appeared in: Proc. SIAM Int. Conf. on Data Mining 2014, Philadelphia, USA, pp 560-568

Less is More: Similarity of Time Series under Linear Transformations

Frank Höppner

Ostfalia University of Applied Sciences, Germany

f.hoeppner@ostfalia.de

Abstract

When comparing time series, z-normalization preprocessing and dynamic time warping (DTW) distance became almost standard procedure. This paper makes a point against carelessly using this setup by discussing implications and alternatives. A (conceptually) simpler distance measure is proposed that allows for a linear transformation of amplitude and time only, but is also open for other normalizations (unachievable by z-normalization preprocessing). Lower bounding techniques are presented for this measure that apply directly to raw series.

1 Introduction

When recording time series, we need two measurement devices, one for the time and one for the value of interest, such as temperature, velocity or distance. Suppose two people observe the *same* process and we compare their observations, what are the differences we will most likely face?

- offset: The devices may not be properly calibrated: the calibration of a thermometer based on the boiling point of water varies with pressure. If the calibration was done under different conditions, we will observe an offset. For the temporal domain, if observers do not start their stop watches at the same time, there will be a constant temporal offset between both series.
- scale: If distance is measured by counting rotations and multiplying the count by the radius of the wheel, a variation in the radius (tire inflation pressure) leads to a different scaling factor of distance. The same applies to the temporal scale if the digital clocks do not have identical quartz crystals.
- noise: As no measurement is exact, we expect some random observational error (typically more pronounced for amplitude than for time).

Of course, more intriguing influences are possible, for instance, in a long term observation the daily temperature and air pressure may influence the measurements and lead to locally fluctuating errors. But the above-mentioned effects, which represent a linear transformation of any of the two domains, are so likely and elementary that we should deal with them first before seeking for more complex explanations for differences in time series. How does the time series mining literature cope with these likely effects?

Time: The problem of a temporal offset is often solved by aligning the time series during preprocessing. This may be done by rather simplistic (e.g., removal of all leading values close to zero) or by highly complex algorithms (e.g. [9]). A second frequently applied approach is to use dynamic time warping (DTW) measures that allow for (almost) arbitrary stretching and squeezing of the temporal axis to achieve a better fit of the series. (We will review DTW later.) Sometimes uniform scaling of time is considered (e.g. [6]), but no temporal offset, so we still need to align the series first.

DTW requires many more parameters (the warping path) than a linear temporal transformation (only offset and scale), so recovering a linear transformation should be more robust. It has also been reported that uniform scaling performs superior for applications such as motion capturing, gait recognition or query-by-humming [6, 7, 4]. Yet, for the general case, linearly transformed time goes almost unconsidered: in [3] (published after work on uniform scaling [6, 7, 4]) a broad comparison of time series similarity measures is carried out, but the list of distance measures does not contain a measure dedicated to the likely setting of just linear transformations.

Amplitude: To cope with differences in amplitude, the most common approach of z-normalization is typically part of the preprocessing: time series are shifted to have zero mean and re-scaled to unit variance. Is the common z-normalization the right answer to all effects mentioned above? Figure 1(a) depicts series, where the classes (red/blue) are distinguished by the absolute slope of the series: in meteorology a change in air pressure by more than 1 hPa per hour is highly relevant for stormy weather, but the exact air pressure is meaningless. For case (b) all data is measured by

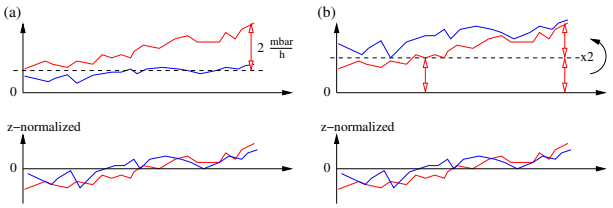

Figure 1: Z-Normalization destroys potentially relevant properties to distinguish the red from the blue series.

the same device (or we have a unique zero point), so the absolute values or ratios may become meaningful: in a financial market application, we may (regardless of the price level) look for situations where the price level doubles. In both cases, after z-normalization (bottom row), we have lost all means to distinguish both cases from each other. These examples show that it may not be advisable to consider z-normalization as the *one and only* normalization step, because scale and shift may carry relevant information, too.

