Judicious Setting of Dynamic Time Warping's Window Width Allows More Accurate Classification of Time Series

Hoang Anh Dau*, Diego Furtado Silva§, François Petitjean†, Germain Forestierψ, Anthony Bagnall‡, Eamonn Keogh*
* Computer Science and Engineering Department, University of California, Riverside, USA. {hdau001, eamonn}@ucr.edu
§ Department of Computing, Federal University of São Carlos, Brazil. diego@dc.ufscar.br
† Faculty of Information Technology, Monash University, Australia. francois.petitjean@monash.edu
ψ MIPS, University of Haute-Alsace, France. germain.forestier@uha.fr
‡ School of Computing Sciences, University of East Anglia, UK. ajb@uea.ac.uk

***Abstract*— While the Dynamic Time Warping (DTW) - based Nearest-Neighbor Classification algorithm is regarded as a strong baseline for time series classification, in recent years there has been a plethora of algorithms that have claimed to be able to improve upon its accuracy in the general case. Many of these proposed ideas sacrifice the simplicity of implementation that DTW-based classifiers offer for rather modest gains. Nevertheless, there are clearly times when even a small improvement could make a large difference in an important medical or financial domain. In this work, we make an unexpected claim; an underappreciated "low hanging fruit" in optimizing DTW's performance can produce improvements that make it an even stronger baseline, closing most or all the improvement gap of the more sophisticated methods. We show that the method currently used to learn DTW's only parameter, the maximum amount of warping allowed, is likely to give the wrong answer for small training sets. We introduce a simple method to mitigate the small training set issue by creating synthetic exemplars to help learn the parameter. We evaluate our ideas on the UCR Time Series Archive and a case study in fall classification, and demonstrate that our algorithm produces significant improvement in classification accuracy.**

Keywords-time series; Dynamic Time Warping; classification

I. INTRODUCTION

There is a growing consensus that the Dynamic Time Warping (DTW) - based *k*-Nearest-Neighbor (*k*-NN) Classification algorithm (NN-DTW) is a strong baseline for time series classification. This agreement stems from the fact that time series classification has a universally used collection of benchmark datasets [5]. There are now many independent comprehensive empirical studies demonstrating strong performance of NN-DTW [2][8][15].

Beyond the *distance measure* used, the accuracy of time series classification mostly depends on the size of the training set. While we are now in the Big Data era and have an ever growing need to classify massive datasets [3][18], the size of *training data* has not increased in decades [19], and is unlikely to do so [14]. For example, getting a million images of cats is trivial, but obtaining just a few dozen time series of different types of falls/trips/stumbles requires days of work [1].

Given that fact, we wish to squeeze the most out of DTW's performance. In this work, we make the following claim, an underappreciated "low hanging fruit" in optimizing DTW's performance can produce improvements that make it a much stronger baseline, closing most of the improvement gap of the more sophisticated methods. This claim is somewhat surprising, given the greatly "picked-over" status of time series classification research [8]. We show that the method currently used to learn DTW's only parameter, the maximum amount of warping allowed (denoted *w*), is very likely to return a poor parameter setting for small training sets, reflecting in suboptimal classification performance.

We can visually preview our results as follows. The method used to learn *w* in [5] (and thus reflected in [7][10]) is using the Leave-One-Out (LOO) cross-validation to test the error-rate for all values of *w*, choosing the one that minimizes the predicted error-rate and breaking ties by choosing the smaller value. For concreteness, we call this the *UCR-method*. Figure 1 shows that on many datasets, this simple method works well. In these three examples, the *UCR-method* predicted the correct optimal value of *w* for *CinC_ECG*, and was only off slightly for *CBF* and *50words.*

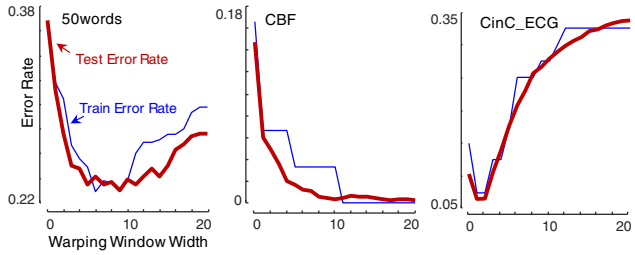

Figure 1. blue/fine: The LOO error-rate of three datasets for increasing values of *w*. red/bold: The holdout error-rate. In these cases, the holdout accuracies closely track the predicted accuracies.

