ML: Lecture 8

Time series classification

Panagiotis Papapetrou, PhD
Professor, Stockholm University

Syllabus

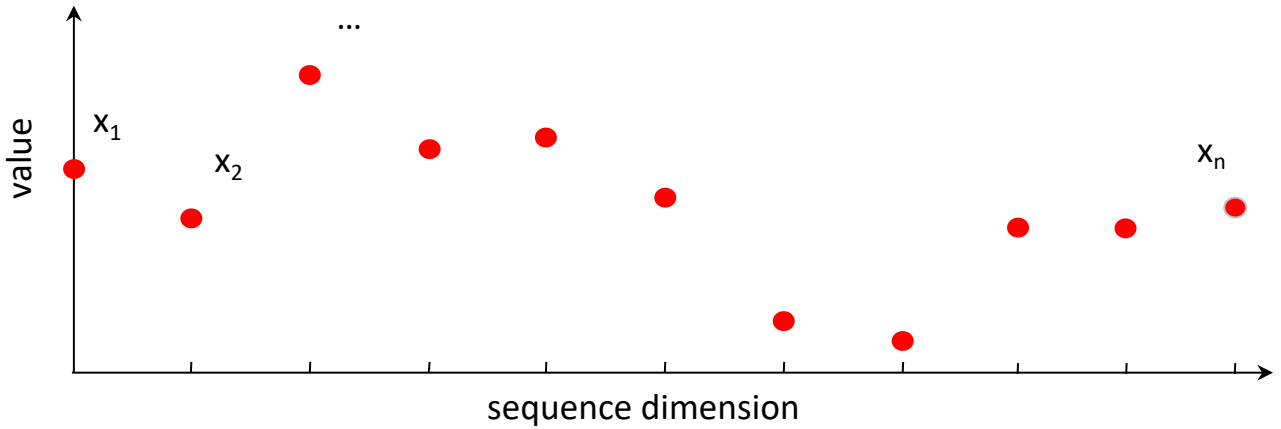
Jan 16	Introduction to machine learning
Jan 18	Regression analysis
Jan 19	Laboratory session 1: numpy and linear regression
Jan 23	Ensemble learning
Jan 25	Deep learning I: Training neural networks
Jan 26	Laboratory session 2: ML pipelines, ensemble learning
Jan 30	Deep learning II: Convolutional neural networks
Feb 1	Laboratory session 3: training NNs and tensorflow
Feb 6	Deep learning III: Recurrent neural networks
Feb 8	Laboratory session 4: CNNs and RNs
Feb 13	Deep learning IV: Autoencoders, transformers, and attention
Feb 20	**Time series classification**

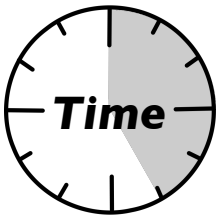
Today

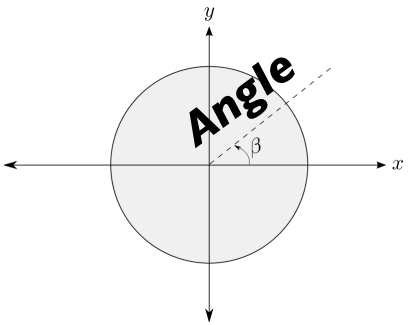
- What is a time series?
- How to compare time series?
- How to perform dimensionality reduction?
- Feature-based time series classification
- Deep learning-based time series classification

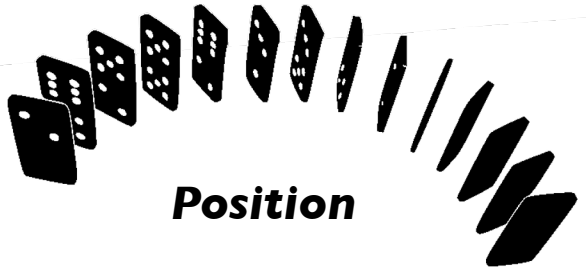
Data series

- Sequence of points ordered along some dimension



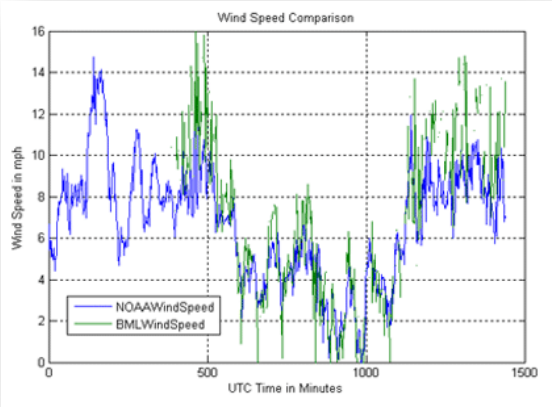


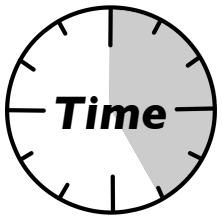




Data series

- Sequence of points ordered along some dimension



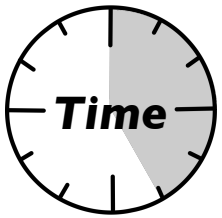


Wind speed

From ocean observing node project, http://bml.ucdavis.edu/boon/wind.html

Data series

- Sequence of points ordered along some dimension

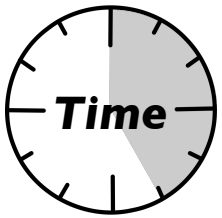


Historical stock quotes

From http://money.cnn.com/2012/04/23/markets/walmart_stock/index.htm

Data series

- Sequence of points ordered along some dimension

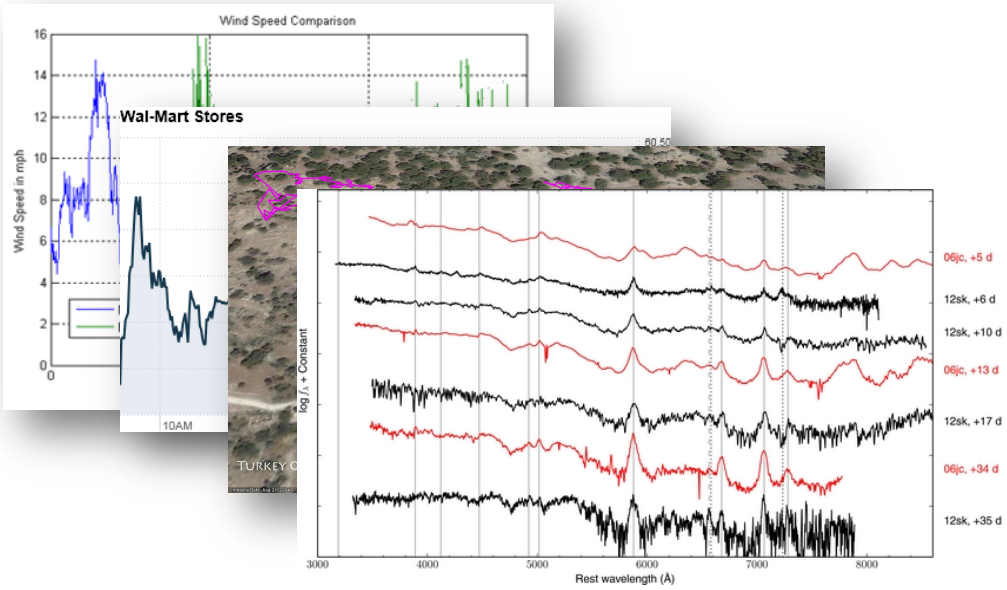


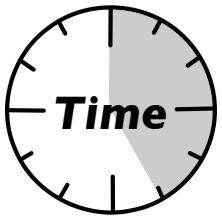
Trajectories from GPS logs

From http://www.flickr.com/photos/kitepuppet/3604115258

Data series

- Sequence of points ordered along some dimension



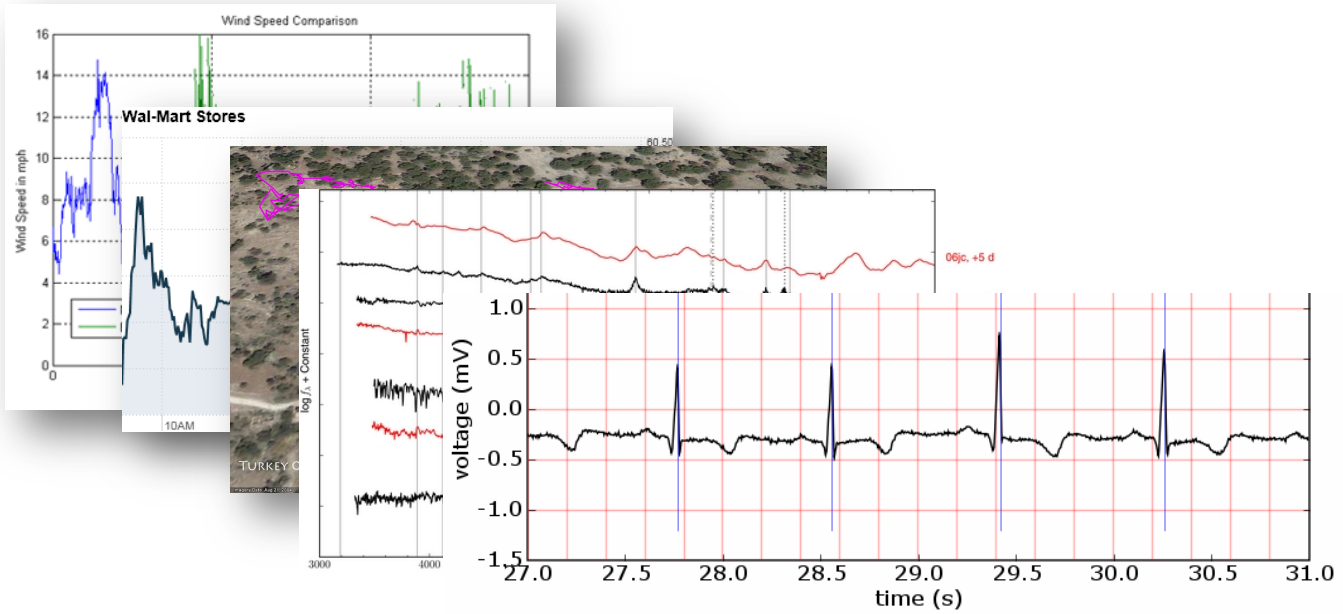


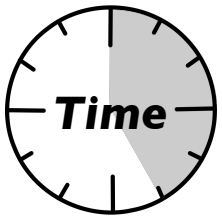
Spectroscopic sequence data (astronomy)

From Sanders et al., http://dx.doi.org/10.1088/0004-637X/769/1/39

Data series

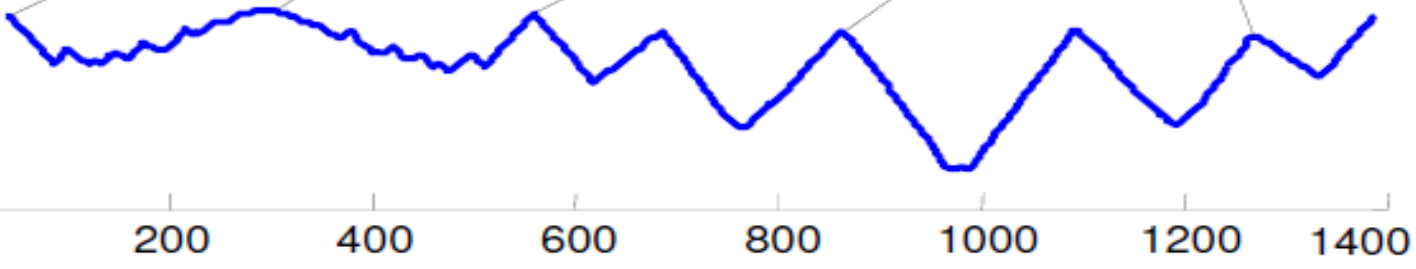
- Sequence of points ordered along some dimension

