Cyber Data Analytics

Aim: identify and prevent cyber criminals using machine learning

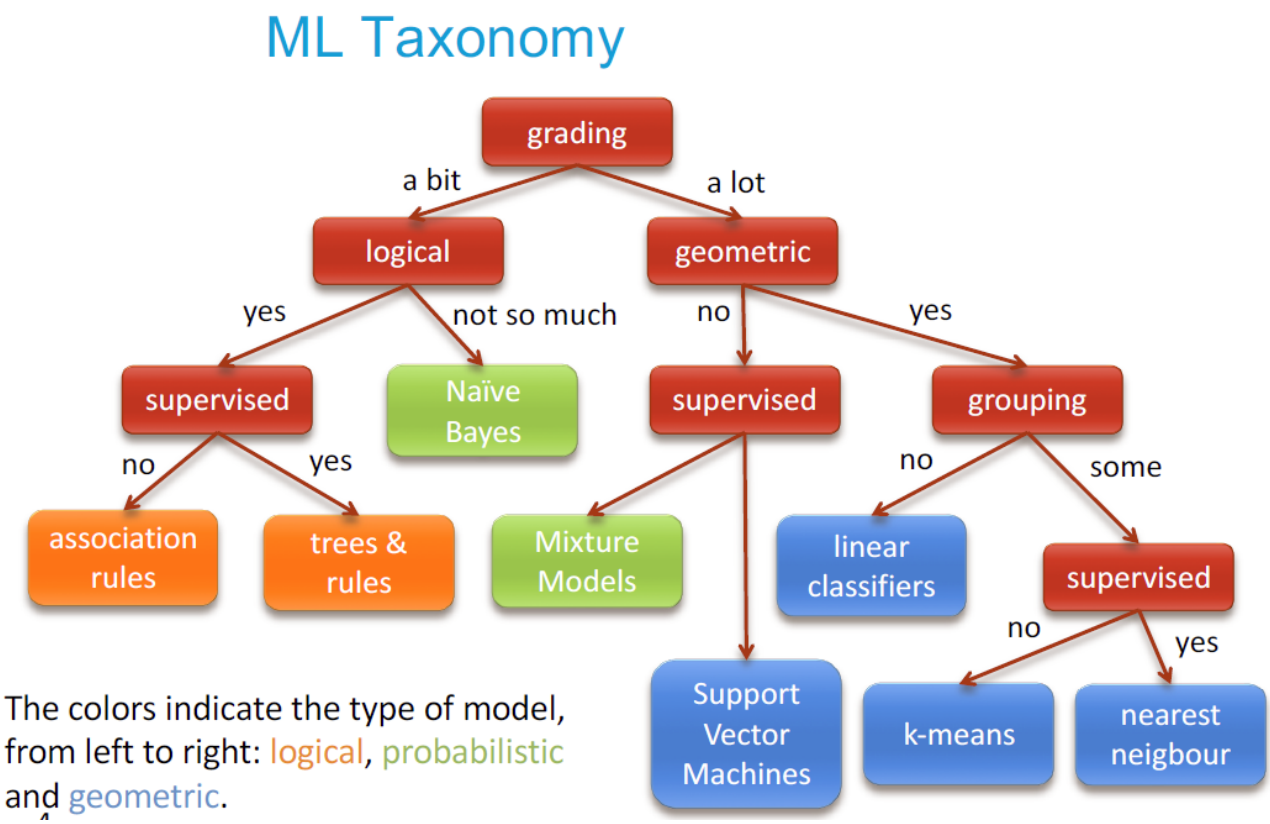
Lecture 1: Fraud Detection

Cyber data:

- huge structured data sets with few positives (fortunately)
- frequently unlabeled
- often containing private information

Machine Learning models

- **Geometric models**: use intuitions from geometry such as separating (hyper-)planes, linear transformations and distance metrics.
- **Probabilistic models**: view learning as a process of reducing uncertainty, modeled by means of probability distributions
- **Logical models**: are defined in terms of easily interpretable logical expressions
- Other classification - Characterized by their way of working
    - Grouping models divide the instance space into segments; in each segment a very simple (e.g., constant) model is learned.
    - Grading models learning a single, global model over the instance space.

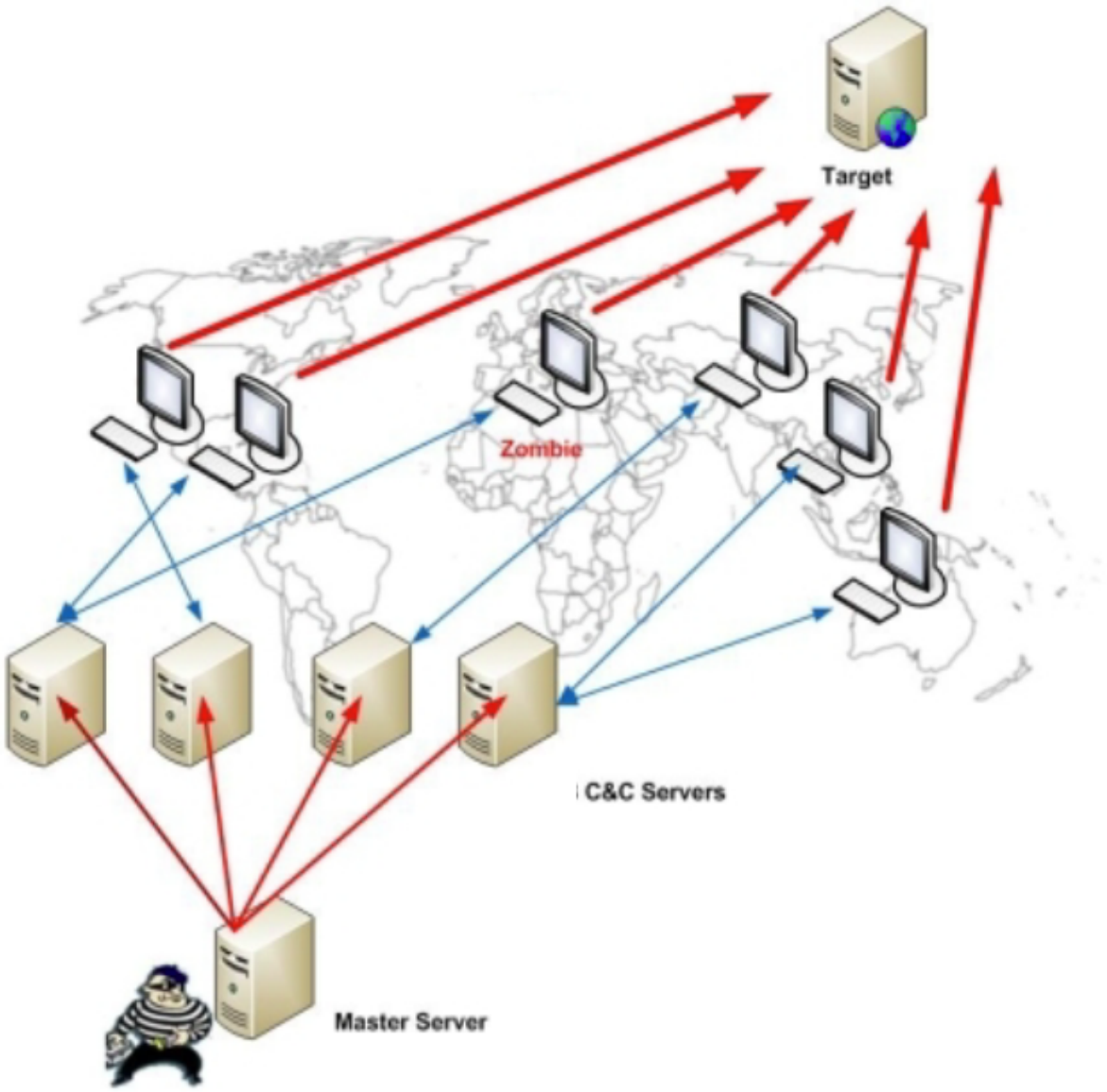


Botnets

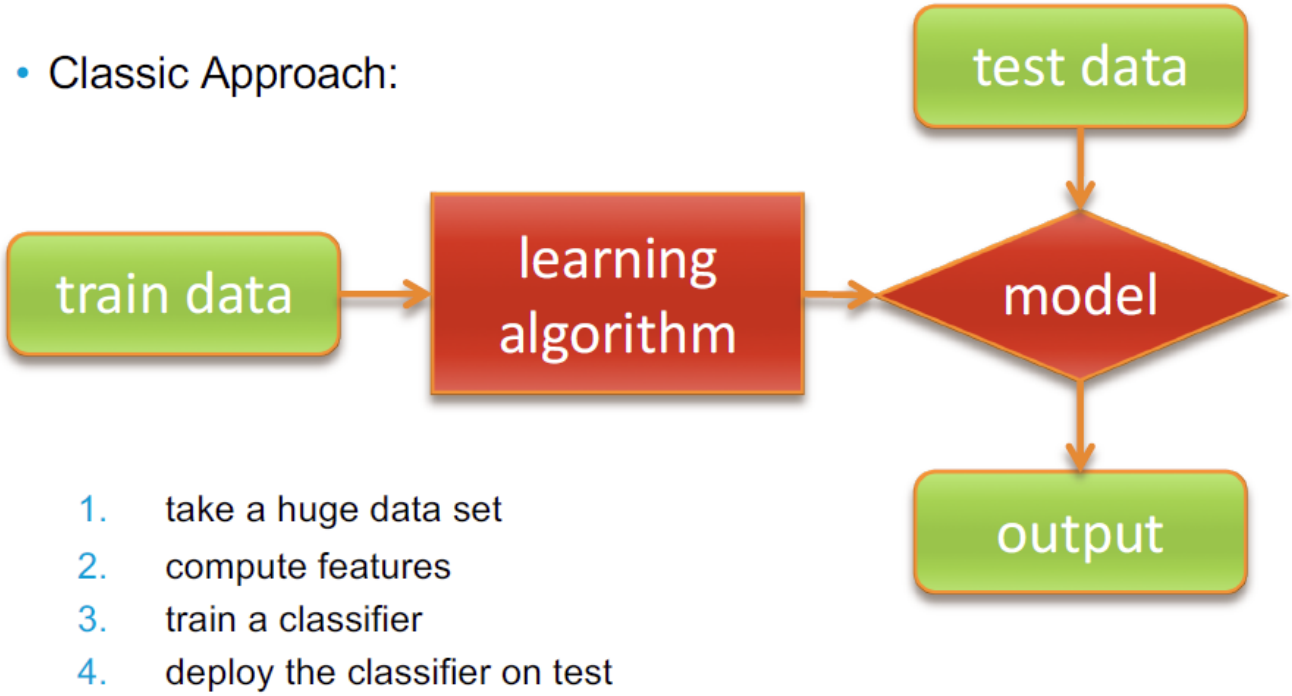
- a number of Internet-connected devices, each of which is running one or more bots
- consisted of a control architecture and thousands-millions of zombies
- Used for massive DDoS (distributed denial-of-service) attacks, sending spam, and keylogging

Prevent botnet

- traditional methods
    - code/binary analysis: is mostly manual and increasingly harder
    - find malware fingerprint
- Behavior-based analysis is much harder to thwart: Bots need to communicate!



Machine learning



Classic Machine Learning Problems

- Large majority (> 99%) of cases are benign
  - Most classifiers are happy with 99% accuracy
- Data is sequential
  - Not IID: Independently and Identically Distributed
- Data is massive and keeps coming in
  - Memory limitations, sometimes loading data fails
- There is an opponent, they learn too
  - If detect pattern, it will change in the future
- Privacy
  - cannot include privacy-related data

Imbalanced data

Solutions:

- Resampling: oversample/undersample minority/majority class
  e.g. flip a coin: head->fraud data, tail-> true data
- Reweighting: assign large/small weights to minority/majority class

- Synthesizing: add artificial minority class instances

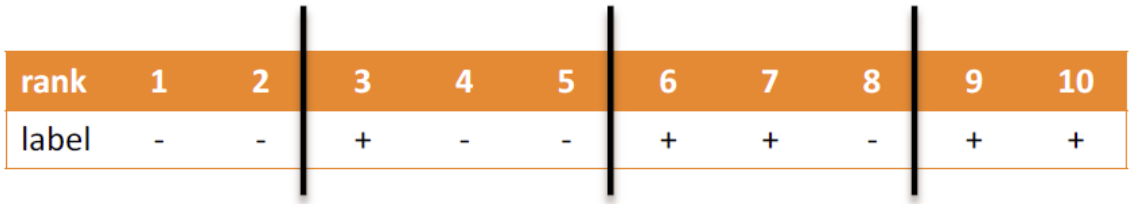
Measuring performance

	Predicted: NO	**Predicted: YES**
Actual: NO	true negatives (TN)	false positives (FP)
Actual: YES	false negatives (FN)	true positives (TP)

- Standard measures
  - Recall = $\frac{TP}{TP+FN}$
  - Precision = $\frac{TP}{TP+FP}$
  - True Positive Rate = $\frac{TP}{TP+FN}$
  - False Positive Rate = $\frac{FP}{FP+TN}$ (most important)
- Acceptable FP rate: learn a ranker

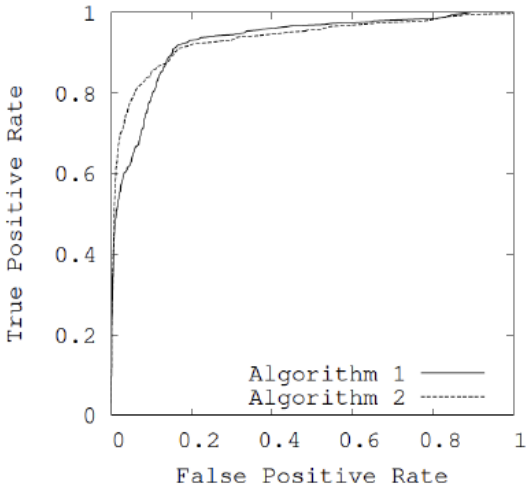
  Steps:

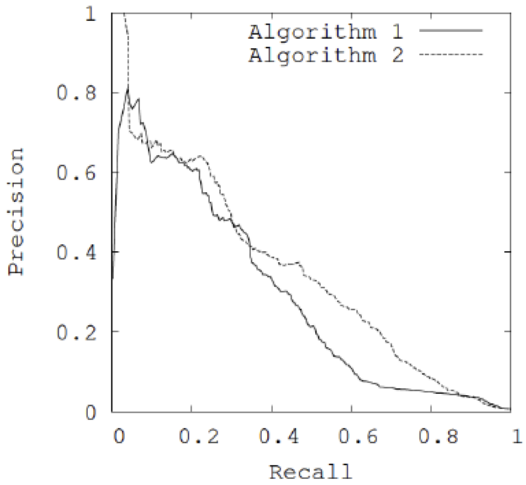
  1. Instead of labeling true/false, assign a value
  2. Sort item by rank
  3. Determine a threshold: over threshold->positive, below->negative
  4. For all thresholds, plot an ROC(Receiver Operating Characteristic, false positive rate on x-axis versus the true positive rate on y-axis) and PR(Precision-Recall) curve
  5. Determine the right thredhold



