Generating synthetic time series to augment sparse datasets

Germain Forestier[1,2], François Petitjean[2], Hoang Anh Dau[3], Geoffrey I. Webb[2], Eamonn Keogh[3]

[1] University of Haute-Alsace, Mulhouse, France, germain.forestier@uha.fr
[2] Faculty of IT, Monash University, Melbourne, Australia, {francois.petitjean,geoff.webb}@monash.edu
[3] Computer Science and Engineering Dpt, University of California, Riverside, USA {hdau001,eamonn}@cs.ucr.edu

In machine learning, data augmentation is the process of creating synthetic examples in order to augment a dataset used to learn a model. One motivation for data augmentation is to reduce the variance of a classifier, thereby reducing error. In this paper, we propose new data augmentation techniques specifically designed for time series classification, where the space in which they are embedded is induced by Dynamic Time Warping (DTW). The main idea of our approach is to average a set of time series and use the average time series as a new synthetic example. The proposed methods rely on an extension of DTW Barycentric Averaging (DBA), the averaging technique that is specifically developed for DTW. In this paper, we extend DBA to be able to calculate a weighted average of time series under DTW. In this case, instead of each time series contributing equally to the final average, some can contribute more than others. This extension allows us to generate an infinite number of new examples from any set of given time series. To this end, we propose three methods that choose the weights associated to the time series of the dataset. We carry out experiments on the 85 datasets of the UCR archive and demonstrate that our method is particularly useful when the number of available examples is limited (e.g. 2 to 6 examples per class) using a 1-NN DTW classifier. Furthermore, we show that augmenting full datasets is beneficial in most cases, as we observed an increase of accuracy on 56 datasets, no effect on 7 and a slight decrease on only 22.

I. Introduction

Machine learning usually benefits from larger training sets. Small training sets lead to overfitting, while overfitting is progressively less of a problem as data quantity increases [1], [2]. For many applications, only small training sets are available. One way to address this problem is to enlarge training sets by generating synthetic (or artificial) examples. In machine learning, *data augmentation* refers to the process of creating synthetic examples in order to *augment* a dataset [3] used to learn a model.

The general idea behind data augmentation is to reduce the errors of the classifier that are due to variance, i.e. when there are too few examples to learn accurate parameters for the model; the classifier is then said to overfit. We can influence the variance by adding/removing representation bias to the classifier (see e.g. [4]); adding representation bias typically reduces variance and conversely. However, in many cases it can be easier to express our knowledge of the problem (i.e. the bias) by generating synthetic data than by modifying the classifier itself. For instance, images containing street numbers on houses can be slightly rotated without changing what number they actually are [5]. Voice can be slightly accelerated or slowed down without modifying the meaning [6]. We can replace some words in a sentence by a close synonym without completely altering its meaning [7]. Data augmentation has also been shown to improve the accuracy of handwriting recognition systems [8]. Macia et al. [9] also showed that classic algorithms, including Naive Bayes and Random Tree, can benefit from data augmentation.

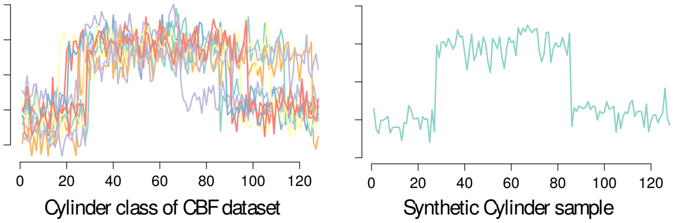


Fig. 1: Example of a generated synthetic series (right) for Cylinder class of CBF dataset (left) by averaging a set of time series taken from the class.

Augmentation with synthetic examples has also been used to address unbalanced classes (i.e. when classes of the training set do not have the same number of elements). For example, Chawla et al. [10] proposed a method named SMOTE (Synthetic Minority Over-sampling Technique) which creates synthetic minority class examples to balance the classes. Experiments showed improvement in error-rate for C4.5, RIPPER and Naive Bayes classifiers.

