Fast and Accurate Template Averaging for Time Series Classification

Phongsakorn Sathianwiriyakhun, Thapanan Janyalikit, Chotirat Ann Ratanamahatana
Department of Computer Engineering,
Chulalongkorn University,
Phayathai Rd., Pathumwan, Bangkok, Thailand, 10330
{phongsakorn.sa, thapanan.ja}@student.chula.ac.th, chotirat.r@chula.ac.th

***Abstract*— Time series data are evidently ubiquitous, as we could see them in all kinds of domains and applications. As a result, various data mining tasks are often performed to discover useful knowledge, including commonly performed tasks like time series classification and clustering. Dynamic Time Warping (DTW) is accepted as one of the best available similarity measures, which has been used for distance calculation in both classification and clustering algorithms. However, its known drawback is its exceedingly high computational cost. Recently, data condensation method through template averaging is applied; each class of data can be represented by one template which could greatly speed up the classification with DTW especially in large datasets, with the trade off in lower classification accuracies. Subsequently, various attempts have been made to increase the number of representative templates to boost up the accuracies while keeping the computation complexity not too high. However, those algorithms still suffer from many predefined and hard-to-set parameters, while some require high computation time for high accuracy results. Therefore, in this work, we propose an accurate yet simple template averaging method that is parameter free and has much less computation time. The experiment results on 20 UCR time series benchmark datasets demonstrate that our proposed method can achieve a few orders of magnitude speedup while maintaining high classification accuracies.**

Keywords-component; Time Series Classification, Dynamic Time Warping, Time Series Template Averaging

I. INTRODUCTION

Time series classification is widely used in many domains such as medicine, finance, and education, among many others [1-5]. One nearest neighbor (1-NN) classifier with Dynamic Time Warping (DTW) distance function is one of the most popular time series classifiers, which has been demonstrated to work extremely well and quite difficult to beat. Although 1-NN with DTW is quite effective, it still has the known weaknesses in its high computational time especially in very large training data. Therefore, shape-based averaging technique [6] has been proposed to meaningfully reduce the number of instances in the training data, representing each class of data by a single template or class prototype through DTW averaging. The class templates are then used instead of the original training data in the classification task. This DTW averaging technique can also be used in time series clustering algorithms, such as a popular *K*-means clustering for time series data, where an average of the cluster is needed to represent the cluster center.

However, for time series classification problems, it has been demonstrated that reducing the training data to one template per class is often insufficient and could greatly reduce the accuracies, as subclasses are typically present in most real-world datasets [7-10]. Fig.1 shows an example of data in the same class with two noticeable characteristics that can be divided to two subclasses. Therefore, combining the different-shape sequences across subclasses would result in the template that loses the class's characteristics and looks like neither of the subclass sequences. Being aware of this problem, most recently, Petijean et al. [11] have proposed a novel technique to discover subclasses using clustering techniques, such as *K*-means, *K*-medoids, and Agglomerative Hierarchical Clustering (AHC), to group the data within each class into two or more subclasses before shape averaging. The experiment results show that their approach can achieve exceptionally high classification accuracy, and even beat the full 1-nearest neighbor (1NN) Classification with DTW. Nevertheless, its major drawback is its high computation time and the need in hard-to-predefined parameters.

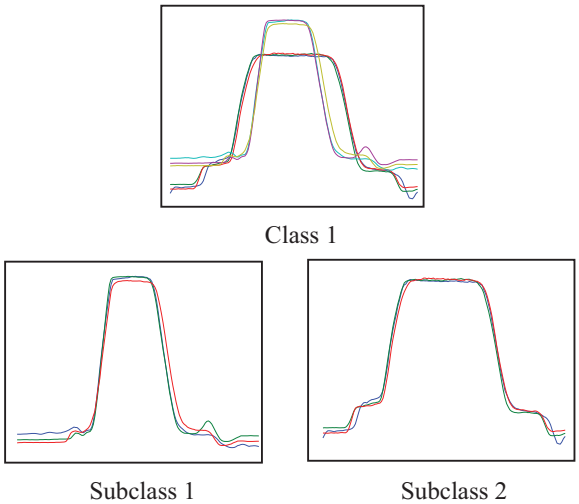

Figure 1. An example of class 1 of Gun-point dataset with two obvious characteristics that can be nicely grouped into two subclasses.