Noise: Despite the presence of noise, most of the literature on comparing similarity measures (e.g. [3]) does not consider, say, smoothing filters, but apply the distance measures directly to (z-normalized) series.

The contributions of this paper are:

- Raise of awareness how preprocessing affects distance and that it should always be adapted to the applications need.
- Definition of a time series similarity measure that respects a linear transformation in both, the longitudinal (temporal) and transversal (vertical) axis, because this appears to be the most likely setting.
- Meaningful restrictions of the linear transformations are motivated and compared to standard z-normalization.
- New lower bounds for this measure (includes normalization step that is part of the distance).
- Experimental evaluation that supports the claim of linear transformations being the most likely setting, shows the benefit of different normalizations, and provides insights in the ambiguous role of DTW's warping flexibility.

2 Background / Related Work

A time series $\mathbf{x}$ of length n is an ordered sequence of real numbers: $\mathbf{x} = (x_i)_{1\le i\le n} \in \mathbb{R}^n$, which is also written as $|\mathbf{x}| = n$. By $\mathbf{x}_{..i}$ we denote a prefix of $\mathbf{x}$ of length $i \le n$: $\mathbf{x}_{..i} = (x_1, \ldots, x_i)$. By $s \cdot \mathbf{x} + m$ for $s, m \in \mathbb{R}$ we denote scalar operations, that is, $s \cdot \mathbf{x} + m = (s \cdot x_i + m)_{1\le i\le n}$.

Among the most frequently used time series similarity measures for two time series $\mathbf{x}$ and $\mathbf{y}$ is the Euclidean distance[1], which is defined for series of equal length n:

$$ED(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$$

ED performs surprisingly well in 1-Nearest Neighbour classification for a range of datasets [3] (although the reasons for this are somewhat subtle).

We can redefine the squared Euclidean distance recursively as $ED(\mathbf{x}_{..i}, \mathbf{y}_{..i}) = ED(\mathbf{x}_{..i-1}, \mathbf{y}_{..i-1}) + (x_i - y_i)^2$, that is, we obtain the total distance from the distance between the $(i-1)$-prefixes of $\mathbf{x}$ and $\mathbf{y}$ and the distance of the last two remaining elements; the latter term corresponds to an alignment of the last elements to each other. With dynamic time warping (DTW) this alignment is more flexible as elements from a series may be skipped, if that leads to smaller costs:

$$\begin{aligned} &DTW(\mathbf{x}_{..i}, \mathbf{y}_{..j}) = (x_i - y_j)^2 + \\ &\min\{DTW(\mathbf{x}_{..i-1}, \mathbf{y}_{..j}), \\ &\qquad DTW(\mathbf{x}_{..i-1}, \mathbf{y}_{..j-1}), DTW(\mathbf{x}_{..i}, \mathbf{y}_{j-1})\} \end{aligned}$$

This scheme allows for a flexible alignment of indices i and j and is solved by dynamic programming in $O(n^2)$ [2]. Many different flavours of DTW have been proposed, see [3] for an overview.

The idea of considering only linear transformation is not really new. In [7] uniform scaling (US) of a query $\mathbf{y}$ (of length m) to a new length p is considered (but no offset), to better match a reference series $\mathbf{x}$ (of length n) under the Euclidean distance:

$$US(\mathbf{x}_{..n}, \mathbf{y}_{..m}) = ED((x_i)_{1\le i\le p}, (y_{i/m\cdot p})_{1\le i\le n})$$

In [4] a combination of US and DTW is proposed: the series is stretched first before then DTW is applied. In [1] linear transformations are considered, but with a focus on finding short series within longer series.

