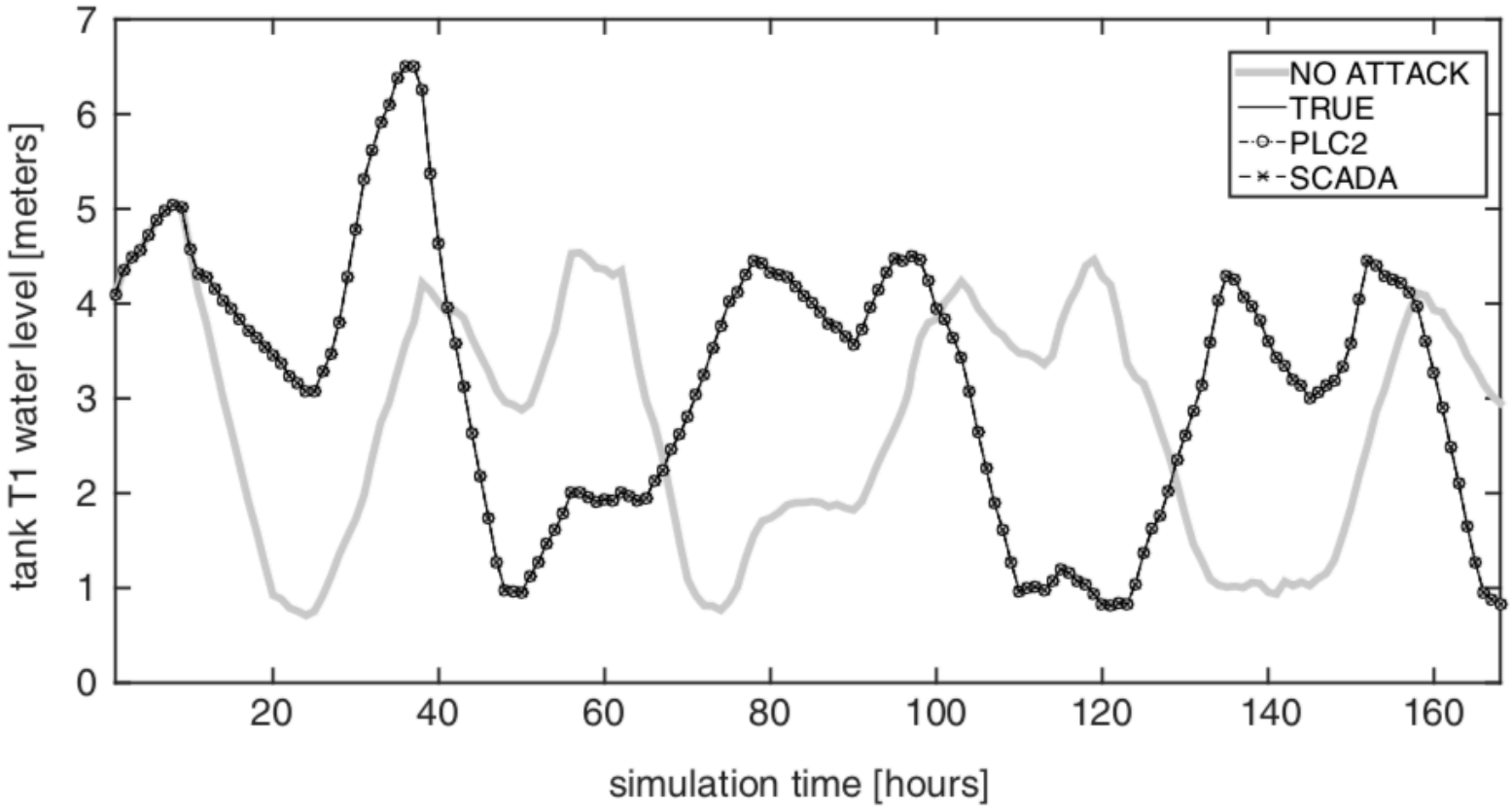
Sequential data mining

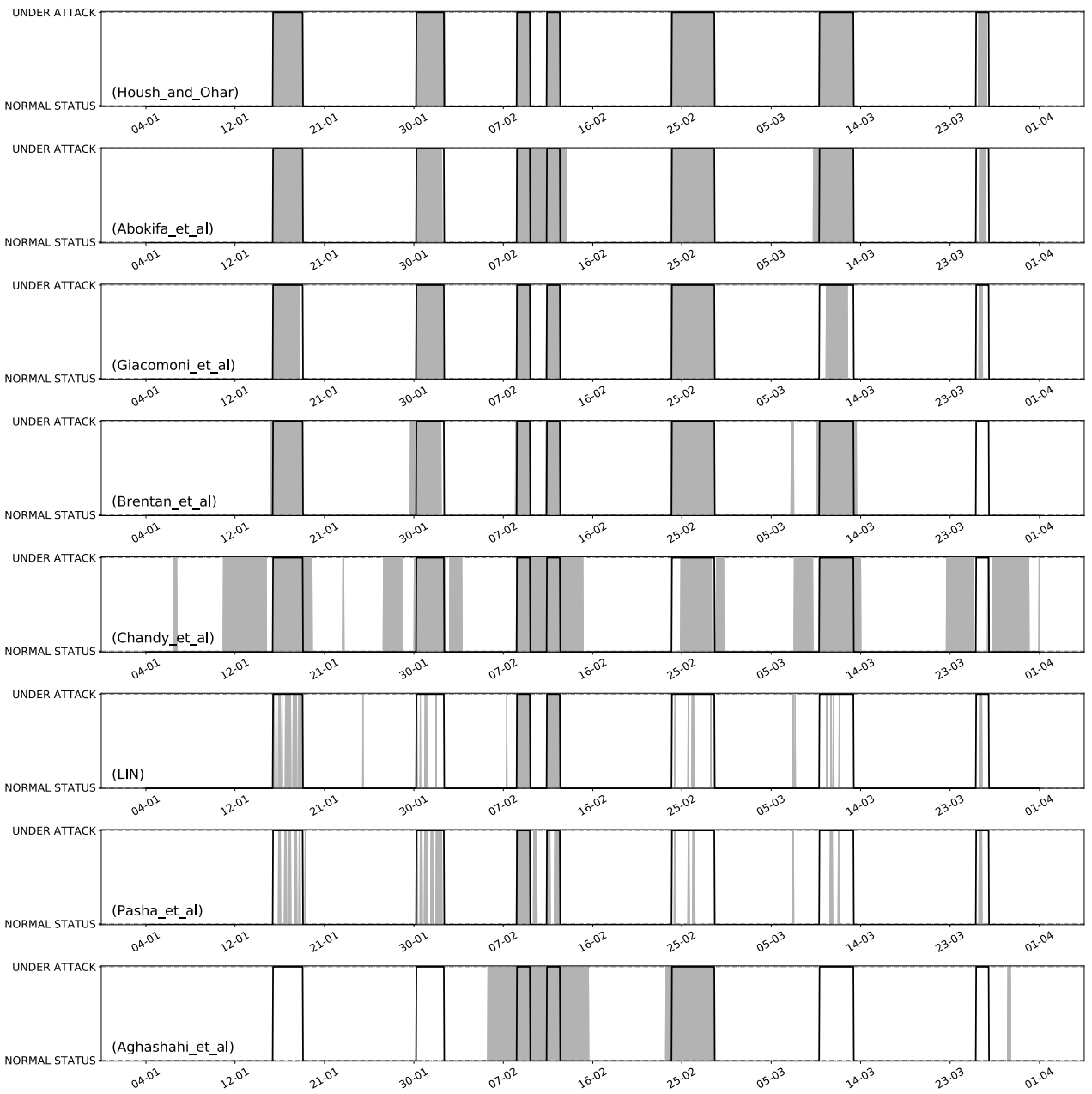
Assignment 2

- The BATADAL dataset:
  - https://batadal.net/index.html
- Several attack scenario's on SCADA / ICS systems
  - Supervisory control and data acquisition
  - Industrial control systems
- In this case, a water distribution network
- Attacks on SCADA are some of the most feared cyber attacks


V3
V4
T5
PLC7
PLC6
T4
PLC2
T1
V2
PLC5
PU9
PU8
PU11
PU10
T7
PLC9
PLC8
PU6
PU7
PLC3
T6
PU5
PU4
V1
T2
T3
PLC4
PLC1

SCADA time series




UNDER ATTACK
NORMAL STATUS
(Housh_and_Ohar)
(Abokifa_et_al)
(Giacomoni_et_al)
(Brentan_et_al)
(Chandy_et_al)
(LIN)
(Pasha_et_al)
(Aghashahi_et_al)
04-01
12-01
21-01
30-01
07-02
16-02
25-02
05-03
14-03
23-03
01-04

Temporal and Sequential Data

id	value	amount	number	quantity	time
1	4	2000	21	30	12:30
2	4	2250	35	40	12:35
1	8	3000	42	50	12:55

- Data often contains temporal information
  - Should be treated different from other features
  - Can be used to derive sequence information

- Special methods exist for such sequential data
  1. Model-free - kNN
  2. Model-based – Markov models

- Important distinction:
  1. Continuous vs. discrete signals

Analysis of Sequential Data

- Sequential structure arises in a huge range of applications
  - Repeated measurements of a temporal process – time series
  - Online decision making & control
  - Text, biological sequences
  - System and process logs
- Standard machine learning methods can be applied, but:
  - They do not exploit temporal patterns
  - Computation & storage scales poorly to realistic applications
- Many methods visualize the sequential structure
  - Useful for understanding and analysis

Overview

- Applying standard machine learning methods
  - Time slicing, sliding windows
  - Computing distances
- Modeling time-series
  - Auto-Regressive Moving Average (ARMA)
  - Filtering noise, Fourier transform
- Discrete models
  - Markov chains
  - N-grams
  - State machines
  - Hidden Markov models

Time slicing

- Simple idea:
- Group rows based on their time value
  - Are they early or late in the process?
- Learn different models for the different groups

Time slicing

id	value	amount	number	quantity	time
1	4	2000	21	30	12:30
2	4	2250	35	40	12:35
1	8	3000	42	50	12:55
3	10	3300	15	55	13:00
1	11	3400	36	60	13:15
2	6	3500	14	65	13:20
2	8	3800	50	70	13:35
3	10	4000	42	80	13:40
1	14	4100	27	85	13:55
3	12	4200	33	90	14:00

Time slicing – what to fill in?

id	value	amount	number	quantity	time	slice
1	4	2000	21	30	12:30	?
2	4	2250	35	40	12:35	?
1	8	3000	42	50	12:55	?
3	10	3300	15	55	13:00	?
1	11	3400	36	60	13:15	?
2	6	3500	14	65	13:20	?
2	8	3800	50	70	13:35	?
3	10	4000	42	80	13:40	?
1	14	4100	27	85	13:55	?
3	12	4200	33	90	14:00	?

Time slicing - overall

combine first/last occurrences of rows

id	value	amount	number	quantity	time	slice
1	4	2000	21	30	12:30	1
2	4	2250	35	40	12:35	1
1	8	3000	42	50	12:55	1
3	10	3300	15	55	13:00	1
1	11	3400	36	60	13:15	1
2	6	3500	14	65	13:20	2
2	8	3800	50	70	13:35	2
3	10	4000	42	80	13:40	2
1	14	4100	27	85	13:55	2
3	12	4200	33	90	14:00	2

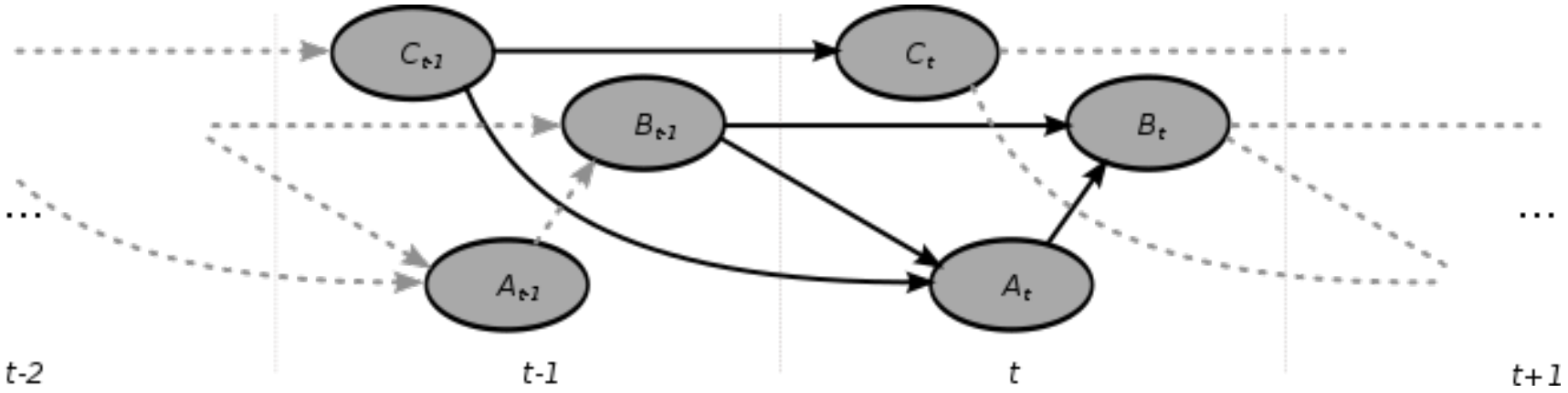
Time slicing – process dependent

combine first/last occurrences of different process ids

id	value	amount	number	quantity	time	slice
1	4	2000	21	30	12:30	1
2	4	2250	35	40	12:35	1
1	8	3000	42	50	12:55	1
3	10	3300	15	55	13:00	1
1	11	3400	36	60	13:15	2
2	6	3500	14	65	13:20	1
2	8	3800	50	70	13:35	2
3	10	4000	42	80	13:40	1
1	14	4100	27	85	13:55	2
3	12	4200	33	90	14:00	2

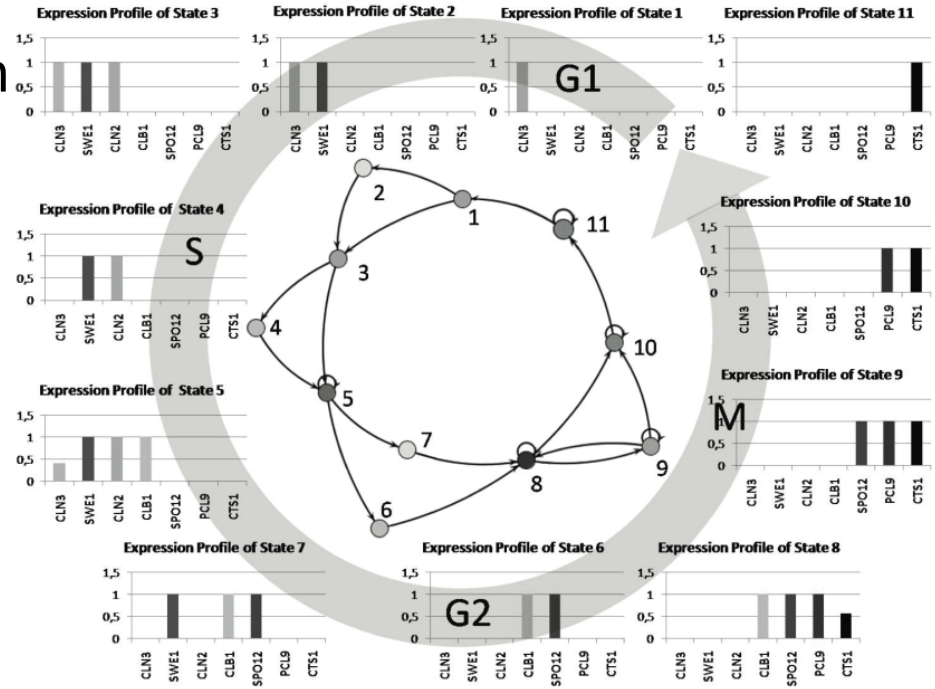
Once sliced

- Learn a model for every slice
- Model and learn the dependencies between the models, eg, using a Dynamic Bayesian Network:



Time slicing

- Often applied in patient modeling (e-health)
- How to slice depends entirely on the application
- Smarter methods exists if you can determine the "state" of a system:
  - Slice value equal to state number
  - State is discovered using **sequential models**



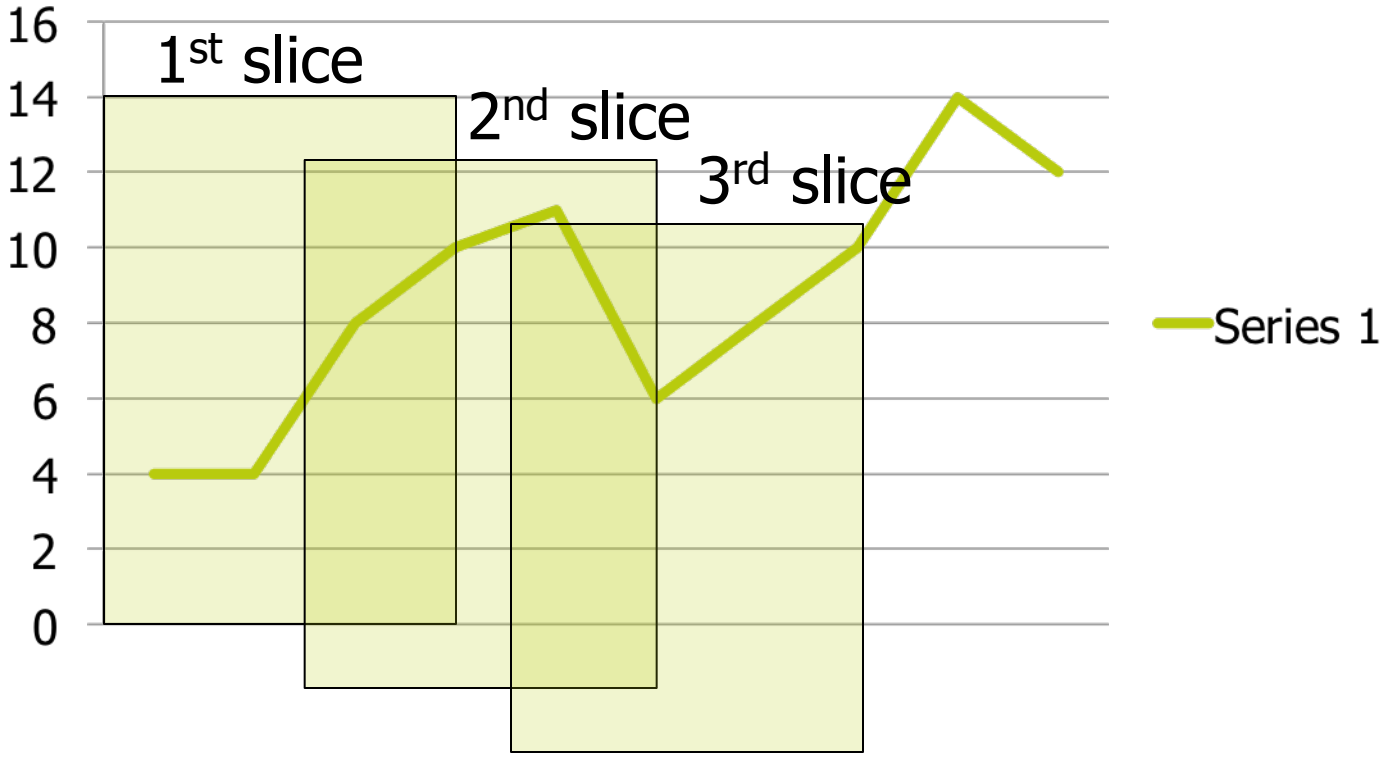
yeast cell cycle – Schmidt et al.

Sliding windows

- Slice the data dependent on the current time point:

Sliding windows

- Slice the data dependent on the current time point:



Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

v1	a1	v2	a2	v3	a3
?	?	?	?	?	?

Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

v1	a1	v2	a2	v3	a3
4	2000	4	2250	8	3000

Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

v1	a1	v2	a2	v3	a3
4	2000	4	2250	8	3000
?	?	?	?	?	?

Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

v1	a1	v2	a2	v3	a3
4	2000	4	2250	8	3000
4	2250	8	3000	10	3300

Sliding windows

value	amount
4	2000
4	2250
8	3000
10	3300
11	3400
6	3500
8	3800
10	4000
14	4100
12	4200

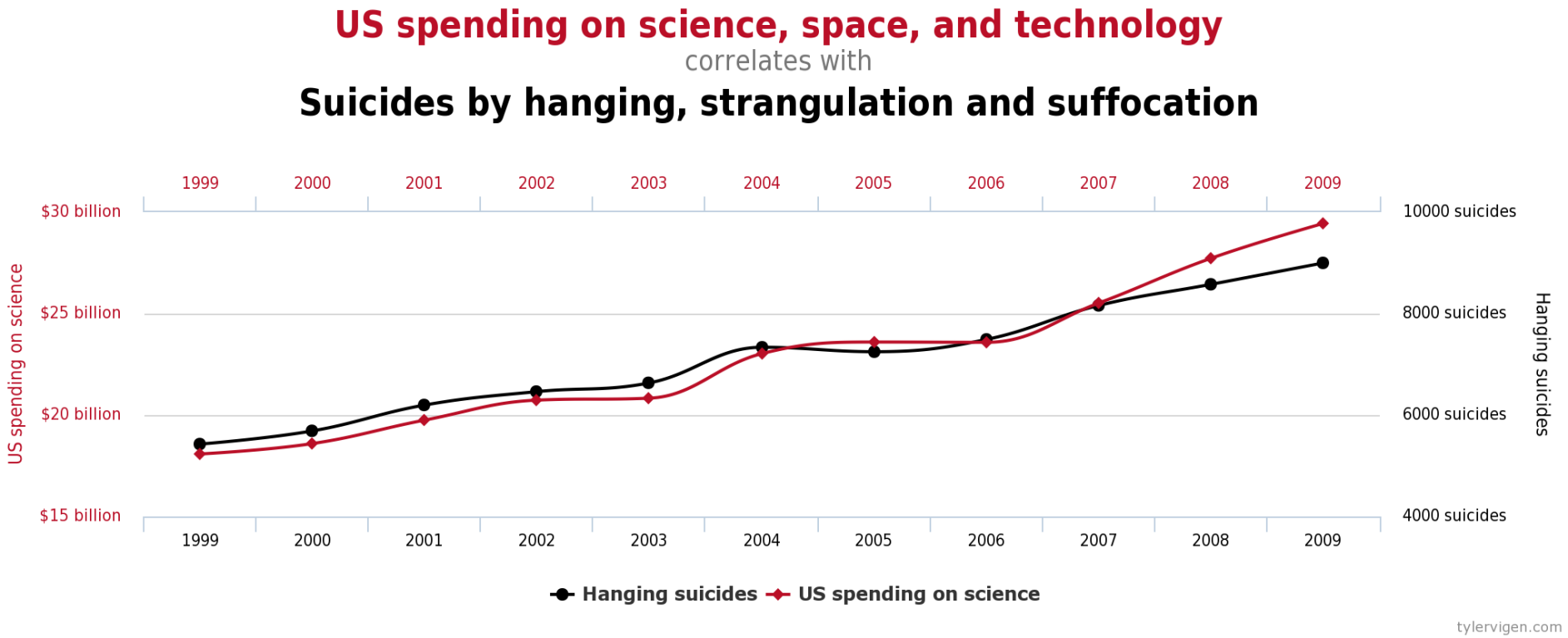
v1	a1	v2	a2	v3	a3
4	2000	4	2250	8	3000
4	2250	8	3000	10	3300
8	3000	10	3300	11	3400
10	3300	11	3400	6	3500
11	3400	6	3500	8	3800
6	3500	8	3800	10	4000
8	3800	10	4000	14	4100
10	4000	14	4100	12	4200
14	4100	12	4200		
12	4200				

Sliding windows

- Usually applied to time-series data:
  - Measurements at a fixed temporal rate

    i.e., financial data, audio signals
- Any machine learning method can be used with:
  - Standard distance measures, e.g., Euclidian
  - Special temporal distances, e.g., Dynamic Time Warping
  - Symbolic approximations
- Only takes a fixed-length history into account
- Hard to apply when timings are not fixed

