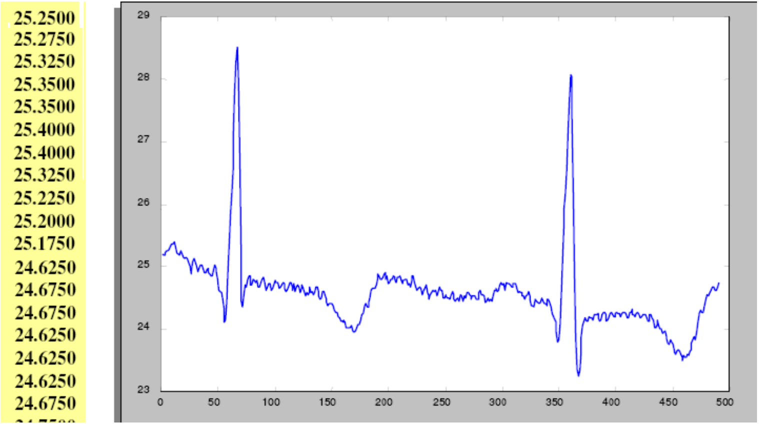
Time-Series Streams

João Gama
LIAAD-INESC Porto,
University of Porto, Portugal
jgama@fep.up.pt

Outline

1. Time-Series Analysis
2. Prediction
   - Filters
   - Neural Nets
3. Similarity between Time-series
   - Euclidean Distance
   - Dynamic Time-Warping
4. Symbolic Approximation – SAX
5. References

What are Time Series?



Thanks to Eamonn Keog

Introduction

- Time Series Analysis refers to applying different data analysis techniques on measurements acquired over temporal basis.
- Data analysis techniques recently applied on time series include clustering, classification, indexing, and association rules.
- The focus of classical time series analysis was on forecasting and pattern identification

Outline

Moving Averages

- Moving Average
  The mean of the previous n data points:

$$MA_t = MA_{t-1} - \frac{x_{t-n+1}}{n} + \frac{x_{t+1}}{n}$$

- Cumulative moving average
  the average of all of the data up until the current data point

$$CA_{i+1} = CA_i + \frac{x_{i+1} - CA_i}{i+1}$$

Weighted Moving Averages

- Weighted moving average
  has multiplying factors to give different weights to different data points

$$WMA_t = \frac{nx_t + (n-1)x_{t-1} + \cdots + 2x_{t-n+2} + x_{t-n+1}}{n + (n-1) + \cdots + 2 + 1}$$

- Exponential moving average
  The weighting for each older data point decreases exponentially, giving much more importance to recent observations while still not discarding older observations entirely.

$$S_t = \alpha \times Y_{t-1} + (1-\alpha) \times S_{t-1}$$

Filters

A filter is a device that removes from a signal some unwanted component or feature. The defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some frequencies and not others in order to suppress interfering signals and reduce background noise.

- low-pass filters
- high-pass filters

Low-pass Filters

A low-pass filter is a filter that passes low-frequency signals but attenuates (reduces the amplitude of) signals with frequencies higher than the cutoff frequency.

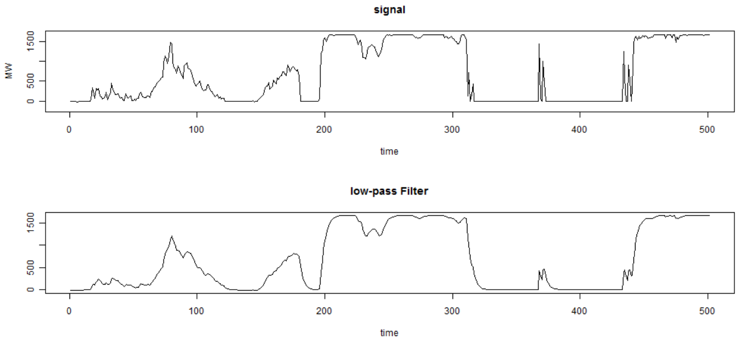
- function lowpass(x, α)
    - y[0] := x[0]
    - for i from 1 to n
        - $y[i] := \alpha * x[i] + (1 - \alpha) * y[i - 1]$
    - return y

Low-pass Filters

$$y_i = y_{i-1} + \alpha * (x_i - y_{i-1})$$

- The change from one filter output to the next is proportional to the difference between the previous output and the next input.
- This exponential smoothing property matches the exponential decay seen in the continuous-time system.
- As expected, as the smoothing parameter α decreases, the output samples respond more slowly to a change in the input samples: the system will have more inertia.