978-1-4673-8139-0/16/$31.00 ©2016 IEEE

Therefore, our main objective is to develop a very efficient and parameter-free template averaging method, yet maintaining high classification accuracies. Our proposed method simply uses the distances between data sequences to split data into subclasses before shape averaging, where nearest neighbor information is utilized to automatically determine the parameter in the averaging steps. Thus, our proposed method can quickly select the proper number of templates for each class. Then, the shape-averaging algorithm based on DTW will produce high-quality templates that would contribute to higher classification accuracies.

The rest of the paper is organized as follows. The next section gives brief details on background and related works. Section 3 provides details of our proposed method. Section 4 provides details on our experiment setup, evaluation methods, results, and discussion, and the last section concludes our work.

II. BACKGOUND AND RELATED WORK

A. Dynamic Time Warping (DTW)

For time series data, Dynamic Time Warping (DTW) is a similarity measure that has gained its popularity over Euclidean distance metric due to its higher accuracy and more flexibility in non-linear alignments between two data sequences [12]. Given two time series sequences, Q and C of length m and n, respectively:

$$Q = q_1, q_2, \dots, q_i, \dots, q_m \quad (1)$$

$$C = c_1, c_2, \dots, c_j, \dots, c_n \quad (2)$$

To align the two time series, an m by n matrix is constructed to hold cumulative distances ($\gamma(i, j)$) between all pairs of data points from these two time series sequences. The cumulative distance in each cell is calculated from a sum of the distance of the current element, $d(q_i, c_j) = (q_i - c_j)^2$, and the minimum of the cumulative distance of the three adjacent elements:

$$\gamma(i,j) = d(q_i, c_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} \quad (3)$$

Dynamic programming is used to obtain optimal alignment and cumulative distance from the elements (1,1) to (m, n). The optimal warping path $W = w_1, w_2, \dots, w_k, \dots, w_K$ is obtained by backtracking. The DTW distance is calculated by equation (4). Note that the squared root can also be omitted to speed up the calculation.

$$\mathrm{DTW}(Q, C) = \sqrt{\gamma(q_m, c_n)} \quad (4)$$

B. Dynamic time warping Barycenter Averaging(DBA)

DBA is the most recently proposed time series averaging method for DTW that has shown to outperform all existing averaging techniques [11][13]. Unlike previous approach that can average 2 sequences at a time, DBA simultaneously averages all of the sequences together. Consequently, DBA can reduce the effect of the averaging order. However, DBA requires a user-specified parameter, which is the number of iterations in the averaging process. In each iteration, DBA consists of two important steps:

- Compute DTW distances between the selected pivot sequence and other sequences to obtain the warping paths that will be used to align all of the sequences.
- Once the warping paths are obtained, they are used to align all of the sequences and calculate the average sequence, and this new averaged sequence will be a new pivot in the next iteration.

The pseudocode of DBA is shown in Table I.

TABLE I. GENERAL ALGORITHM FOR DBA [10]

Algorithm 1. DBA (D, I)

```
require D : The set of sequences with length l
require I : number of iterations

T  = medoid (D)
Do I times T = DBA_update ( D,T )
return T
```

Algorithm 2. DBA_update(D,P)

```
require P : the average sequence to refine (with length l)
require D : the set of sequences with length l

alignment = [Ø,Ø,...,Ø]  //array of empty set with length l
for each S in D
        alignment_for_S = DTW_multiple_alignment (P,S)
        for i = 1 to l
                alignment[i] = alignment [i] U alignment_for_S[i]
        end for
end for
T = sequences with length l
for i = 1 to l
        T(i) = mean(alignment[i])
end for
return T
```

Algorithm 3. DTW_multiple_alignment (R,S)

```
require R : the main sequence for which the alignment is computed
require S : the sequence to align to R using DTW

cost = DTW(R,S) //compute the cumulative matrix of DTW between R
and S
L = length(R)
alignment = [Ø,Ø,...,Ø] //array of empty set with length L
[i , j] = size (cost) //iterates over the element of R and S
while (i>1) && (j>1)
        alignment[i] = alignment[i] U S(j)
        if (i == 1) j = j-1
        else if (j == 1) i = i-1
        else
                score = min (cost[i-1][j-1],cost[i,j-1] , cost[i-1,j])
                if (score == cost[i-1][j-1] )
                        i = i -1
                        j = j - 1
                else if ( score == cost[i-1][j]) then i = i - 1
                else j = j -1
                end if
        end if
end while
return alignment
```