Shape often matters most



Are these series similar? Their y-ranges are very different…
check out: http://www.tylervigen.com/spurious-correlations

Scaling time series

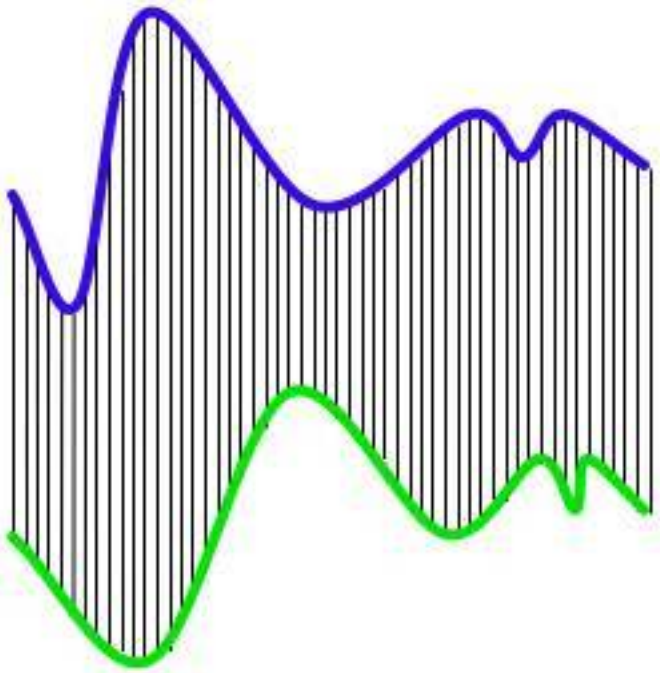
- In order to compare shapes, we need to scale the series:

$$c_i' = \frac{c_i - \mu(C)}{\sigma(C)}$$

- for every data point c_i, where μ is the mean and σ is the standard deviation of the series

- After normalization, we can compute distances between sliding windows

Euclidean distance



(a)

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Euclidean distance

	v1	v2	v3	v4	v5	v6	v7	v8
s1	10	15	20	30	0	25	10	20
s2	20	30	0	30	25	15	20	10
$(s1-s2)^2$	100	225	400	0	625	100	100	100

Distance = sqrt(100+225+400+0+625+100+100+100) = 40.62

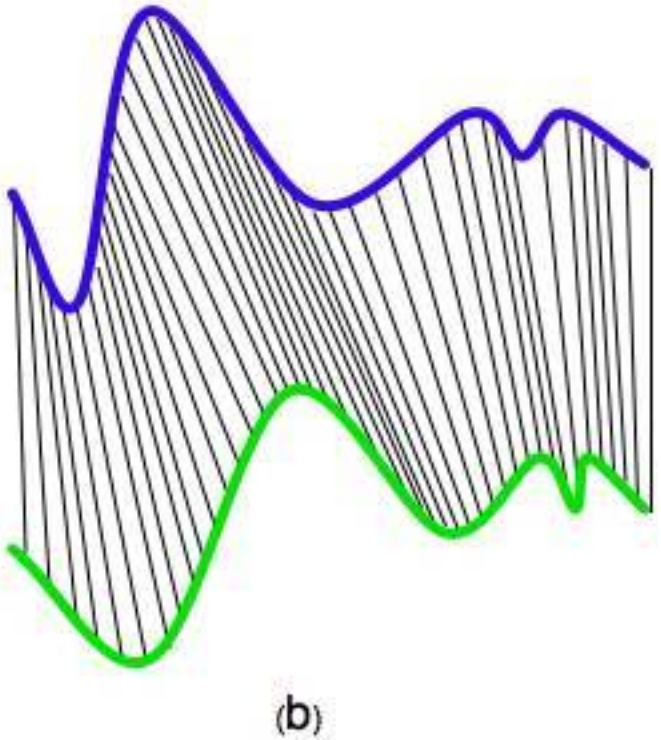
Euclidean distance

	v1	v2	v3	v4	v5	v6	v7	v8
s1	10	15	20	30	0	25	10	20
s2	20	30	0	30	25	15	20	10
$(s1-s2)^2$	100	225	400	0	625	100	100	100

Distance = sqrt(100+225+400+0+625+100+100+100) = 40.62

Signals seem out of phase…

Dynamic Time Warping or sequence alignment



Aligns time-series non-linearly in time, finds the best match

Time-warping distance

	v1	v2	v3	v4	v5	v6	v7	v8
s1	10	15	20	30	0	25	10	20
s2	20	30	0	30	25	15	20	10

Paired squared distances:

s1/s2	**v1**	**v2**	**v3**	**v4**	**v5**	**v6**	**v7**	**v8**
v1	100							
v2		225						
v3			400					
v4				0				
v5					625			
v6						100		
v7							100	
v8								100

Time-warping distance

	v1	v2	v3	v4	v5	v6	v7	v8
s1	10	15	20	30	0	25	10	20
s2	20	30	0	30	25	15	20	10

Paired squared distances:

s1/s2	**v1**	**v2**	**v3**	**v4**	**v5**	**v6**	**v7**	**v8**
v1	100	400	100	400	225	25	100	0
v2	25	225	225	225	100	0	25	25
v3	0	100	400	100	25	25	0	100
v4	100	0	900	0	25	225	100	400
v5	400	900	0	900	625	225	400	100
v6	25	25	625	25	0	100	25	225
v7	100	400	100	400	225	25	100	0
v8	0	100	400	100	25	25	0	100

Time-warping distance

Find a minimum cost path from left top to bottom right

Paired squared distances:

s1/s2	v1	v2	v3	v4	v5	v6	v7	v8
v1	**100**	400	100	400	225	25	100	0
v2	25	225	225	225	100	0	25	25
v3	0	100	400	100	25	25	0	100
v4	100	0	900	0	25	225	100	400
v5	400	900	0	900	625	225	400	100
v6	25	25	625	25	0	100	25	225
v7	100	400	100	400	225	25	100	0
v8	0	100	400	100	25	25	0	**100**

Time-warping distance

F

For instance…

Paired squared distances:

s1/s2	v1	v2	v3	v4	v5	v6	v7	v8
v1	**100**	400	100	400	225	25	100	0
v2	**25**	225	225	225	100	0	25	25
v3	**0**	100	400	100	25	25	0	100
v4	100	**0**	900	0	25	225	100	400
v5	400	900	**0**	900	625	225	400	100
v6	25	25	625	**25**	**0**	100	25	225
v7	100	400	100	400	225	**25**	100	0
v8	0	100	400	100	25	25	**0**	**100**

Time-warping distance

Distance = sqrt(100+25+0+0+0+25+0+25+100) = 16.58

s1/s2	v1	v2	v3	v4	v5	v6	v7	v8
v1	**100**	400	100	400	225	25	100	0
v2	**25**	225	225	225	100	0	25	25
v3	**0**	100	400	100	25	25	0	100
v4	100	**0**	900	0	25	225	100	400
v5	400	900	**0**	900	625	225	400	100
v6	25	25	625	**25**	**0**	100	25	225
v7	100	400	100	400	225	**25**	100	0
v8	0	100	400	100	25	25	**0**	**100**

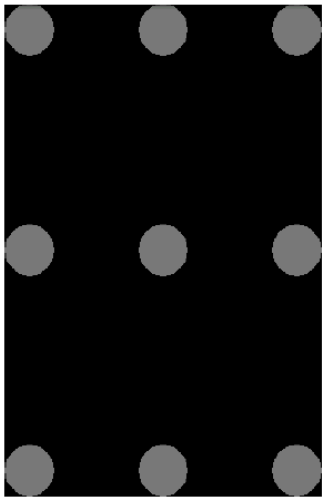
Time-warping distance

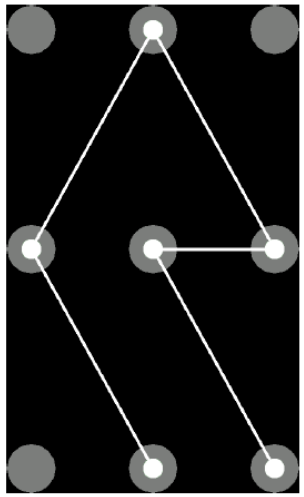
v/v	1/1	2/1	3/1	4/2	5/3	6/4	6/5	7/6	8/7	8/8
s1	10	15	20	30	0	25	25	10	20	10
s2	20	20	20	30	0	30	25	15	20	10

s1/s2	v1	v2	v3	v4	v5	v6	v7	v8
v1	**100**	400	100	400	225	25	100	0
v2	**25**	225	225	225	100	0	25	25
v3	**0**	100	400	100	25	25	0	100
v4	100	**0**	900	0	25	225	100	400
v5	400	900	**0**	900	625	225	400	100
v6	25	25	625	**25**	**0**	100	25	225
v7	100	400	100	400	225	**25**	100	0
v8	0	100	400	100	25	25	**0**	**100**

Time warping

- Frequently used for time-series classification, using kNN
- In cyber: profiling, e,g, smart-phone user id
  - Get time series for
    - X-Y coordinates
    - Pressure
    - Size
    - Time (?)





  - Apply DTW to match with the closest known profile (kNN)

Sequence alignment

- Discrete variant of time-warping
- Build matrix using (dynamic programming):

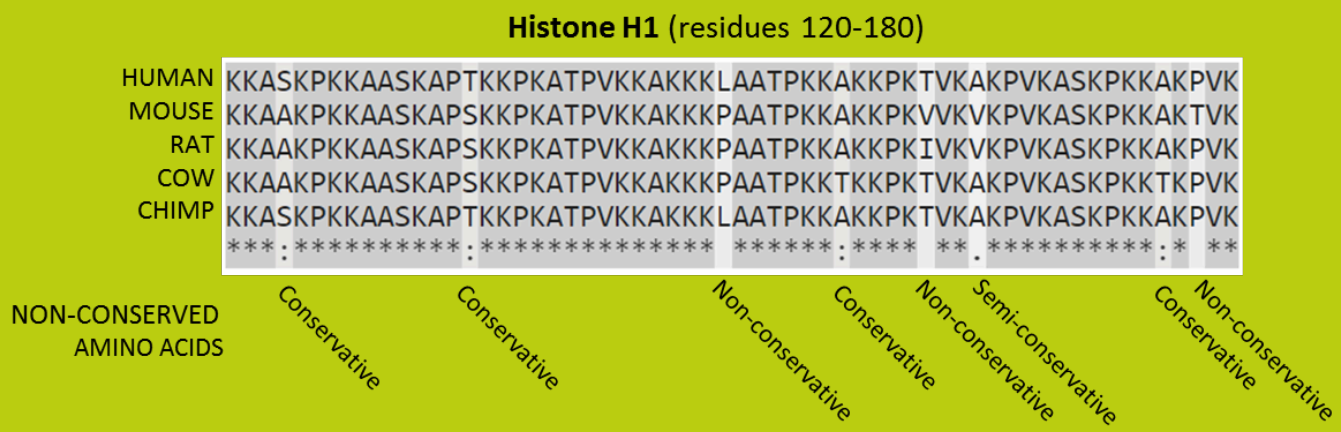
$$
\begin{aligned}
&H(i,0) = 0,\ 0 \le i \le m,\\
&H(0,j) = 0,\ 0 \le j \le n,\\
&\text{and}\\
&H(i,j) = \max \left\{ \begin{array}{c} 0 \\ H(i-1,j-1) + \ Sim(a_i, b_j) \\ \max_{k \ge 1}\{H(i-k,j) + \ W_k\} \\ \max_{l \ge 1}\{H(i,j-l) + \ W_l\} \end{array} \right\},\\
&1 \le i \le m, 1 \le j \le n
\end{aligned}
$$

- where W is the gap penalty

Sequence alignment

- Discrete variant of time-warping
- Build matrix using:

Very popular in bioinformatics:



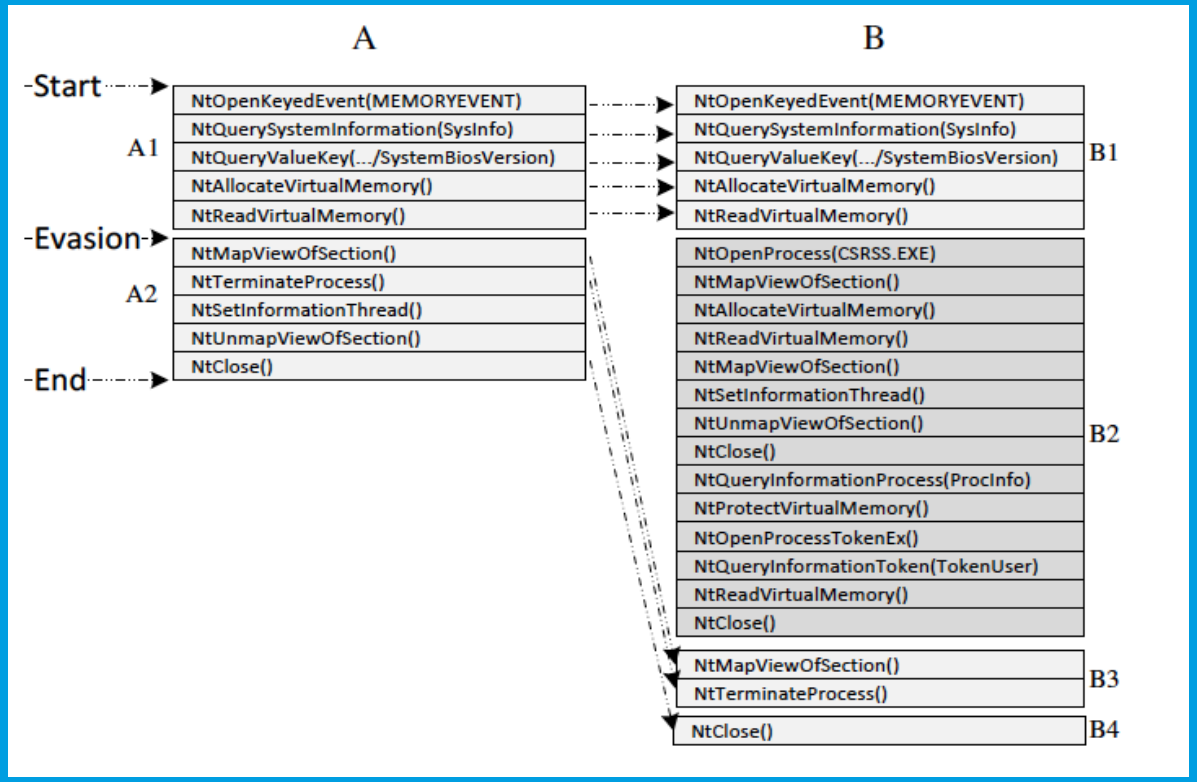
Sequence alignment

- Discre
- Build

HUMAN KK
MOUSE KK
RAT KK
COW KK
CHIMP KK
**

NON-CONSERVED AMINO ACIDS

Also used for system call traces:



So far

- Time slicing:
  - assign slice values, incrementing over time
  - learn different models for different slices
- Sliding windows:
  - obtain fixed length rows by moving a window over the series
  - learn a model from the obtained data table
- Distances with sequential orders:
  - first normalize if shape matters
  - Euclidean distance is OK for synchronized series
  - Time warping/sequence alignment handles out-of-sync series
    - Possible to include gaps to allow for noise

Overview

- Applying standard machine learning methods
  - Time slicing, sliding windows
  - Computing distances
- Modeling time-series
  - Auto-Regressive Moving Average (ARMA)
  - Filtering noise, Fourier transform
- Discrete sequential models
  - Markov chains
  - N-grams
  - State machines
  - Hidden Markov models

Time series prediction

	v1	v2	v3	v4	v5	v6	v7	v8
w1	10	15	20	30	0	25	10	20
w2	15	20	30	0	25	10	20	15
w3	20	30	0	25	10	20	15	10
w4	30	0	25	10	20	15	10	25

- Predict v8, based on v1, .., v7
- Can use any predictor such as Regression Trees, Probabilistic Models, Linear/Polynomial Regression
- Common baseline: persistence – predict the last value
- Moving Average is a special technique for time series that simply predicts the mean…

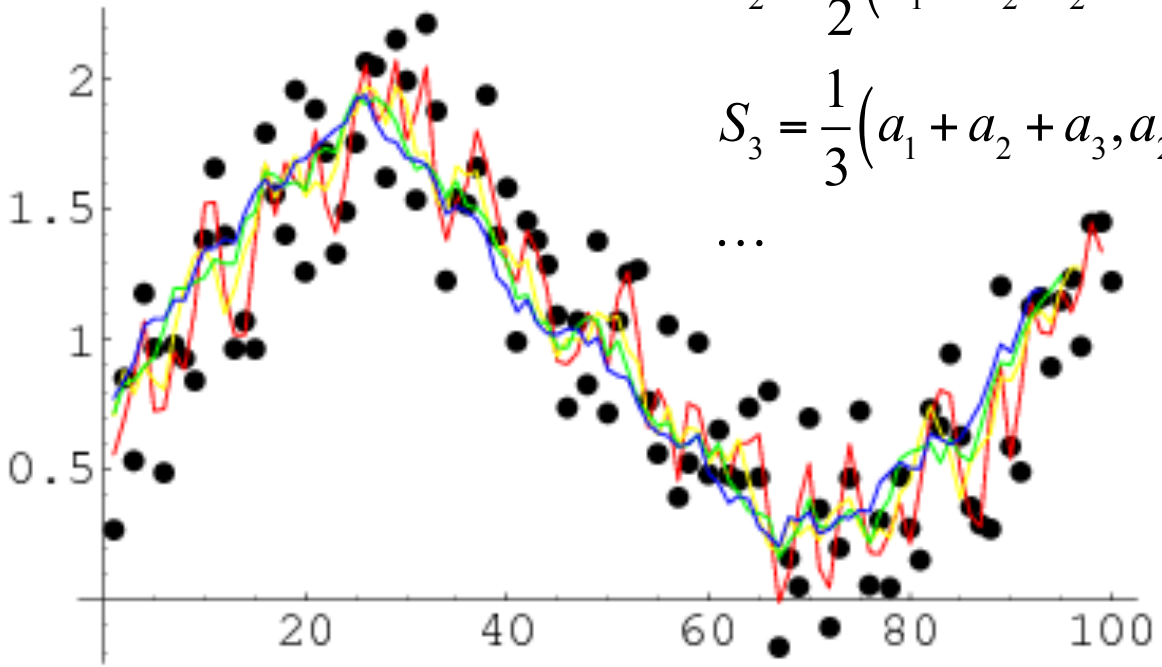
Moving average

Given a sequence, an *n*-moving average is a new sequence defined by taking the arithmetic means of subsequence of *n* terms:

$$S_2 = \frac{1}{2}\left(a_1 + a_2, a_2 + a_3, \cdots, a_{n-1} + a_n\right)$$

$$S_3 = \frac{1}{3}\left(a_1 + a_2 + a_3, a_2 + a_3 + a_4, \cdots, a_{n-2} + a_{n-1} + a_n\right)$$

$$\cdots$$



2- (red), 4- (yellow), 6- (green), and 8- (blue) moving average

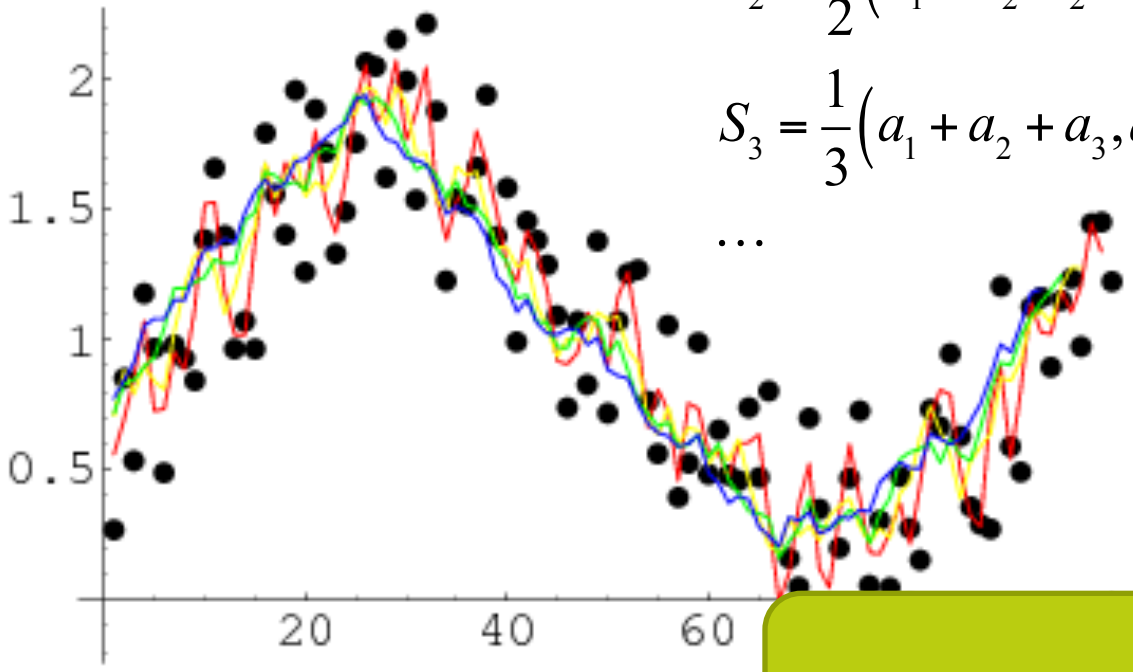
Moving average

Given a sequence, an *n*-moving average is a new sequence defined by taking the arithmetic means of subsequence of *n* terms:

$$S_2 = \frac{1}{2}\left(a_1 + a_2, a_2 + a_3, \cdots, a_{n-1} + a_n\right)$$

$$S_3 = \frac{1}{3}\left(a_1 + a_2 + a_3, a_2 + a_3 + a_4, \cdots, a_{n-2} + a_{n-1} + a_n\right)$$

$\cdots$



2- (red), 4- (yellow), 6- (green), and

Larger n means more smooth!

