Early Classification of Multivatiate Time Series

Azouaoui Melissa
Haouch Lahcen
Joudelat David
Zinabi Yassine

1er juin 2016

M1 Computer Science
UVSQ

Supervisors :

Zeitouni Karine
Mousheimish Raef

Table of contents

1 Introduction

Early classification of time series is about extracting meaningful data in order to predict a certain situation as early as possible. It is critical in some time-sensitive applications such as anomaly detection,health informatics and critical decision making.

The study and classification of multivariate time series introduces new parameters, connections and data, which allows us to yield substantial results in terms of quality and accuracy.

The idea behind the method was inspired from the early classification of univariate time series. In order to extend this method to multivariate time series : from a one dimentional point of view, to the multidimentional aspect of a time serie, we have to, extend the concept of univariate shapelets to multivariate ones, which are multidimentional with a distance threshold along each dimension.

This paper is based on the work of Mohamed F Ghalwash and Zoran Obradovic, described in their research article titled : Early classification of multivariate temporal observations by extraction of interpretable shapelets.

2 Multivariate time series

A multivariate time series with n attributes, is defined as T = [T1, T2 , ... , Tn], where Ti [t] is the value of the i th attribute at a timestamp t. If l is the length of the timeseries, then it is represented by a $l \times n$ matrix.

A multivariate shapelet is defined as $f = (s, l, \Delta, cf)$. The vector s =[s1, s2, ... , sn] where si[t] is the value of the i th attribute of the shapelet at a timestamp t, l the length of the shapelet, Δ the n-dimensional distance threshold, and cf the class of the shapelet.

The distance between a multivariate shapelet f and a multivariate time series T is a vector defined as : dist(s, T) = [dist(s1, T1), dist(s2, T2), ... , dist(sn, Tn)] where :
dist(si, Ti) is the minimum distance between si and all subsequences of Ti of same length as si, i.e the minimal distance between s and T for each dimension.

The distance threshold $\Delta = [\delta 1, \delta 2, ..., \delta n]$ is computed using by algorithm 1.1 additional file : ecmts algorithms.

The distance threshold divides the dataset into two groups. A group DL containing only time series of same class as the shapelet and a group DR of time series with a different class.

The entropy of a dataset is computed as : $-\sum_{c \in C} \frac{m_c}{M} log(\frac{m_c}{M})$, where m_c is the number of time series of class c and M is the number of time series in the dataset.

The information gain of the shapelet f is computed as :
$IG = Entropy - \frac{M_L}{M} EL - \frac{M_R}{M} ER$ where Entropy is the entropy of the current dataset and, EL and ER are the entropy of DL and DR.

3 Multivariate shapelet extraction

The learning algorithm is described in section 1 additional file : ecmts algorithms. The extraction algorithm iterates over each time series and extracts all multivariate shapelets of length in the specified range. For each multivariate shapelet, it computes the distance with every time series. We know that each distance is a vector of length N so the distances between a multivariate shapelet and all time series in a dataset of length M, is a matrix with $N \times M$ dimensions.

Afterward the method computes the distance threshold and utility score for each multivariate shapelet, then selects the shapelets with the highest information gain, that cover the time series in the learning dataset.

To compute the distance threshold of a shapelet, we need to provide a way to compare two multi-dimensional distances. Therefore, two multidimensional distances

$d1 = [d_1^1, d_2^1, ..., d_2^N]$ and $d2 = [d_1^1, d_2^1, ..., d_2^N]$ are defined to be ordered according to the following criterion :

$d1 < d2 \Leftrightarrow dj1 < dj2\ j = 1...N$ (5)

Equation 5 requires all N dimensions of d1 to be less than all corresponding N dimensions of d2. Therefore, we would require all N dimensions to be less than the shapelet's threshold. This way, the method would try to find a pattern very similar to the shapelet at hand, which could lead to overfitting. In order to prevent overfitting, Equation 5 is relaxed and redefined to be partially ordered according to the following criteria :

$d1 < Perc\ d2 \Leftrightarrow d_1^{qj} < d_2^{qj} \forall j = 1...Perc \times N$ where$Perc \in]0, 1]$. (6)

4 Multivariate classification

This algorithm is explained in section 2 additional file : ecmts algorithms.

