Data Mining and Knowledge Discovery (2023) 37:2024–2045
https://doi.org/10.1007/s10618-023-00926-8

Parameterizing the cost function of dynamic time warping with application to time series classification

Matthieu Herrmann[1] · **Chang Wei Tan**[1] · **Geoffrey I. Webb**[1]

Received: 3 August 2022 / Accepted: 6 February 2023 / Published online: 16 April 2023

Abstract

Dynamic time warping (*DTW*) is a popular time series distance measure that aligns the points in two series with one another. These alignments support warping of the time dimension to allow for processes that unfold at differing rates. The distance is the minimum sum of costs of the resulting alignments over any allowable warping of the time dimension. The cost of an alignment of two points is a function of the difference in the values of those points. The original cost function was the absolute value of this difference. Other cost functions have been proposed. A popular alternative is the square of the difference. However, to our knowledge, this is the first investigation of both the relative impacts of using different cost functions and the potential to tune cost functions to different time series classification tasks. We do so in this paper by using a tunable cost function λ_γ with parameter γ. We show that higher values of γ place greater weight on larger pairwise differences, while lower values place greater weight on smaller pairwise differences. We demonstrate that training γ significantly improves the accuracy of both the *DTW* nearest neighbor and Proximity Forest classifiers.

Keywords Time series · Classification · Dynamic time warping · Elastic distances

Responsible editor: Johannes Fürnkranz.

This work was supported by the Australian Research Council award DP210100072.

✉ Chang Wei Tan
chang.tan@monash.edu

Matthieu Herrmann
matthieu.herrmann@monash.edu

Geoffrey I. Webb
geoff.webb@monash.edu

[1] Monash University, Clayton Campus, Woodside Building, 20 Exhibition Walk, Clayton, VIC 3800, Australia

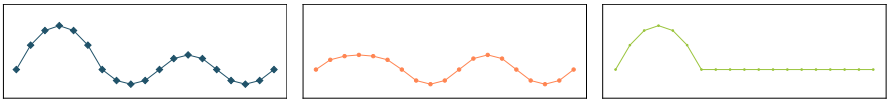
1 Introduction

Similarity and distance measures are fundamental to data analytics, supporting many key operations including similarity search (Rakthanmanon et al. 2012), classification (Shifaz et al. 2020), regression (Tan et al. 2021a), clustering (Petitjean et al. 2011), anomaly and outlier detection (Diab et al. 2019), motif discovery (Alaee et al. 2021), forecasting (Bandara et al. 2021), and subspace projection (Deng et al. 2020).

Dynamic time warping (*DTW*) (Sakoe and Chiba 1971, 1978) is a popular distance measure for time series and is often employed as a similarity measure such that the lower the distance the greater the similarity. It is used in numerous applications including speech recognition (Sakoe and Chiba 1971, 1978), gesture recognition (Cheng et al. 2016), signature verification (Okawa 2021), shape matching (Yasseen et al. 2016), road surface monitoring (Singh et al. 2017), neuroscience (Cao et al. 2016) and medical diagnosis (Varatharajan et al. 2018).

DTW aligns the points in two series and returns the sum of the pairwise-distances between each of the pairs of points in the alignment. *DTW* provides flexibility in the alignments to allow for series that evolve at differing rates. In the univariate case, pairwise-distances are usually calculated using a *cost function*, $\lambda(a \in \mathcal{R}, b \in \mathcal{R}) \rightarrow \mathcal{R}^+$. When introducing *DTW*, Sakoe and Chiba (1971) defined the cost function as $\lambda(a, b) = |a - b|$. However, other cost functions have subsequently been used. The cost function $\lambda(a, b) = (a - b)^2$ (Tan et al. 2018; Dau et al. 2019; Mueen and Keogh 2016; Löning et al. 2019; Tan et al. 2020) is now widely used, possibly inspired by the (squared) Euclidean distance. ShapeDTW (Zhao and Itti 2018) computes the cost between two points by computing the cost between the "shape descriptors" of these points. Such a descriptor can be the Euclidean distance between segments centered on this points, taking into account their local neighborhood.

To our knowledge, there has been little research into the influence of tuning the cost function on the efficacy of *DTW* in practice. This paper specifically investigates how actively tuning the cost function influences the outcome on a clearly defined benchmark. We do so using $\lambda_\gamma(a, b) = |a - b|^\gamma$ as the cost function for *DTW*, where



(a) Series S (b) Series T (c) Series U

Fig. 1 Tuning the cost function changes which series are considered more similar to one another. U exactly matches the first 7 points of S, but then flattens, running through the center of the remaining points in S. In contrast, T starts with lower amplitude than S over the first seven points, but then exactly matches S for the remaining low amplitude waves. The original *DTW* cost function, $\lambda(a, b) = |a - b|$, results in $DTW(S, T) = DTW(S, U) = 9$, with *DTW* rating T and U as equally similar to S. The commonly used cost function, $\lambda(a, b) = (a - b)^2$, results in $DTW(S, U) = 9.18 < DTW(S, T) = 16.66$. More weight is placed on the high amplitude start, and S is more similar to U. Using the cost function $\lambda(a, b) = |a - b|^{0.5}$ results in $DTW(S, U) = 8.98 > DTW(S, T) = 6.64$, placing more weight on the low amplitude end, and S is more similar to T. In general, changing the cost function alters the amount of weight placed on low amplitude vs high amplitude effects, allowing *DTW* to be better tuned to the varying needs of different applications

$\gamma = 1$ gives us the original cost function; and $\gamma = 2$ the now commonly used squared Euclidean distance.

We motivate this research with an example illustrated in Fig. 1 relating to three series, S, T and U. U exactly matches S in the high amplitude effect at the start, but does not match the low amplitude effects thereafter. T does not match the high amplitude effect at the start but exactly matches the low amplitude effects thereafter. Given these three series, we can ask which of T or U is the nearest neighbor of S?

As shown in Fig. 1, the answer varies with γ. Low γ emphasizes low amplitude effects and hence identifies S as more similar to T, while high γ emphasizes high amplitude effects and assesses U as most similar to S. Hence, we theorized that careful selection of an effective cost function on a task by task basis can greatly improve accuracy, which we demonstrate in a set of nearest neighbor time series classification experiments. Our findings extend directly to all applications relying on nearest neighbor search, such as ensemble classification (we demonstrate this with Proximity Forest Lucas et al. 2019) and clustering, and have implications for all applications of *DTW*.

The remainder of this paper is organized as follows. In Sect. 2, we provide a detailed introduction to *DTW* and its variants. In Sect. 3, we present the flexible parametric cost function λ_γ and a straightforward method for tuning its parameter. Section 4 presents experimental assessment of the impact of different *DTW* cost functions, and the efficacy of *DTW* cost function tuning in similarity-based time series classification (TSC). Section 5 provides discussion, directions for future research and conclusions.

2 Background

2.1 Dynamic time warping

The *DTW* distance measure (Sakoe and Chiba 1971) is widely used in many time series data analysis tasks, including nearest neighbor (*NN*) search (Rakthanmanon et al. 2012; Tan et al. 2021a; Petitjean et al. 2011; Keogh and Pazzani 2001; Silva et al. 2018). Nearest neighbor with *DTW* (*NN-DTW*) has been the historical approach to time series classification and is still used widely today.

DTW computes the cost of an optimal alignment between two equal length series, S and T with length L in $O(L^2)$ time (lower costs indicating more similar series), by minimizing the cumulative cost of aligning their individual points, also known as the warping path. The warping path of S and T is a sequence $\mathcal{W} = \langle \mathcal{W}_1, \ldots, \mathcal{W}_P \rangle$ of alignments (dotted lines in Fig. 2). Each alignment is a pair $\mathcal{W}_k = (i, j)$ indicating that S_i is aligned with T_j. $\mathcal{W}$ must obey the following constraints:

- *Boundary Conditions*: $\mathcal{W}_1 = (1, 1)$ and $\mathcal{W}_P = (L, L)$.
- *Continuity* and *Monotonicity*: for any $\mathcal{W}_k = (i, j)$, $1 < k \leq P$, we have $\mathcal{W}_{k+1} \in \{(i+1, j), (i, j+1), (i+1, j+1)\}$.