A technique for bounding the expected outcome of, say, uniform scaling is to determine the upper and/or lower *envelope* around a series [6, 4]: for any point x_i one can restrict the set A of possible indices j such that x_i is eventually aligned with an y_j where $j \in A$. (The more the range of scaling factors is limited, the fewer indices are in A.) Thus, x_i will be matched to a value y_j lying within some bounds $l_i \le y_j \le u_i$ (with $l_i = \min_{j\in A} y_j$ and $u_i = \max_{j\in A} y_j$). The series of l_i and u_i are the lower and upper envelope, resp. These envelopes can be used to find a lower bound for the distance measure

[1] Actually, we use the squared Euclidean distance whenever referring to the Euclidean distance. As the measure will be used to order time series according to their similarity, the application of a strictly increasing function like the square root can safely be skipped without affecting the order.

without having to know the finally chosen alignment: As long as the value of x_i falls within the envelope we may luckily find a j with $x_i = y_j$ (so we assume no contribution to the distance). Only for values x_i outside the envelope we know for sure that the distance is increased (regardless of the alignment):

$$DTW(\mathbf{x},\mathbf{y}) \geq \sum_{i=1}^{n} \begin{cases} (x_i - u_i)^2 & \text{if } x_i > u_i \\ (x_i - l_i)^2 & \text{if } x_i < l_i \\ 0 & \text{else} \end{cases}$$

A typical application is 1-Nearest-Neighbour classification: Once a query has been compared to a series and has achieved a low distance d, we are only interested in the exact distance to yet another series if it drops below d – otherwise we may skip the calculation early.

3 Linear Transformation Distance

3.1 Vertical Scale and Offset As already mentioned, it is common to z-normalize time series prior to the application of a distance measure to account for variations in amplitude. As a result of this standardization ($\bar{\mathbf{x}} = 0$, $\|\mathbf{x}\| = 1$), the Euclidean distance becomes the Pearson's distance

$$ED(\mathbf{x},\mathbf{y}) = 2\left(1 - \sum_{i=1}^{n} x_i y_i\right) = 2 - 2\rho_{\mathbf{x},\mathbf{y}}$$

where $\rho_{\mathbf{x},\mathbf{y}}$ is the correlation coefficient of both series.

Let us address normalization from a different perspective now. An alternative approach is to ask for the Euclidean distance under the best possible amplitude scale $s \in \mathbb{R}$ and mean shift $m \in \mathbb{R}$ of one of the series:

$$RD_a(\mathbf{x},\mathbf{y}) = \min_{s,m\in\mathbb{R}} ED(s\cdot\mathbf{x} + m, \mathbf{y}) \tag{3.1}$$

This is equivalent to a regression task where we try to predict the values of series $\mathbf{y}$ from series $\mathbf{x}$ by means of a linear regression $y_i = s \cdot x_i + m$, hence we name it **regression distance** (RD). From linear regression we know that (3.1) becomes minimal for

$$s = \frac{\sum x_i y_i - \frac{1}{n}\sum x_i \sum y_i}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2} \tag{3.2}$$

$$m = \frac{1}{n}\left(\sum y_i - s\sum x_i\right) \tag{3.3}$$

Note that RD_a is asymmetric: If the range of series $\mathbf{x}$ is [0,1000] and the range of $\mathbf{y}$ is [0,1], we would either scale $\mathbf{x}$ down to [0,1] or $\mathbf{y}$ up to [0,1000]. The sum of (squared) differences will obviously be affected by the scale. In application such as 1-NN classification this is not even a problem, because all training series are compared against a single test series. In general, to overcome this asymmetry, we define RD as:

$$RD(\mathbf{x},\mathbf{y}) = \min\{RD_a(\mathbf{x},\mathbf{y}), RD_a(\mathbf{y},\mathbf{x})\} \tag{3.4}$$