Low-pass Filters



High-pass Filters

A high-pass filter is a filter that passes high-frequency signals but attenuates (reduces the amplitude of) signals with frequencies lower than the cutoff frequency.

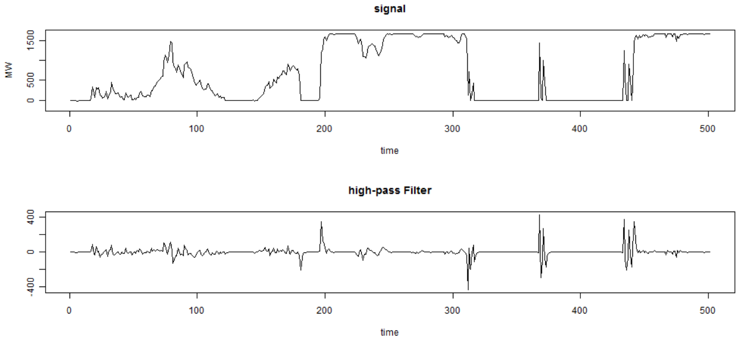
- function highpass(x, α)
  - y[0] := x[0]
  - for i from 1 to n
    - $y[i] := \alpha * y[i-1] + \alpha * (x[i] - x[i-1])$
  - return y

high-pass filters

$y_i = \alpha * (y_{i-1} + x_i - x_{i-1})$

- A large *alpha* implies that the output will decay very slowly but will also be strongly influenced by even small changes in input.
- a constant input (i.e., an input with $(x[i] - x[i-1]) = 0$) will always decay to zero,
- A small α implies that the output will decay quickly and will require large changes in the input (i.e., $(x[i] - x[i-1])$ is large) to cause the output to change much.
- it can only pass relatively high frequencies because it requires large (i.e., fast) changes and tends to quickly forget its prior output values.

High-pass Filters



Neural-Nets and Data Streams

Multilayer Neural Networks

- A general *function approximation* method;
- A 3 layer ANN can approximate any continuous function with arbitrary precision;

Training Neural Networks

- Scan training data several times
- Update weights after processing each example (or each epoch)

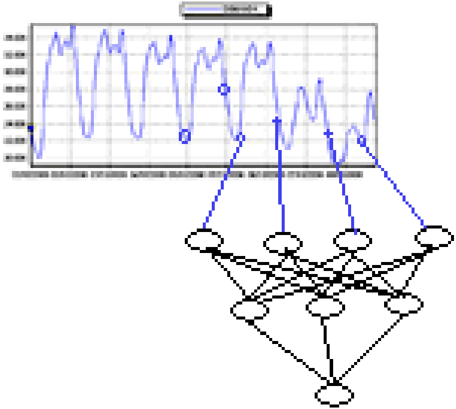
The only reason for multiple scans of training data is: lack of data – small training sets.

Neural-Nets and Data Streams

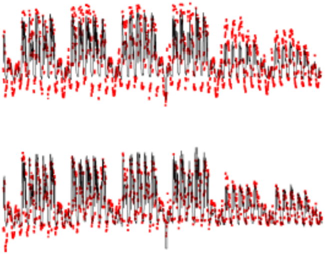
Neural Networks and Data Streams

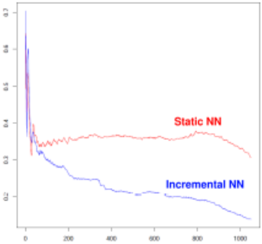
- Stochastic sequential train
- Fast train and prediction:
  - Each example is propagated once
  - The error is back-propagated once
- No Overfitting
  - First: prediction
  - Second: update the model
- Smoothly adjust to gradual changes

NN and Data Streams

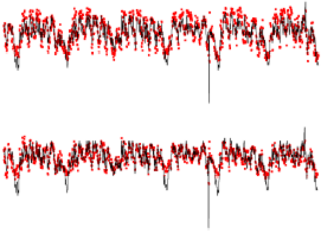


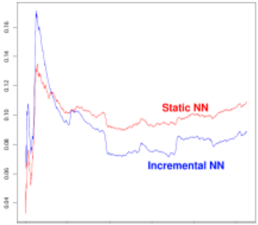
Illustrative Examples





Illustrative Examples





Load Forecast in Data Streams

The goal is to have an **any-time prediction** of the short term horizon forecast value for all sensors.

