2010 IEEE International Conference on Data Mining Workshops

Using Adaptive Downsampling to Compare Time Series with Warping

Chih-Chun Chia
University of Michigan
Ann Arbor, MI 48109 USA

Zeeshan Syed
University of Michigan
Ann Arbor, MI 48109 USA

***Abstract*—Dynamic time warping (DTW) is widely used in many practical domains to compare time series with warping (i.e., signals where similar activity may not be perfectly aligned by sample number). In this paper, we explore different ways to improve upon the efficiency and accuracy of the basic DTW algorithm. Central to our work is the concept of adaptive downsampling using trace segmentation. We describe how the size of the data being compared can be reduced substantially with limited loss of information by downsampling slowly changing parts of the signals much more than rapidly changing regions. We propose two novel measures based on the notion of adaptive downsampling: trace profile comparison (TPC), which compares the reduced representations obtained by adaptive downsampling using a weighting scheme that assesses the relative importance of changes in amplitude and timing, and piecewise linear DTW (PLDTW), which compares piecewise linear segments of the reduced signals using a modified dynamic programming algorithm and cost function. When evaluated on the UCR Time Series dataset, TPC provided an improvement in runtime over the basic DTW algorithm and recent optimizations to it, while PLDTW improved the accuracy of DTW for classification in different datasets.**

***Keywords*-time series; indexing; classification**

I. Introduction

Time series data are generated in almost every application domain, e.g., financial, geodysical, clinical, and scientific. The problem of querying and mining time series data has therefore attracted much attention recently. While different methods have been proposed to compare time series, an important consideration in many practical applications is quantifying similarity in the presence of time warping, i.e., dilation, shrinking and translation in time series such that similar activity in two signals may not be perfectly aligned by sample number or relative to a fiducial point. This requirement makes the Euclidean distance (ED) metric inappropriate for many applications. Instead, dynamic time warping (DTW) is widely used in these cases [3], [20].

The basic DTW algorithm quantifies the similarity between two time series with warping by setting up an alignment or correspondence between matching parts of the signals. This process is quadratic in both runtime and space, and does not scale to long time series or large volumes of data. There is a significant body of work focusing on addressing this limitation. Most of the previous work in this area has approached the goal of increasing the efficiency of DTW either by introducing constraints [21], or through dimensionality reduction methods such as the discrete fourier transform [1], singular value decomposition [17], and downsampling by a constant factor [13], [16].

In this paper, we improve upon the basic DTW algorithm using the concept of adaptive downsampling. We build upon the idea of improving the efficiency of time series comparison by reducing the size of the data being compared. However, in contrast to earlier work focused on reducing signals by a constant factor over time, we explore a trace segmentation-based approach that reduces time series more in regions where the signals are slowly changing relative to parts of the signal associated with rapid changes. In this way, our method accomplishes a reduction in the size of the data while preserving information that would otherwise be lost when downsampling by a constant factor. We note that aspects of this approach resemble earlier work on adaptive dimensionality reduction for indexing using a Euclidean distance metric [14]. We supplement this research by proposing the use of adaptive downsampling to compare signals with warping in a potentially more accurate and efficient manner than DTW. To address this goal of comparing time series with warping, we present solutions to the different problems that arise while comparing adaptively reduced signals.

We propose two approaches that exploit this idea of adaptive downsampling to improve upon the basic DTW algorithm. Our first method, trace profile comparison (TPC), compares the reduced representations obtained by adaptive downsampling using a weighting scheme that assesses the relative importance of changes in amplitude and timing across time series. The TPC measure can be measured in linear time and is intended to provide an improvement over the *runtime* of DTW while achieving a similar accuracy as DTW in indexing or classification problems. Our second method, piecewise linear DTW (PLDTW) compares piecewise linear segments in the reduced representations using an approach similar to DTW. In this case, the goal of PLDTW is to provide an improvement over the *accuracy* of DTW without a substantial increase in runtime. The use of adaptive downsampling in this case is intended to solve the more computationally intensive problem of comparing line segments (rather than time samples) in an efficient manner so that accuracy can be improved while remaining close to the runtime of the basic DTW algorithm. When evaluated on the UCR Time Series dataset, containing time series from

978-0-7695-4257-7/10 $26.00 © 2010 IEEE
DOI 10.1109/ICDMW.2010.94

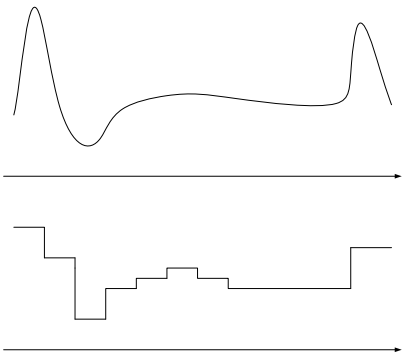

Figure 1. Downsampling time series uniformly can lead to important information being lost. For example, given the RR interval (top), the application of PAA loses the low amplitude region at the start of the signal (bottom) [24].

a variety of application domains, both TPC and PLDTW provide improvements over DTW and optimizations that have been proposed recently to increase efficiency.