The cost of a warping path is minimized using dynamic programming by building a "cost matrix" M_{DTW} for the two series S and T, such that $M_{DTW}(i, j)$ is the minimal cumulative cost of aligning the first i points of S with the first j points of T. The cost

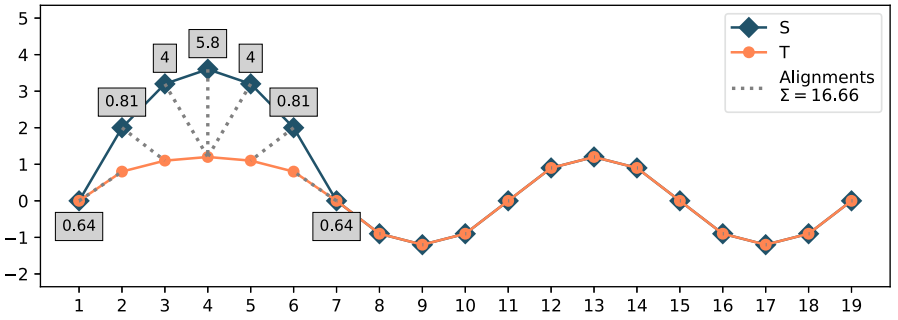
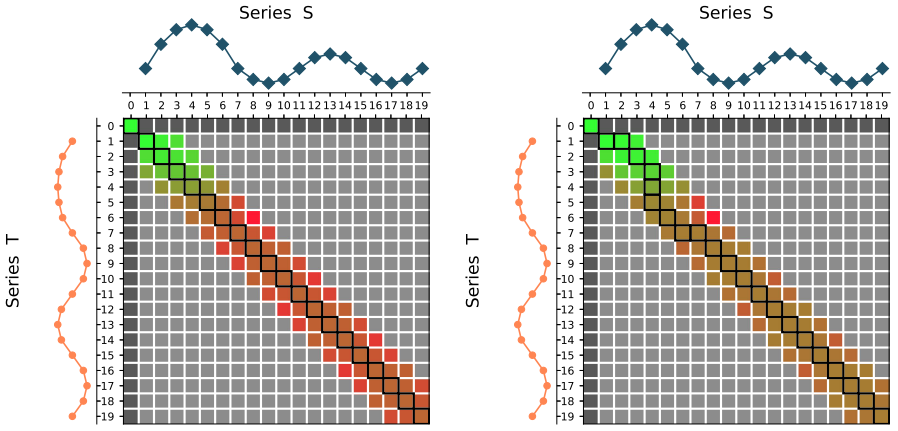


Fig. 2 Pairwise alignments of $DTW(S, T)$ with $\gamma = 2$, accumulating a total cost of 16.66. We only show non-zero alignments



(a) $DTW^1(S, T)$ with $w = 2$ (b) $DTW^2(S, T)$ with $w = 2$

Fig. 3 $M_{DTW(S,T)}$ with warping window $w = 2$, and different cost function exponent, **a** $\gamma = 1$ and **b** $\gamma = 2$. We have $DTW(S, T)$=$M_{DTW(S,T)}(L, L)$. The amplitude of the cumulative cost is represented by a green (minimal) to red (maximal) gradient. Cells cut-out by the warping window are in light gray, borders are in dark gray. The warping path cells are highlighted with black borders. Notice how the deviation from the diagonal in (**b**) corresponds to the alignments Fig. 2 (Color figure online)

matrix is defined in Eqs. (1a) to (1d), where $\lambda(S_i, T_j)$ is the cost of aligning the two points, discussed in Sect. 3. It follows that $DTW(S, T)$=$M_{DTW}(L, L)$.

$$M_{DTW}(0, 0) = 0 \tag{1a}$$

$$M_{DTW}(i, 0) = +\infty \tag{1b}$$

$$M_{DTW}(0, j) = +\infty \tag{1c}$$

$$M_{DTW}(i, j) = \lambda(S_i, T_j) + \min \begin{cases} M_{DTW}(i-1, j-1) \\ M_{DTW}(i-1, j) \\ M_{DTW}(i, j-1) \end{cases} \tag{1d}$$

Figure 3 shows the cost matrix of computing $DTW(S, T)$. The warping path is highlighted using the bold boxes going through the matrix.

DTW is commonly used with a global constraint applied on the warping path, such that S_i and T_j can only be aligned if they are within a window range, w. This limits the distance in the time dimension that can separate S_i from points in T with which it can be aligned (Sakoe and Chiba 1971; Keogh and Ratanamahatana 2005). This constraint is known as the warping window, w (previously Sakoe-Chiba band) (Sakoe and Chiba 1971). Note that we have $0 \leq w \leq L-2$; *DTW* with $w=0$ corresponds to a *direct alignment* in which $\forall_{(i,j)\in\mathcal{W}}\, i=j$; and *DTW* with $w \geq L-2$ places no constraints on the distance between the points in an alignment. Figure 3 shows an example with warping window w=2, where the alignment of S and T is constrained to be inside the colored band. Light gray cells are "forbidden" by the window.

Warping windows provide two main benefits: (1) preventing pathological warping of S and T; and (2) speeding up *DTW* by reducing its complexity from $O(L^2)$ to $O(W \cdot L)$ (Tan et al. 2018, 2021b).

Alternative window constraints have also been developed, such as the Itakura Parallelogram (Itakura 1975) and the Ratanamahatana–Keogh band (Ratanamahatana and Keogh 2004). In this paper, we focus on the Sakoe-Chiba Band which is the constraint defined in the original definition of *DTW*.

2.2 Amerced dynamic time warping

DTW uses a crude step function to constrain the alignments, where any warping is allowed within the warping window and none beyond it. This is unintuitive for many applications, where some flexibility in the exact amount of warping might be desired. The Amerced Dynamic Time Warping (*ADTW*) distance measure is an intuitive and effective variant of *DTW* (Herrmann and Webb in press). Rather than using a tunable hard constraint like the warping window, it applies a tunable additive penalty ω for non-diagonal (warping) alignments (Herrmann and Webb in press).

ADTW is computed with dynamic programming, similar to *DTW*, using a cost matrix M_{ADTW} with $ADTW_\omega(S,T) = M_{ADTW}(L,L)$. Equations 2a to 2d describe this cost matrix, where $\lambda(S_i, T_j)$ is the cost of aligning the two points, discussed in Sect. 3.

$$M_{ADTW}(0,0) = 0 \tag{2a}$$

$$M_{ADTW}(i,0) = +\infty \tag{2b}$$

$$M_{ADTW}(0,j) = +\infty \tag{2c}$$

$$M_{ADTW}(i,j) = \min \begin{cases} M_{ADTW}(i-1,j-1) + \lambda(S_i, T_j) \\ M_{ADTW}(i-1,j) + \lambda(S_i, T_j) + \omega \\ M_{ADTW}(i,j-1) + \lambda(S_i, T_j) + \omega \end{cases} \tag{2d}$$

The parameter ω works similarly to the warping window, allowing *ADTW* to be as flexible as *DTW* with $w = L-2$, and as constrained as *DTW* with $w=0$. A small penalty should be used if large warping is desirable, while large penalty minimizes warping. Since ω is an additive penalty, its scale relative to the time series in context matters, as a small penalty in a given problem maybe a huge penalty in another one. An automated parameter selection method has been proposed in the context of time

series classification that considers the scale of ω (Herrmann and Webb in press). The scale of penalties is determined by multiplying the maximum penalty ω' by a ratio $0 \leq r \leq 1$, i.e. $\omega = \omega' \times r$. The maximum penalty ω' is set to the average "direct alignment" sampled randomly from pairs of series in the training dataset, using the specifed cost function. A direct alignment does not allow any warping, and corresponds to the diagonal of the cost matrix (e.g. the warping path in Fig. 3a). Then 100 ratios are sample from $r_i = (\frac{i}{100})^5$ for $1 \leq i \leq 100$ to form the search space for ω. Apart from being more intuitive, *ADTW* when used in a *NN* classifier is significantly more accurate than *DTW* on 112 UCR time series benchmark datasets (Herrmann and Webb in press).

Note that ω can be considered as a direct penalty on path length. If series S and T have length L and the length of the warping path for $ADTW_\omega(S, T)$ is P, the sum of the ω terms added will equal $2\omega(P - L + 1)$. The longer the path, the greater the penalty added by ω.

3 Tuning the cost function

DTW was originally introduced with the cost function $\lambda(a, b) = |a - b|$. Nowadays, the cost function $\lambda(a, b) = (a - b)^2 = |a - b|^2$ is also widely used (Dau et al. 2019; Mueen and Keogh 2016; Löning et al. 2019; Tan et al. 2020). Some generalizations of DTW have also incorporated tunable cost functions (Deriso and Boyd 2022). To our knowledge, the relative strengths and weaknesses of these two common cost functions has not previously been thoroughly evaluated. To study the impact of the cost function on *DTW*, and its recent refinement *ADTW*, we use the cost function $\lambda_\gamma(a, b) = |a - b|^\gamma$.

We primarily study the cost functions λ_γ for $\gamma \in \Gamma = \{1/2, 1/1.5, 1, 1.5, 2\}$. This includes the original *DTW* cost function $|a - b| = \lambda_1(a, b)$, and the more recent $(a - b)^2 = \lambda_2(a, b)$. To the best of our knowledge, the remaining cost functions, $\lambda_{0.5}(a, b)$, $\lambda_{0.\dot{6}}(a, b)$ and $\lambda_{1.5}(a, b)$, have not been previously investigated. As illustrated in Fig. 4, Relative to 1, larger values of γ penalize small differences less, and larger differences more. Reciprocally, smaller values of γ penalize large differences more, and small differences less.