Time series prediction - MA

	v1	v2	v3	v4	v5	v6	v7	v8
w1	10	15	20	30	0	25	10	20
w2	15	20	30	0	25	10	20	15
w3	20	30	0	25	10	20	15	10
w4	30	0	25	10	20	15	10	25

	v1	v2	v3	v4	v5	v6	v7	pred
w1	10	15	20	30	0	25	10	15.7
w2	15	20	30	0	25	10	20	17.1
w3	20	30	0	25	10	20	15	17.1
w4	30	0	25	10	20	15	10	15.7

Time series prediction - MA

	v1	v2	v3	v4	v5	v6	v7	v8
w1	10	15	20	30	0	25	10	20
w2	15	20	30	0	25	10	20	15
w3	20	30	0	25	10	20	15	10
w4	30	0	25	10	20	15	10	25

	v1	v2	v3	v4	v5	v6	v7	pred
w1	10	15	20	30	0	25	10	15.7
						10	20	17.1
						20	15	17.1
						15	10	15.7

Predictions move along with the series, with some lag

Modeling noise

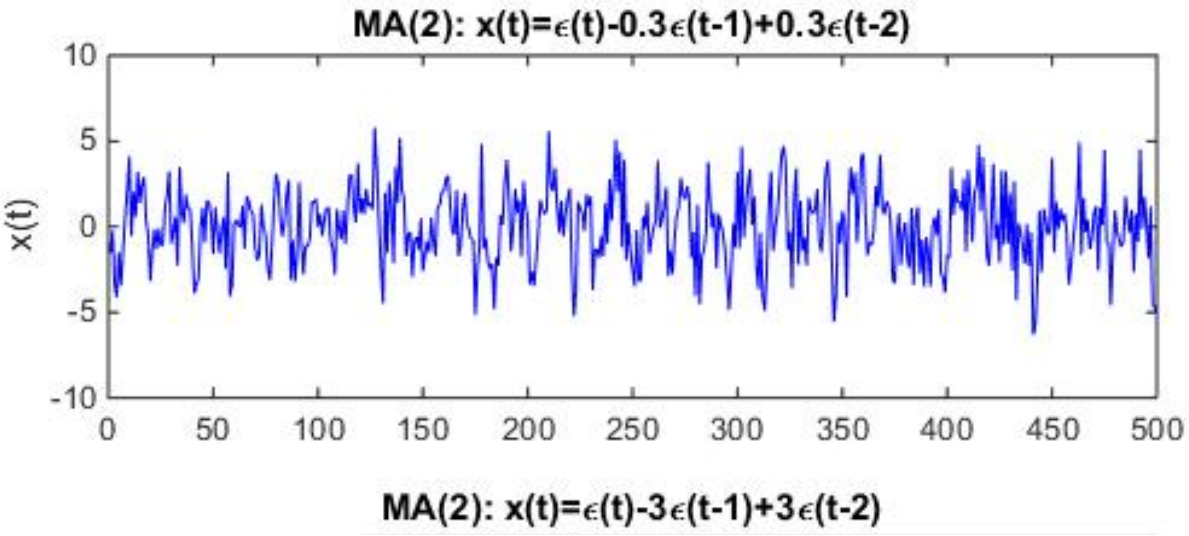
- The notation MA(q) refers to the moving average model of order q. The MA(q) model is written as:

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

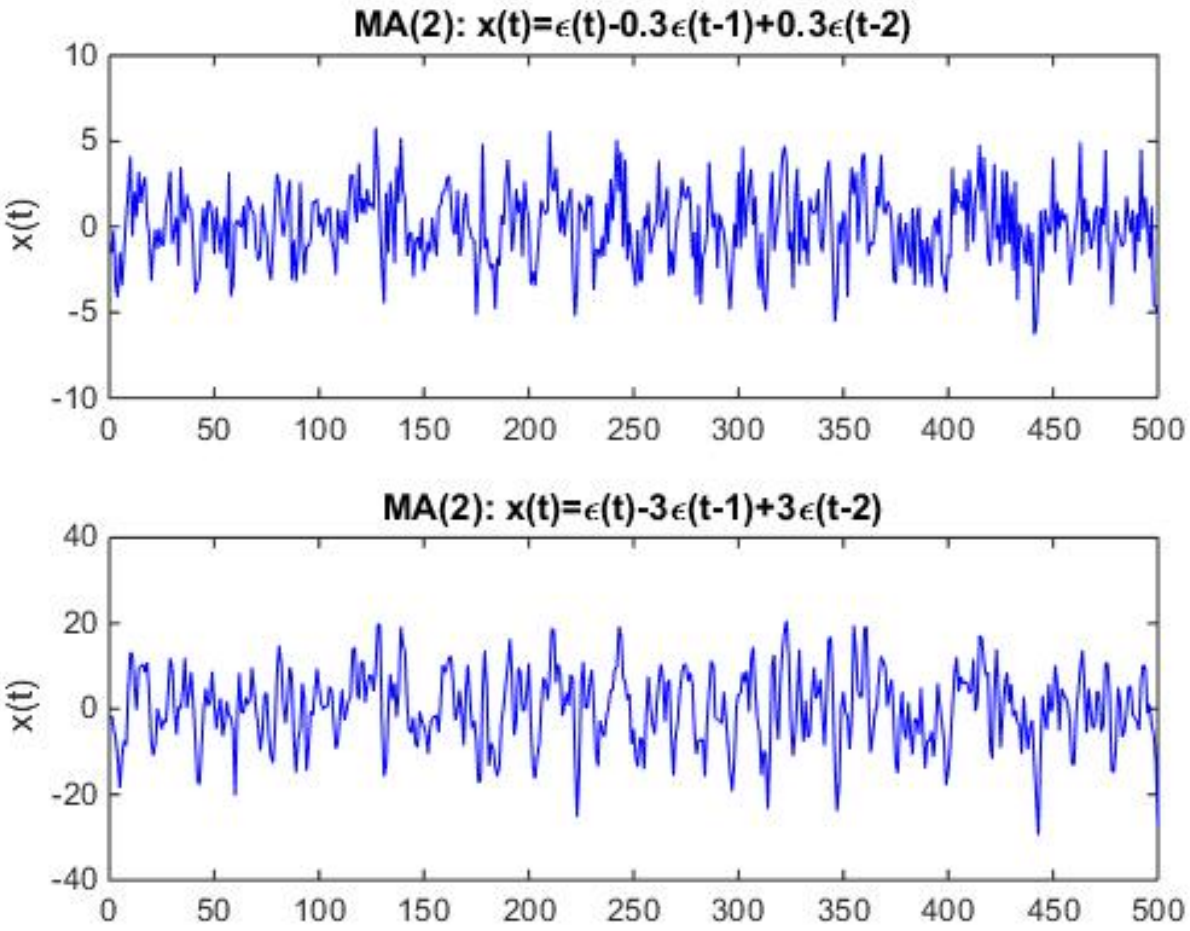
  - where X is a random variable, μ is the mean of the entire series, θ are coefficients, ε is white noise
- a moving-average model is a linear regression of the current value against previous noise error terms

A moving-average model is a linear regression to intuitively describe how the random variables ***fluctuate*** around mean of the series.

MA(q) for q=2 and different θ



MA(q) for q=2 and different θ



Auto regressive

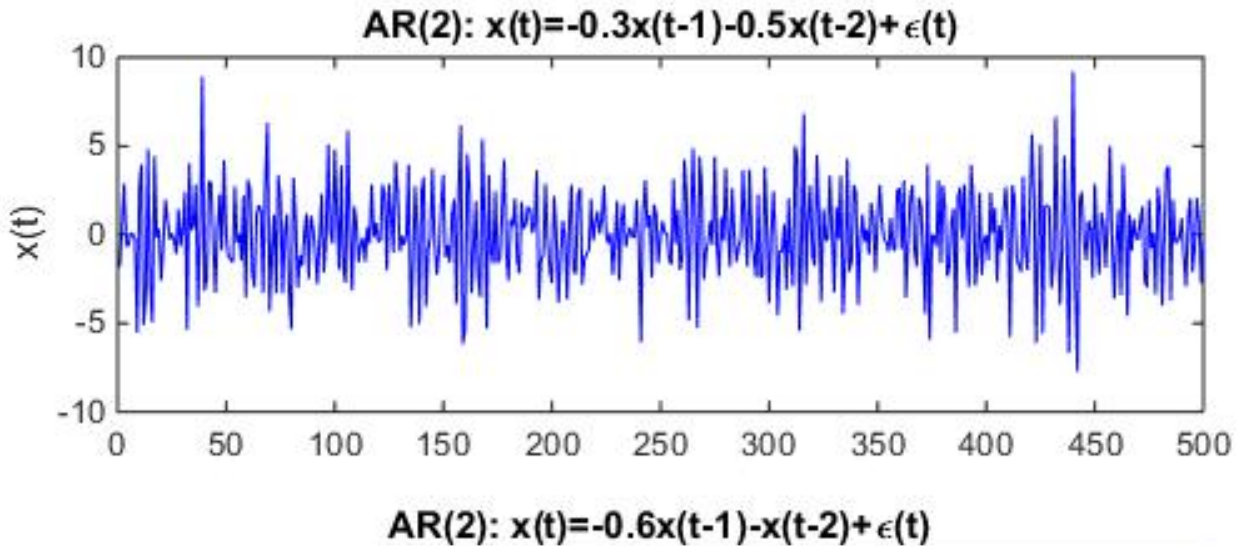
- The notation AR(p) refers to the autoregressive model of order p. The AR(p) model is written as:

$$X_t = \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

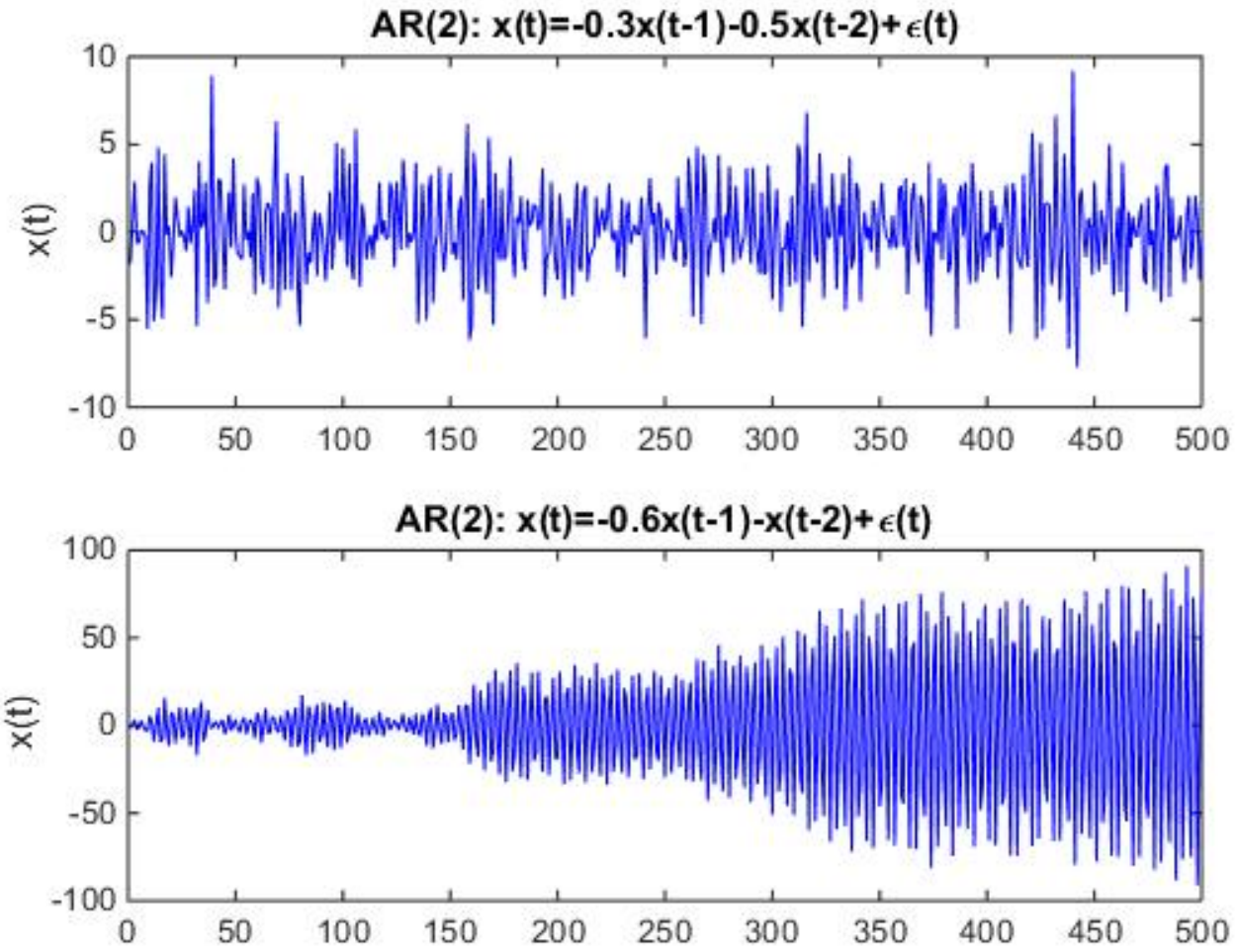
  - where X is a random variable, φ are coefficients, ε is white noise

- An auto-regressive model is a linear regression of the current value against previous values

AR(p) for p=2 and different ϕ

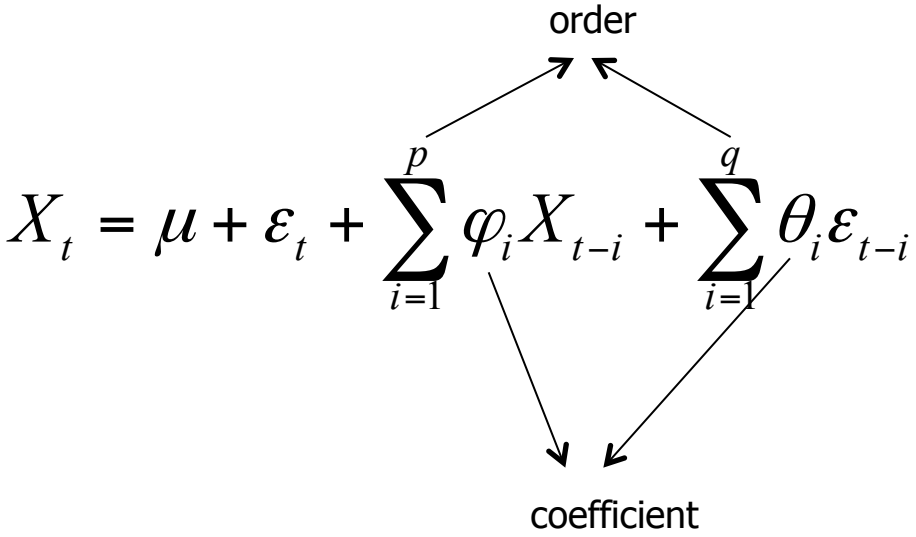


AR(p) for p=2 and different ϕ



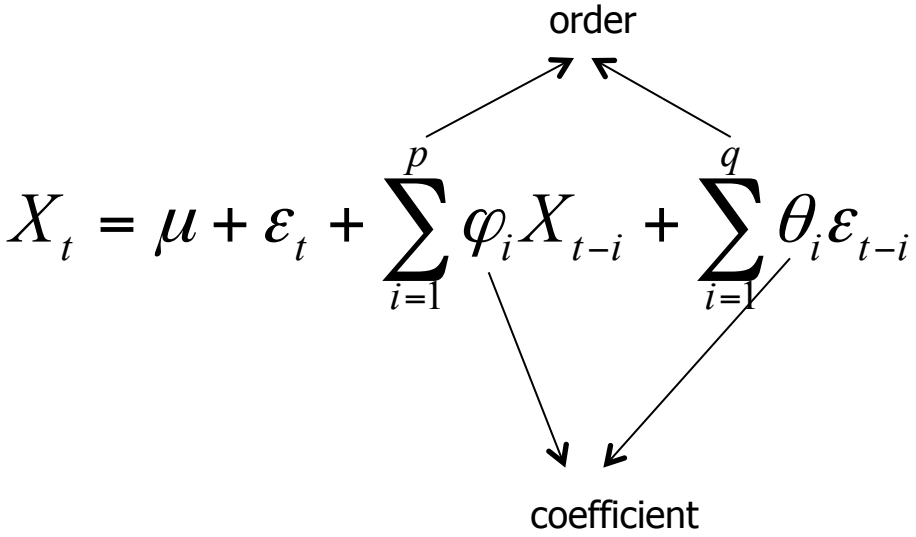
Auto regressive-moving average

- ARMA (p,q) is written as:



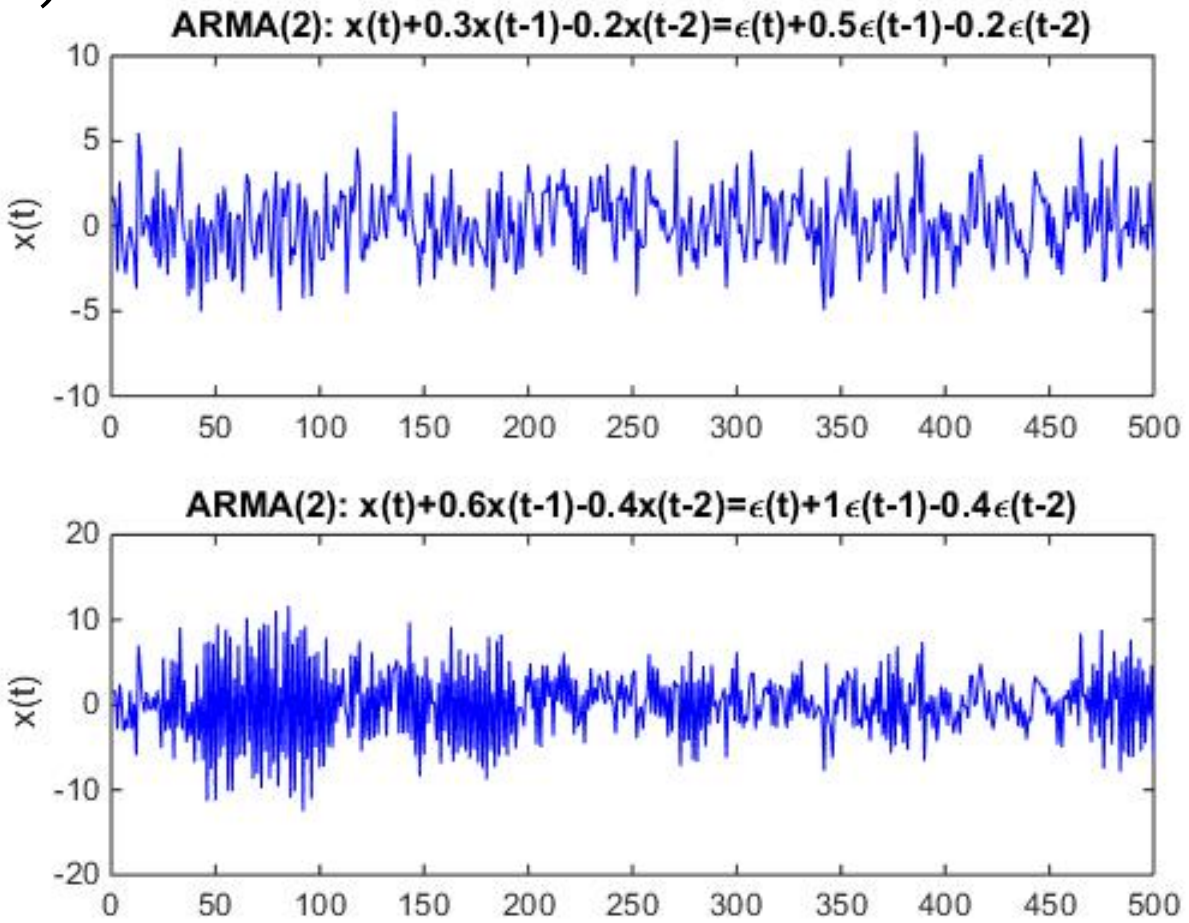
Auto regressive-moving average

- ARMA (p,q) is written as:

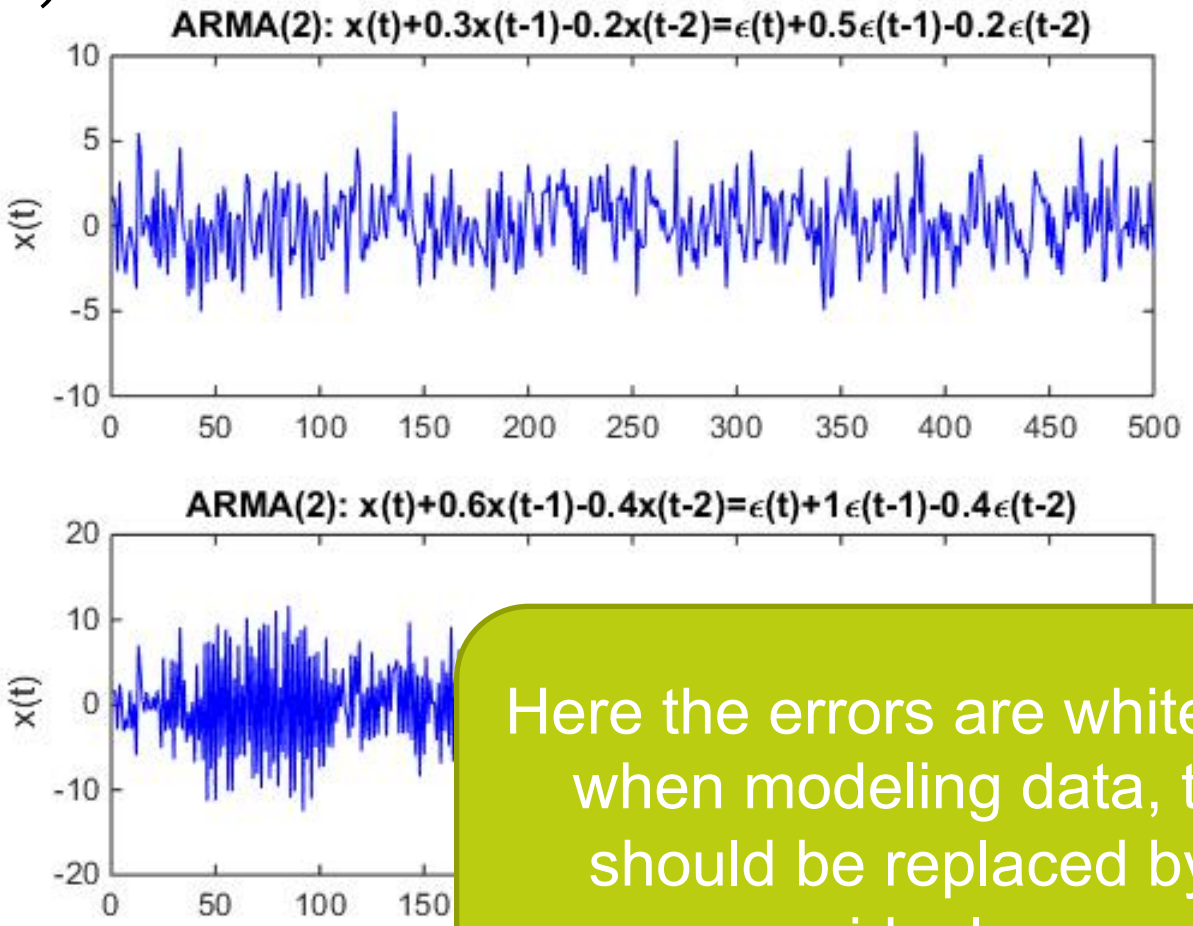


ARMA is one of the most frequently used models for time-series prediction

ARMA(p,q) for p,q=2 and different ϕ, θ

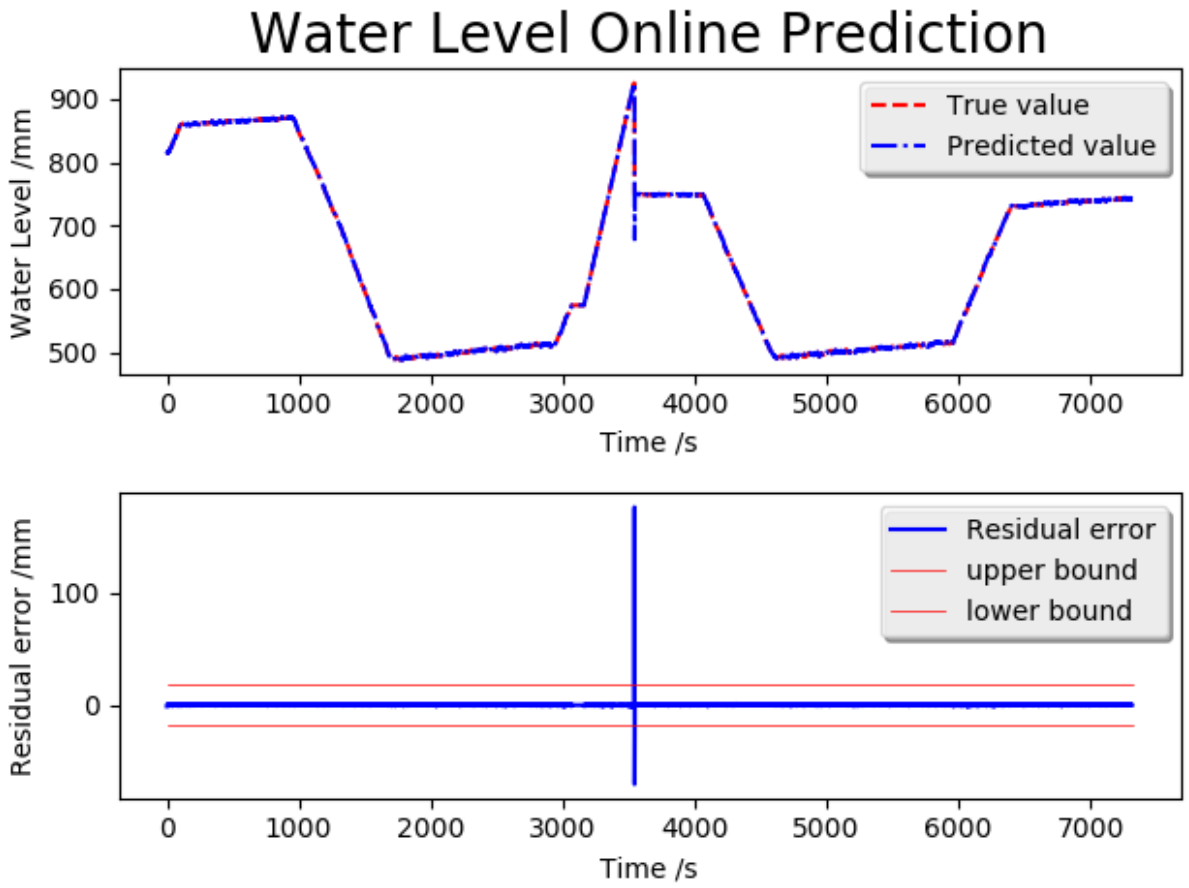


ARMA(p,q) for p,q=2 and different ϕ, θ



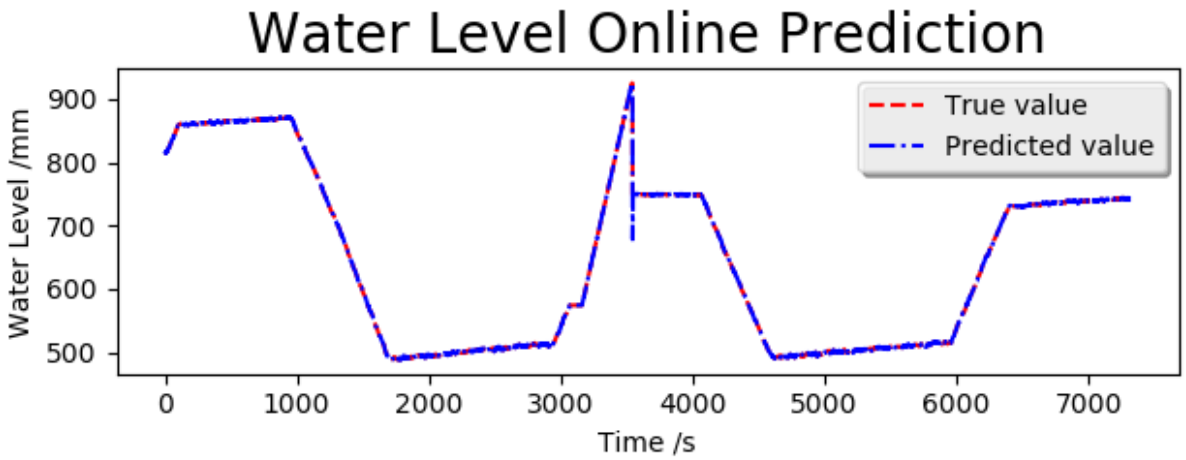
Here the errors are white-noise, when modeling data, these should be replaced by the residual errors