In recent years, good results in time series classification have been achieved by the use of Dynamic Time Warping (DTW) combined with non-parametric classifiers such as nearest neighbor (NN). However, most of the proposed methods still require large amounts of labeled training data to work effectively. Relatively little attention has been paid to the development of augmentation methods for time series classification under time warping. For example, Le Guennec et al. [11] proposed to stretch or shrink randomly selected slices of a time series in order to create synthetic examples.

In this paper, we present methods for generating a set of synthetic time series $\mathbf{D}'$ from a given set of time series $\mathbf{D}$. The addition of the synthetic set $\mathbf{D}'$ to $\mathbf{D}$ ($\mathbf{D} \cup \mathbf{D}'$) forms an augmented dataset. To create the synthetic time series, we propose to average a set of time series and to use the averaged time series as a newly created example. To achieve this goal, we developed a weighted version of the time series averaging method DBA (DTW Barycenter Averaging) [12],

如何有效的改变权重

which makes it possible to create an infinite number of new time series from a given set of time series by simply varying the weights. Moreover, we developed three methods to choose the weights to assign to the series of the dataset, so that the generated examples closely follow the distribution from which $\mathbf{D}$ is sampled. Figure 1 shows an example of a synthetic time series generated using our method for the Cylinder class of the CBF dataset [13].

In previous work, we have formulated the DBA algorithm and demonstrated that it averages time series consistently under DTW [12], [14]. We have shown that DBA can be used to speed up NN-DTW [15], by constructing the most representative time series of each class and using only those for training. In this paper, we take an opposite angle. We increase the size of the training set to improve classification accuracy. Unlike the previously proposed techniques, our new method can generate an unlimited number of synthetic time series and tailor the weights distribution to achieve diversity.

In order to evaluate the effectiveness of the newly created time series, we used augmented training sets for time series classification using the 1-NN classifier in conjunction with DTW. We will present two types of experiments using 85 datasets of the UCR archive [13] to assess our approach The first part is related to the cold start problem, which appears when very few examples are available at the outset of learning a predictive model. We show that in this case creating synthetic examples using our method is almost always beneficial. In the second part, we use synthetic time series to double the training sets size regardless of their original sizes. We show that for most of the datasets in the UCR repository [13] (56 out of 85), we increase the accuracy of the 1-NN DTW classifier just by adding newly created time series to the training set.

II. Methods

We first define the key terms that we use in this work. For our problem, each object in the data set is a time series, which may be of different lengths.

Definition 1: Time Series. A time series $T = \langle t_1, \dots, t_L \rangle$ is an ordered set of real values where L is the length. A dataset $\mathbf{D} = \{T_1, \dots, T_N\}$ is a collection of such time series.

The general intuition behind our approach is to take a set of time series from the same class in $\mathbf{D}$, calculate a weighted average $\overline{T}$, and use this average as a new synthetic time series to augment $\mathbf{D}$ (i.e. $\mathbf{D} \cup \overline{T}$). Note that an important contribution of our method is finding the weights so that we nicely follow 多种多样的、 the manifold of the data. In our case, the objects are time series and the measure is DTW, which leads to the following definition:

Definition 2: Average time series for DTW. Given a set of time series $\mathbf{D} = \{T_1, \dots, T_N\}$ in a space E induced by Dynamic Time Warping, $\overline{T}$ the average time series is the time series that minimizes:

$$\arg\min \overline{T} \in E \sum_{i=1}^{N} DTW^2(\overline{T}, T_i) \qquad (1)$$

问题是怎么求 $\overline{T}$

In this paper, we use DBA (DTW Barycenter Averaging) as the method to minimize this function [14]. DBA uses an expectation-maximization scheme and iteratively refines a starting average $\overline{T}$ by:

1) Expectation: Considering $\overline{T}$ fixed and finding the best multiple alignment M of the set of sequence $\mathbf{D}$ consistently with $\overline{T}$ (see [16] for more details about multiple alignments and DTW).
2) Maximization: Now considering M fixed and updating $\overline{T}$ as the best average sequence consistent with M.