C. Related work

There are good quantities of works in time series averaging methods [6,11,13-15]. Started in 1996, Gupta et al. proposed method that used nonlinear alignment to average two time series data call Non-linear Alignment and Averaging Filters (NLAFF) [6]. This method averages sequences in hierarchical manner, with some limitation that the number of sequences to be averaged should be a power of two to reduce biases in the averaged results, and the order of sequences to be averaged has large effect on the shape of the template. In 2009, Prioritized Shape Averaging (PSA) [14] was proposed to heuristically rearrange the order of the sequences before averaging based on their distances. In 2011, Petitjean et al. proposed Dynamic time warping Barycenter Averaging (DBA) [13] that can average all sequences at once to reduce the effect from different data ordering and to speed up the calculation. Next, in 2012, Shape-based Template Matching Framework (STMF) [15] based on PSA that utilizes both cubic spline interpolation and weights to reduce the effect of the data that are far from the centroid. Finally, in 2014, Petitjean et al. [11] proposed a state-of-the-art Nearest Centroid Classifier (NCC) that applies DBA to 1-NN classification with DTW by using K-means and other clustering algorithms to condense or group the data before averaging, which in turn impressively improves the classification accuracies by very large margins, and in some cases even outperforms 1-NN classification accuracies that uses the whole training data. However, all of the methods that cluster data before averaging do have high computation time and have to spend a lot of time on parameter tuning to get good results.

In this work, we proposed a parameter-free shape averaging method based on DBA, which could speed up the averaging process by large margins while maintaining satisfactorily high classification accuracies.

III. PROPOSED METHOD

Our proposed method consists of three major steps: finding initial pivoting sequence for each class, subclass splitting, and shape averaging.

A. Find initial pivoting sequence for each class

In this step, we propose a quick approach to pick an initial sequence in each class, which will be used in the next step to discover subclasses and later for sequence averaging. Note that we can always select the sequence at random, but good-quality sequence will lead to better classification accuracies. Given that data T is a set of k time series sequences of the same class with the length l as follows:

$$T = \{T_1, T_2, T_3, \dots, T_n, \dots, T_k\} \quad (5)$$

$$T_n = \{T_{n_1}, T_{n_2}, T_{n_3}, \dots, T_{n_m}, \dots, T_{n_l}\} \quad (6)$$

We then sum up all the values within each sequence to calculate the mean value to this particular class of data.

$$S_n = \sum_{i=1}^{l} T_{n_i} \quad (7)$$

$$Mean = \left(\sum_{i=1}^{k} S_i\right) / k \quad (8)$$

The sequence that has its sum closest to the mean will be selected as the initial pivot in the next step.

$$Pivot\ P = argmin_{n\epsilon\{1\dots k\}} \left(|Mean - S_n|\right) \quad (9)$$

B. Subclass splitting

In typical training data, sequences in one same class may have different characteristics, so called subclass. If we average sequences from different subclasses together, it usually create the template that looks like none of the sequences being averaged from. Therefore, if we could discover subclasses within each class of the data before averaging, the resulting template will be more expressive and distinguishable, which would result in better classification results.

First, we use the sequence selected from the previous step as the pivot for the subclass splitting steps:

- Calculate Euclidean distance between the pivot sequence and all other data sequences within the class.
- Sort these distance values, and then compute adjacent discrepancies of the values. Then the standard deviation value of these adjacent discrepancies will be used as a threshold to split data into subclasses.
- Separate the data into subclasses by splitting at the sequence that has difference larger than half of the computed standard deviation [16].

The pseudocode of subclass splitting is shown in Table II.