If we stick to preprocessed data, that is, time series $\mathbf{x}$ and $\mathbf{y}$ with unit variance and zero mean, is there any difference between $ED(\mathbf{x},\mathbf{y})$ and $RD(\mathbf{x},\mathbf{y})$? There is an important difference: Remember that $ED = 2 - 2\rho_{\mathbf{x},\mathbf{y}}$, that is, for ED uncorrelated time series are more similar ($\rho = 0, ED = 2$) than perfectly (negatively) correlated time series ($ED = 4$). With RD, on the other hand, a perfect negative correlation means that we can perfectly predict $\mathbf{y}$ from $\mathbf{x}$, so the sum of squared error becomes small ($RD \approx 0$). With RD uncorrelated time series receive the largest distance. Would such a measure be useful? When looking for similar series, it is as suprising to see the same pattern inverted ($\rho = -1$) as to see a copy of the original series. So, yes, there might be applications where such a measure may actually be very useful (cf. Sect. 6.3). On the other hand, we also believe that the application dictates what is similar and what is not. So in case negatively correlated subsequences are not considered as similar, we can define a constrained regression distance. Let us denote the set of admissible regression parameters (s, m) by Ω_R. So far we had $\Omega_R = \mathbb{R}^2$, now we consider $\Omega_R = \mathbb{R}_{\geq 0} \times \mathbb{R}$ where only non-negative amplitude scaling factors are allowed. Now, (3.1) becomes minimal for

$$s = \max\left\{0, \frac{\sum x_i y_i - \frac{1}{n}\sum x_i \sum y_i}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2}\right\} \tag{3.5}$$

$$m = \frac{1}{n}\left(\sum y_i - s\sum x_i\right) \tag{3.6}$$

(Proof omitted.) For negatively correlated series the solution at $s = 0$ is optimal: a flat line achieves a smaller distance than any other scaling factor $s > 0$.

Apart from the two variants already discussed, we may also define other meaningful sets of admissible values: $\Omega_R = \{1\} \times \mathbb{R}$ for mean shift only, $\Omega_R = \mathbb{R} \times \{0\}$ for amplitude scaling only (and with $\Omega_R = \{1\} \times \{0\}$ RD collapses to ED). It is *important* to note that equivalent distances *cannot be achieved by preprocessing alone*: With $\Omega_R = \mathbb{R} \times \{0\}$ and series $\mathbf{x} = (1,2,1,1,1,1)$, $\mathbf{y} = (1,2,3,2,1,1)$ and $\mathbf{z} = (1,1,2,3,2,1)$ we obtain different optimal scaling factors s when comparing $\mathbf{x}$ to $\mathbf{y}$ or to $\mathbf{z}$ – even after transforming all series to unit variance. Rescaling series only once during preprocessing would thus either overestimate $d(\mathbf{x},\mathbf{y})$ or $d(\mathbf{x},\mathbf{z})$.

3.2 Temporal Scale and Offset Next, we address scale and shift in the temporal domain. Suppose a function $f(i)$ has been sampled at integer time points $1 \leq i \leq n$ leading to time series $y_i = f(i)$. We introduce

parameters $w \in \mathbb{R}$ for time warp (dilation) and $o \in \mathbb{N}$ for temporal offset (translation), which would lead us to time series $y'_i = f(w \cdot i + o)$. When comparing with a reference series $\mathbf{x}$, we may again seek for the best match under the admissible values for w and o. As the function f is typically not available, we have to use the given samples y_i to approximate $y'_i = y_{w \cdot i + o}$ (with $w \cdot i + o$ rounded to closest integer). Thus we consider

$$ED(\mathbf{x}, (y_{w \cdot i + o})_{1 \le i \le n}) \tag{3.7}$$

where we have to take care that the index range of the sampled series $\mathbf{y}$ is not exceeded, that is, $1 \le w \cdot i + o \le |y|$ for $1 \le i \le |x|$. By I_{wo} we denote the set of (rounded integer) indices we obtain with w and o:

$$I_{wo} := \{\lfloor w \cdot i + o + 1/2 \rfloor \mid i = 1..|x|\}$$

Furthermore, we want to avoid spurious matches, say, matching the full series $\mathbf{x}$ to only two points of $\mathbf{y}$. There should be a considerable portion of both time series involved in the comparison. To match a series $\mathbf{x}$ to a series $\mathbf{y}$ of length n using (3.7), we require that at least $\delta \cdot n$ points of $\mathbf{y}$ must be involved in the comparison. We define the set of admissible values $\Omega_T \subseteq \mathbb{R}^2$ as

$$\Omega_T := \{(w, o) \mid I_{wo} \subseteq [1, |\mathbf{y}|] \ \wedge \ |I_{wo}| \ge \delta |\mathbf{y}| \} \tag{3.8}$$

