Time Series Clustering

A B. Tech Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Wakade Yugandhar Moreshwar
(160101078)

under the guidance of

Prof. Vijaya Saradhi

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Time Series Clustering**" is a bonafide work of **Wakade Yugandhar Moreshwar** (**Roll No. 160101078**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Prof. Vijaya Saradhi**

Associate Professor,

November, 2020 Department of Computer Science &

Engineering,

Guwahati. Indian Institute of Technology Guwahati, Assam.

Acknowledgements

we would like to thank our supervisor, Prof.Vijaya Saradhi for his guidance and support since the beginning of this project.

Contents

Chapter 1

Introduction

Clustering is the method defined as the aggregation of data samples into clusters in a way that includes more similar data samples than data samples from a separate cluster. But an algorithm basically aims to optimize the similarity among intra-cluster elements and the variance among inter-cluster elements. It is an unsupervised procedure as the algorithm has no previous idea of the amount of clusters to be generated until specified. A time series is a sequence of consecutive data points at a time interval evenly spaced. The analysis of time series is distinct from comparison of static data, because time series can also comprise of several variable (multivariate time series) in the same or separate intervals. Therefore, a distance measurement is defined for the calculation of similarities and disparities between time series data points.

1.1 Motivation

Classical clustering algorithms may not work well with time series information because most early clustering operations are focused on static data where the characteristics are constant over time. Much of the real world data in the form of data streams has recently been generated. Clustering can be used to gain valuable knowledge.

1.2 Organization of The Report

This report consists of four parts. In Chapter 2, we address the algorithm of Dynamic time warping for distance measurement between two time series. We here deal with the constraint of distance scales, namely Euclidean, Manhattan , etc, which determines the distance between two time series with linear alignments.

Chapter 3 addresses our separate clustering strategies for the clustering of time series. Here we are discussing three sets of time series clusters representing three paradigms: hierarchy, model-based approach and a neural network-drawn approach.

Ultimately, together with future work, the pros and cons of these clustering approaches and distance measure will be discussed in Chapter 4.

Chapter 2

Similarity/Distance Measures

2.1 Dynamic Time Warping

2.1.1 Introduction

Dynamic time warping(DTW) is the algorithm for comparison of two time-series. DTW aligns time series in such a way that their variations are reduced to a minimum. If we have two time series A and B, each with lengths m and n where ,

$$A = a_1, a_2, ..., a_m$$

$$B = b_1, b_2, ..., b_n$$

To align these two sequences using DTW , we construct an n x m matrix where i^{th} and j^{th} entry of matrix represents the distance $d(a_i, b_j)$ between two data points a_i and b_j

2.1.2 Motivation

Any distance whether it be Euclidean, Manhattan , etc., which matches the i^{th} data point in one series with i^{th} data point in one series on the other would lead to a low similarity score. A non-linear alignment provides a more intuitive similarity measure that can project out of phase.

2.1.3 Constraints on warping path

- **Boundary Condition** -

$$i_1 = 1,\ i_k = n$$

$$j_1 = 1,\ j_k = m$$

The alignment direction begins at the bottom left and finishes at the top right corner, which means that the alignment does not partly consider one of the series.

- **Continuity** -

Given $Wi, j = (a, b)$ and $W_{i+1j+1} = (a^{'}, b^{'})$ where a–$a' \leq 1$ and b–$b' \leq 1$. i.e

$$i_t – i_{t-1} \leq 1 \ \textit{and} \ j_t - j_{t-1} \leq 1$$

This limits the allowed steps to adjacent cells in the warping direction. The direction of alignment does not leap in the index of time. Ensures that critical features are not missed by the alignment.

- **Monotonocity** -

Given $W_{ij} = (a, b)$ and $W_{i+1j+1} = (a^{'}, b^{'})$,where $a - a^{'} \geq 0$ *and* b–$b^{'} \geq 0$. *i.e.*

$$i_{t-1} \leq i_t \ \textit{and} \ j_{t-1} \leq j_t$$

. It enforces the criterion to be monotonically spaced in W , meaning that the alignment direction does not return to the index of "time" and ensures that features are not replicated in the alignment.