TABLE II. ALGORITHM FOR SUBCLASS SPLITTING

Algorithm 2. SubclassSplitting (D,P)

require D : the set of sequences with length l
require P : pivot data that used to calculate distance

```
Dist = [Ø,Ø,...,Ø]
//Array of empty set with length equal to number of data in D.
for each S in D
        for i = 1 to l
                Dist[S] = Dist[S] + (P_i - S_i)^2
        end for
        Dist[S] = sqrt(Dist[S])
end for
// compute the distance between pivot and all sequences
Sort_ascending (Dist)
Diff = [0,0,...,0] //Array of zeros with length equals to number of data in D-1.
 for i = 2 to number of data in D
        Diff = Dist[i] - Dist[i-1]
end for
// compute the difference between all distances
T = Std(Diff) / 2 ;
Class = [Ø, Ø,..., Ø]
//Array of empty set with equal to number of data in D + 1.
Class[1] = 1  //Assign Class 1 to pivot data
C = 1; //variable to collect subclass number
for i = 1 to number of data in D - 1
        if (i = 1 ) Class [i+1] = 1 //Assign Class 1 to first data
        else if ( Diff [i] > T )
                Class = C+1
                C = C+1
        end if
end for
return Class
```

The main reason why we use Euclidean distance instead of DTW in this step is that other than being much faster to compute, Euclidean distance can better illustrate the differences between the two sequences than DTW could [17]. As demonstrated in Fig.2, DTW uses non-linear alignment to 'warp' two sequences with discrepancies in the X axis together very nicely, but it may not be suitable to split data into subclasses where all discrepancies should be counted; Euclidean distance can show underlying difference between sequences (two blue shaded areas) better.

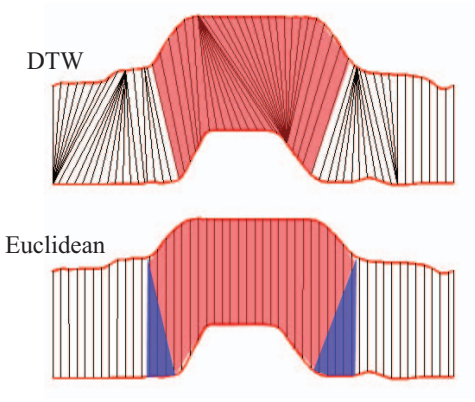

Figure 2. DTW could 'warp' two sequences with discrepancies in the X axis together very nicely and does not quite catch the underlying difference, whereas Euclidean distance can show underlying difference between sequences (two blue shaded areas) better.

C. *Shape Averaging*

In this step, we globally average all the sequences at the same time within each subclass using DBA algorithm described in section II, but instead of using DTW calculation to find medoid of each class, we use much faster method to find the pivot that can obtain satisfying template when averaging. We perform an experiment that shows the importance of choosing the good pivot for the algorithm. Table III shows the error rates of Synthetic control and SwendishLeaf dataset when using different pivots for averaging. Though our approach may not guarantee the optimal values, it *always* picks a 'better-than-average' pivot.

TABLE III. ERROR RATES OF SYNTHETIC CONTROL AND SWEDISHLEAF DATASET WHEN USING DIFFERENT PIVOTS

Order of pivot	**Error rate (%)**	
	Synthetic_control	***SwedishLeaf***
1st	0.67	21.12
2nd	0.67	21.12
3rd	1.67	21.12
4th	1.67	22.08
5th	0.67	20.00
6th	1.33	23.20
7th	1.67	21.60
8th	1.00	21.76
Our chosen pivot	**0.67**	**21.12**

Our DBA pivot is selected by the following algorithm. Given Q is a set of k time series sequences, each with length l. A k-by-k matrix, E, is created to hold Euclidean distances between all pairs of sequences (n, m) in Q, and another k-by-k matrix, NN, is created to hold nearest-neighbor points.

$$E_{(n,m)} = \sqrt{\sum_{i=1}^{l}(Q_{n_i} - Q_{m_i})^2} \qquad (10)$$

The nearest neighbor for each time series sequence is determined by looking at the distance values within matrix E. An example of NN-points calculations is shown in Fig.3, where sequences 1, 2 and 3 are of class 1 and sequences 4 and 5 are of class 2. After the distance calculation, we could scan matrix E row by row to locate the smallest distance (shaded cells). If the sequence itself and its nearest neighbor have the same class label, we will increment NN-point in the corresponding NN matrix cells by one; the rest of the cells in the NN matrix will still be zeros. After summing the points in each column, the sequence with highest NN-points in each class will be the pivot for DBA method. In this example, Seq 1 and Seq 4 with the highest sum of NN-points will be used as a pivot for class 1, and 2, respectively.