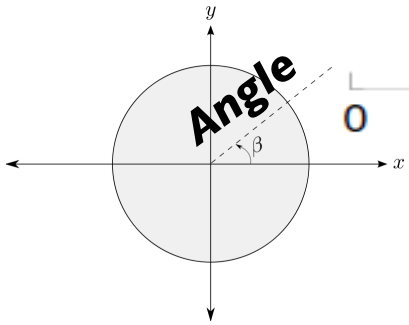




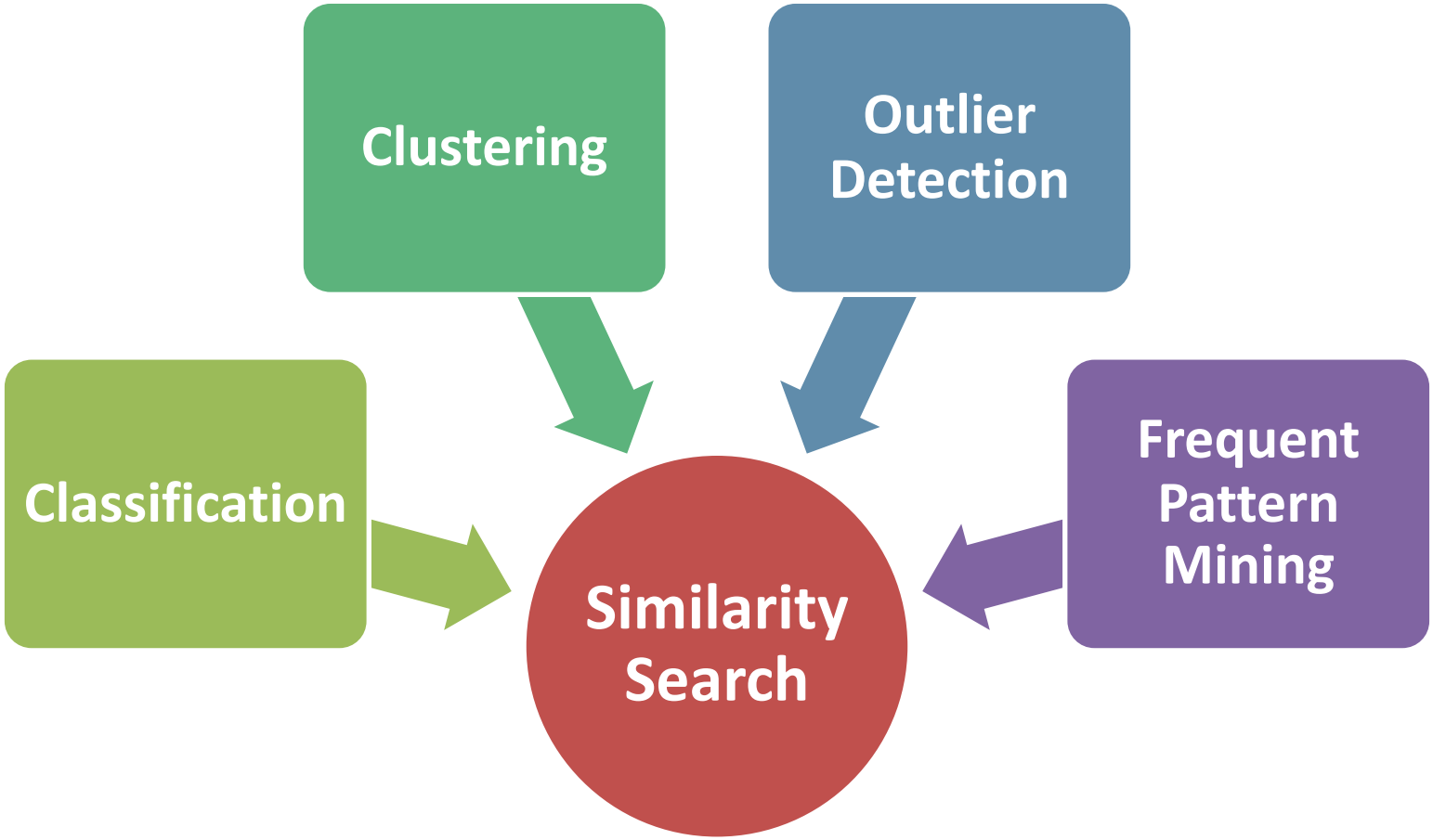
Electocardiograms (cardiology)

https://archive.physionet.org/physiobank/


0
200
400
600
800
1000
1200
1400

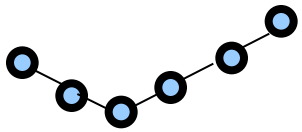

Angle
y
x
β

Analysis tasks

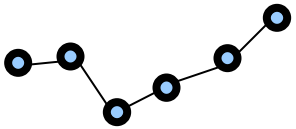


Time series similarity

- Given two time series



$X = (x_1, x_2, \ldots, x_n)$



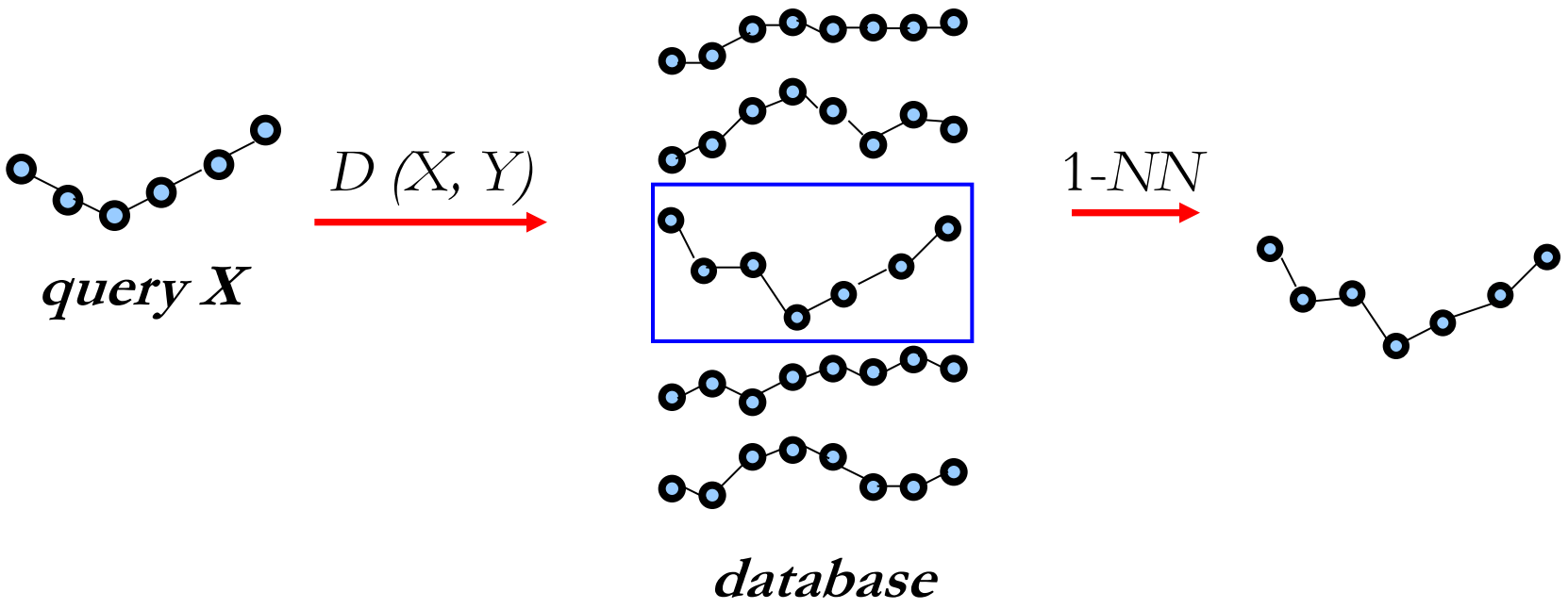
$Y = (y_1, y_2, \ldots, y_n)$

- Define and compute D (X, Y)

- **Or better…**

Time series similarity search

- Given a time series database and a query X
- Find the best match of X in the database



- **Why is that useful?**

Examples

- Find companies with similar stock prices over a time period
- Find users with similar credit card utilization
- Find patients with a suspicious ECG segment
- Find stocks with similar progression patterns

The Lp norm / Euclidean distance

- View each time series as a point in the n-dimensional space

  (n = length)

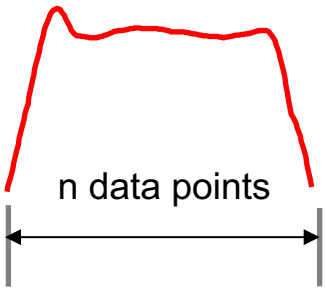
- Define the distance between time series X and Y as

$$L_p = \left(\sum_{i=1}^{n} | x_i - y_i |^p\right)^{1/p}$$

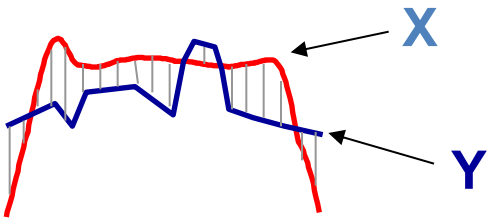
For p = 2 => The Euclidean Distance

The Euclidean Distance

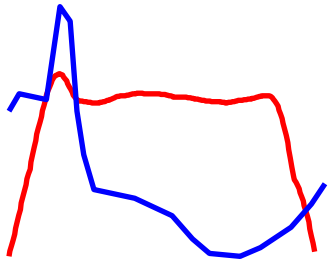
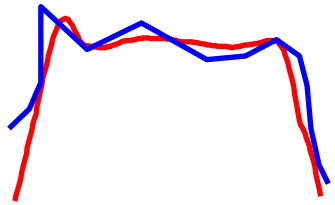
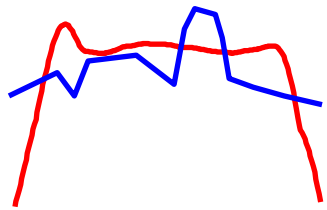
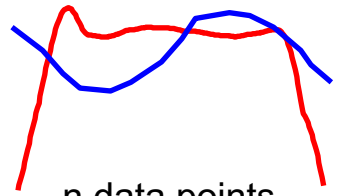
Query X

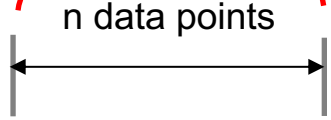


Euclidean Distance between two time series X = {x_1, x_2, …, x_n} and Y = {y_1, y_2, …, y_n}



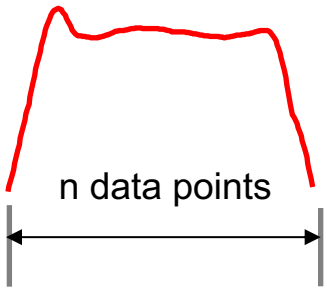
$$D(X,Y) \equiv \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Database	Distance	Rank
	0.98	4
	0.07	1
	0.21	2
	0.43	3

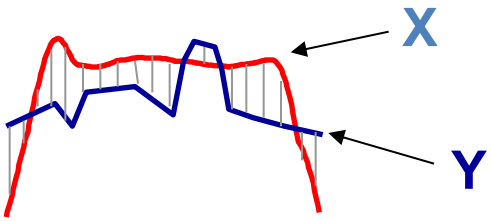


Disadvantages ?

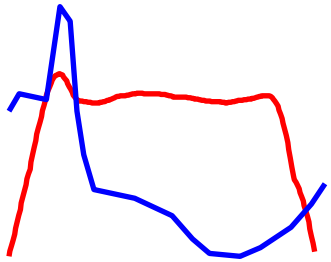
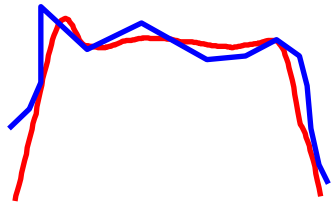
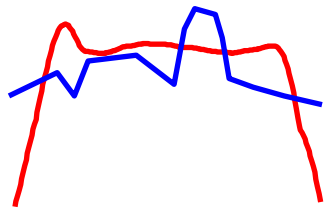
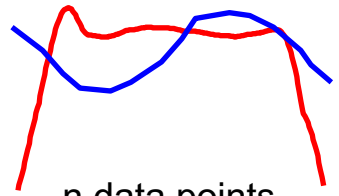
Query X

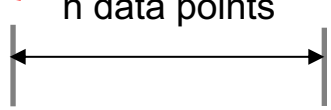


Euclidean Distance between two time series X = {x_1, x_2, …, x_n} and Y = {y_1, y_2, …, y_n}



$$D(X,Y) \equiv \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Database	Distance	Rank
	0.98	4
	0.07	1
	0.21	2
	0.43	3



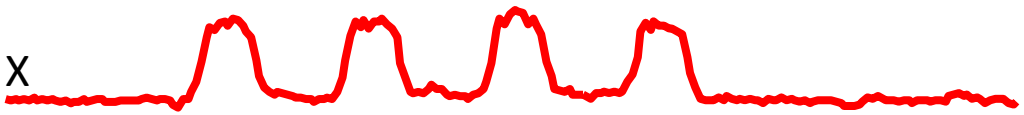
Disadvantages

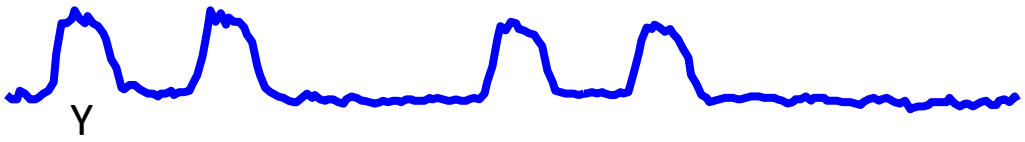
- Query and target lengths should be equal

- Cannot tolerate noise in terms of
  - time shifts
  - sequences out of phase
  - scaling in the y-axis

Limitations of Euclidean Distance

figures taken from Eamonn Keogh, University of California, Riverside





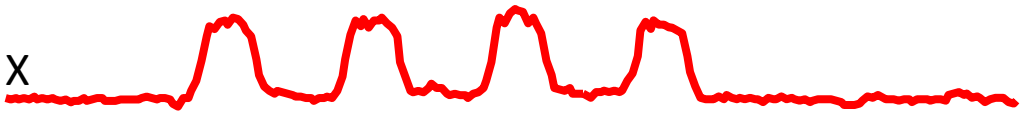
Euclidean Distance

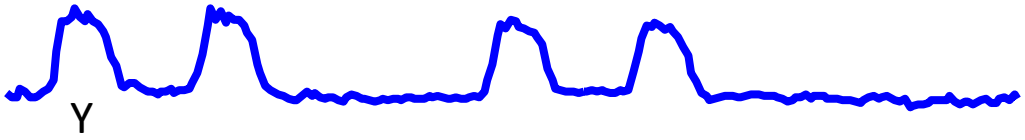
Sequences are aligned "one to one".