2.1.4 Algorithm

Algorithm 1: DTW-TABLE

1: n $\leftarrow \left|X\right|$

2: m $\leftarrow \left|Y\right|$

3: $dtw[] \leftarrow new$ [n x m]

4: $dtw(0,0) \leftarrow 0$

5: **for** $i = 1;\ i \leq n;\ i++$ **do**

6: $\quad dtw(i,1) \leftarrow dtw(i-1,1) + c(i,1)$

7: **end for**

8: **for** $j = 1;\ j \leq n;\ j++$ **do**

9: $\quad dtw(1,j) \leftarrow dtw(1,j-1) + c(1,j)$

10: **end for**

11: **for** $i = 1;\ i \leq n;\ i++$ **do**

12: $\quad$ **for** $j = 1;\ j \leq n;\ j++$ **do**

13: $\quad\quad dtw(i,j) \leftarrow c(i,j) + min\ \{dtw(i-1,j),\ dtw(i,j-1),\ dtw(i-1,j-1)\}$

14: $\quad$ **end for**

15: **end for**

16: **return** dtw

Once matrix is built, the warping path can be traced by simple backtracking from point W(n,m) to point W(1,1) following greedy strategy.

Algorithm 2: Path(dtw)

$path[] \leftarrow new\ array$

$i = rows(dtw)$

$j = columns(dtw)$

while $(i > 1)$ & $(j > 1)$ **do**

  if $i == 1$ **then**

    $j = j - 1$

  else if $j == 1$ **then**

    $i = i - 1$

  else

    if $dtw(i-1, j) == min\{dtw(i-1, j),\ dtw(i, j-1),\ dtw(i-1, j-1)\}$ **then**

      $i = i - 1$

    else if $dtw(i, j-1) == min\{dtw(i-1, j),\ dtw(i, j-1),\ dtw(i-1, j-1)\}$ **then**

      $j = j - 1$

    else

      $i = i - 1;\ j = j - 1$

    end if

    $path.add((i, j))$

  end if

end while

return $path$

2.1.5 Certain modifications :

Slope weighting – If we modify Equation to

$$w(i,j) = min[\ X * W(i-1,j), X * W(i,j-1), w(i-1,j-1)\ ] + d(i,j)$$

as X gets larger , algorithm gets biased to go towards diagonal.

Step patterns – If we modify Equation to

$$w(i,j) = min[w(i-1,j-2), w(i-2,j-1), w(i-1,j-1)] + d(i,j)$$

This forces algorithm to take one step towards diagonal after every step taken parallel to an axis.

Chapter 3

Clustering methods

3.1 Multivariate Clustering by Dynamics

3.1.1 Overview

Given a collection of m multivariate time series consisting of a collection of v uni-variate time series. Algorithm substitute every multivariate time series by a collection of v Markov Chains, these collections of Markov Chains are ranked according to decreasing sequence of distance and related collection are then grouped into clusters only if ranking parameter improves and this procedure proceeds until algorithm stumbles upon certain stopping state. [RSC02]

3.1.2 Markov Chains

Given uni-variate time series, $\{\ x[0], \ldots, x[i-1], x[i], \ldots\ \}$, where {x[i]} represents the state of the variable. The conditional likelihood of the variable X entering state j at time t, depends only on function of visited state at time t *-1.*

So we can say that $p(X[t] = j \mid x[0]..x[t-1]) = p(X[t] = j \mid x[t-1])$ and a Markov Chain can be built as a table P = p_{ij} of transition probabilities, where $p_{ij}$is the likelihood of variable entering state j where present state is i.

The probabilities of state transitions (i to j) can be computed as $p_{ij} = n_{ij} / \sum_j n_{ij}$ where n_{ij} is the frequency of the transitions (i to j) and n_{ij} is summation of the frequency of the transitions (i to j') observed in the time series. But to acknowledge previous information about transition probabilities we prefer a Bayesian estimate of probabilities.

The probability p_{ij} is estimated as

$$p_{ij} = \alpha_{ij} + n_{ij} / (\sum_i \alpha_{ij} + n_{ij})$$

where α is the prior frequency of transition (i to j) and p_{ij} is the posterior probability estimated from prior information α_{ij}.