(with $|\mathbf{y}|$ being the length of $\mathbf{y}$) and the **translation / dilation distance** as the Euclidean distance under the best linear transformation of time:

$$TD_a(\mathbf{x}, \mathbf{y}) = \min_{(w,o) \in \Omega_T} ED(\mathbf{x}, (y_{w \cdot i + o})_{1 \le i \le |x|}) \tag{3.9}$$

Again, this measure is asymmetric, because $\mathbf{x}$ is mapped to a (stretched) subsequence of $\mathbf{y}$ but not the other way round (cf. Fig. 2). We establish symmetry by considering the case of embedding $\mathbf{y}$ into $\mathbf{x}$, too:

$$TD(\mathbf{x}, \mathbf{y}) = \min\{TD_a(\mathbf{x}, \mathbf{y}), TD_a(\mathbf{y}, \mathbf{x})\} \tag{3.10}$$

Unfortunately, there is no closed-form solution to determine the optimal w and o.

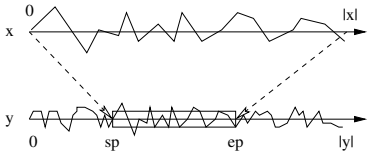

Figure 2: $TD_a(\mathbf{x}, \mathbf{y})$: embedding $\mathbf{x}$ into $\mathbf{y}$.

3.3 Linear Transformation Distance Finally, the distance measure we are interested in, combines RD and TD in one measure. We define the **linear transformation distance** $LTD_a(\mathbf{x}, \mathbf{y}) =$

$$\min_{(w,o) \in \Omega_T, (s,m) \in \Omega_R} ED\left(s \cdot \mathbf{x} + m, (y_{w \cdot i + o})_{1 \le i \le n}\right)$$

That is, we allow for a longitudinal shift and scaling of series $\mathbf{y}$ and amplitude scaling and offset of the series $\mathbf{x}$ such that they fit best under the Euclidean distance. Again, $LTD(\mathbf{x}, \mathbf{y}) := \min\{LTD_a(\mathbf{x}, \mathbf{y}), LTD_a(\mathbf{y}, \mathbf{x})\}$.

4 Direct Calculation of LTD

A straightforward implementation of LTD_a is given by algorithm 4.1. It picks subsequences of $\mathbf{y}$ by choosing start- and end-points that respect the δ-constraint of (3.8). For each setting the optimal parameters (s, m) are determined (according to the chosen Ω_R, e.g. (3.2)-(3.3) or (3.5)-(3.6)).

ALGORITHM 4.1. Calculation of $LTD_a(\mathbf{x}, \mathbf{y})$

```
overlap = δ · |x|, best = ∞
for (sp in 1..(|y|-overlap))                    //map x_1 → y_sp
  for (ep in (sp+overlap)..|y|)                 //map x_|x| → y_ep
    w = (ep − sp)/|x|
    for (i in 1..|x|) y'_i = y_{w·i+sp}         // transf. time
    determine s, m  // e.g. (3.2)-(3.3) or (3.5)-(3.6)
    for (i in 1..|x|) x'_i = s · x'_i + m       // transf. ampl.
    if ED(x', y') < best
      best = ED(x', y')
return best
```

The algorithm appears quite expensive due to three nested loops ($O((1-\delta)^2 \cdot n^3)$ assuming $|x| = |y| = n$). In comparison with DTW the difficulty is that we cannot utilize dynamic programming because we lack a recursive definition of LTD. For $n = m = 10$, $\delta = 0.5$ we obtain 15 admissible parameters (w, o) that lead to the following 15 alignments (one per row):