rank	1	2	3	4	5	6	7	8	9	10
label	-	-	+	-	-	+	+	-	+	+

- Which threshold to use?
  - FPR = 3/5 1/5 0/5
  - TPR = 5/5 4/5 2/5

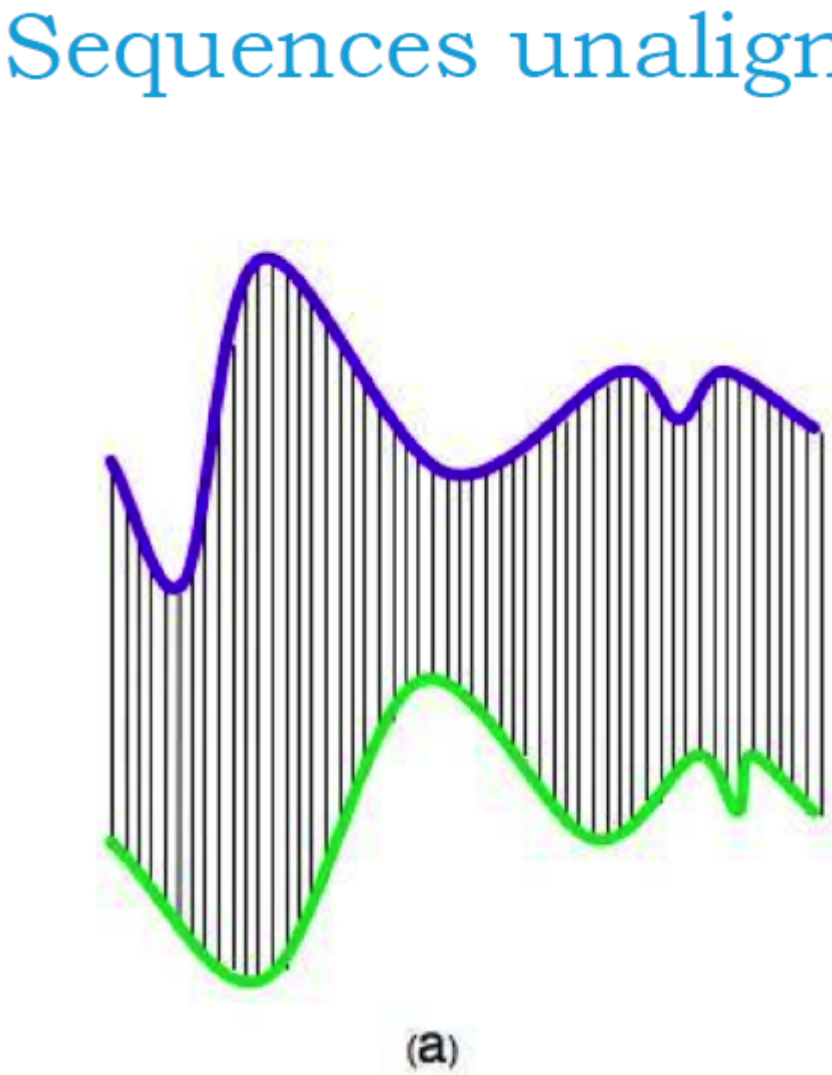




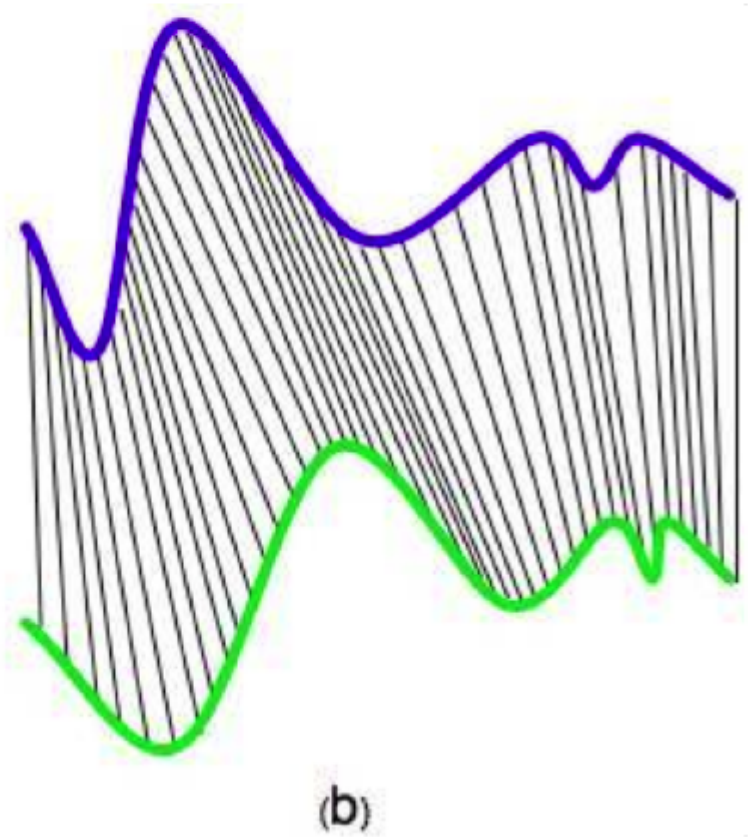
Sequential data

Problem: sequences are unaligned

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



Sequence alignment



Data is massive and continuous

Problem: Machine learning uses counts such as statistics and distances and compress data into a model. Yet as data is massive and keeping coming in, the problem that how to throw away information smartly arises.

Simple solution: count approximately with hash functions

Advanced solution: apply distributed data processing

Problem statement

A set W containing m values, with memory of size n

Goal: data structure that allows efficient checking whether an element in the stream is in W

- return TRUE *with probability 1* if it is indeed in W
- return FALSE *with high probability* if it is not in W

Bloom filter

Given:

- a set of hash functions $h_1, h_2, \ldots, h_k : W \to [1, n]$
- a bit vector of size n (initialized to 0)

Steps:

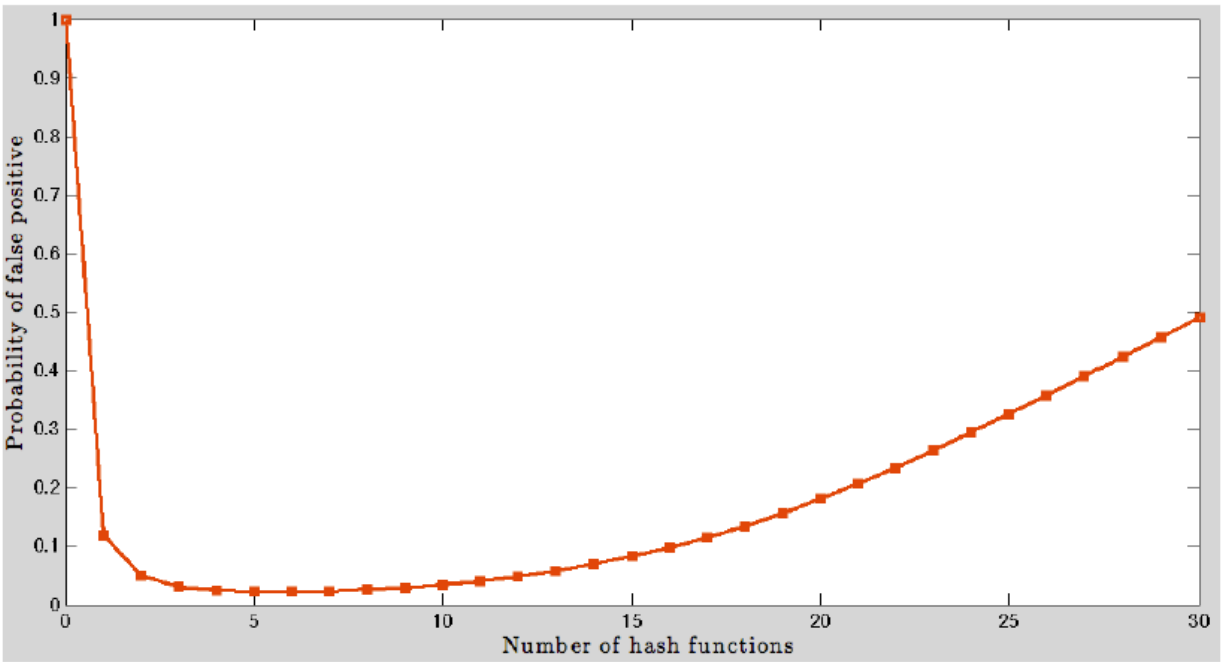
- add a string: update the bits at the index given by hashes to 1
- test a membership: hash the string with the same hash functions.

  If values are set to 1, the element *might be* in the set.

  If are not, the element is not in the set.

Optimal number of hashes

Example: $m = 10^9$ whitelisted IP addresses and $n = 8 \times 10^9$ bits in memory



Dealing with opponents

Problem: For a malware X, we learn a distribution D_X over X's data by Markov chain, n-gram, naïve Bayes. What if we have a new data sample S?

Solution: compute conditional probability $P(S|D_X)$

- assume maximum-likelihood model
- detect same behavior if P > threshold

N-grams

- A popular tool in natural language processing (NLP)
- Also known as bigrams, trigrams, higher-order Markov chains, shingles
- To avoid overfitting, they almost always need to be combined with *smoothing methods*, such as adding a count of 1 to all occurrences (Laplace)
- Simple to use, but the size can blow-up when the alphabet gets large (e.g., all the words in a language)
    - could be solved by HASH
- Used to detect malware using Byte N-grams

Privacy issues

Problem: Some published dataset removed some attributes related privacy issues, but there is still linkage problem between datasets that people could infer information

Solution:

- Compute anonymity set:
    - The size of the set of people with identical values for quantities of interest
    - Ensure the size of anonymity set is large enough, then publish
- Use artificial data: generate synthetic data from a probability distribution

Lecture 2: Imbalanced Data

Lecture 3: Sequential Data

Sequential data 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

Problems of applying standard machine learning methods:

- do not exploit temporal patterns
- Computation & storage scales poorly to realistic applications

Applying standard machine learning methods

Time slicing

Step:

- **Group** time series data based on their time value (e.g. Jan 2015, Feb 2015, Mar 2015).
- Learn a model for each group

There are multiple ways to group data. One way is to group only by time value. Other way could be grouping by time and id.

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

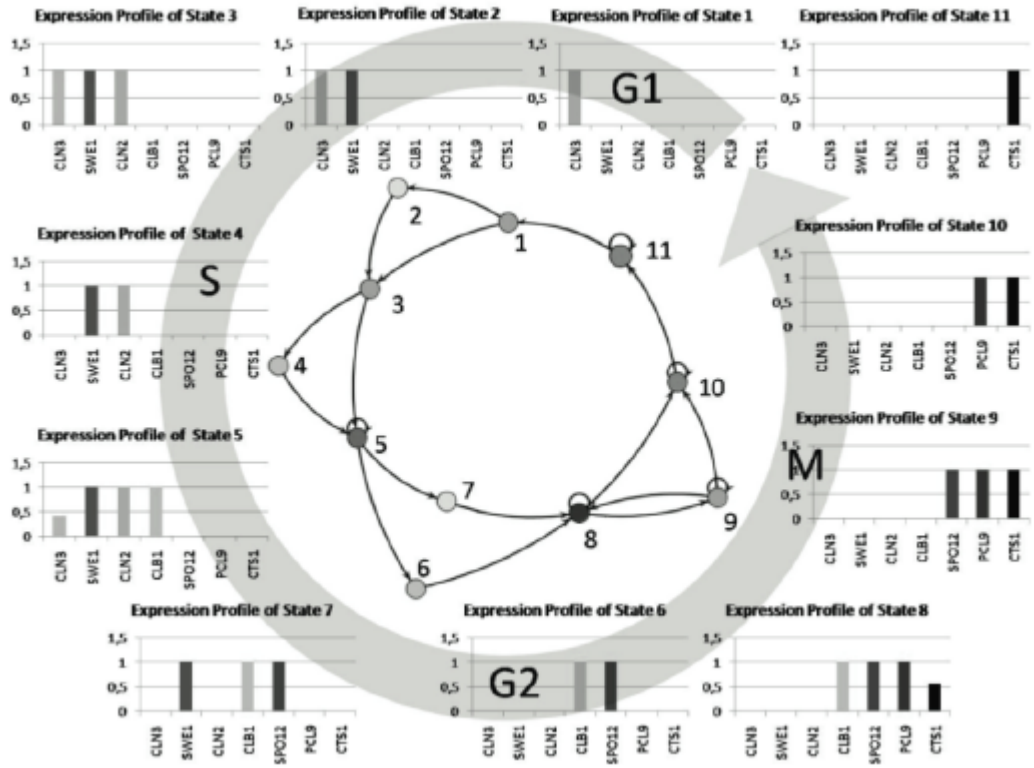
Time slicing – process dependent

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

notes:

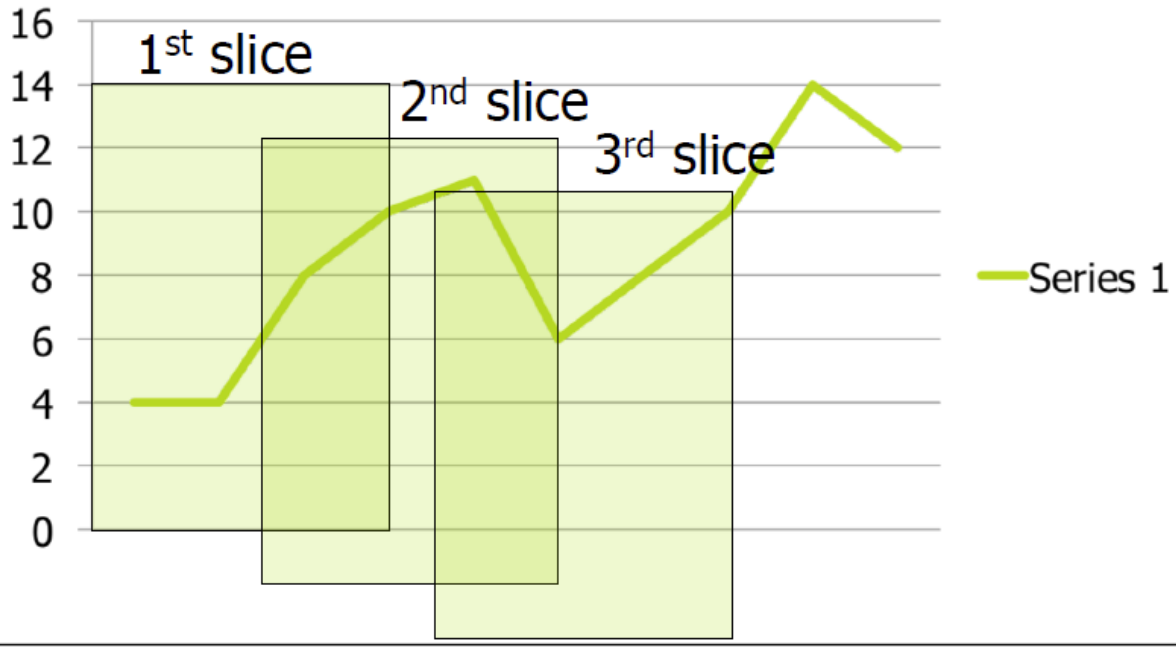
- Often applied in patient modeling (e-health)
- How to slice depends entirely on the application
- Smarter methods exists if you can determine the “state” of a system:
    - Slice value equal to state number
    - State is discovered using sequential models



Sliding windows

Idea: move a fixed size window along sequence data. Each window will be regarded as one slice. Then regard each slice as one unit and compute features for each slice to learn temporal features.

- Slice the data dependent on the current time point:



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

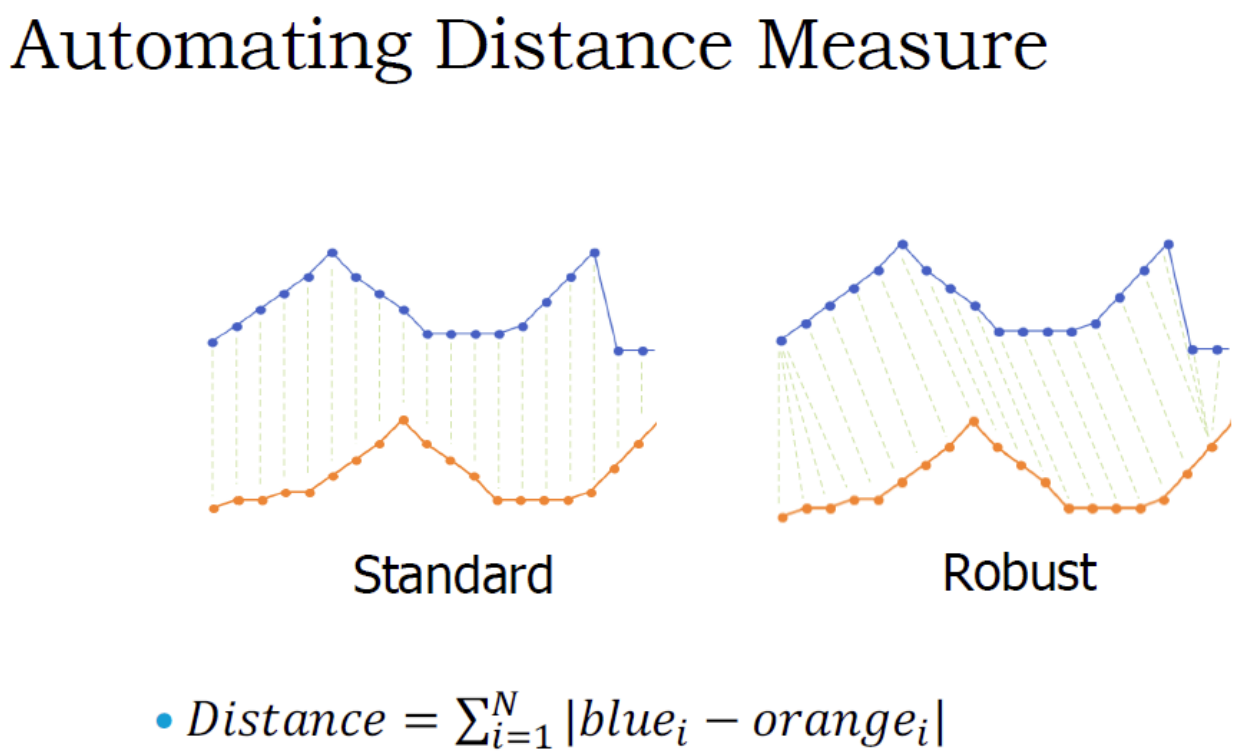
Notes:

- Usually applied to time-series data
  - Measurements at a fixed temporal rate e.g. 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

Computing distances between series data

Problems: how to compute distance between shapes? 1. Some sequence have similar shapes in y-axis but they are not aligned on x-axis. They ought to have small distance between each other but it is impossible if we compute point-to-point distance and then sum up. 2. Some sequence have similar shapes in y-axis but one is scaled on x-axis



- $Distance = \sum_{i=1}^{N} |blue_i - orange_i|$

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

Dynamic Time Warping

- Time Warp => distortions in time axis
- Calculate distance between two curves in the presence of distortions (warps) in the time axis.