E	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
Seq 1 (1)	-	2	3	8	7
Seq 2 (1)	2	-	5	3	5
Seq 3 (1)	3	5	-	6	6
Seq 4 (2)	8	3	6	-	4
Seq 5 (2)	7	5	6	4	-

NN	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5
Seq 1 (1)	-	1	0	0	0
Seq 2 (1)	1	-	0	0	0
Seq 3 (1)	1	0	-	0	0
Seq 4 (2)	0	0	0	-	0
Seq 5 (2)	0	0	0	1	-
Sum of NN-point	**2**	**1**	**0**	**1**	**0**

Figure 3. Example of distance matrix E, and NN-point matrix. Highlighted cells denote the nearest-neighbor information.

After DBA is complete, we will obtain several templates for each class that will be used as representatives of the training data ready for further classification or other tasks.

IV. EXPERIMENT AND RESULTS

In our experiments, 20 benchmark datasets are used to compare the computation time for template construction between our method and the state-of-the-art NCC approach, as well as to compare classification error rates under DTW among our proposed method, a single-template STMF approach, and the state-of-the art NCC approach. The templates are constructed in each class of the training data, and then are used to classify test data using 1NN classifier under DTW. All of the

experiments were carried out using MATLAB software (MATLAB2010b) on desktop with core i7-2600K 3.40 GHz CPU 16GB of RAM.

A. Datasets

The 20 benchmark datasets are obtained from the UCR Time Series Classification Archive [18]. The information about size of the training data, size of the testing data, the number of classes, and length of sequences can be found in the archive. All datasets are z-normalized so that every data sequence will have a zero mean and a standard deviation of 1.

B. Preprocessing

We use the same train/test split as provided on the Time Series Classification Archive. For template construction, only the training data is used. In STMF, it averages every training data in each class to produce one template per class, whereas NCC uses *K*-means to cluster each class with various *K* values, and then averages sequences in the same cluster to produce templates, yielding multiple templates for each class. Our proposed work also constructs multiple templates, but no parameter is required from the user, then automatically decides and constructs multiple templates for each class. After template construction process, only these templates will be used as a new set of training data for 1-nearest-neighbor classification with DTW to test how accurate each approach could perform. In terms of the running time, only the time during template construction is recorded since it is the key difference among different approaches. The classification processes in all three approaches are the same, so we then only focus on the classification accuracies.

C. Results

Comparison of error rates among our proposed method, STMF, and NCC is shown in Table IV. STMF is the best method in local template averaging that uses hierarchical clustering to determine the averaging order which is one pair of sequences at a time, and eventually obtain one single template per class. NCC is the state-of-the-art method in template averaging that performs the averaging globally and obtains multiple templates per class, and could achieve exceptionally high classification accuracies. However, to reach such high accuracies, many iterations are required, giving exceedingly high computation time in many cases. Our proposed work tries to reduce the complexity of the averaging method while maintaining the high accuracies comparing to STMF and NCC.

Table IV Classification error rates on 20 benchmark datasets, comparing our proposed method with STMF and NCC.

Dataset	Error rate (%)		
	Our method	***STMF***	***NCC***
50words	29.23	40.00	31.14
Adiac	42.97	51.15	40.18
Beef	40.00	52.23	36.33
CBF	0.11	4.00	2.20
Coffee	7.14	3.58	16.07
ECG200	23.00	30.00	23.00
FaceAll	19.23	17.04	18.20
FaceFour	15.91	17.05	15.23
FISH	25.71	42.29	17.00
Gun_Point	14.67	36.67	9.40
Lightning2	13.11	44.26	12.30
Lightning7	21.92	34.25	24.30
OliveOil	13.33	23.33	14.67
OSUleaf	44.21	59.09	40.70
SwedishLeaf	21.12	31.04	21.12
synthetic_control	1.00	3.00	6.34
Trace	0	2.00	0
Two_Patterns	0	3.00	0
Wafer	4.19	36.01	2.00
Yoga	26.33	52.00	16.42

We can see from the results that the accuracies of our method generally beats STMF and are quite comparable to NCC except for 2 datasets, FISH and YOGA datasets. But when we look at the running time, our method can achieve a few magnitudes speedup, comparing with the running time of NCC, as shown in Fig.4. We also look at the number of subclass *K* of NCC that are set to run until it beats 1-NN with DTW classification, and we found that the number of subclass *K* required to beat 1-NN is so high that in many cases is the same as the number of all sequences in the training data themselves. However, some datasets have relatively small speedup like FaceAll, FaceFour, Lightning7, OliveOil and Trace dataset. The reason of this small speedup is that NCC aborts when its error rate is better than normal 1-NN with DTW. Therefore, if its error rate beats the normal 1-NN DTW at the small value of *K* parameter in *K*-mean, the computation time is low. However, in practice, we do not know *K* value in advance, so several values of *K* will generally have to be tested on.