$$D(X,Y) \equiv \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Limitations of Euclidean Distance

figures taken from Eamonn Keogh, University of California, Riverside

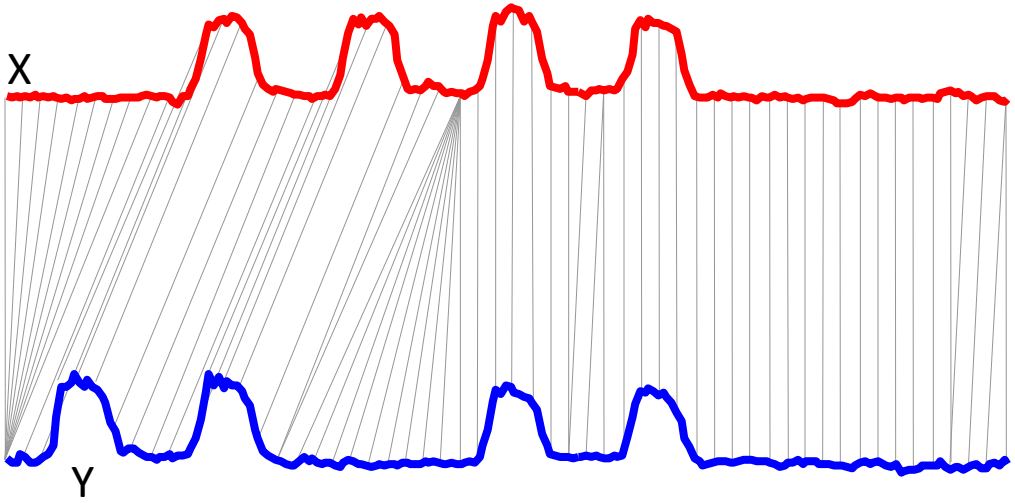




Euclidean Distance

Sequences are aligned "one to one".

$$D(X,Y) \equiv \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

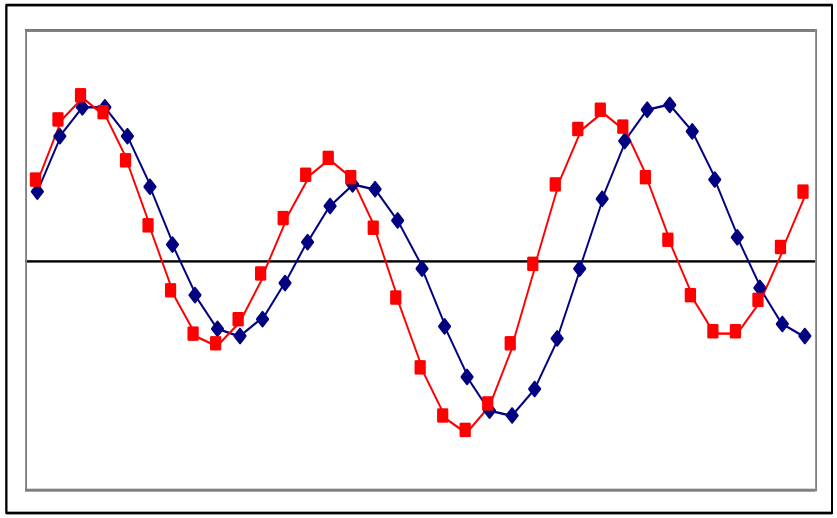


"Warped" Time Axis

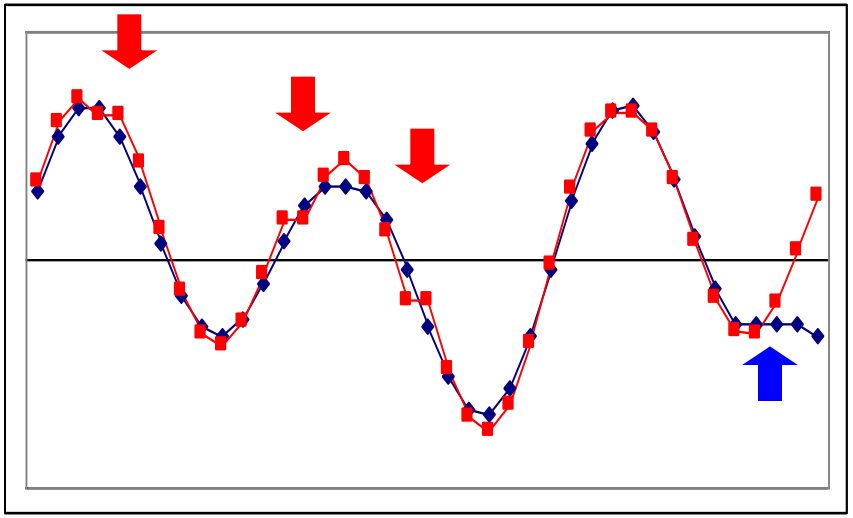
Nonlinear alignments are possible.

Dynamic Time Warping (DTW) [Berndt, Clifford, 1994]

- DTW allows sequences to be stretched along the time axis
  - Insert 'stutters' into a sequence
  - THEN compute the (Euclidean) distance



original



'stutters'

Computing DTW

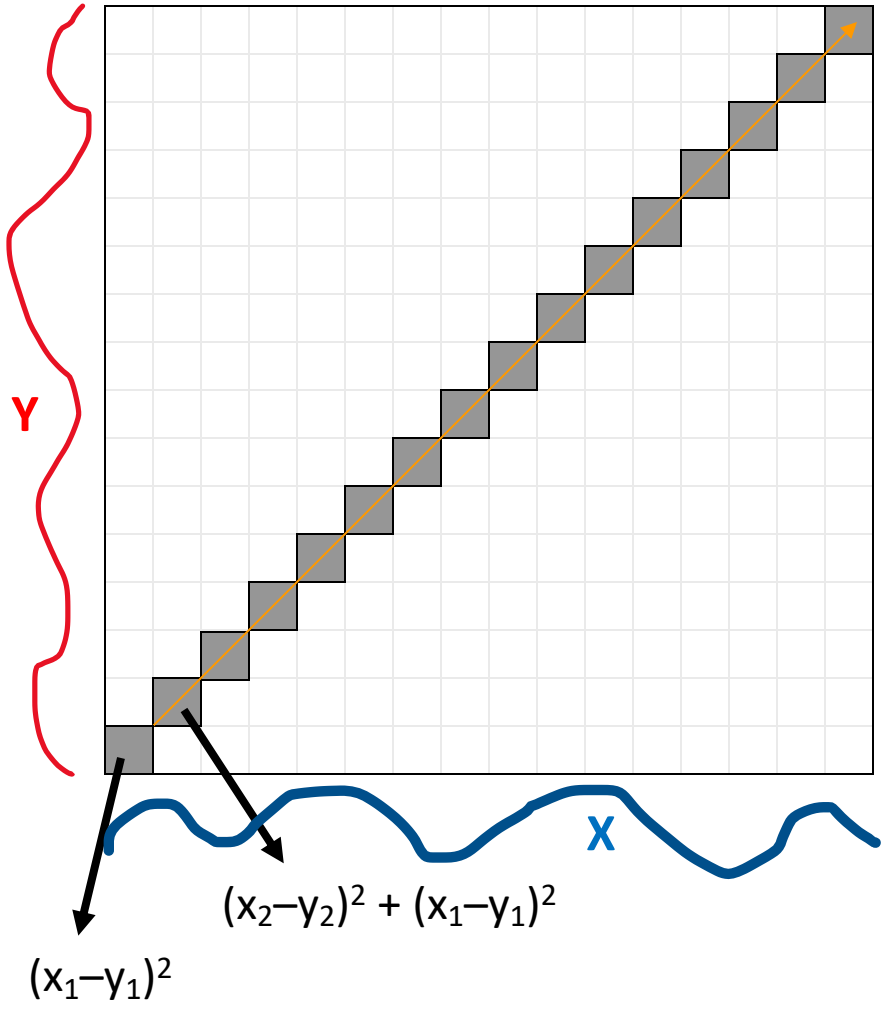
- DTW is computed by dynamic programming (DP)
- Given two sequences
  - $X = \{x_1, x_2, \ldots, x_n\}$
  - $Y = \{y_1, y_2, \ldots, y_m\}$

- Use an n x m *DP* matrix ***f*** to store the scores

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

$$f(i,j) = \left\| x_i - y_j \right\| + \min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases}$$

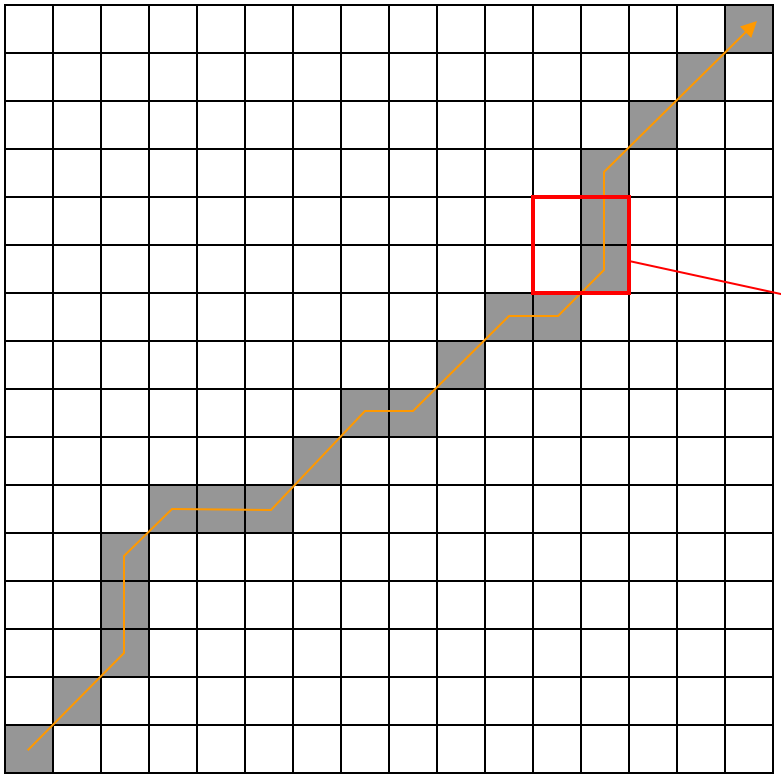
Computing DTW



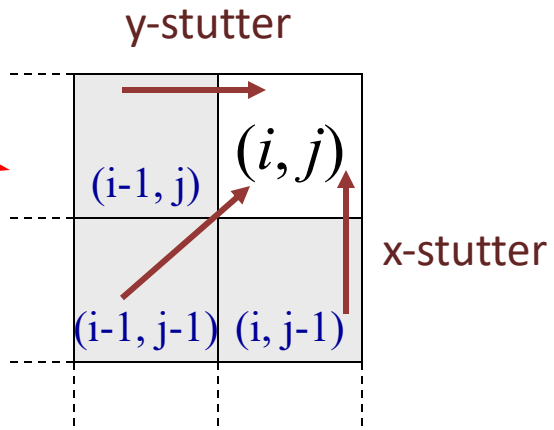
- Each cell (i, j) is a pair of indices
- Cell values are computed using some norm, e.g., p=2
- Euclidean path:
  - $i = j$
  - Ignores off-diagonal cells

$$f(i,j) = \left\| x_i - y_j \right\| + \min \begin{cases} f(i, j-1) \\ f(i-1, j) \\ f(i-1, j-1) \end{cases}$$

Computing DTW



- DTW allows any path
- Examine all paths:

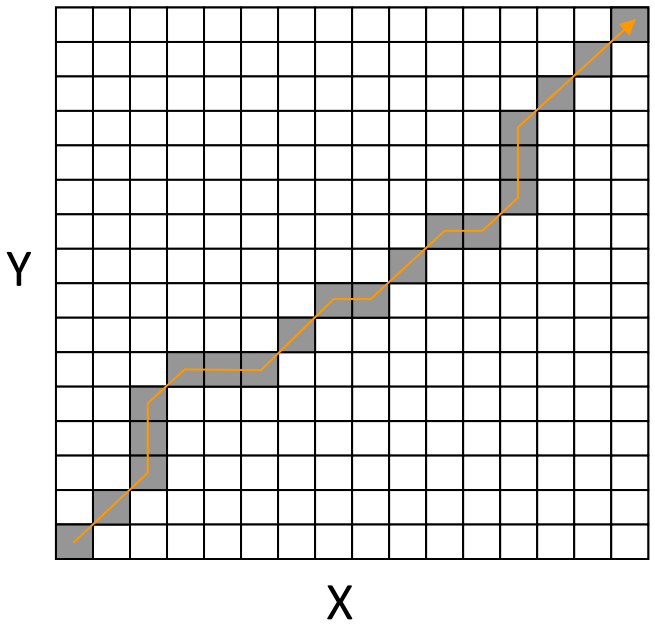


$$f(i,j) = \|x_i - y_j\| + \min\begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases}$$

- Standard dynamic programming to fill in the table
- The top-right cell contains final result

Properties of DTW

- Warping path W:
  - set of grid cells in the time warping matrix
- DTW finds an optimum warping path W:
  - the path with the smallest matching score