Idea: Extract the best mapping that minimizes a flexible distance

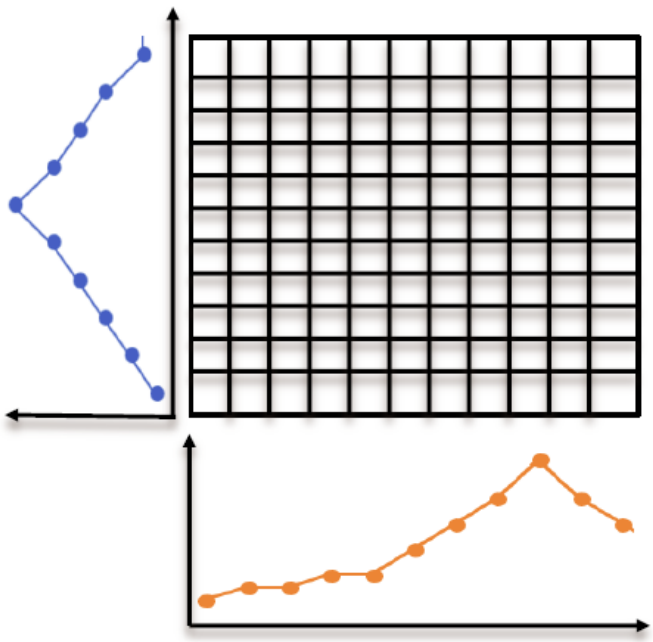
$D(i,j) = |A_i - B_j| + min(D(i-1,j), D(i,j-1), D(i-1,j-1))$

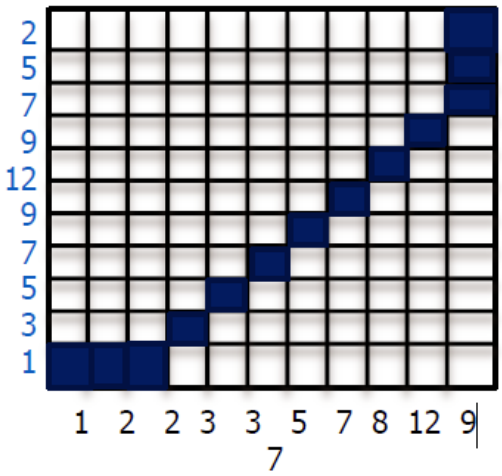
similar with dynamic programming

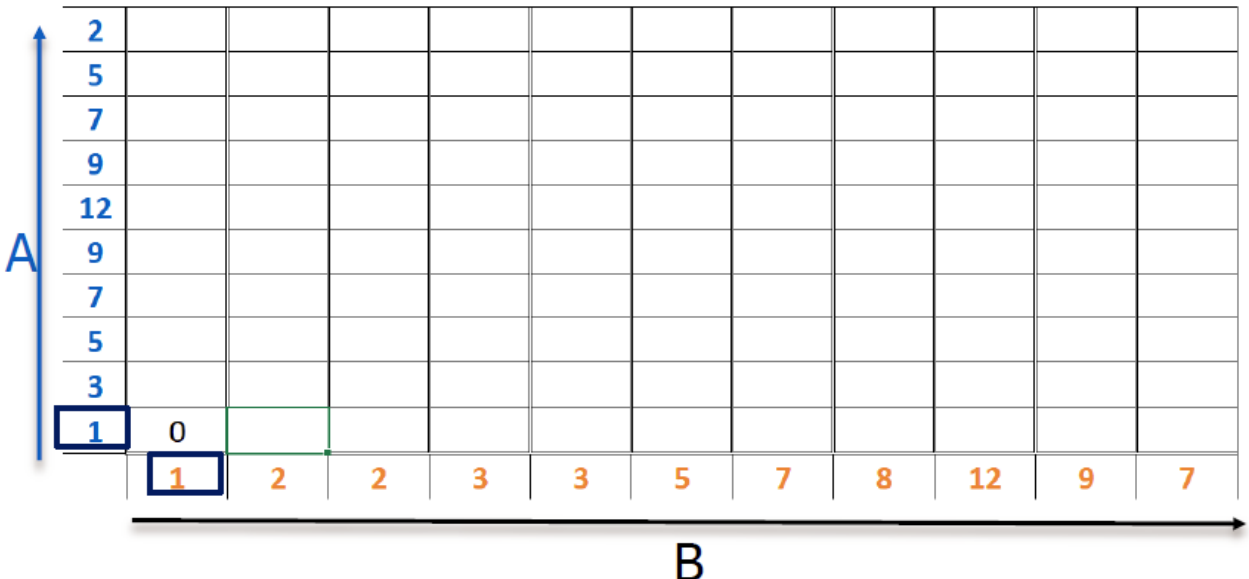
Steps:

- matrix calculation -- forward pass
- find path -- backward pass

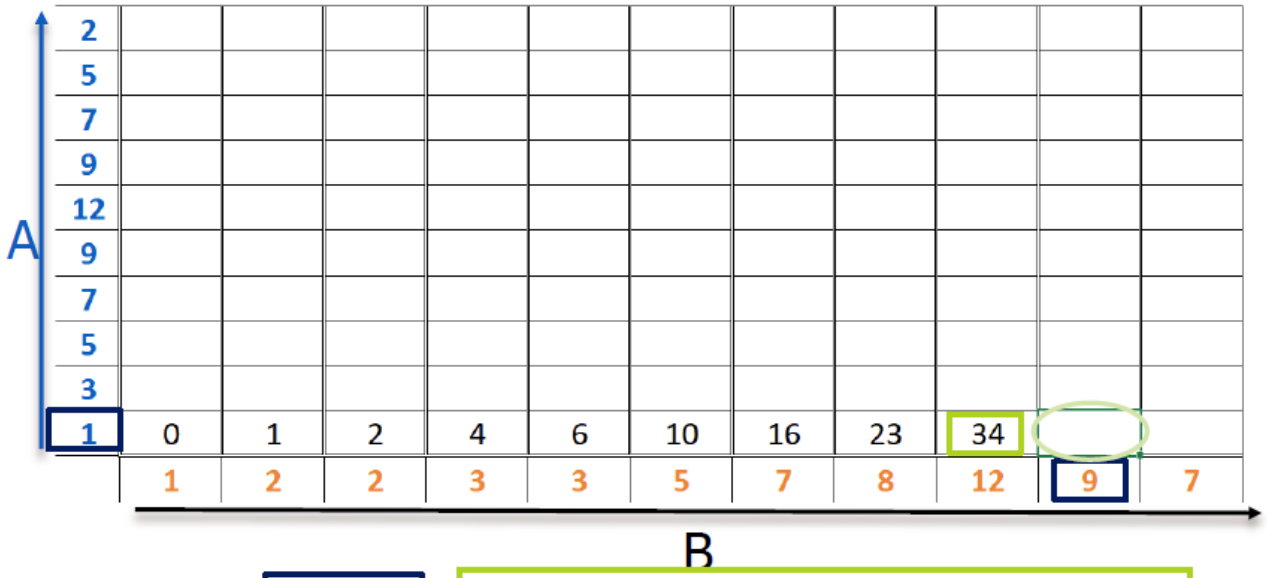
Matrix calculation



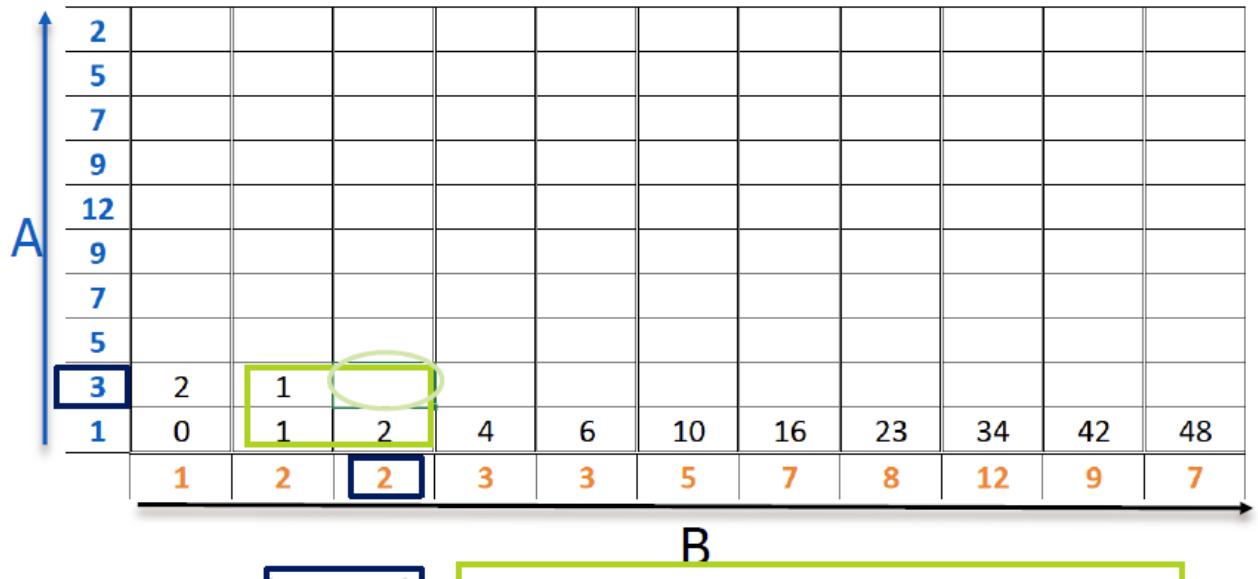




$$D(i,j) = \boxed{|A_i - B_j|} + \min(D(i-1,j), D(i,j-1), D(i-1,j-1))$$



$$D(i,j) = |A_i - B_j| + \min(D(i-1,j), D(i,j-1), D(i-1,j-1))$$



$$D(i,j) = |A_i - B_j| + \min(D(i-1,j), D(i,j-1), D(i-1,j-1))$$

A	B										
2	50	41	41	36	36	24	18	18	22	16	10
5	49	41	41	35	35	21	13	12	16	9	5
7	45	38	38	33	33	21	11	9	11	5	3
9	39	33	33	29	29	19	11	8	6	3	5
12	31	26	26	23	23	15	9	7	3	6	11
9	20	16	16	14	14	8	4	3	6	6	8
7	12	9	9	8	8	4	2	3	8	10	10
5	6	4	4	4	4	2	4	7	14	18	20
3	2	1	2	2	2	4	8	13	22	28	32
1	0	1	2	4	6	10	16	23	34	42	48
	1	2	2	3	3	5	7	8	12	9	7

$$D(i,j) = |A_i - B_j| + \min(D(i-1,j), D(i,j-1), D(i-1,j-1))$$

Path finding

A	B										
2	50	41	41	36	36	24	18	18	22	16	10
5	49	41	41	35	35	21	13	12	16	9	5
7	45	38	38	33	33	21	11	9	11	5	3
9	39	33	33	29	29	19	11	8	6	3	5
12	31	26	26	23	23	15	9	7	3	6	11
9	20	16	16	14	14	8	4	3	6	6	8
7	12	9	9	8	8	4	2	3	8	10	10
5	6	4	4	4	4	2	4	7	14	18	20
3	2	1	2	2	2	4	8	13	22	28	32
1	0	1	2	4	6	10	16	23	34	42	48
	1	2	2	3	3	5	7	8	12	9	7

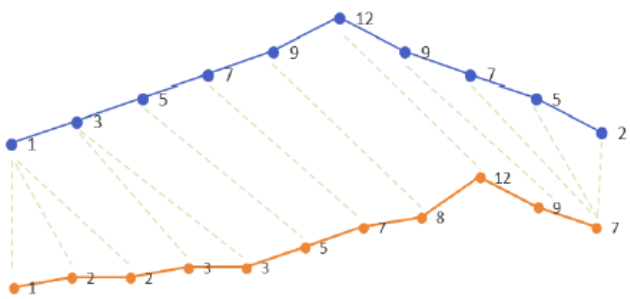
Path finding

A											
2	50	41	41	36	36	24	18	18	22	16	10
5	49	41	41	35	35	21	13	12	16	9	5
7	45	38	38	33	33	21	11	9	11	5	3
9	39	33	33	29	29	19	11	8	6	3	5
12	31	26	26	23	23	15	9	7	3	6	11
9	20	16	16	14	14	8	4	3	6	6	8
7	12	9	9	8	8	4	2	3	8	10	10
5	6	4	4	4	4	2	4	7	14	18	20
3	2	1	2	2	2	4	8	13	22	28	32
1	0	1	2	4	6	10	16	23	34	42	48
	1	2	2	3	3	5	7	8	12	9	7

B

Path finding

2	50	41	41	36	36	24	18	18	22	16	10
5	49	41	41	35	35	21	13	12	16	9	5
7	45	38	38	33	33	21	11	9	11	5	3
9	39	33	33	29	29	19	11	8	6	3	5
12	31	26	26	23	23	15	9	7	3	6	11
9	20	16	16	14	14	8	4	3	6	6	8
7	12	9	9	8	8	4	2	3	8	10	10
5	6	4	4	4	4	2	4	7	14	18	20
3	2	1	2	2	2	4	8	13	22	28	32
1	0	1	2	4	6	10	16	23	34	42	48
	1	2	2	3	3	5	7	8	12	9	7



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
  - difference: DMT: Find minimum distance (different time data could be projected to same point)
  - sequence alignment: each sequence is aligned one by one (这里是两项，上面是三项)
- 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

Summary:

- 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

Modelling time-series

Time series prediction

- Goal: predict t_n data based on $t_1, t_2, \ldots t_{n-1}$
- 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

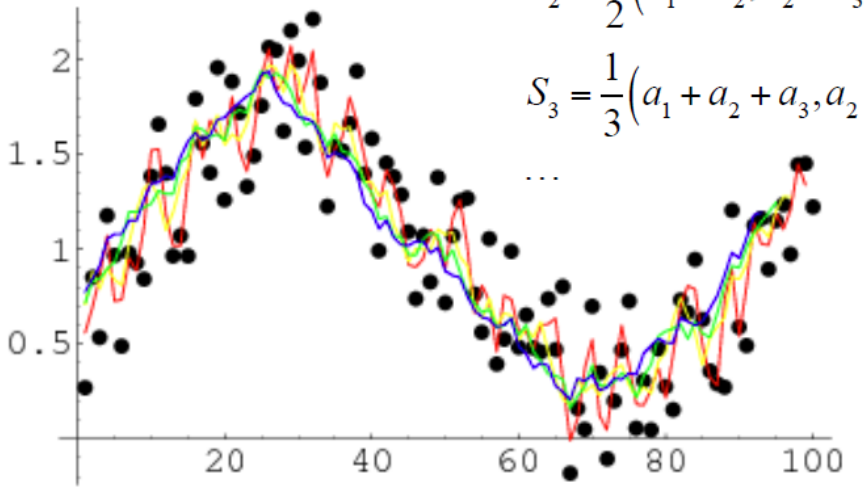
Moving average

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

Large n means more smooth!