Note that predicted accuracy can be a slightly optimistic estimation of holdout error. However, that is not the issue here. We are only concerned with whether we are minimizing this error by picking a suitable value for *w*. Contrast the results above with the examples shown in Figure 2.

In these cases, our estimation of *w* is worse, and this has a detrimental effect on our holdout error. For example, on *DiatomSizeReduction,* we predicted $w = 0$ would be an appropriate setting, but only an oracle would have chosen $w = 13$ and seen a 3.27% reduction in error-rate. Likewise, we would predict that $w = 0$ is the ideal setting for *Gun_Point*, but a $w = 2$ would have reduced the misclassifications by 6%.

The importance of a better estimate of *w* is difficult to overstate. In dozens of cases, it would do more than closing

the improvement "gap" of recently proposed time series classification algorithms [7][10][12][13]. For example, [7] proposes a time series forest ensemble method. One of their reported successes is in halving the error-rate on *Gun_Point* to 4.7%. However, Figure 2.*right* shows that simply finding a better choice of *w* for 1-NN-DTW could further *halve* their reported error-rate to just 2.7%.

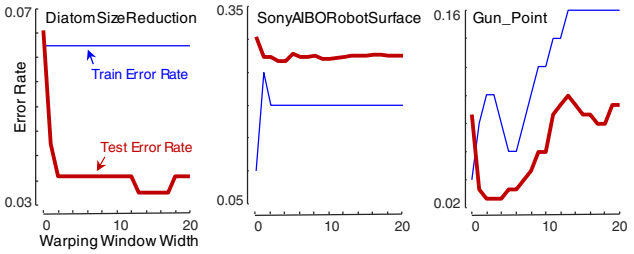

Figure 2. blue/fine: The LOO error-rate of three datasets for increasing values of *w*. red/bold: The holdout error-rate. In these examples, the holdout accuracies do *not* track the predicted accuracies.

Similarly, [10] introduces a new distance measure DD_{DTW} that combines the DTW distances calculated both on the raw data and its derivatives (i.e., the mixture weights being learned by cross-validation). Among the datasets that are considered are *Lightning2* and *Lightning7*. The authors note that they can reduce the error-rate of *Lightning2* to 13.11%, but a better choice of *w* for 1-NN-DTW could significantly improve upon this with just an 8.2% error-rate.

It is important to clarify that we are not claiming the works mentioned above are without merit. Any improvement in setting *w* might help all of them, especially [12] and [13]. However, in most cases, the community is proposing rather complex methods for relatively modest gains in accuracy. The results in Figure 2 suggest that similar or greater improvements are possible with existing techniques, if we just had a better method to discover a suitable value of *w*. There are also strong reasons to prefer existing techniques, as they are amenable to many optimizations that allow them to scale to trillions of data points or to real-time deployment on resource-constrained devices [18].

In this work, we show that it is possible to learn *w* more accurately; this is particularly useful when the training data is limited. Our approach is based on resampling the training data. Resampling is normally ill-advised in small datasets, since using only a subset of the data compounds all the inherent problems encountered while working with the limited data. However, we will show that we can address this issue by replacing the non-sampled data with synthetic replacements.

II. Background and Related Work

A. Dynamic Time Warping

While many other time series distance measures have been proposed in recent years (see [2][8] and the references therein), the community has come to the consensus that DTW (including its special case of Euclidean distance) is one of the best distance measures for most data mining problems [6][8][10][13][17]. In [18], the authors state: "*after an exhaustive literature search of more than 800 papers, we are not aware of any distance measure that has been shown to outperform DTW by a statistically significant amount.*"

As illustrated in Figure 3.*left*, DTW allows a one-to-many mapping between data points, thus enabling a meaningful comparison between two time series that have similar patterns but are locally out of phase, or "warped." We call an alignment between them a warping path. Among all the possible warping paths, we choose the path that minimizes the differences between two time series.