If the length of the shortest shapelets extracted is l, then we need to observe l time points in order to classify the time series.

The classification method initially reads l time stamps from the time series, after that it gets the highest-ranked shapelet based on the information gain. If the shapelet covers the current stream of the time series then the time series is classified as the class of the shapelet and the prediction is done.

Otherwise, it gets the next shapelet from the ranked list and repeats the process. If none of the shapelets cover the current stream of the test time series the method reads one more time stamp and continues classifying the time series.

If the method reaches the end of the time series and none of the shapelets cover it, the method marks the time series as a not-classified .

5 Early classification of multivariate time series

5.1 Software architecture

The software structure is shown in figure 1 : Software diagram.

The GUI is the data entry point. Through this interface, we will provide the algorithm with name files, preferences and parameters such as : minimum and maximum shapelet length and the parameter perc, which stand for percentage and is mainly used for comparison when it comes to multivariate time series and multivariate shapelets.

The GUI is able to display multivariate time series and multivariate shapelets. In order to achieve that, the JFreeChart java library was used.

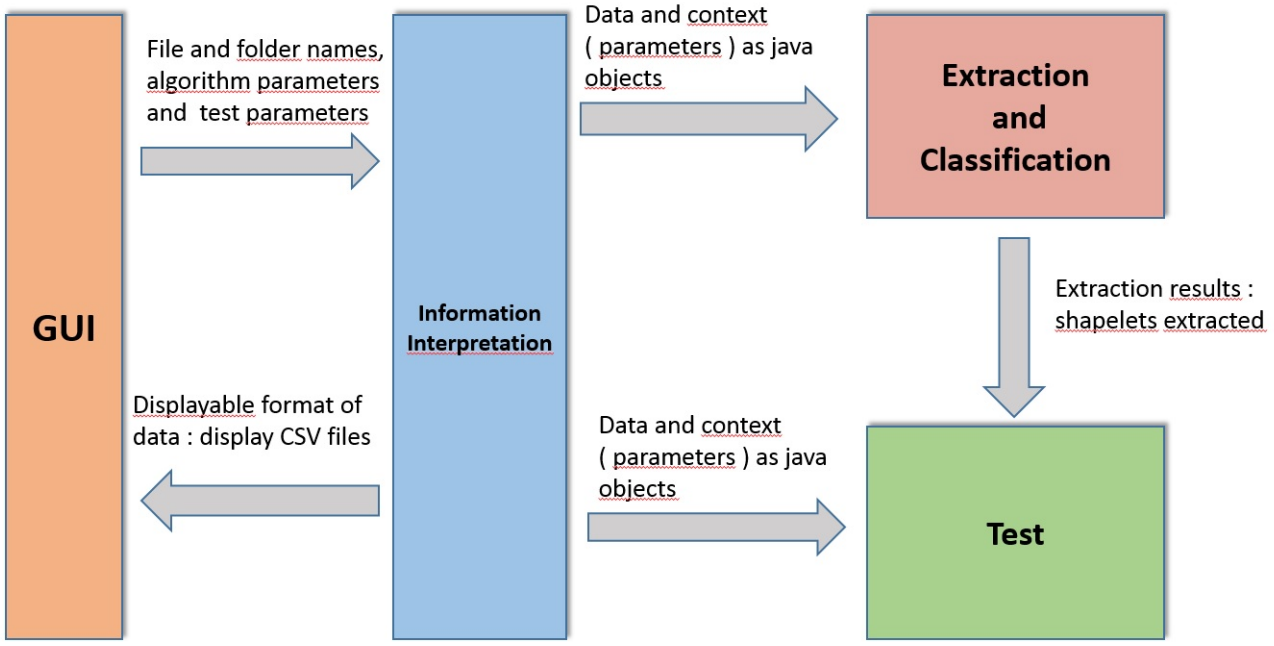
Upon the information collected by the GUI, the information interpretation module, will access and read the CSV files via the OpenCSV library. This module will then process the data and the parameters into java objects, ready to be used by the extraction and classification module.