Moving average model

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

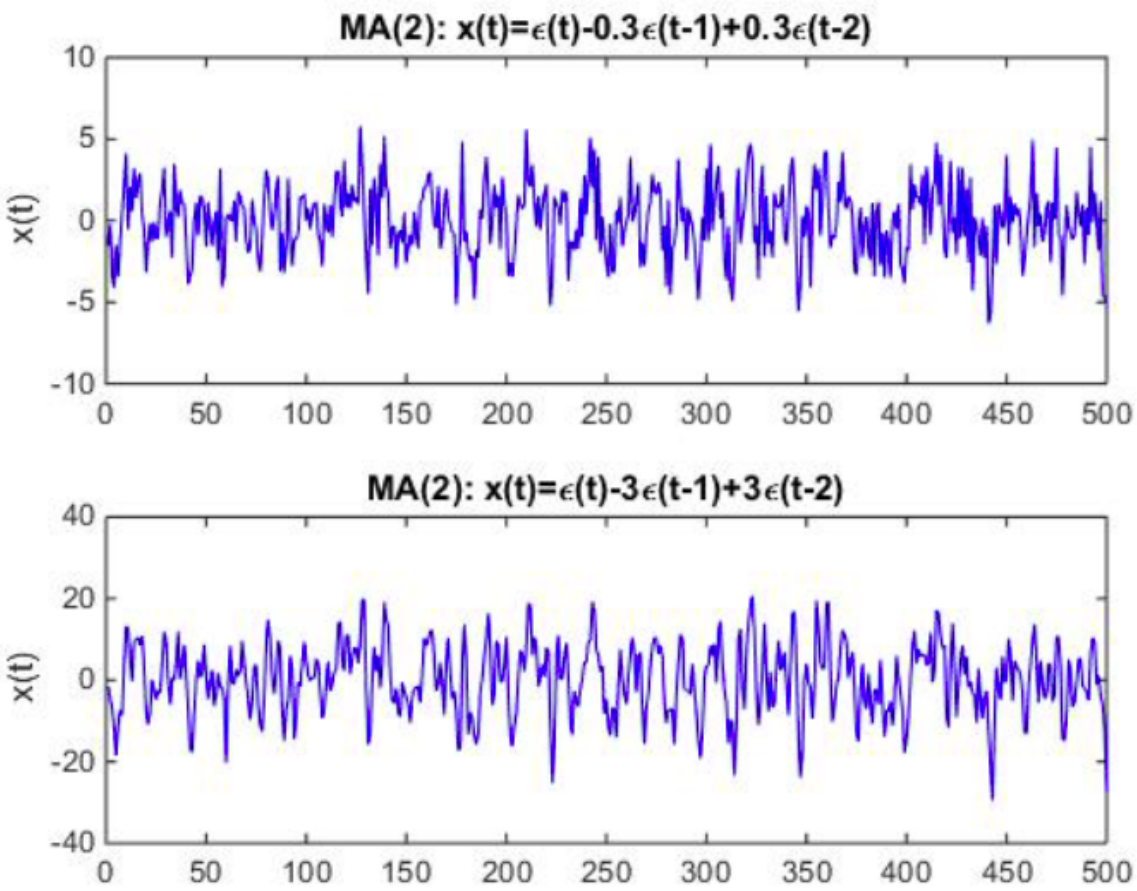
- The notation $MA(q)$ refers to the moving average model of order q. The $MA(q)$ model is written as: $X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-1}$

  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

Example: $MA(q)$ for different θ

With θ increases, amplitude also increases, resulting in high variance.

MA(q) for q=2 and different θ



Auto-Regressive

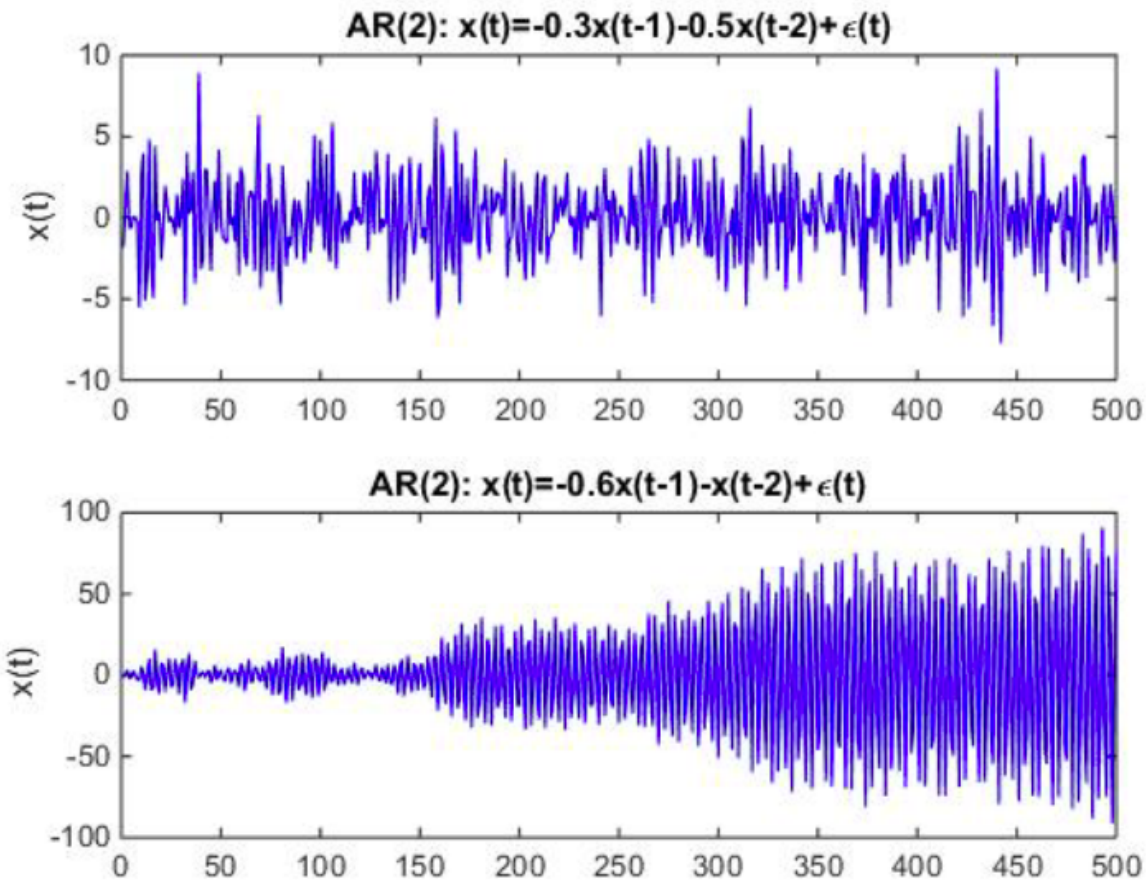
- The notation $AR(p)$ refers to the autoregressive model of order p. The $AR(p)$ model is written as: $X_t = \sum_{i=1}^{p} \phi_i X_{t-1} + \epsilon_t$

  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

Example: $AR(p)$ for different ϕ

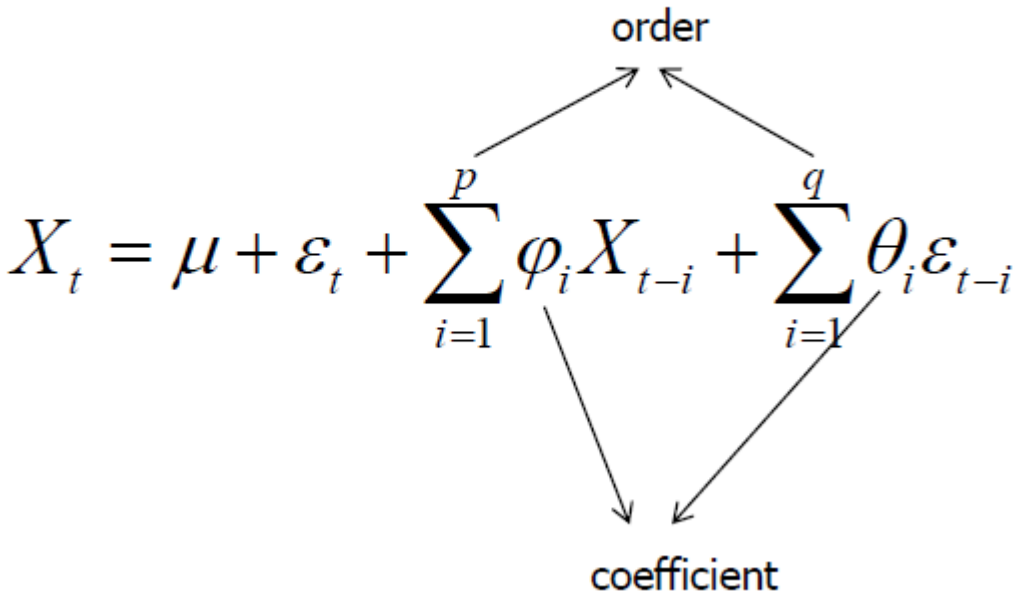
With ϕ increases, fluctuation also increases

AR(p) for p=2 and different φ

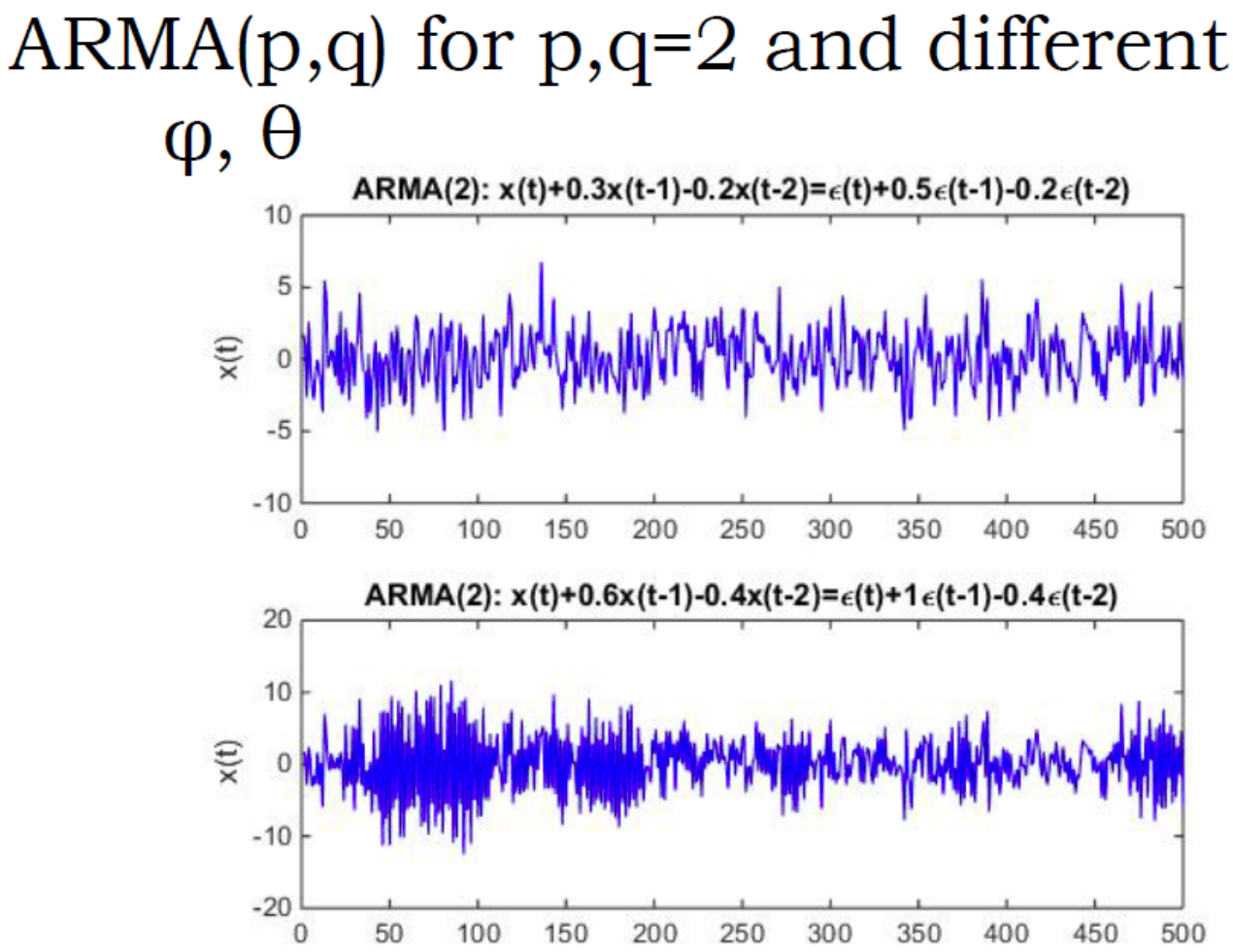


Auto-Regressive Moving Average (ARMA)

ARMA (p,q) is written as:



Example:



Notes:

- one of the most frequently used models for timeseries prediction
- Model gets “reset” after every prediction
- Removes the effect we get when generating new series

In conclusion:

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

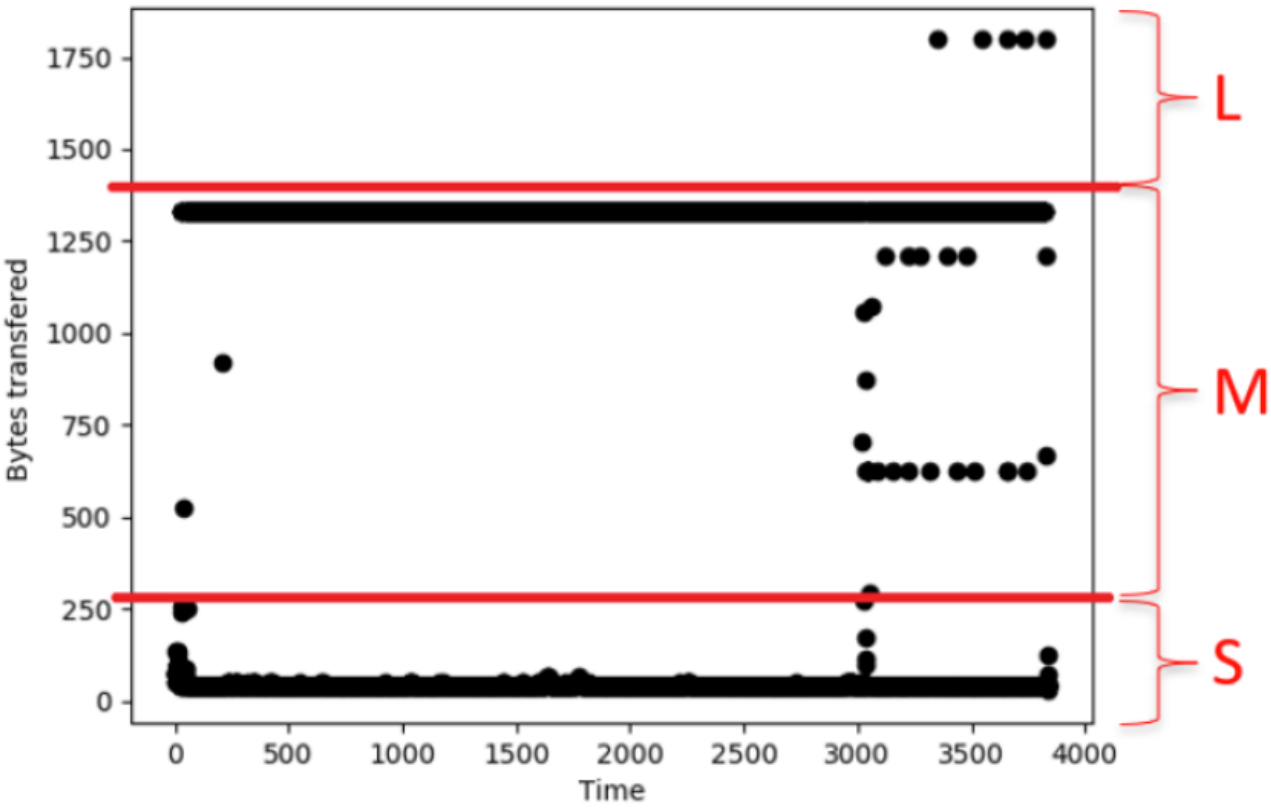
- Time series prediction:
  - Predict t+1 using t-N…t
- Moving average:
  - Model free, simply average the past x values (or noise terms)
  - Possibly add exponential weights
  - In an *MA model*, the weights are learned from data (hard)
- Auto regressive:
  - Takes the previous values as input, model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

