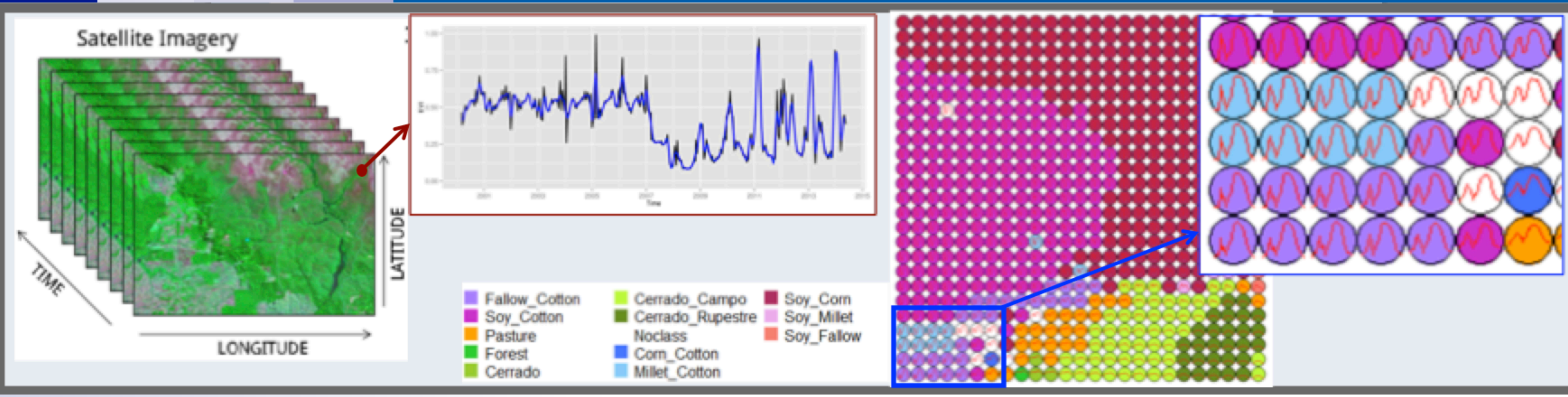
MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Time Series Clustering using SOM



Karine Ferreira e Lorena Santos

Agosto de 2018

http://wiki.dpi.inpe.br/doku.php?id=cap241:2018:tsclustering

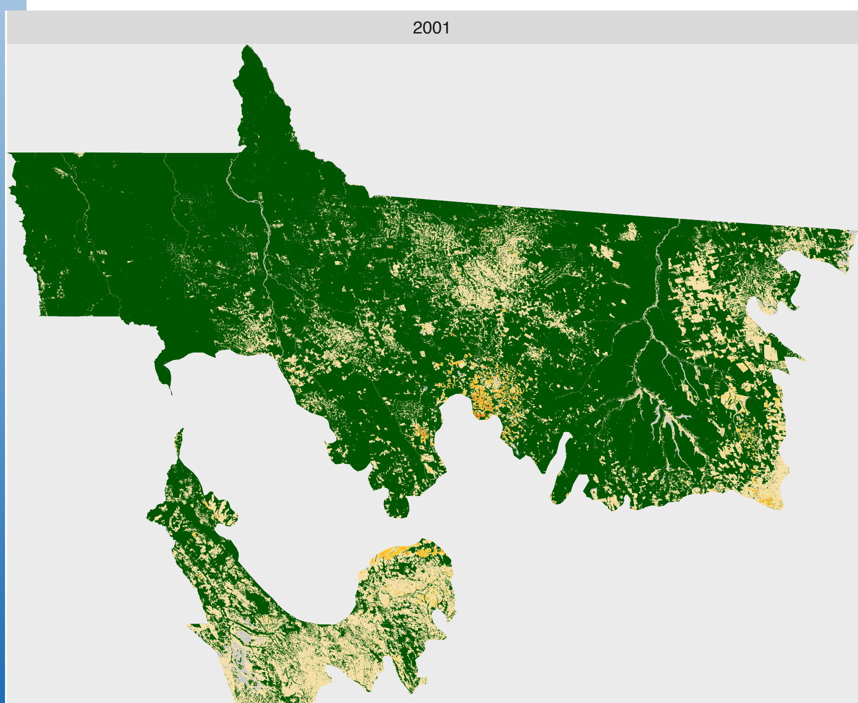
Motivation – E-sensing Project

Extract information on land use and cover change (LUCC) from big Earth observation (EO) data sets.

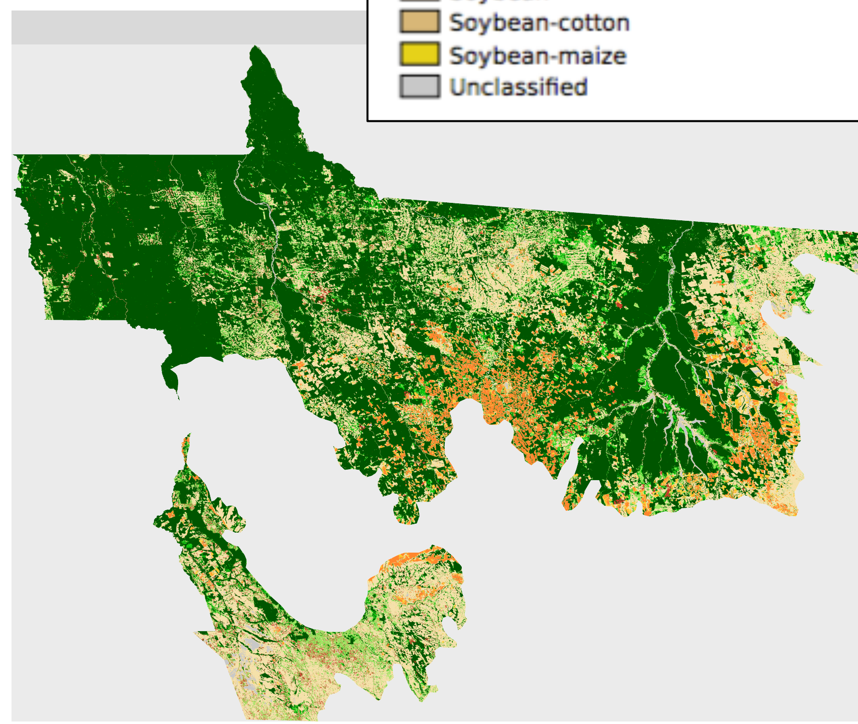
Example - Land cover change trajectories in the Amazonian biome of Mato Grosso state - Brazil (2001-2014)

Classes of land use

Class
- Primary forest
- Deforestation/Forest degradation
- Post-extraction/fire secondary forest
- Rehabilitated secondary forest
- Pasture
- Cotton
- Soybean
- Soybean-cotton
- Soybean-maize
- Unclassified



2001

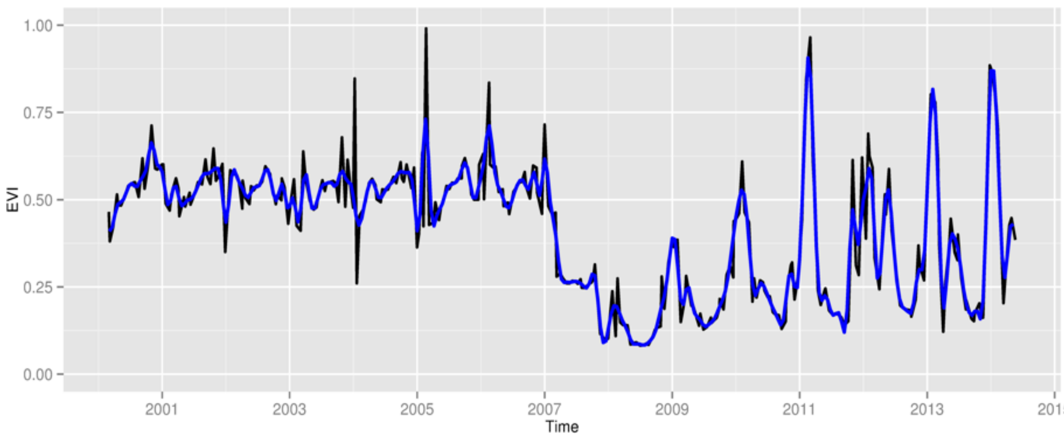


2014

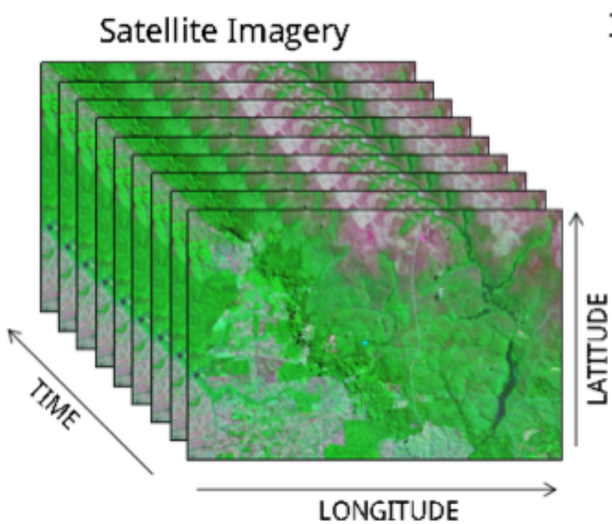
graphics: Victor Maus (INPE, IFGI)

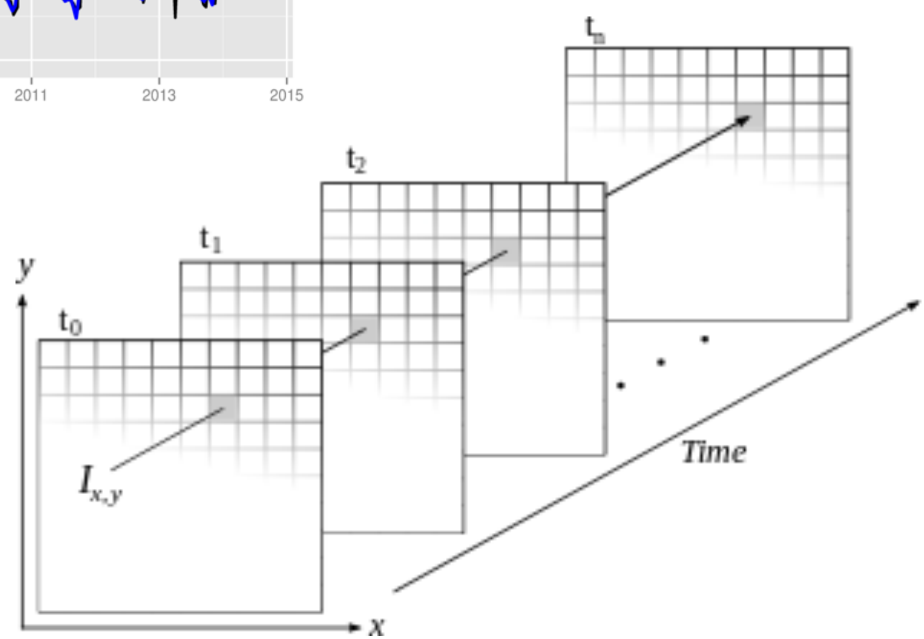
Motivation – E-sensing Project

How to create land use and cover change (LUCC) maps from satellite image time series ?

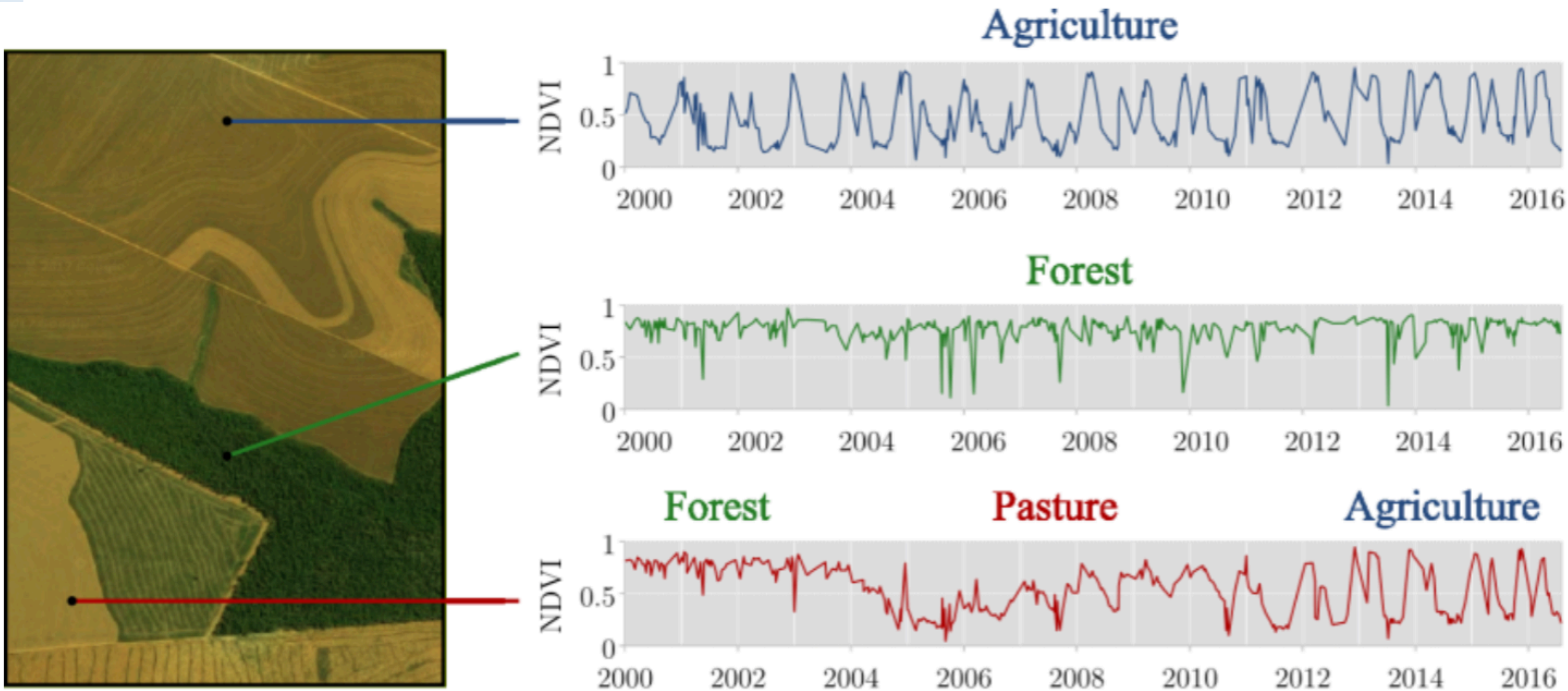


Vegetation Indices (e.g. EVI and NDVI) characterize vegetation dynamics across different temporal scales (FENSHOLT et al., 2015).





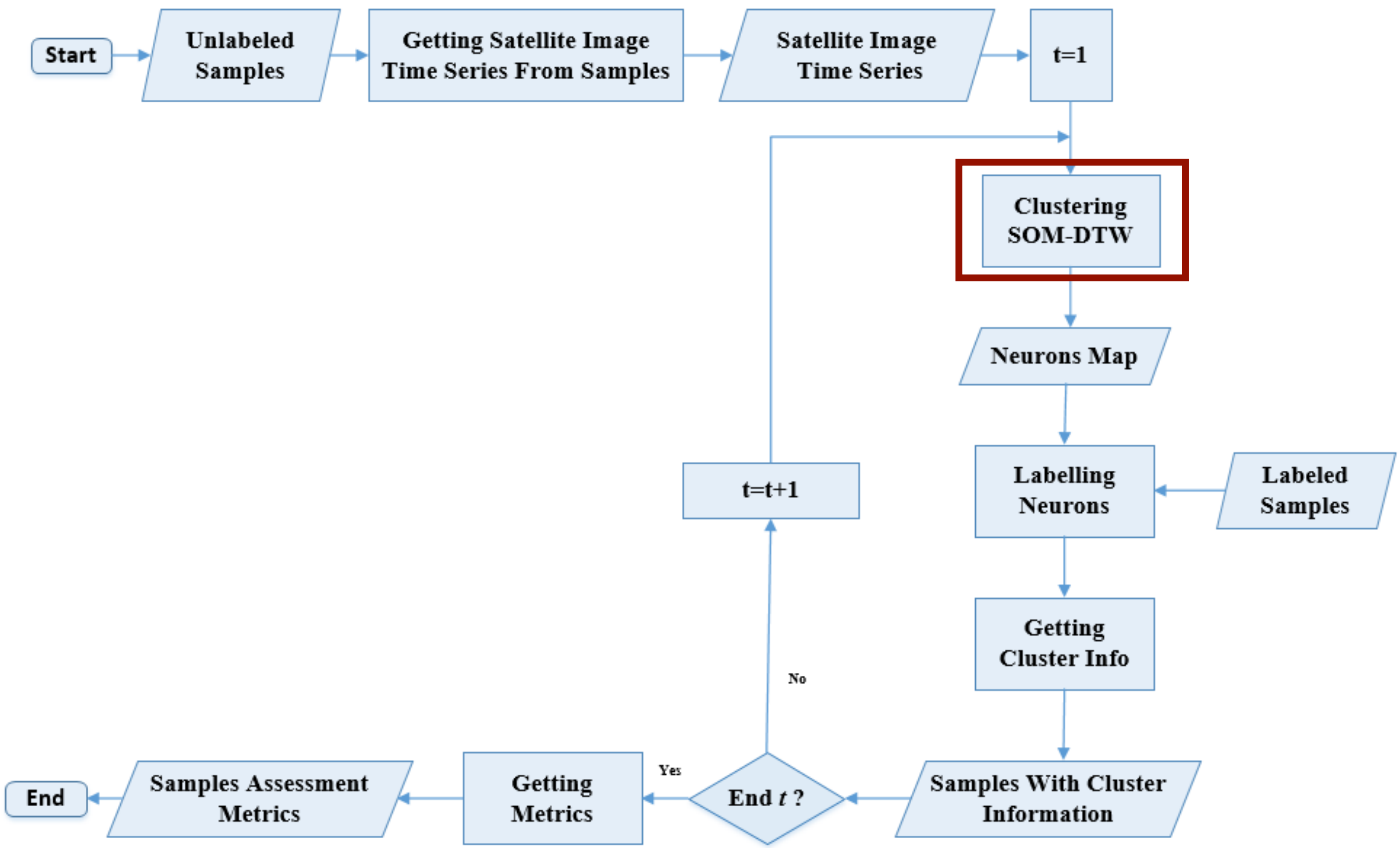
Samples of land use and cover change (LUCC)



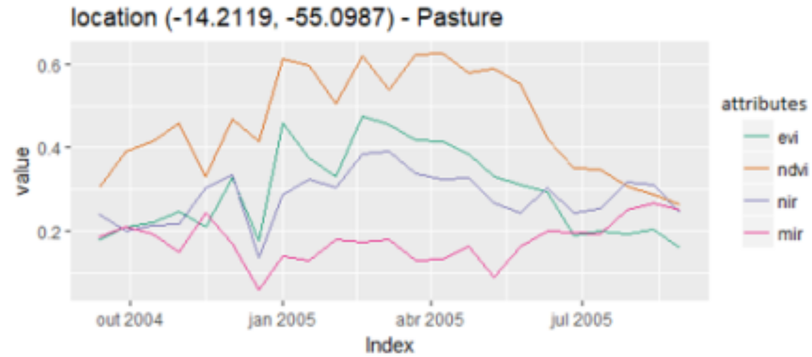
Land Trajectory

"The transformations of land cover due to actions of land use" (Camara, 2017). Adapted from: Maus, V. (IIASA, INPE)

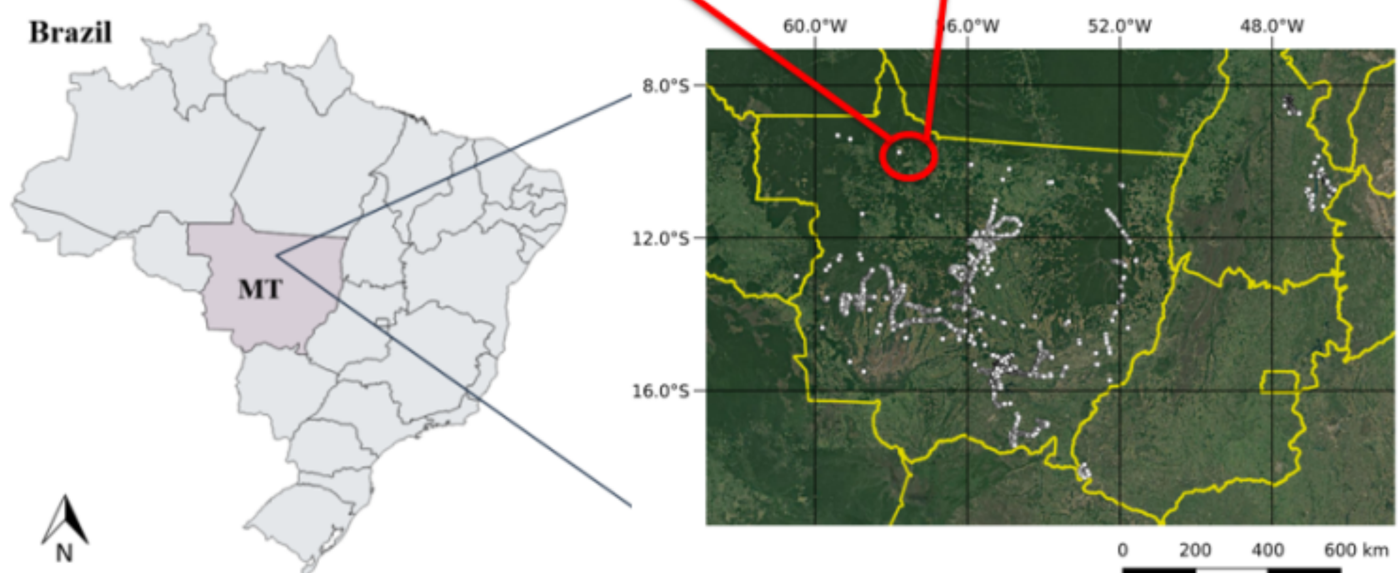
A method to assess LUCC samples from satellite image time series