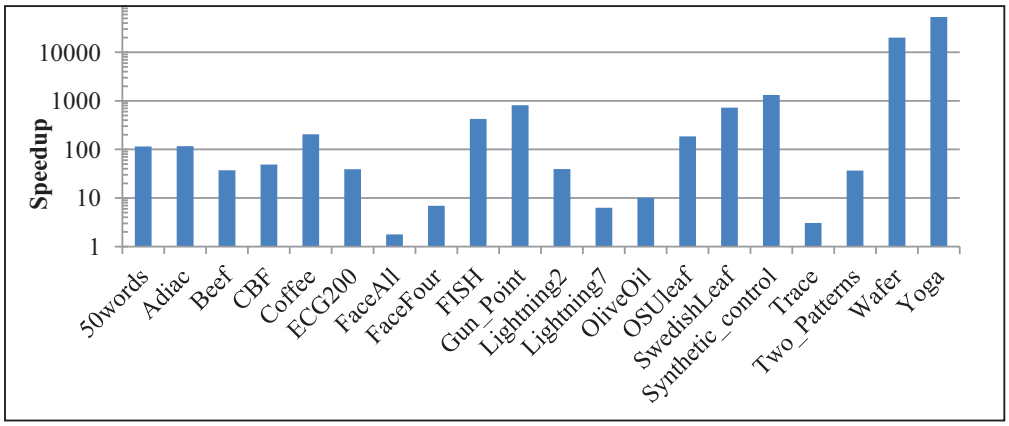

Figure 4. Speedup of our proposed method over NCC.

Overall, our method works well on most datasets. However, there are two datasets that have the accuracy worse than NCC at 5% significance level. We look into the FISH dataset and found the misclassification happened in classes 1 and 6 that are very similar in shapes and both contain outliers. One way to achieve higher classification accuracy is to use a lot of templates, since the templates obtained from our algorithms did average the outlier with other normal sequences, making the shape of the template a little bit off. For YOGA dataset, according to dataset information [7], YOGA dataset was collected from two actors (actor 1 = class 1; actor 2 = class 2) performing various yoga movements. And since both actors perform the "same" yoga movements, both classes will contain very similar-shape sequences, as shown in Fig.5, and their subclasses will also be very similar, which could be considered the limitation of our algorithm.

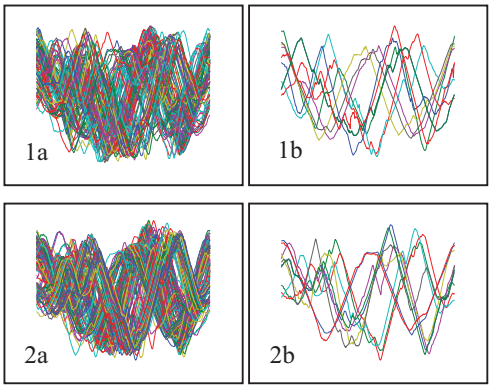

Figure 5. 1a and 2a are all data in 2 classes of YOGA dataset. 1b and 2b are sample of data in any class.

V. Conclusion

In this work, we propose a fast, accurate, and parameter-free shape averaging method that can automatically discover the proper number of subclasses within the training data, and then globally average sequences within these subclasses to generate multiple templates for classification task. The experiment results show that our proposed work can speed up the overall classification tasks by very large margin, while being able to maintain high accuracies, comparing with the state-of-the-art NCC approach. It is also observed that our proposed feature can achieve comparable classification accuracies (little lower in some and little higher in some, with no statistically significance). However, in some datasets where the error rates drop down below significant level of 5%, good speedups on all other datasets are achieved, indicating the tradeoff between the classification accuracies and the running time it could save.