3.1.3 Clustering process

$$\begin{array}{cccc} S & X_1 & \cdots & X_v \\ \hline S_1 & S_{11} & \cdots & S_{1v} \\ \vdots & & \vdots & \\ S_m & S_{m1} & \cdots & S_{mv} \end{array}$$

$$\Downarrow$$

$$\begin{array}{cccc} S & X_1 & \cdots & X_v \\ \hline P_1 & P_{11} & \cdots & P_{1v} \\ \vdots & & \vdots & \\ P_m & P_{m1} & \cdots & P_{mv} \end{array}$$

Fig. 3.1 The 1st step of the algorithm replaces the original time series by Markov Chains represented by transition probability tables

The algorithm is initialised by replacing the original time series by Markov chains represented by transition probability tables. This conversion process transforms each multivariate time series into a set P_k of transition probability matrices P_{kh}, one for each variable. This algorithm is agglomerative in nature. It poses each set of a transition matrices Pk as a separate cluster and then iteratively merge these clusters until a certain stopping condition is satisfied. Merging operation of two sets of the matrices $P_k = (P_{k1}, .., P_{kv}$

and $P_t = (P_{t1}, .., P_{tv}$ involves creating a new set Cn of transition probability matrices $P_n = P_{n1}, .., P_{nv}$. This new cluster formed would be still a set of v transition matrices and each transition probability matrix P_{nh} in C_n is estimated from the cumulative transition frequencies of the variable.

Merging of two Markov chains sets only happens if the resulting clustering has a higher posterior probability when these two sets are not merged.The algorithm will halt when there are no available clustering gives a partition with greater posterior probability.

3.1.4 Method

The posterior probability here as a measurement factor and a search strategy are the main keys for choice of model, in order to explore the clustering power.

We refer a partition of the m sets of Markov Chains into clusters as a model M_c, where each cluster merges m_k sets of Markov Chains. This model consists of sets of markov chains and a variable C $C_1, .., C_C$ which denotes cluster membership. Initially algorithm doesn't know the number of states of C, but it C has an upper bound because the number of clusters will never be higher than the number of multivariate time series given.

Posterior Probability

Suppose S consists only two sets of multivariate time series S_1 and S_2 ,there are two models M1 and M2 implying partitions of the data:

M1 :- two sets are merged into one cluster

M2 :- two sets are not merged.

The posterior probability of these models by Bayes' Theorem is given as :

$$p(M_c|S) = (p(M_c) * p(S|M_c))/p(S)$$

. Where $P(M_c)$ is a partition prior probability and *p(S)* is the marginal probability of the data. *p(S)* is constant because we are comparing all models over the same given data. For the purpose of maximizing $p(M_c|S)$, it is sufficient to consider $p(M_c) * p(S|M_c)$.

moreover the comparison can be done independently on the marginal likelihood $p(S|M_c)$ as all models are a priori equally likely.

Heuristic Search

As the number of partitions increases exponentially as the number of multivariate time series increases, the calculation costs will be very large for search of partitions by brute strength process. A similarity-based heuristic search is carried out to overcome this problem algorithm. The aim is to improve statistical risk, and when different clusters are combined, they have better chances for it. According to this, we can reach the maximum posterior probability partition earlier if we merge more similar clusters.

P_k is a set of v transition probability matrices.These transition probability matrices are comparable only when they are refereeing to the same variable h. So, similarity measure for these two sets of Markov Chains can be defined by calculating row by row distance between pairs of comparable transition probability table P_{kh} and P_{ih}.

S	X_1		$\cdots$		X_v	D
S_k	P_{k1}		$\cdots$		P_{kv}	
S_l	P_{l1}		$\cdots$		P_{lv}	
	$\downarrow$				$\downarrow$	
	D_{kl1}	$+$	$\cdots$	$+$	D_{klv}	$\rightarrow D_{kl}$

Fig. 3.2 Similarity measure between S_k and S_l multivariate time series consisting of sets of v transtion probability matrices

Algorithm uses Kullback-Liebler distance as similarity measure. Given p_{khij} and p_{lhij} be the probabilities of transition (i to j) in P_{kh} and P_{lh}, the Kullback-Liebler distance of the two probability distributions in row i is

$$D_{klhi} = \sum_j p_{khij} * \log p_{khij}/p_{lhij}$$

The average distance between P_{kh} and P_{lh} is $D = \sum_i D_{khij}/s_h$ then ,where s_h denoting number of states of variable X_h and the overall distance between two sets S_k and S_n is $\sum_h D_{klh}$.