We will show in Sect. 4 that learning γ at train time over these 5 values is already enough to significantly improve nearest neighbor classification test accuracy. We will also show that expanding Γ to a *larger* set {1/5, 1/4, 1/3, 1/2, 1/1.5, 1, 1.5, 2, 3, 4, 5}, or a *denser* set {1/2, 1/1.75, 1/1.5, 1/1.25, 1, 1.25, 1.5, 1.75, 2} does not significantly improve the classification accuracy, even with doubling the number of explored parameters. Note that all the sets have the form $\{\frac{1}{n} \ldots 1 \ldots n\}$. Although this balancing is not necessary, we did so to strike a balance in the available exponents.

Tuning λ_γ amounts to learning the parameter γ at train time. This means that we now have two parameters for both *DTW* (the warping window w and γ) and *ADTW* (the penalty ω and γ). In the current work, the w and ω parameters are always learned independently for each γ, using the standard method (Herrmann and Webb in press). We denote *DTW* with $\lambda_x = |S_i - T_j|^x$ as DTW^x, and *ADTW* with λ_x as $ADTW^x$. We

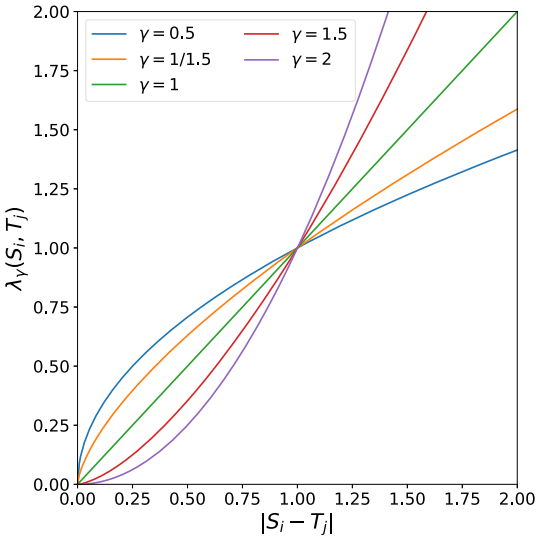


Fig. 4 Illustration of the effect of $\gamma \in \{1/2, 1/1.5, 1, 1.5, 2\}$ on λ_γ

indicate that the cost function has been tuned with the superscript $+$, i.e. DTW^+ and $ADTW^+$.

Note that with a window $w = 0$,

$$DTW^+(S, T) = \sum_{i=1}^{n} (S_i - T_i)^\gamma \tag{3}$$

for the selected exponent γ. In other words, it is the Minkowski distance (Thompson and Thompson 1996) to the power γ, providing the same relative order as the Minkowski distance, i.e. they both have the same effect for nearest neighbor search applications.

The parameters w and ω have traditionally been learned through leave-one-out cross-validation (LOOCV) evaluating 100 parameter values (Tan et al. 2018, 2020, 2021b; Lines and Bagnall 2015). Following this approach, we evaluate 100 parameter values for w (and ω) per value of γ, i.e. we evaluate 500 parameter values for DTW^+ and $ADTW^+$. To enable a fair comparison in Sect. 4 of DTW^+ (resp. $ADTW^+$) against DTW^γ (resp. $ADTW^\gamma$) with fixed γ, the latter are trained evaluating both 100 parameter values (to give the same space of values for w or ω) as well as 500 parameter values (to give the same overall number of parameter values).

Given a fixed γ, LOOCV can result in multiple parameterizations for which the train accuracy is equally best. We need a procedure to break ties. This could be achieved through random choice, in which case the outcome becomes nondeterministic (which may be desired). Another possibility is to pick a parameterization depending on other considerations. For DTW, we pick the smallest windows as it leads to faster computations. For $ADTW$, we follow the paper (Herrmann and Webb in press) and pick the median value.

We also need a procedure to break ties when more than one pair of values over two different parameters all achieve equivalent best performance. We do so by forming a hierarchy over the parameters. We first pick a best value for w (or ω) per possible γ, forming dependent pairs (γ, w) (or (γ, ω)). Then, we break ties between pairs by picking the one with the median γ. In case of an even number of equal best values for γ, taking a median would result in taking an average of dependent pairs, which does not make sense for the dependent value (w or ω). In this case we select between the two *middle* pairs the one with a γ value closer to 1, biasing the system towards a balanced response to differences less than or greater than zero.

Our method does not change the overall time complexity of learning *DTW*'s and *ADTW*'s parameters. The time complexity of using LOOCV for nearest neighbor search with this distances is $O(M.N^2.L^2)$, where M is the number of parameters, N is the number of training instances, and L is the length of the series. Our method only impacts the number of parameters M. Hence, using 5 different exponents while keeping a hundred parameters for w or ω effectively increases the training time 5 fold.

4 Experimentation

We evaluate the practical utility of cost function tuning by studying its performance in nearest neighbor classification. While the technique has potential applications well beyond classification, we choose this specific application because it has well accepted benchmark problems with objective evaluation criteria (classification accuracy). We experimented over the widely-used time series classification benchmark of the UCR archive (Dau et al. 2018), removing the datasets containing series of variable length or classes with only one training exemplar, leading to 109 datasets. We investigate tuning the exponent γ for DTW^+ and $ADTW^+$ using the following sets (and we write e.g. DTW^{+a} when using the set a):

- The *default* set $a = \{1/2, 1/1.5, 1, 1.5, 2\}$
- The *large* set $b = \{1/5, 1/4, 1/3, 1/2, 1/1.5, 1, 1.5, 2, 3, 4, 5\}$
- The *dense* set $c = \{1/2, 1/1.75, 1/1.5, 1/1.25, 1, 1.25, 1.5, 1.75, 2\}$.

The default set a is the one used in Fig. 4, and the one we recommend.

We show that a wide range of different exponents γ each perform best on different datasets. We then compare DTW^{+a} and $ADTW^{+a}$ against their classic counterparts using $\gamma = 1$ and $\gamma = 2$. We also address the question of the number of evaluated parameters, showing with both *DTW* and *ADTW* that tuning the cost function is more beneficial than evaluating 500 values of either w or ω with a fixed cost function. We then show that compared to the large set b (which looks at exponents beyond $1/2$ and 2) and to the dense set c (which looks at more exponents between $1/2$ and 2), a offer similar accuracy while being less computationally demanding (evaluating less parameters). Just as *ADTW* is significantly more accurate than *DTW* (Herrmann and Webb in press), $ADTW^{+a}$ remains significantly more accurate than DTW^{+a}. This holds for sets b and c.

Finally, we show that parameterizing the cost function is also beneficial in an ensemble classifier, showing a significant improvement in accuracy for the leading similarity-based TSC algorithm, Proximity Forest (Lucas et al. 2019).

4.1 Analysis of the impact of exponent selection on accuracy

Figure 5 shows the number of datasets for which each exponent results in the highest accuracy on the test data for each of our NN classifiers and each of the three sets of exponents. It is clear that there is great diversity across datasets in terms of which γ is most effective. For *DTW*, the extremely small $\gamma = 0.2$ is desirable for 12% of datasets and the extremely large $\gamma = 5.0$ for 8%.

The optimal exponent differs between DTW^{γ} and $ADTW^{\gamma}$, due to different interactions between the window parameter w for *DTW* and the warping penalty parameter ω for *ADTW*. We hypothesize that low values of γ can serve as a form of pseudo ω, penalizing longer paths by penalizing large numbers of small difference alignments. *ADTW* directly penalizes longer paths through its ω parameter, reducing the need to deploy γ in this role. If this is correct then *ADTW* has greater freedom to deploy γ to focus more on low or high amplitude effects in the series, as illustrated in Fig. 1.

4.2 Comparison against non tuned cost functions

Figures 6 and 7 present accuracy scatter plots over the UCR archive. A dot represents the test accuracy of two classifiers on a dataset. A dot on the diagonal indicates equal performance for the dataset. A dot off the diagonal means that the classifier on the corresponding side (indicated in top left and bottom right corners) is more accurate than its competitor on this dataset.

On each scatter plot, we also indicate the number of times a classifier is strictly more accurate than its competitor, the number of ties, and the result of a Wilcoxon signed ranks test indicating whether the accuracy of the classifiers can be considered significantly different. Following common practice, we use a significance level of 0.05.

Figures 6 and 7 show that tuning the cost function is beneficial for both *DTW* and *ADTW*, when compared to both the original cost function λ_1, and the popular λ_2. The Wilcoxon signed ranks test for DTW^{+} show that DTW^{+} significantly outperforms both DTW^{1} and DTW^{2}. Similarly, $ADTW^{+}$ significantly outperforms both $ADTW^{1}$ and $ADTW^{2}$.