The main contributions of this paper are: (1) we present the idea of adaptive downsampling to reduce time series while retaining information in rapidly changing parts of the signals; (2) we describe a trace segmentation-based approach to adaptively downsample signals; (3) we propose two measures to improve upon the accuracy and runtime of DTW; and (4) we supplement the previous comparison of different time series similarity measures on the UCR Time Series dataset [11] with the additional comparison of PAA and FastDTW, as well as our new measures.

II. Overview of Approach

In this section, we review some of the high level ideas underlying our work. We start with a discussion of adaptive downsampling, and then focus on two approaches to use adaptively downsampled time series to improve upon the efficiency and accuracy of DTW.

A. Adaptive Downsampling

Our work builds upon the use of downsampling to improve the efficiency of the basic DTW algorithm. We observe that existing methods such as PAA and FastDTW downsample signals uniformly, i.e., by a constant factor across time. In the case of PAA, this corresponds to a fixed width of each of theframes. For FastDTW, the signal is reduced into equal sized time windows at each iteration. We note that while the use of downsampling improves the runtime and space efficiency of DTW, the decision to carry out this downsampling by a constant factor over time causes both rapidly and slowly changing parts of a signal to be treated similarly. Downsampling in this case can be associated with significant loss of information (Figure 1).

We believe that this process can be improved by exploiting slowly changing parts of a signal by downsampling them at a higher rate than rapidly changing regions. In contrast to PAA and FastDTW, we therefore propose the idea of adaptive downsampling where the rate of reduction of time series varies according to the rate of changes taking place locally. This allows for the reduction of time series, while retaining sharp changes that would otherwise be smeared if downsampling were applied uniformly to the entire signal.

We achieve adaptive downsampling using trace segmentation [18]. While we describe this approach in more detail subsequently, the basic idea underlying trace segmentation is to divide the signal into regions with equal cumulative derivative activity. This approach can be used to adaptively downsample signals because it places boundaries with a greater density in regions that are changing rapidly (i.e., have higher cumulative derivative activity).

B. Improving Efficiency over DTW

Our first measure for time series comparison uses adaptive downsampling to improve efficiency over DTW. We use the trace segmentation boundaries across two signals as a way of relating similar activity. In this way, adaptive downsampling serves as an efficient linear time alignment of the signals. While this approach relates potentially corresponding parts of two time series, it does not provide a direct approach to translate that alignment into a distance. We therefore supplement the adaptive downsampling with a function learned during training that relates the amplitude and timing of differences between the adaptive downsampled representations to produce an overall measure of similarity. The resulting TPC measure is intended to provide an improvement in efficiency over DTW while achieving comparable accuracy.

C. Improving Accuracy over DTW

Our second measure for time series comparison uses adaptive downsampling to create piecewise linear representations of signals. Our hypothesis is that comparing time series in terms of their interpolated morphology may produce more meaningful comparisons than comparing these time series in terms of samples. In particular, comparing time series in terms of samples does not directly factor in trends that may take place in the signal between the sample times. We therefore use the trace segmentation boundaries for a time series to identify parts of the signal that can be connected with a straight line without significant loss of information. These line segments are then compared in a DTW-like manner across signals to obtain an alignment-based measure of similarity. We note that comparing line segments is more computationally intensive than comparing individual time samples, but the overall algorithm is made efficient by the reduction of a potentially large number of samples into a small number of line segments. We also observe that relating line segments in a DTW-like manner is non-trivial since the line segments are not of uniform width (Figure 2). As a result, the basic DTW dynamic programming algorithm

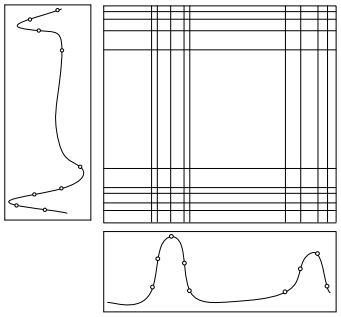

Figure 2. Comparing piecewise linear representations of adaptively downsampled signals. The dynamic programming algorithm of basic DTW needs to be evolved to operate robustly on variably sized segments.

and cost function need to be modified to factor in the variable width of line segments being aligned. We describe the process through which PLDTW is computed in more detail subsequently.

III. Methods

This section presents details of our adaptive downsampling approach using trace segmentation, and the process through which both the TPC and PLDTW measures are computed.