Three Horizon Forecasts: **next hour**, **next day** and **next week**. Although each model produces only **one output** prediction, after a short period (one week) the system has, at any-time:

- 1 prediction for the next hour;
- 24 predictions for the next day;
- 168 predictions for the next week;

Load Forecast in Data Streams

MLP Topology: 10 inputs, 4 hidden neurons (tanh-activated) and a linear output.

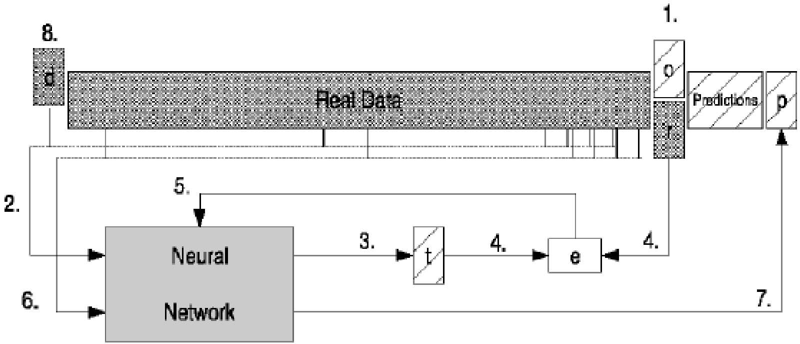
The input vectors for predicting time k depend on the predictive horizon:

- One hour:
  - k minus $\{1, 2, 3, 4\}$ hours and
  - k minus $\{7, 14\}$ days.
- One day:
  - k minus $\{1, 6, 7\}$ days,
  - k minus 7 days and 1 hour, and
  - k minus $\{8, 14\}$ days.
- One week:
  - k minus $\{7\}$ days,
  - k minus 7 days and 1 hour, and
  - k minus $\{13, 14, 15, 21\}$ days.

The remaining inputs for each model are fed with 4 cyclic variables, for hourly and weekly periods (*sin* and *cos*) identification.

Buffering the Input Data and the Predictions

Online prediction and training is achieved with **buffered input**.



Variance Reduction

MLP's are prone to high-Variance error.
Sensor data is noisy and uncertain

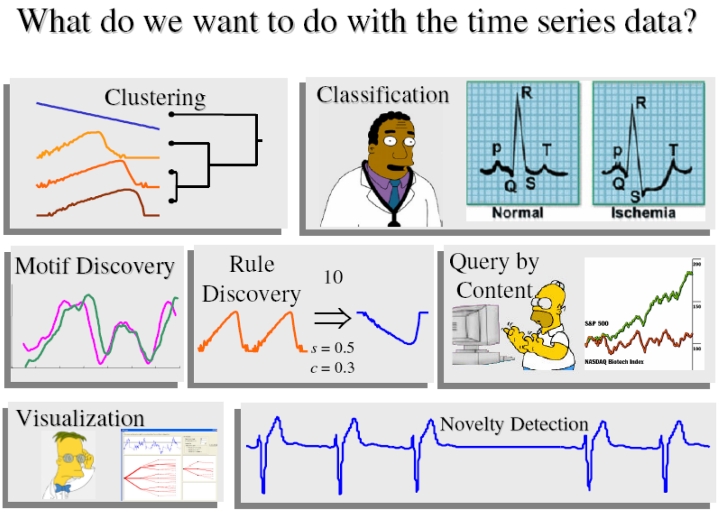
Dual perturb and combine

Perturb test examples several times: add white noise to the attribute-values.
For each perturb version the MLP makes a prediction
Final prediction obtained by aggregating these predictions.

Advantages:

Works online
Efficient variance reduction method

Outline



Thanks to Eamonn Keog

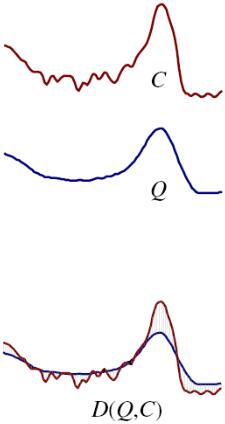
Similarity

- Similarity measures over time series data represent the main step in time series analysis.
- Euclidean and **dynamic time warping** represent the major similarity measures used in time series.
- Longer time series could be represent computationally hard for the analysis tasks.
- Different time series representations have been proposed to reduce the length of a time series.