4.3 Investigation of the number of parameter values

DTW^{+} and $ADTW^{+}$ are tuned on 500 parameter options. To assess whether their improved accuracy is due to an increased number of parameter options rather than due to the addition of cost tuning per se, we also compared them against DTW^{1} and $ADTW^{1}$ also tuned with 500 options for their parameters w and ω instead of the usual 100. Figure 8 shows that increasing the number of parameter values available to DTW^{1} and $ADTW^{1}$ does not alter the advantage of cost tuning. Note that the warping window

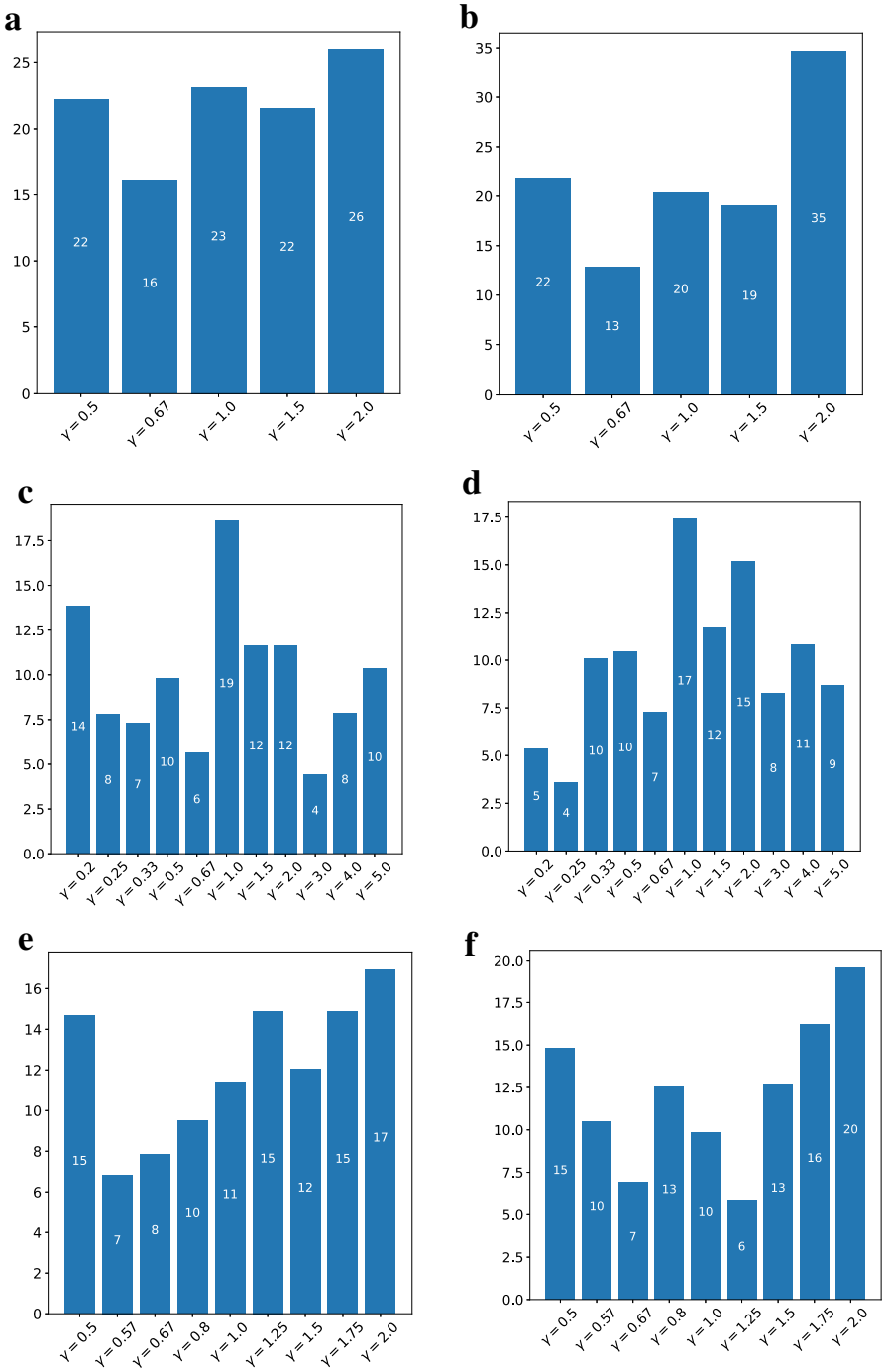
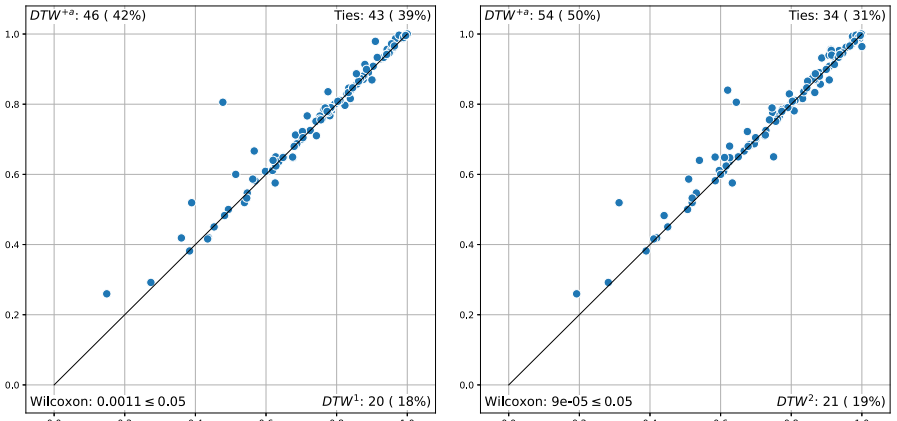
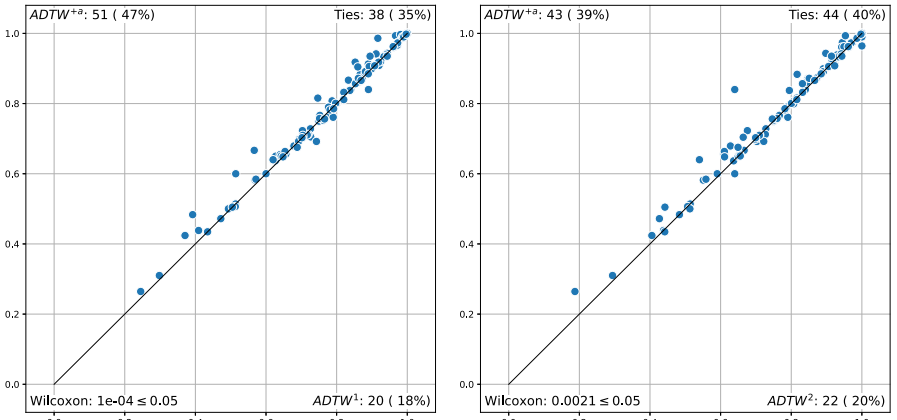


Fig. 5 Counts of the numbers of datasets for which each value of γ results in the highest accuracy on the test data



(a) DTW^{+a} vs. DTW^{1} (b) DTW^{+a} vs. DTW^{2}

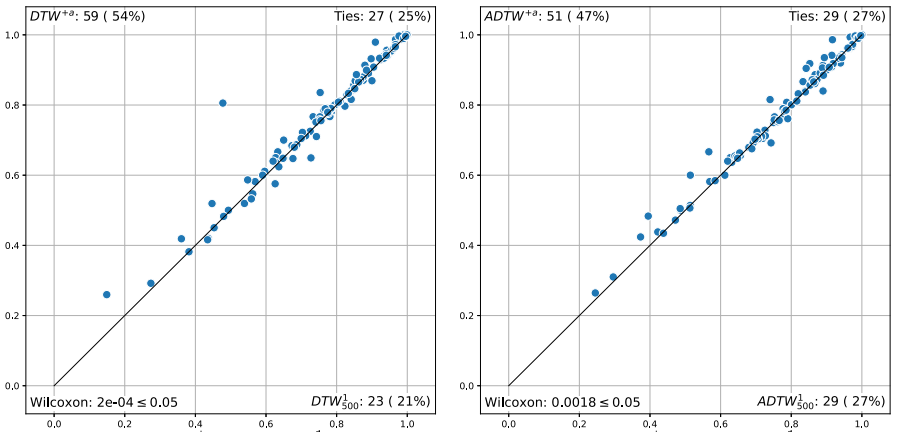
Fig. 6 Accuracy scatter plot over the UCR archive comparing DTW^{+a} against DTW^{1} and DTW^{2}