A. Adaptive Downsampling

Given a signal $Q = q_1, \ldots, q_n$ and a number of frames θ to downsample this signal to, we first calculate the cumulative difference $D_Q[k]$ for $k = 2, \ldots, n$ between each neighboring pair of samples:

$$D_Q[k] = \sum_{i=2}^{k} \mid q_i - q_{i-1} \mid \tag{1}$$

with $D_Q[1] = 0$. The sum of the total differences in Q is given by $D_Q[n]$. The cumulative difference in each adaptive downsampling bin is then set to $d_Q = \frac{D_Q[n]}{\theta}$. Using this, downsampling proceeds by finding the sample numbers t_i for $i = 0, \ldots, \theta$ such that for all values of i we have:

$$t_i = \min\{k \mid D_Q[k] \geq d_Q i\} \tag{2}$$

The corresponding amplitudes of Q at samples t_i are given by $x_i = q_{t_i}$. We can then use interpolation to approximate the fractional sample numbers $\hat{t}_i$ where we would expect $D_Q[\hat{t}_i] = d_Q i$. For $i = 0, \ldots, \theta$ using the notation:

$$\beta_i = \frac{D_Q[t_i] - d_Q i}{D_Q[t_i] - D_Q[t_i - 1]} \tag{3}$$

we have:

$$\begin{aligned} \hat{t}_i &= t_i - \beta_i \\ \hat{x}_i &= q_{t_i} - \beta(q_{t_i} - q_{t_i - 1}) \end{aligned} \tag{4}$$

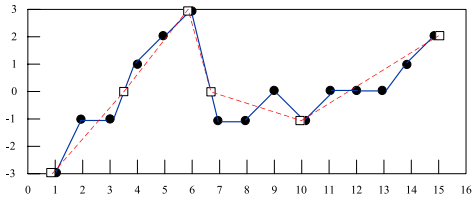

Figure 3. Example of adaptive downsampling using trace segmentation. Black circles represent original sample points (connected by solid blue line). White rectangles represent the reduced points with interpolation (connected with dashed red line). In the figure above, we represent $D_Q = 0, 2, 2, 4, 5, 6, 10, 10, 11, 12, 13, 13, 13, 14, 15$ with $\theta = 5$.

An example illustrating the adaptive downsampling process using trace segmentation is presented in Figure 3. We note that when the data is noisy, the differences $D_Q[n]$ calculated might be mainly from the noise but not the signal itself. In our work, we therefore introduce an exponential filtering pre-processing step that reduces the effect of noise.

The resulting adaptively downsampled representation of the original signal Q is given by two series corresponding to time and amplitude:

$$\begin{aligned} T^Q &= \hat{t}_0, \hat{t}_1, \ldots, \hat{t}_i, \ldots, \hat{t}_\theta \\ X^Q &= \hat{x}_0, \hat{x}_1, \ldots, \hat{x}_i, \ldots, \hat{x}_\theta \end{aligned} \tag{5}$$

This process can be carried out in time that is linear in the size of the input.

B. Trace Profile Comparison (TPC)

Using the results of adaptive downsampling for time series Q and C, we define the distance between these signals as the weighted sum of the l_1 norm of their respective T series and the l_2 norm of respective X series.

$$\begin{aligned} TPC(Q, C) &= (1 - \omega) \sum_{k=0}^{\theta} (X^Q[k] - X^C[k])^2 \\ &+ \omega \sum_{k=0}^{\theta - 1} \mid (l_{Q_k} - l_{C_k}) \mid \end{aligned} \tag{6}$$

where the parameter $\omega \in [0, 1]$ represents the relative importance of changes in timing and amplitude between the time series, while $l_{Q_k} = T^Q[k+1] - T^Q[k]$ and $l_{C_k} = T^C[k+1] - T^C[K]$. Both ω and the parameter θ can be obtained by training on a small amount of data to find values that are best suited for the specific type of signals being compared. To address the situation where the values X and l may occupy different ranges, these series can be normalized. While this does not affect accuracy (i.e., since changes in the relative scaling of X and l lead to different ω values producing identical results), it allows for more intuition on the relative importance of time and amplitude for the task being considered.

TPC can be measured in linear time with the runtime being dominated by the adaptive downsampling process.

C. Piecewise Linear DTW (PLDTW)

In contrast to the basic DTW algorithm, PLDTW quantifies the difference between time series by comparing linear segments of the signals rather than sample values. These segments are derived from the adaptively reduced representations of the time series using linear interpolation. The results of adaptive downsampling are refined prior to the interpolation process (i.e., to match turning points in the neighborhood of the original values) to ensure that the resulting linear segments cannot be extended at either end to provide a better match of the data. In this way, our use of adaptive downsampling is analogous to providing a rough segmentation of the time series.

The comparison of linear segments by PLDTW is carried out in a manner similar to DTW, i.e., using a dynamic programming algorithm to find the minimum cost alignment of linear segments across the signals. We present the PLDTW algorithm by first describing how to measure the cost of alignment (both when segments are aligned one-to-one or one-to-many across signals). We then detail the dynamic programming algorithm that uses these costs to determine the optimal alignment.