Euclidean Distance

Given two time series:

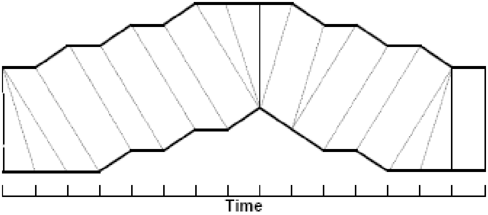
- $Q = q_1, q_2, \ldots, q_n$
- $S = s_1, s_2, \ldots, s_n$
- $D(Q, S) = \sqrt{\sum_{i=1}^{n} (q_i - s_i)^2}$



Dynamic Time-Warping

Dynamic time warping is an algorithm for measuring similarity between two sequences which may vary in time or speed.

- DTW is a method to find an optimal match between two given sequences (e.g. time series) with certain restrictions.
- The optimization process is performed using dynamic programming
- The problem for one-dimensional time series can be solved in polynomial time.



Problem Formulation

S. Stan & P. Chan. *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*, Intelligent Data Analysis, 2007

The dynamic time warping problem is stated as follows:

- Given two time series $X = x_1, x_2, \ldots, x_n$, and $Y = y_1, y_2, \ldots, y_p$, of lengths n and p
- construct a warp path $W = w_1, w_2, \ldots, w_K$
- K is the length of the warp path and
  - $max(n, p) \leq K < n + p$
  - the k^{th} element of the warp path is $w_k = (i, j)$ where
    - i is an index from time series X,
    - j is an index from time series Y.

Problem Formulation

- The warp path must start at the beginning of each time series at $w_1 = (1, 1)$
- and finish at the end of both time series at $w_K = (n, p)$.
- This ensures that every index of both time series is used in the warp path.
- Another constraint on the warp path that forces i and j to be monotonically increasing in the warp path.

Problem Formulation

The optimal warp path is the minimum-distance warp path, where the distance of a warp path W is

$$Dist(W) = \sum_{k=1}^{k=K} dist(w_{ki}, w_{kj})$$

where $dist(w_{ki}, w_{kj})$ is the Euclidean distance between the two data point indexes (one from X and one from Y) in the k^{th} element of the warp path.

Dynamic Time-Warping

W = (1,1), (2,1),(3,1), (4,2), (5,3), (6,4), (7,5), (8,6), (9,7), (9,8), (9,9), (9,10), (10,11), (10,12), (11,13), (12,14), (13,15), (14,15), (15,15), (16,16).

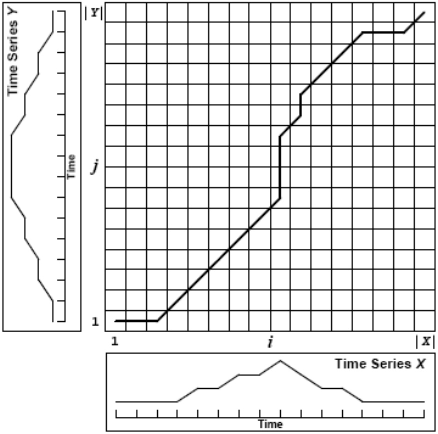


Figure 2. A cost matrix with the minimum-distance warp path traced through it.

Dynamic Time-Warping

- Vertical sections of the warp path, means that a single point in time series X is warped to multiple points in time series Y,
- Horizontal sections means that a single point in Y is warped to multiple points in X
- Since a single point may map to multiple points in the other time series, the time series do not need to be of equal length.
- If X and Y were identical time series, the warp path through the matrix would be a straight diagonal line.

Filling the Cost Matrix

- To find the minimum-distance warp path, every cell of the cost matrix must be filled.
- The value of a cell in the cost matrix is:
  $D(i,j) = Dist(i,j) + min[D(i-1,j), D(i,j-1), D(i-1,j-1)]$
- The cost matrix is filled one column at a time from the bottom up, from left to right
- After the entire matrix is filled, a warp path must be found from D(1, 1) to D(n, p).

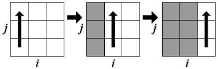


Figure 3. The order that the cost matrix is filled.