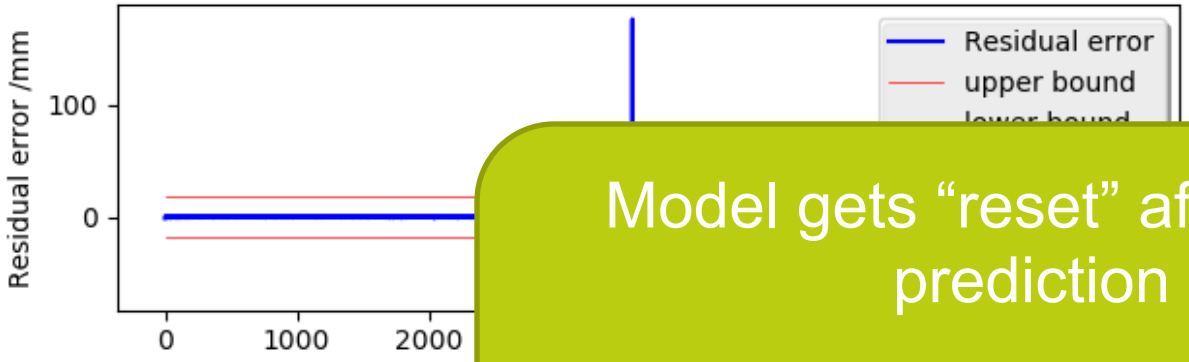
Predicting SCADA using AR



$$X_t = 1.4*X_{(t-1)} - 0.063*X_{(t-2)} - 0.338*X_{(t-3)} + e$$

Predicting SCADA using AR





X_t = 1.4*X_(t−1)

Model gets “reset” after every prediction

Removes the effect we get when generating new series

Estimating parameters for MA, AR, ARMA models

- AR is a special instance of linear regression, φ parameters can be estimated via ordinary least squares
- MA can be a non-linear form of regression, estimation of θ parameters requires some form of search, it can be hard to find the optimal values
- ARMA is hard since MA is hard

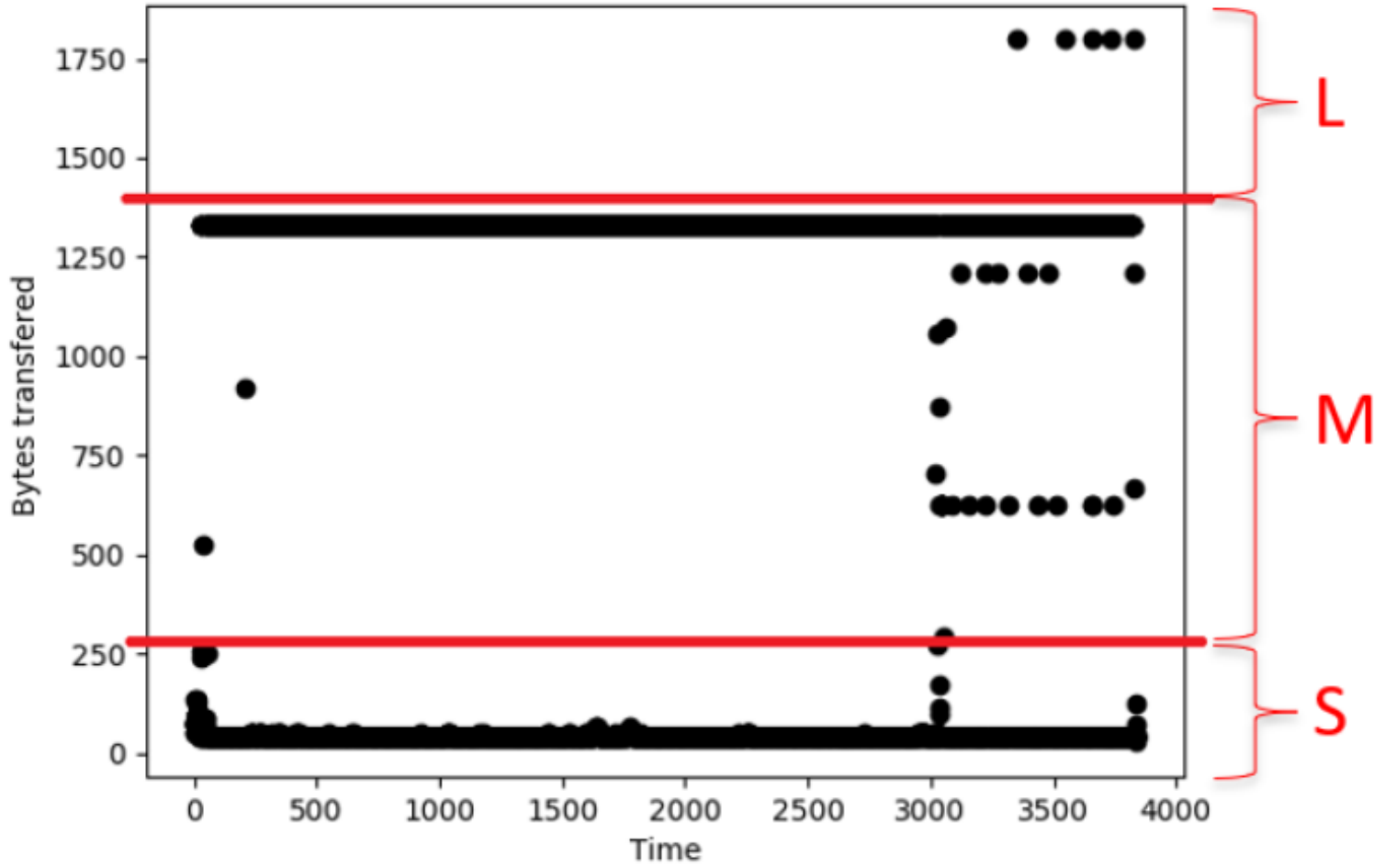
So far

- Time series prediction:
  - Predict t+1 using t-N⋯t
- Moving average:
  - Model free, simply average the past x values (or noise terms)
  - Possibly add exponential weights
  - In an *MA model*, the weights are learned from data (hard)
- Auto regressive:
  - Takes the previous values as input, model based
  - A type of linear regression
- ARMA is a very powerful combination of MA and AR

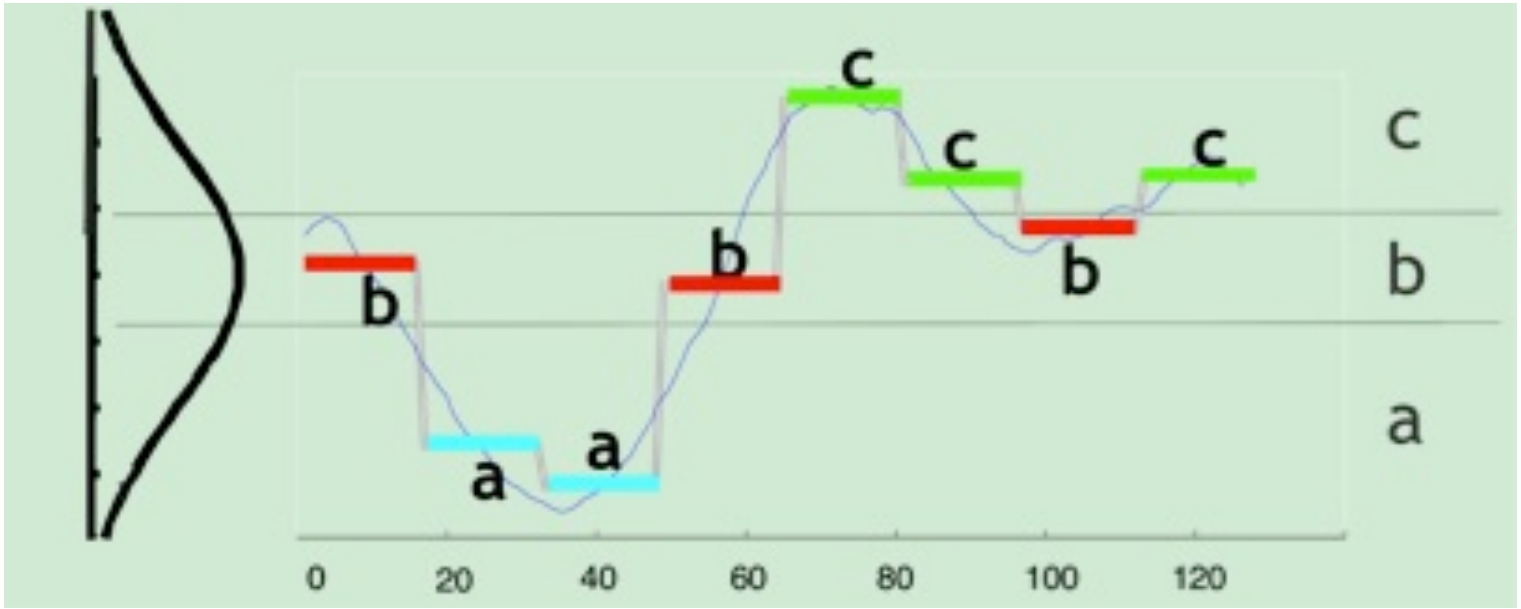
Overview

- Applying standard machine learning methods
  - Time slicing, sliding windows
  - Computing distances
- Modeling time-series
  - Auto-Regressive Moving Average (ARMA)
  - Filtering noise, Fourier transform
- Discrete sequential models
  - Markov chains
  - N-grams
  - State machines
  - Hidden Markov models

Making data points symbolic: percentiles

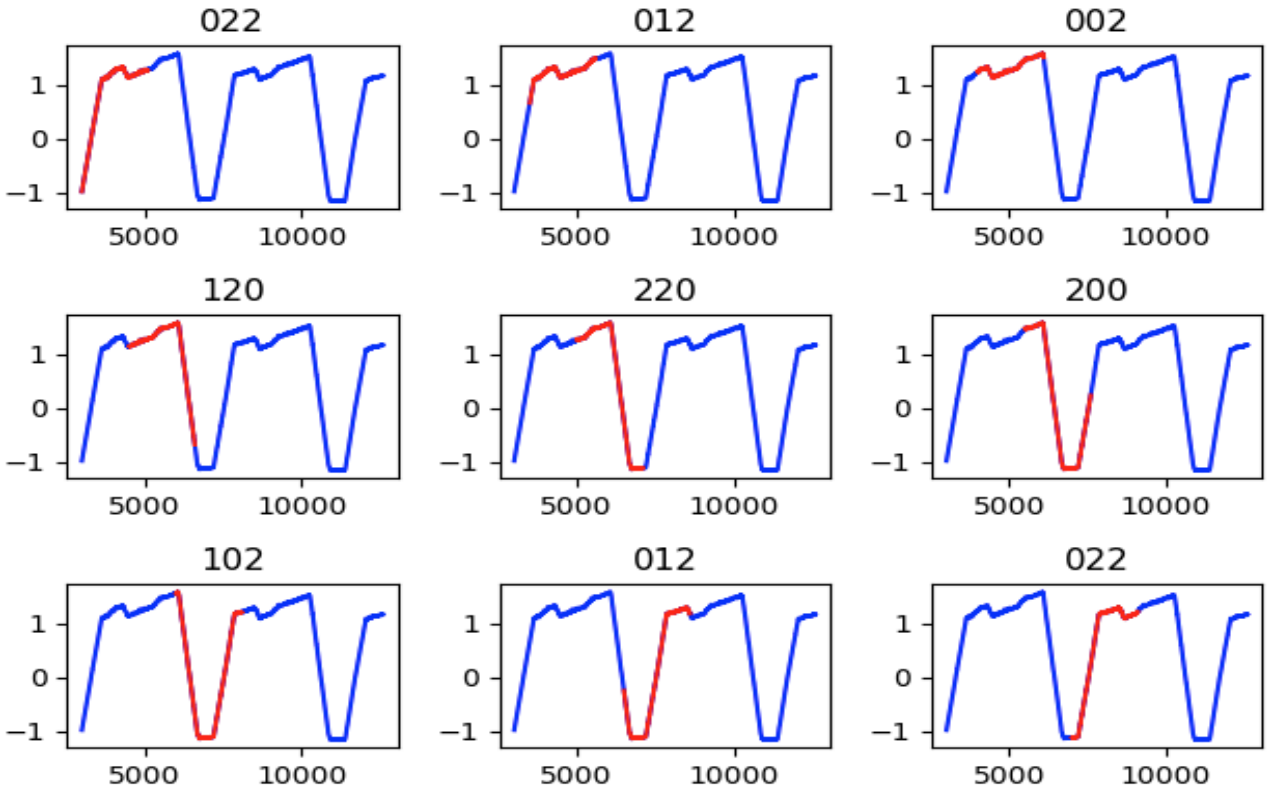


Symbolic approximation of time series



Approximate a value using (fixed) ranges to get symbolic sequences

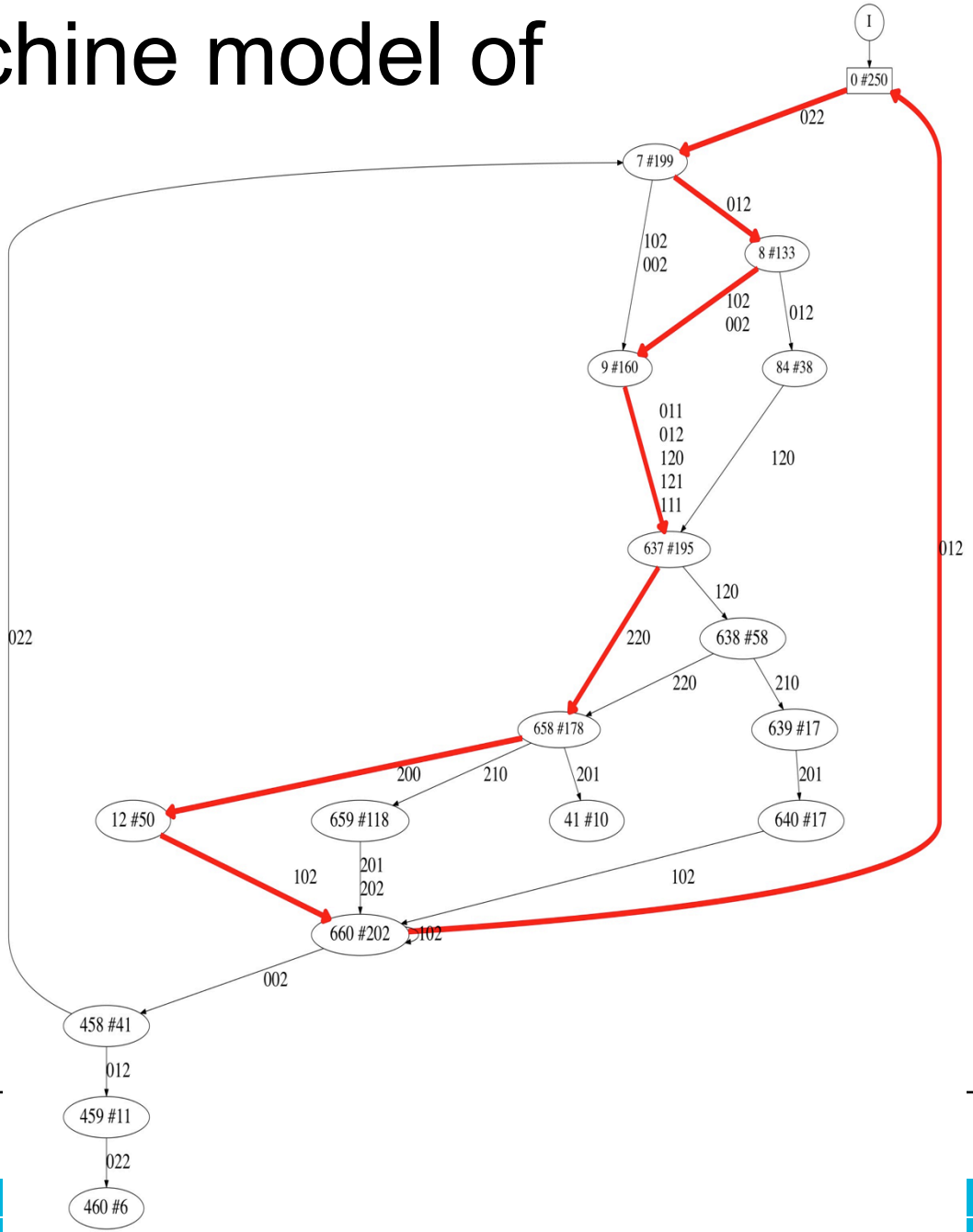
SAX (Symbolic Aggregate approXimation)



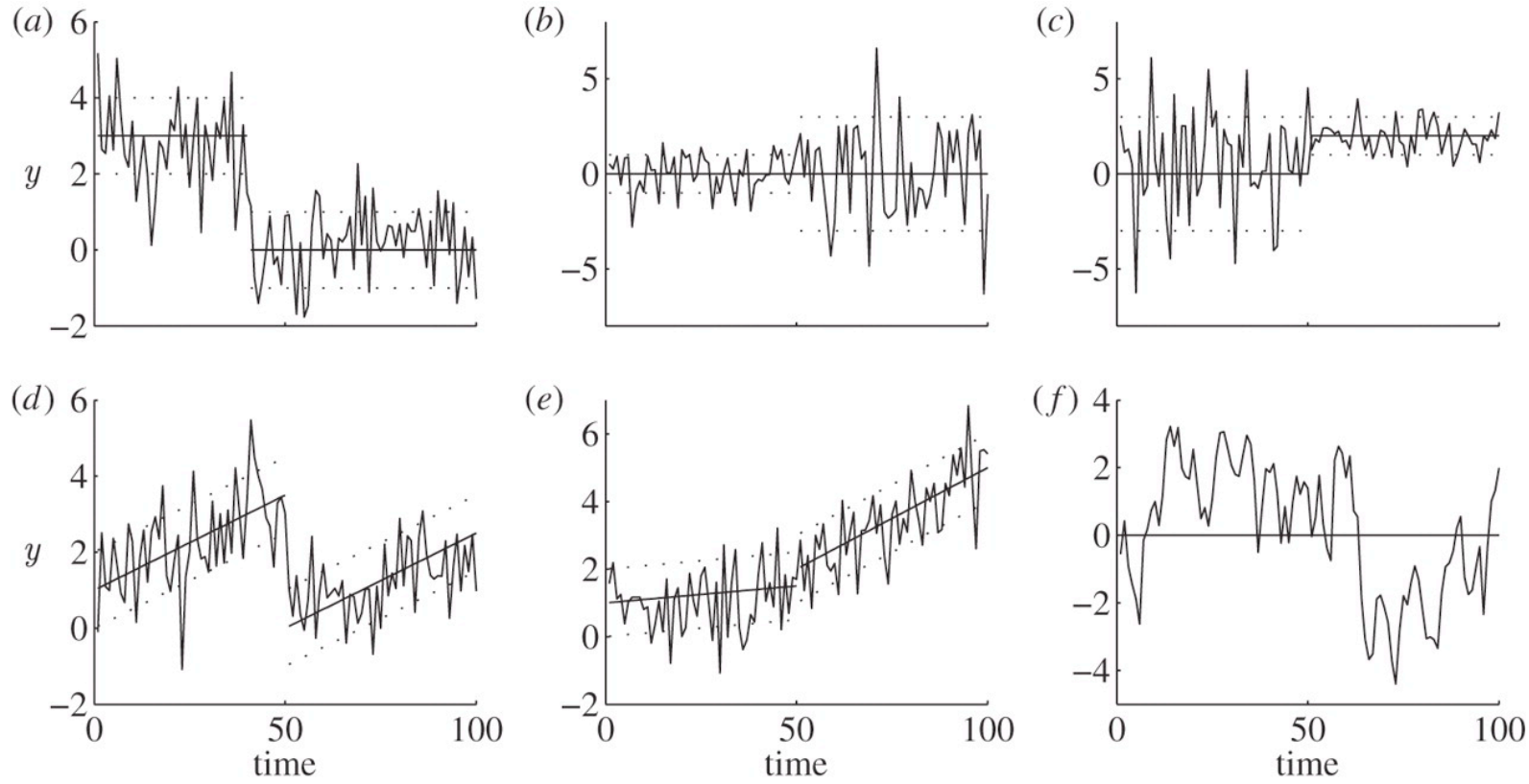
- Estimate mean values of same-size windows
- Normalize values in every window seperately
- 0 -> low, 1 -> middle, 2 -> high

Build state machine model of signal

- Model shows cyclic behavior in SCADA signals
- Can be used for anomaly detection