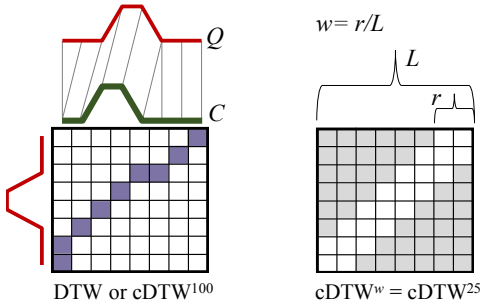

Figure 3. *left*: The unconstrained warping path for time series *Q* and *C*. Such warping paths can pass through any cell of the matrix. *right*: The warping path is restricted to not pass through cells that are far from the diagonal.

A constrained DTW imposes a limit on how much the warping path can deviate from the diagonal. This limit is known as the warping window width (*w*). For example, in Figure 3.right, the warping path cannot traverse the gray cells.

The constrained DTW helps avoid pathological mappings, in which one point in the first time series is mapped to too many points in the other one. For example, DTW should be able to align American and Australian utterances of "Minutiae" (*Min-OOSH-a*, and, *min-OOSH-ee-AY* respectively), but it would never make sense to map an utterance of "Minutiae" to "Galton constructed a statistical proof of the uniqueness, by minutiae, of individual prints."

In addition, the constraints also reduce the computation cost by narrowing the search for qualified paths. The fastest known techniques for indexing DTW further exploit these constraints to produce tight lower bounds [18]. The most commonly used constraint is the Sakoe-Chiba Band, which expresses *w* as a percentage of the time series length. We denote DTW with a constraint of *w* as $cDTW^w$. This review is unavoidably brief; we refer the interested reader to [8][20] and the references therein for more details.

B. DTW-based 1-NN Classification Algorithm

The nearest neighbor classifier (NN) assigns an unseen object to the class of its closest neighbor in the feature space. The general algorithm is referred to as *k*-NN, in which *k* is the number of nearest neighbors under consideration. In case of 1-NN, the new object is automatically assigned the class label of its nearest neighbor, breaking ties randomly.

Most practitioners who adopt 1-NN do so for its simplicity, i.e., requiring no parameter tuning. The research focus has thus shifted to improving the distance measure used.

1-NN using DTW has emerged as the new benchmark for many time series classification tasks. This practice of using 1-NN-DTW is supported by a recent survey in time series classification: "*When using a NN classifier with DTW on a new problem, we would advise that it is not particularly important to set k through cross validation, but that setting the warping window size is worthwhile*" [2].

C. Classic Learning of Warping Window Size

The most cited method for learning the maximum warping window width for DTW is the *UCR-method* [5]. The best value of *w* is determined by performing leave-one-out cross-validation on the training set over all warping window constraints possible, from 0% to 100% at 1% increments. The window size that maximizes training accuracy is selected, as it is deemed most likely to give the best testing accuracy. The creators of the UCR Time Series Archive's disclaimer states that this may not be the best way to learn *w*, but it is simple, parameter-free, and works reasonably well in practice. We estimate that at least four hundred papers have used this approach [3][8][12] [19], by either explicitly implementing it, or directly comparing results to the numbers published in the UCR Archive, which were computed this way [5].

D. Related Work

The closest work we are aware of is a recent paper in which the authors propose a method to learn *w* for DTW in the context of time series *clustering* [6]. At first glance, it is tempting to apply this algorithm to our closely related problem of time series classification. However, as the authors have noted in this paper, the best setting of *w* for clustering is generally no indicator of the best setting of *w* for classification: "*In retrospect it is not surprising that these values are at best weakly related. For 1-NN classification, only the distance between the unlabeled exemplar and its single nearest neighbor matters. However, for clustering, the mutual distance among small groups of objects matters.*"

The more general idea of creating synthetic data to mitigate the problems of imbalanced datasets [4] or to learn a distance measure [11] is known. However, we are not aware of any other research suggesting window size for improving DTW-based classification. We suspect that the dearth of study on this important problem is likely due to the community's lack of appreciation of the importance of *w* setting.