(a) $ADTW^{+a}$ vs. $ADTW^{1}$ (b) $ADTW^{+a}$ vs. $ADTW^{2}$

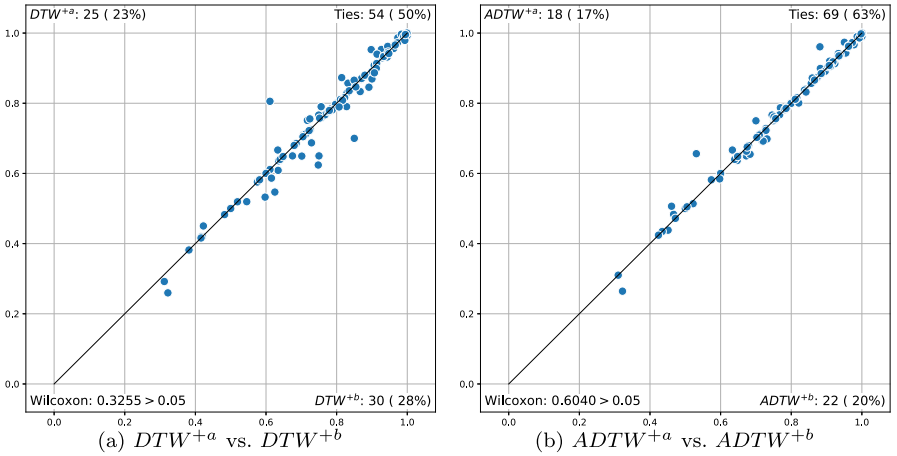
Fig. 7 Accuracy scatter plot over the UCR archive comparing $ADTW^{+a}$ against $ADTW^{1}$ and $ADTW^{2}$

w of *DTW* is a natural number for which the range of values that can result in different outcomes is $0 \leq w \leq \ell - 2$. In consequence, we cannot train *DTW* on more than $\ell - 1$ meaningfully different parameter values. This means that for short series ($\ell < 100$), increasing the number of possible windows from 100 to 500 has no effect. *ADTW* suffers less from this issue due to the penalty ω being sampled in a continuous space. Still, increasing the number of parameter values yields ever diminishing returns, while increasing the risk of overfitting. This also means that for a fixed budget of parameter values to be explored, tuning the cost function as well as w or ω allows the budget to be *spent* exploring a broader range of possibilities.



(a) DTW^{+a} vs DTW^{1}_{500} (b) $ADTW^{+a}$ vs $ADTW^{1}_{500}$

Fig. 8 Comparison of DTW^{+a} and $ADTW^{+a}$ trained over 500 different values (5 values for gamma, 100 values for w and ω per gamma), against DTW^{1}_{500} and $ADTW^{1}_{500}$ with 500 values for w and ω

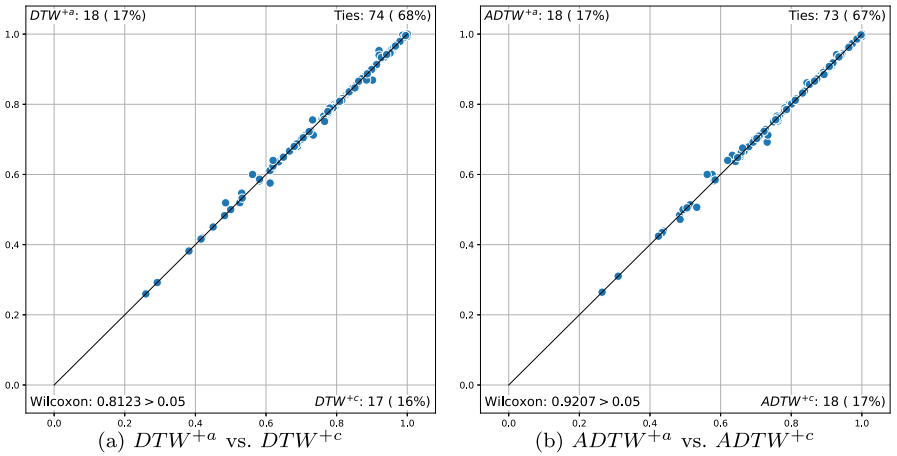


(a) DTW^{+a} vs. DTW^{+b} (b) $ADTW^{+a}$ vs. $ADTW^{+b}$

Fig. 9 Comparison of default exponent set a and *larger* set b

4.4 Comparison against larger tuning sets

Our experiments so far allow to achieve our primary goal: to demonstrate that tuning the cost function is beneficial. We did so with the set of exponents a. This set is not completely arbitrary (1 and 2 come from current practice, we added their mean 1.5 and the reciprocals). However, it remains an open question whether or not it is a reasonable default choice. Ideally, practitioners need to use expert knowledge to offer the best possible set of cost functions to choose from for a given application. In particular, using an alternative form of cost function to λ_γ could be effective, although we do not investigate this possibility in this paper.



(a) DTW^{+a} vs. DTW^{+c} (b) $ADTW^{+a}$ vs. $ADTW^{+c}$

Fig. 10 Comparison of default exponent set a and *denser* set c

Figure 9 shows the results obtained when using the larger set b, made of 11 values extending a with 3, 4, 5 and their reciprocals. Compared to a, the change benefits DTW^{+} (albeit not significantly according to the Wilcoxon test), at the cost of more than doubling the number of assessed parameter values. On the other hand, $ADTW^{+}$ is mostly unaffected by the change.

Figure 10 shows the results obtained when using the denser set c, made of 9 values between 0.5 and 2. In this case, neither distance benefits from the change.

4.5 Runtime

There is usually a tradeoff between runtime and accuracy for a practical machine learning algorithm. Sections 4.2 and 4.3 show that tuning the cost function significantly improves the accuracy of both *ADTW* and *DTW* in nearest neighbor classification tasks. However, this comes at the cost of having more parameters (500 instead of 100 with a single exponent). TSC using the nearest neighbor algorithm paired with $O(L^2)$ complexity elastic distances are well-known to be computationally expensive, taking hours to days to train (Tan et al. 2021b). Therefore, we discuss in this section, the computational details of tuning the cost function γ and assess the tradeoff in accuracy gain.

We performed a runtime analysis by recording the total time taken to train and test both *DTW* and *ADTW* for each γ from the default set a. Our experiments were coded in C++ and parallelised on a machine with 32 cores and AMD EPYC-Rome 2.2 Ghz Processor for speed. The C++ `pow` function that supports exponentiation of arbitrary values is computationally demanding. Hence, we use specialized code to calculate the exponents 0.5, 1.0 and 2.0 efficiently, using `sqrt` for 0.5, `abs` for 1.0 and multiplication for 2.0.

Figure 11 shows the LOOCV training time for both *ADTW* and *DTW* on each γ, while Fig. 12 shows the test time. The runtimes for $\gamma = 0.67$ and $\gamma = 1.5$ are both

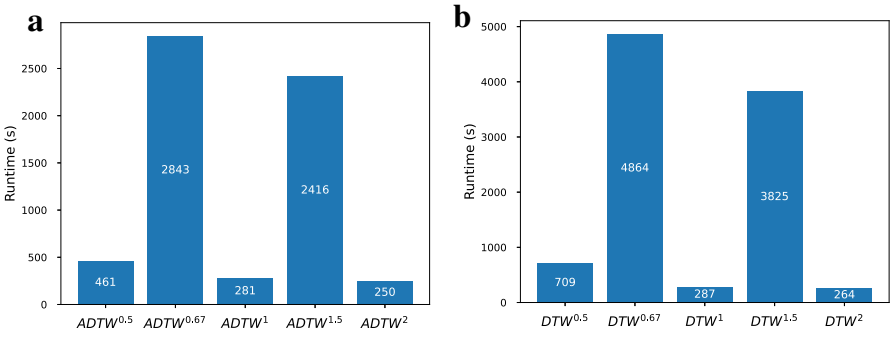


Fig. 11 LOOCV train time in seconds on the UCR Archive (109 datasets) of each distance, per exponent. These timings are done on a machine with 32 cores and AMD EPYC-Rome 2.2 Ghz Processor