This separation between the information interpretation and the actual processing of the data, allows complete isolation between the data format and the algorithm. By changing the way the data is presented, the algorithm module will not undergo any changes. It will be compatible, provided that the interpretation module will be altered to fit the new data template.

Finally, the extraction and classification module applies the multivariate shapelet extraction and the multivariate classification algorithms on the data provided by the information interpretation module, in order to extract shapelets with the most information gain, from a previously classified dataset, and classify an unclassified dataset or time series.

After the data has been processed by the algorithm, the findings (i.e the set of extracted shapelets) will be transmitted to the test module, where the accuracy, sensitivity and F-score will be computed and the confusion matrix will be calculated for each class of the training dataset, in order to rank and evaluate the efficiency of the algorithm.

FIGURE 1 – Software diagram



5.2 Module features

5.2.1 GUI features

- Enter algorithm parameters (shapelet length, minimal utility score, perc).
- Display time series and extracted shapelets.

5.2.2 Information interpretation module features

- Process the CSV files into the used data structures and required java objects.
- Collect the information gathered by the GUI forms to be used in the algorithm.
- Save the extracted shapelets as CSV files.

5.2.3 Extraction and classification algorithm features

- Shapelets extraction from a dataset, to be used in multivariate time series classification.
- Classification of a dataset containing multivariate time series, using a set of extracted shapelets.

5.2.4 Test module features

- Computes the F-score of the set of extracted shapelets provided by the extraction and classification module.
- Evaluates the performance of the extraction algorithm.

5.3 Test module

5.3.1 Goal

The test part of the software is supposed to allow the user to run performance tests on the software while variating parameters, in order to obtain the best learning parameters like earliness or accuracy. This part of the software gives the user a .csv file, with results regarding time of execution of shapelets extraction or quality informations (F-score).

5.3.2 Functioning

The Test part is included in the main program, even if it could have been an entire disctinct software, since it is not necessary to the program fonctionning.

The interface view allows the user to select the <test mode> to activate it. Three mode exists for the tests :

The Variation min mode, which extract shapelets by making the minimal size of shapelets variate : the software extract shapelets having a size between $minL$ and $maxL$, then $minL+x$ and $maxL$, where x is determined by the number of iterations.

The variation max mode, which extract shapelets by making the maximal size of shapelets variate : the software extract shapelets having a size between minL and

maxL, then minL and $maxL+x$, where x is determined by the number of iterations.

The perc variation mode does extractions while variating the perc parameter.

The amount of iteration is fixed by the user.

The test product a .csv file called metrics.csv.

It contains 7 columns : Time detection : time necessary to extract the shapelets Classification time : time necessary to classify the dataset Fscore time : time necessary to execute all the function related to Fscore calculation Fscore result Specificity, sensitivity and accuracy, which are parameters used to judge the quality and behaviour of the software and to calculate the Fscore.

5.3.3 Fscore calculation

The Fscore uses the following formula :

$Fscore(prcision, recall, perc) =$
$\frac{1}{|C|} \sum_{cl \in C} \frac{2 \times precision(cl) \times recall(cl)}{precision(cl) + recall(cl)}$
Where $Precision = (Sensivity + Specificity)/2$
$Sensivity(recall) = TP/(TP + FP)$
and $Specificity = TN(TN + FP)$

TP, TN, FP, FN represent the sum of true positives, true negatives... calculated for each class.

5.3.4 Results

•$Fscore$: *Due to technical issues, the* $Fscore$ *calculation fails* :

There was a missunderstanding of the definition of True positives, True Negatives, False positives and false negatives during the conception of the test part.
TP, TN used in the software were those used as attribute on shapelets.