Although DBA is only deterministic given a fixed starting average, modifying this starting time series is not sufficient to create enough diversity in the synthesized dataset. ??? If we look at the problem in a one dimensional Euclidean space, given two values, for example 4 and 6, we would like to generate n additional and different values. If we compute the arithmetic mean, we only end up with the new value 5. However, if we weight each input example and use the weighted arithmetic mean, we can compute infinitely many additional values. In this case, instead of each of data point contributing equally to the final average, some data points contribute more than others. Another issue with using a “uniformly weighted” average is that it can result in undesirable time series when the data distribution is not spherical. The generated data should ideally be located on the manifold of the distribution. Imagine that the data distribution follows an U-shape, computing the center of this U will typically construct very unlikely objects.

The remaining questions are: (1) how to compute a weighted average consistently with dynamic time warping; and (2) how to decide upon the weights to give to each times series.

A. Weighted average of time series for DTW

DBA [12] is an iterative algorithm which starts by taking one time series from the set to average (generally the medoid) and then updating this time series. Calculating the weighted average simply changes the objective function.

Definition 3: Weighted Average of time series under DTW. Given a weighted set of time series $\mathbf{D} = \{(T_1, w_1), \dots, (T_N, w_n)\}$ in a space E induced by DTW, $\overline{T}$ the average time series is the time series that minimizes:

$$\arg\min \overline{T} \in E \sum_{i=1}^{N} w_i \cdot DTW^2(\overline{T}, T_i) \qquad (2)$$

As we can see in the formula, weighting does not affect how DTW is computed, which means that it does not change the mapping that DTW forms between the $\overline{T}$ and the series in $\mathbf{D}$. For this reason, the expectation part of DBA is exactly the same as with the non-weighted version of DBA (see [12], [14], [15]). The main difference is in the maximization phase, which we describe in Algorithm 2. We record the sum of the weights associated to each element of the current average when mapped to it by DTW. Table I give the pseudocode of Weighted-DBA. To ensure reproducibility of our work, we make an implementation of weighted DBA available at [17].

TABLE I: Weighted DBA algorithm.

```
Algorithm 1: Weighted_DBA($\mathbf{D}$,$W$,$I$)
input : $\mathbf{D}$: the set of sequences to average
input : $W$: the set of weights
input : $I$: the number of iterations
$\overline{T}$ = medoid($\mathbf{D}$,$W$) // get the medoid of the set of sequence $\mathbf{D}$
do $I$ times: $\overline{T}$ = Weighted_DBA_update($\overline{T}$,$\mathbf{D}$,$W$)
return $\overline{T}$
```

```
Algorithm 2: Weighted_DBA_update($\overline{T_{init}}$,$\mathbf{D}$,$W$)
input : $\overline{T_{init}}$: the average sequence to refine (of length $L$)
input : $\mathbf{D}$: the set of sequences to average
input : $W$: the set of weights
output: $\overline{T}$: the updated mean
$\overline{T} = \langle 0, \ldots, 0 \rangle$ // sequence of length $L$
sumWeights = $[0, \ldots, 0]$ // array of size $L$
for $i = 1 \to |\mathbf{D}|$ do
    alignment = ~~DTW_alignment~~($\overline{T_{init}}, \mathbf{D}(i)$)
    for $l = 1 \to L$ do
        $\overline{T}(l) = \overline{T}(l)+$ alignment$[l] \cdot W(i)$
        sumWeights$[l]$ += $W(i) \cdot |$alignment$[l]|$
    end
end
for $l = 1 \to L$ do
    $\overline{T}(l) = \overline{T}(l)/$sumWeights$[l]$
end
```

B. Average All (AA)

学习了解这个方法

The first method averages all the input time series to create a synthetic example. We want to give different weights to all the time series to create diversity in the synthesized ones.