1) Aligning Segments One-to-One: Denoting the line segment connecting the adaptively downsampled points $(X^Q[i], T^Q[i])$ and $(X^Q[i+1], T^Q[i+1])$ by S_{Q_i}, and the line segment connecting $(X^C[j], T^C[j])$ and $(X^C[j+1], T^C[j+1])$ by S_{C_j}, we define the normalized distance $ND(S_{Q_i}, S_{C_j})$ between these segments as the difference between the linear equations:

$$\begin{aligned} &X^Q[i] + (X^Q[i+1] - X^Q[i])t \\ &X^C[j] + (X^C[j+1] - X^C[j])t \end{aligned} \tag{7}$$

integrated over the interval $[0, 1]$. The total distance $D(S_{Q_i}, S_{C_j})$ between S_{Q_i} and S_{C_j} is then defined as:

$$ND(S_{Q_i}, S_{C_j}) \max(l_{Q_i}, l_{C_j}) \tag{8}$$

where $l_{Q_i} = T^Q[i+1] - T^Q[i]$ and $l_{C_j} = T^C[j+1] - T^C[j]$.

Intuitively, this approach is analogous to setting up a correspondence at every time instant between S_{Q_i} and S_{C_j} such that time instants along the line segments at the same relative position from the start of each segment are aligned. The resulting difference is weighted by the maximum length of the segments to account for differences in timing. This approach is roughly similar to DTW using a constraint, where the comparison of l_i samples with l_j samples gives rise to a warping path that is roughly of length $\max(l_i, l_j)$ (Figure 4).

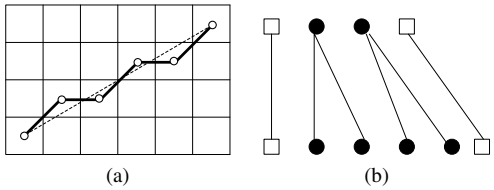

Figure 4. Example warping path (4a) and sample correspondence (4b) when comparing a time series of length $l_i = 4$ with a time series of length $l_j = 6$ under the constraint that the warping path should be close to the diagonal. In this case, the resulting path has a length that is equal to $\max(l_i, l_j) = 6$.

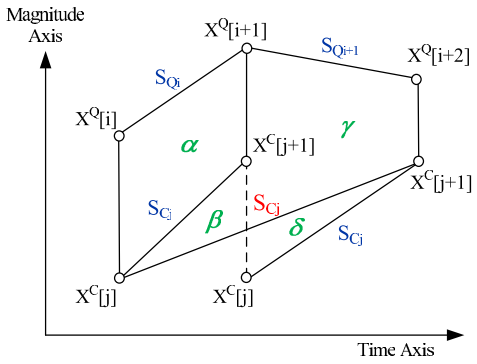

Figure 5. Graphical representation of comparing S_{Q_I} and S_{C_j}.

2) Aligning Segments One-to-Many: Comparing line segments in PLDTW in a manner analogous to DTW also requires us to handle cases where a single line segment from one time series is aligned against multiple contiguous line segments from the other time series. For ease of presentation, we consider the case where two line segments $S_{Q_I} = \{S_{Q_i}, S_{Q_{i+1}}\}$ are aligned against S_{C_j}, although the ideas presented here can be applied to a larger number of segments drawn from either signal in a straightforward manner. In this case, the distance $D(S_{Q_I}, S_{C_j})$ cannot be measured in a simple manner as $D(S_{Q_i}, S_{C_j}) + D(S_{Q_{i+1}}, S_{C_j})$. Instead, it depends on the relationship between the lengths of the segments in I and the segment j.

Figure 5 represents the problem of comparing S_{Q_I} and S_{C_j} graphically. The area marked α corresponds to the normalized distance $ND(S_{Q_i}, S_{C_j})$, while the area marked $\gamma + \delta$ corresponds to the normalized distance $ND(S_{Q_{i+1}}, S_{C_j})$. The normalized distance $ND(S_{Q_I}, S_{C_j})$ is given by $(\alpha + \beta)(l_{Q_i}/(l_{Q_i} + l_{Q_{i+1}})) + \gamma(l_{Q_{i+1}}/(l_{Q_i} + l_{Q_{i+1}}))$ to capture the relative lengths of S_{Q_i} and $S_{Q_{i+1}}$ during normalization. We also note that the product of β and l_{Q_i} is equal to the product of δ and $l_{Q_{i+1}}$. This is a useful property that we draw upon subsequently.

We consider three separate scenarios. Denoting $l_{Q_i}/(l_{Q_i} + l_{Q_{i+1}})$ by λ_i and $l_{Q_{i+1}}/(l_{Q_i} + l_{Q_{i+1}})$ by λ_{i+1}, we first focus on the case where S_{C_j} is longer than S_{Q_I}, i.e., $l_{C_j} > l_{S_i} + l_{S_{i+1}}$. As in Equation 8, we measure the distance between the segments as:

$$\begin{aligned} D(S_{Q_I}, S_{C_j}) &= \max(l_{C_j}, l_{Q_{i,i+1}}) ND(S_{Q_I}, S_{C_j}) \\ &= l_{C_j}\{(\alpha+\beta)\lambda_i + (\gamma)\lambda_{i+1}\} \\ &= l_{C_j}\{(\alpha)\lambda_i + (\gamma+\delta)\lambda_{i+1}\} \\ &= D(S_{Q_i}, S_{C_j})\lambda_i + ND(S_{Q_{i+1}}, S_{C_j})\lambda_{i+1} l_{C_j} \end{aligned} \tag{9}$$