Discrete models

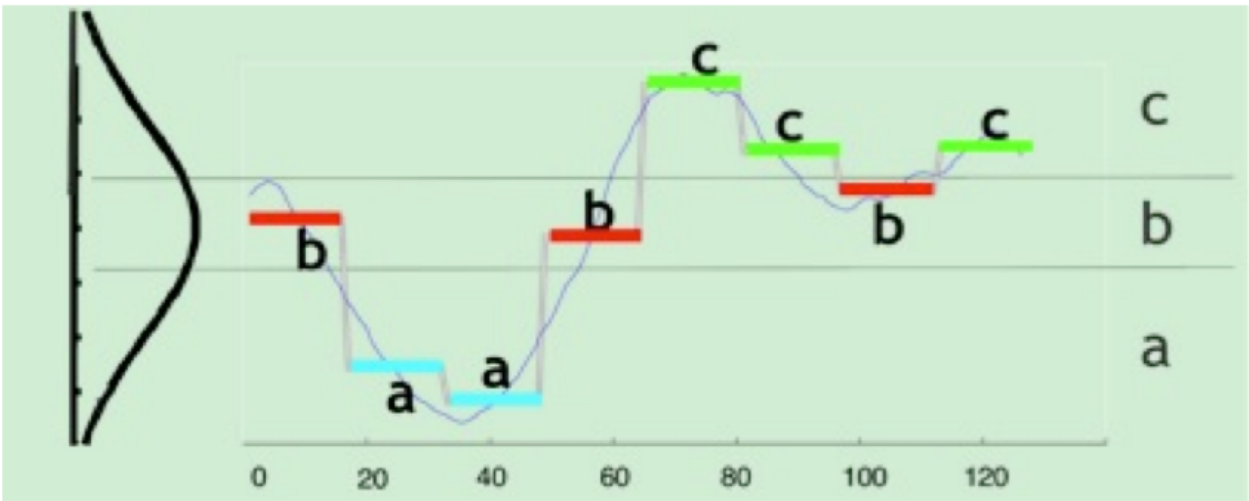
Discretization

transferring continuous functions, models, variables, and equations into discrete counterparts

Percentiles: way to make data symbolic:

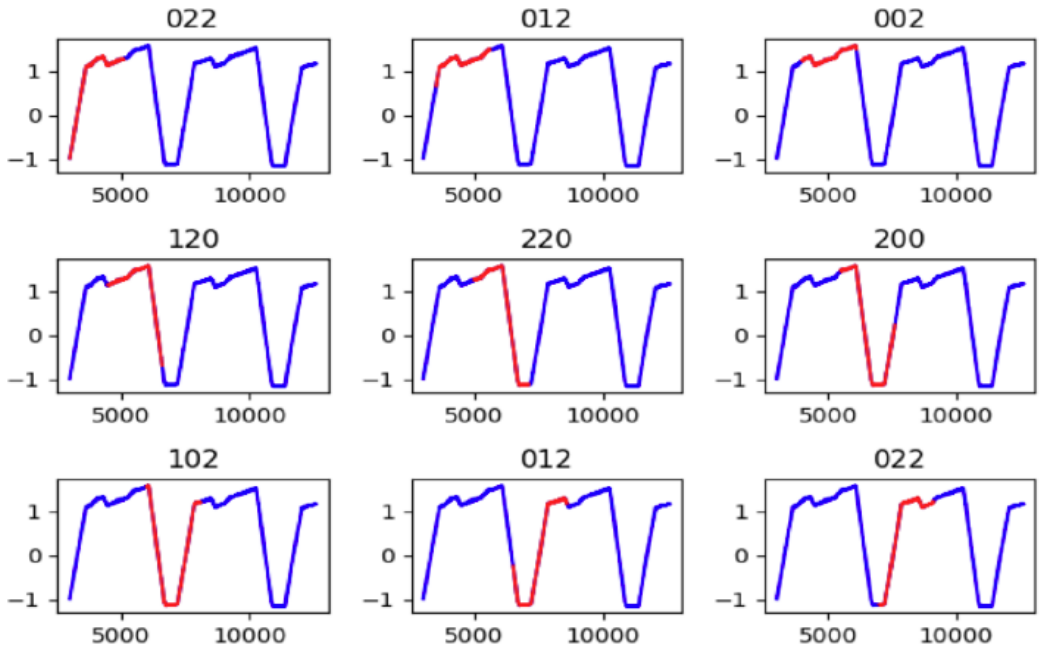


Symbolic approximation of time series



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

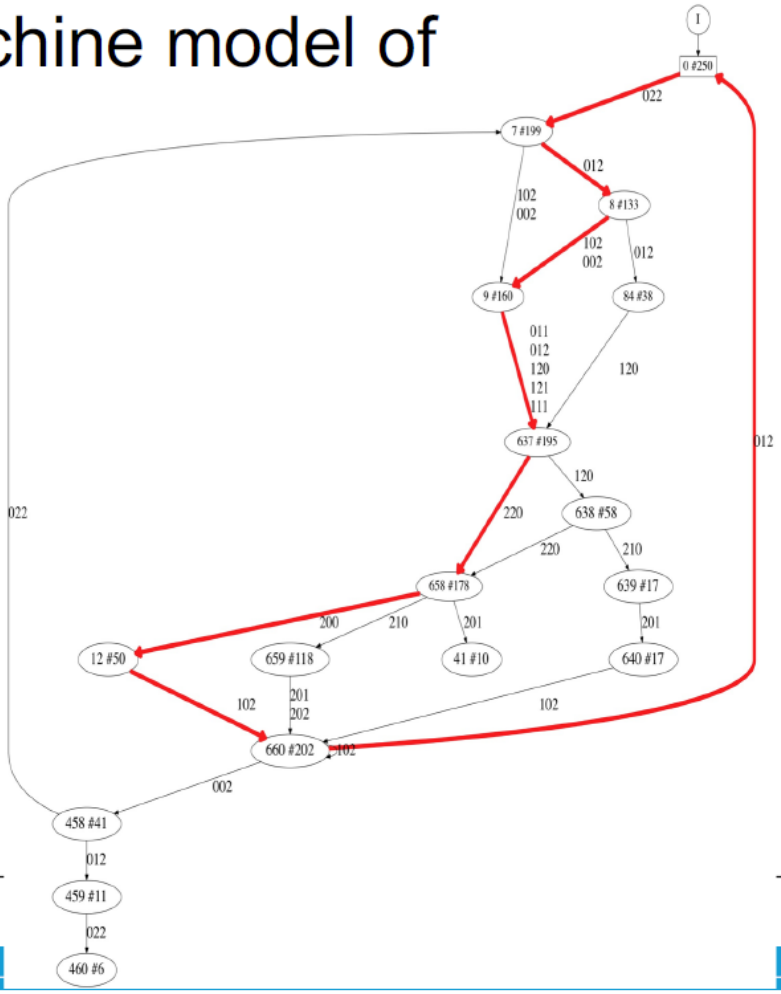
SAX (Symbolic Aggregate approXimation)



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

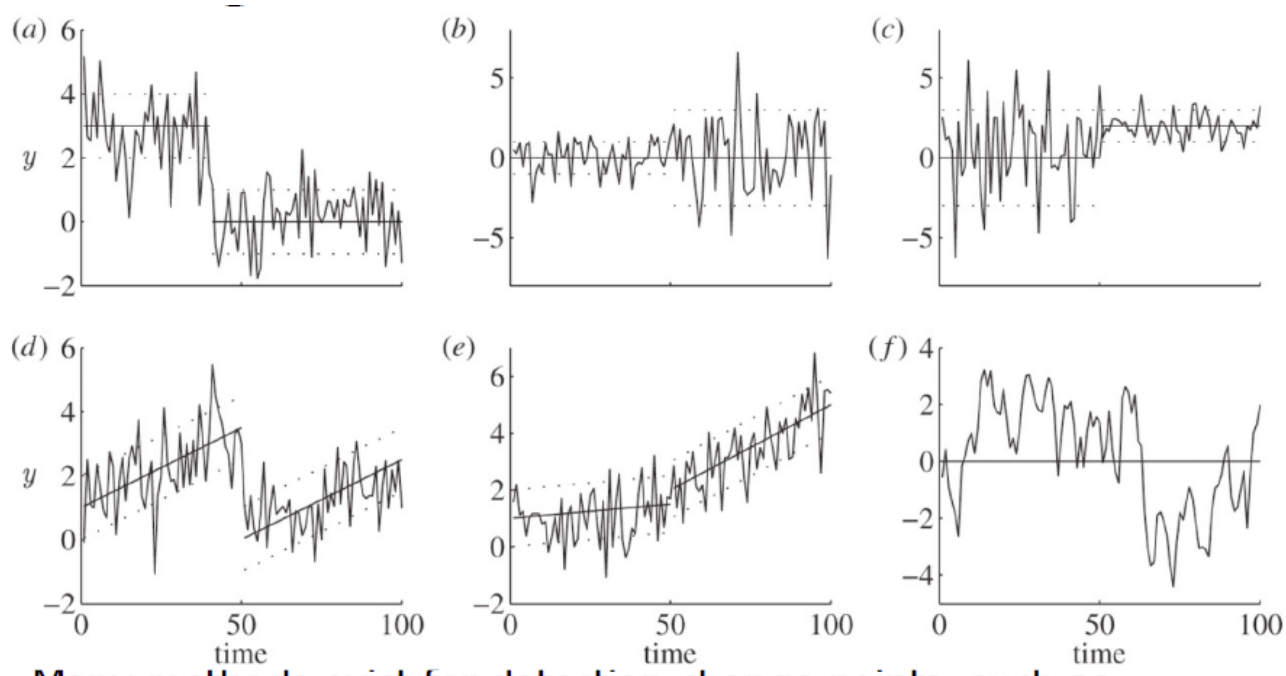
Build state machine model of signal

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



TU Delft

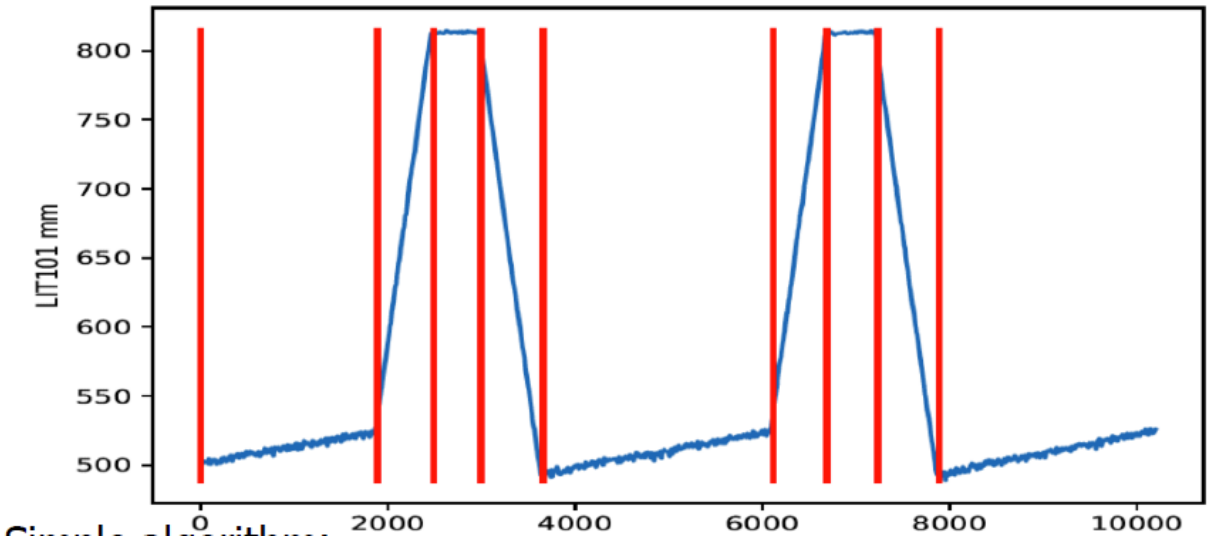
Change point detection and segmentation



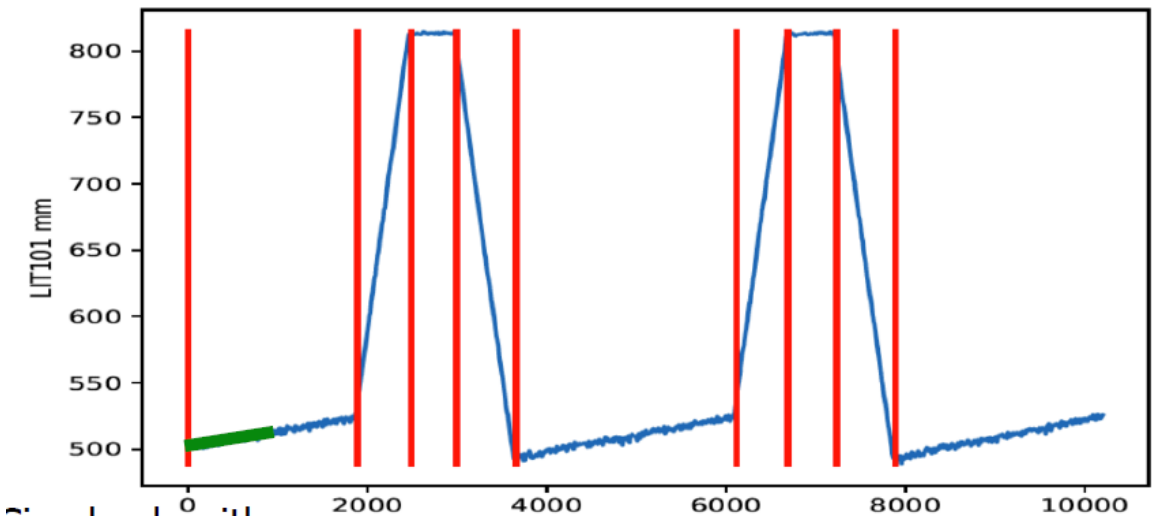
- Many methods exist for detection change points, such as abrupt trend or distribution changes

Simple algorithm:

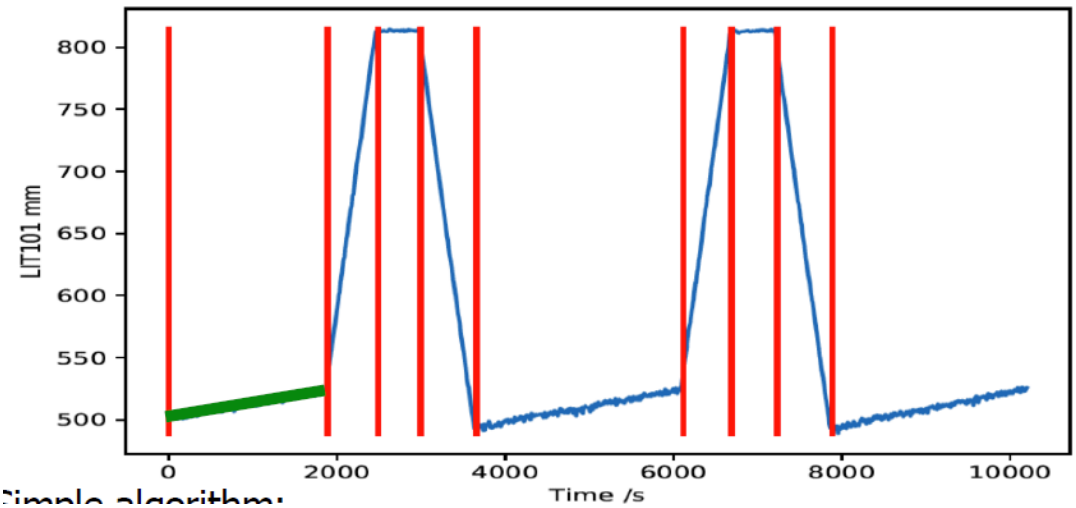
1. Add next point to linear segment

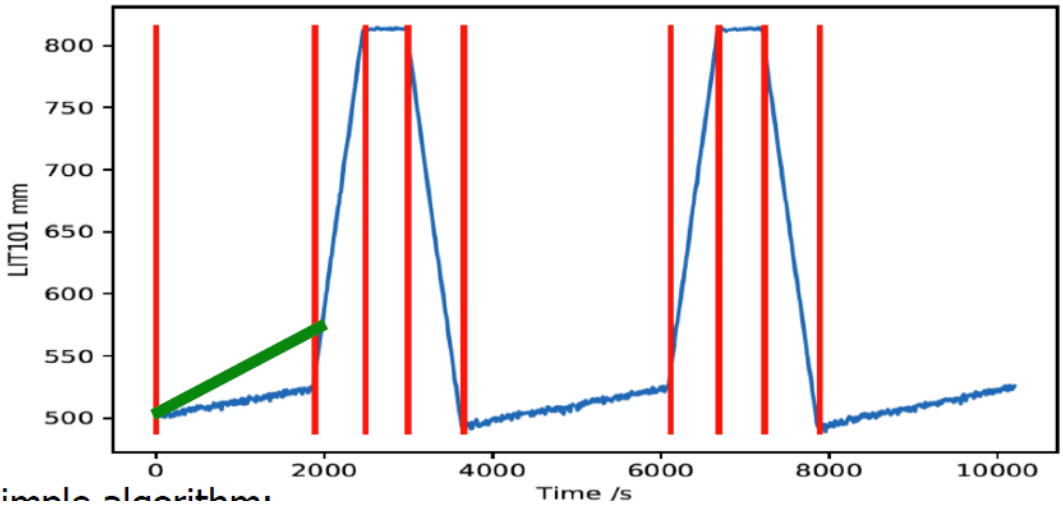


2. Compute regression error e



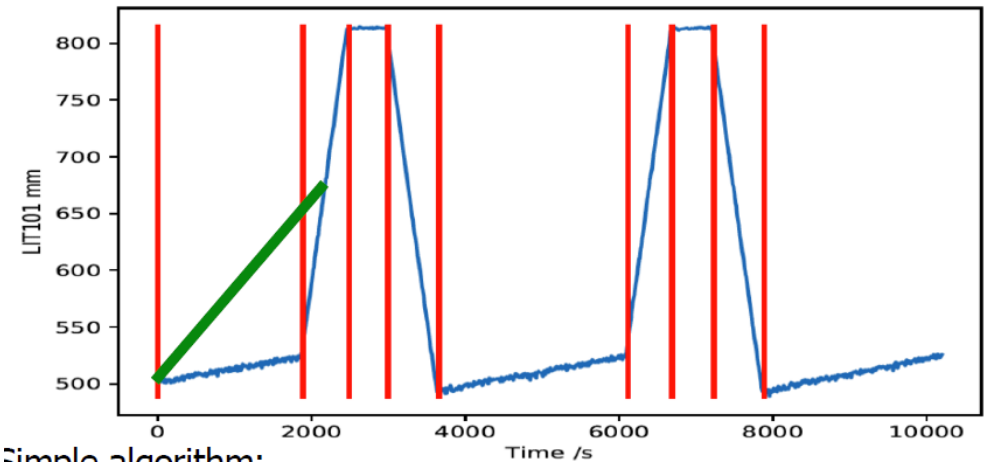
3. If e < bound, goto step 1



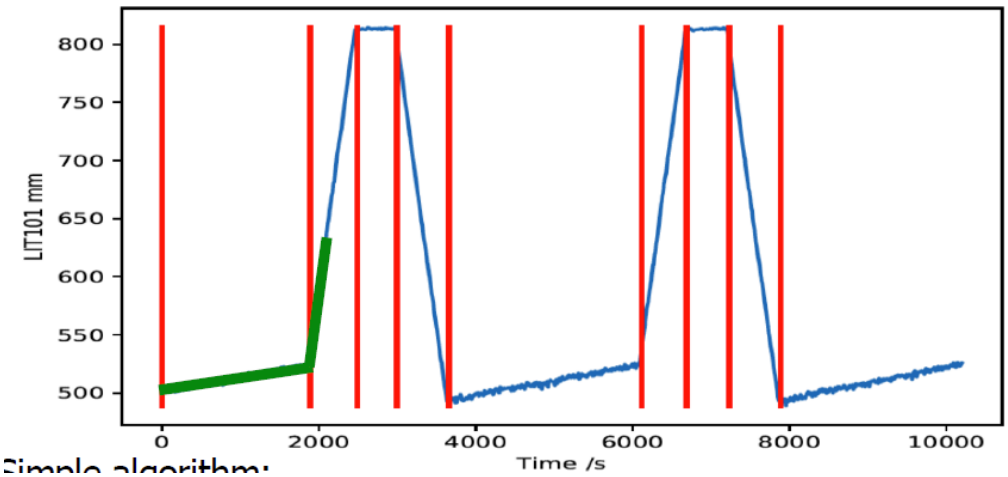


Simple algorithm:

4. else cut at point of largest error, continue from there



Simple algorithm:

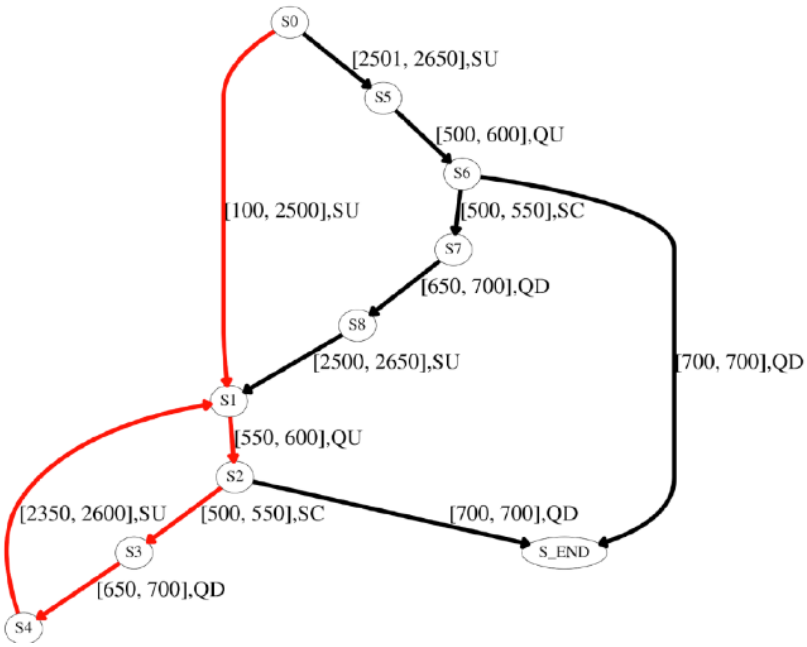


Simple algorithm:

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?
- Suggestion:
  - 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

  e.g. $x_t \in 1, 2, \ldots, N$ = 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 chains

Markov property: Conditioned on the present, the future is independent from the past

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

State Transition Matrices:

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

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

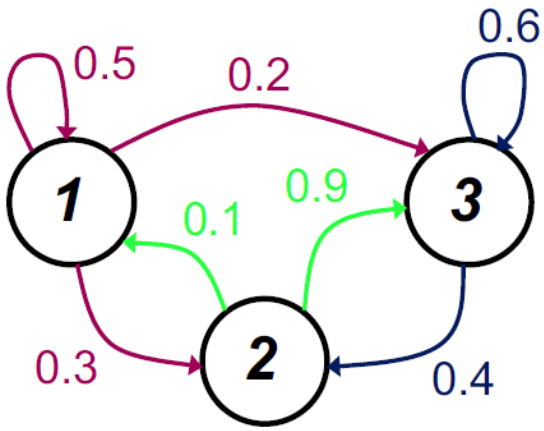
$$q_{ij} \triangleq p(x_{t+1} = i \mid x_t = j)$$

- 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

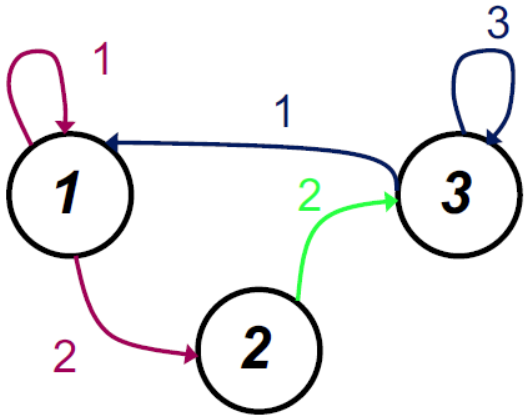
$$Q = \begin{bmatrix} 0.5 & 0.1 & 0.0 \\ 0.3 & 0.0 & 0.4 \\ 0.2 & 0.9 & 0.6 \end{bmatrix}$$



Step1: Get state transition matrices: how to estimate the parameter(probability)?

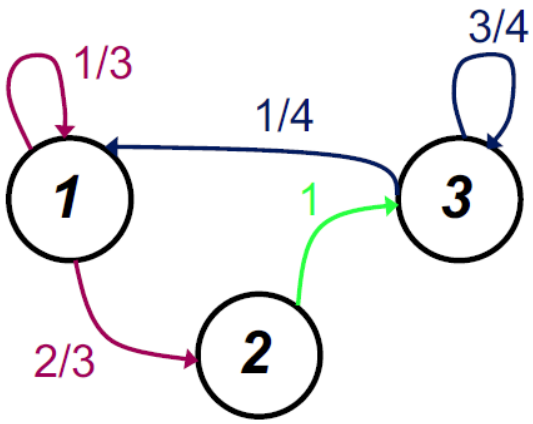
1. Count transition occurrences
2. Normalize transition counts to sum up to 1.0

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

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

Step2: Predict new state sequence:

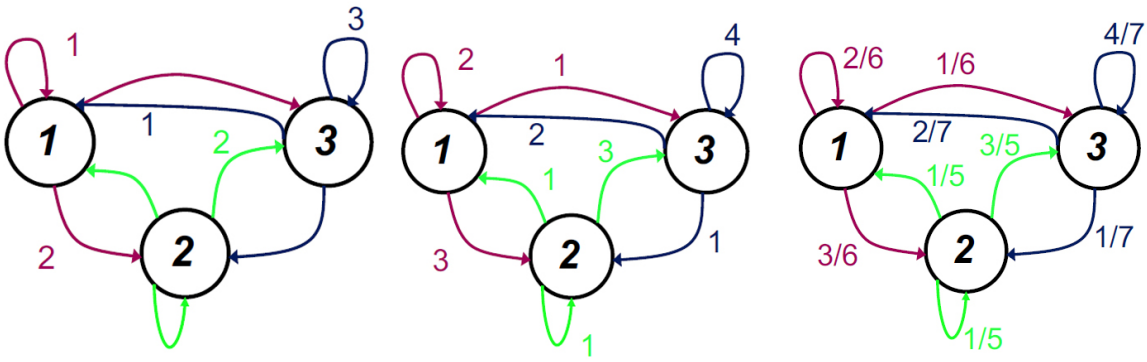
$p(x_0, x_1, \ldots, x_T) = p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1})$

e.g. $Prob(1123) = P(1) * P(1 \to 1) * P(1 \to 2) * P(2 \to 2)$

Step3: Avoid zero probabilities

- 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



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

- 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

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

- 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

In conclusion

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

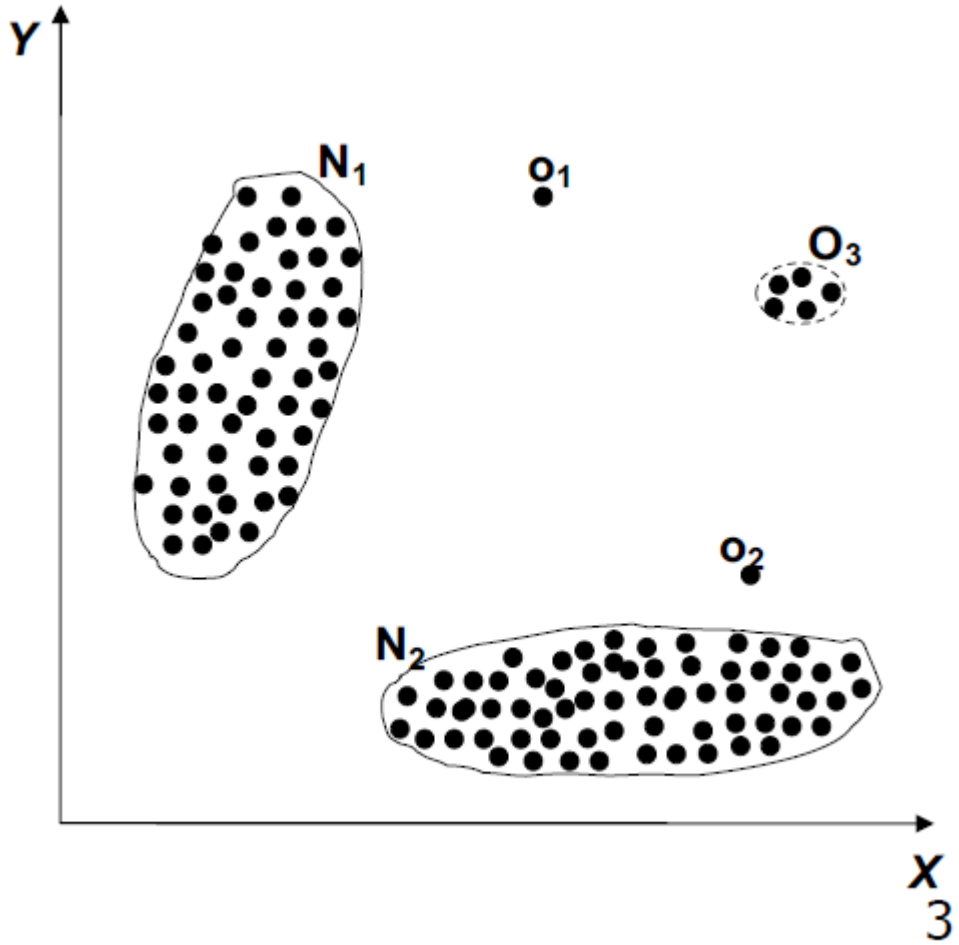
Lecture 4: Anomaly Detection

Anomaly is a pattern in the data that does not conform to the expected/normal behavior

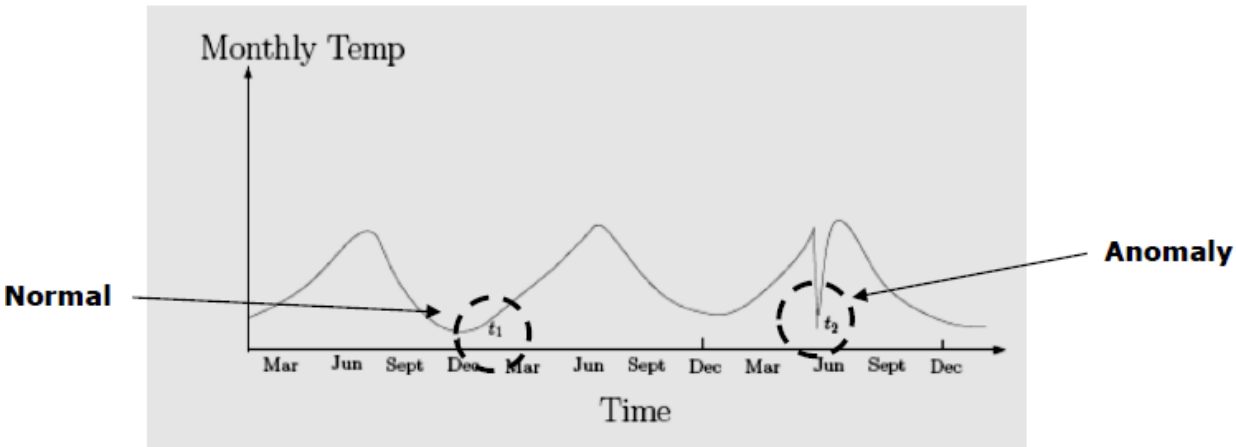
- Also called outliers, exceptions, peculiarities or surprise.
- Anomalies translate to significant (often critical) real life entities:
  - Cyber intrusions
  - Credit card fraud
  - Equipment malfunction

Types of anomalies

- Point Anomalies: An individual data instance is anomalous with regard to the data
  Answer may differ from people

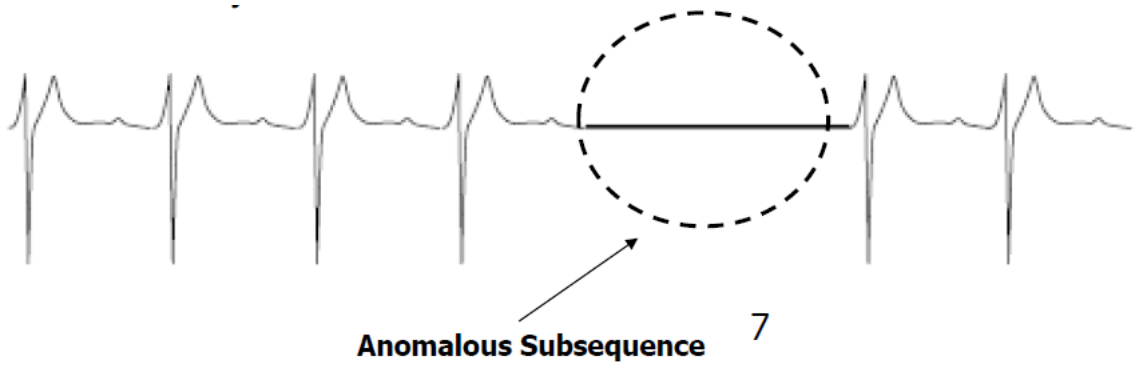


- Contextual Anomalies(conditional anomalies):
  - An individual data instance is anomalous within a context
  - Requires a notion of context
  - e.g. t_2 is contextual anomaly, because it differs from periodic context