[Lorena Alves, 2018, "A method to assess LUCC samples from satellite image time series "]

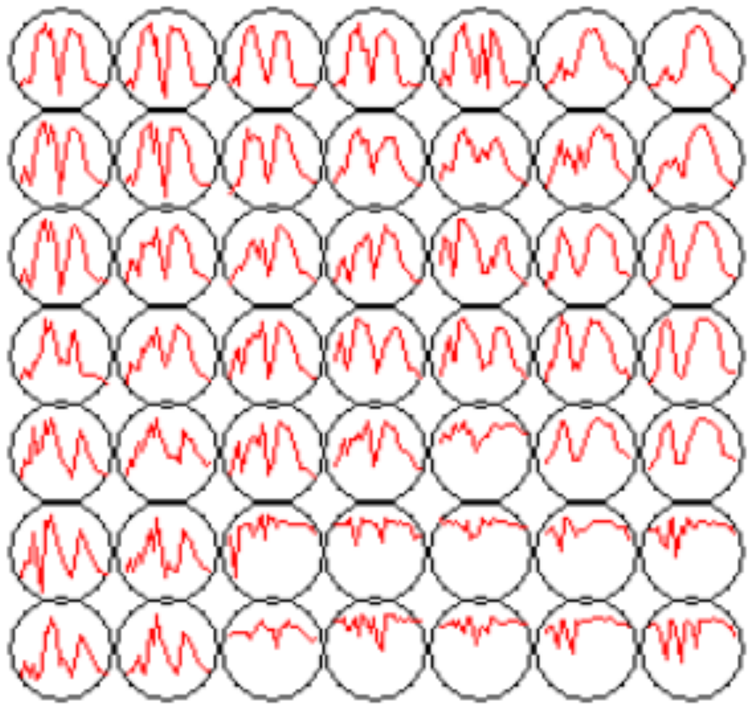


id	longitude	latitude	start_date	end_date	label
1	-55.1852	-10.8378	2013-09-14	2014-08-29	Pasture
2	-57.7940	-9.7573	2006-09-14	2007-08-29	Pasture
3	-51.9412	-13.4198	2014-09-14	2015-08-29	Pasture
4	-55.9643	-10.0621	2005-09-14	2006-08-29	Pasture



[Lorena Alves, 2018, “A method to assess LUCC samples from satellite image time series ”]

A method to assess LUCC samples from satellite image time series



SOM – Output neurons – Clusters

[Lorena Alves, 2018, “A method to assess LUCC samples from satellite image time series ”]

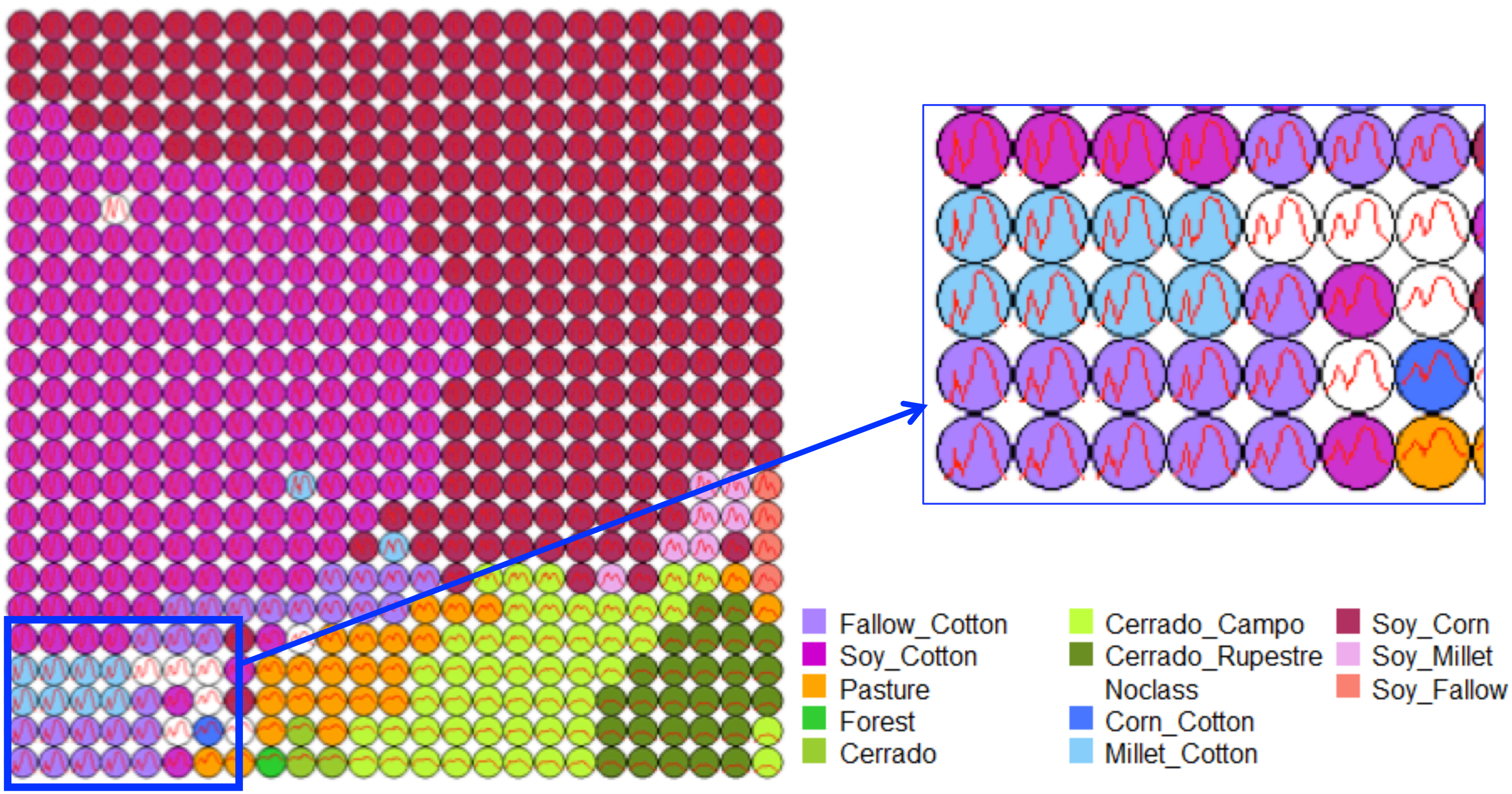
A method to assess LUCC samples from satellite image time series

neighboring neurons are similar!

SOM – Output neurons – Clusters

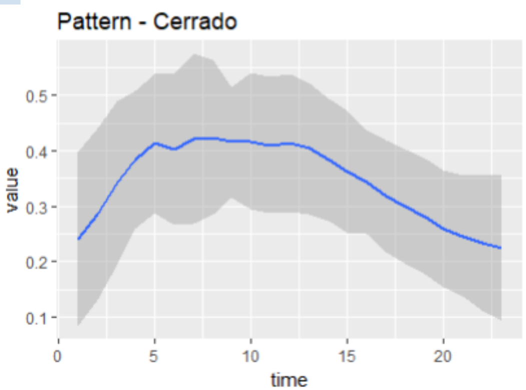
[Lorena Alves, 2018, "A method to assess LUCC samples from satellite image time series "]

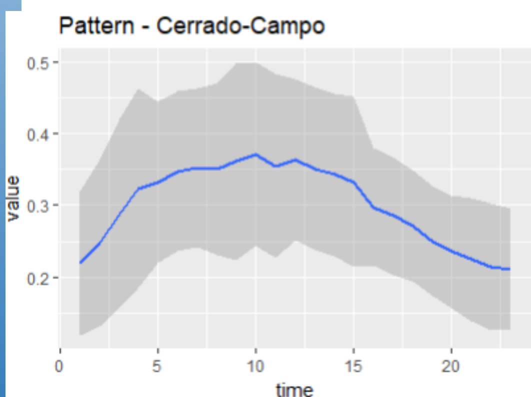
A method to assess LUCC samples from satellite image time series



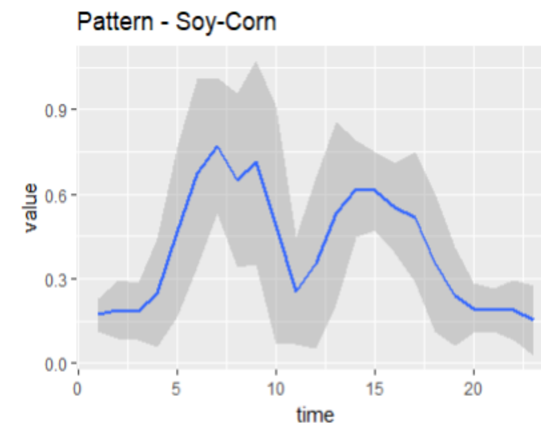
[Lorena Alves, 2018, “A method to assess LUCC samples from satellite image time series ”]

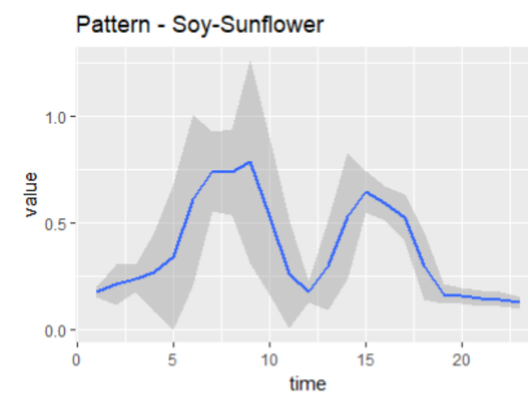
A method to assess LUCC samples from satellite image time series





	Id_class	class	Label	percentage_class
1	1	Cerrado	Cerrado_Campo	65.75000000
2	1	Cerrado	Forest	11.75000000
3	1	Cerrado	Cerrado	8.25000000
4	1	Cerrado	Cerrado_Rupestre	7.50000000
5	1	Cerrado	Pasture	6.75000000
6	2	Cerrado_Campo	Cerrado_Campo	94.50199203
7	2	Cerrado_Campo	Cerrado_Rupestre	4.38247012
8	2	Cerrado_Campo	Pasture	0.87649402
9	2	Cerrado_Campo	Cerrado	0.23904382
10	3	Cerrado_Rupestre	Cerrado_Rupestre	76.21440536
11	3	Cerrado_Rupestre	Cerrado_Campo	23.78559464
12	4	Corn_Cotton	Soy_Cotton	97.22222222
13	4	Corn_Cotton	Corn_Cotton	2.77777778
14	5	Fallow_Cotton	Fallow_Cotton	93.26315789
15	5	Fallow_Cotton	Soy_Cotton	4.42105263
16	5	Fallow_Cotton	Millet_Cotton	1.47368421
17	5	Fallow_Cotton	Cerrado_Campo	0.63157895
18	5	Fallow_Cotton	Cerrado_Rupestre	0.21052632
58	13	Soy_Sunflower	Soy_Corn	82.75862069
59	13	Soy_Sunflower	Soy_Cotton	13.79310345
60	13	Soy_Sunflower	Soy_Fallow	3.44827586

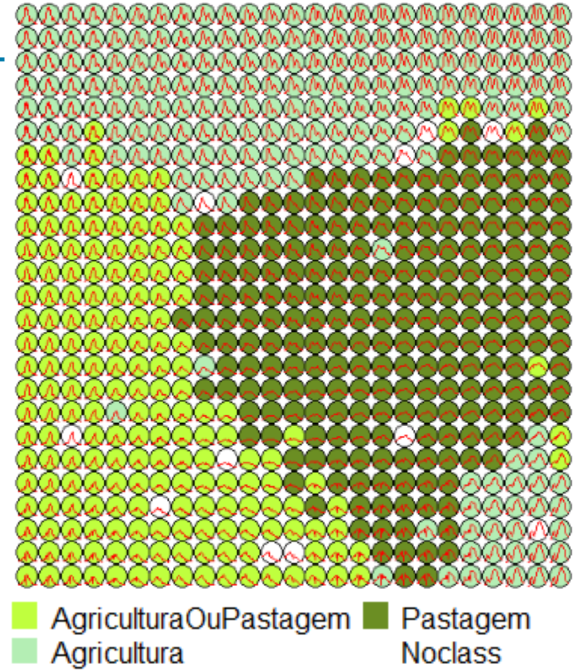


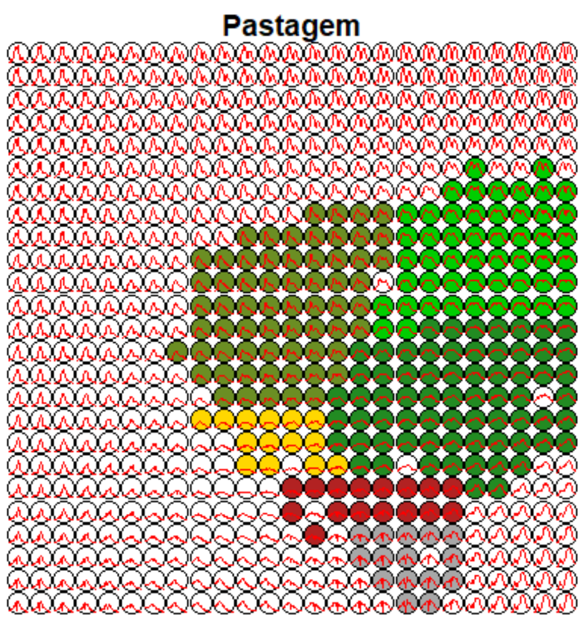


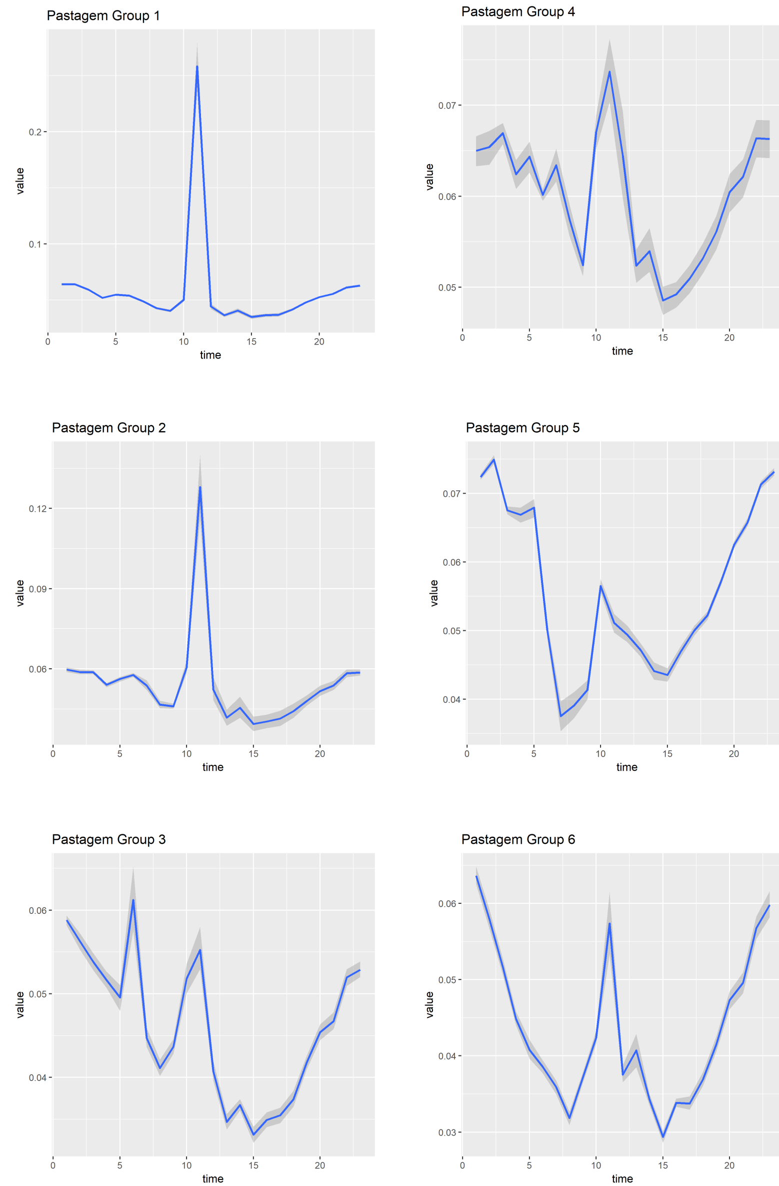
[Lorena Alves, 2018, "A method to assess LUCC samples from satellite image time series "]

A method to assess LUCC samples from satellite image time series

[Lorena Alves, 2018, “A method to assess LUCC samples from satellite image time series ”]







[Lorena Alves, 2018, "A method to assess LUCC samples from satellite image time series "]

Time Series

Definition: A time series T is an ordered sequence of *n* real-valued variables

$$T = (t_1,\ldots,t_n),\ t_i \in R$$

A time series is often the result of the observation of an underlying process which values are collected from measurements made at *uniformly spaced time instants* and according to a given *sampling rate*.

Definition: Given a time series $T = (t_1,\ldots,t_n)$ of length *n*, a subsequence S of T is a series of length $m \leq n$ consisting of contiguous time instants from T

$$S = (t_k, t_{k+1}, \ldots, t_{k+m-1}) \quad \textit{with} \quad 1 \leq k \leq n-m+1$$

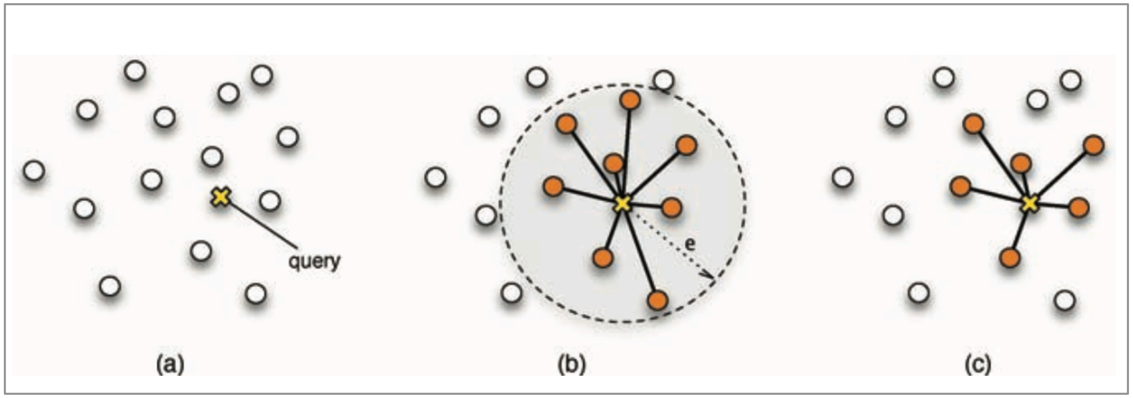
Source: (Esling and Agon, 2012)

Time Series Data Mining – Main Tasks

- Preprocessing
- Query by Content
- Clustering
- Classification
- Segmentation
- Prediction
- Anomaly Detection
- Motif Discovery

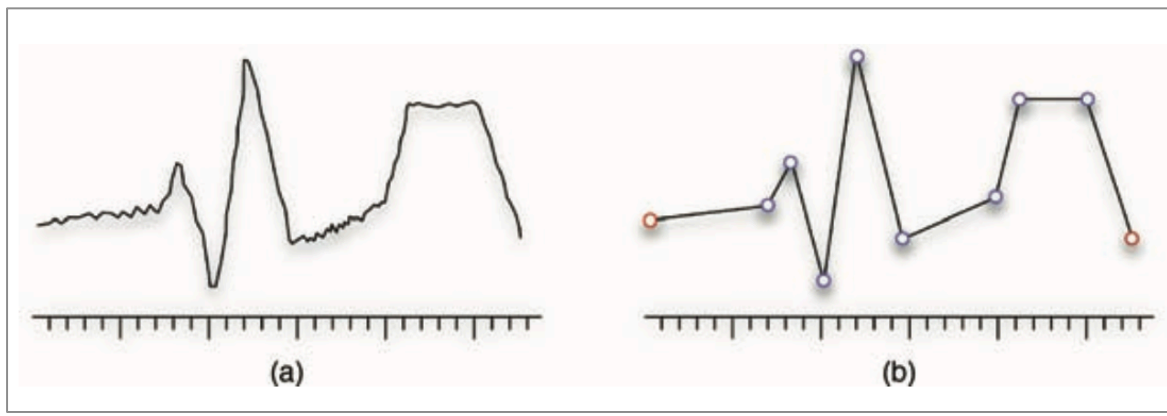
Source: (Esling and Agon, 2012)

Time Series Data Mining – Main Tasks



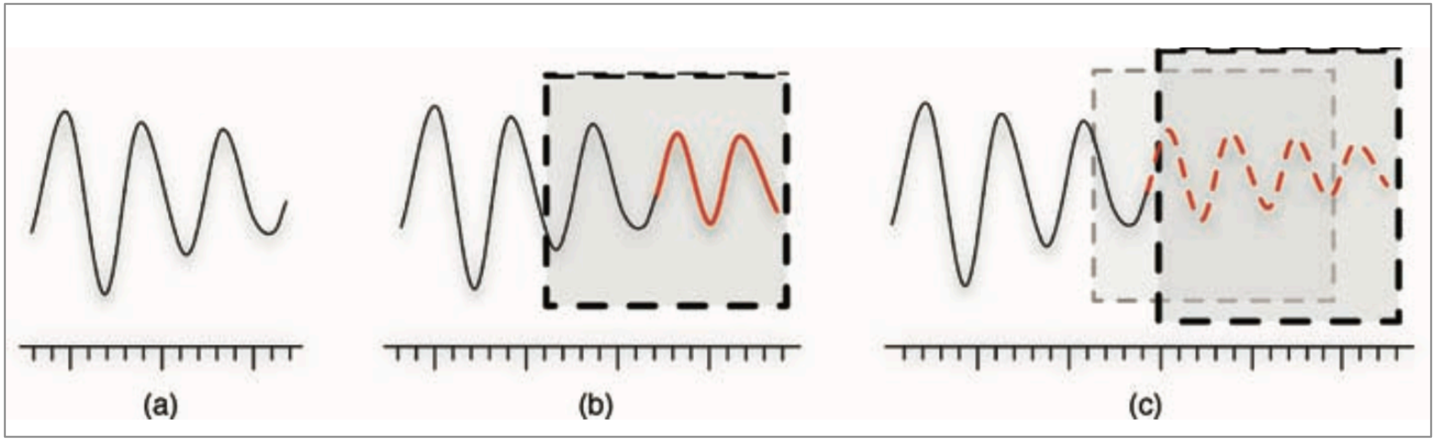
Query by content:
(a) query representation;
(b) ε-range query – distance ε
(c) K-Nearest Neighbors query.