III. Our Approach

In this section, we begin by developing the reader's appreciation for the factors that affect the best setting for DTW's warping window width. Then we give an intuitive explanation to our approach before formalizing the algorithm.

A. Factors Affecting the Best Warping Window

To understand the effect(s) of dataset size on the most suitable warping window width, we performed the following experiment. We created a two-class dataset that we call Single Plateau (SP). This dataset and all other datasets used in this paper are available in [22]. Each item in SP is 500-points long and consists mostly of a constant value with a small amount of random noise added. However, as we show in Figure 4, we add a tall "plateau" with a length randomly chosen in the range of five to twenty to each exemplar. If the plateau's location is within 1 to 250, it is in class A; if it is between 300 and 500, it is in class B.

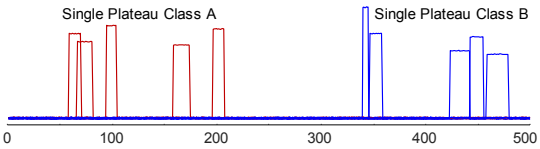

Figure 4. Five examples of each class of Single Plateau dataset.

We classify increasingly large instances of SP. For each size, we search over all possible *w*'s and record the one that minimizes the error-rate. Figure 5 shows the optimal *w* curve vs. dataset size, averaged over 100 runs.

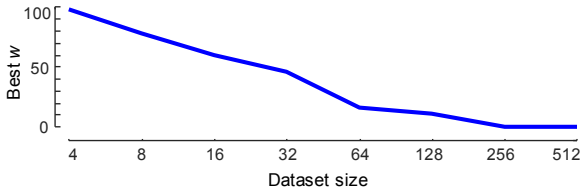

Figure 5. Classification of increasingly large instances of SP shows the effect of dataset size on the best *w*.

Consistent with observations by Ratanamahatana and Keogh [19], small datasets tend to require much larger settings of *w* compared to larger ones. Note that this size versus the best curve for *w* itself varies for each dataset. Thus, we cannot generalize the best setting for *w* on one subset of a dataset to a different sized subset of the same datasets.

As shown in Figure 5, the best value for *w* on this dataset, given that we have 32 objects, is 46. Let us further consider this particular sized subset of the training set. Figure 6 displays the effect of *w* on the misclassification rate of the 32-object SP dataset. We can see that allowing too much warping is as detrimental as too little warping. In this case, the *w* vs. error-rate curve has a broad flat valley, meaning that even if we choose a *w* value that is too large or small, we could still achieve a low misclassification. However, as Figure 6 subtlety shows, this curve can take on more complex shapes, which makes the choice of *w* more critical.

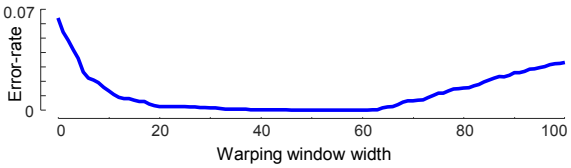

Figure 6. Classification of the 32-object Single Plateau demonstrates effects of *w* on the LOO error-rate.

In summary, the best value of *w* depends on both the data size and the structure of the data. This fact bodes ill for any attempt to learn a fixed one-time domain independent value for it. For example, there is *no* single prototypical *w* vs. error-

rate curve for heartbeats or for gestures. We must learn this curve on a case-by-case basis, which is the goal of this paper.

B. An Intuition to Our Proposed Approach

To understand the effect(s) of dataset size on the most suitable warping window width, we begin with a simple experiment, which acts as both a sanity check to see if what we hope to achieve is possible and offers intuition on how to achieve it. Consider the *TwoPatterns* dataset. Because it has 1000 training objects, we will denote it as *TwoPatterns*^{1000}. As shown in Figure 7.*left*, *TwoPatterns*^{1000} is a dataset for which we can correctly learn the best maximum warping window with cross-validation.