- Collective Anomalies
  - A collection of related data instances is anomalous (whole sequence is anomaly)
  - Requires a relationship among data instances
    - Sequential Data, Spatial Data, Graph Data

- The individual instances within a collective anomaly are not anomalous by themselves



Detecting anomalies

Detecting in sequences

Compute the residual (differences between observed and predicted values of data)

steps:

step1: Use training data to learn a model

- Use sliding windows
  - translate to standard point anomaly detection
  - compute distances using sequence alignment
- Learn a sequential model (e.g. n-gram), and
  - compute the probability of observing a sequence
  - if below a threshold, raise an alarm

step2: Use past test data to make a one-step prediction: $y_k = f(y_{k-1} + \epsilon)$

step3: Compute the residual: $y_{k|k-1} = f(y_{k-1})$

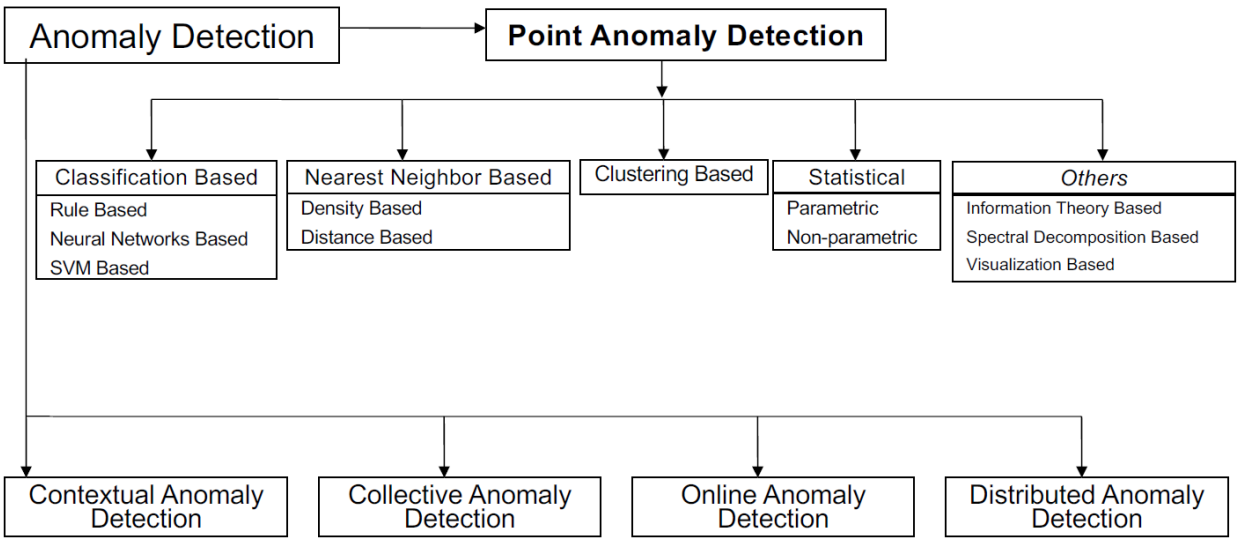
step4: Evaluate the residual error through statistical test (depends on noise assumptions)

$r_k = y_k - yk|k-1$

step5: Use a threshold to decide anomaly, typical

- 2 or 3 times the standard error
- or simply sort on residual error and return largest ones

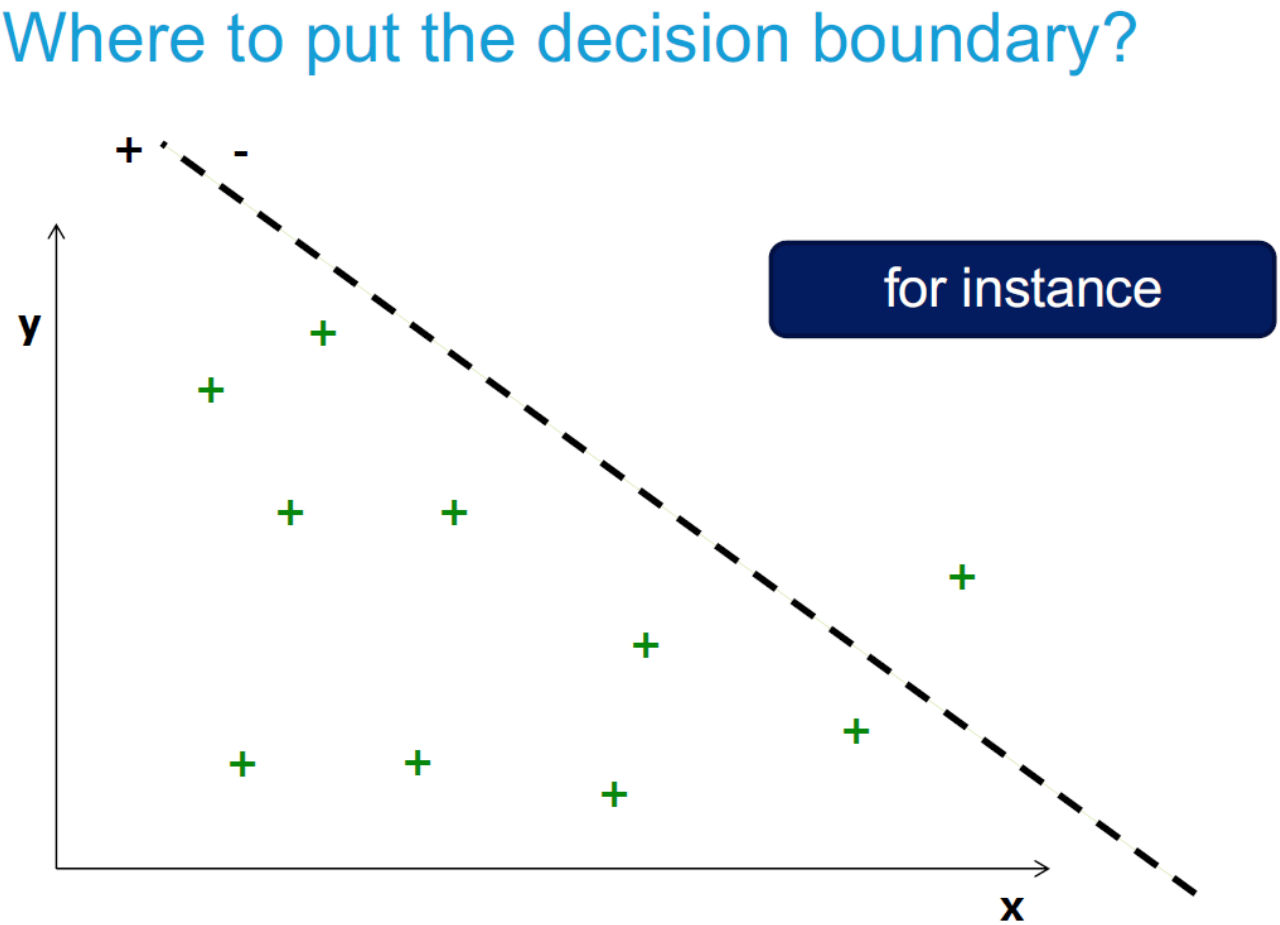
Detecting in multivariate data sets/non-sequential

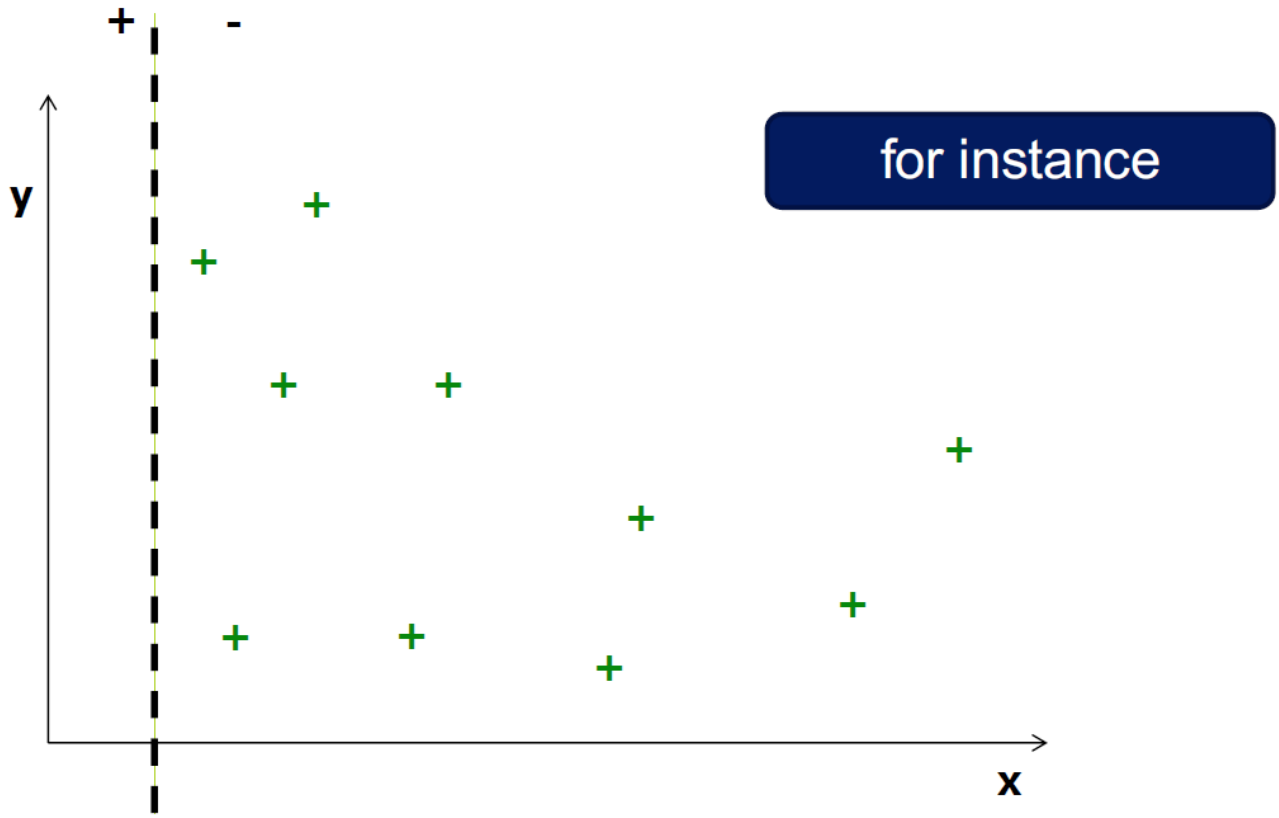


Classification based methods

OSVM: one-class support vector machine

example:





- supervised learning
- Goal: separate positive data from remaining input space
- Possible assumptions:
  - Further away from origin is anomalous (one-class SVM)
  - Close to the origin is anomalous (different one-class SVM)
  - Close to origin is also anomalous (improved one-class SVM)
  - Further from centroid is anomalous (non-linear one-class SVM)
- Key ingredient
  - Maximize negative/outlier space
  - Minimize positive/normal space

Isolation Forest

- Procedure: Repeat N times:

1. Randomly pick a feature f
2. Split the f uniformly at randomly between [min,max]
3. Continue until all leafs contain singletons

- The path length to reach a leaf is the isolation score
- Average this length over all trees to get the anomaly score
- Intuition: isolating anomalies is easier because only a few conditions are needed to separate those cases from the normal observations

  In short, anomaly gets lower score

Classification Based Techniques

- Advantage
  - Can be used in unsupervised setting
  - Models can be (easily) understood
  - Computationally inexpensive when testing
- Drawback
  - Make assumptions about data distribution
    - Where is the origin? Is it normal or anomalous?
    - Intuitively less appealing

Nearest Neighbor Based Techniques

- Key assumption: normal points have close neighbors while anomalies are located far from other points
- Two-step approach
  1. Compute neighborhood for each data record
  2. Analyze the neighborhood to determine whether data record is anomaly or not
- Two key approaches
  - Distance-based: A point is anomalous when it is far from other points
  - Density-based: A point is anomalous when it is in a low density region

Distance based Outlier Detection: **Nearest Neighbor (NN) approach**

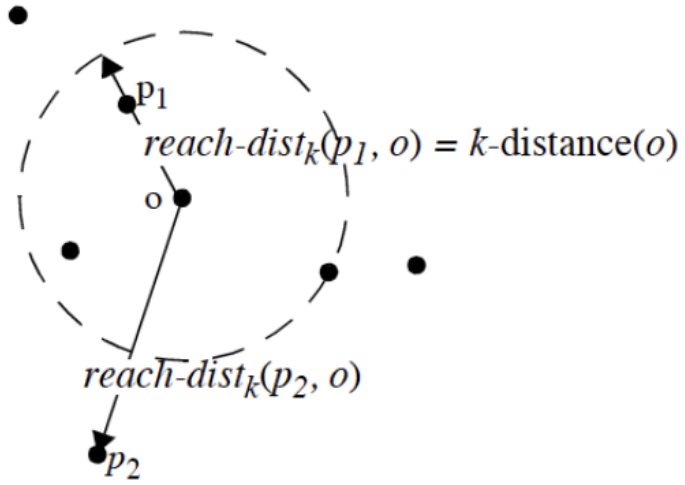
- Steps:
  1. For each data point d compute the distance to the k-th nearest neighbor dk
  2. Sort all data points according to the distance to dk
  3. Outliers are points that have the largest distance dk and therefore are located in the more sparse neighborhoods
  4. Usually data points that have top n% distance dk are identified as outliers (n – user parameter)
- Not suitable for datasets that have modes with varying density

Density based Outlier Detection: **Local Outlier Factor (LOF)**

- For each data point q compute the distance to the k-th nearest neighbor (k-distance(q))

- Compute reachability distance (reach-dist) for each data example q with respect to data example p as:

reach-dist(q, p) = max{k-distance(p), d(q,p)}



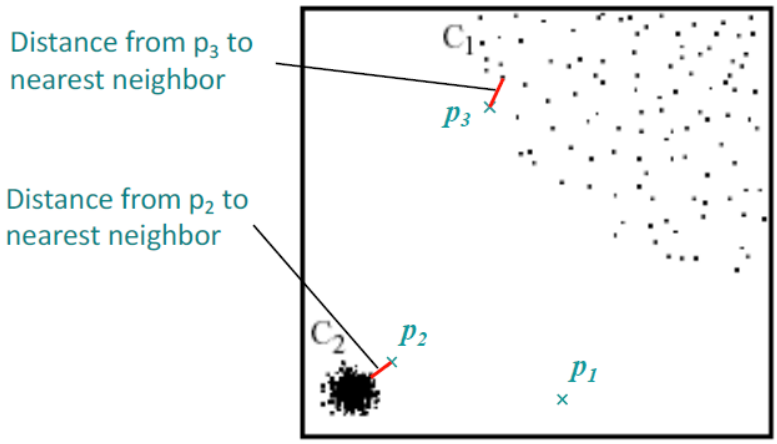
- Compute local reachability density (lrd) of data example q as inverse of the average reachability distance based on the MinPts (k) nearest neighbors of data example q

$$\text{lrd}(q) = \frac{MinPts}{\sum_{p} reach\_dist_{MinPts}(q,p)}$$

- Compute LOF(q) as ratio of average local reachability density of q's k-nearest neighbors and local reachability density of the data record q

$$\text{LOF}(q) = \frac{1}{MinPts} \cdot \sum_{p} \frac{lrd(p)}{lrd(q)}$$

- Local Outlier Factor (LOF) approach
  - Example:



In the *NN* approach, p_2 is not considered as outlier, while the *LOF* approach find both p_1 and p_2 as outliers

NN approach may consider p_3 as outlier, but LOF approach does not

Nearest Neighbor Based Techniques

- Advantage
  - Can be used in unsupervised setting
  - Do not make any assumptions about data distribution
  - Intuitively appealing, uses distances
- Drawbacks
  - Computationally expensive (when testing)
  - Requires distances, may be unintuitive
  - In high dimensional spaces, data is sparse and the concept of similarity may not be meaningful anymore: Due to the sparseness, distances between any two data records may become quite similar => Each data record may be considered as potential outlier!