Segmentation: the goal is to find the closest approximation of the input time series with the maximal **dimensionality reduction** factor without losing any of its essential features.



Source: (Esling and Agon, 2012)

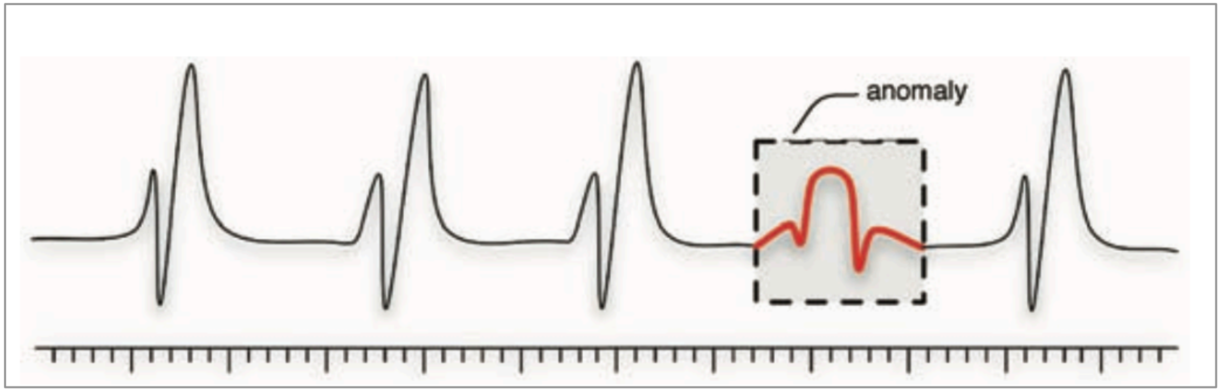
Time Series Data Mining – Main Tasks



Predition:
(a) The input time series may exhibit a periodical and thus predictable structure. (b) The goal is to forecast a maximum number of upcoming datapoints within a prediction window. (c) The task becomes really hard when it comes to having recursive prediction, that is, the long-term prediction of a time series implies reusing the earlier forecast values as inputs in order to go on predicting.

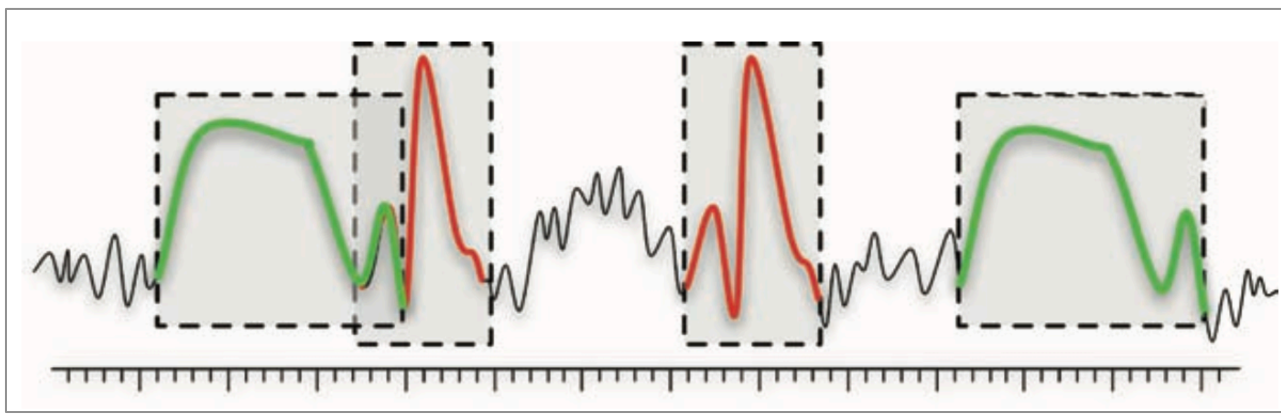
Source: (Esling and Agon, 2012)

Time Series Data Mining – Main Tasks



Anomaly Detection: a long time series which exhibits some kind of periodical structure can be modeled thanks to a reduced pattern of "standard" behavior. The goal is thus to find subsequences that do not follow the model and may therefore be considered as anomalies

Motif Discovery: consists in finding every subsequence that appears recurrently in a longer time series. These subsequences are named ***motifs***. This task exhibits a high combinatorial complexity as several motifs can exist within a single series, motifs can be of various lengths, and even overlap.



Source: (Esling and Agon, 2012)

Time Series Data Mining – Main Tasks

Preprocessing

Query by Content

Clustering

Classification

Segmentation

Prediction

Anomaly Detection

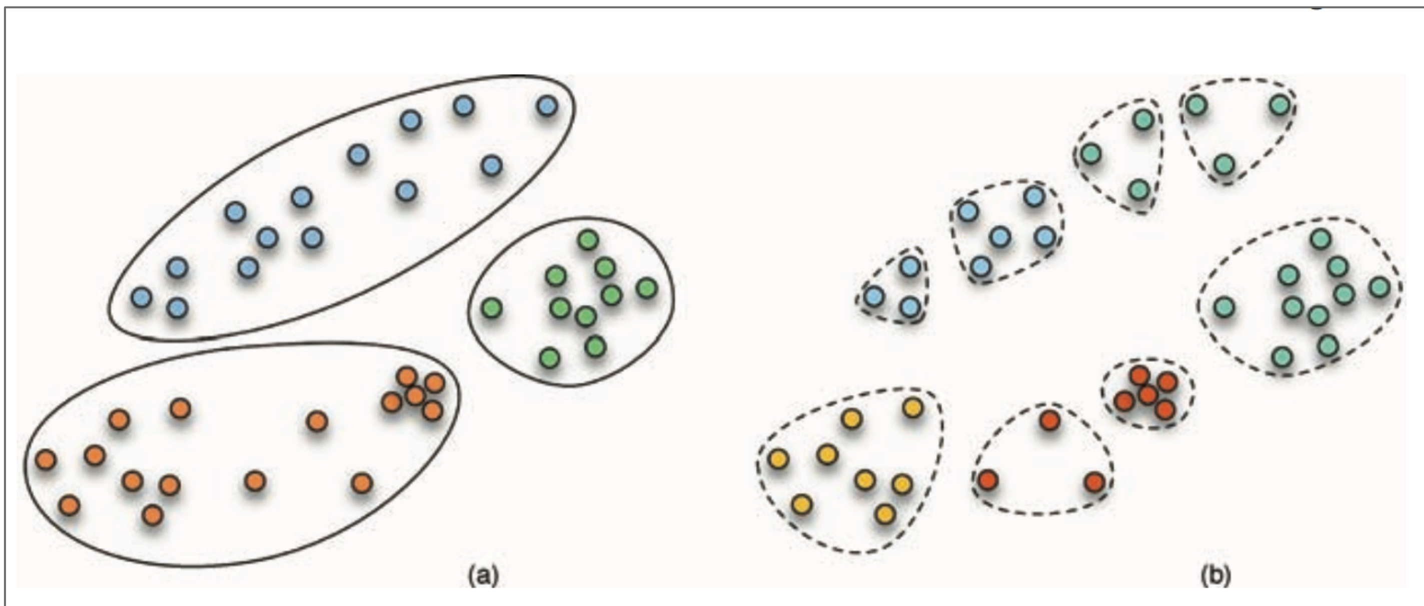
Motif Discovery

Source: (Esling and Agon, 2012)

Clustering

Clustering is the process of finding natural groups, called *clusters*, in a dataset.

The objective is to find the most homogeneous clusters that are as distinct as possible from other clusters. The grouping should maximize *intercluster* variance while minimizing *intracluster* variance.



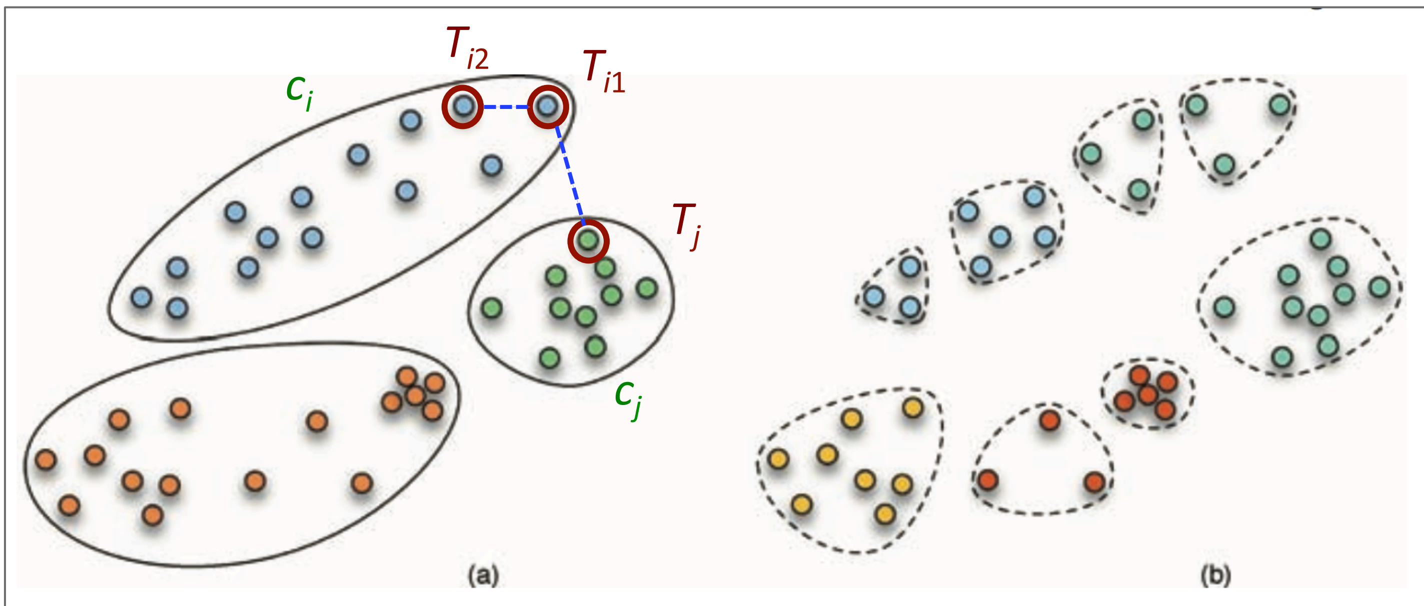
(a) N = 3 and (b) N = 8

Source: (Esling and Agon, 2012)

Time Series Clustering

Definition: Given a time-series database *DB* and a similarity measure D(*Q*, *T*), find the set of clusters C = {c_i} where c_i = {T_k | $T_k \in DB$} that maximizes *intercluster* distance and minimizes *intracluster* variance.

More formally $\forall i_1, i_2, j$ such that $T_{i1}, T_{i2} \in c_i$ and $T_j \in c_j$ $D(T_{i1}, T_j) \gg D(T_{i1}, T_{i2})$.



(a) N = 3 and (b) N = 8

Source: (Esling and Agon, 2012)

Time Series Clustering Taxonomy

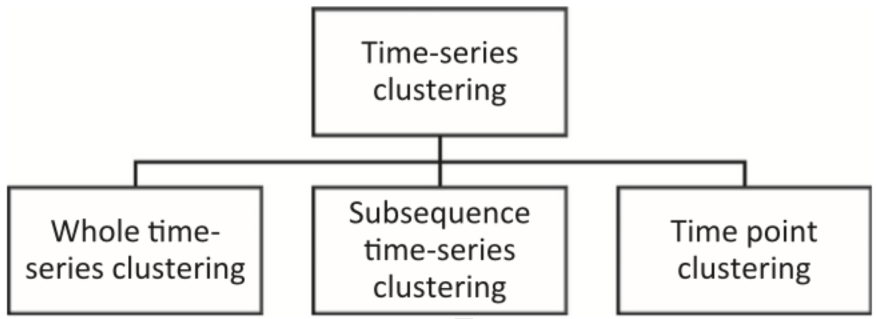


Fig. 1. Time-series clustering taxonomy.

clustering of a set of individual time-series with respect to their similarity.

clustering on a set of subsequences of a time-series that are extracted via a sliding window, that is, clustering of segments from a single long time-series.

clustering of time points based on a combination of their temporal proximity of time points and the similarity of the corresponding values.

This approach is similar to time-series segmentation. However, it is different from segmentation as all points do not need to be assigned to clusters, i.e., some of them are considered as noise.

Keogh and Lin (2003) represented that subsequence time series clustering is meaningless!

Source: (Aghabozorgi et al. 2015)

Time Series Clustering Taxonomy

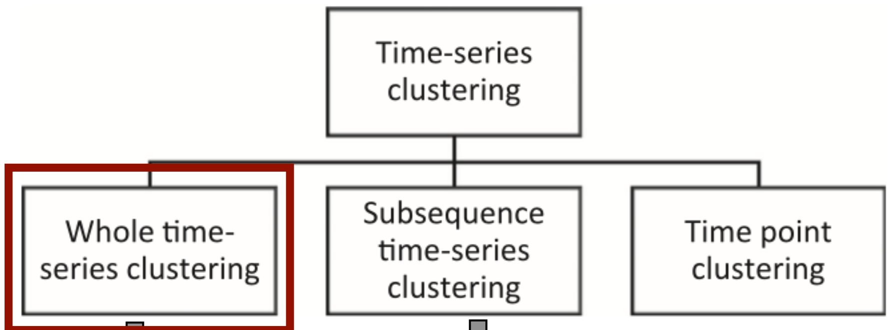


Fig. 1. Time-series clustering taxonomy.

clustering of a set of individual time-series with respect to their similarity.

clustering on a set of subsequences of a time-series that are extracted via a sliding window, that is, clustering of segments from a single long time-series.

clustering of time points based on a combination of their temporal proximity of time points and the similarity of the corresponding values.

This approach is similar to time-series segmentation. However, it is different from segmentation as all points do not need to be assigned to clusters, i.e., some of them are considered as noise.

Keogh and Lin (2003) represented that subsequence time series clustering is meaningless!

Source: (Aghabozorgi et al. 2015)

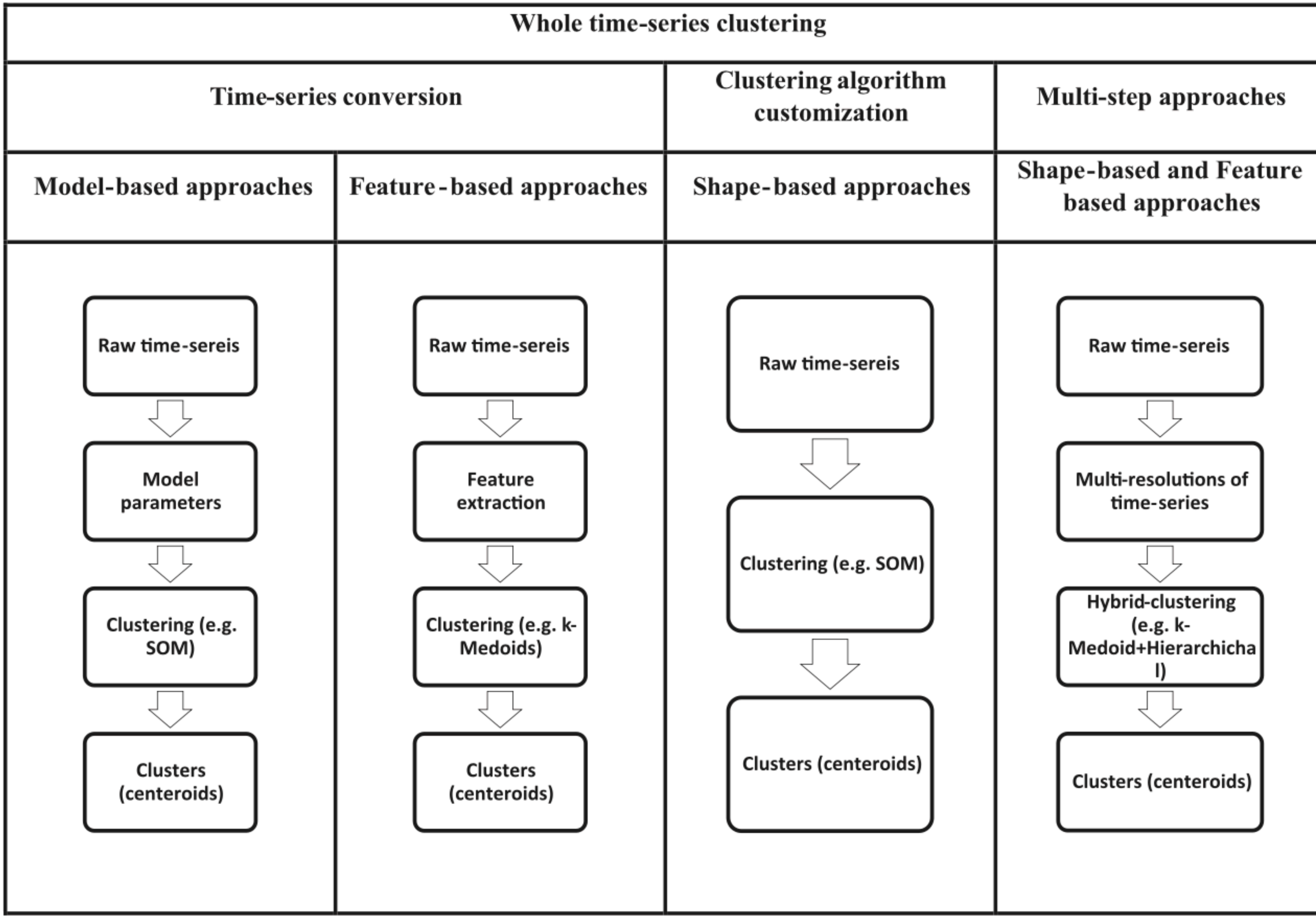


Fig. 2. The time-series clustering approaches.

Source: (Aghabozorgi et al. 2015)

Whole time series clustering

(1) Model based approaches: raw time-series is transformed into model parameters (a parametric model for each time-series,) and then a suitable model distance and a clustering algorithm (usually conventional clustering algorithms) is chosen and applied to the extracted model parameters.

(2) Feature-based approach: raw time-series are converted into a feature vector of lower dimension. Later, a conventional clustering algorithm is applied to the extracted feature vectors. Usually in this approach, an equal length feature vector is calculated from each time-series followed by the Euclidean distance measurement.

(3) Shape-based approach: shapes of two time-series are matched as well as possible, by a non-linear **stretching and contracting** of the time axes. This approach has also been labelled as a raw-data-based approach because it typically works directly with the raw time-series data. Shape-based algorithms usually employ conventional clustering methods, which are compatible with static data while their distance/similarity measure has been modified with an appropriate one for time-series.