For the case where $l_{Q_i} + l_{Q_{i+1}} \geq l_{C_j} \geq l_{Q_i}$, we then have:

$$\begin{aligned} D(S_{Q_I}, S_{C_j}) &= \max(l_{C_j}, l_{Q_{i,i+1}}) ND(S_{Q_I}, S_{C_j}) \\ &= (l_{S_i} + l_{S_{i+1}})\{(\alpha+\beta)\lambda_i + (\gamma)\lambda_{i+1}\} \\ &= D(S_{Q_i}, S_{C_j})\frac{l_{Q_i}}{l_{C_j}} + ND(S_{Q_{i+1}}, S_{C_j}) l_{Q_{i+1}} \end{aligned} \tag{10}$$

Finally, for the case where $l_{C_j} < l_{Q_i}$ and therefore also $l_{C_j} < l_{Q_i} + l_{Q_{i+1}}$, the distance between the segments can be found by modifying the second case above:

$$\begin{aligned} D(S_{Q_I}, S_{C_j}) &= ND(S_{Q_i}, S_{C_j}) l_{Q_i} + ND(S_{Q_{i+1}}, S_{C_j}) l_{Q_{i+1}} \\ &= D(S_{Q_i}, S_{C_j}) + ND(S_{Q_{i+1}}, S_{C_j}) l_{Q_{i+1}} \end{aligned} \tag{11}$$

We note that for all three cases, $D(S_{Q_I}, S_{C_j})$ can be measured in terms of the distance $D(S_{Q_i}, S_{C_j})$ and the normalized distance $ND(S_{Q_{i+1}}, S_{C_j})$ between the individual segments. In this way, the cost of aligning multiple segments can be easily and efficiently reconstructed from the costs of computing the individual segments earlier by constant additions and multiplications, and the process of integrating the difference between equations does not need to be carried out repeatedly.

3) Dynamic Programming Algorithm: The dynamic programming algorithm for PLDTW is similar to the approach used in basic DTW. One notable difference is that as described in Section III-C2, the cost aligning multiple segments such as $D(S_{Q_I}, S_{C_j})$ cannot be measured as a simple sum $D(S_{Q_i}, S_{C_j}) + D(S_{Q_{i+1}}, S_{C_j})$. As a result, for each horizontal or vertical edge along the path explored by the dynamic programming algorithm, the cost of alignment needs to be recomputed based on the specific segments spanned. This does not, however, affect the theoretical worst case runtime of PLDTW relative to the basic DTW algorithm. While we do not include a formal proof of this property in this paper, this result is due to the reduction effected by adaptive downsampling, as well as our ability to compute the cost of alignment for multiple segments using pre-computed values for alignments of single segments.

IV. Evaluation

We evaluated both TPC and PLDTW on the UCR Time Series dataset [11] drawn from highly diverse application domains. Our choice of this dataset was motivated by its previous use in the unified validation of many different measures for time series comparison [9]. Consistent with this previous work, we used the dataset to evaluate the accuracy and efficiency of our measures in a 1-nearest neighbor task, i.e., the time taken by TPC and PLDTW to find the match for queries using 1-nearest neighbor matching, and the correctness of the resulting match.

The UCR time series dataset is publicly available for research use and contains separate training and test sets for the 1-nearest neighbor task, allowing parameters to be optimized and evaluated separately. We used time series from all twenty application domains in the public dataset. Most of these time series were hundreds of samples long. In addition, for each application domain, time series were associated with labels identifying the classes they belonged to. For most applications, there were fewer than 10 labels or classes. The maximum number of labels was 50 for the 50words application. The size of the training set and testing set varied from 24 to 6,175 signals.

We compared TPC and PLDTW with ED, DTW, CDTW, PAA, and FastDTW to assess the potential improvements in runtime and accuracy provided by TPC and PLDTW. The parameters for TPC (i.e., ω and θ) and PLDTW (i.e., θ) were chosen separately for each dataset to minimize the average 1-nearest neighbor error in the training set. Conversely, ED and DTW did not require any training while for CDTW, PAA and FastDTW we varied the constraint window and downsampling frame sizes separately for each dataset to investigate how these changes affected accuracy and timing. The range of parameters explored for each of these measures was chosen in a similar manner to earlier work on the UCR Time Series dataset [9]. For TPC, we explored values of θ from a minimum of 5 to a maximum corresponding to the length of the original signals in each dataset with a step size of 2% of signal length. PLDTW explored similar values of θ although the maximum length in this case was chosen to be one-third of the length of the original time series. The parameter ω for TPC was varied between 0.01, 0.1, 0.25, 0.5, 0.75, 0.9 and 0.99. For CDTW, we investigated constraint windows R from a minimum value of 1 to a maximum value of 25% of signal length using a step size of 2%. For PAA, we varied the number of segments from 5 till 50% of the signal length using a step size of 2% as well. Finally, for FastDTW, we adopted parameter choices similar to the original presentation of this method [22], which explores the radii 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20 and 30. We report the results for all measures on the test set.