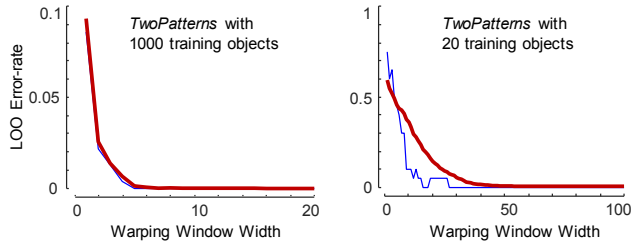

Figure 7. *left*: The LOO error-rate of the *TwoPatterns*^{1000} dataset for increasing values of *w*. *right*: The LOO error-rate of the *TwoPatterns*^{20} dataset for increasing values of *w* (blue/fine) and the holdout error-rate (red/bold).

Suppose the dataset had significantly fewer training instances, and we will call this dataset *TwoPatterns*^{20}. We would expect that the holdout error-rate would increase, and we know from [19] that we should expect the best value for *w* might slightly increase. As we can see in Figure 7.*right*, these both do occur. However, the most visually jarring thing we observe is that we have lost the ability to correctly predict the best value for *w*, as the training error oscillates wildly as we vary *w*. In fact, Figure 7.*right* strongly resembles some of the plots shown in Figure 2, and for the same reason that we do not have enough training data.

Let us further suppose that while we are condemned to using *Two_Patterns*^{20} to classify new instances, we have one thousand more labeled instances at our disposal. One might ask, if we have more labeled examples, why not use them in the training set? Perhaps the time available at classification time is only enough to compare to twenty instances. Clearly, we do not want to use all one thousand labeled instances to learn the best value for *w*, because, as shown in Figure 8.*left*, we will learn the best value of *w* for *TwoPatterns*^{1000}, not for *TwoPatterns*^{20}, which is our interest.

The solution suggests itself. Doing cross-validation with *TwoPatterns*^{1000} gives us low variance, but it is biased toward the wrong value of *w*. In contrast, doing cross-validation with *TwoPatterns*^{20} is biased toward the correct value for *w* but has high variance. If we resample many subsets of size twenty from *TwoPatterns*^{1000}, do cross-validation on each, and average the resulting *w* vs. error-rate curves, we expect that this average mirrors the curve for the test error-rate, and therefore predicts a good value for *w*. As we can see in Figure 8.*right*, this is the case.

Figure 8. *left*: The LOO error-rate of the *TwoPatterns*^{1000} dataset is a poor predictor of the holdout error on *TwoPatterns*^{20} *right*: In contrast, the *average* LOO error-rate of 20 random samples of *TwoPatterns*^{20} dataset is an excellent predictor of the holdout error on *TwoPatterns*^{20}.

The observations above seem to be non-actionable. In general, we do not have 1,000 spare objects to resample from. Our key insight is that we can *synthetically* generate plausible training exemplars. We can use these synthetic objects to resample from, make as many new instances of the training set as we wish, and learn the best setting for *w*.

Note that this task is easier than it seems. We do not need to produce synthetic exemplars that are perfect in every way or even visually resemble the true objects to the human eye. It is sufficient to create synthetic objects that have the same properties with regards to the best setting for *w*. In the next section, we show our strategy for generating an arbitrary number of such instances.

C. Our Algorithm

We are finally in a good position to explain our algorithm, which can be tersely summarized as follows:

> Make N copies of the original training set. For each copy, replace a fraction of the data with synthetically generated data, and use cross-validation to learn the error vs. *w* curve. Use the average of all N curves to predict *w*.