Source: (Aghabozorgi et al. 2015)

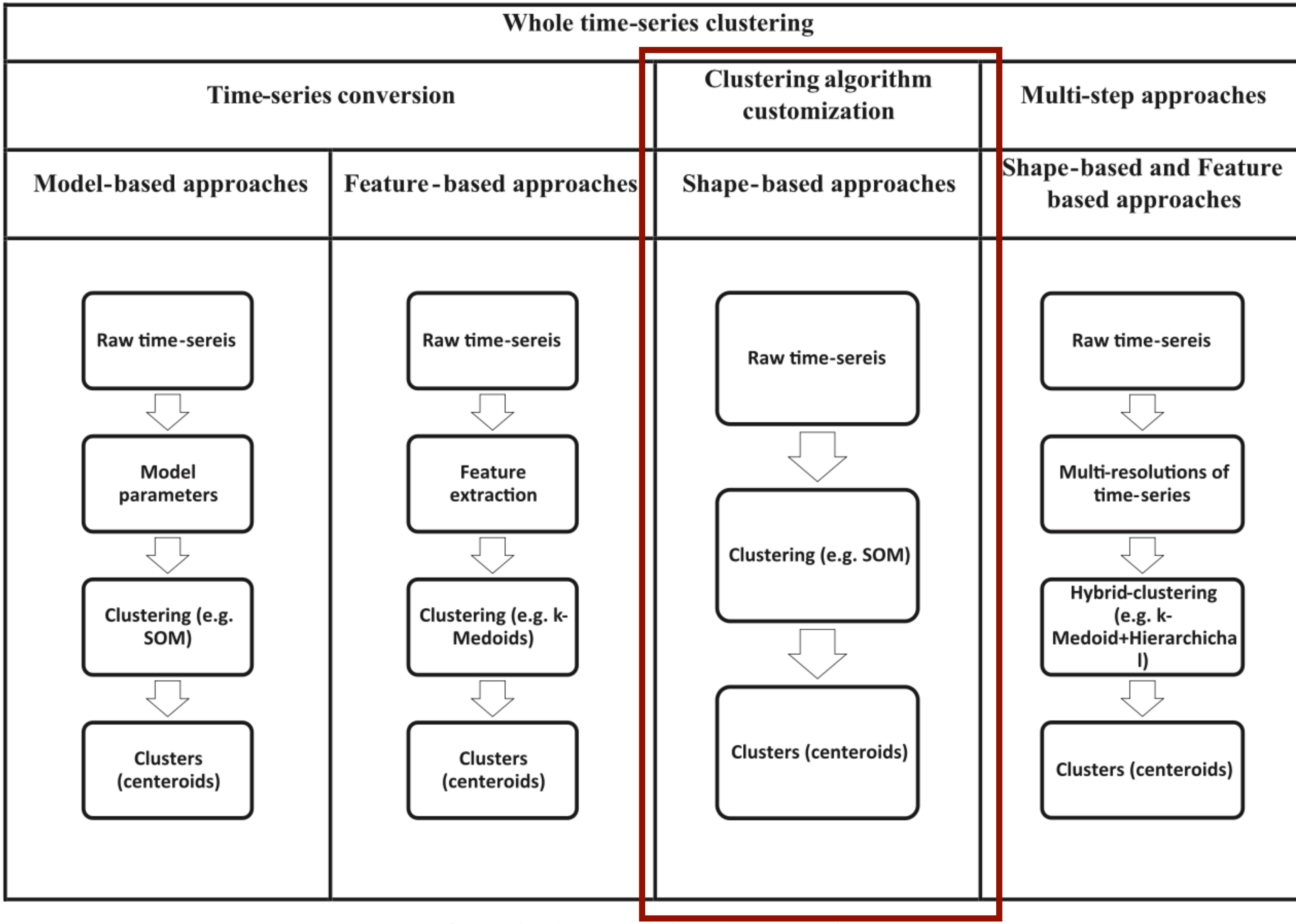


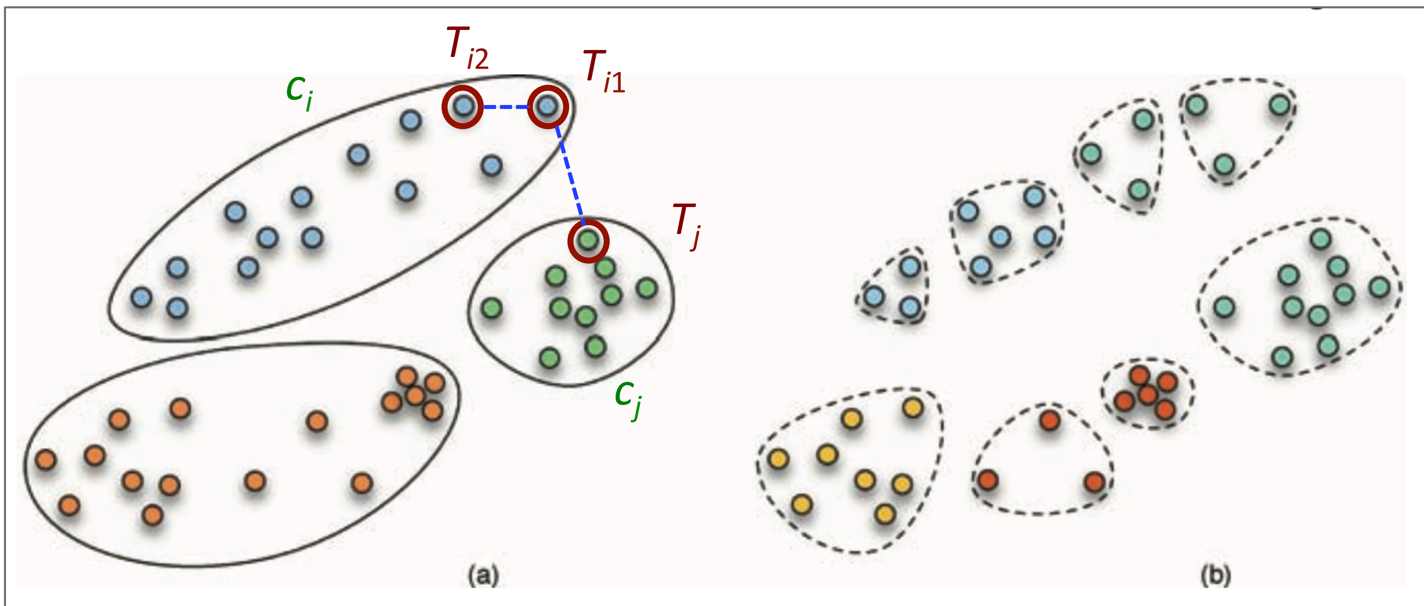
Fig. 2. The time-series clustering approaches.

Source: (Aghabozorgi et al. 2015)

Time Series Clustering

Definition: Given a time-series database *DB* and a similarity measure $D(Q, T)$, find the set of clusters $C = \{c_i\}$ where $c_i = \{T_k \mid T_k \in DB\}$ that maximizes *intercluster* distance and minimizes *intracluster* variance.

More formally $\forall i_1, i_2, j$ such that $T_{i1}, T_{i2} \in c_i$ and $T_j \in c_j$ $D(T_{i1}, T_j) \gg D(T_{i1}, T_{i2})$.



(a) N = 3 and (b) N = 8

Source: (Esling and Agon, 2012)

Time Series – Similarity Measures

There is different distance measures designed for specifying similarity between time series.

The most popular distance measurement methods that are used for time series data: (1) The Hausdorff distance, (2) modified Hausdorff (MODH), (3) HMM-based distance, (4) Dynamic Time Warping (DTW), (5) Euclidean distance, (6) Euclidean distance in a PCA subspace, and (7) Longest Common Sub-Sequence (LCSS).

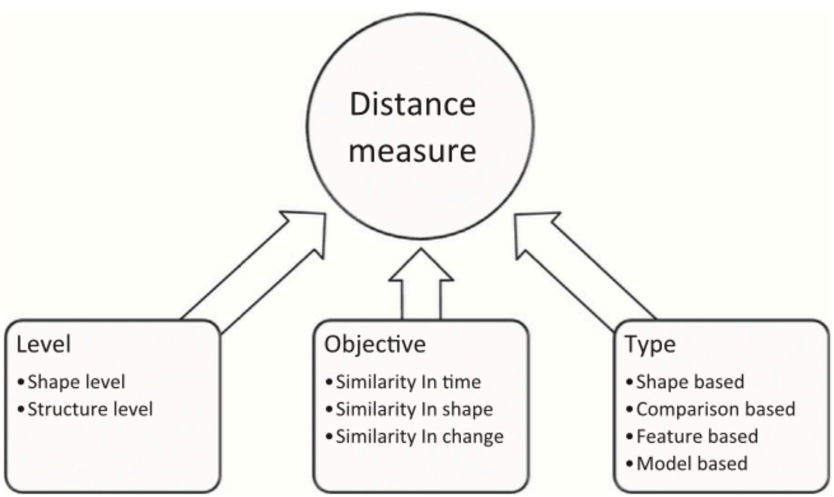


Fig. 5. Distance measure approaches in the literature.

The choice of a proper distance approach depends on the **characteristic** of time series, **length** of time series, representation method, and the **objective** of clustering time series to a high extent (similarity in time, in shape or in change).

Source: (Aghabozorgi et al. 2015)

Shape-based similarity measures

The time of occurrence of patterns is not important to find similar time series in shape. Shape-based similarity measure is to find the similar time series in time and shape. (Aghabozorgi et al. 2015)

A good comparasion of time series distance measures can be found in (Ding et al., 2008)

Shape-based similarity measures

Distance Measure	Characteristics
Euclidean Distace (ED)	Lock-step Measure (one-to-one) using in indexing, clustering and classification, Sensitive to scaling.
Dynamic Time Warping (DTW)	Elastic Measure (one-to-many/one-to-none) Very well in deal with temporal drift. Better accuracy than Euclidean distance. Lowe efficiency than Euclidean distance and triangle similarity.
Longest Common Sub-Sequence (LCSS)	Noise robustness
Minimal Variance Matching (MVM)	Automatically skips outliers
Edit Distance on Real sequence (EDR)	Elastic measure (one-to-many/one-to-none), uses a threshold pattern
Cross-correlation based distances	Noise reduction, able to summarize the temporal structure
Edit Distance with Real Penalty (ERP)	Robust to noise, shifts and scaling of data, a constant reference point is used
Histogram-based	Using multi-scale time-series histograms
DISSIM	Proper for different sampling rates
Sequence Weighted Alignment model (Swale)	Similarity score based on both match rewards and mismatch penalties.
Triangle similarity measure	Can deal with noise, amplitude scaling very well and deal with offset translation, linear drift well in some situations.

Source: (Aghabozorgi et al. 2015)

Time Series Clustering

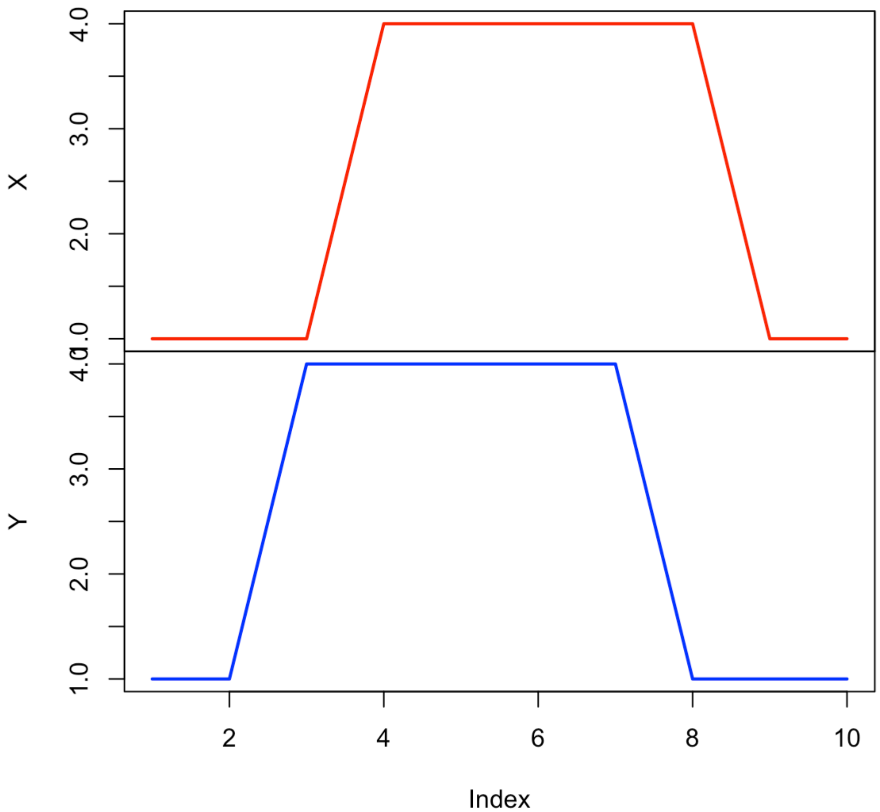
Clustering is a common solution performed to discovery **patterns** on time-series datasets.

Time-series clustering is the most-used approach as an **exploratory technique**, and also as a subroutine in more complex data mining algorithms, such as rule discovery, indexing, classification, and anomaly detection.

Euclidean distance and **DTW** are the most common methods for similarity measure in time-series clustering.

Source: (Aghabozorgi et al. 2015)

Euclidean Distance

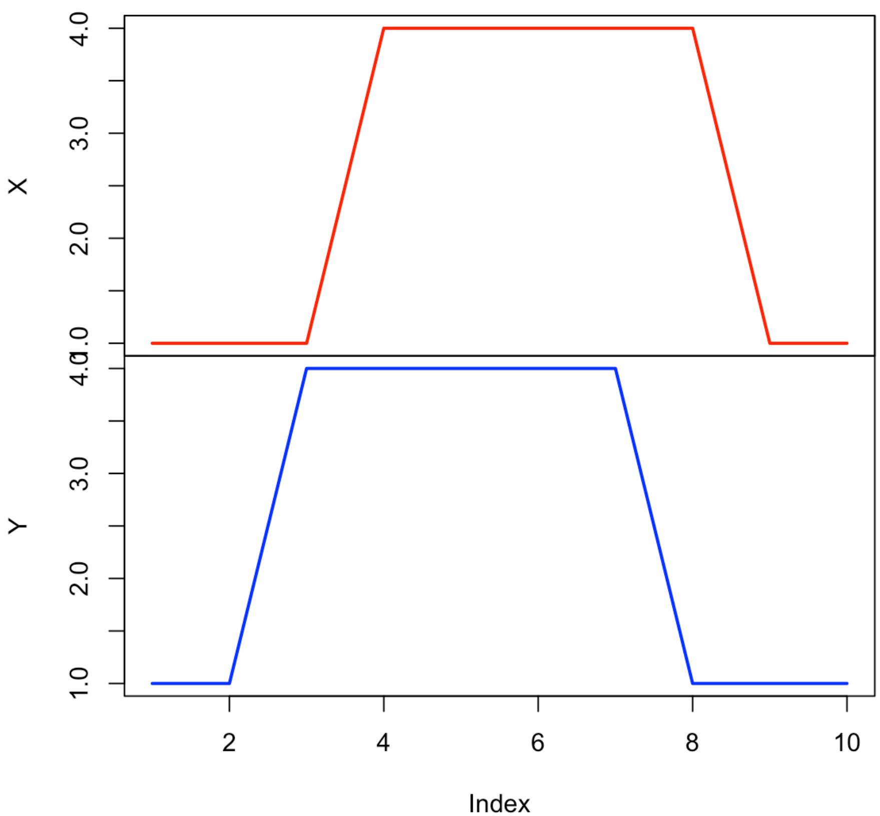



```r
# Two time series
X <- c(1,1,1,4,4,4,4,4,1,1)
Y <- c(1,1,4,4,4,4,4,1,1,1)

# Euclidean distance
TSdist::EuclideanDistance(X, Y)
```

```
[1] 4.242641
```

Dynamic Time Warping (DTW)




```
# Two time series
X <- c(1,1,1,4,4,4,4,4,1,1)
Y <- c(1,1,4,4,4,4,4,1,1,1)

# Euclidean distance
TSdist::EuclideanDistance(X, Y)
```

[1] 4.242641

```
# Two time series
X <- c(1,1,1,4,4,4,4,4,1,1)
Y <- c(1,1,4,4,4,4,4,1,1,1)

# Euclidean distance
TSdist::DTWDistance(X, Y)
```

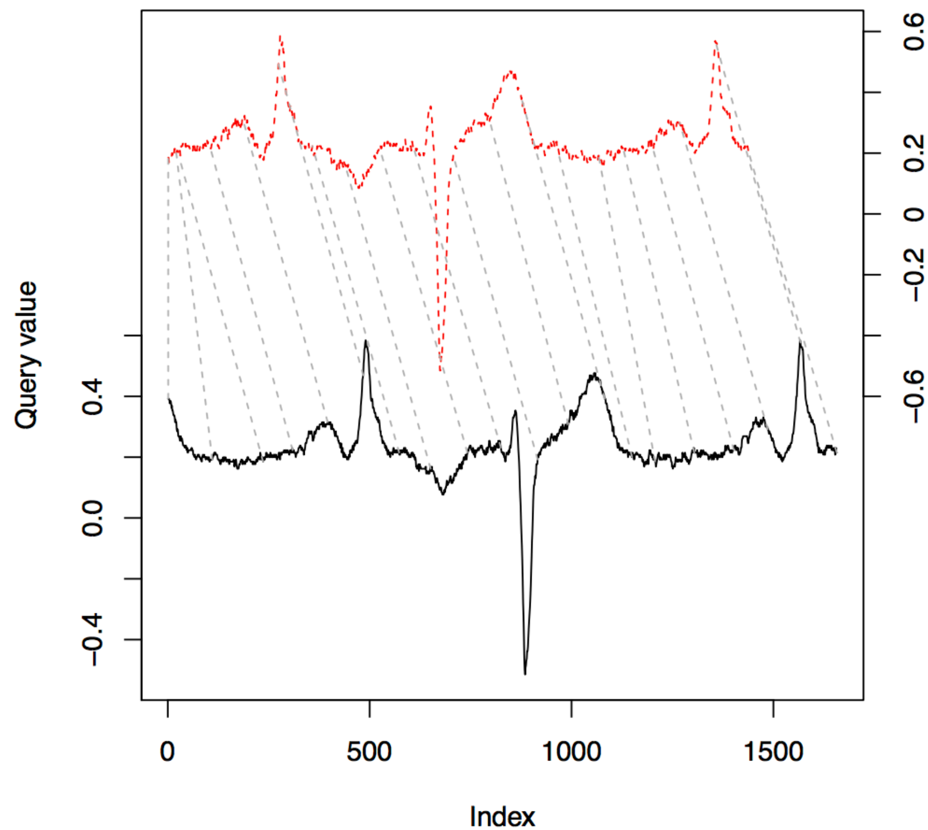
[1] 0

The choice of a proper distance approach depends on the **objective** of clustering time series!

Dynamic Time Warping (DTW)

Proposed around 1970. Given two time series, DTW stretches or compresses them locally in order to make one resemble the other as much as possible.

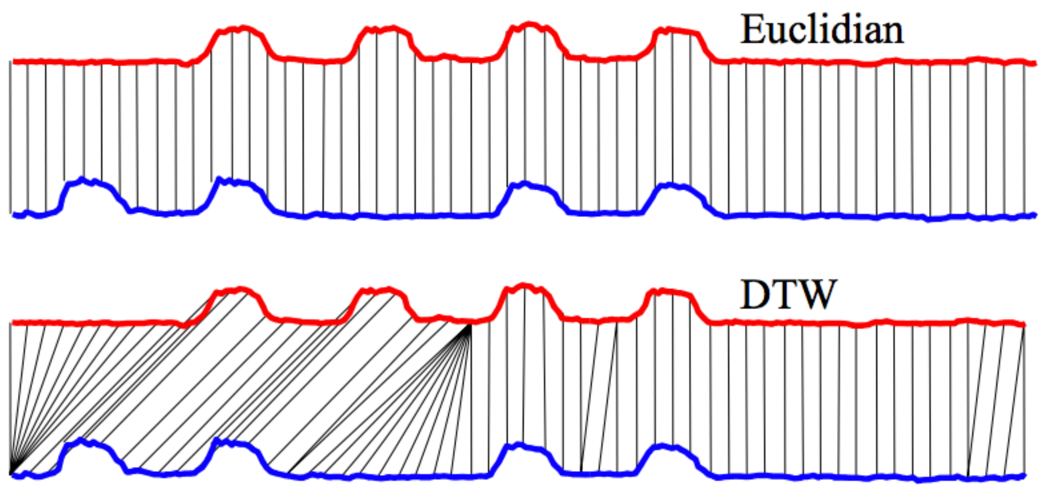
The distance between the two is computed, after stretching, by summing the distances of individual aligned elements.



Source: (Giogino, 2009)

Dynamic Time Warping (DTW)

DTW is a much more robust distance measure for time series, allowing similar shapes to match even if they are out of phase in the time axis.



Source: (Keogh and Ratanamahatana, 2005)

DTW

Two time series Q and C, of length n and m

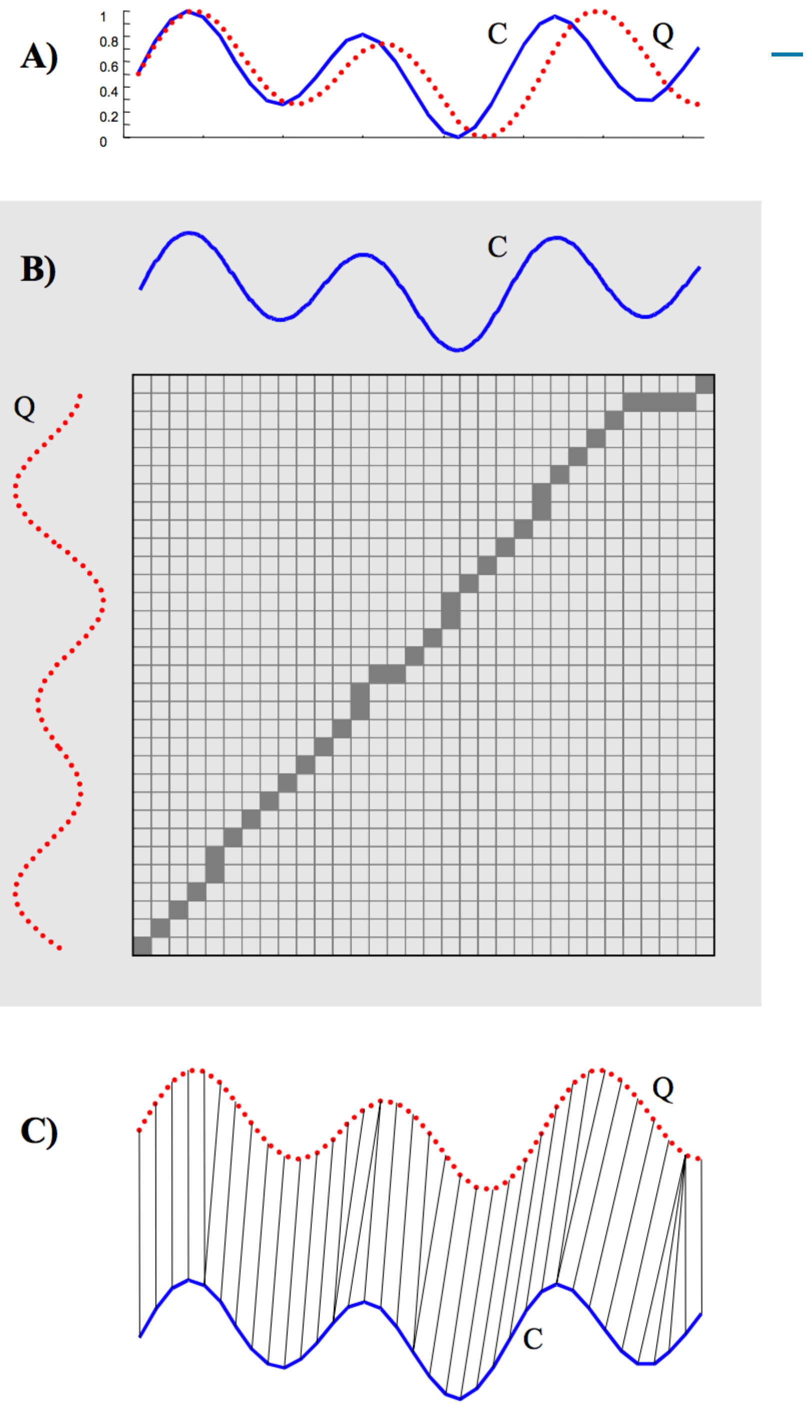
$Q = q_1, q_2, \ldots, q_i, \ldots, q_n$
$C = c_1, c_2, \ldots, c_j, \ldots, c_m$

n-by-m matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $d(q_i, c_j)$ between the two points q_i and c_j

$d(q_i, c_j) = (q_i - c_j)^2$

Each matrix element (i,j) corresponds to the alignment between the points q_i and c_j.