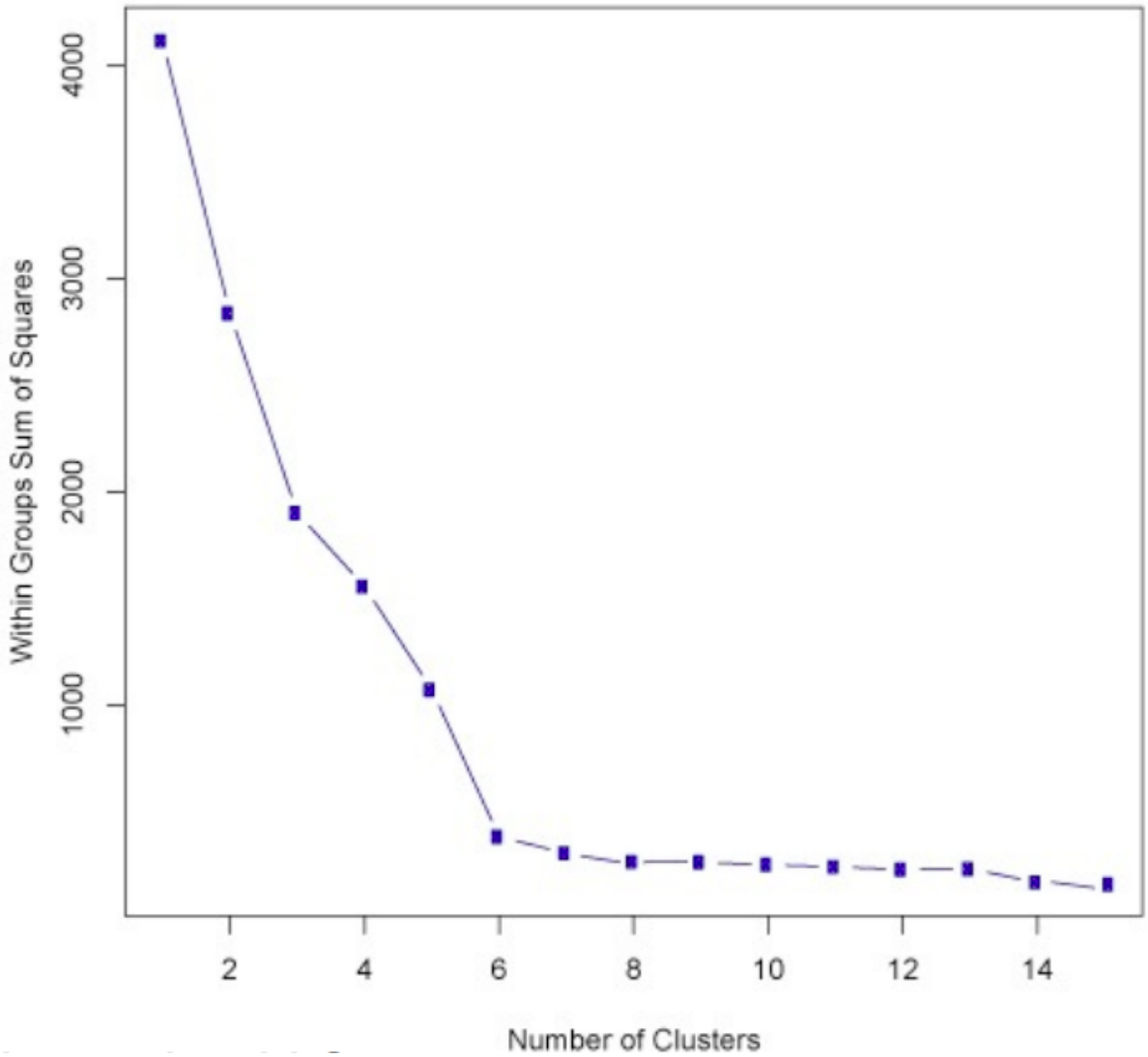
Clustering based

The idea of clustering is to define clusters such that the total intra-cluster variation [or total within-cluster sum of square (WSS)] is minimized.

- Key assumption
  - normal data records belong to large and dense clusters
  - anomalies do not belong to any cluster or form very small clusters
- Local density using clustering
  - Local anomalies are distant from other points within the same cluster

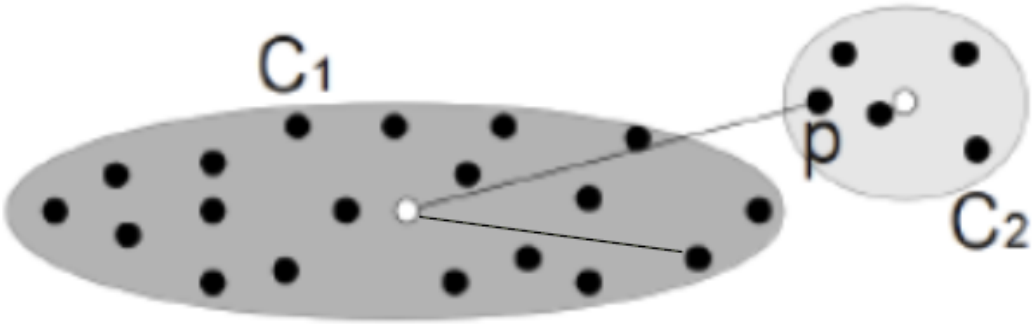
Deciding the number of clusters: ELBOW

The optical number of clusters would be adding another cluster doesn't improve much better the total WSS.



Cluster-based Local Outlier Factor (CBLOF)

- Determine CBLOF for each point using the cluster size and cluster distance
  - if point is in a small cluster, CBLOF is the product of the cluster size and its distance to the closest larger cluster
  - if point is in a large cluster CBLOF is the product of the cluster size and the distance between the point and its own cluster



Clustering Based Techniques

- Advantages:
    - No need to be supervised
    - Easily adaptable to on-line / incremental mode suitable for anomaly detection from temporal data
- Drawbacks
    - Computationally expensive
        - Using indexing structures (k-d tree, R* tree) may alleviate this problem
    - If normal points do not create any clusters, the techniques may fail
    - In high dimensional spaces, data is sparse and distances between any two data records may become quite similar.
        - Clustering algorithms may not give any meaningful clusters

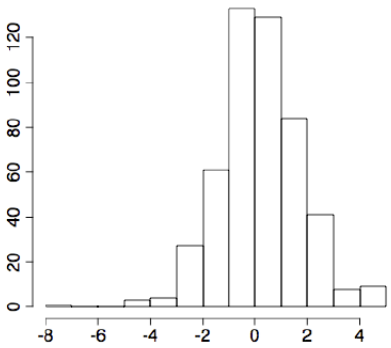
Detecting in multivariate sequences

Statistics Based Techniques

- Data points are modeled using stochastic distribution
    - points are determined to be outliers depending on their relationship with this model
- Advantage
    - Utilize existing statistical modeling techniques to model various type of distributions
- Challenges
    - With high dimensions, difficult to estimate distributions
    - Parametric assumptions often do not hold for real data sets

Hypothesis testing

$H_0 : \mu = 0$



$H_0 : \mu = 0$ null hypothesis

$H_1 : \mu > 0$ alternative hypothesis

Test statistic (t-student): $t = \frac{\overline{X}}{s}$

Reject H_0 if $t > c_\alpha$

for desired false negative rate α

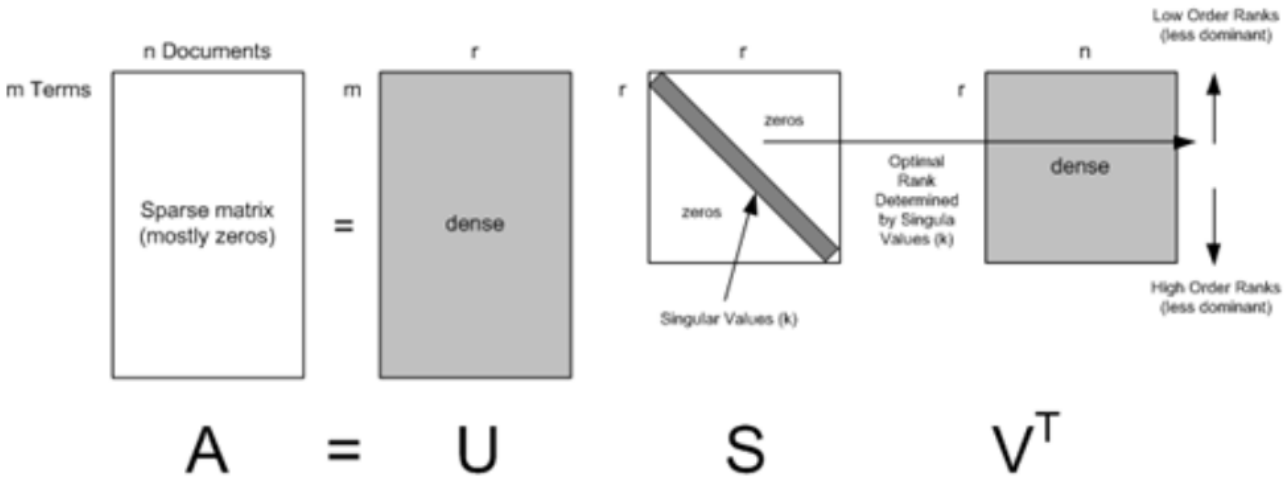
Types of Statistical Techniques

- Parametric Techniques
  - Assume that the normal (and possibly anomalous) data is generated from an underlying parametric distribution
  - Learn the parameters from the normal sample
  - Determine the likelihood of a test instance to be generated from this distribution to detect anomalies
- Non-parametric Techniques
  - Do not assume any knowledge of parameters
  - Use non-parametric techniques to learn a distribution
    - e.g. parzen window estimation

Spectral Techniques

- Analysis based on Eigen decomposition of data
- PCA (Principal Component Analysis)
  - Orthogonal transformation to reduce dimension
  - Most data patterns are captured by the several principal vectors
- Key Idea
  - Find combination of attributes that capture bulk of variability
  - Reduced set of attributes can explain normal data well
  - But do not necessarily explain the outliers
- Several methods use Principal Component Analysis

- Top few principal components capture variability in normal data
- Smallest principal component should have constant values
- Outliers have variability in the smallest component



PCA (Principal Component Analysis)

- Deriving principal vectors
  - Deriving the principal vector which captures the maximum variance

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \mathrm{Var}\{\mathbf{w}^{\mathrm{T}}\mathbf{X}\} = \arg\max_{\|\mathbf{w}\|=1} E\left\{\left(\mathbf{w}^{\mathrm{T}}\mathbf{X}\right)^2\right\}$$

  - Find next component

$$\hat{\mathbf{X}}_{k-1} = \mathbf{X} - \sum_{i=1}^{k-1} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}} \mathbf{X}$$

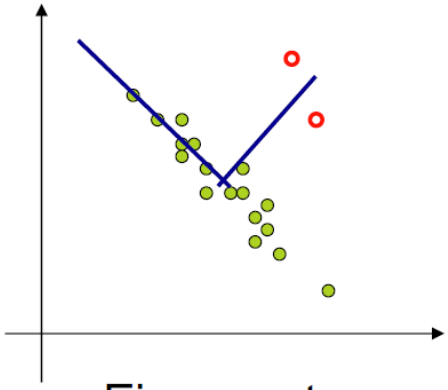
$$\mathbf{w}_k = \arg\max_{\|\mathbf{w}\|=1} E\left\{\left(\mathbf{w}^{\mathrm{T}}\hat{\mathbf{X}}_{k-1}\right)^2\right\}.$$

Example: Network traffic

Data matrix

$$\mathbf{Y}= \begin{bmatrix} \ldots. \\ 100 \;\; 30 \;\; 42 \;\; 212 \;\; 1729 \;\; 13 \\ \ldots. \end{bmatrix}$$

Perform PCA on matrix **Y**



Low-dimensional data

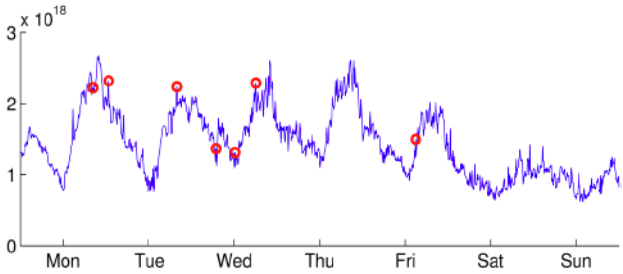
$$\mathbf{Yv}= \begin{bmatrix} \ldots. \\ y_t^T v_1 \;\; y_t^T v_2 \\ \ldots. \end{bmatrix}$$

Eigenvectors (components)

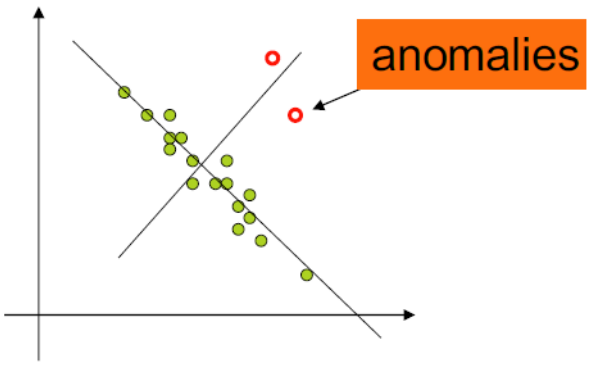
$$\begin{bmatrix} v_1 \;\; v_2 \;\; \ldots \end{bmatrix}$$

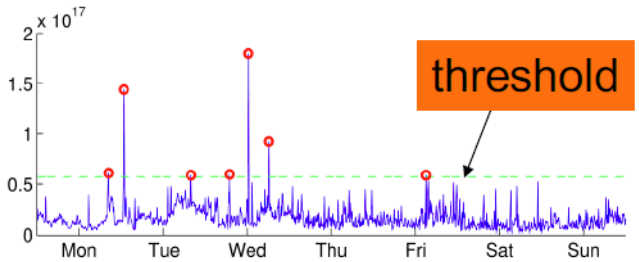
Example: Network traffic

Abilene backbone network traffic volume over 41 links collected over 4 weeks



Perform PCA on 41-dim data
Select top 5 components





Projection to residual subspac

$$\mathbf{y} = \hat{\mathbf{y}} + \tilde{\mathbf{y}} \qquad \hat{\mathbf{y}} = \mathbf{P}\mathbf{P}^T\mathbf{y} = \mathbf{C}\mathbf{y} \qquad \tilde{\mathbf{y}} = \left(\mathbf{I} - \mathbf{P}\mathbf{P}^T\right)\mathbf{y} = \tilde{\mathbf{C}}\mathbf{y}$$

$$P = [v_1 \; v_2 \; \ldots v_r]$$

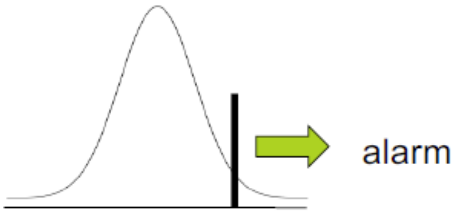
Using Robust PCA*

- Variability analysis based on robust PCA
  - Compute the principal components of the dataset
  - For each test point, compute its projection on these components
  - If yi denotes the ith component, then the following has a chi-squared distribution

$$\sum_{i=1}^{q} \frac{y_i^2}{\lambda_i} = \frac{y_1^2}{\lambda_1} + \frac{y_2^2}{\lambda_2} + \ldots + \frac{y_q^2}{\lambda_q}, q \leq p$$

  - An observation is outlier if for a given significance level (statistical test)

$$\sum_{i=1}^{q} \frac{y_i^2}{\lambda_i} > \chi_q^2(\alpha)$$



- Have been applied to intrusion detection, outliers in space-craft components, etc.

Spectral Techniques

- Remember to first normalize your data, if your PCA method does not do it for you
- Advantage
  - Useful for multi-variate signals
  - Computationally efficient (use graphic cards!)
- Disadvantage
  - Based on the assumption that anomalies and normal instances are distinguishable in the reduced space
  - Does not take context into account
  - PCA is sensitive to outliers...

Evaluating anomaly detection

Anomaly detection is hard to evaluate

- Often little/no information on positives
  - Rely on quality of clustering, no clear quality measure exists
  - Good distances are often hard to find
- Anomalies are usually time periods instead of points

    - An attack starts and stops
    - Is every detection within that period a true positive?
- Unclear how to count positives
    - Many alarms are raised in a few seconds, is this a single positive?
    - Should we group them over time or per host/group?

Lecture 5: Deep Learning and State Based Models

Lecture 6: Data Streams

Lecture 7: Adversarial Machine Learning