alignment of $x_i \to y_j$ with i:j shown below	w	o
1:1 2:2 3:2 4:3 5:3 6:4 7:4 8:5 9:5 10:6	0.5	1
1:1 2:2 3:2 4:3 5:4 6:4 7:5 8:5 9:6 10:7	0.6	1
1:1 2:2 3:3 4:3 5:4 6:5 7:5 8:6 9:7 10:8	0.7	1
1:1 2:2 3:3 4:4 5:5 6:5 7:6 8:7 9:8 10:9	0.8	1
1:1 2:2 3:3 4:4 5:5 6:6 7:7 8:8 9:9 10:10	0.9	1
1:2 2:3 3:3 4:4 5:4 6:5 7:5 8:6 9:6 10:7	0.5	2
1:2 2:3 3:3 4:4 5:5 6:5 7:6 8:6 9:7 10:8	0.6	2
1:2 2:3 3:4 4:4 5:5 6:6 7:6 8:7 9:8 10:9	0.7	2
1:2 2:3 3:4 4:5 5:6 6:6 7:7 8:8 9:9 10:10	0.8	2
1:3 2:4 3:4 4:5 5:5 6:6 7:6 8:7 9:7 10:8	0.5	3
1:3 2:4 3:4 4:5 5:6 6:6 7:7 8:7 9:8 10:9	0.6	3
1:3 2:4 3:5 4:5 5:6 6:7 7:7 8:8 9:9 10:10	0.7	3
1:4 2:5 3:5 4:6 5:6 6:7 7:7 8:8 9:8 10:9	0.5	4
1:4 2:5 3:5 4:6 5:7 6:7 7:8 8:8 9:9 10:10	0.6	4
1:5 2:6 3:6 4:7 5:7 6:8 7:8 8:9 9:9 10:10	0.5	5

Here, a pair $i : j$ means that x_i is mapped to y_j (and to calculate the distance in one of the 15 alignments all $(x_i - y_j)^2$ have to be summed up). The first four index pairs in the first two lines are identical and similar situations can be spotted elsewhere, which raises the hope for saving computational costs when computing all uniform warping paths simultaneously to find the best. But there seems to be no systematic

way of utilizing calculations from one row for the calculation of another row. To discover the saving potential we employed Sequitur [10] to find a more compact calculation: Whenever there is a partial sum of at least two distances $(x_i - y_j)^2$ (shown as i:j) that is shared among two paths, it introduces a local variable that is computed only once (represented by a rule in Sequitur). Utilizing intermediate variables saves us from calculating individual distances twice. In the end, we determined how many operations are still left compared to the naive implementation. The result is shown in Table 1. Sequitur does a very good job, but as the last column shows, the computational effort still remains above $O(n^2)$.

n	$\#_{naive}$	$\#_{seq}$	$\#_{seq}/\#_{naive}$	$\#_{seq}/n^2$
20	1100	638	0.58	2.75
30	3600	1895	0.52	2.10
40	8400	3951	0.47	2.47
50	16250	6984	0.42	2.79
60	27900	11171	0.40	3.10

Table 1: How often are distance terms $(x_i - y_j)^2$ accumulated by the naive algorithm ($\#_{naive}$) and by Sequitur ($\#_{seq}$) depending on series length n.

However, we do not intend to choose a δ as low as 0.5 but want to compensate for small translational and dilational differences only. (We will use $\delta = 0.9$ throughout all experiments.) For short series, *LTD* is computed quite fast with the naive implementation. To speed up the calculation of LTD in settings such as 1-NN classification as well as longer series, we consider lower bounds of LTD in the next section.

5 Bounding LTD

The goal of bounding any distance measure is to skip distance calculations because we are not interested in an exact calculation if the value is guaranteed to stay above some thresholds (cf. Sect. 2). The difficulties with bounding LTD are twofold: First, we have to estimate the effect of two linear transformations on the distance and secondly, compared to uniform scaling the additional temporal offset enlarges the set of indexes j to which a point x_i is eventually mapped in $\mathbf{y}$.