TABLE 1. ALGORITHM FOR MAKING AUGMENTED TRAINING SET

```
   Input:  D, the original training set with n objects
   Input:  M, the amount of warping to add
   Input:  R, the ratio of synthetic objects to create
   Output: Dnew, a new version of dataset D
1  realObjects ← random_sample(D,(1-R)*n objects)
2  fakeObjects ← random_sample(D,(R*n) objects)
3  for i ← 1:1:numberOfInstances(fakeObjects)
4   fakeObjects_i ← add_warping(fakeObjects_i,M)
5  end
6  Dnew ← [realObjects;fakeObjects]
```

This algorithm, outlined in Table 2, contains a subroutine presented in Table 1. The essence of the method is in making N new training data sets using the algorithm in Table 1. These datasets will be used in lieu of the original training set to learn *w*. While each of these individual datasets may produce a noisy error vs. *w* curve (as in Figure 7.*right*), the average of all such curves will be smoother, and it will more closely resemble the true noisy error vs. *w* curve (as in Figure 8.*right*).

TABLE 2. ALGORITHM FOR FINDING THE WARPING WINDOW WIDTH

```
   Input:  D, the original training set
   Output: w, the predicted best warping window
1  for i ← 1:1:numberOfIterations
2   Dnew ← make_new_train_set(D)          // See Table 1
3   for j ← 1:1:maximumWarpingWindow
4    errorRate[i,j] ← run_cross_validation(Dnew)
5   end
6  end
7  meanOfAllIterations ← mean(errorRate)
8  [minValue, minIndex] ← min(meanOfAllIterations)
9  w ← minIndex - 1
```

The sub-routine of making new training set is invoked over a number of iterations, as shown in line 2 of the main algorithm in Table 2. For each new training set, we run cross-validation to compute the classification error-rate at each setting of the maximum warping width allowed, from 0% (Euclidean distance) to 100% (unconstrained DTW), in steps of 1%. Finally, we calculate the mean error-rate of all runs in line 7 and obtain the index of the minimum error.

D. Add Warping to Make New Time Series

To add warping to a time series, we nonlinearly shrink it to a smaller length by randomly removing data points and then linearly stretching the down-sampled time series back to its original size. Figure 9 illustrates how a time series is transformed into its warped version. For concreteness, Table 3 contains the MATLAB code that we used.

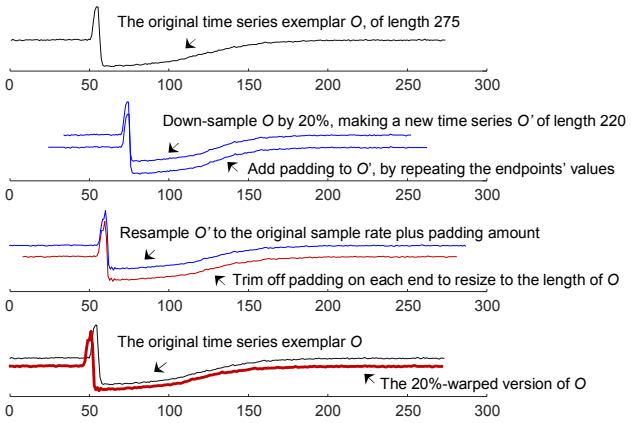

Figure 9. Adding 20% warping to an exemplar of *Trace*. In the bottom panel, the generated times series (bold/red) is a slightly warped version of the original time series.

To eliminate the possible "endpoint effects" introduced by the resampling process, we add "paddings" at the beginning and end of the down-sampled time series by repeating its endpoint/start point values ten times. These paddings are removed from the final time series later (Figure 9. *middle*). It is important, as a recent work indicates, the endpoints can result in a misleading DTW distance [21]. Recall that DTW's constraints require it to match the pairs of beginning and end points, even though they may be a poor match.

It may be possible to further improve our overall method if we find better ways to make more "natural" synthetic exemplars. We experimented with several methods to generate a synthetic time series [9][16]. In brief, there are *dozens* of ways to produce synthetic examples (averaging, grafting, perturbing etc.), and many of these ideas work well. We chose the method shown in Table 3, because it is simple and effective.

TABLE 3. CODE TO ADD WARPING TO A TIME SERIES

```
1  function [warped_T] = add_warping(T,p)
2   i = randperm(length(T));
3   t = T(sort(i(1:end-length(T) * p)));
4   t = [repmat(t(1),1,10), t, repmat(t(end),1,10)];
5   warped_T = resample(t,length(T) + 20, length(t));
6   warped_T = warped_T(11:end - 10);
7  end
```

IV. EMPIRICAL EVALUATION

All code and datasets used in the paper are archived in perpetuity at [22]. We also include a technical report, which provide more details for interested readers.