Source: (Keogh and Ratanamahatana, 2005)



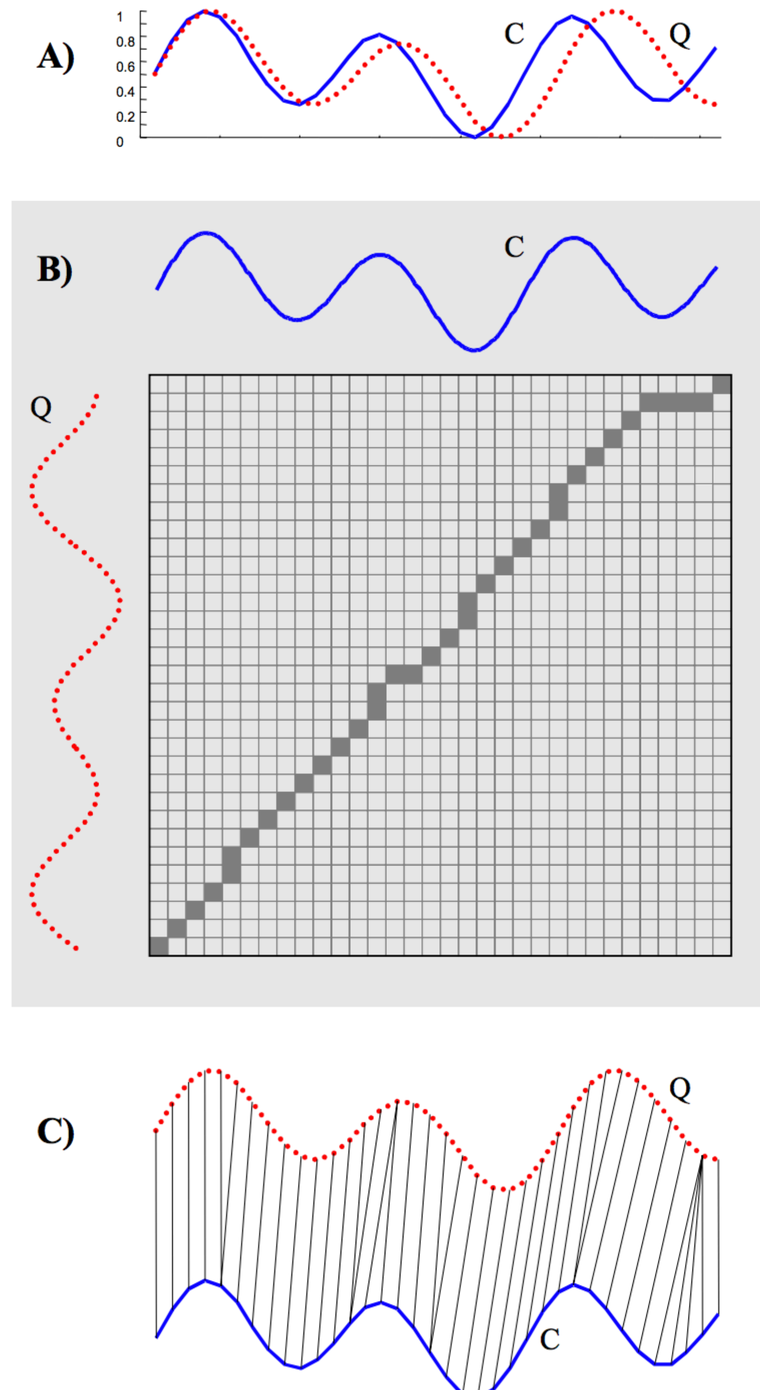
DTW

A *warping path W*, is a contiguous set of matrix elements that defines a mapping between *Q* and *C*.

The k^{th} element of *W* is defined as $w_k = (i,j)_k$ so we have:

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K$$

$$\max(m,n) \leq K < m+n-1$$



Source: (Keogh and Ratanamahatana, 2005)

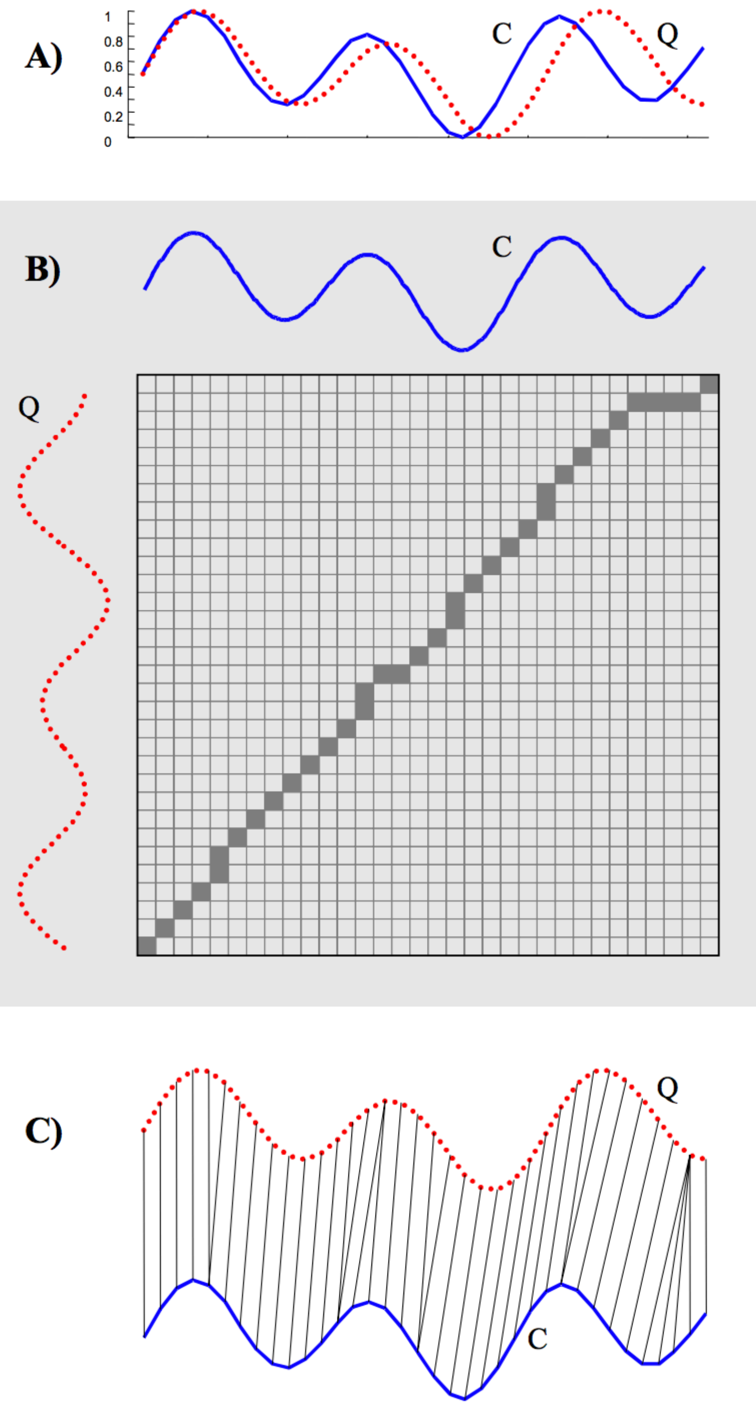
DTW

The warping path is typically subject to several constraints:

Boundary conditions: this requires the warping path to start and finish in diagonally opposite corner cells of the matrix.

Continuity: This restricts the allowable steps in the warping path to adjacent cells (including diagonally adjacent cells).

Monotonicity: This forces the points in *W* to be monotonically spaced in time



Source: (Keogh and Ratanamahatana, 2005)

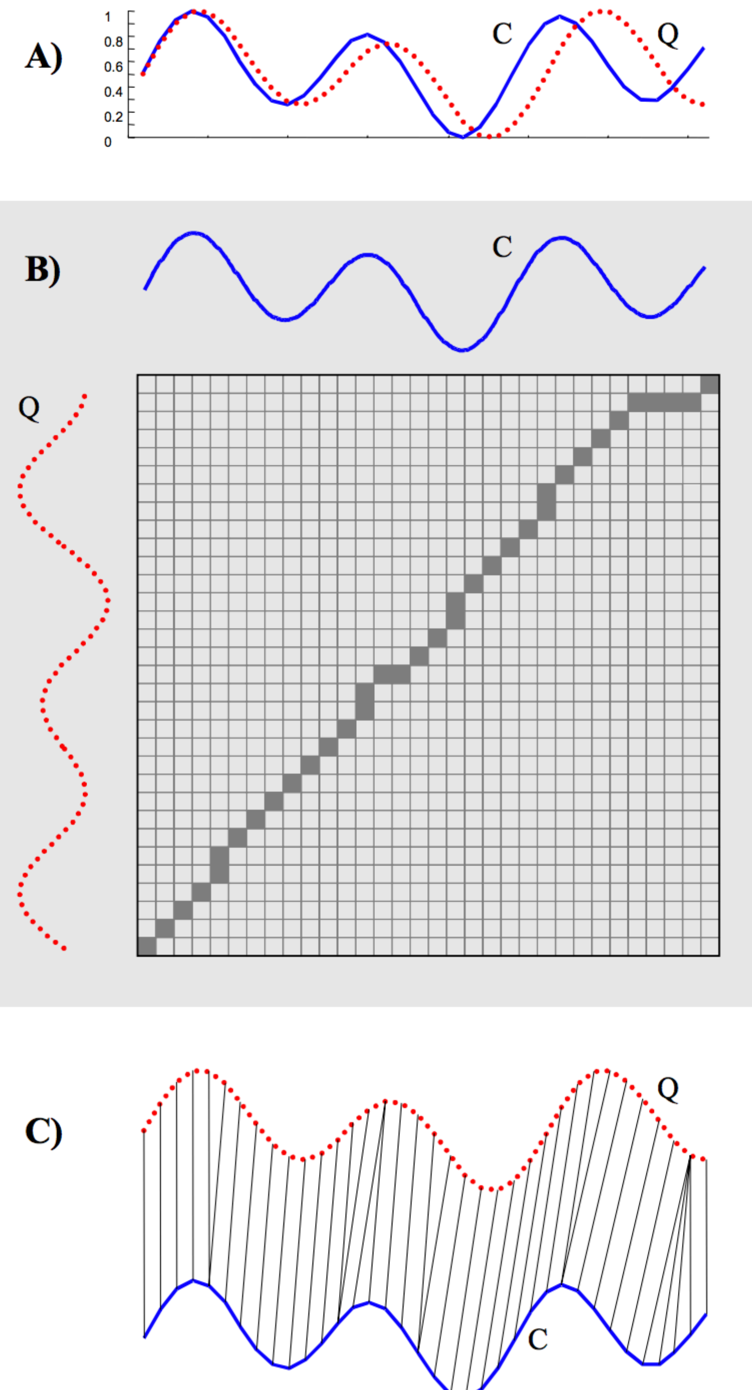
DTW

There are exponentially many warping paths that satisfy the constraints, however we are only interested in the path that *minimizes the warping cost*:

$$DTW(Q,C) = \min\left\{ \sqrt{\sum_{k=1}^{K} w_k} \right.$$

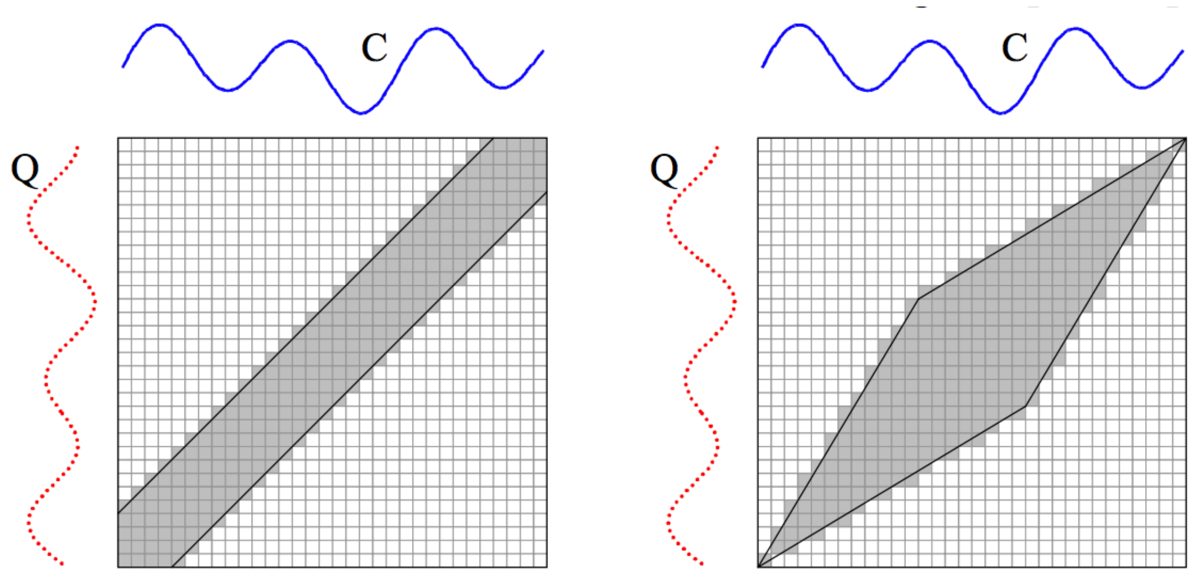
Complexit: O(*nm*)

Source: (Keogh and Ratanamahatana, 2005)



DTW – Global constraints on time warping

DTW also constraint the warping path in a global sense by limiting how far it may stray from the diagonal. The subset of matrix that the warping path is allowed to visit is called the *warping window*.



Sakoe-Chiba Band   Itakura Parallelogram

Two advantages:

(1) Speed up the DTW distance calculation

(2) Prevent pathological warpings, where a relatively small section of one sequence maps onto a relatively large section of another.

Source: (Keogh and Ratanamahatana, 2005)

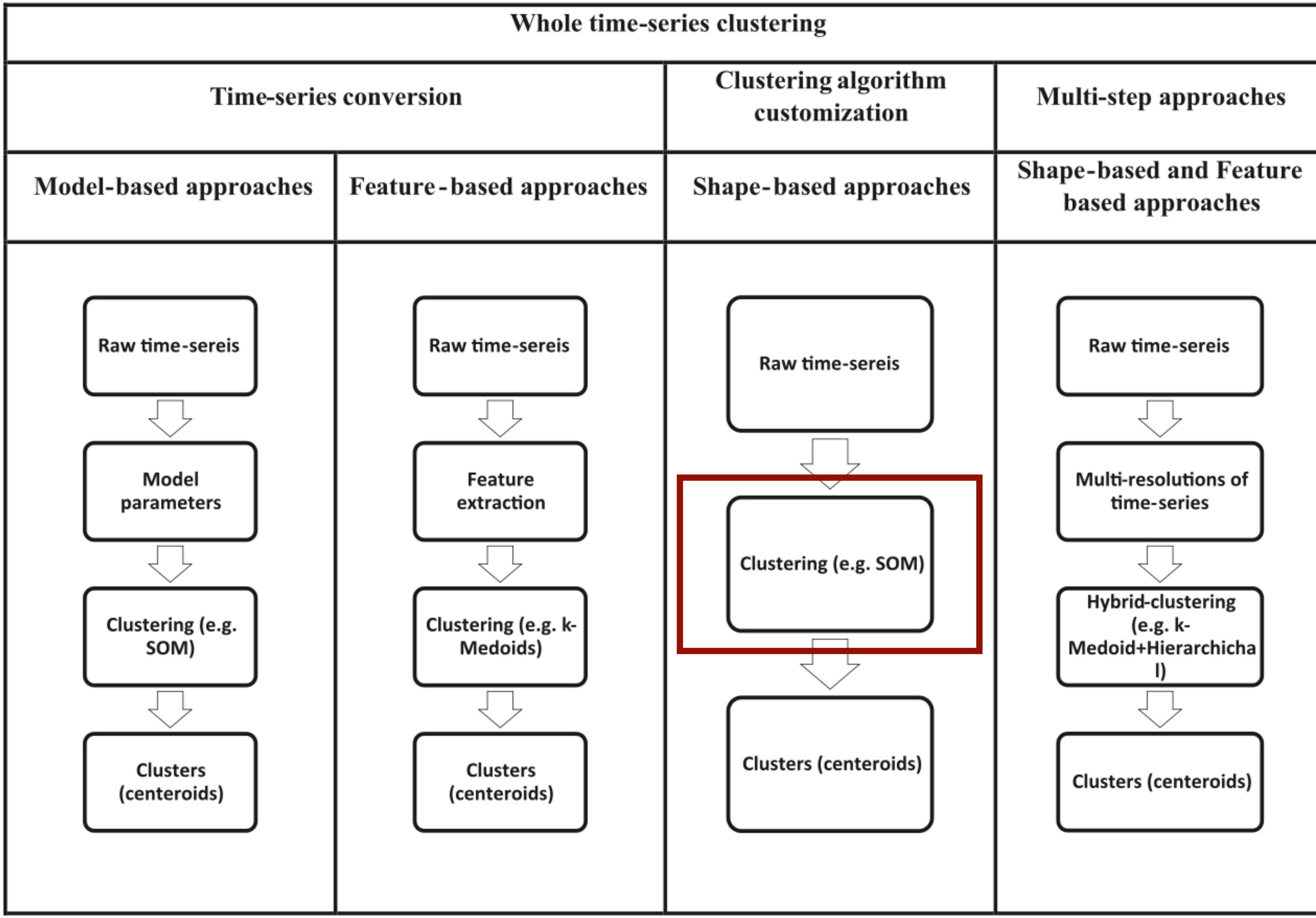


Fig. 2. The time-series clustering approaches.

Source: (Aghabozorgi et al. 2015)

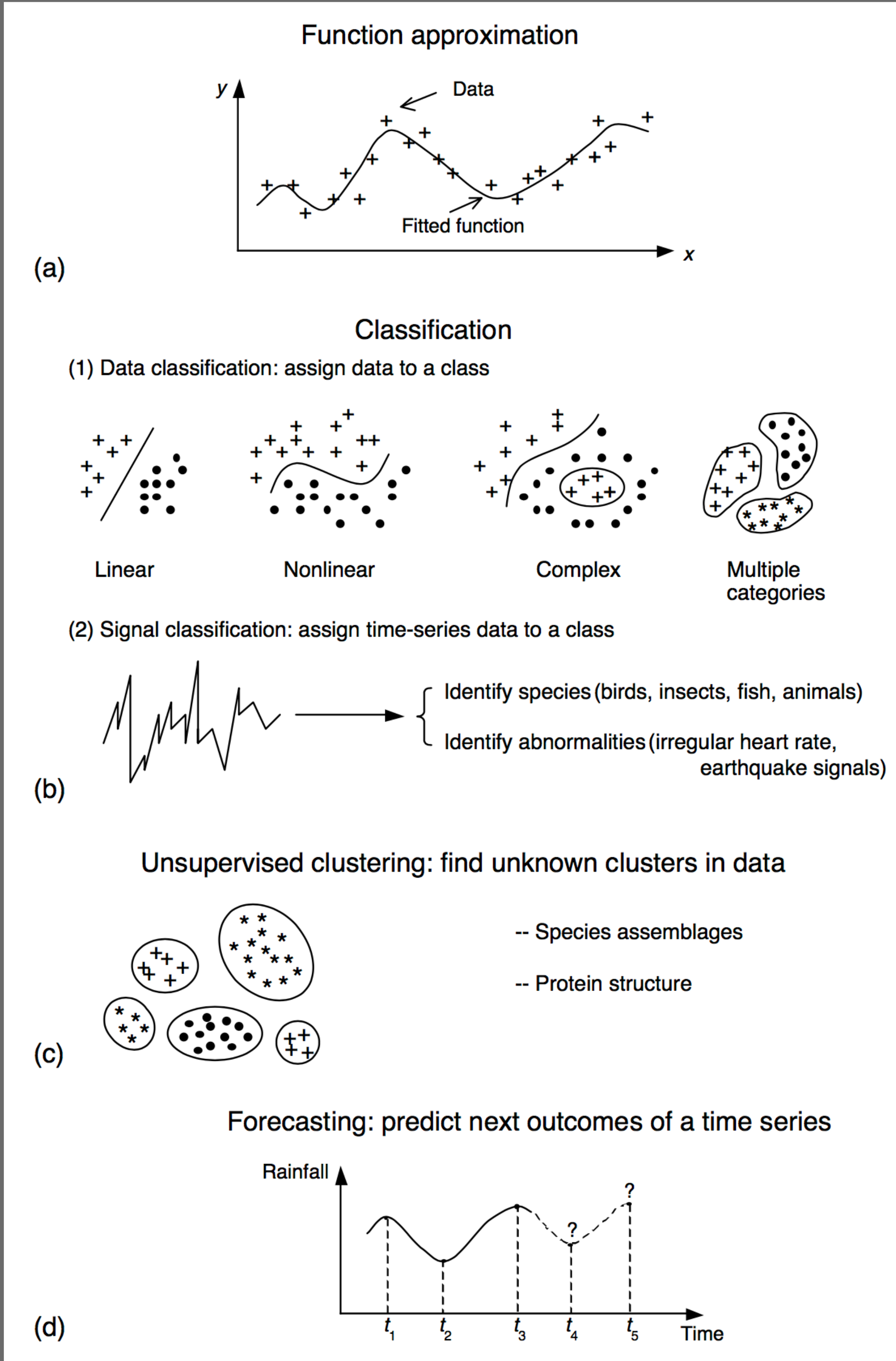
Neural Network

Neural network is a **collection of interconnected neurons** that incrementally learn from their environment (data) to capture essential linear and nonlinear trends in complex data.

Neurons are the basic computing units that perform local data processing inside a network.