Properties of a DTW legal path

I. Boundary conditions

$W_1=(1,1)$ and $W_K=(n,m)$

II. Continuity

Given $W_k = (a, b)$, then

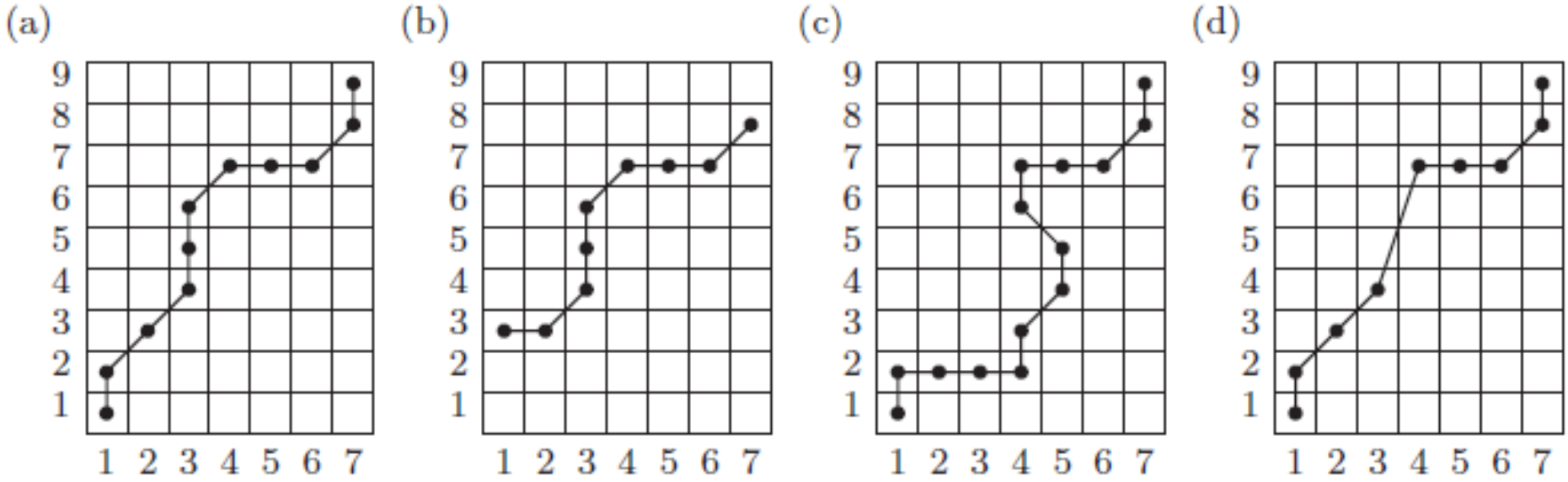
$W_{k-1} = (c, d)$, where $a-c \leq 1$, $b-d \leq 1$

III. Monotonicity

Given $W_k = (a, b)$, then

$W_{k-1} = (c, d)$, where $a-c \geq 0$, $b-d \geq 0$

Properties of DTW



I. Boundary conditions

W_1=(1,1) and W_K=(n,m)

II. Continuity

Given W_k = (a, b), then

W_{k-1} = (c, d), where a-c ≤ 1, b-d ≤ 1

III. Monotonicity

Given W_k = (a, b), then

W_{k-1} = (c, d), where a-c ≥ 0, b-d ≥ 0

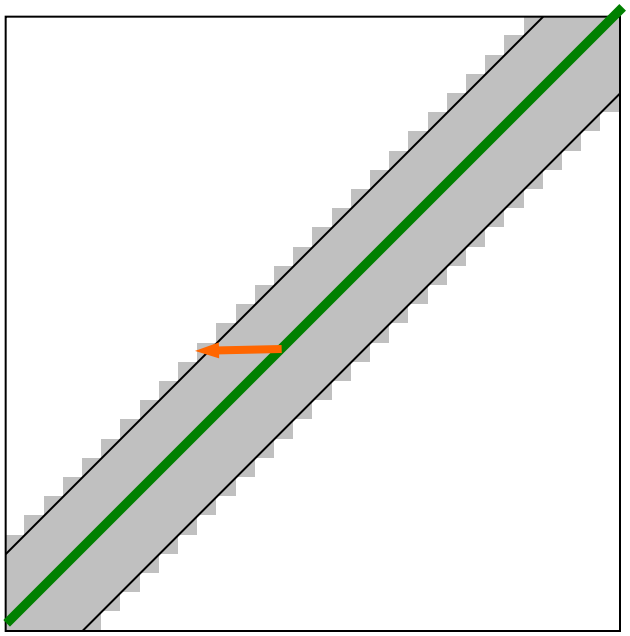
Which paths violate these properties?

Global Constraints

figures taken from Eamonn Keogh

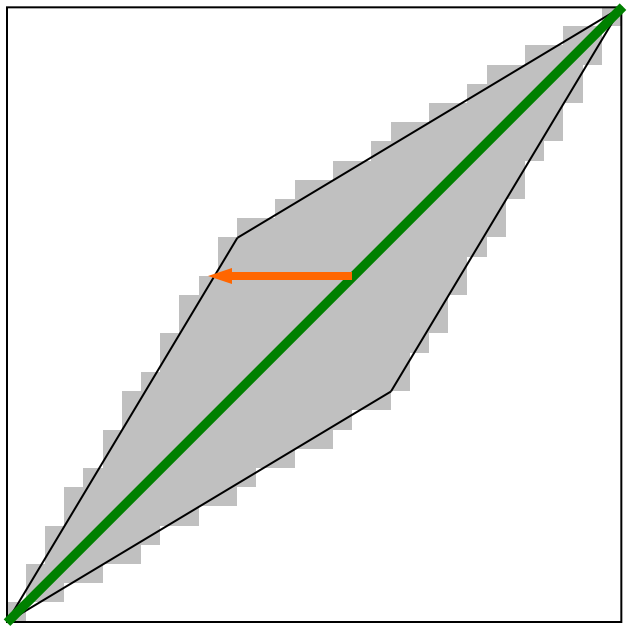
- Slightly speed up the calculations and prevent pathological warpings
- A global constraint limits the indices of the warping path

  $w_k = (i, j)_k$ such that $j\text{-}r \leq i \leq j\text{+}r$
- Where r is a term defining allowed range of warping for a given point in a sequence



Sakoe-Chiba Band

$r =$



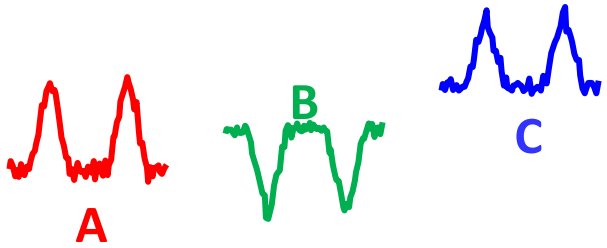
Itakura Parallelogram

Pros and cons

- **Advantages**
  - Query and target lengths **may not** be of equal length ☺
  - Can tolerate:
    - time shifts
    - sequences out of phase
    - scaling in the y-axis (how?)
- **Disadvantages**
  - Computational complexity: O(n m) for unconstrained and O(*r* n) for constrained
  - May not be able to handle missing or noisy values

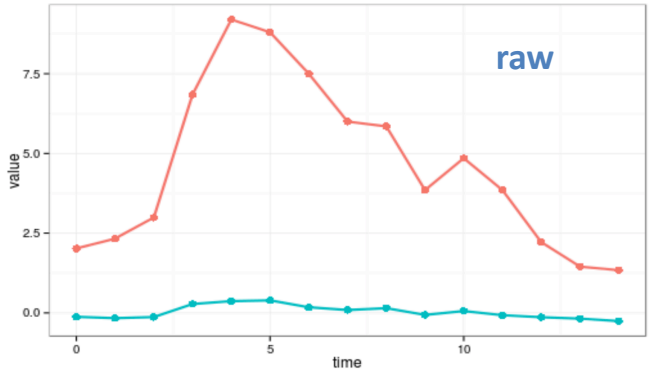
Z-normalization

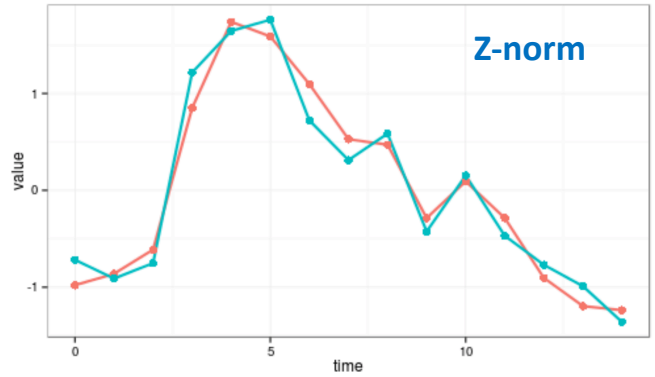
- Needed when interested in *detecting trends* and not absolute values



- For streaming data:
  - each subsequence of interest should be z-normalized before being compared to the z-normalized query
  - otherwise the trends are lost
- Z-normalization guarantees:
  - offset invariance
  - scale/amplitude invariance

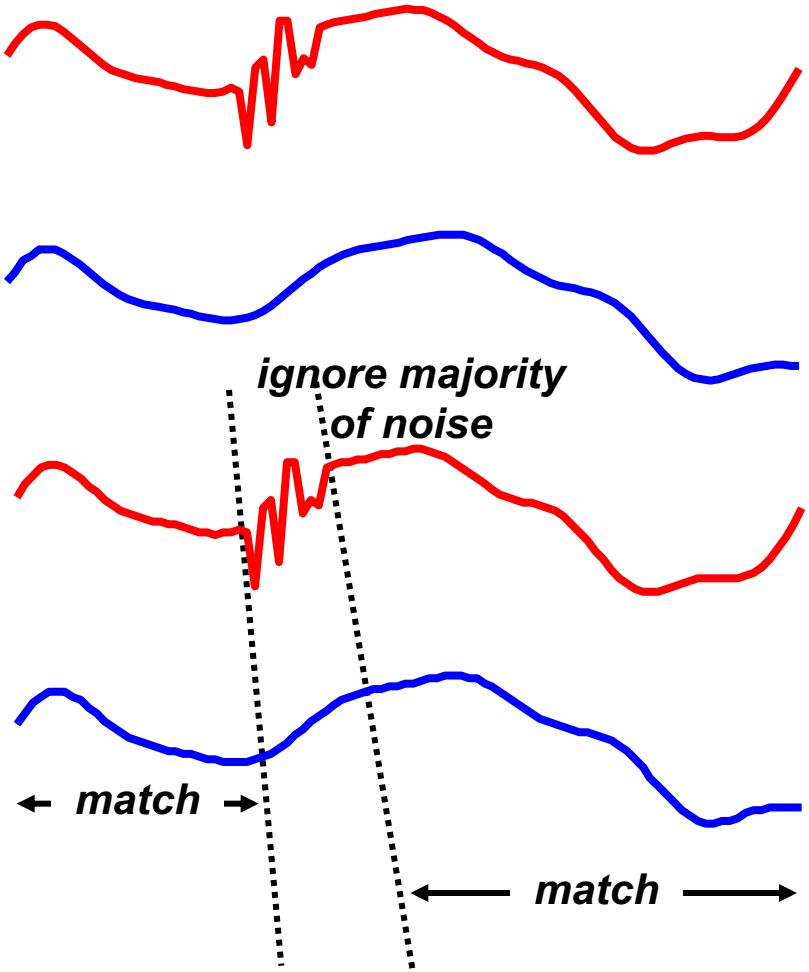
$$x_i' = \frac{x_i - \mu}{\sigma}, \text{ where } i \in \mathbb{N}$$





Longest Common Subsequence (LCSS)

LCSS is more resilient to noise than DTW



Disadvantages of DTW:

A. All points are matched

B. Outliers can distort distance

C. One-to-many mapping

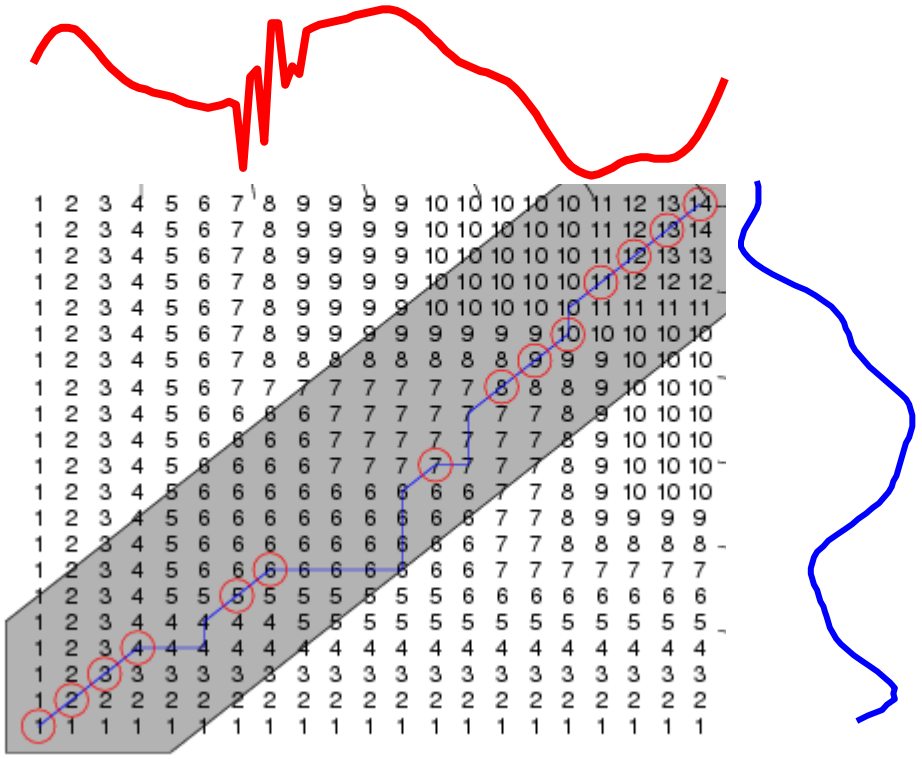
Advantages of LCSS:

A. Outlying values not matched

B. Distance/Similarity distorted less

Longest Common Subsequence (LCSS)

$$LCSS[i,j] = \begin{cases} 0 & \text{if} \quad i = 0 \\ 0 & \text{if} \quad j = 0 \\ 1 + LCSS[i-1, j-1] & \text{if} \quad |a_{i,k} - b_{j,k}| < \epsilon, \quad k = 1 \ldots d \\ max(LCSS[i-1,j], LCSS[i,j-1]) & \text{otherwise} \end{cases}$$



Similar dynamic programming solution as DTW, but now we measure similarity not distance

DFT

Agrawal, Faloutsos, & Swami. FODO 1993
Faloutsos, Ranganathan, & Manolopoulos. SIGMOD 1994