A. Datasets

We use the UCR Time Series datasets for our experiments [5]. As of May 2017, the UCR Time Series Archive, which has 85 datasets from various domains, has served as the benchmark for the time series community.

As we have demonstrated in Figure 7.*left*, our ability to learn *w* greatly depends on the amount of training data. With enough data, the simple *UCR-method* is effective, and we have little to offer [5]. The ideas proposed in this work are most useful for smaller datasets. Some of the train/test splits in the UCR datasets have large enough training sets that our ideas do not offer any advantages. Rather than ignoring these datasets, we will recast them to a smaller uniform size.

We merge the original train and test set together, then randomly sample ten objects per class for training. The remaining objects are used for testing. As three datasets do not have enough ten objects per class, we excluded them from the experiment (the excluded are: *OliveOil*, *50words* and *Phoneme*). Therefore, we are left with 82 datasets. These new splits are published in [22] for reproducibility. Note that with these new splits, the training sets all have equal class distribution. This, however, may not be true for the test set.

B. Performance Evaluation

Using the algorithm in Table 2 to learn the warping window size, we classified the holdout test data on the training set with 1-NN. Figure 10 and Figure 11 show a visual summary of the results. Perceptibly, our method wins more often and by larger margins.

We call our proposed method a *success* if it can reduce the error-rate, in absolute value, by at least 0.5% (i.e., we round the error-rate to two decimal places). We call it a *failure* if our method increases the error-rate by 0.5% or more compared to the *UCR-method*. If the newly learned *w* results in test error-rate that is less than 1% different from the test error-rate obtained by the traditional method, we consider our method *neutral*. This can happen in two ways. Our method suggests the same value of *w* as the classic *UCR-method*, or it

recommends a different value of *w*, which offers similar accuracy.

Given this nomenclature, we can say that out of the 82 datasets tested, our method improves the classification accuracy of twenty-four, with an average improvement of 3.2%, and it decreases the accuracy on just thirteen of them with a much smaller average of 1.6%. This statement can in turn be visualized with the linear plot in Figure 10.

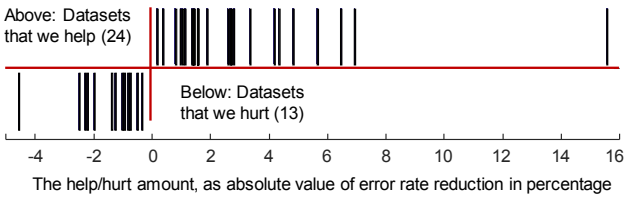

Figure 10. The number of datasets that we *help* are nearly twice the number of datasets that we *hurt*.

Another way to demonstrate how our proposed method outperforms the traditional method is to look at the possible room for improvement, which is the difference between the error-rate achieved by the learned *w* and the error-rate by the best *w* of a dataset, which was found by exhaustive search. The smaller the difference, the better the method is. This is illustrated in Figure 11.

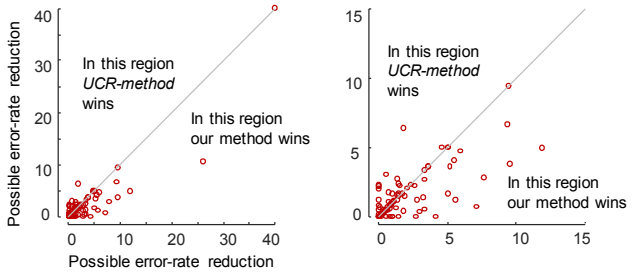

Figure 11. Possible error-rate reduction of the *UCR-method* and our method (how close a method's error-rate to the optimal error-rate is).

While the results are visually compelling, we turn to statistical tests to ensure that they are statistically significant. Both the paired-sample *t*-test and the one-sided Wilcoxon signed rank test confirm that our method is better than the *UCR-method* at the 5% significance level [22].