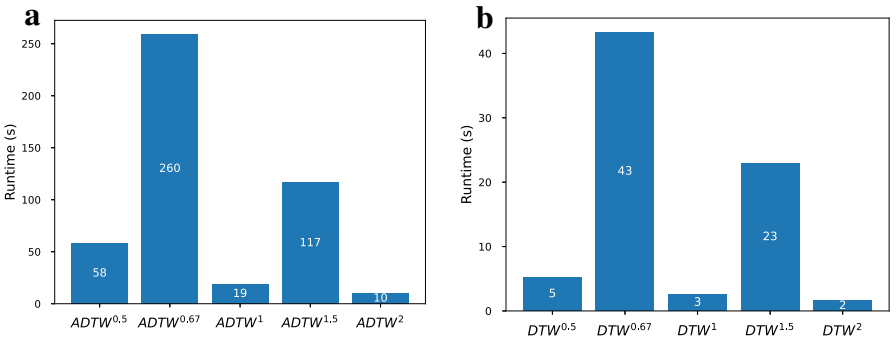


Fig. 12 Test time in seconds on the UCR Archive (109 datasets) of each distance, per exponent. These timings are done on a machine with 32 cores and AMD EPYC-Rome 2.2 Ghz Processor

substantially longer than those of the specialized exponents. The total time to tune the cost function and their parameters on 109 UCR time series datasets are 6250.94 (2 h) and 9948.98 (3 h) seconds for *ADTW* and *DTW* respectively. This translates to $ADTW^{+}$ and DTW^{+} being approximately 25 and 38 times slower than the baseline setting with $\gamma = 2$. Potential strategies for reducing these substantial computational burdens are to only use exponents that admit efficient computation, such as powers of 2 and their reciprocals. Also, the parameter tuning for w and ω in these experiment does not exploit the substantial speedups of recent *DTW* parameter search methods (Tan et al. 2021b). Despite being slower than both distances at $\gamma = 2$, completing the training of all 109 datasets under 3 h is still significantly faster than many other TSC algorithms (Tan et al. 2022; Middlehurst et al. 2021)

4.6 Noise

As γ alters *DTW*'s relative responsiveness to different magnitudes of effect in a pair of series, it is credible that tuning it may be helpful when the series are noisy. On one hand, higher values of γ will help focus on large magnitude effects, allowing *DTW*

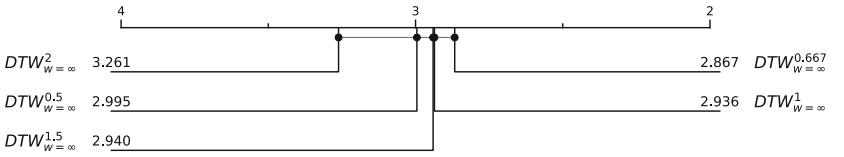


Fig. 13 Critical difference diagram for $DTW_{w=\infty}$ on the UCR Archive (109 datasets) with no additional noise

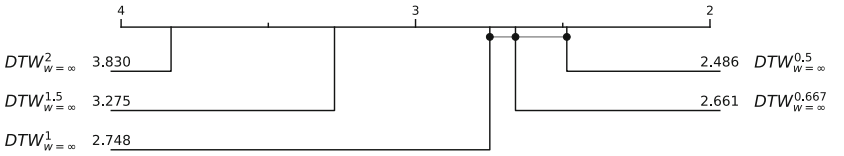


Fig. 14 Critical difference diagram for $DTW_{w=\infty}$ on the UCR Archive (109 datasets) with moderate additional noise

to pay less attention to smaller magnitude effects introduced by noise. On the other hand, lower values of γ will increase focus on small magnitude effects introduced by noise, increasing the ability of DTW^{γ} to penalize long warping paths that align sets of similar values.

To examine these questions we created two variants of each of the UCR datasets. For the first dataset we added moderate random noise, adding $0.1 \times \mathcal{N}(0, \sigma)$ to each time step, where σ is the standard deviation in the values in the series. For the second dataset (substantial noise) we added $\mathcal{N}(0, \sigma)$ to each time step.

The results for $DTW^{\gamma}_{w=\infty}$ (DTW with no window) are presented in Fig. 13 (no additional noise), Fig. 14 (moderate additional noise) and Fig. 15 (substantial additional noise). Each figure presents a critical difference diagram. DTW^{γ} has been applied with all 109 datasets at each $\gamma \in a$. For each dataset, the performance for each γ is ranked in descending order on accuracy. The diagram presents the mean rank for each DTW^{γ} across all datasets, with the best mean rank listed rightmost. Lines connect results that are not significantly different at the 0.05 level on a Wilcoxon singed rank test (for each line, the settings indicated with dots are not significantly different). With no additional noise, no setting of γ significantly outperforms the others. With a moderate amount of noise, the three lower values of γ significantly outperform the higher values. We hypothesize that this is as a result of DTW using the small differences introduced by noise to penalize excessively long warping paths. With high noise, the three lowest γ are still significantly outperforming the highest level, but the difference in ranks is closing. We hypothesize that this is due to increasingly large differences in value being the only ones that remain meaningful, and hence increasingly needing to be emphasized.

The results for $ADTW^{\gamma}$ are presented in Fig. 16 (no additional noise), Fig. 17 (moderate additional noise) and Fig. 18 (substantial additional noise). With no additional noise, γ values of 1.5 and 1.0 both significantly outperform 0.5. With a moderate amount of noise, $\gamma = 2.0$ increases its rank and no value significantly outperforms any other. With substantial noise, the two highest γ significantly outperform all others.

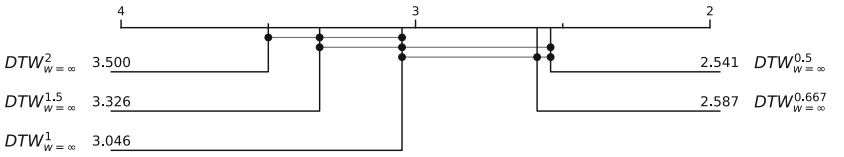


Fig. 15 Critical difference diagram for $DTW_{w=\infty}$ on the UCR Archive (109 datasets) with substantial additional noise

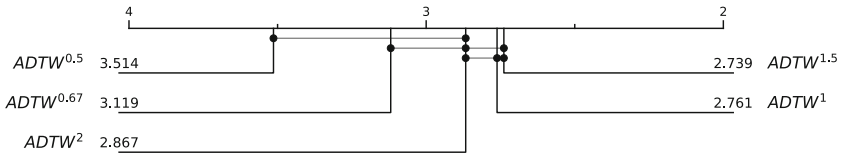


Fig. 16 Critical difference diagram for *ADTW* on the UCR Archive (109 datasets) with no additional noise

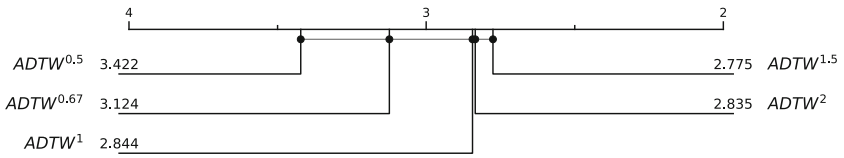


Fig. 17 Critical difference diagram for *ADTW* on the UCR Archive (109 datasets) with moderate additional noise

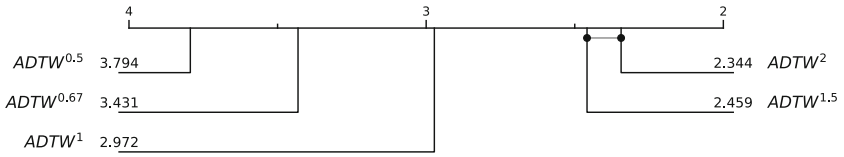


Fig. 18 Critical difference diagram for *ADTW* on the UCR Archive (109 datasets) with substantial additional noise

As *ADTW* has a direct penalty for longer paths, we hypothesize that this gain in rank for the highest γ is due to *ADTW* placing higher emphasis on larger differences that are less likely to be the result of noise.

The results for *DTW* with window tuning are presented in Fig. 19 (no additional noise), Fig. 20 (moderate additional noise) and Fig. 21 (substantial additional noise). No setting of γ has a significant advantage over any other at any level of noise. We hypothesize that this is because the constraint a window places on how far a warping path can deviate from the diagonal only partially restricts path length, allowing any amount of warping within the window. Thus, *DTW* still benefits from the use of low γ to penalize excessive path warping that might otherwise fit noise. However, it is also subject to a countervailing pressure towards higher values of γ in order to focus on larger differences in values that are less likely to be the result of noise.

It is evident from these results that γ interacts in different ways with the w and ω parameters of *DTW* and *ADTW* with respect to noise. For *ADTW*, larger values of γ are an effective mechanism to counter noisy series.