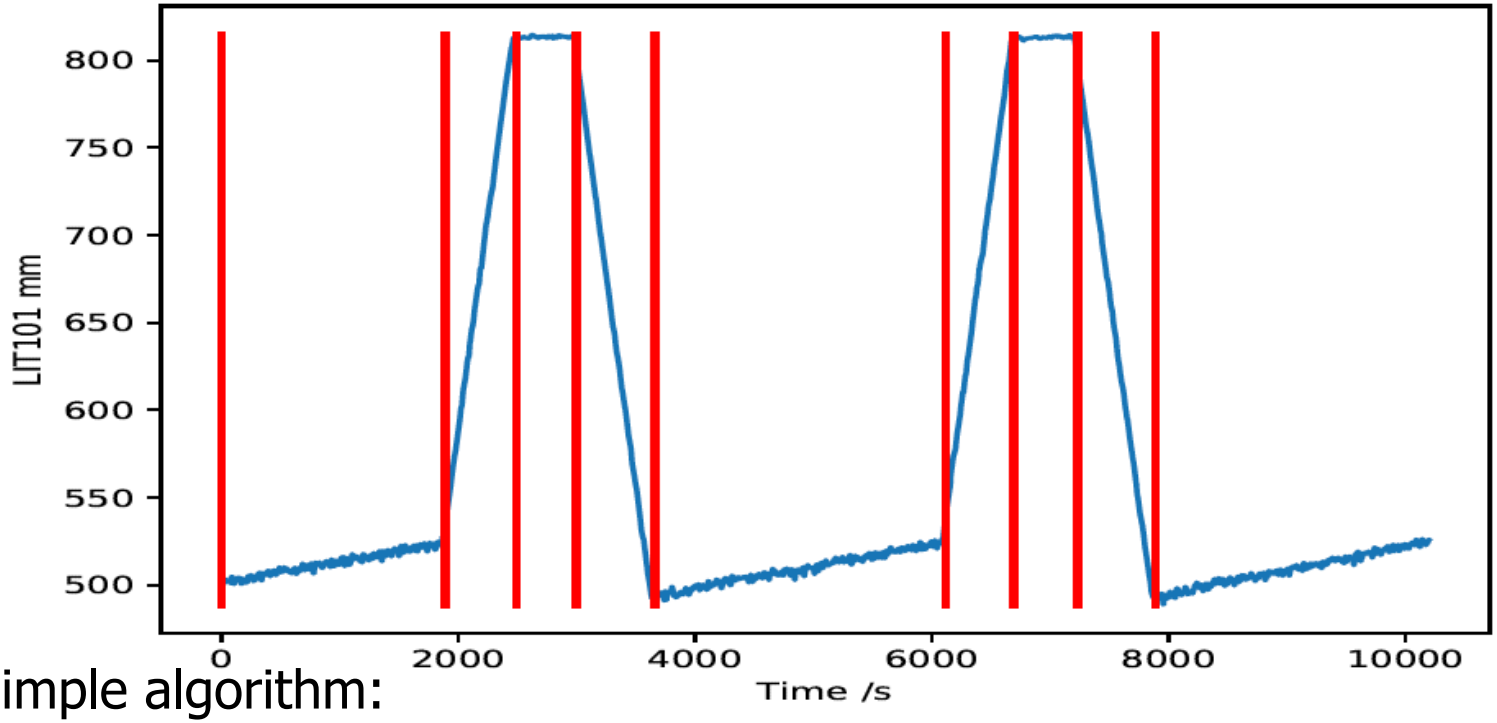


Change point detection and segmentation



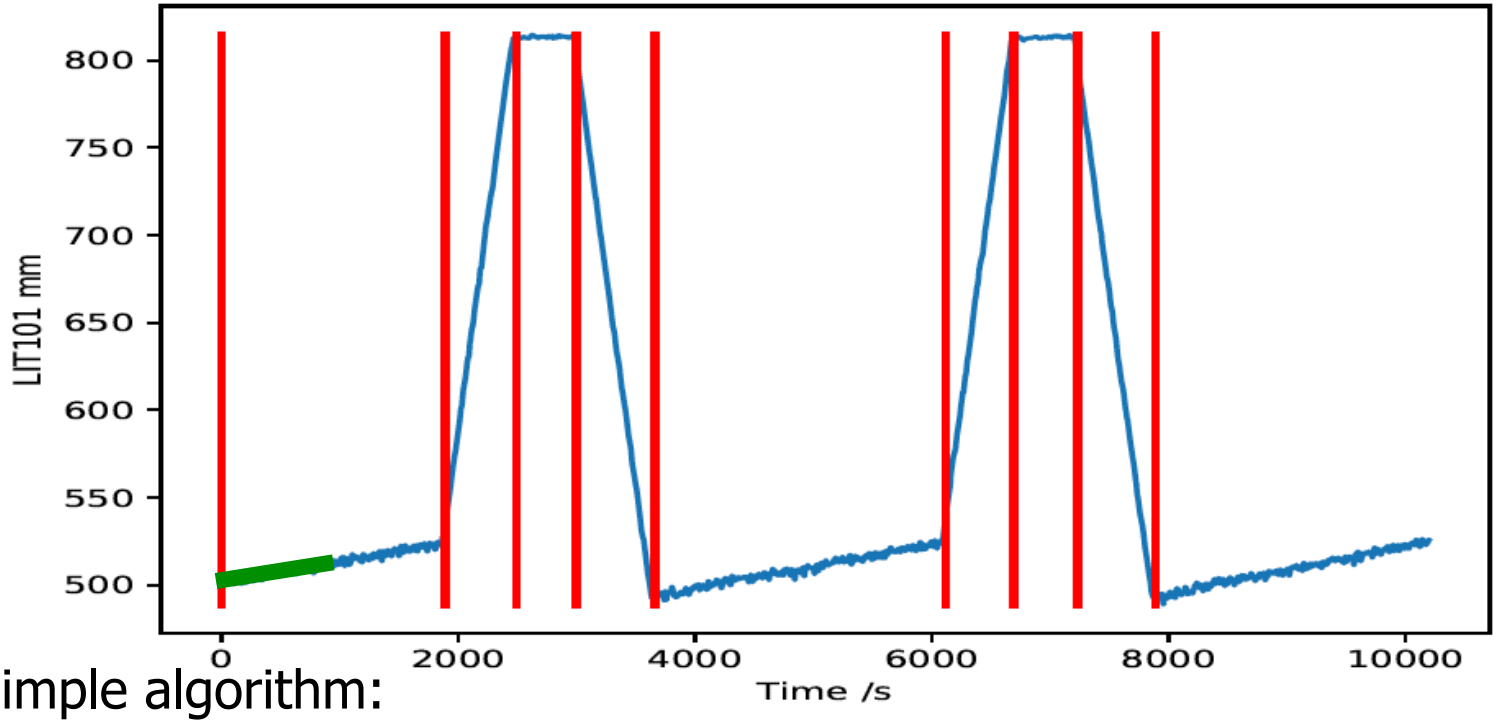
- Many methods exist for detection change points, such as abrupt trend or distribution changes

In SCADA cyclic behavior



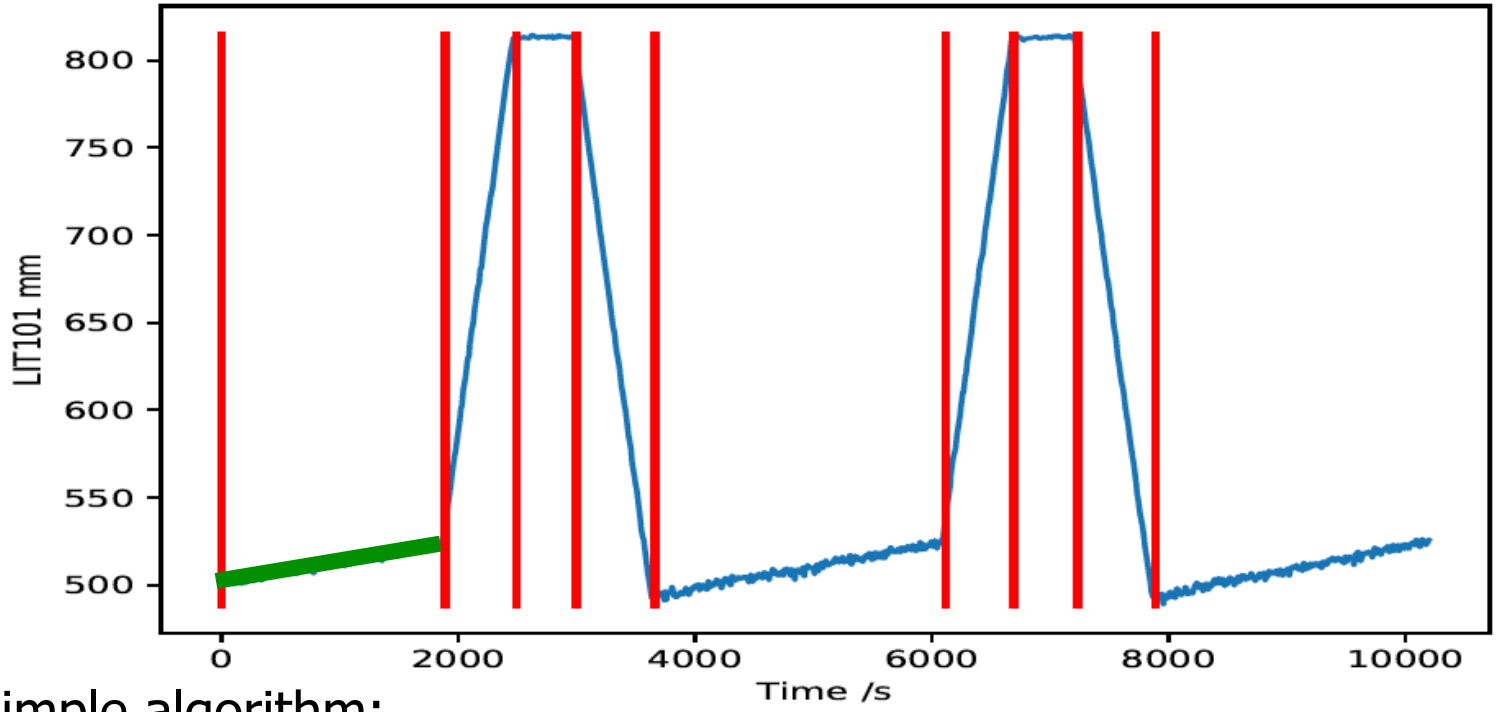
- Simple algorithm:
  1. Add next point to linear segment

In SCADA cyclic behavior



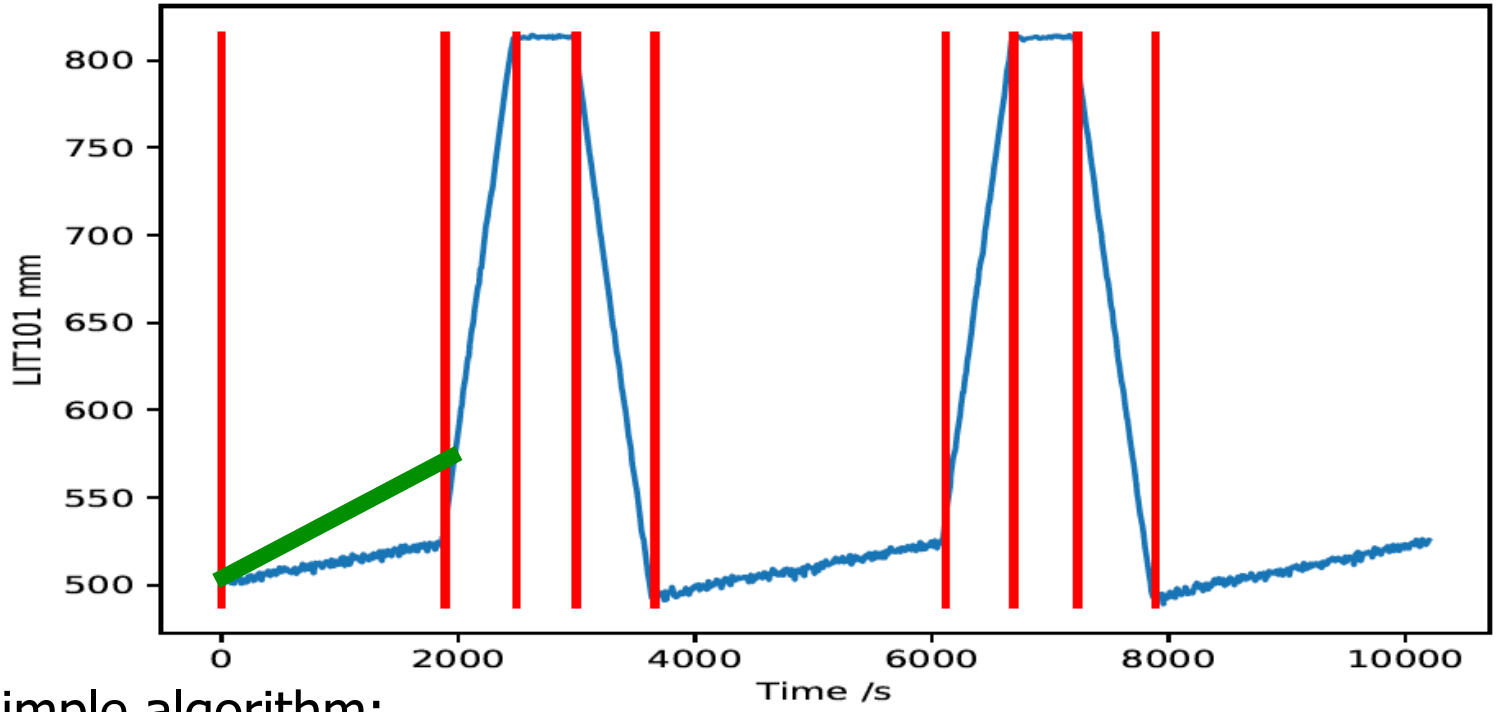
- Simple algorithm:
  1. Add next point to linear segment
  2. Compute regression error e

In SCADA cyclic behavior



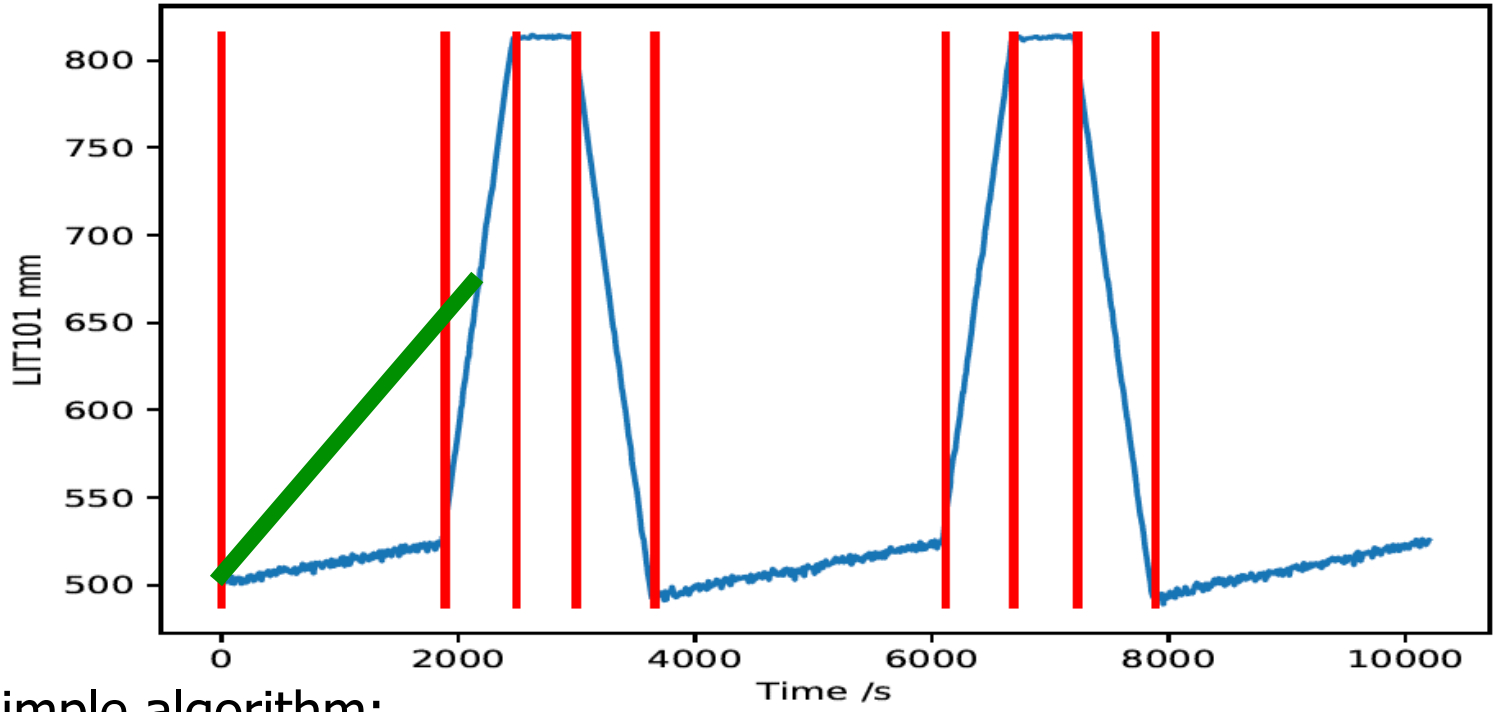
- Simple algorithm:
  1. Add next point to linear segment
  2. Compute regression error e
  3. If e < bound -> goto 1

In SCADA cyclic behavior



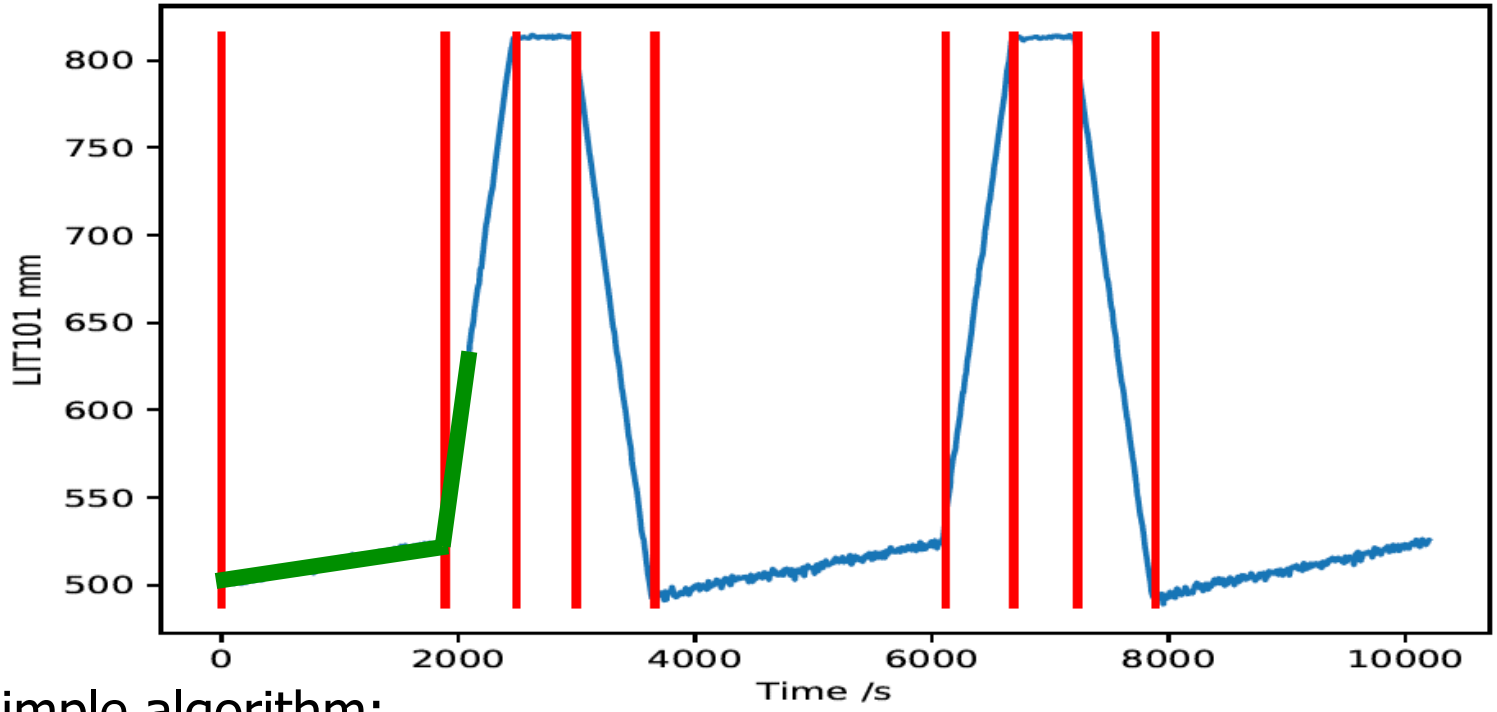
- Simple algorithm:
  1. Add next point to linear segment
  2. Compute regression error e
  3. If e < bound -> goto 1

In SCADA cyclic behavior



- Simple algorithm:
  1. Add next point to linear segment
  2. Compute regression error e
  3. If e < bound -> goto 1
  4. Else -> cut at point of largest error, continue from there

In SCADA cyclic behavior

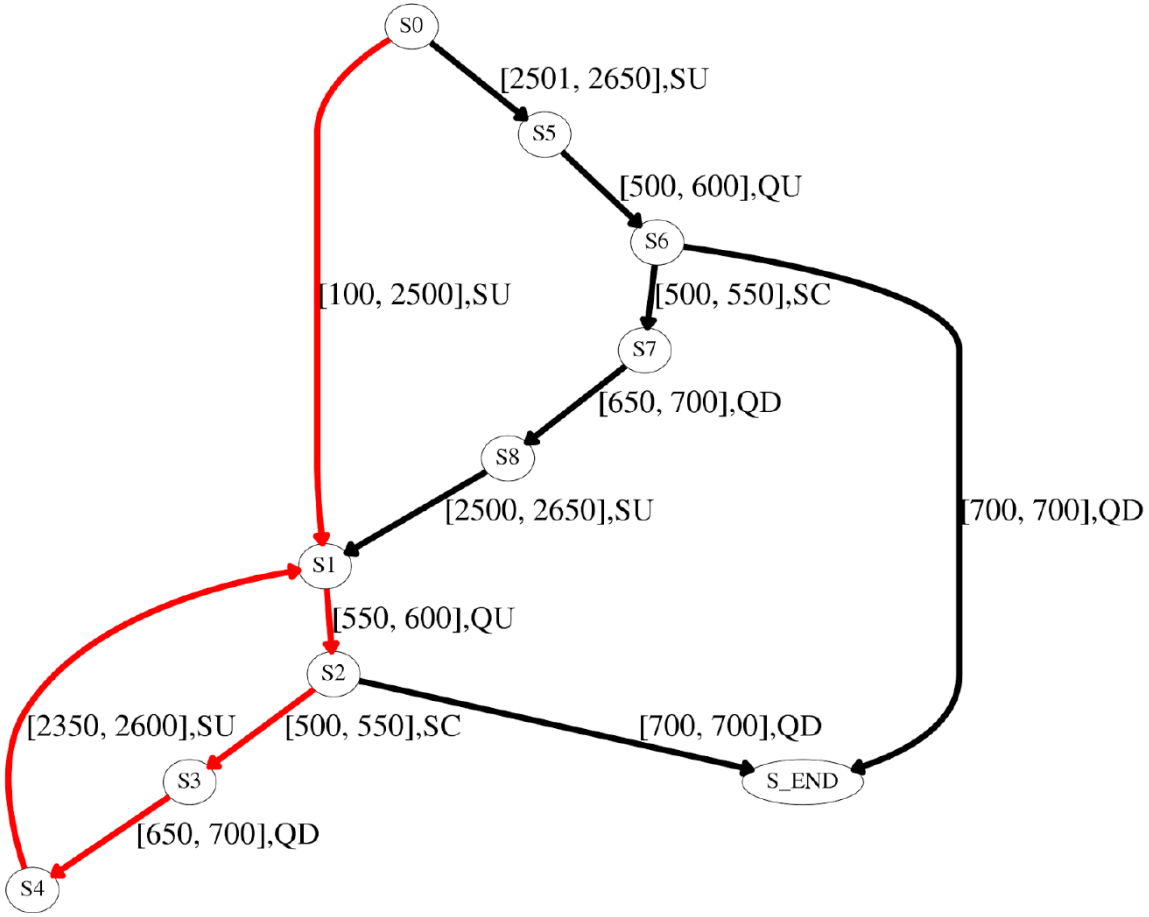


- Simple algorithm:
  1. Add next point to linear segment
  2. Compute regression error e
  3. If e < bound -> goto 1
  4. Else -> cut at point of largest error, continue from there

More interpretable state machine model of signal

- SU = slightly up
- QU = quickly down
- SC = stay constant
- QD = quickly down

- [u,v] are temporal
- bounds



Important choices in discretization

- The level of aggregation
  - Discretize every individual data point, a sequence of points, with or without gaps?
- The behavior to model
  - Exact values, trends, differences, fluctuations?
- My advice:
  - do what makes sense and leads to **interpretable outcomes**
  - **when uncertain: try different possibilities**