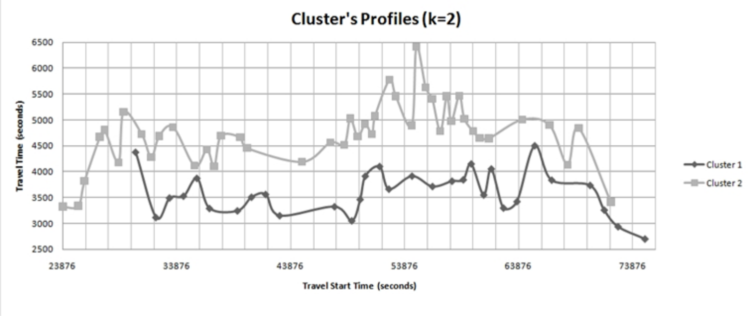
DTW Algorithm

Function DTWDistance(s[1..n], t[1..m])

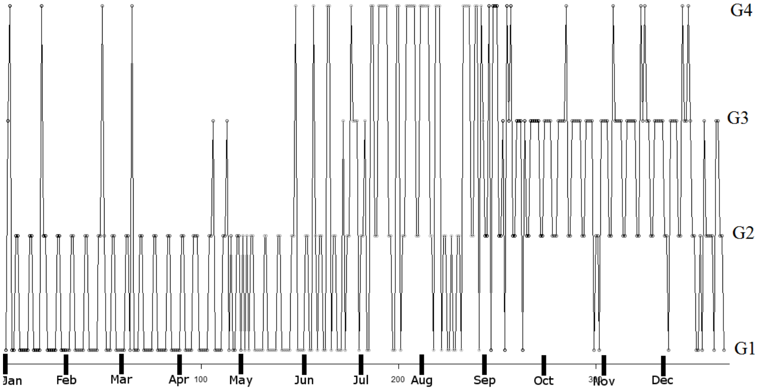
- int DTW[0..n, 0..m]
- int i, j, cost
- for i := 1 to m DTW[0, i] := infinity
- for i := 1 to n DTW[i, 0] := infinity
- DTW[0, 0] := 0
- for i := 1 to n
  - for j := 1 to m
    - cost:= d(s[i], t[j])
    - DTW[i, j] := cost + minimum(
      DTW[i-1, j], // insertion
      DTW[i , j-1], // deletion
      DTW[i-1, j-1]) // match

return DTW[n, m]

Clustering Bus Travel times



Clustering Bus Travel times



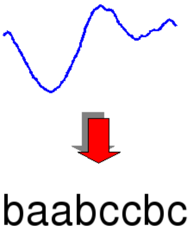
- G1: working days from January 1st to July 15th (beginning of the school holiday period)
- G2: all non-working days including holidays and all weekends.
- G3: working days from September 15th to December 31st
- G4: working days from July 15th to September 15th (school holiday period)

Outline

Symbolic ApproXimation – SAX

J. Lin, E. Keogh, S. Lonardi, and B. Chiu, *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*, 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2003.

- SAX is a fast symbolic approximation of time series.
- It allows a time series with a length n to be transformed to an approximated time series with an arbitrarily length w, where $w < n$.
- SAX follows three main steps:
  - Piecewise Aggregate Approximation (PAA)
  - Symbolic Discretization
  - Distance measurement
- SAX is generic and could be applied to any time series analysis technique.
- It takes linear time

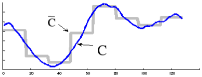


Piecewise Aggregate Approximation (PAA)

A time series with size n is approximated using PAA to a time series with size w using the following equation:

$$\bar{c}_i = \frac{w}{n} \sum_{j=\frac{w}{n}(i-1)+1}^{\frac{w}{n}i} c_j$$

Where $\bar{c}_i$ is the i^{th} element in the approximated time series.



Symbolic Discretization

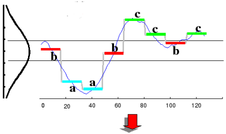
Requires two parameters:

- Cardinality of the alphabet of symbols.
- Number of episodes

Breakpoints are calculated that produce equal areas from one point to another under Gaussian distribution.

According to the output of PAA:

- If a point is less than the smallest breakpoint, then it is denoted as *a*.
- Otherwise and if the point is greater than the smallest breakpoint and less than the next larger one, then it is denoted as *b*.
- etc.



baabccbc

Distance Measure

- The following distance measure is applied when comparing two different time series:
$MINDIST(\hat{Q}, \hat{C}) = \sqrt{\frac{n}{w}}\sqrt{\sum_1^w dist(\hat{q}_i, \hat{c}_i)^2}$
- It returns the minimum distance between the original time series.
- A lookup table is calculated and used to find the distance between every two letters.