DWT

Chan & Fu. ICDE 1999

PAA

Keogh, Chakrabarti, Pazzani & Mehrotra SIGMOD 2001
Yi & Faloutsos VLDB 2000

APCA

Keogh, Chakrabarti, Pazzani & Mehrotra SIGMOD 2001

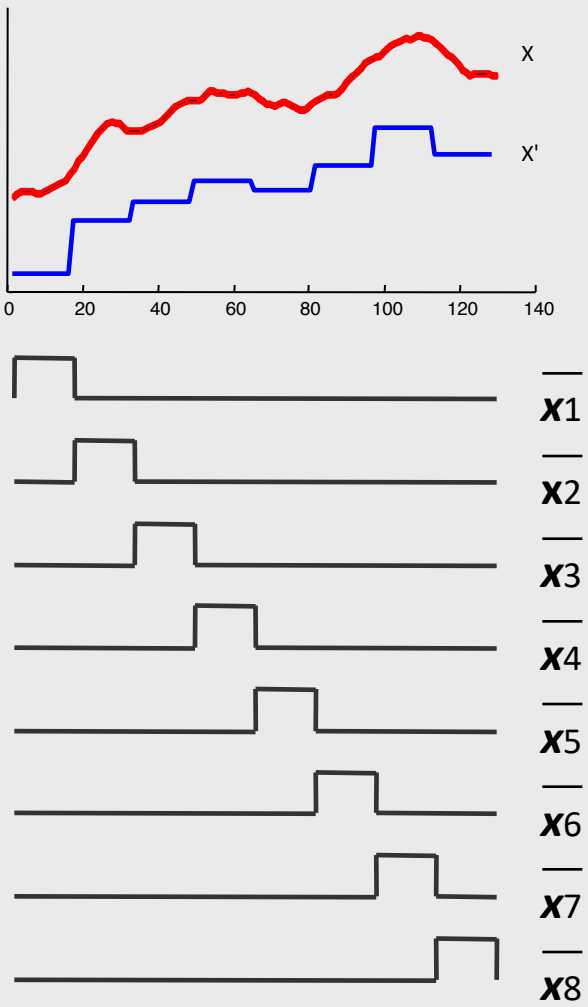
PLA

Morinaka, Yoshikawa, Amagasa, & Uemura, PAKDD 2001

SAX

Lin, Keogh, Wein, & Lonardi, DMKD 2007

Piecewise Aggregate Approximation (PAA)



Basic Idea: Represent the original time series of length *n* as a sequence of box basis functions, each box being of the same length resulting in a new time series of length *N* << *n*

Computation:

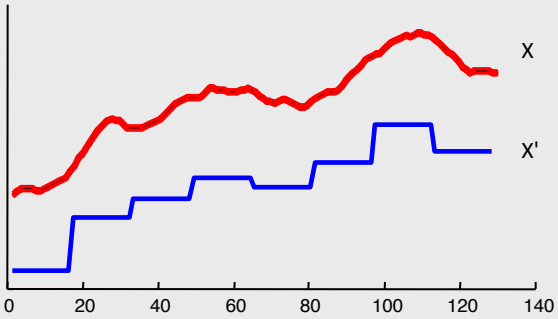
- X: original time series of *length n*
- It can be represented in the N-dimensional space as:

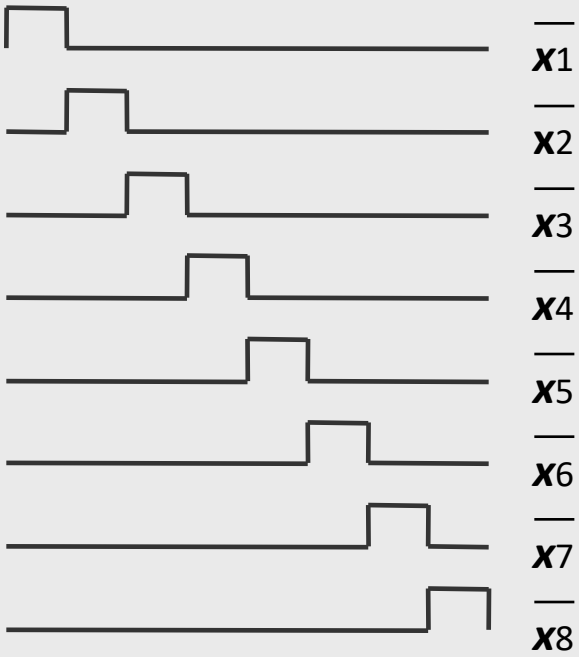
$$\bar{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j$$

Keogh, Chakrabarti, Pazzani & Mehrotra, KAIS (2000)

Byoung-Kee Yi, Christos Faloutsos, VLDB (2000)

Piecewise Aggregate Approximation (PAA)

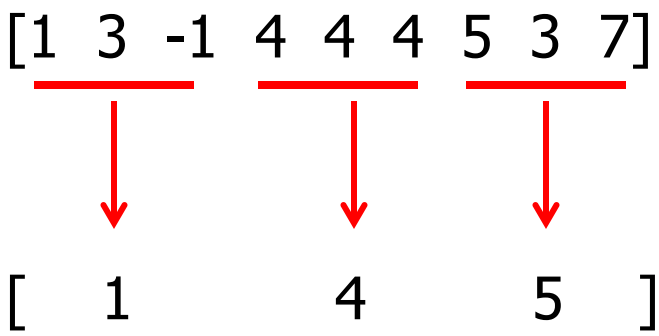




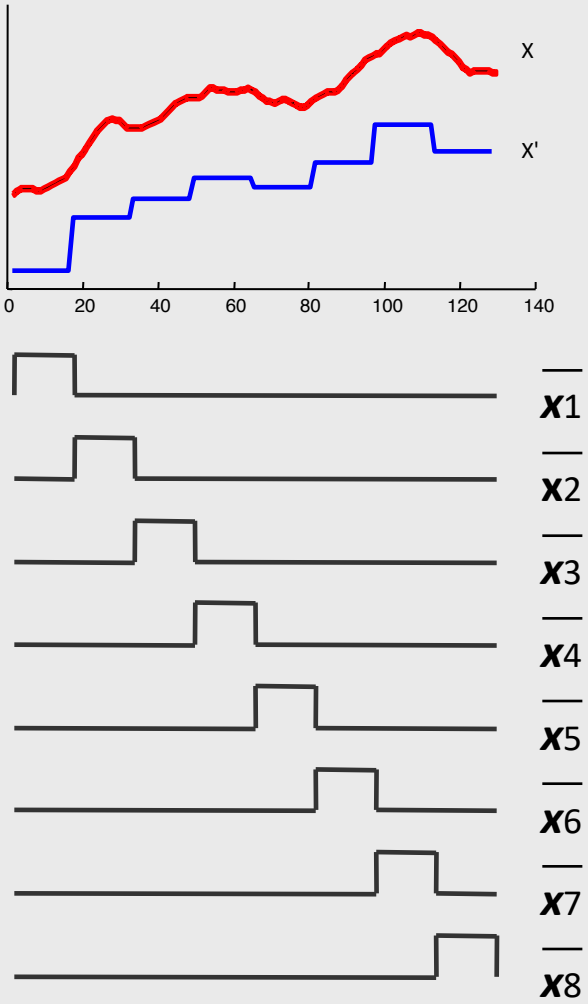
Example

Let X = [1 3 -1 4 4 4 5 3 7]

- X can be mapped from its original dimension *n = 9* to a lower dimension, e.g., *N = 3*, as follows:



Piecewise Aggregate Approximation (PAA)



Pros and Cons of PAA as a time series representation

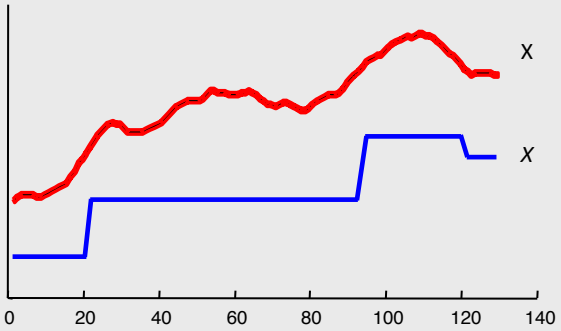
Pros:

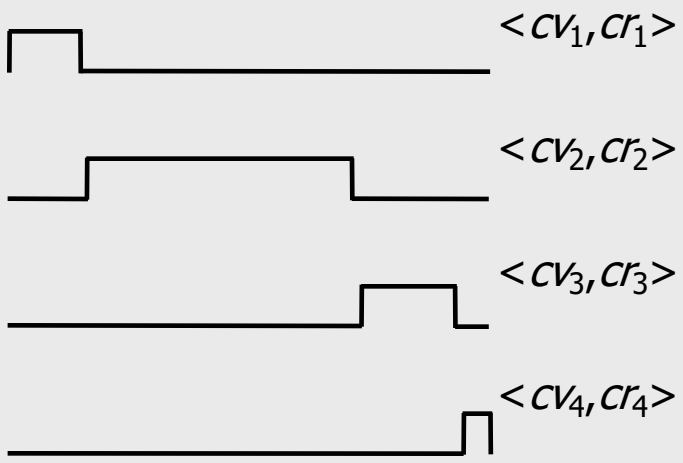
- *Extremely* fast to calculate
- As efficient as other approaches (empirically)
- Support queries of arbitrary lengths
- Can support any Minkowski metric
- Supports non-Euclidean measures
- Supports weighted Euclidean distance
- *Simple!* Intuitive!

Cons:

- It assumes a fixed-length non-overlapping summary window over the time series

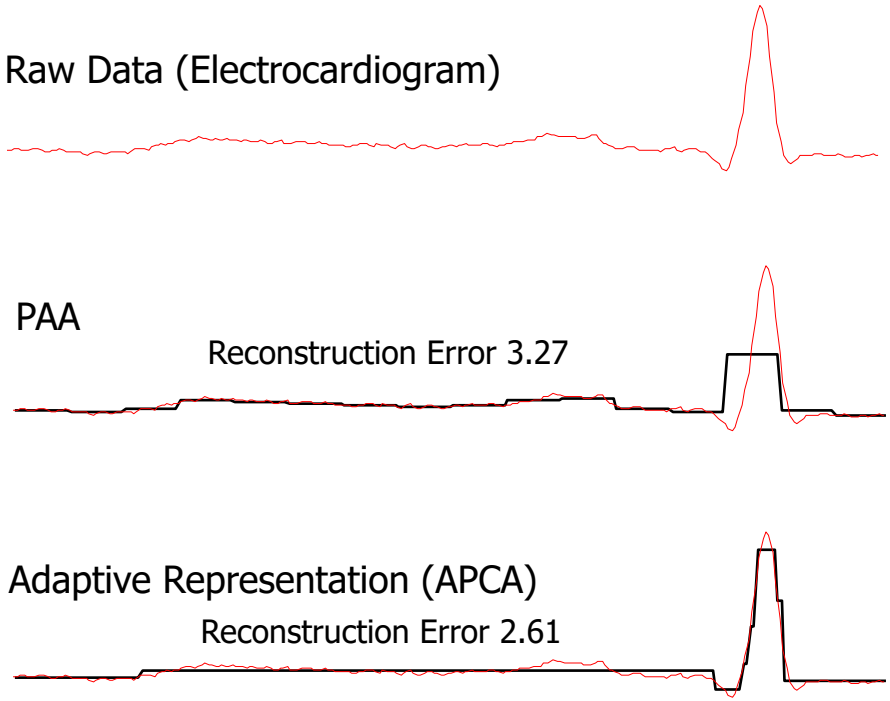
Adaptive Piecewise Constant Approximation (APCA)





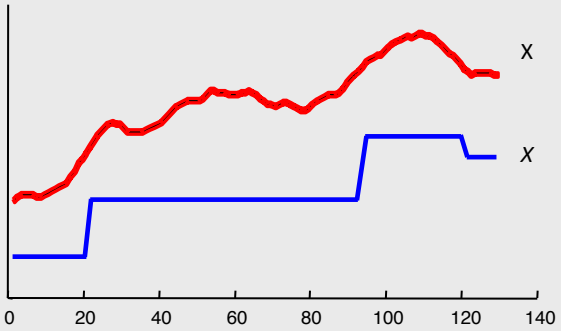
Generalize PAA to allow the piecewise constant segments to have arbitrary lengths

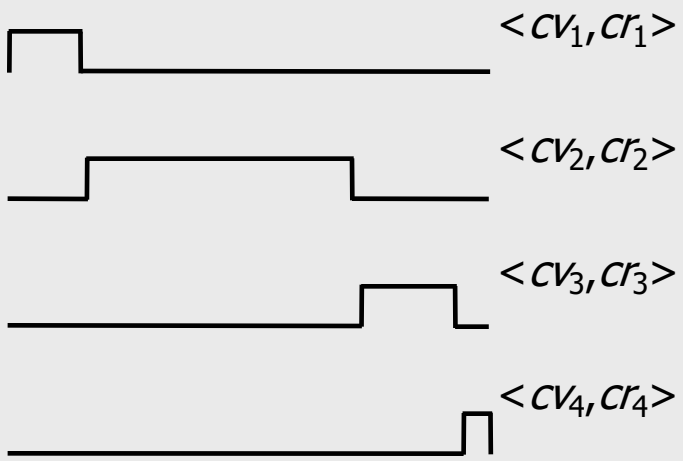
*We need two coefficients to represent each segment: its value (**cv**$_i$) and its length (**cr**$_i$).*



Intuition: many signals have little detail in some places and high detail in other places; APCA can *adaptively* fit itself to the data

Adaptive Piecewise Constant Approximation (APCA)





Pros and Cons of APCA as a time series representation

Pros:

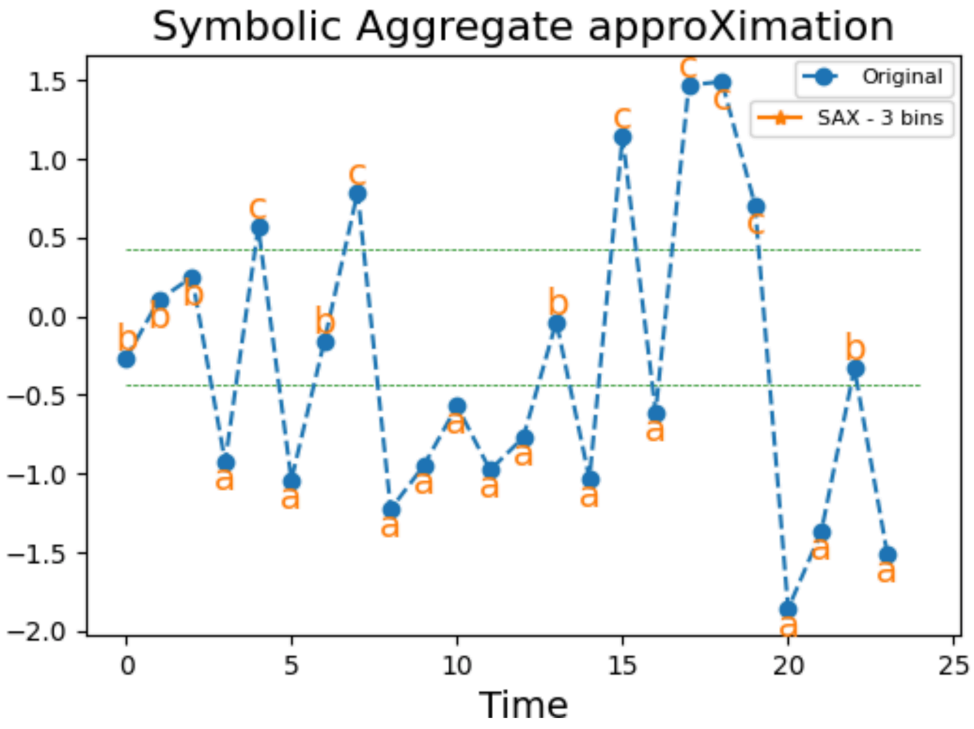
- Fast to calculate O(*n*)
- *More* efficient than other approaches
- Supports queries of arbitrary lengths
- Supports non Euclidean measures
- Support fast exact queries, and even faster approximate queries on the same data structure

Cons:

- Performance depends highly on the segmentation technique used

Symbolic ApproXimation (SAX)

- Similar in principle to PAA
  - uses segments to represent data series
- Represents segments with symbols (rather than real numbers)
  - small memory footprint



Symbolic ApproXimation (SAX)

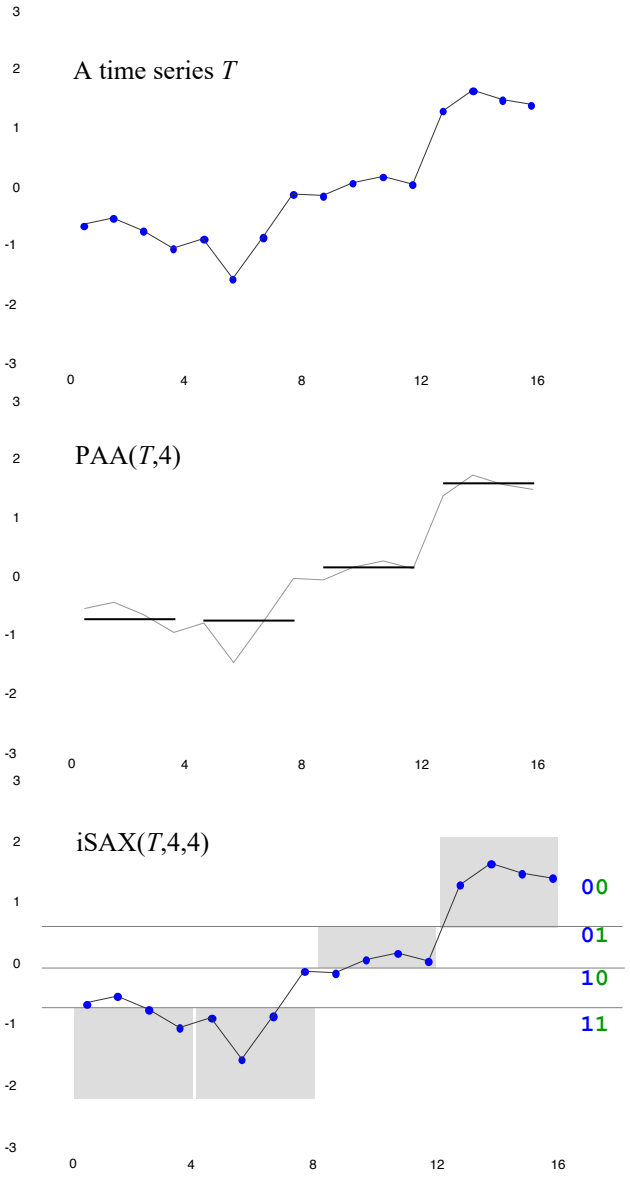
Represent the time series ***T*** of length ***n*** with ***w*** segments using PAA

- *T* typically normalized to $\mu = 0$, $\sigma = 1$
- PAA(*T*, *w*): $\overline{T} = \bar{t}_1, \ldots, \bar{t}_w$,

  with $\bar{t}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} T_j$

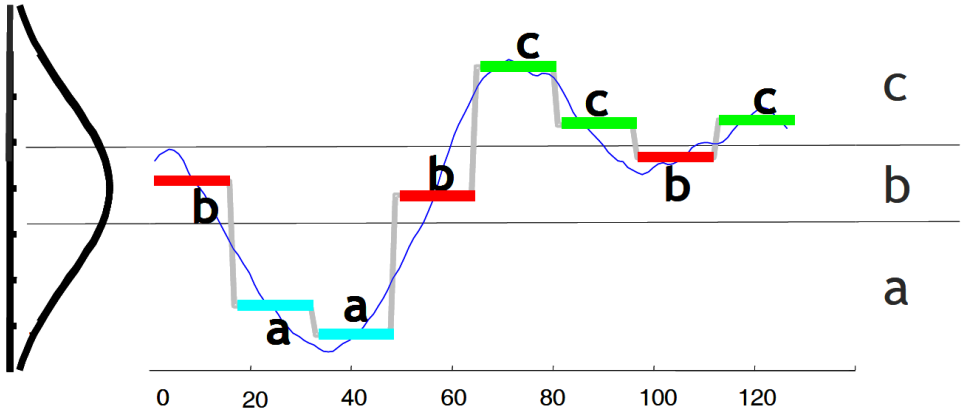
Discretize into a vector of symbols

- breakpoints map to small alphabet ***α*** of symbols



Symbolic ApproXimation (SAX)

- **Assumption:** normalized time series follow a Gaussian distribution
- Determine the breakpoints that will produce k equal-sized areas under the Gaussian curve

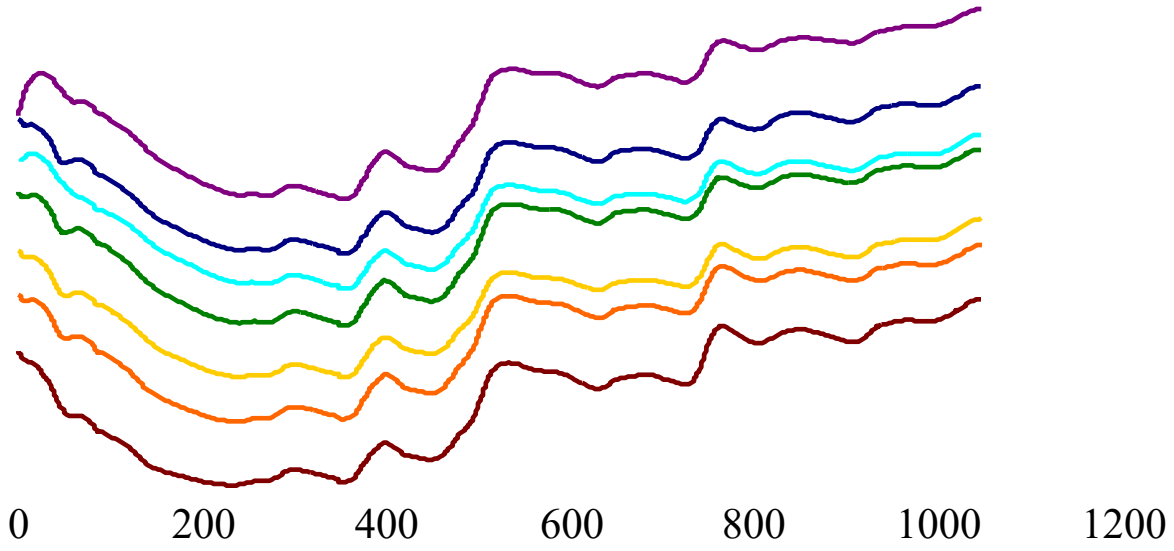


- Use a lookup table with breakpoints that divide a Gaussian distribution to an arbitrary number of equi-probable regions

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

Time Series classification

- Application: Finance, Medicine, Music



- 1-Nearest Neighbor
  - Use any of the existing distance measures, e.g., Euclidean, DTW, LCSS, or variants
  - **Pros:** accurate, robust, simple
  - **Cons:** time and space complexity (lazy learning); results are not interpretable

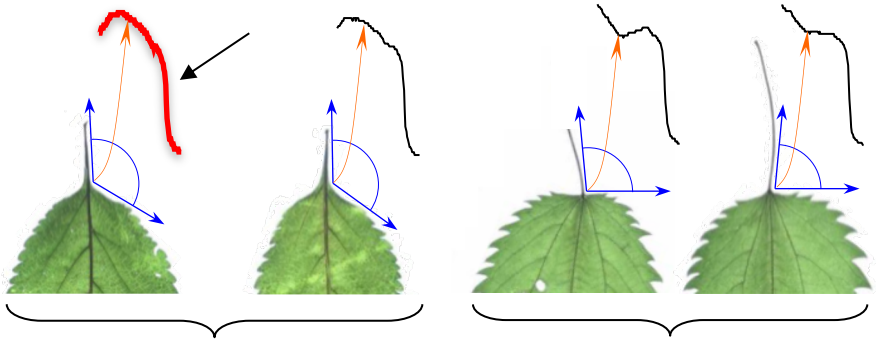
How about feature-based classification?

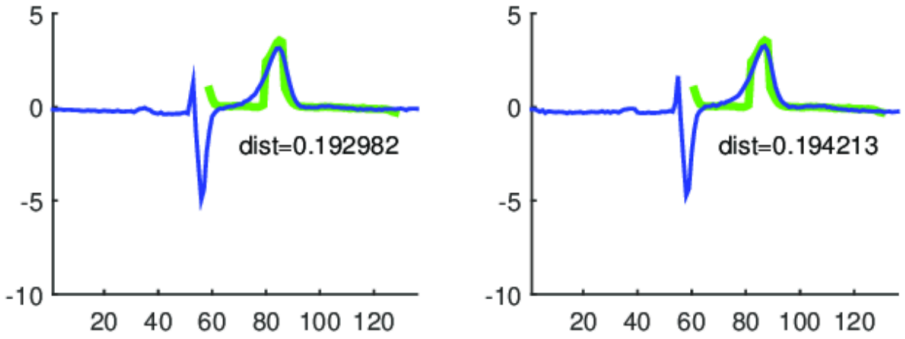
- Use **shapelets** as “attributes” or “features” for splitting a node in the decision tree

- **Shapelets:**
  - time series subsequence
  - *maximally representative* of a class
  - *discriminative* from other classes

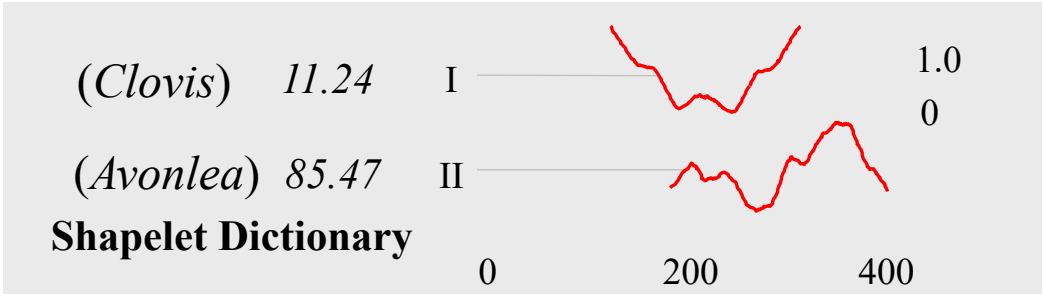
shapelet

Examples

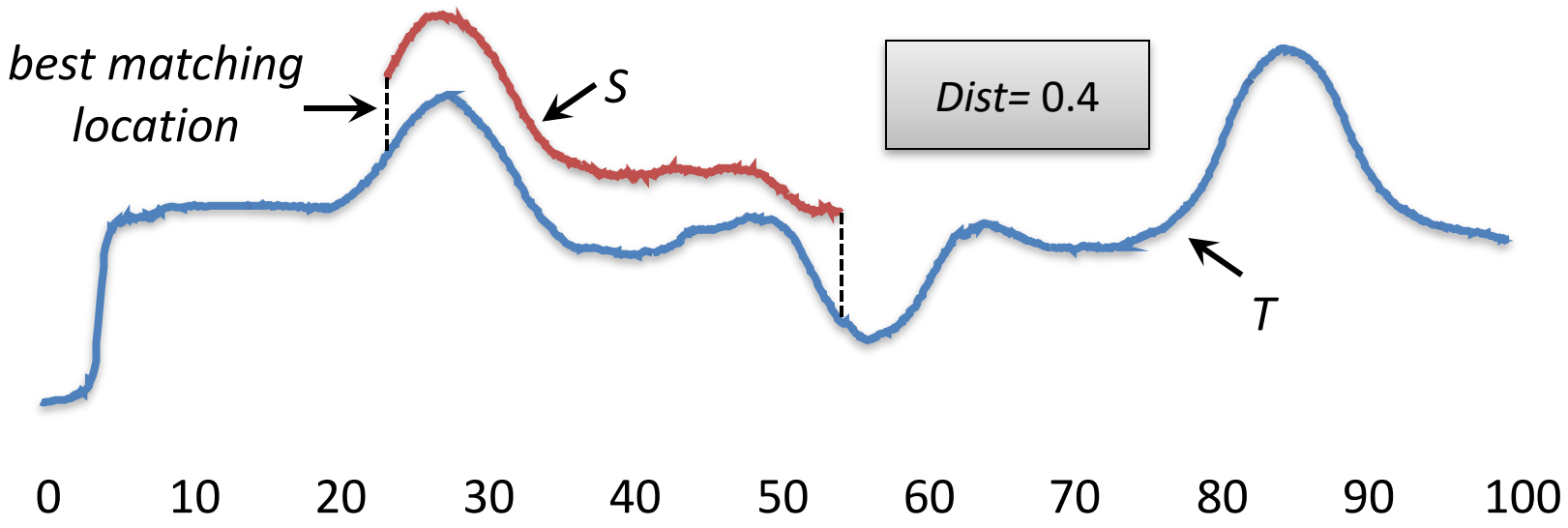

false nettles
stinging nettles


5
0
-5
-10
20 40 60 80 100 120
dist=0.192982
5
0
-5
-10
20 40 60 80 100 120
dist=0.194213