MBCD sorts the calculated pair wise distances between sets of transition probability tables and tries to merge the nearest sets. Algorithm verifies if the new model M_c where two sets of Markov Chains has to be merged, has more posterior probability than the model M_{c+1} where these sets are not merged. This substitutes the two sets of Markov Chains with a cluster formed by merging them if the probability of model M_c i.e. $P(M_c\big|S)$ is higher than $P(M_c + 1\big|S)$ and If this is not the case, the algorithm will take into account second, third and so forth. This lasts until there is no more integration.

Another thing to note is that the algorithm uses posterior probability as sorting criteria, and this similarity measure which is only for heuristic guides to check for potential partitions.

3.2 Self Organizing Maps

3.2.1 Overview

Self-organizing map is a special type of artificial neural network used for higher-dimensional data visualization. SOM is trained using unsupervised learning method by mapping higher-dimensional data to create a realistic co-ordinate system similar to input subspace. The algorithm requires the use of competition to learn. It is known as the winner neuron that is most similar to input data. Winner neuron determines the neuron weight compared with the neighborhood's other neurons.[Koh90][LXY08]

3.2.2 Training process

Algorithm training involves random weight vector initialisation. Weight vectors are then calibrated for neighboring neurons.The weight vector change is initially drastic, but becomes less significant as the practice progresses. It relies on a monotonically decreasing (alpha) parameter(learning coefficient). With the help of the neighborhood function, the algorithm determines which neurons to scale and how much to scale. The winning neuron is pulled along with neurons in his vicinity to the input level, while other neurons gets the least affected.This results in the formation of a cluster center that includes each sample input mapped to that specific neuron.

As this training progresses, the neighborhood radius also decreases with each epoch. Resultant neighborhood has fewer and fewer neurons, allowing neurons to converge where they should be and not be influenced by other neuron winners.

3.2.3 Competative learning

Let $x = [x_1, x_2, ..., x_n]^T$ be the n–dimensional input vector and reference weight vector of equal dimension is denoted by: $m = [m_1, m_2, ..., m_n]^T$ The closest possible match for input vector with the weight vectors is given by,

$$\|x - w_c\| = min_i\{\|x_i - m_i\|\}$$

Unfortunately there is no closed-form solution is usually possible for optimum position of m, so iterative methodology strategies must be employed.

For computation of distance of every reference vector from input vector algorithm uses square-error criterion-

$$D_j = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_i - w_{ij})^2$$

where, w_{ij} denotes i^{th} weight value for j^{th} reference vector

x_i denotes input vector value at i^{th} coordinate

3.2.4 Co-operation

The winner neuron determines the surrounding area and locates the topological neighborhood hub.The topological neighborhood should be decreasing function of lateral distance s_{ci} between neuron c (winner neuron) and neuron i.The lateral distance is defined by Euclidean measure :

$$S_{ci} = \|r_c - r_i\|$$

, here r_i is position of neuron i and r_c defines center of neighborhood i.e. winning neuron c

Topological neighborhood size decreases as lateral distance increases. Later this help for convergence of neighborhood.

3.2.5 Weight adaptation

The weight vectors of neurons in the neighborhood is updated as follows:

$$m_i(t+1) = m_i + h_{ci}(t)[x(t) - m_i(t)]$$

Where,$h_{ci}(t) = \alpha(t)$ within the neighborhood and $h_{ci}(t) = 0$ otherwise.

$$h_{ci} = h_0 \, exp(-\|S_{ci}\|^2/\sigma^2)$$

Whereas $h_0 = h_0(t)$ and $\sigma = \sigma(t)$ as suitable decreasing functions of time.