COROLLARY 5.1. *Given time series* $\mathbf{x}$ *and* $\mathbf{y}$ *of length* n *with* $s_y := \sum_i y_i$, $s_{yy} := \sum_i y_i^2$, $s_{xy} := \sum_i x_i y_i$ *and bounds* $l_y \le s_y \le u_y$, $l_{yy} \le s_{yy}$ *and* $l_{xy} \le s_{xy} \le u_{xy}$. *The regression distance* $RD_a(\mathbf{x}, \mathbf{y})$ *is bounded by*

$$RD_a(\mathbf{x}, \mathbf{y}) \ge \frac{(ns_{xx} - s_x^2)l_{yy} - s_{xx}b_y + 2b_* - nb_{xy}}{ns_{xx} - s_x^2}$$

$$\text{with} \quad s_x = \sum_{i=1}^n x_i, \qquad s_{xx} = \sum_{i=1}^n x_i^2$$

$$b_y = \max\{l_y^2, u_y^2\}, \qquad b_{xy} = \max\{l_{xy}^2, u_{xy}^2\}$$
$$b_* = \min\{s_x l_{xy} l_y,\ s_x u_{xy} u_y,\ s_x l_{xy} u_y,\ s_x u_{xy} l_y\}$$

Proof. Substituting (3.2) and (3.3) into the definition of $RD_a(\mathbf{x}, \mathbf{y})$ leads to

$$RD_a(\mathbf{x}, \mathbf{y}) = \frac{(ns_{xx} - s_x^2)s_{yy} - s_{xx}s_y^2 + 2s_x s_{xy} s_y - ns_{xy}^2}{ns_{xx} - s_x^2}$$

Note that the term $(ns_{xx} - s_x^2)$ corresponds to $n\sum_i (x_i - \bar{x})^2$ and is thus guaranteed to be positive. For s_{yy}, s_y and s_{xy} we do not know exact values but have upper and lower bounds. As the denominator is known, we have to consider the nominator only. We obtain a lower bound by (1) replacing s_{yy} in the first term by its lower bound, (2) replacing s_y^2 in the second term by b_y, (3) replacing $s_x s_{xy} s_y$ by b_* and (4) replacing s_{xy}^2 by b_{xy}.

To use Corollary 5.1 we have to provide the various bounds under a linear temporal transformation of $\mathbf{y}$.

COROLLARY 5.2. *Given time series* $\mathbf{x}$ *and* $\mathbf{y}$ *of length* n *and a linear index mapping* f *(aligning* x_i *with* $y_{f(i)}$*)*

$$f(i) := \left\lfloor \frac{(ep - sp)}{n} \cdot i + sp + \frac{1}{2} \right\rfloor = j$$

where $sp \in [fsp, tsp]$ *(fsp is an acronym for "from start point", tsp for "to start point"),* $ep \in [fep, tep]$*, we have* $l_y \le s_y \le u_y$, $l_{yy} \le s_{yy}$ *and* $l_{xy} \le s_{xy} \le u_{xy}$ *for*

$$l_{yy} = \min\{\textstyle\sum_{j=k}^{k+\delta n} y_j^2 | fsp \le k < tsp\}$$
$$l_y = C + \min P + \min S + \textstyle\sum_{i=1}^G \min\{0, \min D_i\}$$
$$u_y = C + \max P + \max S + \textstyle\sum_{i=1}^G \max\{0, \max D_i\}$$
$$l_{xy} = \textstyle\sum_i \min\{x_i \cdot l_i, x_i \cdot u_i\}$$
$$u_{xy} = \textstyle\sum \max\{x_i \cdot l_i, x_i \cdot u_i\}$$

where $A_i = [fsp + i \cdot (fep - fsp)/n, tsp + i \cdot (tep - tsp)/n]$, $l_i = \min_{j \in A_i} y_j$, $u_i = \max_{j \in A_i} y_j$, $g = 1/(1 - (fep - tsp)/n)$, $G = \lceil n/g \rceil$, $C = \sum_{j=tsp}^{fep} y_j$, $P = \{\sum_{j=k}^{tsp-1} y_j | fsp \le k < tsp\}$, $S = \{\sum_{j=fep}^{k} y_j | fep \le k < tep\}$ *and* $D_i = \{y_j | i \cdot g \le j < (i+1) \cdot g\}$.