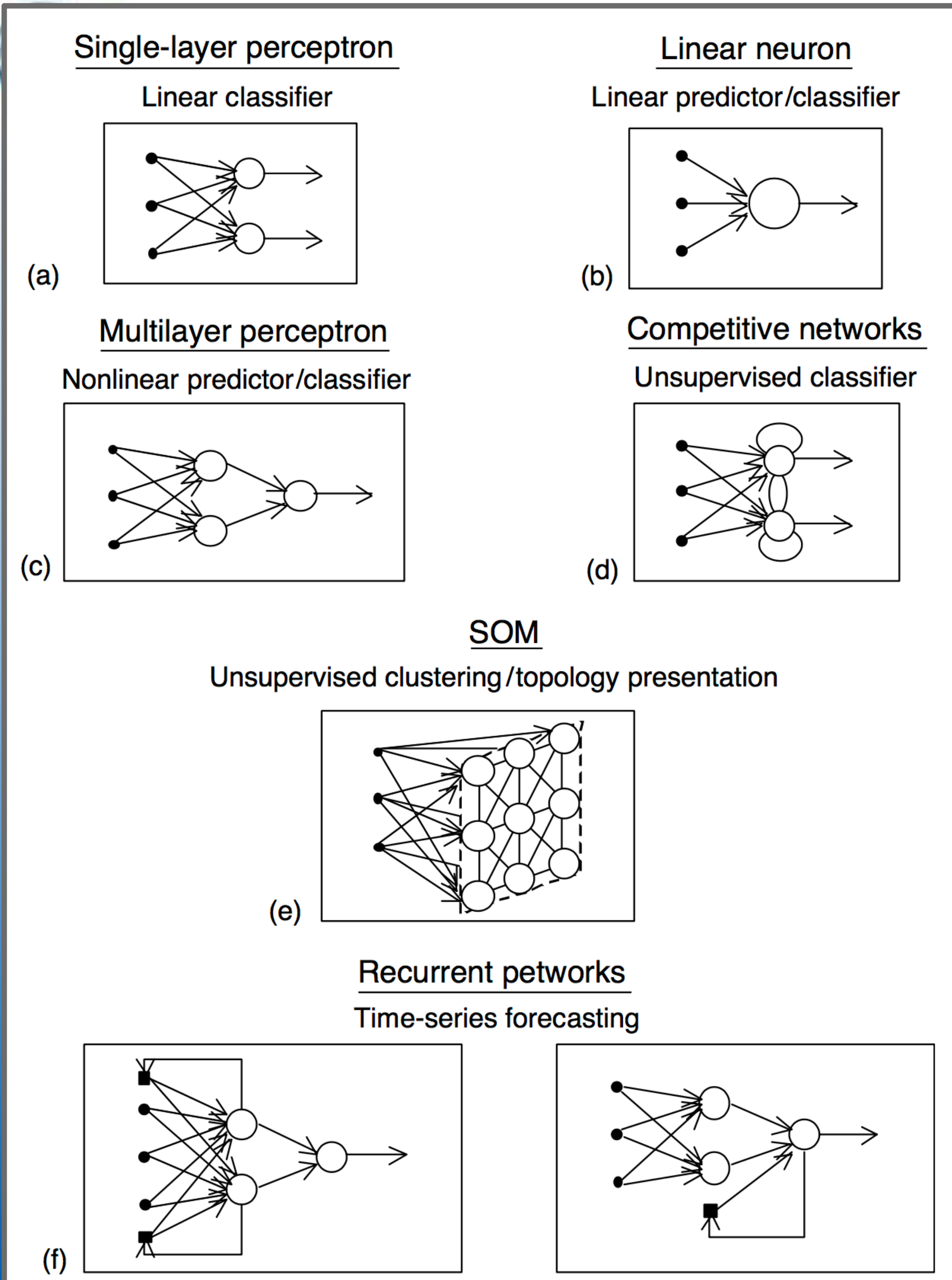
It resembles the brain in two respects: (1) Knowledge is acquired by the network through a **learning process**; (2) Interconnection strengths between neurons, known as *synaptic weights* or *weights*, are used to **store knowledge**. (Haykin, 1994)

Source: (Samarasinghe, 2016)



Neural networks perform a variety of tasks, including prediction or function approximation (a), pattern classification (b), clustering (c), and forecasting (d).

Source: (Samarasinghe, 2016)



← Some neural networks types.

Key features of neural networks:

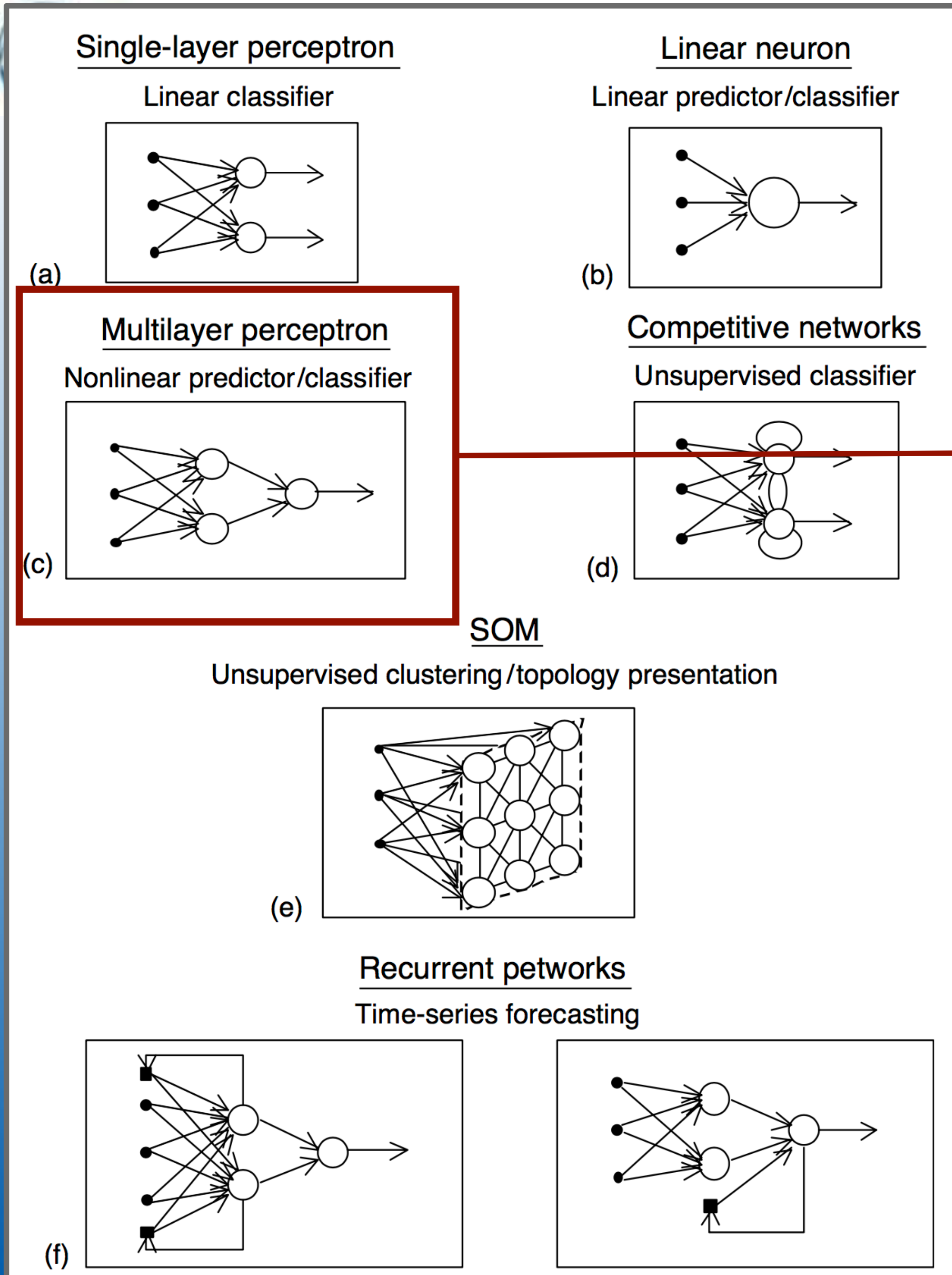
(1) they process information locally in neurons;

(2) neurons operate in parallel and are connected into a network through weights depicting the connection strength;

(3) networks acquire knowledge from the data in a process called *learning*, which is stored or reflected in the weights;

(4) a network that has undergone learning captures the essential features of a problem and can therefore make reliable predictions.

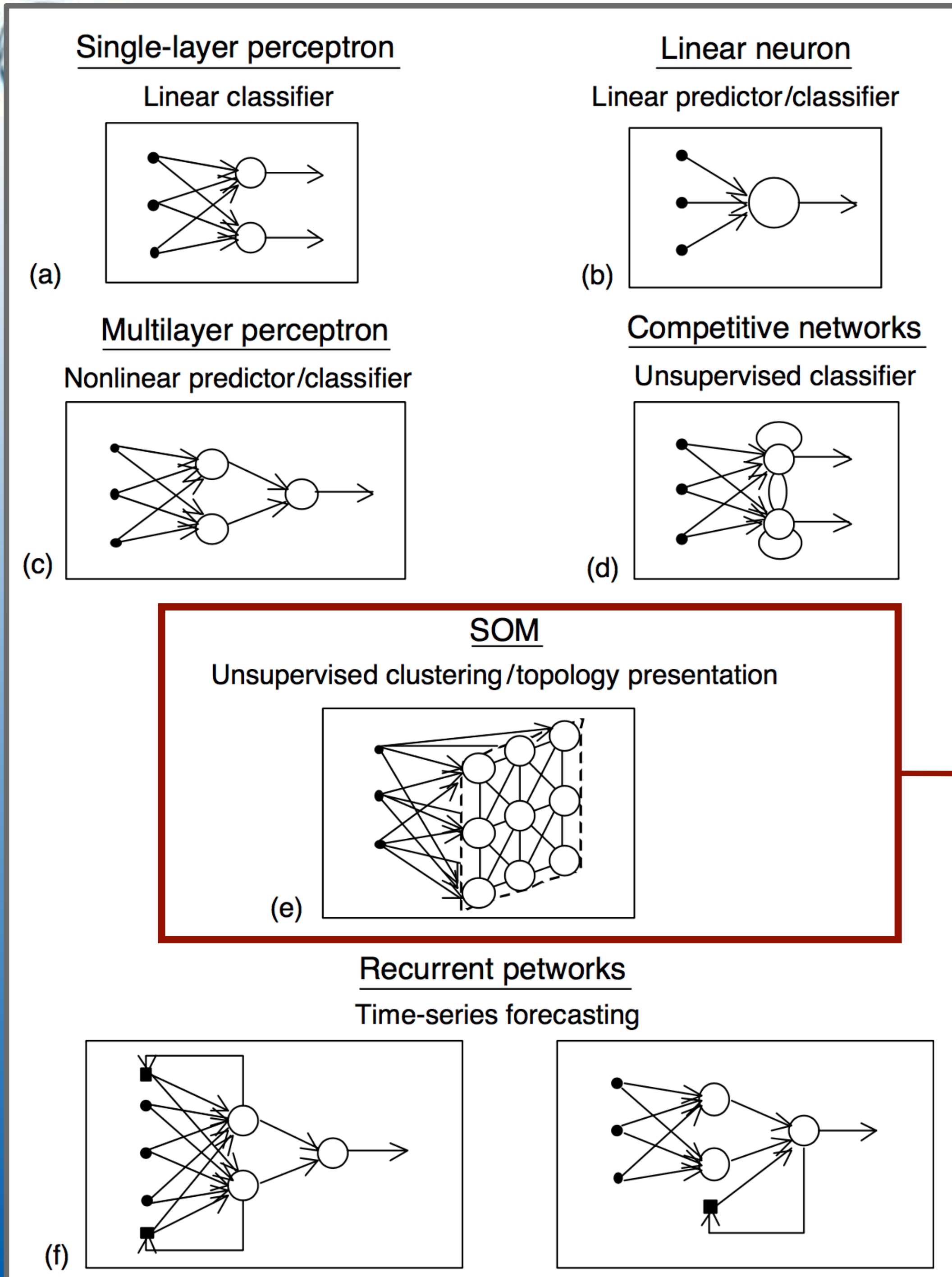
Source: (Samarasinghe, 2016)



There are several methods suitable for **nonlinear analysis**, including multilayer perceptron (MLP) networks, radial basis function (RBF) networks, support vector machines (SVMs), generalized model for data handling (GMDH), also called polynomial nets, generalized regression neural network (GRNN) and generalized neural network (GNN).

Most of these networks have several processing layers that give them nonlinear modeling capability.

Source: (Samarasinghe, 2016)



The self-organizing map (SOM) not only finds unknown clusters in the data but also preserves the topological structure (spatial relations) of the data and clusters.

Source: (Samarasinghe, 2016)

Unsupervised Neural Networks

Unsupervised neural networks are used to **find structures** in complex data.

They are useful because there are many real-life phenomena in which the data is multidimensional and its **structure and relationships are unknown a priori**; in these situations, the **data must be analyzed to reveal the patterns inherent in** it.

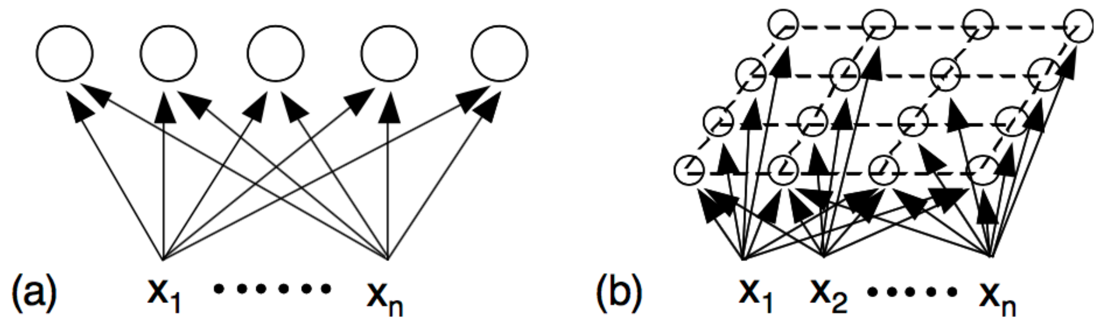
The Self-Organizing Map (SOM) is a unsupervised neural network.

SOM was proposed by Kohonen in 1990.

Source: (Samarasinghe, 2016)

SOM - Structure

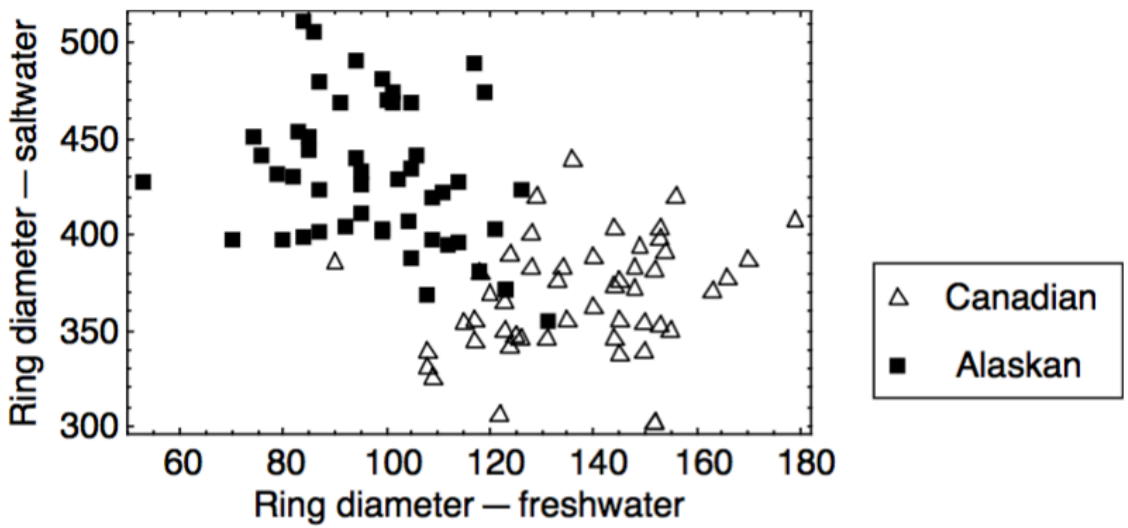
An unsupervised network usually has **two layers of neurons**: an *input layer* and an *output layer*. The input layer represents the input variables, x_1, x_2, ..., x_n, for the case of *n* inputs. The output layer may consist of neurons arranged in a single line (a) (one-dimensional) or a two-dimensional grid (b), forming a two-dimensional layer.



Source: (Samarasinghe, 2016)

SOM – Example 1

Example 1: identify the breed of salmon, whether Alaskan or Canadian, from the growth-ring diameter of the scales in freshwater and ring diameter of scales in seawater.



Input layer:

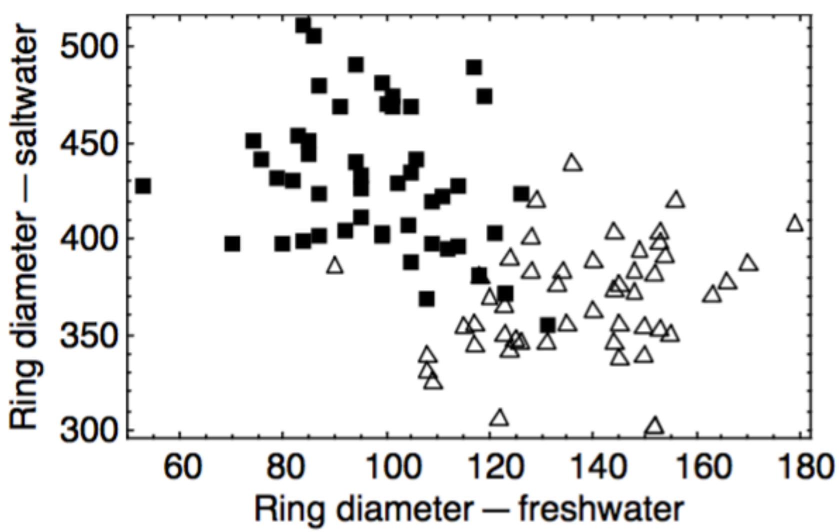
n = 50 inputs of two variables

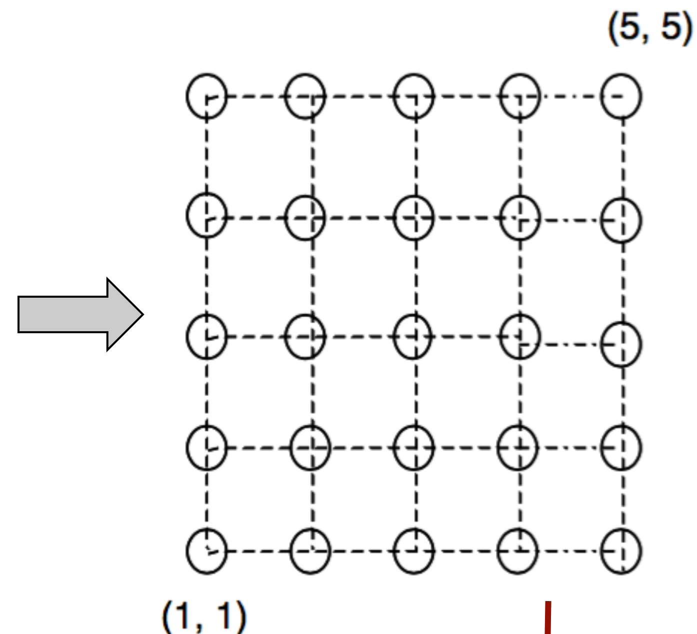
x_i = (diam_fresh$_i$, diam_salt$_i$)

Source: (Samarasinghe, 2016)

SOM – Example 1

Example 1: identify the breed of salmon, whether Alaskan or Canadian, from the growth-ring diameter of the scales in freshwater and ring diameter of scales in seawater.





Input layer:

n = 50 inputs of two variables

x_i = (diam_fresh$_i$, diam_salt$_i$)

Output layer:

25 neurons

Source: (Samarasinghe, 2016)

SOM - Structure

The **number of output neurons** must be determined.

In many cases, the number of data clusters is *unknown*; it is therefore necessary to use a reasonable estimate based on the current understanding of the problem.

When there is uncertainty, it is better to have a larger number of output neurons than the possible number of clusters because redundant neurons can be eliminated.

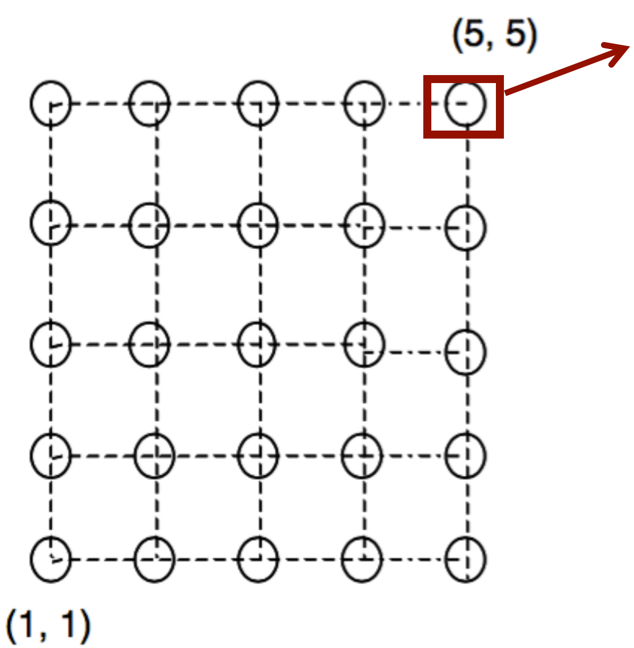
Source: (Samarasinghe, 2016)

SOM - Weights and learning process

Each output neuron has a **weight vector** (**codebook vector**) that has the same dimension as the input vectors.

The **learning process** consistis in adjusting incrementally these **weights**.