Learning from text

- Typically simple approaches:

- Bags of Words
  - count word occurrences (spam filters, not this course)
- Markov chains
  - predict next word given current word (time series modeling, **malware detection**)
- N-grams
  - predict next word, given pervious N-1 words (author attribution, **malware detection**)

Sequential Processes

- Model state-based systems

  $x_t \in \{1, 2, \ldots, N\}$ = state at time *t*

- State evolution is typically random

- Used to model the probability of state sequences and to predict future states or events

Markov Processes

- Assumes the Markov property:

$$p(x_{t+1} \mid x_0, \ldots, x_t) = p(x_{t+1} \mid x_t)$$

"Conditioned on the present, the future is independent from the past"

State Transition Matrices

- A stationary Markov chain with *N* states is described by an *NxN* transition matrix:

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

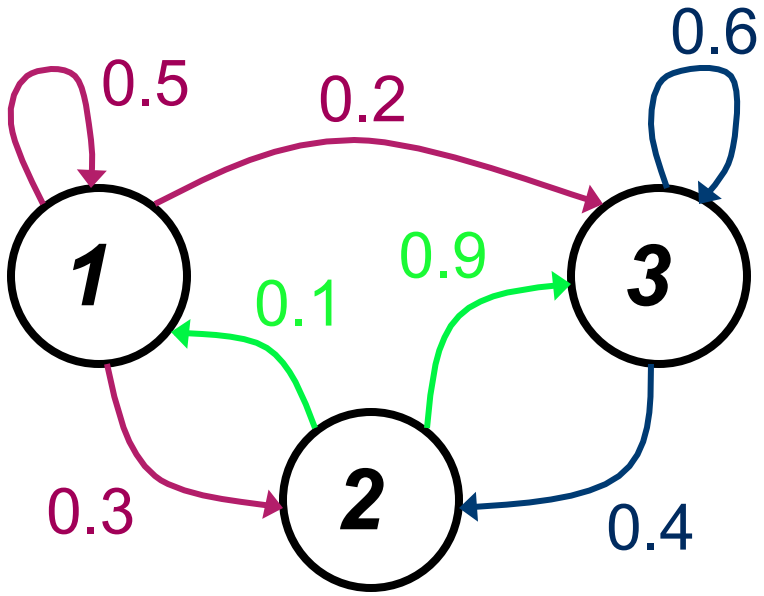
$$q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j)$$

- Constraints on valid transition matrices:

$$q_{ij} \geq 0 \qquad \sum_{i=1}^{N} q_{ij} = 1 \quad \text{for all } j$$

State Transition Diagrams

$$Q = \begin{bmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{bmatrix}$$

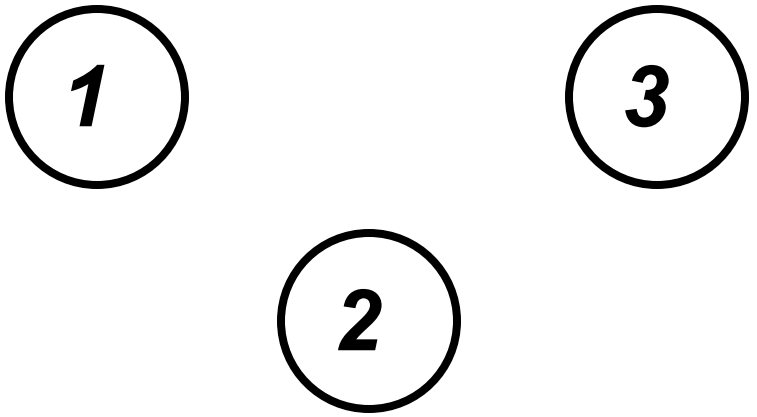


Parameter Estimation

value	state
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3

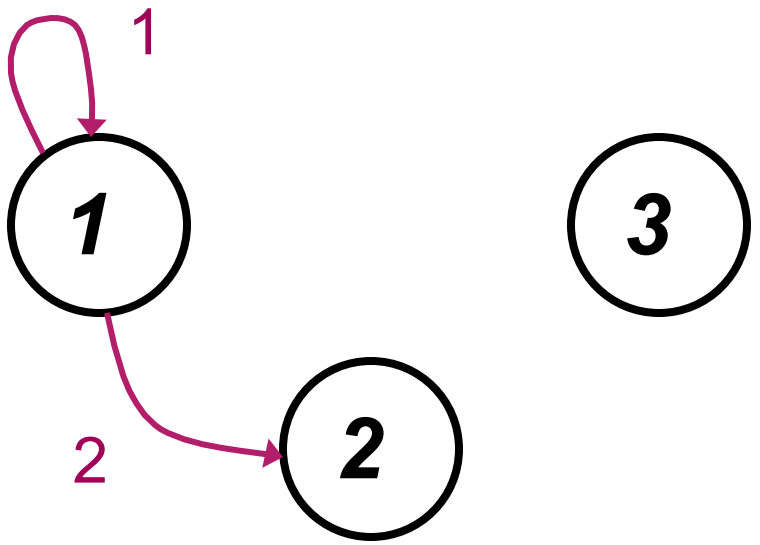
Parameter Estimation

value	state
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3



Parameter Estimation

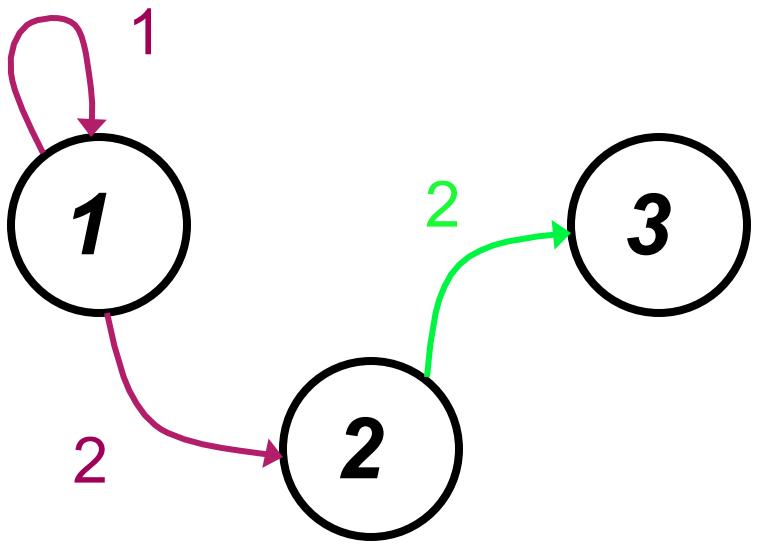
value	state
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3



Count transition occurrences

Parameter Estimation

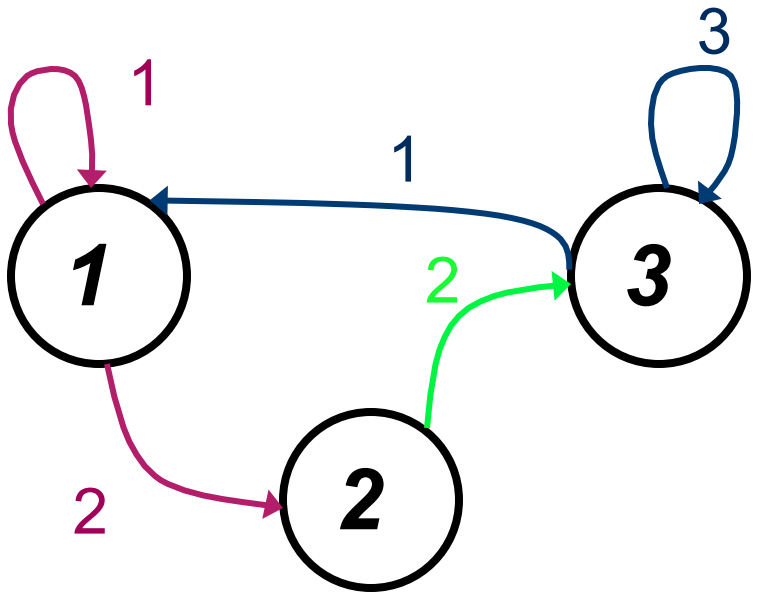
value	state
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3



Count transition occurrences

Parameter Estimation

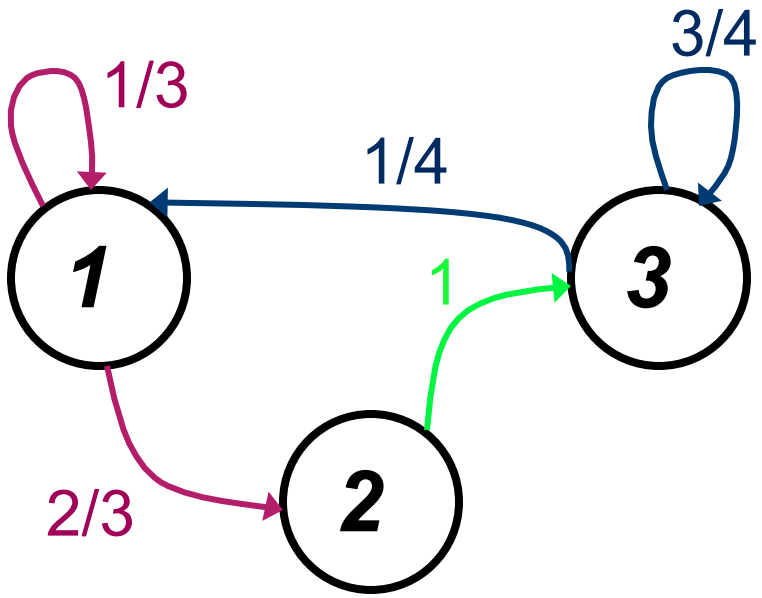
value	state
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3



Count transition occurrences

Parameter Estimation

value	state
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3



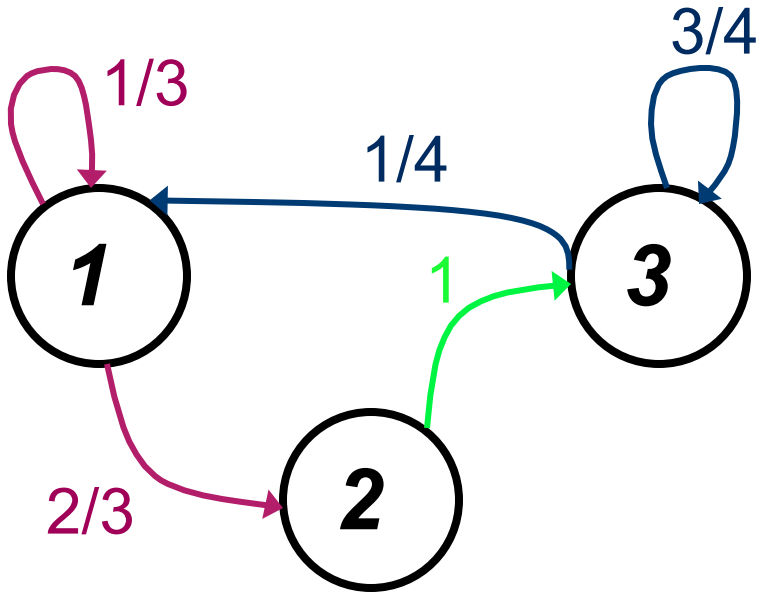
Normalize transition counts to sum up to 1.0

Predicting new state sequences

Prob(1123) =

P(1) * P(1→1) *
P(1→2) * P(2 →2) =

?



$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1})$$

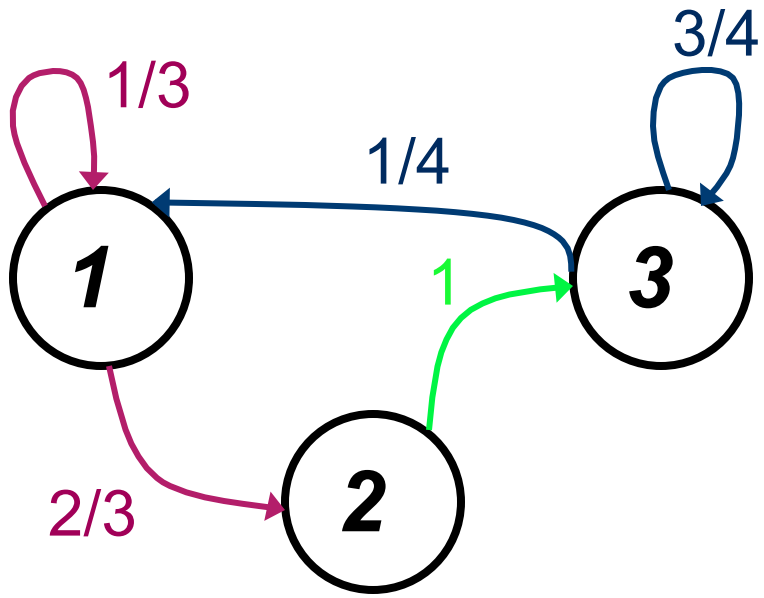
Predicting new state sequences

Prob(1123) =

P(1) * P(1→1) * P(1→2) * P(2 →2) =

3/10 * 1/3 * 2/3 * 1 =

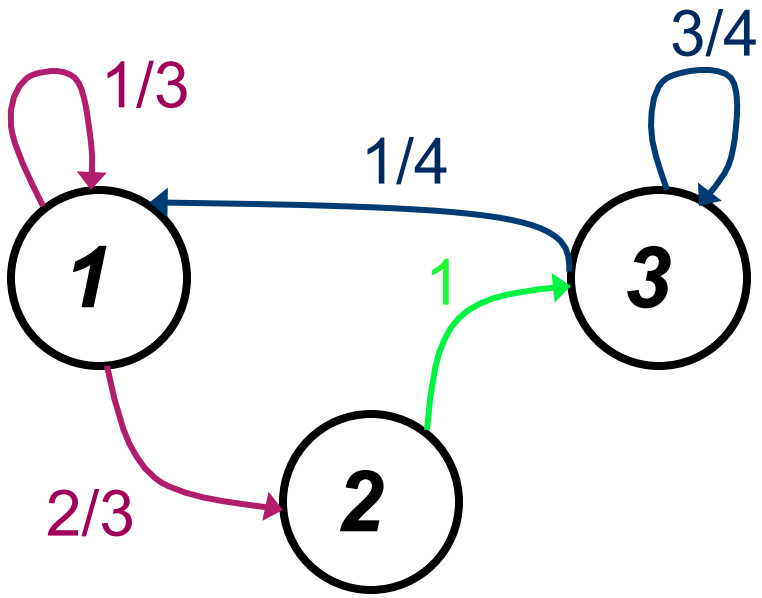
6/90



$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1})$$

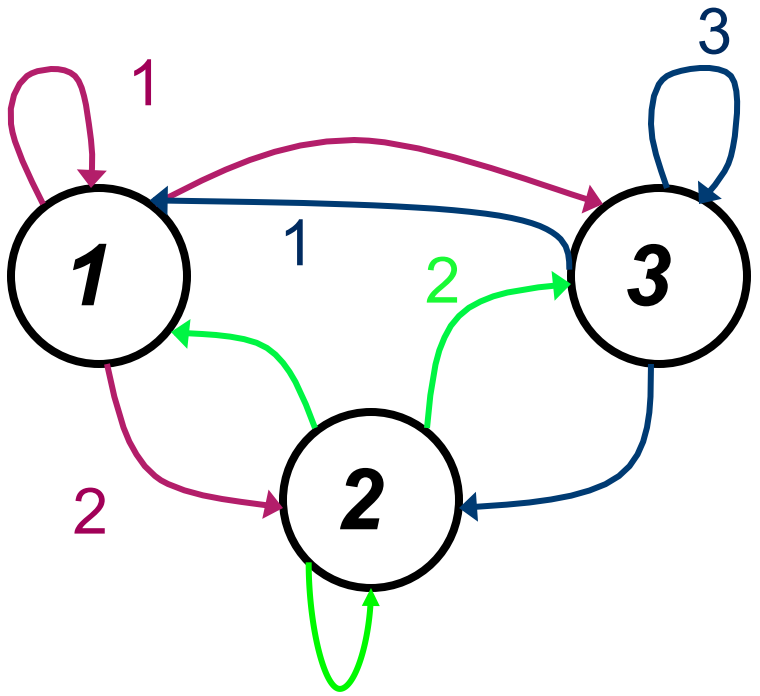
Avoiding zero probabilities

- Some sequences, i.e., 1121, have zero probability under this model
- Models typically assume that everything has non-zero probability
- This is avoided using smoothing



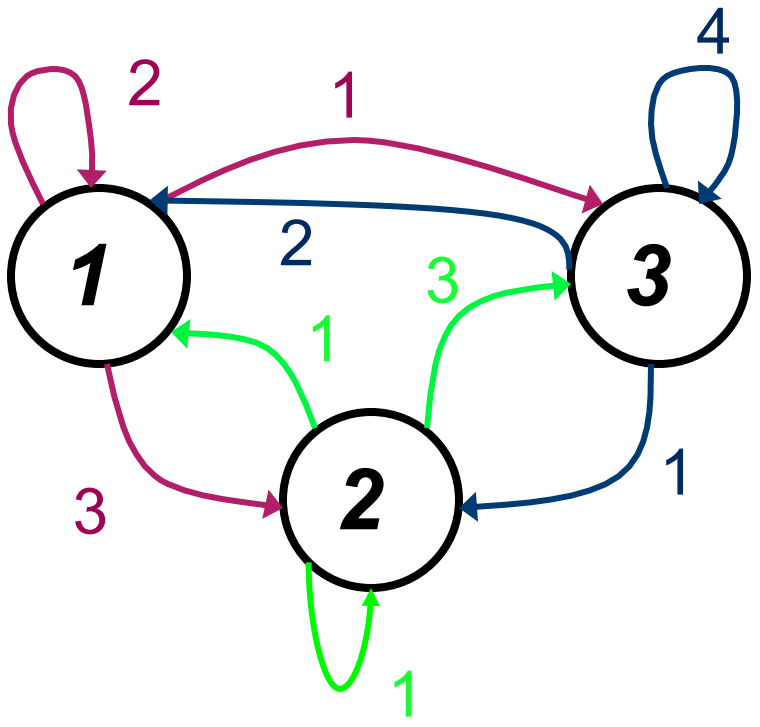
Laplace smoothing

- Add a count of 1 to every possible state transition before normalization



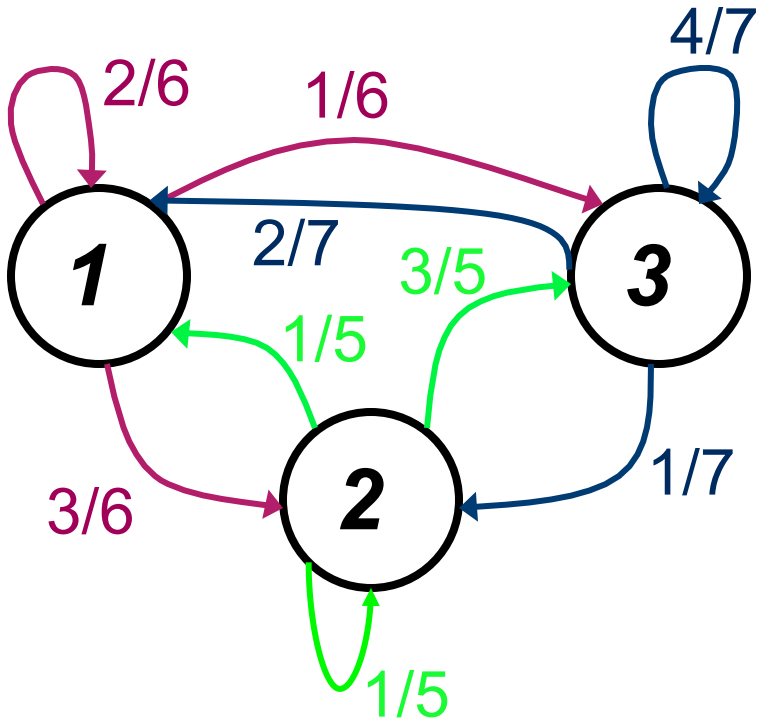
Laplace smoothing

- Add a count of 1 to every possible state transition before normalization



Laplace smoothing

- Add a count of 1 to every possible state transition before normalization



Markov chains

- Intuitive representation of sequence probabilities
  - Structure is informative
- Easy to estimate and use
  - Large bias, low variance
- Only applicable when the state is observed!
- Huge amount of applications in physics, statistics, economics, …
- N-grams are a type of Markov chains based on sliding windows…

N-grams

value	symbol
4	1
4	1
8	2
10	3
11	3
6	1
8	2
10	3
14	3
12	3

s1	s2	s3
1	1	2
1	2	3
2	3	3
3	3	1
3	1	2
1	2	3
2	3	3
3	3	3

N-grams

- Every combination of past sequences is a state
- States:
  - 11, 21, 31
  - 12, 22, 32
  - 13, 23, 33
- Transitions:
  - P(11→12) = P(2|11) = P(112) / P(11?)

s1	s2	s3
1	1	2
1	2	3
2	3	3
3	3	1
3	1	2
1	2	3
2	3	3
3	3	3

N-grams

- Every combination of past sequences is a state

Prob(1123) =

P(11) * P(11→12) * P(12→23) =

?

s1	s2	s3
1	1	2
1	2	3
2	3	3
3	3	1
3	1	2
1	2	3
2	3	3
3	3	3

$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}, x_{t-2})$$

N-grams

- Every combination of past sequences is a state

Prob(1123) =

P(11) * P(11→12) * P(12→23) =

1/8 * 1 * 1 = 1/8

s1	s2	s3
1	1	2
1	2	3
2	3	3
3	3	1
3	1	2
1	2	3
2	3	3
3	3	3

$$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t \mid x_{t-1}, x_{t-2})$$

N-grams

- Popular tools in natural language processing (NLP)
  - Other names: bigrams, trigrams, higher-order Markov chains
- Almost always need to be combined with smoothing methods to avoid overfitting
  - Laplace (add X counts)
  - Back-off, learn N-grams for N in {1, .. , N} if Nth model is badly estimated, use N-1th model, ...
  - …
- Only approximates the true model, structure is not very meaningful to visualize or analyze

Malware detection using Byte n-grams

- Learn profiles from known malicious code:
  - n-grams, where every word/symbol is a byte from a malicious executable
  - Keep top L n-grams (most frequent for each file)
- Match these profiles using Nearest Neighbor, with somewhat arbitrary distance:

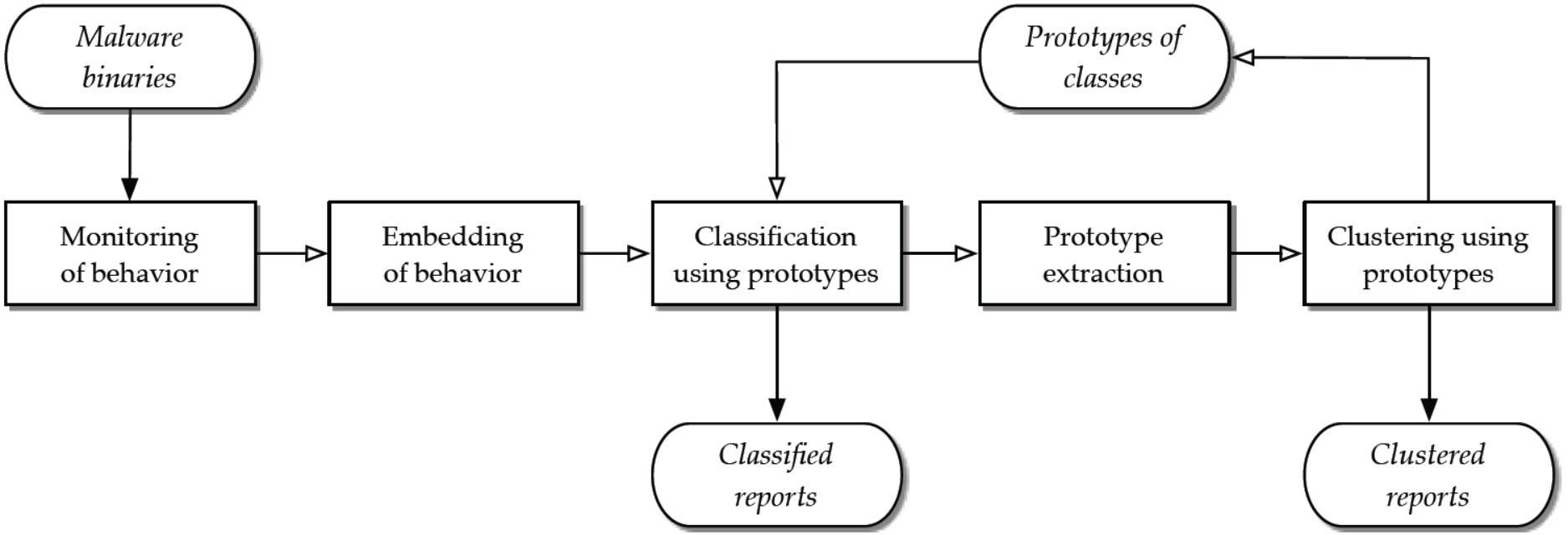
$$\sum_{s \in profiles} \left(\frac{f_1(s) - f_2(s)}{\frac{f_1(s)+f_2(s)}{2}} \right)^2$$