Clovis
Avonlea

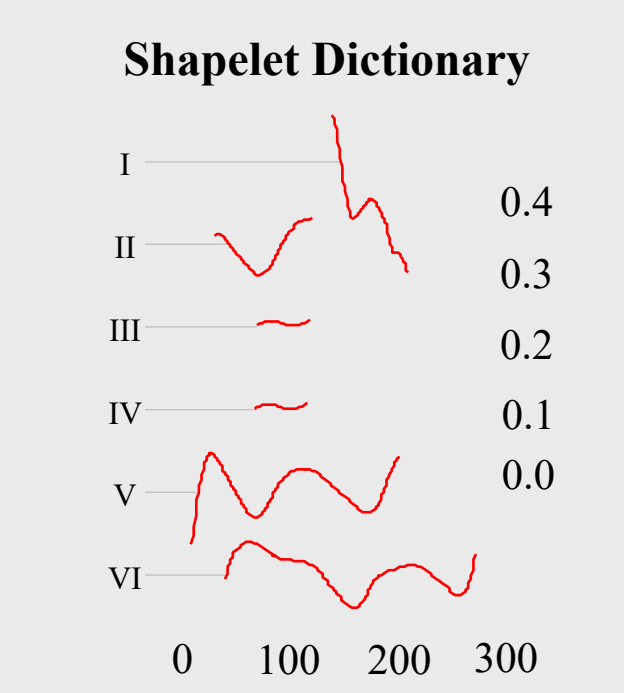

(Clovis) 11.24
I
(Avonlea) 85.47
II
Shapelet Dictionary
1.0
0
0
200
400

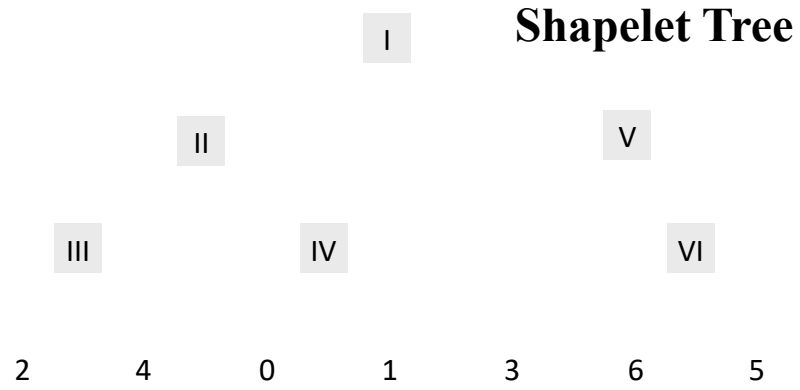
Distance of a shapelet S to a time series T



- Use a sliding window and run DTW, Euclidean distance, LCSS, etc
- Report the distance value of the best match (smallest value)

The Shapelet Tree Classifier





Method	Accuracy	Time
Shapelet	0.720	0.86
Nearest Neighbor	0.543	0.65

Alternative approaches

- Transformations + k-NN
  - Improved subsequence searching and matching, using online normalization, early abandoning, and re-ordering
  - Dimensionality reduction using SAX
- Synthetic shapelet generation
  - Initialize using K-means clustering
  - Learn synthetic Shapelets
- Shapelet-based features
  - Select the top k most informative shapelets as features
  - Learn any suitable classifier (e.g., SVM, Random Forest) using the transformed dataset

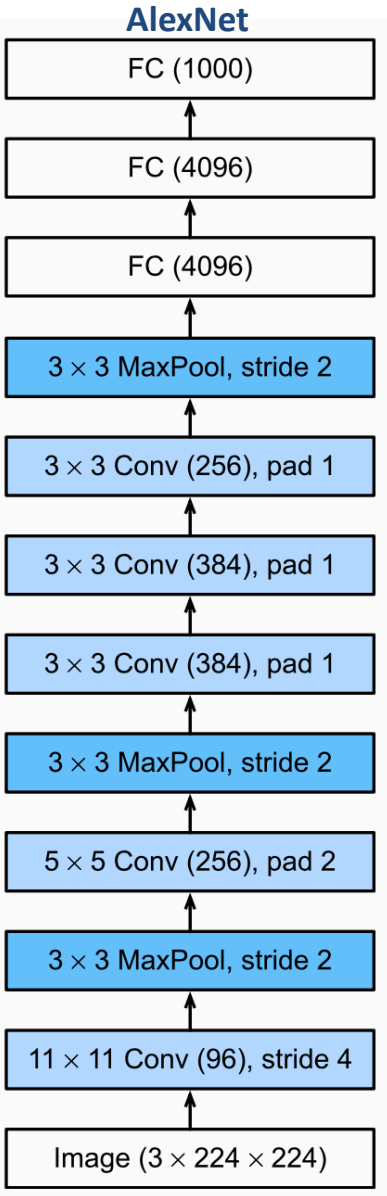
	s_1	s_2	...	s_k
d_1	0.3	3.3	...	0.1
d_2	0.2	3.2	...	3.8
⋮	⋮	⋮	⋮	⋮
d_n	3.1	0.9	...	9.6

Random Shapelet Forest

- A tree-structured ensemble based on the classic Random Forest and the Shapelet Tree classifier
- **Learn:**
  - Build ***T*** random shapelet trees
  - Each tree is built from a random (with replacement) sample of time series in the database D
  - Inspect ***r*** random shapelets at each node
- **Predict:**
  - Let each tree $t_1, \ldots, t_T$ vote for a class label

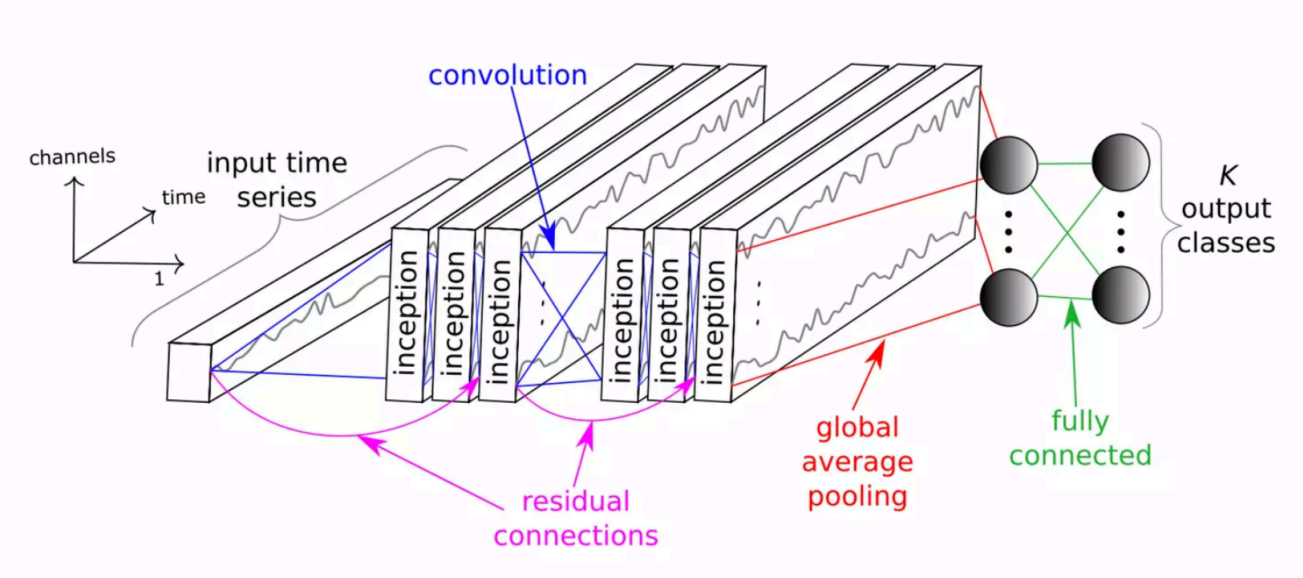
Inception Time [Fawaz 2020]

- The equivalent of **AlexNet** for time series
- An ensemble of five **deep learning models**
  - each created by cascading multiple Inception modules
  - each having exactly the same architecture but with different randomly initialized weight values
- **Core idea of an inception module:**
  - apply multiple filters simultaneously to an input time series
  - includes filters of varying lengths allowing the network to automatically extract relevant features from both long and short time series



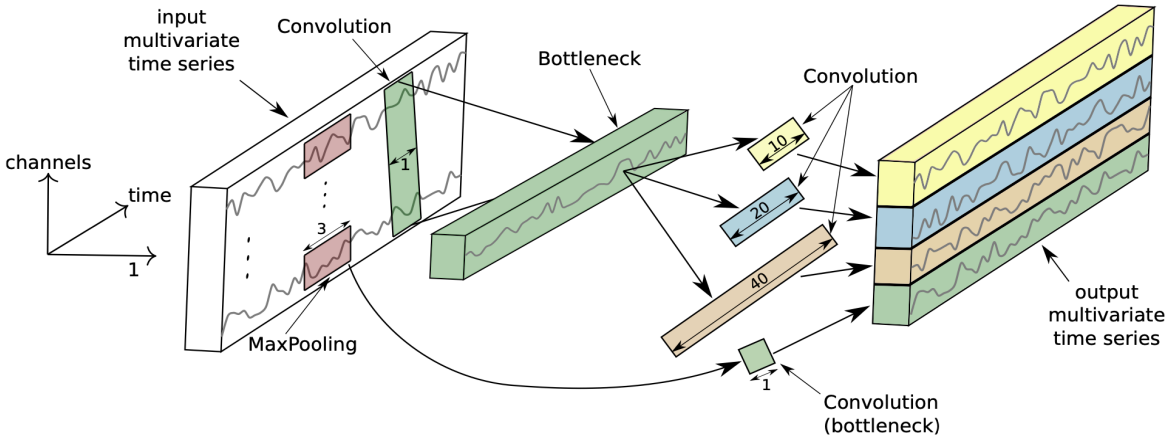
https://arxiv.org/abs/1909.04939

Inception Time [Fawaz 2020]



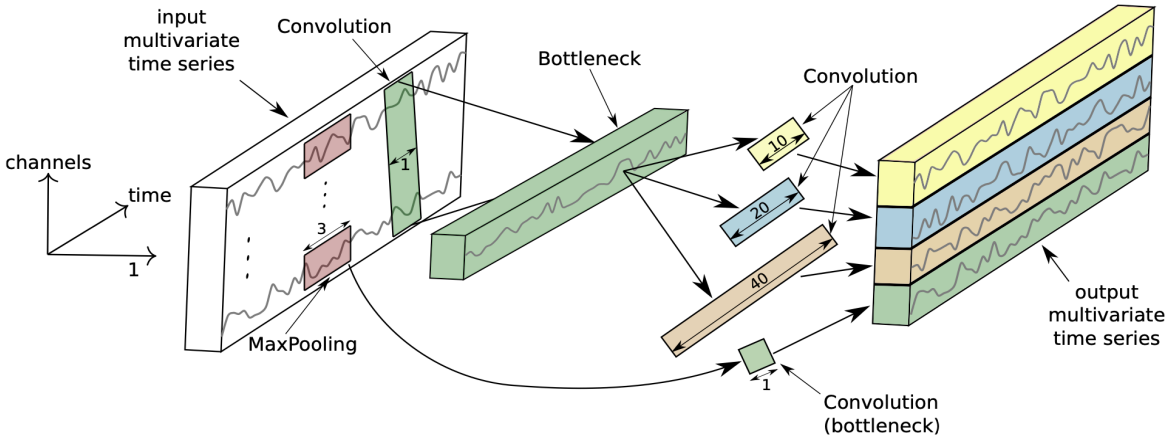
- Two residual blocks, each comprising three inception modules
- Each residual block's input is transferred via a linear connection to be added to the next block's input
- **Global Average Pooling (GAP) layer:** averages the output multivariate time series over the *whole time dimension*
- **Fully-connected softmax layer:** number of **neurons** equals the number of **classes**

Inception Time [Fawaz 2020]