Our experiments were carried out on a machine with dual quad-core Intel Xeon E5520 processors (2.26 GHz, 8MB

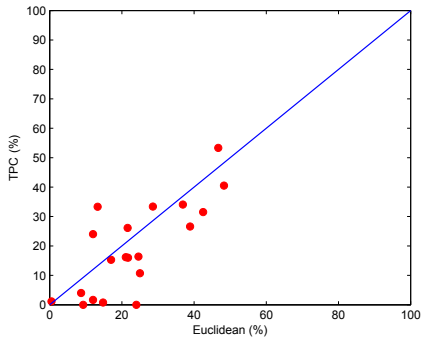

Figure 6. Comparison of average error percentage obtained by ED and TPC on different time series datasets (above the line ED outperforms TPC). TPC achieved a higher accuracy for 14 of the 20 datasets. TPC provided an improvement of greater than 5% in 10 datasets (relative to ED providing a similar improvement over TPC in 3 of the datasets).

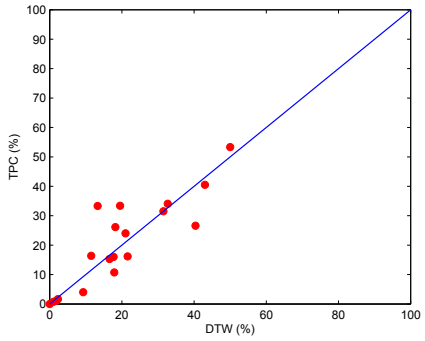

Figure 7. Comparison of average error percentage obtained by DTW and TPC on different time series datasets (above the line DTW outperforms TPC). TPC achieved a higher accuracy for 10 of the 20 datasets and had an identical accuracy to DTW for two additional datasets. TPC provided an improvement of greater than 5% in 4 datasets (relative to DTW providing a similar improvement over TPC in 3 datasets).

Cache) and 12 GB RAM. The measures were uniformly implemented in C++ on the Ubuntu 9.10 operating system.

V. Results

This section presents the results of our evaluation.

A. Using TPC to Improve the Runtime of DTW

Figures 6 and 7 compare the accuracy of TPC with both ED and DTW using the methodology proposed in earlier studies on the UCR Time Series dataset [9]. Generally speaking, TPC outperformed ED and had comparable results to the basic DTW algorithm. However, TPC provided a substantial improvement over the runtime of DTW on all datasets (Figure 8).

We further compared the efficiency of TPC against the results obtained for CDTW, PAA and FastDTW. Since the parameters for each of these measures can be chosen differently to either improve runtime or to obtain accuracy similar to DTW, we report two sets of results: the minimum runtime (MinTime) for these methods to obtain results that were equal to or better than the results obtained for TPC, and the minimum average error (MinErr) of these methods to achieve a runtime equal to or better than TPC (Table I).

The runtime of TPC was superior to the runtime of both CDTW and FastDTW, as neither of these methods was able to achieve efficiency similar to TPC over any of the parameter choices investigated (Section IV), regardless of accuracy. Moreover, while PAA was able to achieve an efficiency comparable to TPC for all datasets, the accuracy of the results obtained for the 1-nearest neighbor task were generally much worse. Similarly, the runtime of CDTW, PAA and FastDTW to achieve accuracy comparable to TPC was typically an order of magnitude higher. In some cases, CDTW, PAA and FastDTW were also unable to achieve an accuracy better than TPC.

B. Using PLDTW to Improve the Accuracy of DTW

Figure 9 compares the accuracy of PLDTW with DTW. Although the overall performance of PLDTW and DTW was comparable, for certain applications PLDTW improved accuracy in the 1-nearest neighbor task significantly. In particular, the performance of PLDTW was generally superior to DTW on longer datasets. For the 8 datasets where the training and testing sequences were 300 samples or longer (roughly the average length across all datasets), PLDTW was outperformed by the basic DTW algorithm in only 2 cases. In contrast to this, for the remaining 12 datasets with fewer than 300 samples, DTW achieved a higher accuracy than PLDTW in 8 cases. We also note that the runtime of PLDTW was similar to that of DTW. The median ratio of the PLDTW runtime to the DTW runtime across the datasets was 0.99, with an interquartile range from 0.82 to 1.56.

VI. Discussion

In this paper, we explored the idea of adaptive downsampling to improve upon the efficiency and accuracy of DTW. Our work focused on a trace segmentation-based approach to distinguish between rapidly and slowly changing parts of a time series. We described how this approach can be used to reduce the size of time series being compared with limited loss of information, by downsampling slowly changing parts of the signals much more than rapidly changing regions.