Each cluster is represented by one or more final **weight vectors** of neurons.



weight vector (**codebook vector**)

w_f = (diam_fresh$_f$, diam_salt$_f$)

Source: (Samarasinghe, 2016)

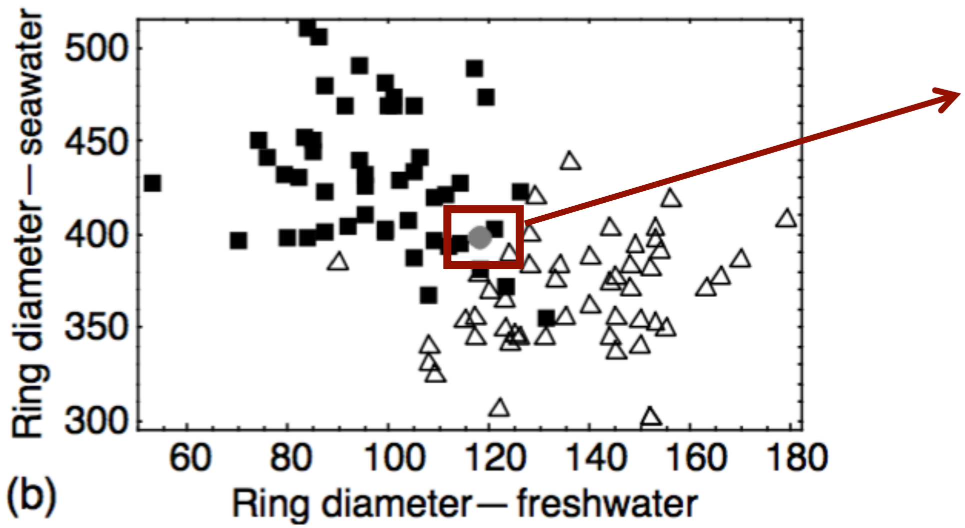
SOM - Competitive learning

Before starting the learning process, the vector weights must be initialized. There are two options: (1) Use random values; or (2) Randomly choose some input vectors and use their values for the weights.

Source: (Samarasinghe, 2016)

SOM - Competitive learning

Before starting the learning process, the vector weights must be initialized. There are two options: (1) Use random values; or (2) Randomly choose some input vectors and use their values for the weights.



All 25 initial weight values are centered in the original data.

Source: (Samarasinghe, 2016)

SOM - Competitive learning

In competitive learning, an input is presented to the network and the winner is selected based on the neuron activation. A neuron is declared the **winner** if it has the highest activation.

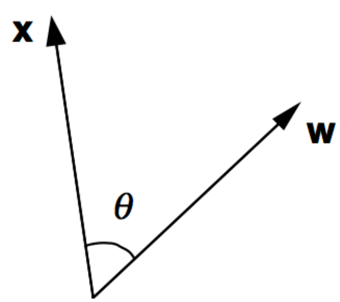
The competition can be implemented by using the concept of **distance** between an input and a weight vector. That is, **a weight that is closer to an input vector would cause a larger activation than one that is far away from the vector.**

Source: (Samarasinghe, 2016)

SOM - Competitive learning

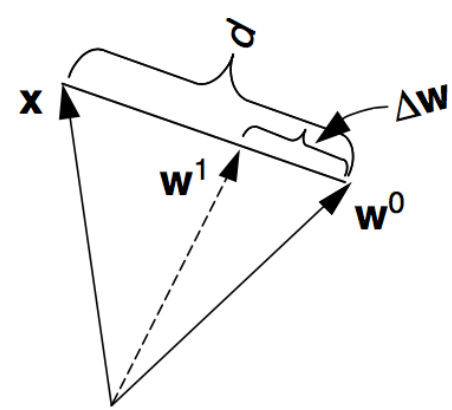
Euclidean distance (d_j) between input vector x and the weight vector w_j associated with the *j*th output neuron.

$$d_j = \mathbf{x} - \mathbf{w}_j = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{w}_{ij})^2}$$



Once the distance between an input vector and all the weights has been found, the neuron with the smallest distance to the input vector is chosen as the **winner**, and its weights are updated so that it moves closer to the input vector.

$$\Delta \mathbf{w}_j = \beta(\mathbf{x} - \mathbf{w}_j) = \beta d_j$$

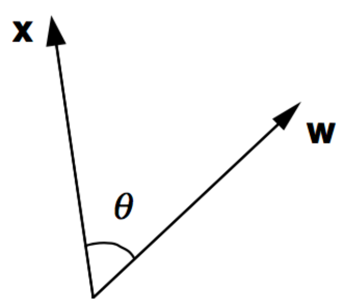


Source: (Samarasinghe, 2016)

SOM - Competitive learning

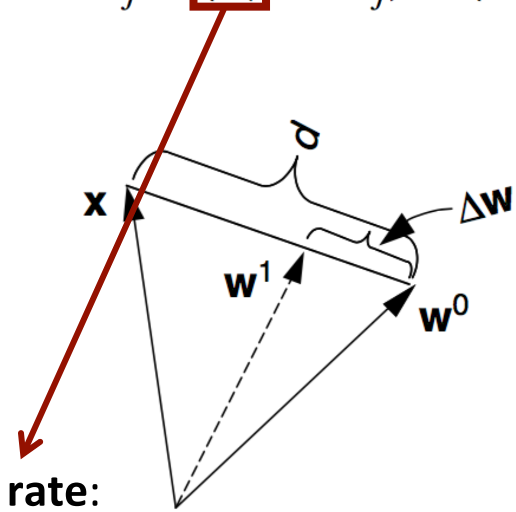
Euclidean distance (d_j) between input vector x and the weight vector w_j associated with the *j*th output neuron.

$$d_j = \mathbf{x} - \mathbf{w}_j = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{w}_{ij})^2}$$



Once the distance between an input vector and all the weights has been found, the neuron with the smallest distance to the input vector is chosen as the **winner**, and its weights are updated so that it moves closer to the input vector.

$$\Delta \mathbf{w}_j = \beta(\mathbf{x} - \mathbf{w}_j) = \beta d_j$$



learning rate:
from 0 to 1

Source: (Samarasinghe, 2016)

SOM – Topology preservation

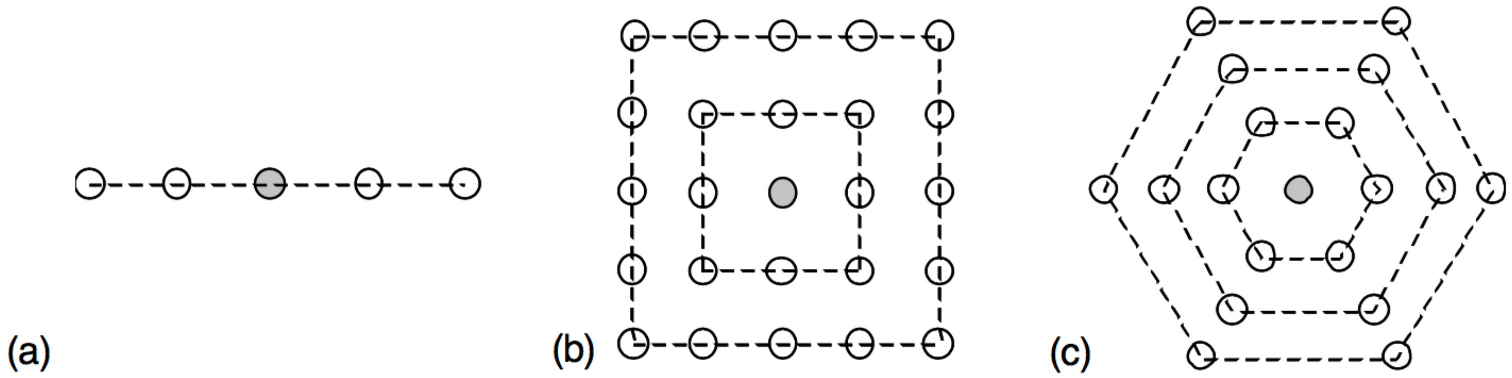
Main feature: **topology preservation** => regions closer in input space are represented by neurons that are closer in the map.

In SOM learning, not only the winner but also the neighboring neurons adjust their weights. Neurons closer to the winner adjust weights more than those that are far from it.

neighboring neurons are similar!

Source: (Samarasinghe, 2016)

SOM - Neighborhood



Neighborhood definitions: (a) linear (b) square, and (c) hexagonal neighborhood surrounding a winning neuron (solid circle denotes winner and empty circles denote neighbors).

Distance radius *r*: 1, 2, ...

Source: (Samarasinghe, 2016)

SOM – Competitive learning

All weight (codebook) vectors w_j of the **winner and neighbors** are adjusted to w'_j according to:

$$\mathbf{w}_j' = \mathbf{w}_j + \beta \, \mathrm{NS}[\mathbf{x} - \mathbf{w}_j]$$

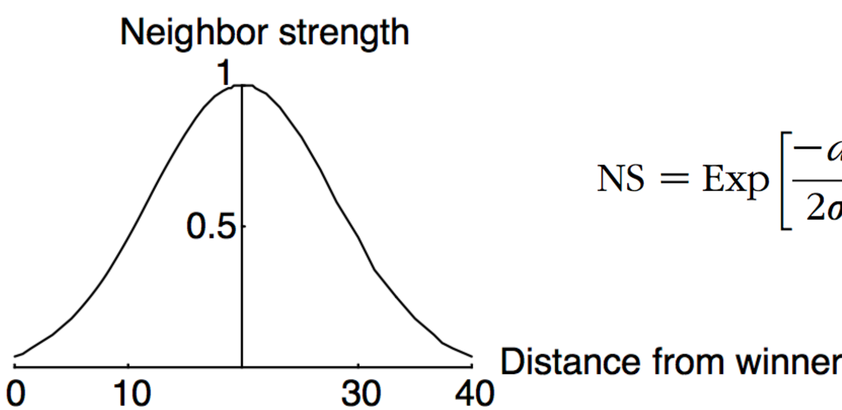
Source: (Samarasinghe, 2016)

SOM – Competitive learning

All weight (codebook) vectors w_j of the **winner and neighbors** are adjusted to w'_j according to:

$$\mathbf{w}'_j = \mathbf{w}_j + \beta\,\mathrm{NS}\,[\mathbf{x} - \mathbf{w}_j]$$

Neighbor Strength: The NS function determines how the weight adjustment decays with distance from the winner. There are several possibilities for this function: linear, Gaussian, and exponential.



$$\mathrm{NS} = \mathrm{Exp}\left[\frac{-d_{i,j}^2}{2\sigma^2}\right]$$

where $d_{i,j}$ is the distance between the winning neuron *i* and any other neuron *j*, and σ is the width of the Gaussian.

Source: (Samarasinghe, 2016)

SOM – Training phase

Training is usually performed in two phases: **Ordering** and **Convergence**.

(1) Ordering (topological ordering): learning rate and neighborhood size are reduced with iterations until the winner or a few neighbors around the winner remain.

(2) Covergence (fine tuning): the feature map is fine tuned with the shrunk neighborhood so that it produces an accurate representation of the input space.

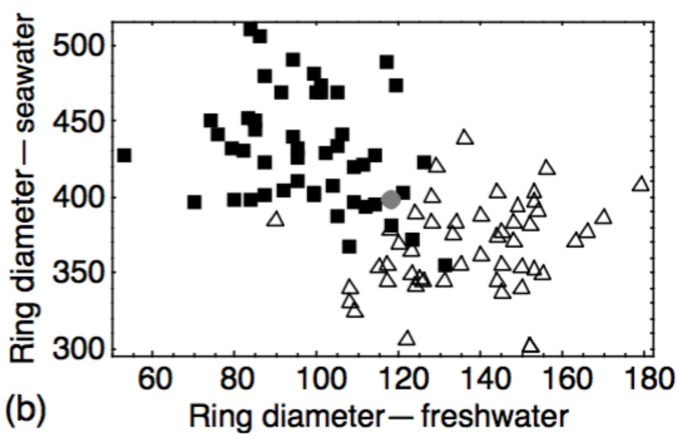
Training terminates when the mean distance between the winning neurons and the inputs they represent stops changing.

Source: (Samarasinghe, 2016)

SOM – Training phase

Ordering phase	Covergence phase
(1) The **neighbor size** should initially cover almost all neurons in the network and then shrink with iterations.	(1) The **neighbor strength** should contain only the nearest neighbors of the winning neuron and may slowly reduce to one or zero neighbors.
(2) The **learning rate** should begin with a relatively high value and should thereafter gradually decrease, but must remain above 0.01.	(2) The **learning rate** is maintained at a small value, on the order of 0.01.
(3) Trainning in **recursive mode**: the weights of the winning neurons and their neighbors are updated after each presentation of an input vector.	(3) Trainning in **batch mode**: the weight adjustment is made after the entire batch of inputs has been processed. After an epoch (i.e., one pass of the whole training dataset through the network), the weights are changed.

Source: (Samarasinghe, 2016)



SOM – Example 1 – Phase 1

Learning rate function:

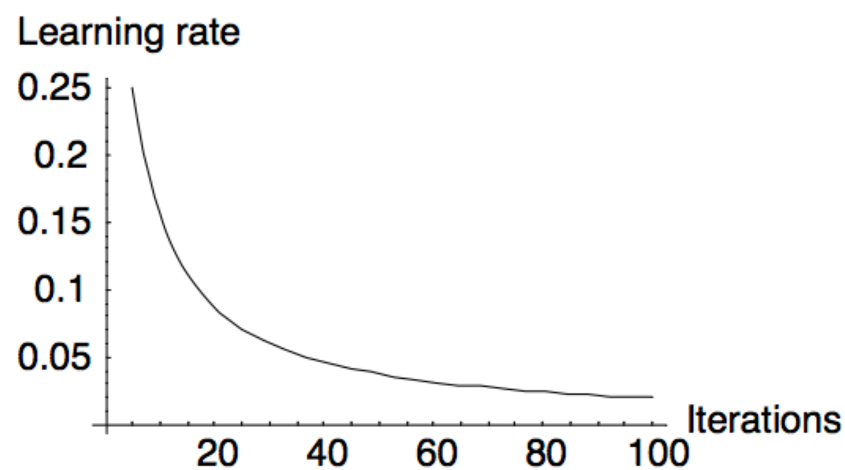
$$\beta = 0.01 \quad \{t < 5$$

$$= \frac{2}{3+t} \quad \{t > 5$$

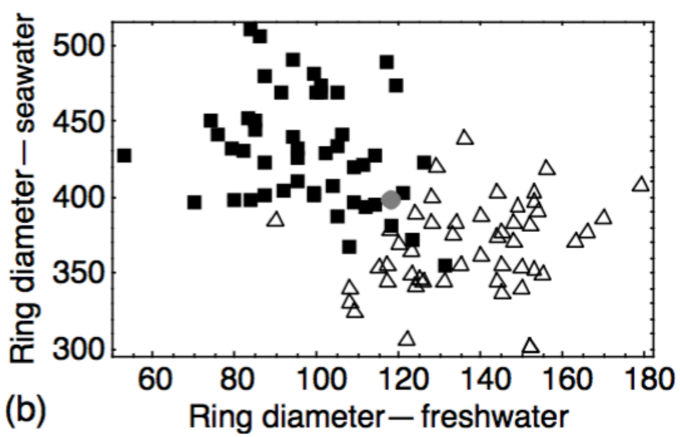
Learning rate: small constant value in the first four iterations so that the codebook vectors **find a good orientation**.

Then it is increased to 0.25 at the fifth iteration to speed up the convergence.

From this high value, the step length is slowly decreased until the map converges.



Source: (Samarasinghe, 2016)



SOM – Example 1 – Phase 1

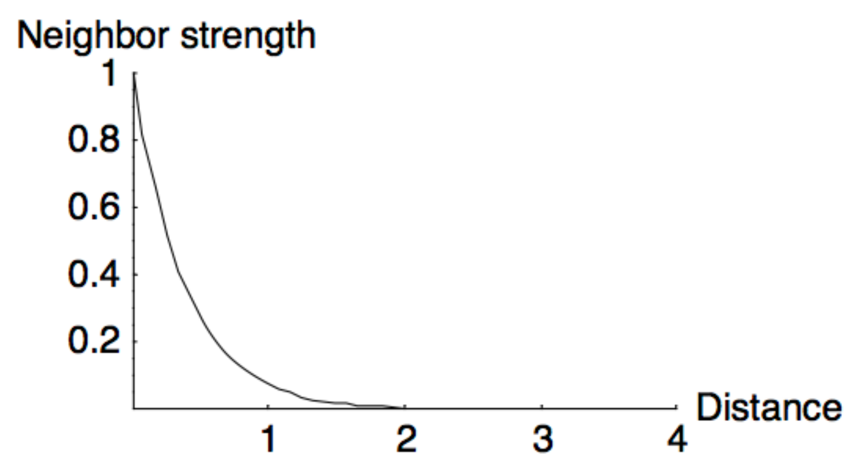
Neighbour strength function:

$$
\begin{aligned}
NS &= \mathrm{Exp}[-0.1d] \qquad \text{if } t < 5 \\
&= \mathrm{Exp}\left[-\frac{(t-4)}{10}d\right] \quad \text{otherwise}
\end{aligned}
$$

Neighbour strength: during the first four iterations, all neurons on the map are neighbors of a winning neuron and all neighbors are strongly influenced.

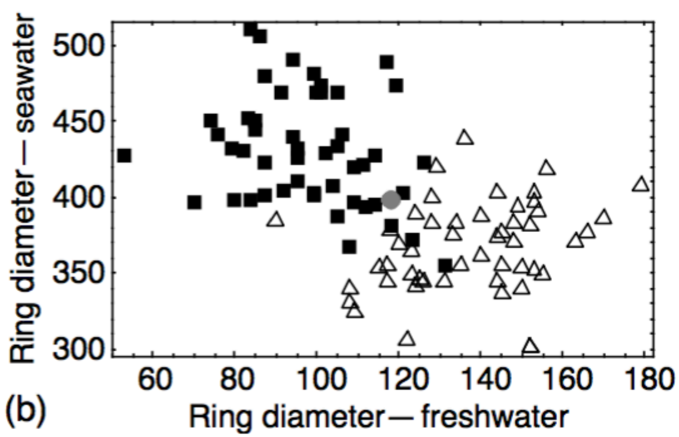
The stronger influence on the neighbors in the initial iterations makes the network conform to a nice structure and avoids knots.

From the fifth iteration, the influence on neighbors decreases with iterations.



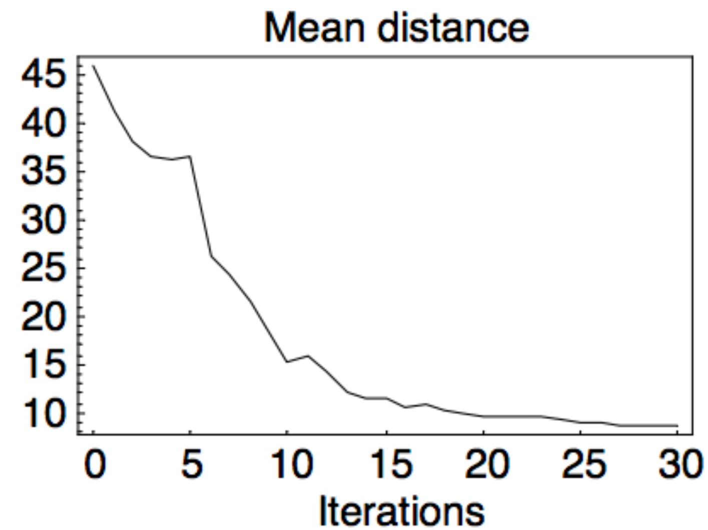
Neighbor strength in relation to distance from winner after 30 iterations.