V. CONCLUSIONS

We have demonstrated that the choice of warping window width *w* is critical for an accurate DTW-based classification of time series. In many cases, a more careful setting of the value of *w* can close the performance gap gained by other more complicated algorithms recently proposed in the literature.

Our method is parameter-free (or equivalently, we hardcoded all parameters). However, experimenting with adaptive parameters may allow others to improve upon our results. We note that the ideas we are proposing are *very simple*. We hope that the reader sees this simplicity as the strength it is intended to be, not as a weakness.

ACKNOWLEDGMENT

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. We acknowledge funding from NSF IIS-1161997 II and NSF IIS-1510741 and ARC DE170100037.

REFERENCES

[1] Albert, Mark V., et al. "Fall classification by machine learning using mobile phones." *PloS one* 7.5 (2012): e36556.

[2] Bagnall, Anthony, and Jason Lines. "An experimental evaluation of nearest neighbour time series classification." In *arXiv preprint arXiv:1406.4757* (2014).

[3] Bagnall, Anthony, et al. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances." *Data Mining and Knowledge Discovery* (2016): 1-55.

[4] Batista, Gustavo EAPA, et al. "A study of the behavior of several methods for balancing machine learning training data." *ACM Sigkdd Explorations Newsletter* 6.1 (2004): 20-29.

[5] Chen, Yanping, et al. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/, 2015.

[6] Dau, Hoang Anh, et al. "Semi-Supervision Dramatically Improves Time Series Clustering under Dynamic Time Warping." *Proceedings of International on Conference on Information and Knowledge Management*. ACM, 2016.

[7] Deng, Houtao, et al. "A Time Series Forest for Classification and Feature Extraction." *Information Sciences* 239 (2013): 142-153.

[8] Ding, Hui, et al. "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures." *Proceedings of the VLDB Endowment* 1, no. 2 (2008): 1542-1552.

[9] Forestier, Germain, et al. "Generating Synthetic Time Series to Augment Sparse Datasets." *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016.

[10] Górecki, Tomasz, and Maciej Łuczak. "Using Derivatives in Time Series Classification." *Data Mining and Knowledge Discovery* 26, no. 2 (2013): 310-331.

[11] Ha, Thien M., and Horst Bunke. "Off-line Handwritten Numeral Recognition by Perturbation Method." *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 19(5), pp. 535–539, May 1997.

[12] Jeong, Young-Seon, et al. "Weighted dynamic time warping for time series classification." *Pattern Recognition* 44.9 (2011): 2231-2240.

[13] Kate, Rohit J. "Using Dynamic Time Warping Distances as Features for Improved Time Series Classification." *Data Mining and Knowledge Discovery* (2015): 1-30.

[14] Lam, Xuan Nhat, et al. "Addressing cold-start problem in recommendation systems." *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. ACM, 2008.

[15] Lines, Jason, and Anthony Bagnall. "Time series classification with ensembles of elastic distance measures." *Data Mining and Knowledge Discovery* 29.3 (2015): 565-592.

[16] Petitjean, François, et al. "Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification." *2014 IEEE International Conference on Data Mining*. IEEE, 2014.

[17] Plouffe, Guillaume, and Ana-Maria Cretu. "Static and dynamic hand gesture recognition in depth data using dynamic time warping." *IEEE Transactions on Instrumentation and Measurement* 65.2 (2016): 305-316.

[18] Rakthanmanon, Thanawin, et al. "Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping." *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012.

[19] Ratanamahatana, Chotirat Ann and Eamonn Keogh. "Three Myths about Dynamic Time Warping Data Mining." *Proceedings of the SIAM International Conference on Data Mining*, 2005.

[20] Shokoohi-Yekta, Mohammad, et al. "On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case." *Proceedings of the SIAM International Conference on Data Mining,* 2015.

[21] Silva, Diego Furtado, Gustavo EAPA Batista, and Eamonn Keogh. "Prefix and Suffix Invariant Dynamic Time Warping." *2016 IEEE International Conference on Data Mining*. IEEE, 2016.

[22] Supporting webpage: https://sites.google.com/site/dtwclassification/