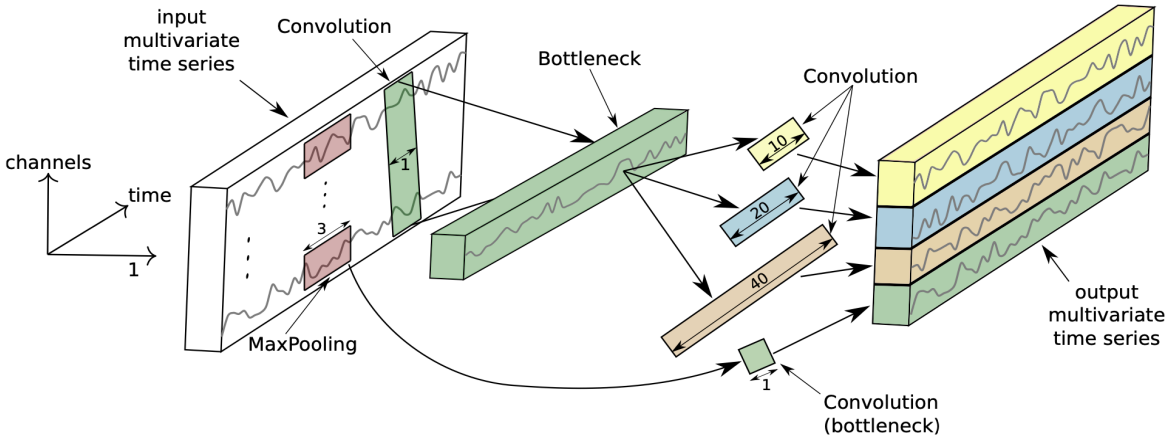
- The Inception module contains four layers:
  - Bootleneck Layer
  - A set of parallel Convolutional Layers with varying filter sizes
  - MaxPooling Layer
  - Depth Concatenation Layer
- The network can extract features of multiple resolutions using these filters

Inception Time [Fawaz 2020]



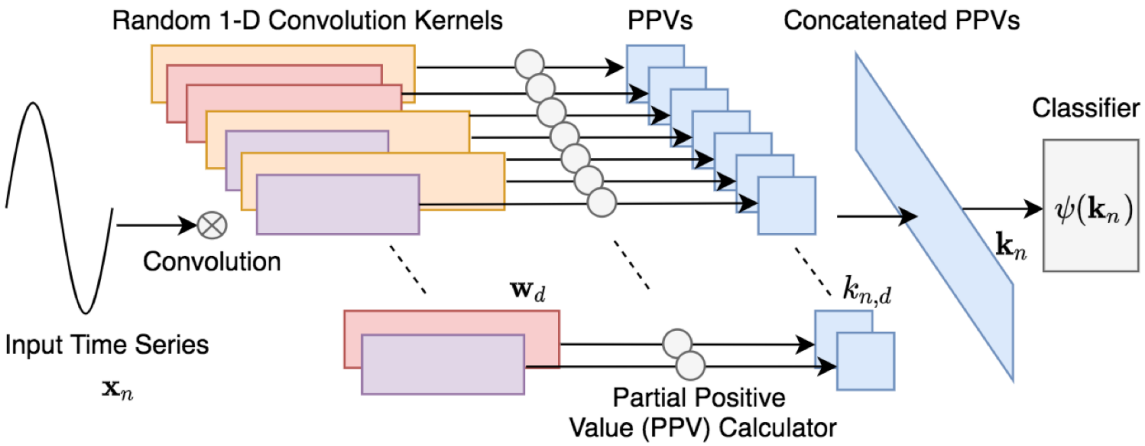
- Bootleneck Layer:
  - performs an operation of sliding m filters of *length 1* with a *stride of 1*
  - reduces the dimension of the original time series from M to **$m \ll M$**
- Multiple filters:
  - **three** different convolutions with lengths $l \in \{10, 20, 40\}$ are applied to the input time series, which is technically the output of the bottleneck layer

Inception Time [Fawaz 2020]



- MaxPooling Layer:
  - parallel MaxPooling followed by a bottleneck layer to reduce dimensionality
  - the output of a MaxPooling window is computed by taking the maximum value in this given window of time series
- Depth Concatenation Layer:
  - the output of each independent parallel convolution/MaxPooling is concatenated to form the output multivariate time series
  - the latter operations are repeated for each individual Inception module

Random Convolutional Kernel Transform (ROCKET)



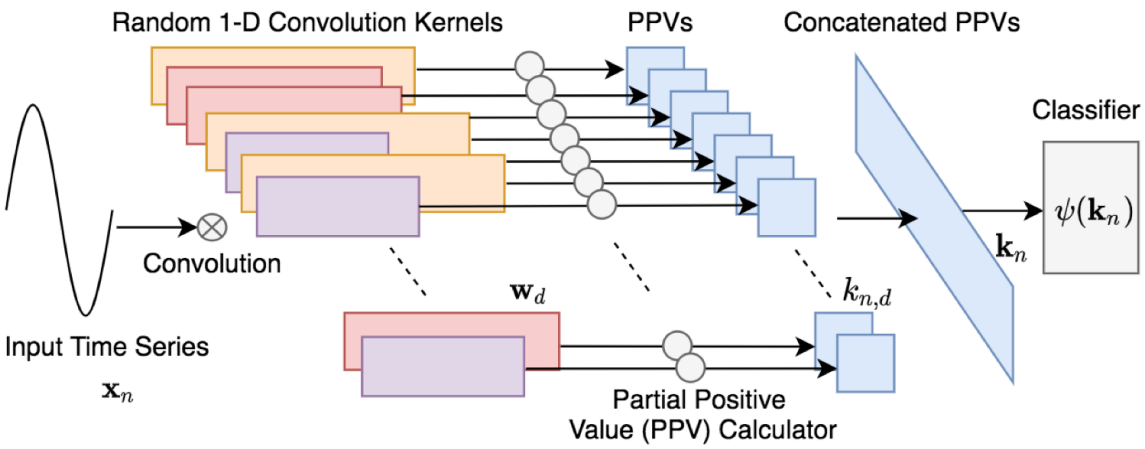
In short…

- **ROCKET** initializes a bank of random convolution kernels (e.g., 10 000)
- The convolution of each kernel with an input time series produces a feature vector
- Each feature vector is represented by the proportion of (or partial) positive values (**PPV**) and/or the maximum value (**max pooling**)
- The concatenation of PPV values from the kernels + the max pooling values is used as the input feature vector to train a Ridge regression classifier

https://arxiv.org/pdf/1910.13051.pdf

https://github.com/angus924/rocket

Small random convolution kernels

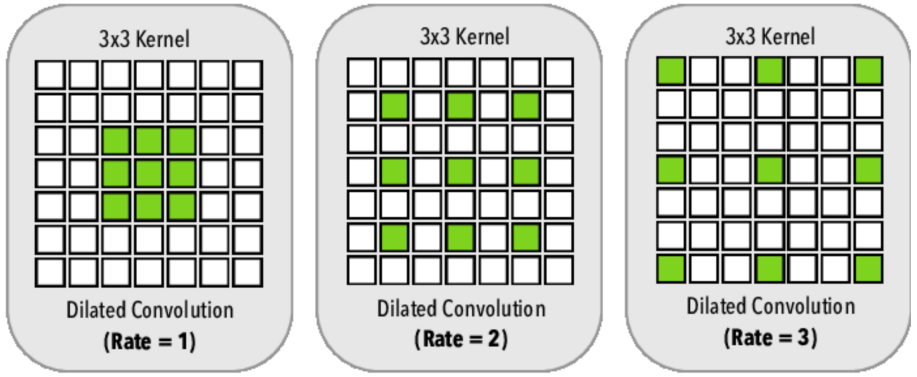


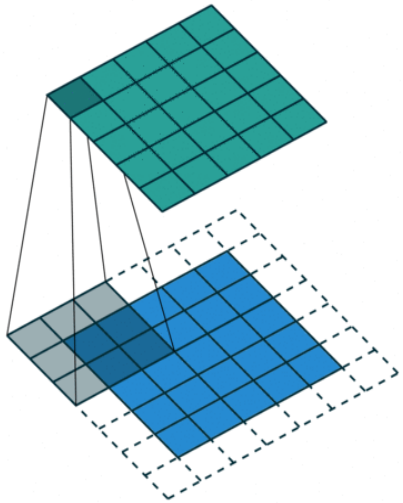
- Typical kernel size: {7, 9, 11}, selected randomly with equal probability
- Kernel weights are sampled from a normal distribution, $w \sim N(0,1)$, then mean centered: $w = W - \overline{W}$
- Bias $b \sim U(-1,1)$: the same kernel with different biases highlights different aspects of the resulting feature maps by shifting the values in a feature map above or below zero by a fixed amount

Dilated convolution

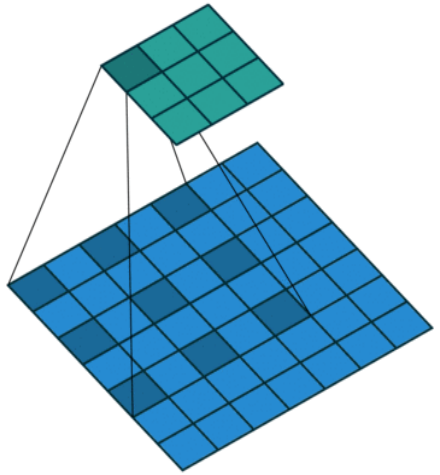
- A 3x3 kernel with a dilation rate of 2 will have the same field of view as a 5x5 kernel, while only using **9 parameters**
- Equivalent to a 5x5 kernel *"ignoring" every second column and row*
- Allows kernels with similar weigths to capture patterns at different scales, despite rescaling

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)\, k(t)$$





3x3 Kernel without dilation

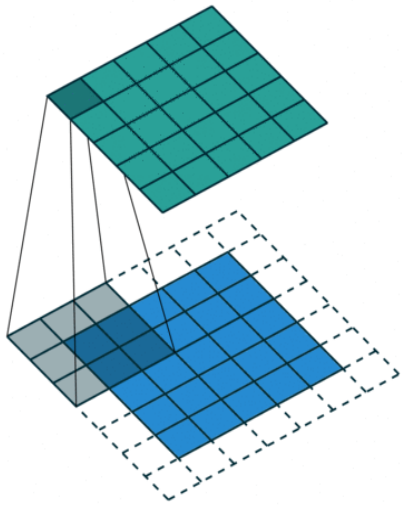


3x3 Kernel with dilation of 2

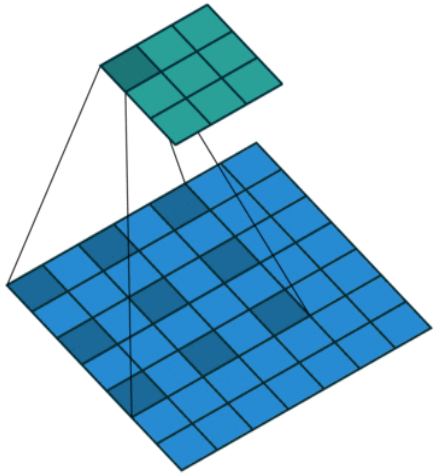
Dilated convolution – time series

- Frequency information is captured, since *large dilations* correspond to lower frequencies and *smaller dilations* to higher frequencies
- Kernels with similar weigths capture patterns at different scales
- Multiple kernels with different dilations can capture discriminative patterns despite complex warping

- Sampled from on an exponential scale, $d = \lfloor 2^x \rfloor, \quad x \sim U(0, A)$, with $A = log_2 \frac{l_{input}-1}{l_{kernel}-1}$
- l_{input}: length of the input time series
- l_{kernel}: length of the Kernel



3x3 Kernel without dilation



3x3 Kernel with dilation of 2

Padding

- When each kernel is generated, a decision is made (at random, with equal probability) whether or not padding will be used when the kernel will be applied
- An amount of zero padding is appended to the start and end of each time series, so that the *middle* element of the kernel is centered on every time series point

$$padding = \frac{(l_{kernel} - 1) \times d}{2}$$

- Why is padding useful?
  - without padding, kernels are not centered on the first and last $\lfloor l_{kernel}/2 \rfloor$ points of the time series; hence focusing only on central regions
  - with padding, kernels also match patterns at the start or end of time series

Final convolution features

- Each convolution

$$X_i * \omega = \sum_{j=0}^{l_{\text{kernel}}-1} X_{i+(j \times d)} \times \omega_j$$

  results into two feature values:

  - **max**: maximum value
  - **PPV**: proportion of positive values

The final feature space is fed into a linear classifier, i.e, a Ridge regressor or a Logistic regressor

- stride = 1 for all convolutions
- ReLU is not applied to the resulting feature maps

Final feature set: for **k kernels**, ROCKET produces **2k features per time series** (i.e., ppv and max). For example, for 10 000 kernels (the default), ROCKET produces 20 000 features.

Variants

- **Mini-ROCKET[1]**

	ROCKET	MINIROCKET
length	$\{7, 9, 11\}$	9
weights	$\mathcal{N}(0, 1)$	$\{-1, 2\}$
bias	$\mathcal{U}(-1, 1)$	from convolution output
dilation	random	fixed (rel. to input length)
padding	random	fixed
features	PPV + max	PPV
num. features	20K	10K

- **S-ROCKET[2]:**
  - optimized kernels: minimizes the number of active kernels
  - uses only 36-39% (on average) of the kernels to achieve similar classification accuracy as **mini-ROCKET** and **ROCKET**

1. https://arxiv.org/pdf/2012.08791.pdf

2. https://arxiv.org/pdf/2203.03445.pdf

Coming up next…

Feb 23	Laboratory session 5: Time series classification
Feb 27	Explainable machine learning
Mar 6	Reinforcement learning I
Mar 8	Reinforcement learning II
Mar 13	Exam review
Mar 20	**Examination**
Apr 28	**Re-take Examination**

TODOs

- **Reading:**
  - The Deep Learning Book: Chapter 20
  - Additional material: online
- **Homework 2**
  - Due: Feb 23