We first propose to sample the weights vector following a flat Dirichlet distribution with unit concentration parameter $\overline{w} \sim \text{Dir}(\overline{1})$. We used a low value for the shape parameter (0.2 in this paper) of the Gamma-distributed random variable used for the Dirichlet distribution in order to give more weight to a time series that is then used as the initial object to update in Weighted DBA algorithm (see Table I).

While the intuition behind this first method is appealing, we will see that this is often not an ideal solution, because it has potential to fill up the complete convex hull of the original data. In the case where the classes form two interlaced U-shapes, filling the inside part of the 'U' would lead to inappropriate examples. That is why the following two methods first select a subset of the time series to average.

C. Average Selected (AS)

The intuition behind our second method is to use a subset of close time series and fill their bounding boxes. Take again the case of the U-shape, as extremities of the 'U' are not close neighbors, we will not fill the inside of the 'U', thus following the manifold more consistently. The method starts by randomly choosing a time series T^* from $\mathbf{D}$ and giving it a weight of 0.5. Then, it looks for the 5 nearest neighbors of T^* and picks 2 of the 5 at random (selecting j of the k nearest neighbors, rather than all of the k nearest neighbors, in order to create variability). We assign them a 0.15 weight; thus summing up to 80%. Finally, we assign the remaining 20% uniformly across the rest of the time series (each receives a weight of $0.2/N$). If only two time series are available, the nearest neighbor receives a weight of 0.5. For this method, we also use the fact that we are generating time series in the neighborhood of $T^\star$ and initialize DBA with it.

D. Average Selected with Distance (ASD)

The third method has the same spirit as AS method but it takes into account the relative distance between the initially selected time series and its nearest neighbor. The idea is that if other time series are relatively far from T^* and its nearest neighbor, they should receive a relatively lower weight than if they were almost as close to T^* as its nearest neighbor. In some sense, we are trying to assess the density of the distribution in the neighborhood of $T^\star$ by using the distance to its nearest neighbor as a proxy for it.

As for the previous method, we first choose randomly a time series $T^\star$ from $\mathbf{D}$ and give it a weight of 1. We then assign its nearest neighbor a weight of 0.5. Next, we define an exponential decay function that maps a distance (DTW) to a weight. Our 2 points in this function are sufficient to build the function:

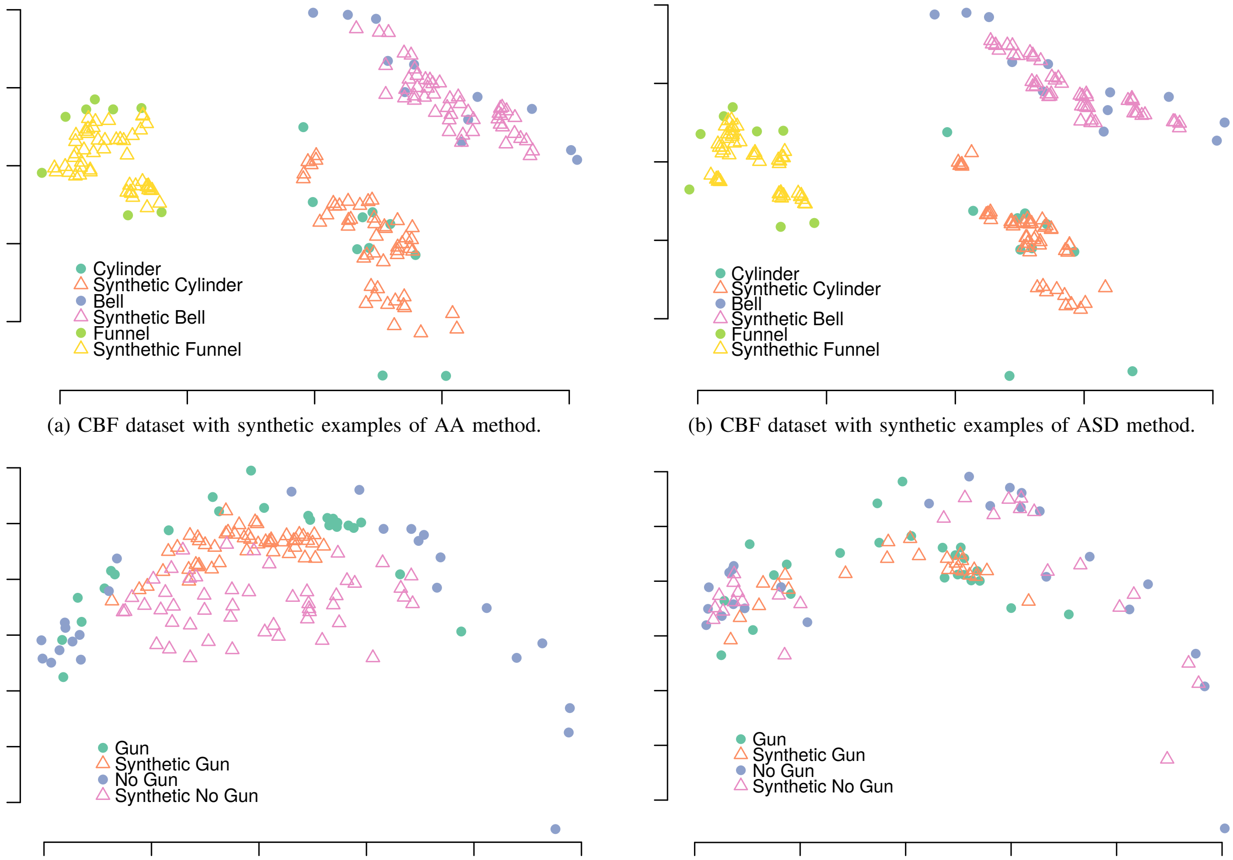
$$w_i = e^{\ln(0.5) * \frac{DTW(T_i, T^\star)}{d^\star_{NN}}} \tag{3}$$

where $d^\star_{NN}$ is the distance between $T^\star$ and its nearest neighbor. We then use this function and normalize its outputs to decide upon the weights for all time series in $\mathbf{D}$. Intuitively, we make $T^\star$ the center of gravity of our synthetization and create a power law around it whose width is decided by how close the nearest neighbor is. If it is very close, then only elements that are extremely close to $T^\star$ and its nearest neighbor will influence the generation. If it is far, then many other time series will. Here again, we initialize DBA with $T^\star$.

了解下这个、

E. Visualizing synthetic data

In order to have a spatial interpretation of the synthetic data generation, it is possible to visualize the result using Multi-Dimensional Scaling (MDS) [18]. MDS aims at placing each object in a N-dimensional space such that the between-object distances are preserved as well as possible. Using DTW on a set of time series, it is then possible to create a similarity matrix and to use MDS on it to display the set into a two dimensional space. As DTW is not a metric, we used Kruskal's nonmetric MDS [18], which only uses the rank order of elements in the dissimilarity matrix instead of the actual dissimilarities. Figure 2 gives the MDS representation of the original and synthetic objects on the `CBF` (a,b) and `Gun Point` (c,d) datasets for the AA and ASD methods. In this example, we created 50 additional synthetic examples for each of the class of the original training sets using the Average All (AA) and Average Selected with Distance (ASD) methods. For the `CBF` dataset (Figure 2(a)), averaging all the time series of the classes works well as it fills their convex hulls. Averaging only selected time series produces similar results as presented on Figure 2(b) with the ASD method.



(a) CBF dataset with synthetic examples of AA method.

(b) CBF dataset with synthetic examples of ASD method.

(c) GunPoint dataset with synthetic examples of AA method.

(d) GunPoint dataset with synthetic examples of ASD method.

Fig. 2: MDS representation of the `CBF` (a,b) `Gun Point` (c,d) datasets [13] with synthetic data generated using Average All (AA) method and Average Selected With Distance (ASD).