Acknowledgment

This research is partially supported by CP Chulalongkorn Graduate Scholarship (to P. Sathianwiriyakhun) and the Thailand Research Fund and Chulalongkorn University given through the Royal Golden Jubilee Ph.D. Program (PHD/0057/2557 to T. Janyalikit).

References

[1] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "FTW: Fast similarity search under the time warping distance," Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2005, pp.326-337

[2] H. Sivaraks and C. A. Ratanamahatana, "Robust and Accurate Anomaly Detection in ECG Artifacts Using Time Series Motif Discovery," Computational and Mathematical Methods in Medicine, vol. 2015, pp. 1-20, 2015.

[3] P. Tsinaslanidis, A. Alexandridis, A. Zapranis, and E. Livanis, "Dynamic time warping as a similarity measure: applications in finance," in: 13th Annual Conference of Hellenic Finance and Accounting Association (HFAA), December, 2014.

[4] K. Slaninová, T. Kocyan, J Martinovič, P. Dráždilová, and V. Snášel, "Dynamic Time Warping in Analysis of Student Behavioral Patterns," Proceedings of the Dateso 2012 Annual International Workshop on DAtabases, TExts, Specifications and Objects, vol.837, pp.49-59, , 2012.

[5] T. M. Rath and R. Manmatha, "Word image matching using dynamic time warping," Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol.2, pp 521-527, June, 2003.

[6] L. Gupta, D. Molfese, R. Tammana, and P. Simos, "Nonlinear alignment and averaging for estimating the evoked potential," IEEE Transactions on Biomedical Engineering, vol.43, no.4, pp.348-356, April, 1996

[7] X. Xi, E. Keogh, L. Wei, and A. Mafra-Neto, "Finding Motifs in a Database of Shapes," Proceedings of the 2007 SIAM International Conference on Data Mining, pp. 249-260, 2007.

[8] D. Eads, D. Hill, S. Davis, S. Perkinsa, J. Ma, R.Pertera, and J. Theilera "Genetic Algorithms and Support Vector Machines for Time Series Classification," Proceeding of SPIE 4787 , Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation, vol. 74, December, 2012.

[9] C. A. Ratanamahatana, and E. Keogh. "Making Time-series Classification More Accurate Using Learned Constraints", in SIAM International Conference on Data Mining, 2004, pp. 11-22.

[10] D.J. Lee, R.B. Schoenberger, D.K. Shiozawa, X. Xu, and P. Zhan, "Contour Matching for a Fish Recognition and Migration Monitoring System", SPIE Optics East, Two and Three-Dimensional Vision Systems for Inspection, Control, and Metrology II, vol. 5606-05, p. 37-48, Philadelphia, PA, USA, October, 2004.

[11] F. Petijean, G. Forestier, G.I. Webb, A.E. Nicholson, Y. Chen, E. Keogh, "Dynamic Time Warping Averaging of Time Series allows Faster and more Accurate Classification ," 2014 IEEE International Conference on Data Mining, pp.470-479, December, 2014.

[12] E. Keogh, and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," ACM SIGKDD international conference on Knowledge discovery and data mining, pp.285-289, 2000.

[13] F. Petitjean, A. Ketterlin, and P. Gancarski, "A global averaging method for dynamic time warping, with applications to clustering," Pattern Recognition, vol. 44, no. 3, pp. 678–693, 2011.

[14] V. Niennattrakul and C.A. Ratanamahatana, "Shape Averaging under Time Warping," in 6th International Conference on Electrical Engineering/Electronics, Computer,Telecommunications, and Information Technology (ECTI-CON 2009) vol.2, pp.626-629, Pattaya, Thailand, 2009.

[15] V. Niennattrakul, D. Srisai, and C.A. Ratanamahatana "Shape-based template matching for time series data," in Knowledge-Based Systems, vol. 26, pp.1-8, February, 2012.

[16] G. Norman, J.A. Sloan, and K.W. Wyrwich, "Interpretation of Changes in Health-related Quality of Life: The Remarkable Universality of Half a Standard Deviation," in Medical Care, vol. 41, issue 5, pp.582-592, May, 2003.

[17] F. Iglesias and W. Kastner, "Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns," Energies 2013, vol.6, issue 2, pp. 579-597, January, 2013.

[18] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei and C.A. Ratanamahatana. The UCR Time Series Classification/Clustering. 2011. Homepage: www.cs.ucr.edu/~eamonn/time_series_data/