- where $f_i(s)$ is the frequency of n-gram s in file I
- another metric would be to use a Chi-squared test

Reported detection rates, see paper

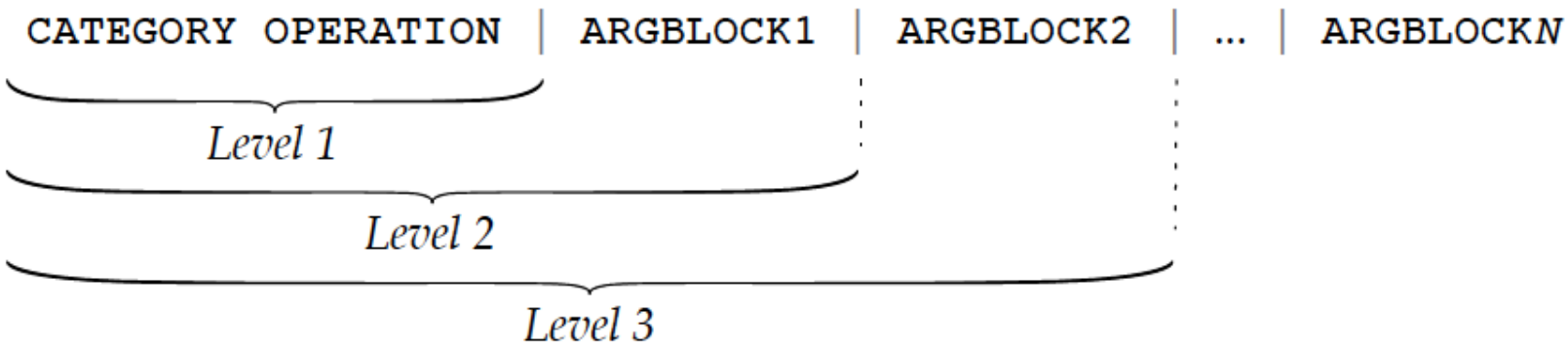
L	n									
	1	2	3	4	5	6	7	8	9	10
20	0.59	0.49	0.61	0.64	0.72	0.64	0.57	0.49	0.50	0.45
50	0.67	0.55	0.80	0.76	0.83	0.83	0.63	0.58	0.60	0.55
100	0.81	0.73	0.72	0.73	0.70	0.84	0.81	0.79	0.81	0.82
200	0.77	0.69	0.69	0.66	0.79	0.85	0.86	0.85	0.87	0.86
500	0.56	0.85	0.85	0.78	0.82	0.86	0.87	0.85	0.86	0.86
1000	0.56	0.84	0.89	0.89	0.90	0.88	0.85	0.87	0.88	0.87
1500	0.56	0.82	0.89	0.91	0.88	0.89	0.87	0.85	0.84	0.85
2000	0.56	0.83	0.89	0.89	0.87	0.87	0.85	0.83	0.82	0.84
3000	0.56	0.82	0.88	0.88	0.87	0.84	0.80	0.82	0.80	0.81
4000	0.56	0.80	0.87	0.86	0.83	0.81	0.79	0.81	0.78	0.80
5000	0.56	0.79	0.86	0.83	0.81	0.81	0.79	0.79	0.77	0.78

Malware detection using MIST n-grams



Malware detection using MIST (malware instructions set) n-grams

- MIST instructions are words

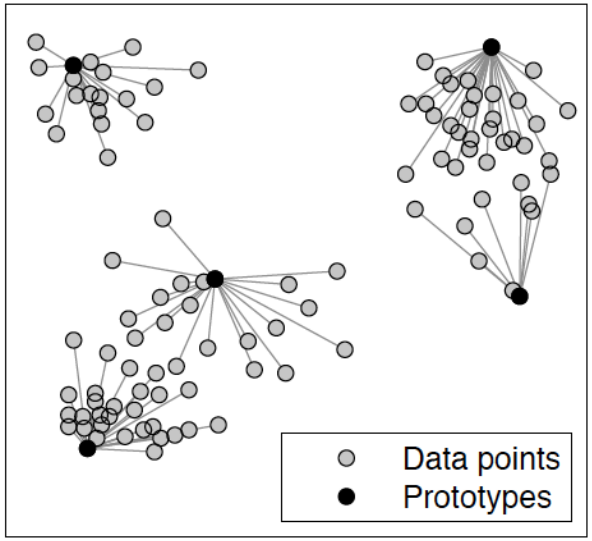


- Classify using nearest prototype...

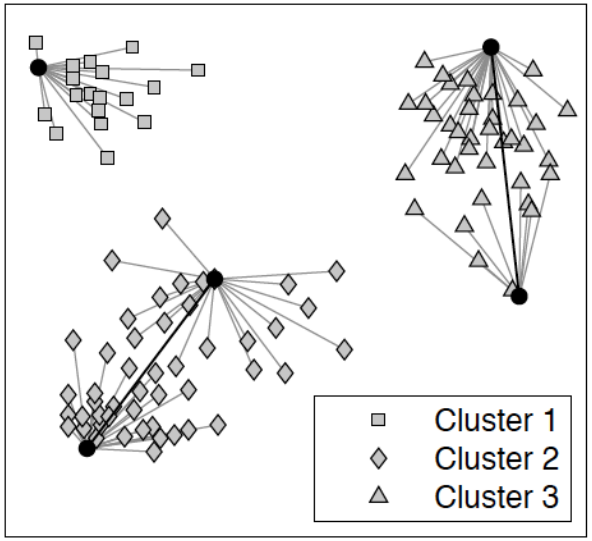
$$\varphi(x) = (\varphi_s(x))_{s \in \mathcal{S}} \text{ with } \varphi_s(x) = \begin{cases} 1 & \text{if report } x \text{ contains } q\text{-grams } s \\ 0 & \text{otherwise.} \end{cases}$$

Prototyping

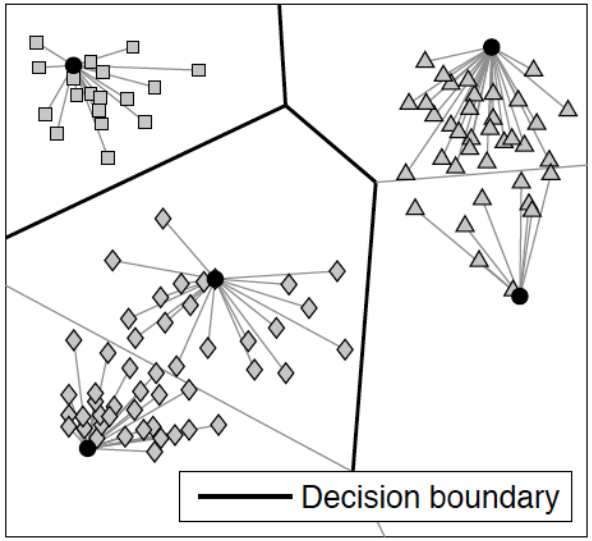
- Simple solution to speed up Nearest Neighbor



(a) Prototypes



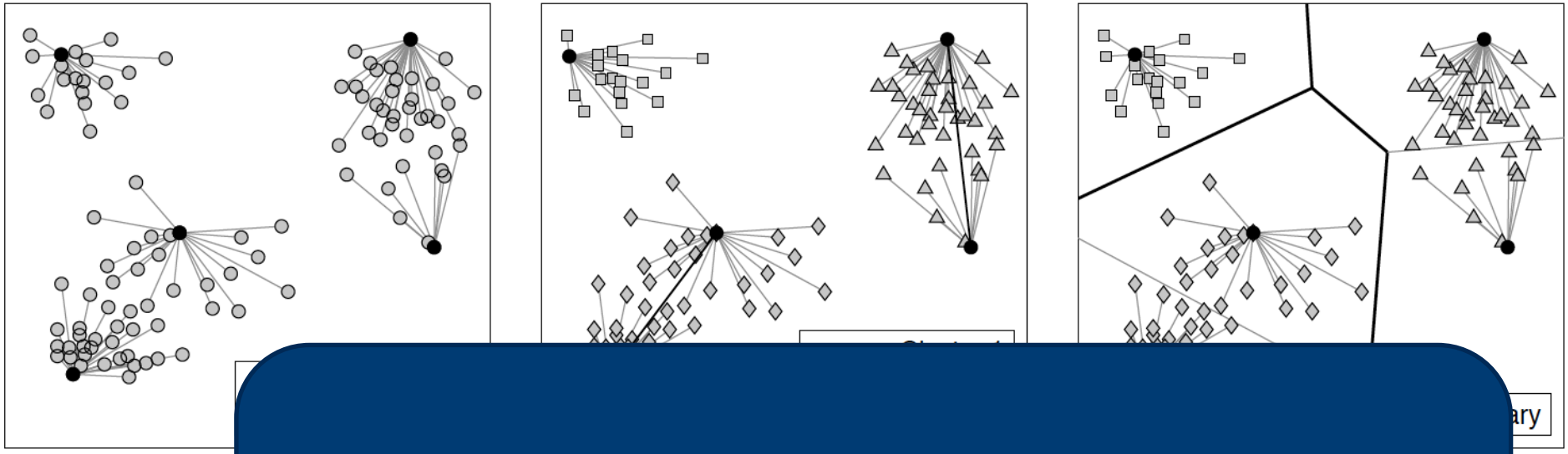
(b) Clustering



(c) Classification

Prototyping

- Simple solution to speed up Nearest Neighbor



(a) Pr

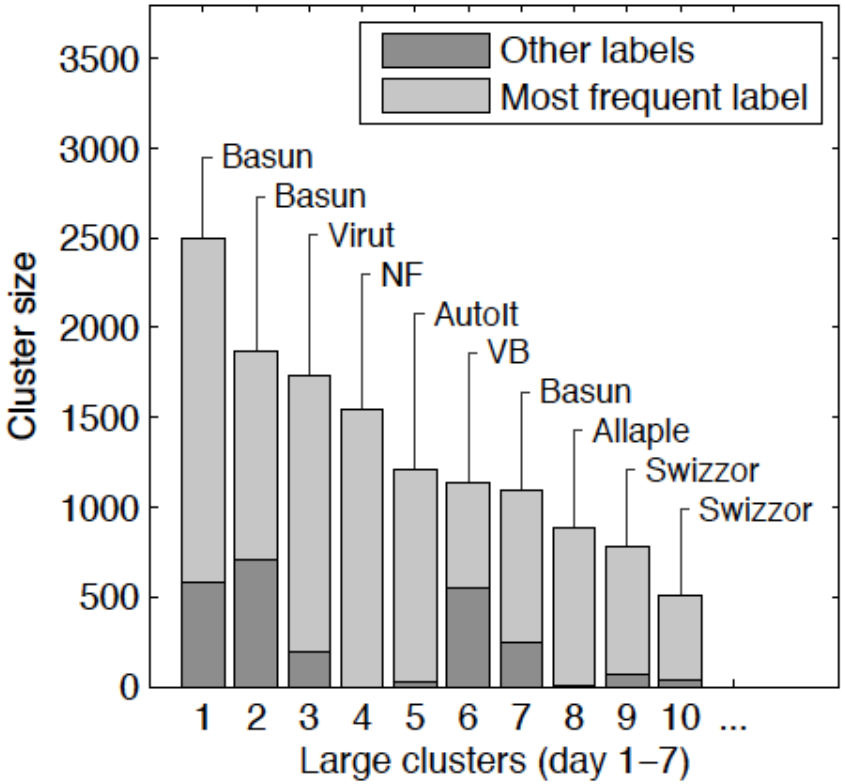
Reduces number of required comparisons from thousands to a handful!

Some results

- High accuracy scores

Classification methods	F_k	F_u
Classification using prototypes with MIST level 1	0.981	0.997
Classification using prototypes with MIST level 2	0.972	0.964

- and, sensible clusters



Encrypted traffic classification using MCs

- SSL/TLS sessions, ignoring payloads

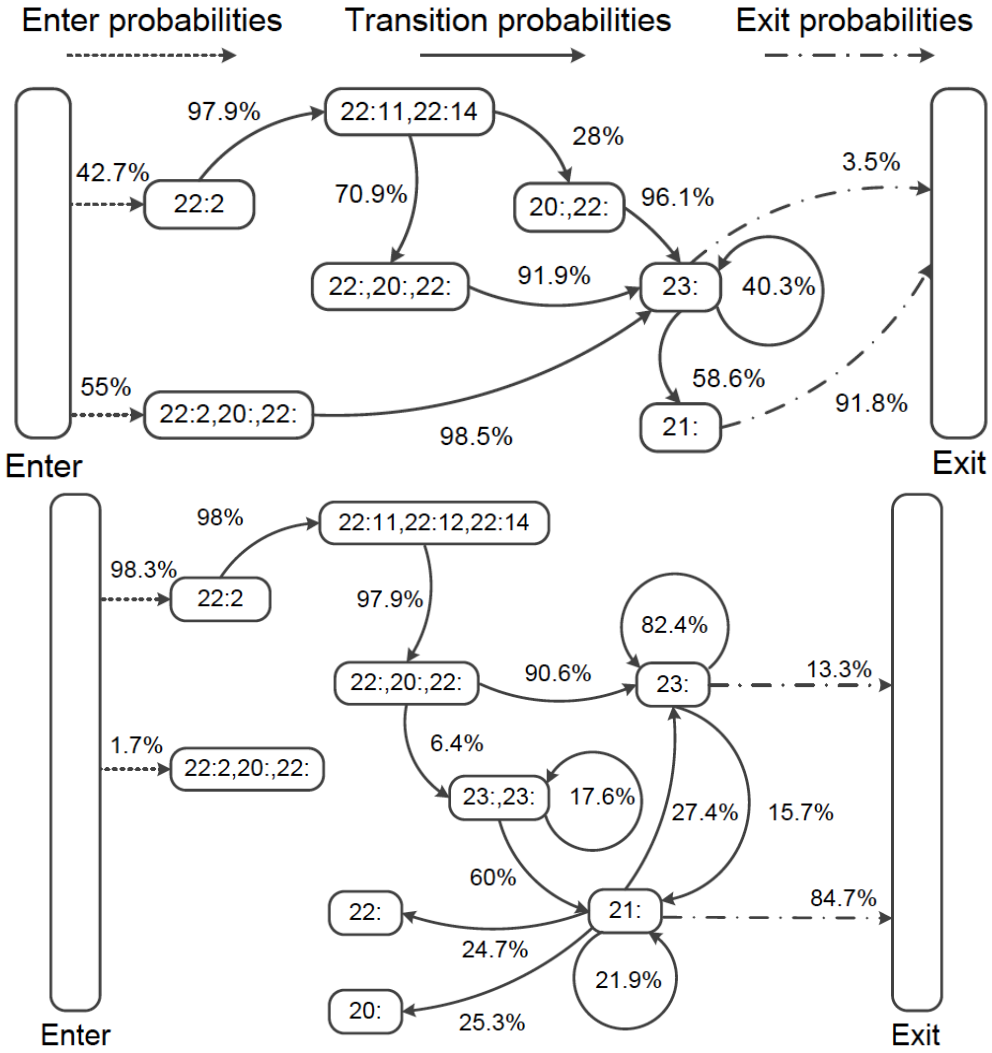
Decimal Code	Protocol Type
20	Change Cipher Spec
21	Alert
22	Handshake
23	Application Data

Decimal Code	Handshake Message Type
0	Hello Request
1	Client Hello
2	Server Hello
11	Certificate
12	Server Key Exchange
13	Certificate Request
14	Server Hello Done
15	Certificate Verify
16	Client Key Exchange
20	Finished

- Input:
  - 22:2-22:11,22:14-20:,22:-23:
  - 22:2,20:,22:-23:
  - 22:2-22:11,22:14-20:,22:-23:-23:-21:
- Every message type (22:2-22:11, etc.) is a state in a Markov chain

Example Markov chains

- For Twitter:
- For Dropbox:



Classification using maximum likelihood

- Classify new traffic using the Markov chain that maximizes likelihood:

$$H = \arg\max_{H_i} \log L(\{Y_1, \ldots, Y_T\}|H_i)$$

- Some results:

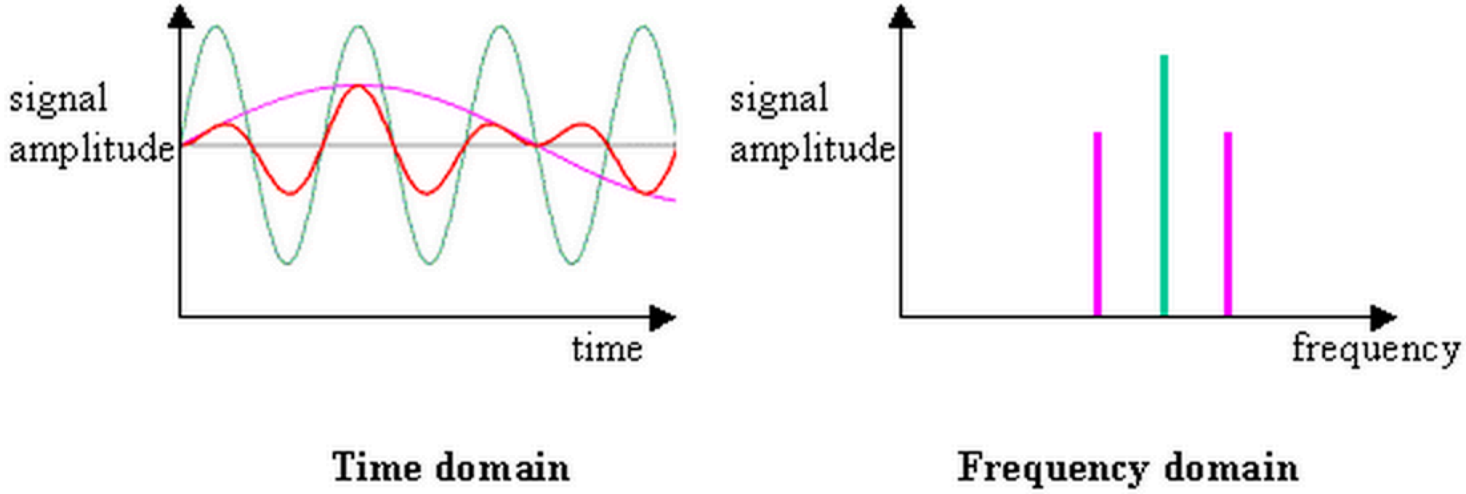
	Campus1		Campus2		Campus3	
Application	TPR	FPR	TPR	FPR	TPR	FPR
Twitter	0.936	0.01	0.911	0.041	0.887	0.026
Dropbox	0.672	0.005	0.7	0.005	0.919	0.06
Gadu-Gadu	0.975	0.011	0.929	0.01	0.684	0.013
Mozilla	0.001	0.029	0.0	0.023	0.29	0.035
MBank	0.0	0.013	0.0	0.01	0.903	0.037
PKO	0.521	0.005	0.489	0.003	0.575	0.032
Dziekanat	0.959	0.092	0.929	0.012	0.994	0.0
Poczta	0.924	0.002	0.935	0.003	0.97	0.0
Amazon S3	0.982	0.142	0.99	0.112	0.994	0.0
Amazon EC2	0.146	0.044	0.001	0.079	0.598	0.01

So far

- Discretization:
  - Often required to make continuous signals discrete
  - Use percentiles, SAX, change point detection
  - Whatever you choose, make sure it makes sense
- Markov processes (chains):
  - *The future is independent of the past given the current state*
  - The simplest type of state-transition diagrams
  - **Smooth** in order to avoid 0 probabilities
- N-grams:
  - Markov chains of order N: state is N-1 symbols instead of 1
- Markov chains and N-grams are very successful in cyber

De-noising of signals (extra info, not required for exam)

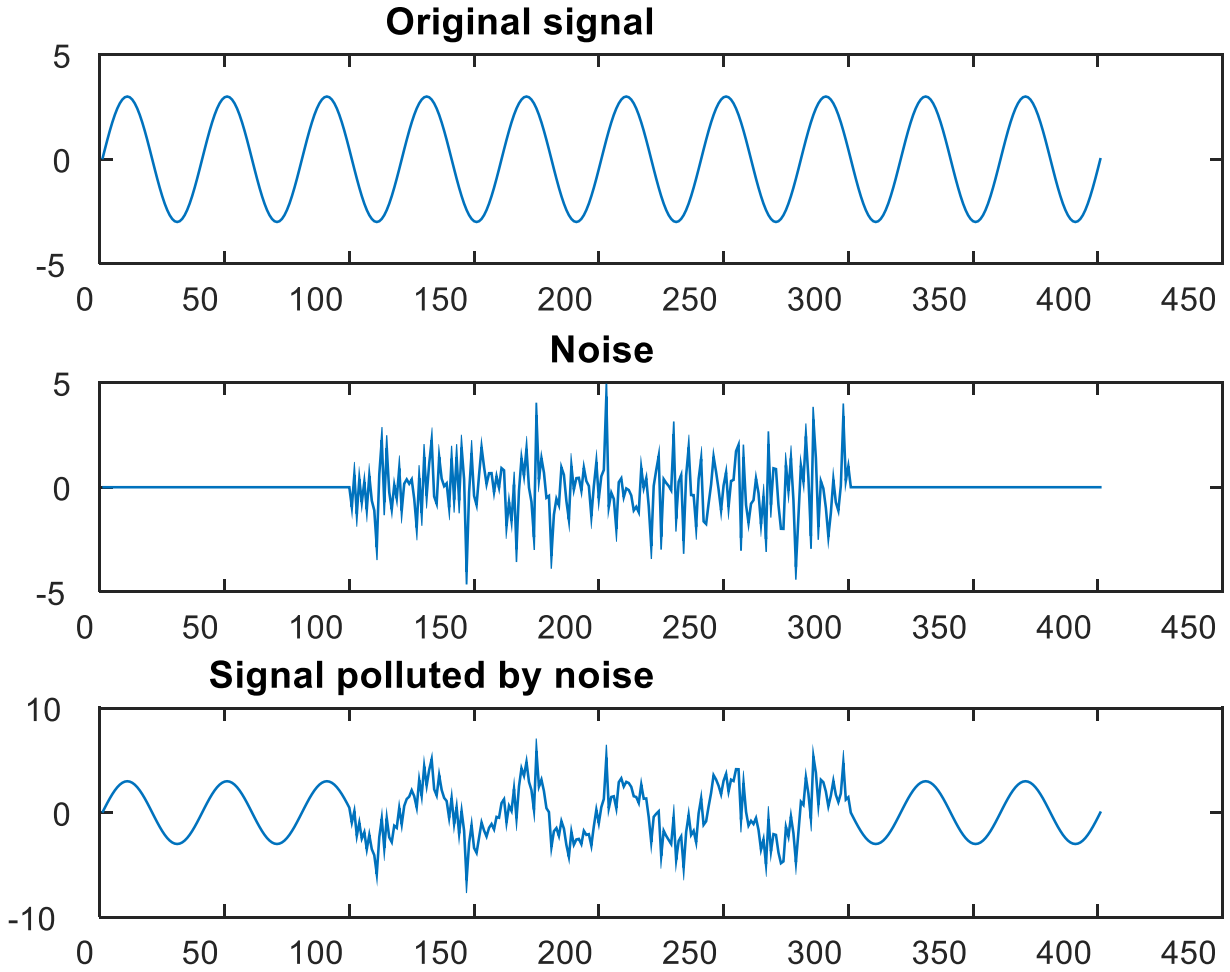
- Signals are inevitably polluted by noise due to some uncontrollable events, which causing bad influence on time series modeling and prediction.