However, the result obtained with the `Gun Point` dataset (Figure 2(c)) illustrates the problem mentioned with the AA method when the distribution of the classes is not convex. In this case, it is more appropriate to use the local distribution of the class, as does the ASD method on Figure 2(d).

III. Experiments

A. Cold start

We want to compare the performance of the different methods when they only have access to a few instances of the learning set. To this end, we follow standard practices for statistical comparison of classifiers [19] and use the average ranking of each method over all the datasets. This allows us to assess which algorithm exhibits, on average, the best classification performance under the given configuration of available examples and to test if the superiority is statistically significant.

For each dataset, we simulate the cold start by randomly selecting a few samples from the training set. We progressively grow the size of the sub-sampled data from 2 (at least 2 are needed for our proposed methods) to the full original dataset. We then apply our algorithms to generate as many synthetic time series as the size of the sub-sampled data (i.e. we double the size of the training set). We then report the error-rate of each algorithm on each dataset as we grow the size of the starting dataset. We compare our proposed algorithms with two other methods. The *Original* method keeps the original examples only, i.e. the ones used by our methods to create synthetic examples. The *Windows Warping* (WW) method discussed in [11] involves warping a randomly selected slice of a time series by speeding it up or down. We follow the

TABLE II: Average ranking over 85 UCR datasets of the synthetic time series generation methods for 2 to 6 examples available per class.

Algorithm	**Average rank** R_j using k examples per class to generate k additional synthetic examples per class				
	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
Original	3,48	3,47	3,45	3,38	3,69
WW [11]	3,10	3,08	3,40	3,42	3,38
AA	3,20	3,16	2,95	3,30	2,84
AS	2,66	2,65	2,81	2,42	2,74
ASD	2,56	2,64	2,38	2,48	2,35
χ_F^2	20,04	17,17	24,66	34,59	38,63
R_{ori} - R_{ASD}	**0.92**	**0.81**	**1.06**	**0.89**	**1.34**

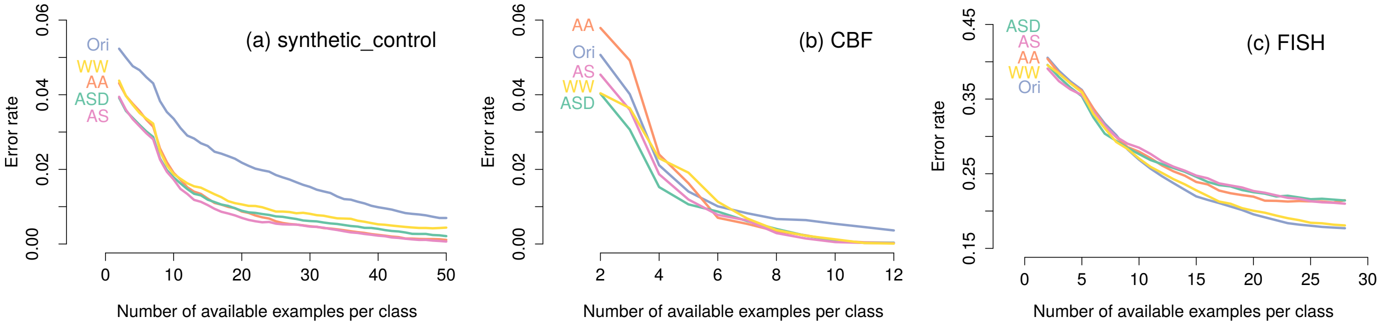


Fig. 3: Evolution of the error-rate on (a) `synthetic_control`, (b) `CBF`, and (c) `FISH` datasets for our three methods (AA, AS, ASD), Window warping (WW) and Original (Ori) with increasing number of available examples.

recommendations given in [11] and use warping ratios equal to either $\frac{1}{2}$ or 2 on slices representing 10% of time series's length. The methods AA (Average All), AS (Average Selected) and ASD (Average Selected with Distance) correspond to the methods proposed in the previous sections.