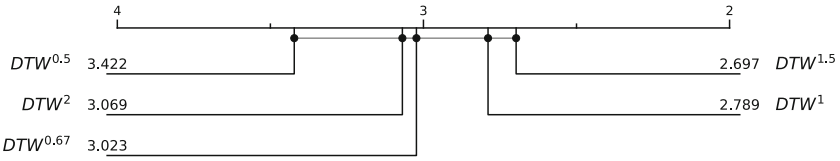


Fig. 19 Critical difference diagram for *DTW* on the UCR Archive (109 datasets) with no additional noise

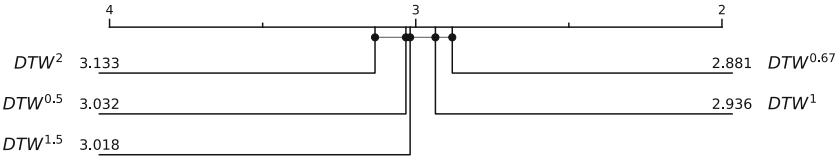


Fig. 20 Critical difference diagram for *DTW* on the UCR Archive (109 datasets) with lomoderate additional noise

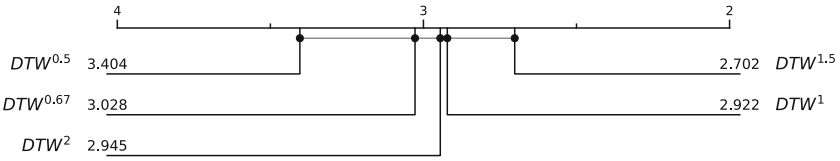


Fig. 21 Critical difference diagram for *DTW* on the UCR Archive (109 datasets) with substantial additional noise

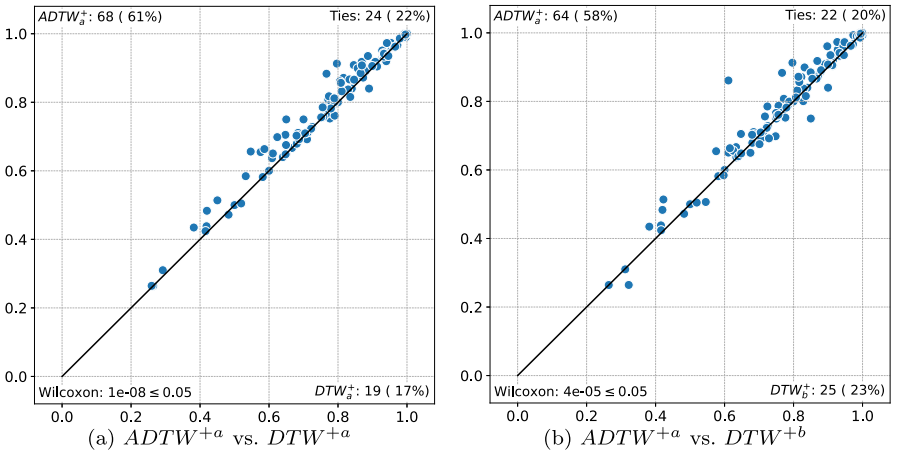
4.7 Comparing DTW^+ versus $ADTW^+$

From Herrmann and Webb (in press), $ADTW^2$ is more accurate than DTW^2. Figure 22 shows that $ADTW^{+a}$ is also significantly more accurate than DTW^{+a}. Interestingly, it also shows that $ADTW^{+a}$ is also more accurate than DTW^{+b}, even though the latter benefits from the larger exponent set b.

4.8 Comparing PF versus PF^+

Proximity Forest (*PF*) (Lucas et al. 2019) is an ensemble classifier relying on the same 11 distances as the Elastic Ensemble (EE) (Lines and Bagnall 2015), with the same parameter spaces. Instead of using LOOCV to optimise each distance and ensemble their result, PF builds trees of proximity classifiers, randomly choosing an exemplar, a distance and a parameter at each node. This strategy makes it both more accurate and more efficient than EE and the most accurate similarity-based time series classifier on the UCR benchmark.

Proximity Forest and the Elastic Ensenble use the following distances: the (squared) Euclidean distance (*SQED*); *DTW* with and without a window; *DDTW* adding the derivative to *DTW* (Keogh and Pazzani 2001); *WDTW* (Jeong et al. 2011); *DWDTW* adding the derivative to *WDTW*; *LCSS* (Hirschberg 1977); *ERP* (Chen and Ng 2004); *MSM* (Stefan et al. 2013); and *TWE* (Marteau 2009).



(a) $ADTW^{+a}$ vs. DTW^{+a} (b) $ADTW^{+a}$ vs. DTW^{+b}

Fig. 22 Accuracy scatter plot over the UCR archive comparing $ADTW^{+a}$ against DTW^{+} tuned over a and b

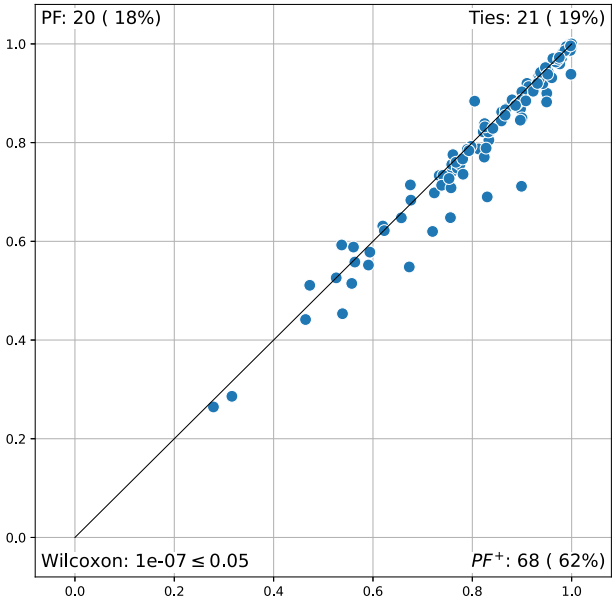


Fig. 23 Accuracy scatter plot over the UCR archive comparing the original Proximity Forest (*PF*) against Proximity Forest using λ_γ for *DTW*, and its variants (PF^+)

We define a new variant of Proximity Forest, PF^+, which differs only in replacing original cost functions for *DTW* and its variants by our proposed parameterized cost function. We replace the cost function of *DTW* (with and without window), *WDTW*, *DDTW*, *DWDTW* and *SQED* by λ_γ, and randomly select γ from the a set at each node. Note that the replacing the cost function of *SQED* in this manner makes it similar to a Minkowski distance.

Table 1 Six benchmark UCR datasets for which PF+ is more accurate than all four algorithms that have been identified as defining the current state of the art in TSC

Dataset	PF^+	HC2	TS-C	MR	IT
ArrowHead	0.8971	0.8629	0.8057	0.8629	0.8629
Earthquakes	0.7698	0.7482	0.7482	0.7482	0.7410
Lightning2	0.8689	0.7869	0.8361	0.6885	0.8197
SemgHandGenderCh2	0.9683	0.9567	0.9233	0.9583	0.8700
SemgHandMovementCh2	0.8800	0.8556	0.8778	0.7756	0.5689
SemgHandSubjectCh2	0.9311	0.9022	0.9244	0.9244	0.7644

We leave the tuning of other distances and their specific cost functions for future work. This is not a technical limitation, but a theoretical one: we first have to ensure that such a change would not break their properties.

The scatter plot presented in Fig. 23 shows that PF^+ significantly outperforms PF, further demonstrating the value of extending the range of possible parameters to the cost function.

While similarity-based approaches no longer dominate performance across the majority of the UCR benchmark datasets, there remain some tasks for which similarity-based approaches still dominate. Table 1 shows the accuracy of PF^+ against four TSC algorithms that have been identified (Middlehurst et al. 2021) as defining the state of the art—HIVE-COTE 2.0 (Middlehurst et al. 2021), TS-CHIEF (Shifaz et al. 2020), MultiRocket (Tan et al. 2022) and InceptionTime (Fawaz et al. 2020). This demonstrates that similarity-based methods remain an important part of the TSC toolkit.

5 Conclusion

DTW is a widely used time series distance measure. It relies on a cost function to determine the relative weight to place on each difference between values for a possible alignment between a value in one series to a value in another. In this paper, we show that the choice of the cost function has substantial impact on nearest neighbor search tasks. We also show that the utility of a specific cost function is task-dependent, and hence that *DTW* can benefit from cost function tuning on a task to task basis.

We present a technique to tune the cost function by adjusting the γ exponent in a family of cost functions $\lambda_\gamma(a, b) = |a - b|^\gamma$. We introduced new time series distance measures utilizing this family of cost functions: DTW^+ and $ADTW^+$. Our analysis shows that larger γ exponents penalize alignments with large differences while smaller γ exponents penalize alignments with smaller differences, allowing the focus to be tuned between small and large amplitude effects in the series.

We demonstrated the usefulness of this technique in both the nearest neighbor and Proximity Forest classifiers. The new variant of Proximity Forest, PF^+, establishes a new benchmark for similarity-based TSC, and dominates all of HiveCote2, TS-Chief,

MultiRocket and InceptionTime on six of the UCR benchmark tasks, demonstrating that similarity-based methods remain a valuable alternative in some contexts.

We argue that cost function tuning can address noise through two mechanisms. Low exponents can exploit noise to penalize excessively long warping paths. It appears that *DTW* benefits from this when windowing is not used. High exponents direct focus to larger differences that are least affected by noise. It appears that *ADTW* benefits from this effect.