Frequency analysis: A completely different perspective on time series data

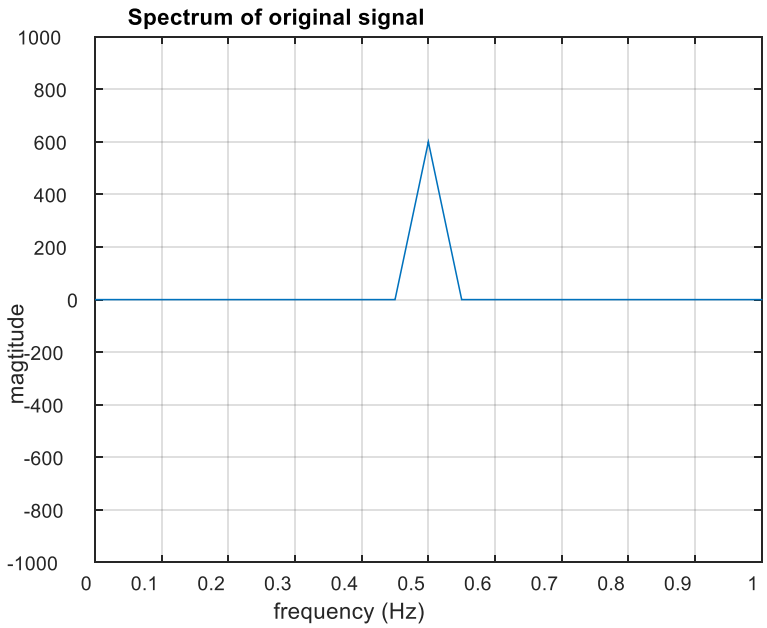
Fourier transform

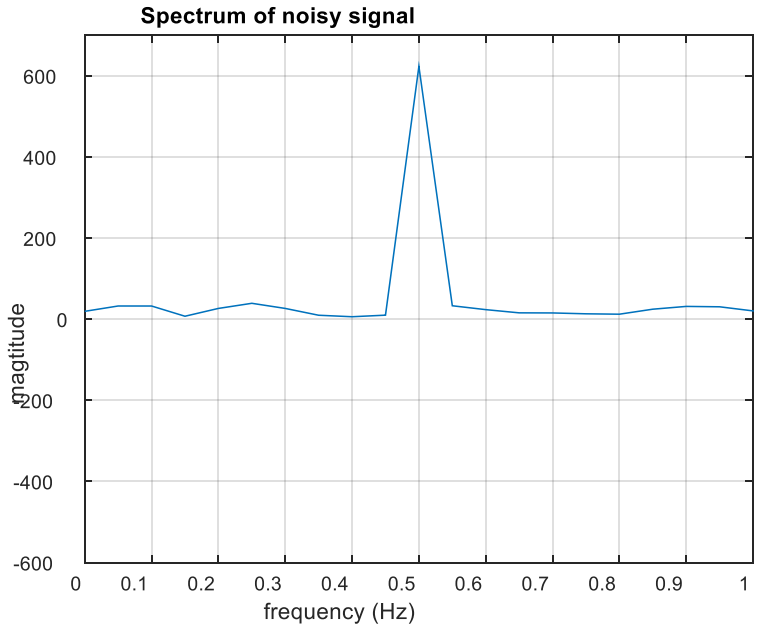
Example: noisy sine signal



Removing the noise

Step1: frequency analysis of the signal with Fourier transform (from **time domain** to **frequency domain**)

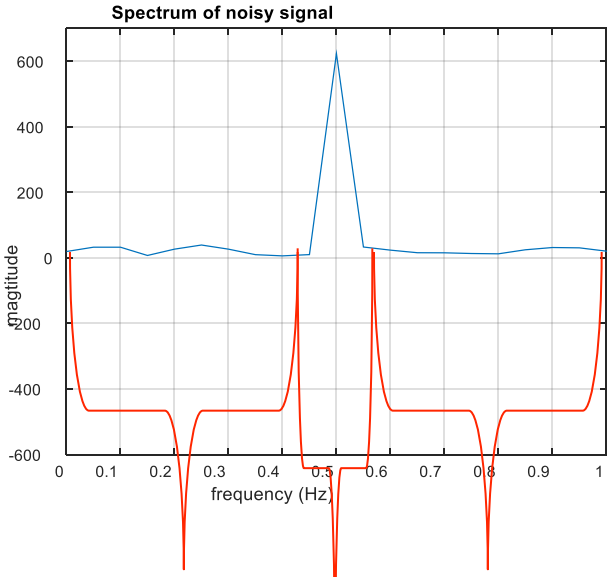




spectrum of signal

Removing the noise

Step2: select the frequency area of useful signal and that of noise (in **frequency domain**)

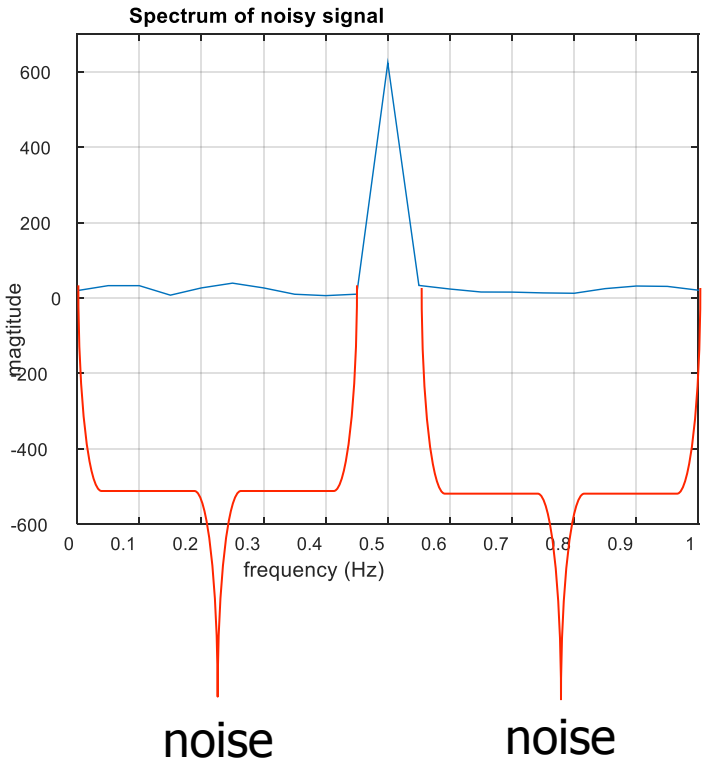


noise noise

useful signal

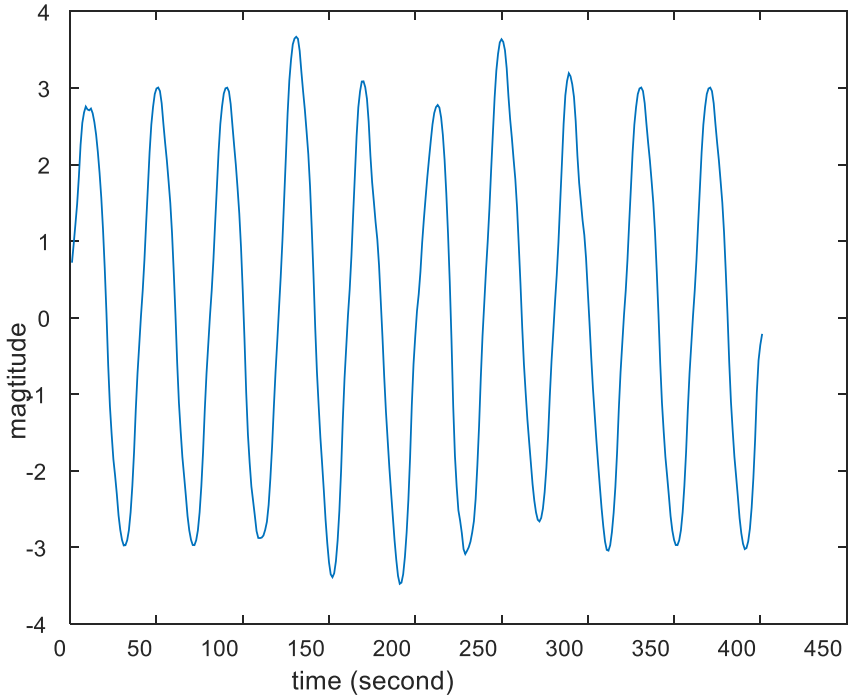
Removing the noise

Step3: reduce the magnitude of noise (in **frequency domain**)



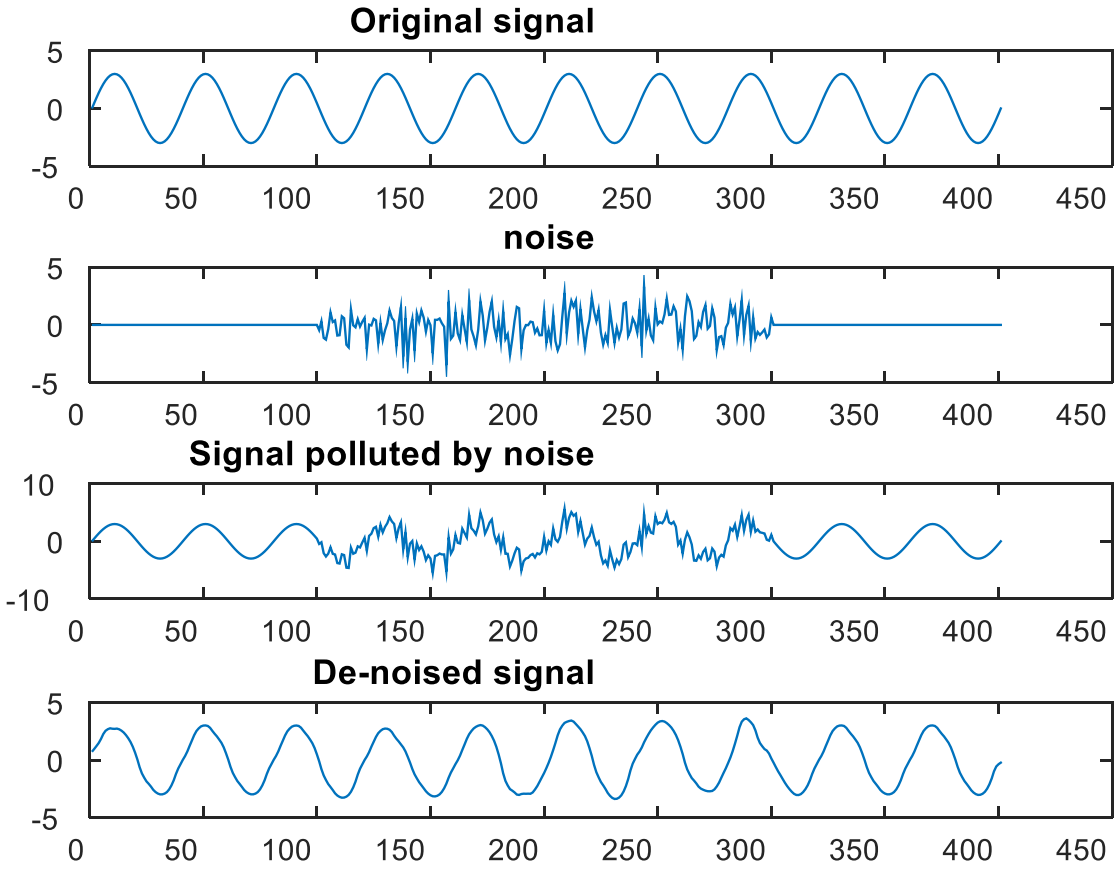
Removing the noise

Step4: reverse the spectrum with inverse-Fourier transform (from **frequency domain** to **time domain**)

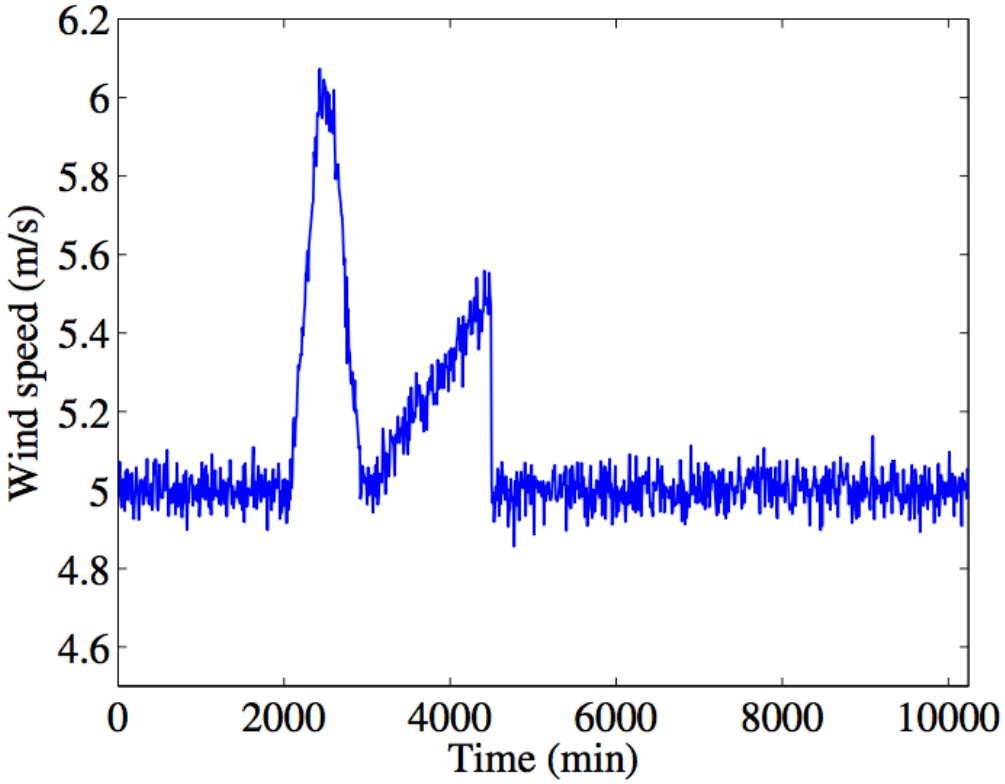


de-noised signal

Results

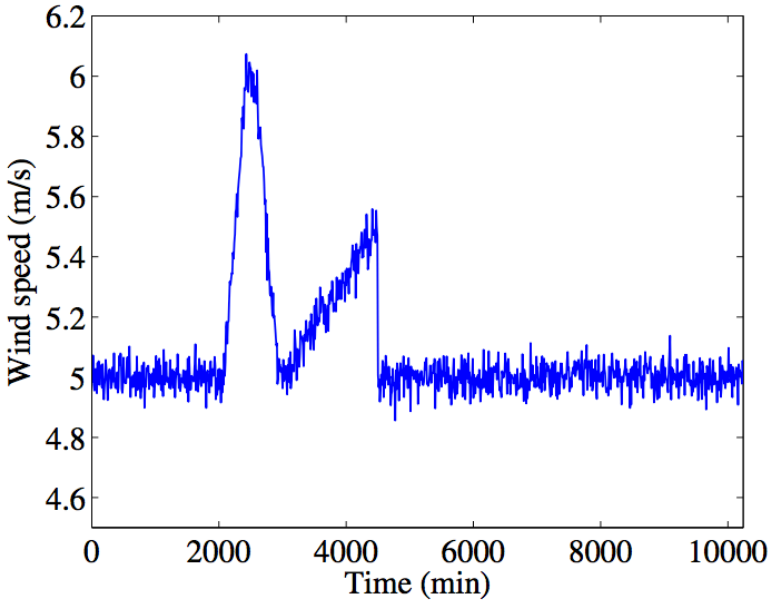

Original signal
5
0
-5
0 50 100 150 200 250 300 350 400 450
noise
5
0
-5
0 50 100 150 200 250 300 350 400 450
Signal polluted by noise
10
0
-10
0 50 100 150 200 250 300 350 400 450
De-noised signal
5
0
-5
0 50 100 150 200 250 300 350 400 450

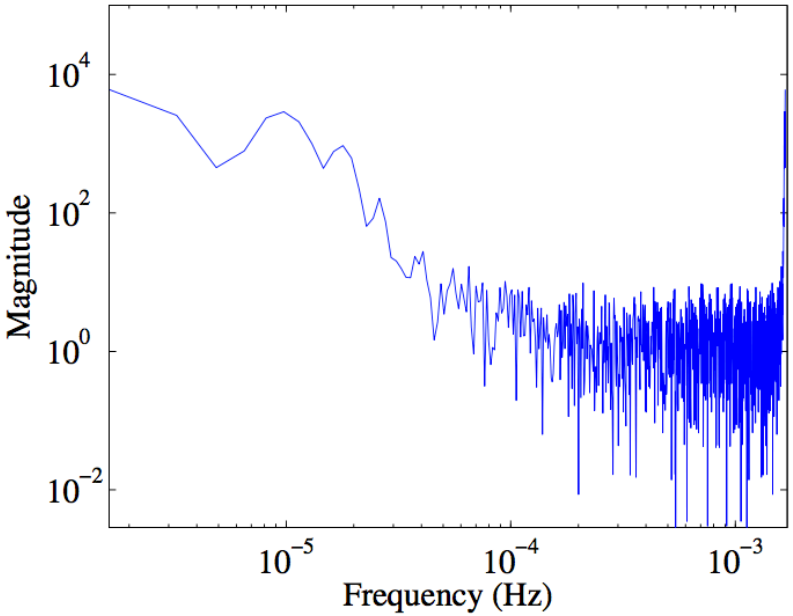
Wind-speed signal



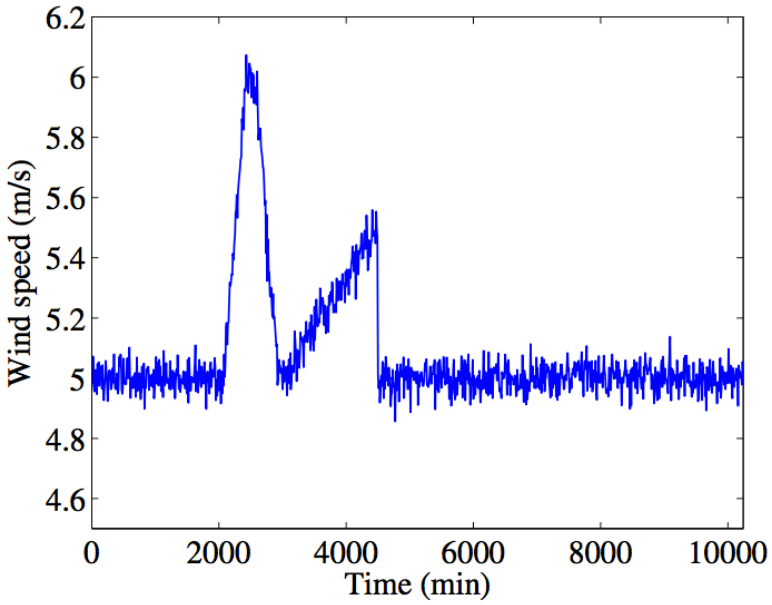
signal polluted by noise

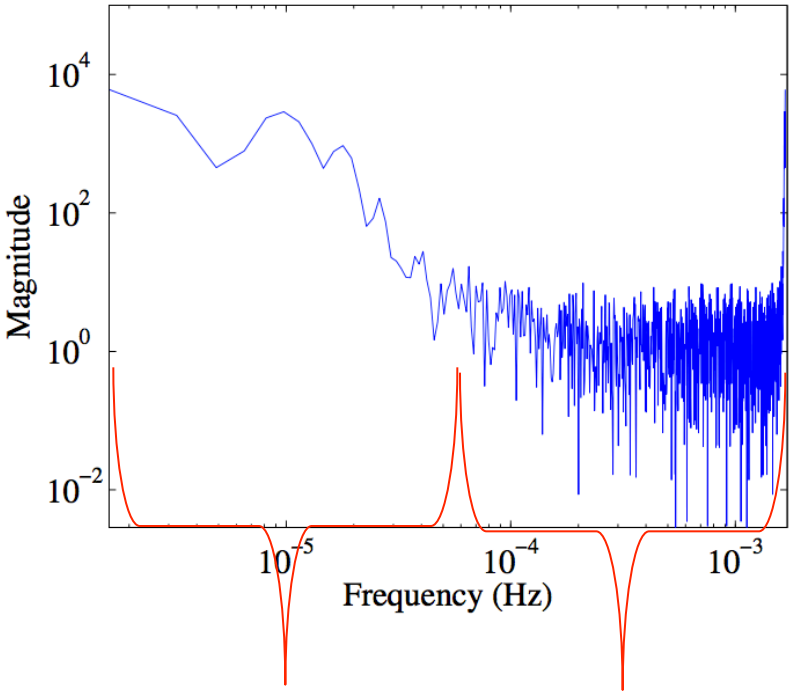
Wind-speed signal





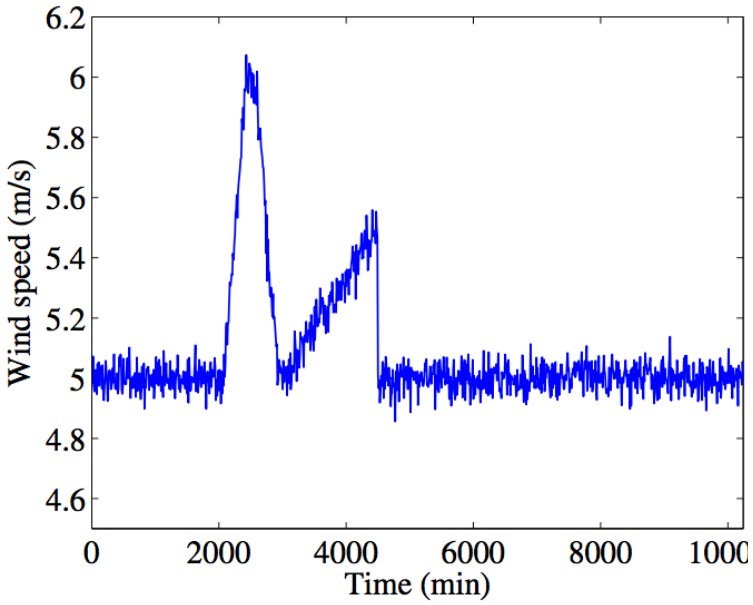
Wind-speed signal

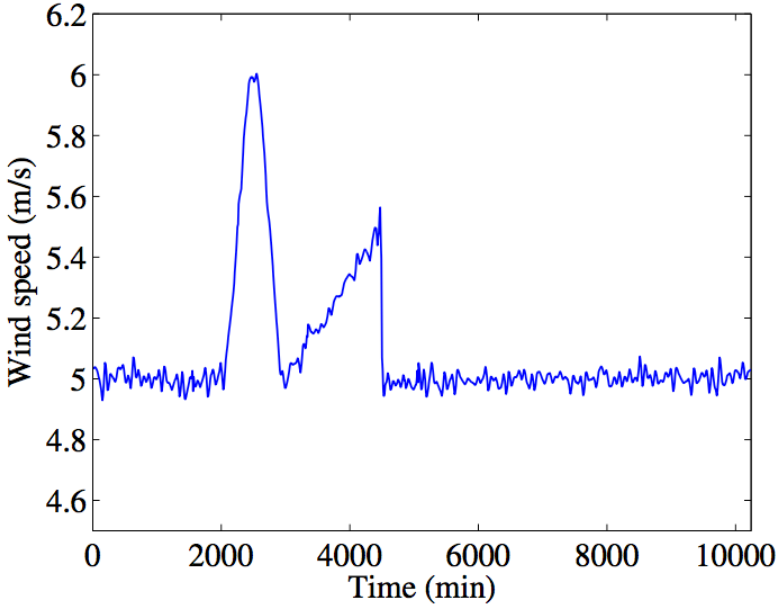




useful signal noise

Wind-speed signal





de-noised signal