Then, for every dataset, we rank the methods by error-rates: rank 1 is assigned to the method with the lowest error; rank 5 is assigned to the method with the highest[1]. We then compute the average rank for every method. Let r_i^j be the rank of the j^{th} of A algorithms on the i^{th} of N_d datasets. The average rank for algorithm j is computed as $R_j = \frac{1}{N_d} r_j^i$. This gives a direct general assessment of all the algorithms: the lowest rank corresponds to the method that, on average, obtains the best error-rate with the available examples.

Table II shows the average rank of all algorithms over the datasets of [13]. These results demonstrate that ASD method outperforms other methods. It is interesting to note that generating synthetic data is always more effective than keeping only the original examples (i.e. the average ranking of the Original method is always the highest). It means that if only few examples are available, it is beneficial to generate synthetic examples.

In addition, we test the statistical significance of these results. We want to assess whether 85 datasets is a large enough sample to state that this difference in the ranking is statistically significant. We first perform a Friedman test [19], in order to assess whether the results are significantly different. This test is used to evaluate whether there is enough evidence to confidently state that the different methods are not performing equally.

$$\chi_F^2 = \frac{12N_d}{A(A+1)} \left[\sum_j R_j^2 - \frac{A(A+1)^2}{4} \right] \qquad (4)$$

The values are reported in the χ_F^2 line of Table II. Given that the Friedman test follows a χ^2 distribution with $A-1$ degrees of freedom, these results yield a significant difference among the methods ($p < 10^{-5}$).

Having rejected the null hypothesis, we can proceed with a detailed comparison of the methods. Again, we follow standard practices for classifier comparison [19] and perform a Nemenyi test to compare pairs of methods. Comparing 5 methods over 85 datasets, [19] shows that, to be statistically significant ($\alpha = 0.05$) the difference between the average rankings has to be greater than:

$$CD = q_{0.05} . \sqrt{\frac{A(A+1)}{6N_d}} = 2.728 . \sqrt{\frac{30}{510}} \approx 0.662 \qquad (5)$$

We report the difference between the average rank obtained by the Original method (no augmentation) and the one obtained by ASD method over the 85 datasets in the last line of Table II. It shows that the difference is greater than the critical difference *CD*, regardless of the number of available examples. As a result, we can confidently conclude that ASD is statistically significantly better than Original, and thus that the use of synthetic examples yields better results than using only original examples for the cold start problem.

B. When is it beneficial to add synthetic examples?

The previous experiments were conducted for available examples from 2 to 6 per class. The remaining question is: Is it still useful to generate synthetic examples when the number of available examples is larger?

To address this question, we studied the evolution of the error-rate with increasing number of available examples, until we reached the point of using the entire training set. Figure 3 presents the evolution of the error-rate for three datasets (`synthetic_control`, `CBF` and `FISH`) for our three methods (AA, AS, ASD), Window warping (WW) and Original (Ori), with increasing numbers of available examples.

When the dataset drawn from a simple distribution and the classes are homogeneous, the addition of synthetic examples is clearly beneficial. This observation is true for the `synthetic_control` dataset (Figure 3(a)) where it is always beneficial to add synthetic examples. This dataset was indeed synthetically generated and we can posit that the model is easily inferred by our method to successfully create additional synthetic examples.

[1] In case of ties, we assign the average (or fractional) ranking. For example, if there is one winner, two seconds and a loser [1,2,2,4], then the fractional ranking will be [1,2.5,2.5,4].

For most of the datasets, the differences in error-rate are substantial when the number of available examples is low; for example for the CBF datasets in Figure 3(b). The expected benefit is likely to be correlated with the complexity of the dataset and how well the available example represents the class distribution. Furthermore, the *quality* of the synthetic examples directly depends on the available original examples. For the CBF dataset, the ASD method creates beneficial synthetic examples from the start and it appears that with 6 available examples per class, all the generative methods managed to create good synthetic examples, improving the error-rate compared to using only the original examples. Synthetic examples make it possible to completely eliminate errors in classification.