3.3 Hierarchical Clustering

The existing hierarchical clustering algorithm for time serial information was outlined in this chapter. Hierarchical clustering is defined by the creation of a binary-tree-like structure composed of clusters of specific order.[Lia05]

3.3.1 Overview

Such clusters are created by grouping data samples based on their calculation of similarity. The distance matrix is used as clustering criterion for determining similarities. The clustering hierarchy can be interpreted to see how cluster is combined and split into a dendrogram , in which each node reflects the cluster and leaf node comprises one single data test.

The hierarchical cluster algorithm comprises of two types. Agglomerative clustering requiring the bottom-up building of tree. Each data sample starts by putting it in its own separate cluster. The nearest pair of clusters then merges after each epoch until all information samples are in one cluster. The other technique is division clustering using a top down approach of cluster creation.

Algorithm

1: **Given:**

2: A set X of time series $x_1, x_2., , ., x_m$

3: A distance function $dist(g_1, g_2)$

4: **for** $i = 1$ to m

5: $\quad g_i = x_i$

6: **end for**

7: $G = \{g_1, ..., g_n\}$

8: $l = m + 1$

9: **while** $G.size > 1$ **do**

10: $\quad (g_{min1}, g_{min2}) = minimum\ dist(g_i, g_j)\ for\ all\ g_i, g_j\ in\ G$

11: $\quad remove\ g_{min1}\ and\ g_{min2}\ from\ G$

12: $\quad add\ \{g_{min1,min2}\}\ to\ G$

13: $\quad l = l + 1$

14: **end while**

Originally, every data sample is allocated to a different atomic cluster of its own. The distance matrix is then created by calculating all pair-wise distance between the clusters. Once all distances are measured, the clusters which has the shortest distance between them are combined into a new cluster. Those two clusters will then be replaced and the newly formed cluster will be introduced. With the subsequent iteration procedure, all distances from this new cluster are estimated and the distance matrix modified. This method is carried out until all data samples are combined to form a single cluster or until termination criteria is met.

3.3.2 Cluster distance measure

Distance between two clusters is measured from clusters to clusters as the length of a straight line. Typically, Euclidean distance, Dynamic time warping is used to seek similarities.

The single linkage algorithm calculates similarity as a distance between the nearest data point pair of various clusters i.e. smallest distance between an element in one cluster and an element in the other

$$d(C_i, C_j) = min\{d(x_{ip}, x_{jq})\}$$

The similarly complete linkage algorithm uses the most distant pair of data points.

$$d(C_i, C_j) = min\{d(x_{ip}, x_{jq})\}$$

And average linkage algorithm uses average distance between elements in one cluster and elements in other.

$$d(C_i, C_j) = avg\{d(x_{ip}, x_{jq})\}$$

The hierarchical algorithm is not also restricted to time series of equal length. Dynamic Time Warping is used for distance computation of uneven time series.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Hierarchical clustering has some drawbacks, such as the reversal of the previous stage of merger. After a group has been allocated the time series, membership of that time series can not be modified. Hierarchical clustering algorithm has high time complexity as algorithm calculates distance for every pair of time series, so it is probably not appropriate for huge data sets. The original structure has a major impact on the final effects of clustering. It is also a sensitive algorithm for outliers. In case of Self organizing map, requires a large and sufficient size of dataset for training in order to develop desired clusters.Also The outcome of the dynamic time warping algorithm is not optimal, since it is susceptible to distortion in the direction of time. If two data point having identical values but one of them lies on peak and other lies on valley part of sequence should not be mapped together.

4.2 Future Plans

We are planning study other neural network based time series clustering algorithms in future.

References

[Koh90] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep. 1990.

[Lia05] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38:1857–1874, 2005.

[LXY08] Luzhou Liu, Jian Xiao, and Long Yu. Interval self-organizing map for nonlinear system identification and control. In Fuchun Sun, Jianwei Zhang, Ying Tan, Jinde Cao, and Wen Yu, editors, *Advances in Neural Networks - ISNN 2008*, pages 78–86, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[RSC02] Marco Ramoni, Paola Sebastiani, and Paul Cohen. Bayesian clustering by dynamics. *Machine Learning*, 47(1):91–121, Apr 2002.