We proposed two novel measures based on adaptive downsampling. Our first measure, TPC, compared the reduced representations of time series activity obtained by adaptive downsampling by employing a weighting scheme that captured the relative importance of changes in amplitude and timing. When evaluated on the UCR Time Series dataset, with time series drawn from a variety of real-world applications, TPC provided a substantial improvement over the runtime of DTW while achieving similar accuracy in a 1-nearest

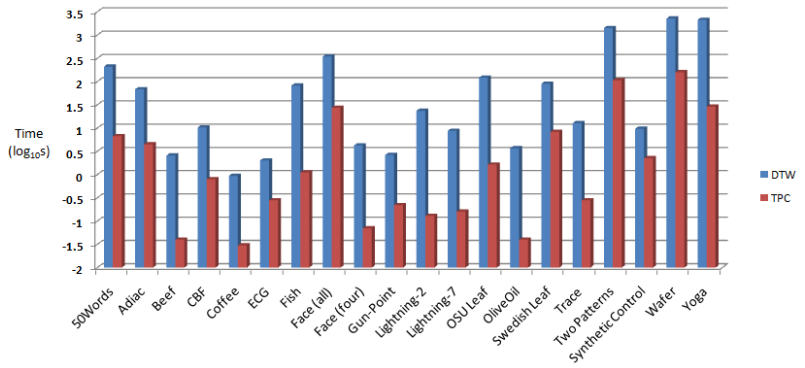

Figure 8. Comparison of DTW and TPC runtime on different datasets (time is shown on log scale).

	TPC		CDTW		PAA		FastDTW	
	Error	Time (s)	MinTim	MinErr	MinTim	MinErr	MinTim	MinErr
50words	29.7%	6.72	N/A	99.69	44.6%	N/A	N/A	N/A
Adiac	37.1%	4.49	N/A	33.90	45.3%	34.91	N/A	N/A
Beef	50.0%	0.04	N/A	1.08	53.3%	N/A	N/A	N/A
CBF	1.6%	0.80	N/A	4.94	3.0%	1.14	N/A	3.25
Coffee	14.3%	0.03	N/A	N/A	21.4%	N/A	N/A	0.19
ECG200	18.0%	0.28	N/A	0.98	16.0%	0.19	N/A	N/A
Fish	24.0%	1.11	N/A	35.85	22.9%	1.06	N/A	10.97
FaceAll	33.0%	27.42	N/A	143.74	39.1%	58.58	N/A	116.68
FaceFour	34.1%	0.07	N/A	1.51	27.3%	0.05	N/A	0.59
Gun-Point	3.3%	0.22	N/A	1.41	10.7%	N/A	N/A	N/A
Lighting2	16.4%	0.13	N/A	8.30	23.0%	0.29	N/A	1.94
Lighting7	42.5%	0.16	N/A	3.10	37.0%	0.10	N/A	1.34
OSULeaf	37.6%	1.64	N/A	N/A	47.9%	N/A	N/A	N/A
OliveOil	33.3%	0.04	N/A	1.54	16.7%	0.02	N/A	1.11
SwedishLeaf	19.0%	8.30	N/A	44.88	22.6%	N/A	N/A	103.70
Trace	0.0%	0.28	N/A	N/A	10.0%	N/A	N/A	N/A
TwoPatterns	0.0%	107.55	N/A	N/A	4.9%	N/A	N/A	N/A
SyntheticControl	1.7%	2.26	N/A	6.12	2.7%	N/A	N/A	12.87

Table I
COMPARISON OF TPC WITH CDTW, PAA AND FASTDTW. IN EACH CASE, THE MINIMUM RUNTIME (MINTIME) FOR CDTW, PAA AND FASTDTW TO OBTAIN RESULTS THAT WERE EQUAL TO OR BETTER THAN THE RESULTS OBTAINED FOR TPC, AND THE MINIMUM AVERAGE ERROR (MINERR) OF THESE METHODS AT A RUNTIME EQUAL TO OR BETTER THAN THAT OF TPC, ARE REPORTED. N/A CORRESPONDS TO CASES WHERE CDTW, PAA AND FASTDTW COULD EITHER NOT ACHIEVE AN ACCURACY OR RUNTIME COMPARABLE TO OR BETTER THAN TPC.

neighbor task. Moreover, our results showed that TPC also provided an improvement over CDTW, PAA and FastDTW in runtime, and also in some cases obtained more accurate results than were possible with these other measures. The TPC metric, which exhibits a tight linear runtime bound, therefore benefits from the desirable runtime properties of ED while achieving accuracy that is comparable to DTW.

Our second metric, PLDTW, compared piecewise linear segments constructed from adaptively reduced time series using a modified dynamic programming algorithm and cost function. Our results on the UCR Time Series dataset showed that PLDTW achieved a comparable accuracy to DTW. However, for some datasets, the comparison of piecewise linear segments rather than sample values improved results substantially. This was particularly true for datasets with longer time series. Our experiments also suggested that PLDTW can theoretically improve the result of DTW for both short and long time series considerably. This improvement is subject to developing better ways to pick parameters for PLDTW during training.