However, in some cases, for instance the FISH dataset (Figure 3(c)), the addition has very slight benefit when the size of the available sample is small, but at some point, adding synthetic examples actually increases the error-rate (when 10 examples are available per class in Figure 3(c)). In order to evaluate whether adding synthetic examples is harmful, we carried out experiments in which we used the entire training sets that we augmented.

C. Full training set experiment

In this experiment, we report results where we used the entire training set as the starting point, and we doubled the number of time series. We doubled the size of each class regardless of possible unbalancing in class size. We focus on ASD method as it provided the best result according to the experiment of the previous section.

Of the 85 UCR datasets, for 56 the augmented dataset led to higher accuracy, for 7 they were identical and for 22 the accuracy decreased with the addition of the synthetic examples. The average increase of accuracy among the 56 dataset was of 3.81% while the average diminution of the 22 datasets was of -1.72%. For four datasets, the increase in accuracy was higher than 10%, with the highest improvement for the Phoneme dataset (18.42%). We performed a one-sided Wilcoxon signed rank test in order to assess the statistical significance of the error-rate differences. The increase in accuracy attributed to our proposed data augmentation method is statistically significant at the 0.001 level.

IV. Conclusion

In this paper, we introduced a framework for generating synthetic time series under Dynamic Time Warping. To this end, we defined weighted averaging for time series under Dynamic Time Warping (Weighted DBA). Using this foundation, we then created smart methods to choose weights in order to generate useful synthetic time series by weighted averaging. Experiments reveal that our methods are useful both when very few time series are available and even when the full learning set is used.

In future work, we plan to study how our synthetic example generation methods could be helpful in conjunction with other learning methods (e.g. SVM, Deep Networks, etc.). The genericity of our techniques makes it possible to use them in conjunction with any existing learner.

Acknowledgments

This research was supported by the Australian Government through the Australian Research Council's *Discovery Early Career Award* (DE170100037). This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-16-1-4023.

References

[1] D. Brain and G. I. Webb, "On the effect of data set size on bias and variance in classification learning," in *4th Australian Knowledge Acquisition Workshop*, 1999, pp. 117–128.

[2] J. Nonnemaker and H. S. Baird, "Using synthetic data safely in classification," *Proc. SPIE 7247, Document Recognition and Retrieval XVI,*, vol. 7247, 2009.

[3] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, 2012.

[4] N. A. Zaidi, G. I. Webb, M. J. Carman, F. Petitjean, and J. Cerquides, "ALR^n: accelerated higher-order logistic regression," *Machine Learning*, vol. 104, no. 2, pp. 151–194, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[6] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[7] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[8] T. Varga and H. Bunke, "Comparing natural and synthetic training data for off-line cursive handwriting recognition," in *IEEE Workshop on Frontiers in Handwriting Recognition*. IEEE, 2004, pp. 221–225.

[9] N. Macia, E. Bernadó-Mansilla, and A. Orriols-Puig, "Preliminary approach on synthetic data sets generation based on class separability measure," in *IEEE International Conference on Pattern Recognition (ICPR)*. IEEE, 2008, pp. 1–4.

[10] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[11] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data augmentation for time series classification using convolutional neural networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[12] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh, "Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm," *Knowledge and Information Systems*, vol. 47, no. 1, pp. 1–26, 2016.

[13] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

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

[15] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh, "Dynamic time warping averaging of time series allows faster and more accurate classification," in *IEEE International Conference on Data Mining (ICDM)*, 2014, pp. 470–479.

[16] F. Petitjean and P. Gançarski, "Summarizing a Set of Time Series by Averaging: from Steiner Sequence to Compact Multiple Alignment," *Theoretical Computer Science*, vol. 414, no. 1, pp. 76–91, Jan. 2012.

[17] G. Forestier *et al.*, "Supporting website : http://germain-forestier.info/src/icdm2017/," 2017.

[18] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[19] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 1–30, 2006.