SAX

SAX lets us do things that are difficult or impossible with other representations.

- Finding motifs in time series (ICDM 02, SIGKDD 03)
- Visualizing massive time series (SIGKDD04, VLDB 04)
- Cluster from streams (ICDM 03, KAIS 04)
- Kolmogorov complexity data mining (SIGKDD 04)
- Finding discords in time series (ICDM 05)

Motifs

previously unknown, frequently occurring [sequential] patterns within same sequence.

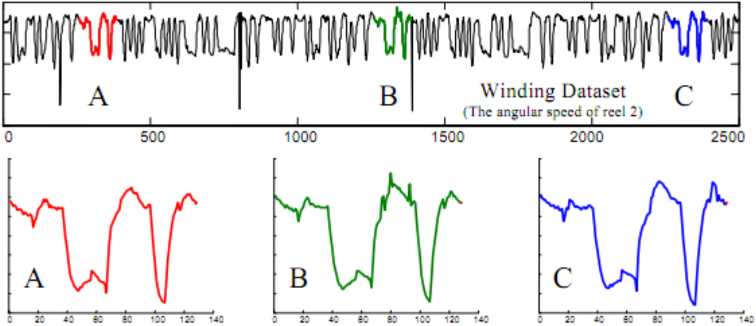


Figure 1: *Above*) An example of a motif that occurs three times in a complex and noisy industrial dataset. *Below*) a zoom-in reveals just how similar the three occurrences are to each other

HOT–SAX

SAX has been used to discover *discords* in time series. The technique is termed as *Hot SAX*.

Keogh, E., Lin, J. and Fu, A., *HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence*. In the 5th IEEE International Conference on Data Mining, New Orleans, LA. Nov 27-30, 2005.

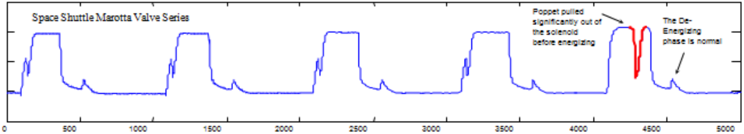
- *Discords* are the time series subsequences that are maximally different from the rest of the time series subsequences.
- It is 3 to 4 times faster than brute force technique.
- This makes it a candidate for data streaming applications

HOT–SAX

- The process starts with sliding widows of a fixed size over the whole time series to generate subsequence
- Each generated subsequence is approximated using SAX
- The approximated subsequence is then inserted in an array indexed according to its position in the original time series
- The number of occurrences of each SAX word is also inserted in the array.
- The array is then transformed to a tries where the leaf nodes represent the array index where the word appears.
- The two data structures (array and trie) complement each other.

Discords

time series subsequences that are maximally different from the rest of the time series subsequences.



In this case the anomaly is very obvious, and the discord (marked in red) easily finds it.

Outline

Resources

- UCR Time-Series Data Sets
  - Maintained by Eamonn Keogh, UCR, US
  - http://www.cs.ucr.edu/~eamonn/time_series_data
- Mining Data Streams Bibliography
  - Maintained by Mohamed Gaber
  - http://www.csse.monash.edu.au/ ~mgaber/WResources.html

Master References

- J. Gama, *Knowledge Discovery from Data Streams*, CRC Press, 2010.
- J. Gama, M. Gaber (Eds), *Learning from Data Streams – Processing Techniques in Sensor Networks*, Springer, 2007.
- J. Lin, E. Keogh, S. Lonardi, and B. Chiu,*A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*, ACM SIGMOD Workshop, 2003.
- Keogh, Pazzani. *Scaling up Dynamic Time Warping for Datamining Applications*. ACM SIGKDD, 2000
- A. Camerra, Jin Shieh, T. Palpanas, T. Rakthanmanon, E. Keogh: *Beyond one billion time series: indexing and mining very large time series collections with i SAX2+0*. Knowl. Inf. Syst. 39(1): 123-151 (2014)
- S. Stan & P. Chan. *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*, Intelligent Data Analysis, 2007.
- Keogh, E., Lin, J. and Fu, A., *HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence*. In IEEE ICDM, 2005.
- N. Begum, E. Keogh: *Rare Time Series Motif Discovery from Unbounded Streams*. PVLDB 8(2): 149-160 (2014)