Our choice of evaluation on the UCR Time Series dataset was based on its ability to provide a uniform assessment of time series similarity measures on real-world data from diverse sources. We believe that in addition to our exploration of the idea of adaptive downsampling, and introduction of TPC and PLDTW, our work may also represents the first detailed investigation of PAA and FastDTW on these data.

We conclude with a discussion of some limitations of our study. While the UCR Time Series dataset represents a useful resource, our measures need to be evaluated more completely on data from additional applications, larger datasets, and on tasks beyond 1-nearest neighbor classification. We

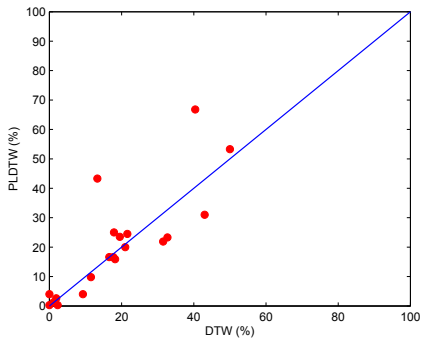

Figure 9. Comparison of average error percentage obtained by DTW and PLDTW on different time series datasets (above the line DTW outperforms PLDTW). PLDTW achieved a higher accuracy for 9 of the 20 datasets and had an identical accuracy to DTW for one additional datasets. PLDTW provided an improvement of greater than 5% in 4 datasets (relative to DTW providing a similar improvement over PLDTW in 3 of the datasets).

also believe that more work is needed to choose the parameters for PLDTW so that this approach can realize the theoretical improvements possible. Finally, while we use trace segmentation as one approach to carry out adaptive downsampling, the general idea of reducing rapidly changing parts of signals more than slowly changing ones can be implemented using a variety of other approaches. For example, while previous work on locally adaptive dimensionality reduction [14] and time series segmentation [15] have not directly addressed the goal of improving upon the efficiency and accuracy of DTW, there are ideas in both sets of works that are relevant to the goal of adaptive downsampling. The development and comparative exploration of such orthogonal approaches is necessary to identify the best approach for achieving this reduction.

ACKNOWLEDGMENT

We thank Eamonn Keogh for suggesting and sharing the datasets used in our experiments.

REFERENCES

[1] R. Agrawal, C. Faloutsos and A. Swami. Efficient similarity search in sequence databases. *FODO*, 1993;69-84.

[2] J. Aßfalg, H.P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin and M. Renz. Similarity search on time series based on threshold queries. *EDBT*, 2006;276-294

[3] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. *SIGKDD*, 1994.

[4] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. *SIGMOD*, 2004.

[5] K. Chan and A.W.C. Fu. Efficient Time Series Matching by Wavelets. *ICDE*, 1999.

[6] L. Chen and R.T. Ng. On the marriage of lp-norms and edit-distance. *VLDB*, 2004.

[7] L. Chen, M. T. Özsu and V. Oria. Robust and fast similarity search for moving object trajectories. *SIGMOD*, 2005.

[8] Y.Chen, M.A. Nascimento, B.C. Ooi and A.K.H.Tung. SpADe :on shape-based pattern detection in streaming time series. *ICDE*, 2007.

[9] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *VLDB*, 2008.

[10] C. Faloutsos, M. Ranganathan and Y. Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD*, 1994.

[11] E. Keogh, X. Xi, L. Wei and C. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage: $http : /www.cs.ucr.edu/\tilde{}eamonn/time\_series\_data/$

[12] E. Keogh and M. Pazzani. Derivative dynamic time warping. *SDM*, 2001.

[13] E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series database. *KAIS*, 2001:3;263-283.

[14] E. Keogh, K. Chakrabarti, S. Mehrotra and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *SIGMOD*, 2001.

[15] E. Keogh, S. Chu, D. Hart and M. Pazzani. An online algorithm for segmenting time series. *ICDM*, 2002.

[16] E. Keogh and C.A. Ratanamahatana. Exact indexing of dynamic time warping. *KAIS*, 2005:7;358-386.

[17] F. Korn, H. Jagadish and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *SIGMOD*, 1997.

[18] M.H. Kuhn, H. Tomaschewski and N. Hermann Fast nonlinear time alignment for isolated word recognition. *ICASSP*, 1981.

[19] M.D. Morse and J.M. Patel. An efficient and accurate method for evaluating time series similarity. *SIGMOD*, 2007.

[20] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans Acoust Speech Signal Process*, 1978:26;43-49.

[21] H. Sakoe and S. Chiba. *Readings in Speech Recognition.* Morgan Kaufmann, 1990.

[22] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intell Data Anal*, 2007:11;561-580.

[23] M.Vlachos, D.Gunopulos and G.Kollios. Discovering similar multidimensional trajectories. *ICDE*, 2002.

[24] Z.Syed, J. Guttag, and C. Stultz. Clustering and symbolic analysis of cardiovascular signals: discovery and visualization of medically relevant patterns in long-term data using limited prior knowledge. *EURASIP J Adv Sig Proc*, 2007:1.