We need to stress that we only experimented with one family of cost function, on a limited set of exponents. Even though we obtained satisfactory results, we urge practitioners to apply expert knowledge when choosing their cost functions, or a set of cost functions to select from. Without such knowledge, we suggest what seems to be a reasonable default set of choices for DTW^{+} and $ADTW^{+}$, significantly improving the accuracy over *DTW* and *ADTW*. We show that a *denser* set does not substantially change the outcome, while *DTW* may benefit from a *larger* set that contains more extreme values of γ such as 0.2 and 5.

A small number of exponents, specifically 0.5, 1 and 2, lead themselves to much more efficient implementations than alternatives. It remains for future research to investigate the contexts in which the benefits of a wider range of exponents justify their computational costs.

We expect our findings to be broadly applicable to time series nearest neighbor search tasks. We believe that these finding also hold forth promise of benefit from greater consideration of cost functions in the myriad of other applications of *DTW* and its variants.

Acknowledgements This work was supported by the Australian Research Council award DP210100072. The authors would like to thank Professor Eamonn Keogh and his team at the University of California Riverside (UCR) for providing the UCR Archive.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Alaee S, Mercer R, Kamgar K, Keogh E (2021) Time series motifs discovery under DTW allows more robust discovery of conserved structure. Data Min Knowl Disc 35(3):863–910

Bandara K, Hewamalage H, Liu YH, Kang Y, Bergmeir C (2021) Improving the accuracy of global forecasting models using time series data augmentation. Pattern Recogn 120:108148
Cao Y, Rakhilin N, Gordon PH, Shen X, Kan EC (2016) A real-time spike classification method based on dynamic time warping for extracellular enteric neural recording with large waveform variability. J Neurosci Methods 261:97–109
Chen L, Ng R (2004) On the marriage of Lp-norms and edit distance. In: Proceedings 2004 VLDB conference, pp 792–803
Cheng H, Dai Z, Liu Z, Zhao Y (2016) An image-to-class dynamic time warping approach for both 3d static and trajectory hand gesture recognition. Pattern Recogn 55:137–147
Dau HA, Keogh E, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, Ratanamahatana CA, Yanping, Hu B, Begum N, Bagnall A, Mueen A, Batista G, Hexagon-ML (2018) The UCR time series classification archive
Dau HA, Bagnall A, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, Ratanamahatana CA, Keogh E (2019) The UCR time series archive. arXiv:1810.07758 [cs, stat]
Deng H, Chen W, Shen Q, Ma AJ, Yuen PC, Feng G (2020) Invariant subspace learning for time series data based on dynamic time warping distance. Pattern Recogn 102:107210. https://doi.org/10.1016/j.patcog.2020.107210
Deriso D, Boyd S (2022) A general optimization framework for dynamic time warping. Optim Eng. https://doi.org/10.1007/s11081-022-09738-z
Diab DM, AsSadhan B, Binsalleeh H, Lambotharan S, Kyriakopoulos KG, Ghafir I (2019) Anomaly detection using dynamic time warping. In: 2019 IEEE International conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC). IEEE, pp 193–198
Fawaz HI, Lucas B, Forestier G, Pelletier C, Schmidt DF, Weber J, Webb GI, Idoumghar L, Muller PA, Petitjean F (2020) Inceptiontime: finding alexnet for time series classification. Data Min Knowl Discov 34:1936–1962. https://doi.org/10.1007/s10618-020-00710-y
Herrmann M, Webb GI (in press) Amercing: an intuitive and effective constraint for dynamic time warping. Pattern Recogn
Hirschberg DS (1977) Algorithms for the longest common subsequence problem. J ACM (JACM) 24(4):664–675. https://doi.org/10.1145/322033.322044
Itakura F (1975) Minimum prediction residual principle applied to speech recognition. IEEE Trans Acoust Speech Signal Process 23(1):67–72. https://doi.org/10.1109/TASSP.1975.1162641
Jeong YS, Jeong MK, Omitaomu OA (2011) Weighted dynamic time warping for time series classification. Pattern Recogn 44(9):2231–2240. https://doi.org/10.1016/j.patcog.2010.09.022
Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7(3):358–386
Keogh EJ, Pazzani MJ (2001) Derivative dynamic time warping. In: Proceedings of the 2001 SIAM international conference on data mining, society for industrial and applied mathematics, pp 1–11. https://doi.org/10.1137/1.9781611972719.1
Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Disc 29(3):565–592. https://doi.org/10.1007/s10618-014-0361-2
Löning M, Bagnall A, Ganesh S, Kazakov V (2019) Sktime: a unified interface for machine learning with time series. arXiv:1909.07872
Lucas B, Shifaz A, Pelletier C, O'Neill L, Zaidi N, Goethals B, Petitjean F, Webb GI (2019) Proximity forest: an effective and scalable distance-based classifier for time series. Data Min Knowl Disc 33(3):607–635. https://doi.org/10.1007/s10618-019-00617-3
Marteau PF (2009) Time warp edit distance with stiffness adjustment for time series matching. IEEE Trans Pattern Anal Mach Intell 31(2):306–318. https://doi.org/10.1109/TPAMI.2008.76
Middlehurst M, Large J, Flynn M, Lines J, Bostrom A, Bagnall A (2021) HIVE-COTE 2.0: a new meta ensemble for time series classification. Mach Learn 110(11):3211–3243
Mueen A, Keogh E (2016) Extracting optimal performance from dynamic time warping. In: Proceedings of the 22nd ACM sigkdd international conference on knowledge discovery and data mining—KDD'16. ACM Press, pp 2129–2130. https://doi.org/10.1145/2939672.2945383
Okawa M (2021) Online signature verification using single-template matching with time-series averaging and gradient boosting. Pattern Recogn 112:107699
Petitjean F, Ketterlin A, Gançarski P (2011) A global averaging method for dynamic time warping, with applications to clustering. Pattern Recogn 44(3):678–693

Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: Proc. 18th ACM SIGKDD Int. Conf. knowledge discovery and data mining, pp 262–270

Ratanamahatana C, Keogh E (2004) Making time-series classification more accurate using learned constraints. In: SIAM SDM

Sakoe H, Chiba S (1971) Recognition of continuously spoken words based on time-normalization by dynamic programming. J Acoust Soc Jpn 27(9):483–490

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49. https://doi.org/10.1109/TASSP.1978.1163055

Shifaz A, Pelletier C, Petitjean F, Webb GI (2020) TS-CHIEF: a scalable and accurate forest algorithm for time series classification. Data Min Knowl Disc 34(3):742–775. https://doi.org/10.1007/s10618-020-00679-8

Silva DF, Giusti R, Keogh E, Batista GEAPA (2018) Speeding up similarity search under dynamic time warping by pruning unpromising alignments. Data Min Knowl Disc 32(4):988–1016. https://doi.org/10.1007/s10618-018-0557-y

Singh G, Bansal D, Sofat S, Aggarwal N (2017) Smart patrolling: an efficient road surface monitoring using smartphone sensors and crowdsourcing. Pervasive Mob Comput 40:71–88

Stefan A, Athitsos V, Das G (2013) The move-split-merge metric for time series. IEEE Trans Knowl Data Eng 25(6):1425–1438. https://doi.org/10.1109/TKDE.2012.88

Tan CW, Herrmann M, Forestier G, Webb GI, Petitjean F (2018) Efficient search of the best warping window for dynamic time warping. In: Proc. 2018 SIAM Int. Conf. data mining. SIAM, pp 225–233

Tan CW, Petitjean F, Webb GI (2020) FastEE: fast ensembles of elastic distances for time series classification. Data Min Knowl Disc 34(1):231–272. https://doi.org/10.1007/s10618-019-00663-x

Tan CW, Bergmeir C, Petitjean F, Webb GI (2021a) Time series extrinsic regression. Data Min Knowl Disc 35(3):1032–1060. https://doi.org/10.1007/s10618-021-00745-9

Tan CW, Herrmann M, Webb GI (2021b) Ultra fast warping window optimization for dynamic time warping. In: 2021 IEEE international conference on data mining. IEEE, pp 589–598

Tan CW, Dempster A, Bergmeir C, Webb GI (2022) Multirocket: multiple pooling operators and transformations for fast and effective time series classification. Data Min Knowl Disc 36(5):1623–1646

Thompson AC, Thompson AC (1996) Minkowski geometry. Cambridge University Press, Cambridge

Varatharajan R, Manogaran G, Priyan MK, Sundarasekar R (2018) Wearable sensor devices for early detection of Alzheimer disease using dynamic time warping algorithm. Clust Comput 21(1):681–690

Yasseen Z, Verroust-Blondet A, Nasri A (2016) Shape matching by part alignment using extended chordal axis transform. Pattern Recogn 57:115–135

Zhao J, Itti L (2018) shapeDTW: Shape dynamic time warping. Pattern Recogn 74:171–184. https://doi.org/10.1016/j.patcog.2017.09.020