Source: (Samarasinghe, 2016)



SOM – Example 1 – Phase 1

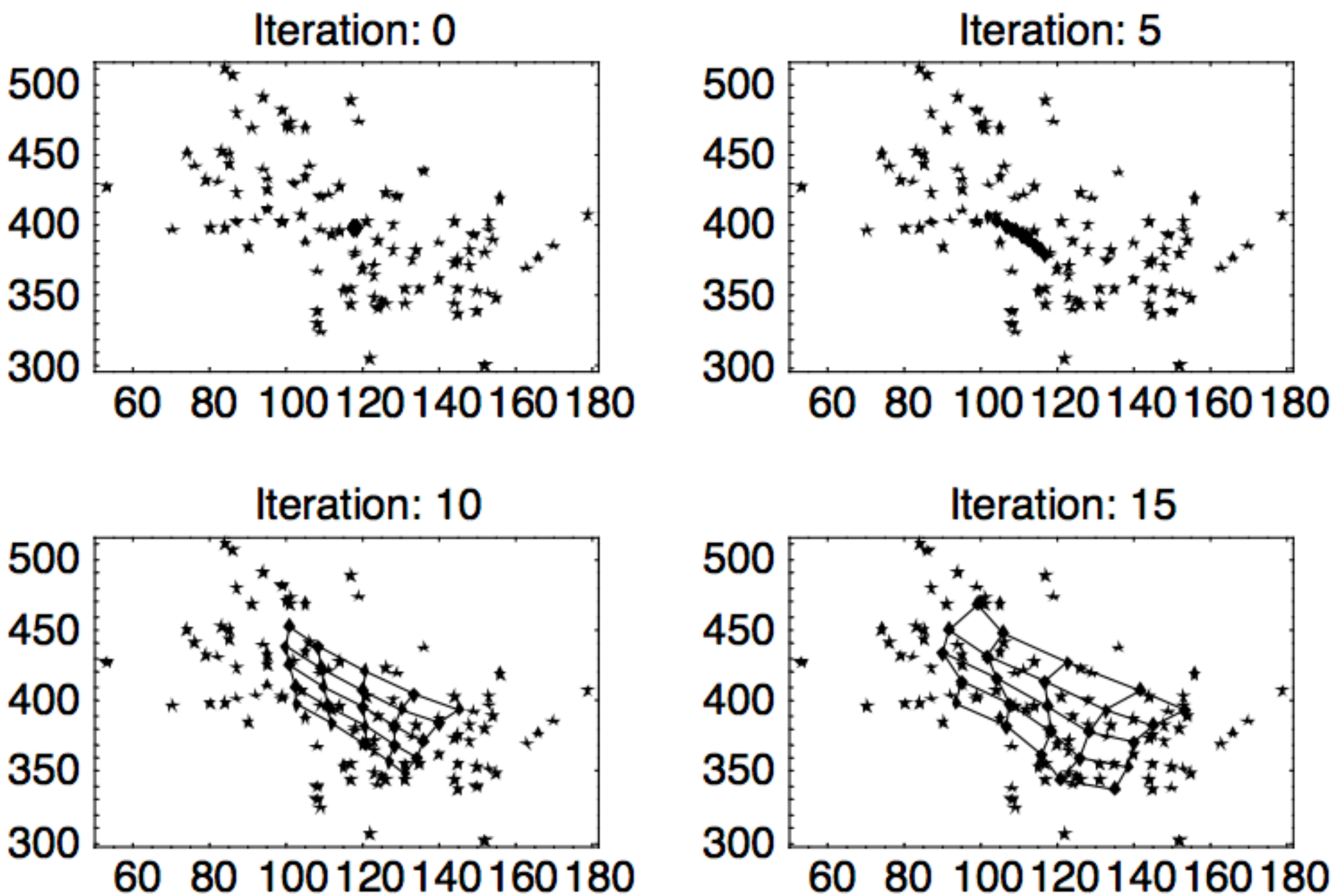
Mean distance:



The network has reached stability in about 30 iterations and at this stage only the winner and the nearest neighbors are active

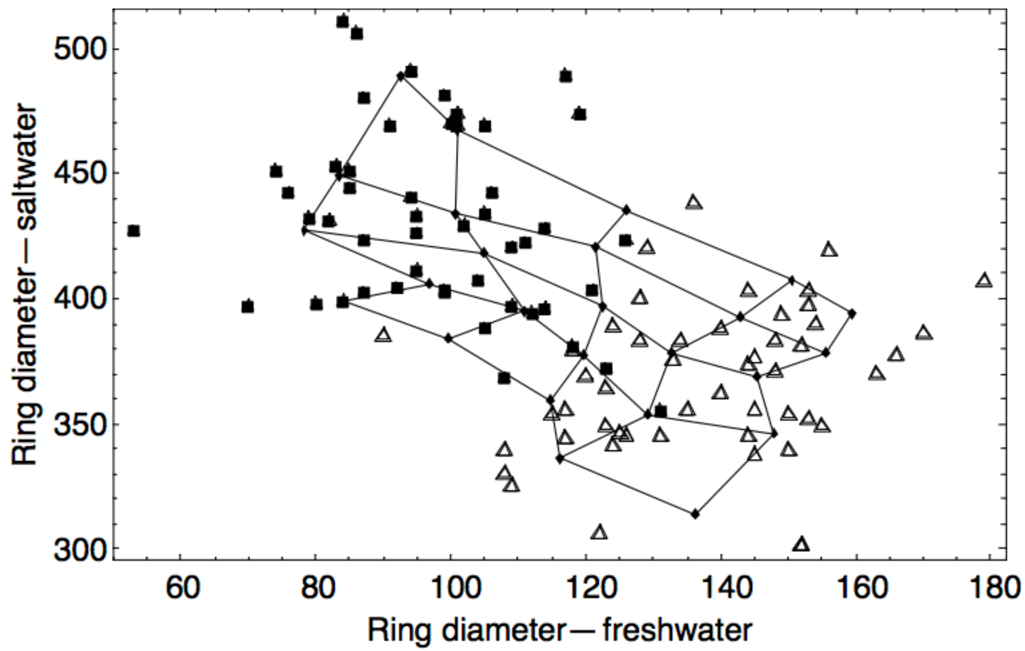
Source: (Samarasinghe, 2016)

SOM – Example 1 – Phase 1



Source: (Samarasinghe, 2016)

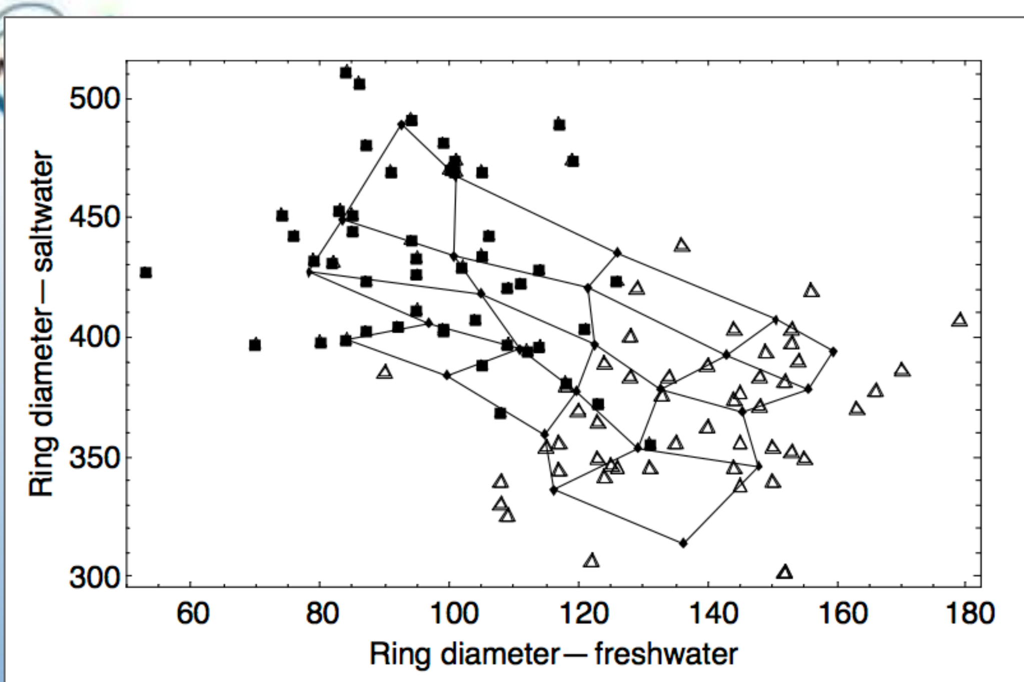
SOM – Example 1 – Phase 1



Codebook vectors of the trained map at the completion of the **ordering phase** superimposed on input data (after 30 iterations).

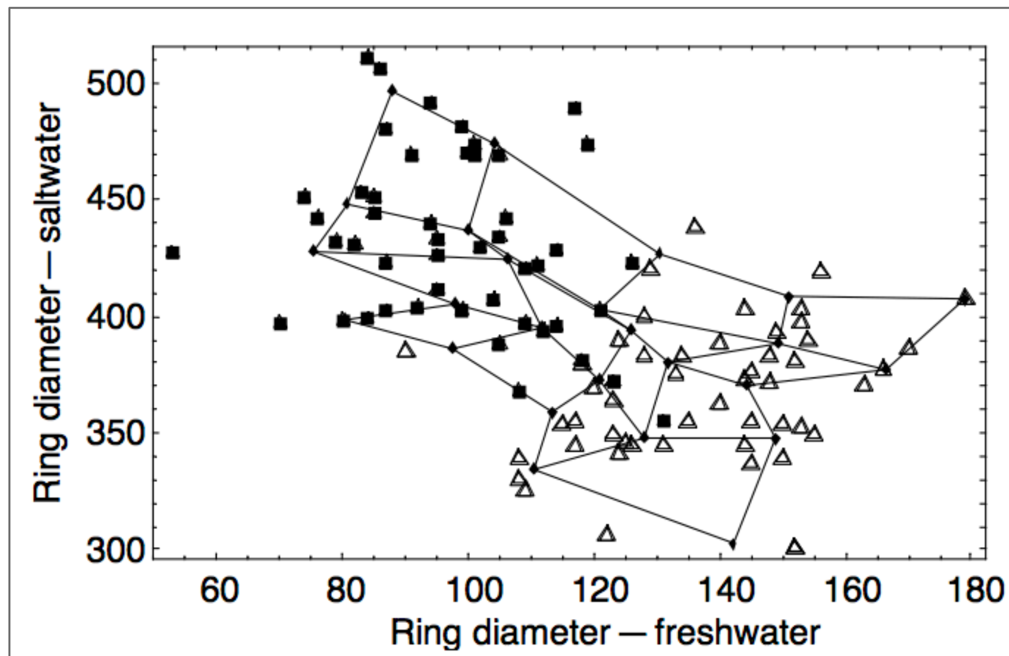
Source: (Samarasinghe, 2016)

SOM – Example 1 – Phase 2



Result of the **ordering phase.**

Result of the **covergence phase.**

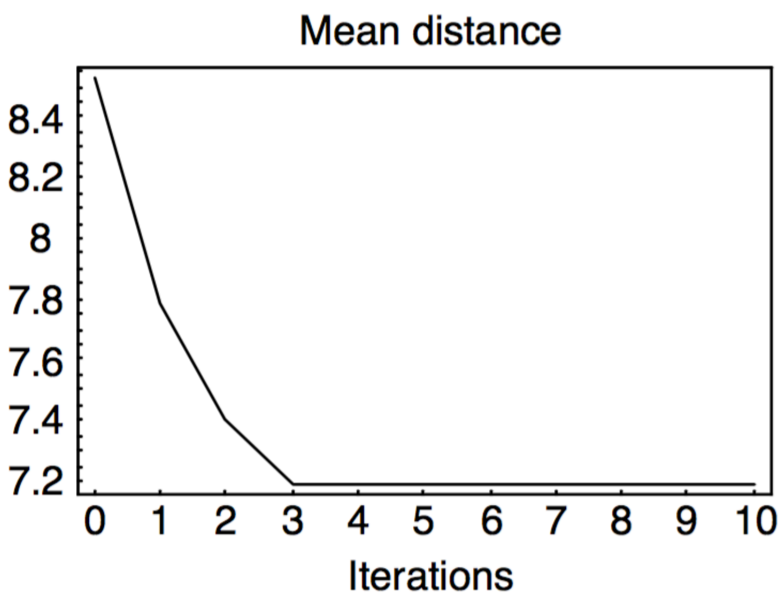


Source: (Samarasinghe, 2016)

SOM – Example 1 – Phase 2

The trained map was trained further in **batch mode** for ten epochs.

The neighbor strength is limited to the winning neuron.



The training performance with respect to reduction in mean distance which indicates that the map has now converged.

Source: (Samarasinghe, 2016)

SOM – Example 1 – Clusters

Because there are two classes of salmon (Canadian and Alaskan), **a cluster of codebook vectors**, not a single vector, **defines each class**.

This gives the map its ability to form **nonlinear cluster boundaries**. This cluster structure can be used to discover **unknown clusters** in data.

	1	2	3	4	5
5	**2×1**	**4×1**	**2×1** 1×0	8×0	5×0
4	**2×1**	**6×1**	1×0	6×0	4×0
3	**4×1**	**2×1**	**2×1**	3×0	4×0
2	**7×1**	**6×1** 1×0	**2×1** 3×0	3×0	5×0
1	**3×1**	**4×1**	**3×1** 1×0	**1×1** 1×0	4×0

Canadian salmon (class-0) are mapped to the right side and the Alaskan salmon are mapped to the left side

Source: (Samarasinghe, 2016)

SOM – Example 1 – Clusters

Because there are two classes of salmon (Canadian and Alaskan), **a cluster of codebook vectors**, not a single vector, **defines each class**.

This gives the map its ability to form **nonlinear cluster boundaries**. This cluster structure can be used to discover **unknown clusters** in data.

	1	2	3	4	5
5	**2×1**	**4×1**	**2×1** 1×0	8×0	5×0
4	**2×1**	**6×1**	1×0	6×0	4×0
3	**4×1**	**2×1**	**2×1**	3×0	4×0
2	**7×1**	**6×1** 1×0	**2×1** 3×0	3×0	5×0
1	**3×1**	**4×1**	**3×1** 1×0	**1×1** 1×0	4×0

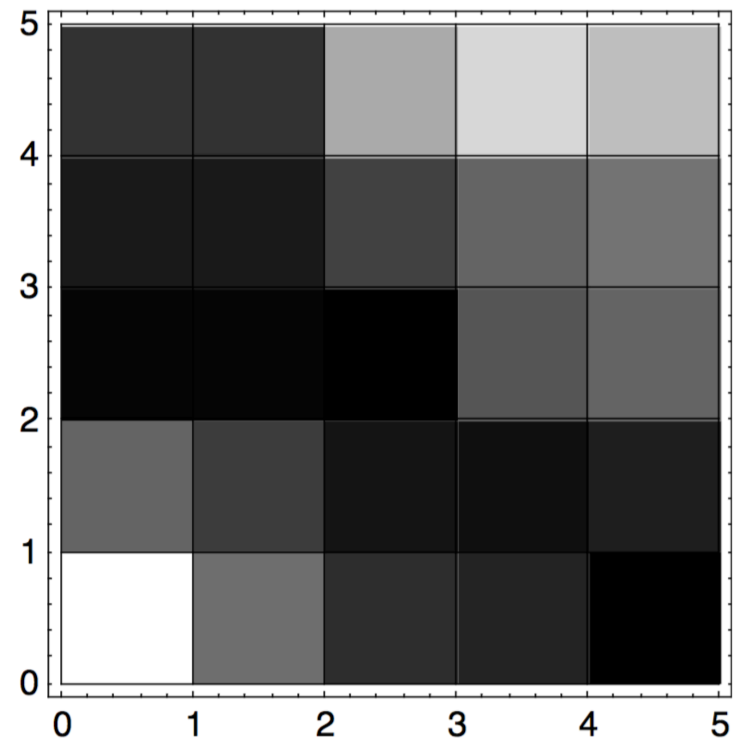
Canadian salmon (class-0) are mapped to the right side and the Alaskan salmon are mapped to the left side

Source: (Samarasinghe, 2016)

SOM – Example 1 – U-Matrix

From the trained map, the average distance between a neuron and its neighbors is called **unified distance** and the matrix of these values is called the **U-matrix.**

	1	2	3	4	5
5	2×1	4×1	2×1 1×0	8×0	5×0
4	2×1	6×1	1×0	6×0	4×0
3	4×1	2×1	2×1	3×0	4×0
2	7×1	6×1 1×0	2×1 3×0	3×0	5×0
1	3×1	4×1	3×1 1×0	1×1 1×0	4×0

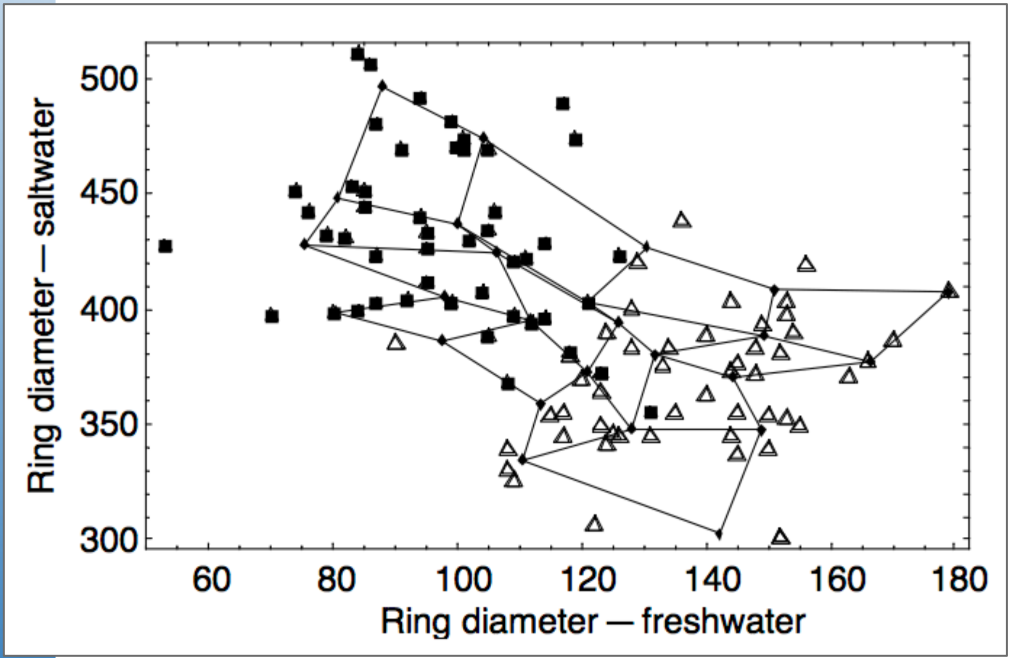


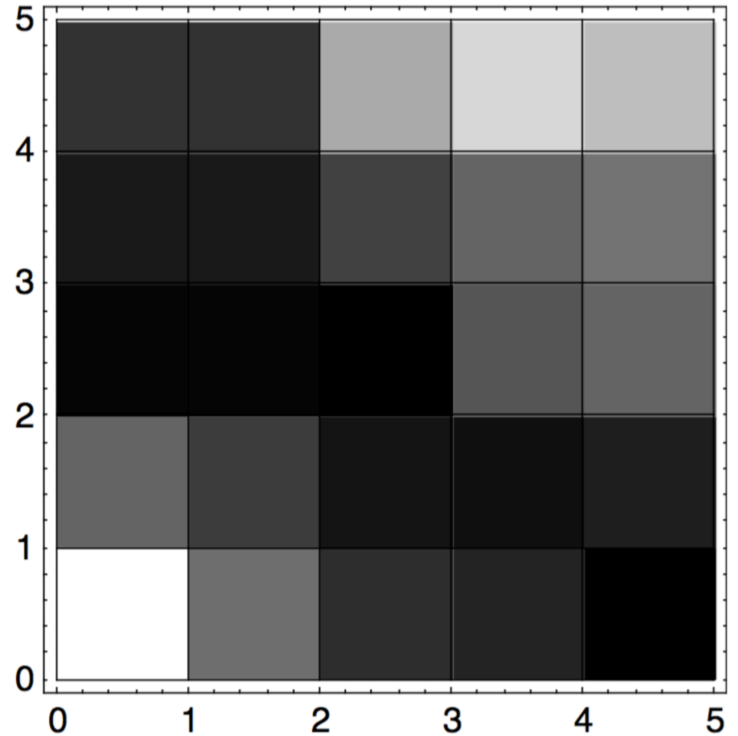
Darker colors indicate smaller distances and lighter colors indicate larger distances between neighboring neurons.

Source: (Samarasinghe, 2016)

SOM – Example 1 – U-Matrix

From the trained map, the average distance between a neuron and its neighbors is called **unified distance** and the matrix of these values is called the **U-matrix.**





Darker colors indicate smaller distances and lighter colors indicate larger distances between neighboring neurons.

Source: (Samarasinghe, 2016)

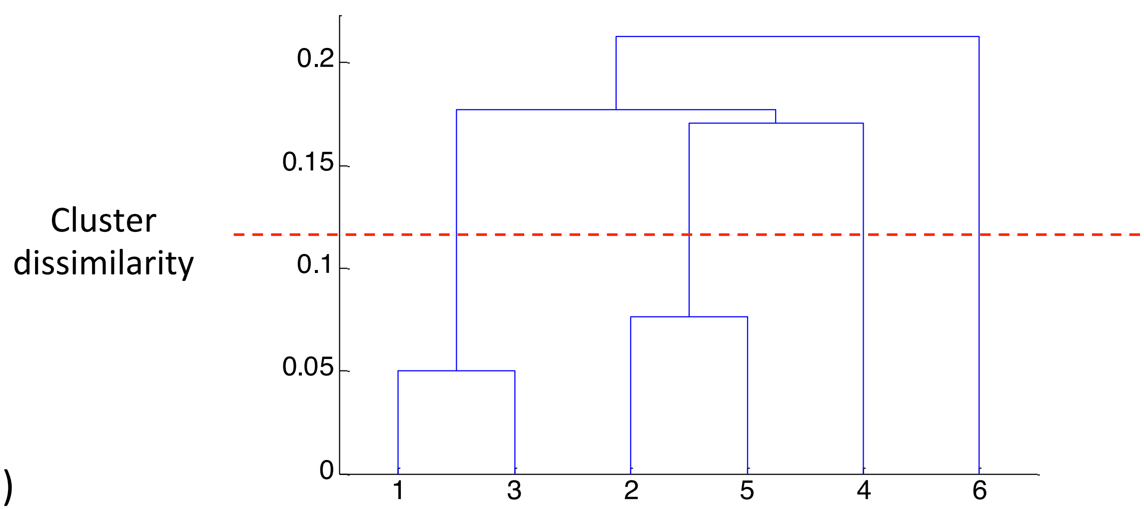
SOM – Forming Clusters on the Map

In many practical situations, the number of clusters is unknown.

How many distinct clusters are present in the SOM map?

U-matrix: can be used to find borders between data. The larger the distance, the more likely a cluster boundary exists between the vectors.

Any established clustering method can be used for clustering the codebook vectors, such as **hierarchical** clustering (dendrogram) or **K-means** clustering.



Source: (Samarasinghe, 2016)

References

(Esling and Agon, 2012) Esling, Philippe, and Carlos Agon. "Time-series data mining." *ACM Computing Surveys* (CSUR) 45.1 (2012): 12.

(Aghabozorgi et al. 2015) Aghabozorgi, Saeed, Ali Seyed Shirkhorshidi, and Teh Ying Wah. "Time-series clustering–A decade review." *Information Systems.* 53 (2015): 16-38.

(Keogh and Lin, 2003) Keogh, Eamonn, and Jessica Lin. "Clustering of time-series subsequences is meaningless: implications for previous and future research." *Knowledge and information systems* 8.2 (2005): 154-177.

(Ding et al. 2008) Ding, Hui, et al. "Querying and mining of time series data: experimental comparison of representations and distance measures." *Proceedings of the VLDB Endowment* 1.2 (2008): 1542-1552.

(**Mori et al. 2016**) Mori, Usue, Alexander Mendiburu, and Jose A. Lozano. "Distance measures for time series in R: The TSdist package." *R J.* 8 (2016): 451-459.

References

(Giogino, 2009) Giorgino, Toni. "Computing and visualizing dynamic time warping alignments in R: the dtw package." *Journal of statistical Software* 31.7 (2009): 1-24.

(Keogh and Ratanamahatana, 2005) Keogh, Eamonn, and Chotirat Ann Ratanamahatana. "Exact indexing of dynamic time warping" *Knowledge and information systems* 7.3 (2005): 358-386.

(Samarasinghe, 2016) Samarasinghe, Sandhya. "*Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*." Auerbach publications, (2016).