Proof. First, note that all considered mappings (with $sp \in [fsp, tsp]$, $ep \in [fep, tep]$) share a common index subrange: The full index range of $\mathbf{x}$, $1 \le i \le n$, is mapped to the index range $sp \le j \le ep$ of $\mathbf{y}$ and thus *always* contains the subrange $[tsp, fep]$ (denoted as the "core" range hereafter, cf. Fig. 3). Thus, the sum $\sum_{i=1}^n y_{f(i)}$ addresses all elements y_j with $j \in [fep, tsp]$ *at least* once (possibly more often).

Bounding s_{yy} is the easiest part because all summands are positive. We need only a lower bound of

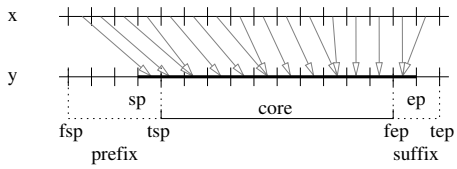

Figure 3: Mapping $\mathbf{x}$ to $\mathbf{y}$: when choosing $sp \in [fsp, tsp]$ and $ep \in [fep, tep]$ the *core* index range $[tsp, fep]$ is always involved.

s_{yy} and obtain it from summing δn elements (which is the required overlap), starting from the admissible start position that yields the smallest sum ($= l_{yy}$).

Bounding $s_y = \sum y_{f(i)}$ is more complicated. All core elements contribute to the sum s_y (summand C), but that leaves some elements unconsidered. The core sum may become larger or smaller by adding further elements from the prefix or suffix (cf. Fig. 3). From the set of possible prefix sums P and suffix sums S we find the smallest (resp. largest) partial sum and consider it in l_i (resp. u_i) via the second and third summand. Finally, we have to take care of the elements that may be added more than once. From the range of possible scaling factors we can find how many y_j will be double-counted at most (for scaling factor w every $g := 1/(1-w)$ indices we sum an element twice; this number becomes largest for the smallest factor $w_{min} = (fep - tsp)/n$). We organize the y_i values in groups of g elements (sets D_i) and add the smallest/largest element of this group to the lower/upper bound of s_y (last sum of l_i/u_i). Note the surrounding $\max\{0, \cdot\}$: By double-counting these elements only optionally (that is, only when the sum is increased (for the upper bound) or decreased (for the lower bound)) we implicitly cover all cases with a larger scaling factor than w_{min}.

To bound s_{xy} we use an upper and lower envelope (as with LB_{Keogh}, cf. Sect. 2), that depends on the possible range of index assignments: An index i of series $\mathbf{x}$ is assigned under any of the considered mappings to an index $j \in A_i$ with the set of admissible indices $A_i = [fsp + i \cdot (fep - fsp)/n, (tsp + i \cdot (tep - tsp)/n]$. (We compute l_i and u_i by iterating once through $i = 1..n$ and keep all y_j with $j \in A_i$ in a sorted data structure from which we can easily obtain the minimal value $l_i = \min_{j \in A_i} y_j$ and maximal value $u_i = \max_{j \in A_i} y_j$.) The upper and lower bound of $\sum x_i y_j$ are then obtained from $l_{xy} = \sum \min\{x_i \cdot l_i, x_i \cdot u_i\}$ and $u_{xy} = \sum \max\{x_i \cdot l_i, x_i \cdot u_i\}$.[2]

[2]Note that we obtain a lower and upper bound from the envelope directly, but the proposed way provides a tighter bound.

These bounds apply for the general case $\Omega_R = \mathbb{R}^2$. Tighter bounds can be derived for other Ω_R, which we omit due to lack of space.

As already mentioned, the intention is to skip calculations by the naive algorithm whenever the lower bound indicates that we will not arrive at a distance smaller than a certain threshold τ. Note that Corollary 5.2 provides a bound for arbitrary index ranges $[fsp, tsp]$ and $[fep, tep]$, which offers the possibility to subdivide the index range subsequently to find tighter bounds for parts of the search space Ω_T.

6 Experimental Evaluation

All experiments are reproducible: the datasets are publicly available or can be obtained from `http://public.ostfalia.de/~hoeppnef/ltd.html`.

6.1 UCR time series As already mentioned, it is somewhat surprising that the conceptually simple linear transformations were not appropriately considered in the literature on time series comparison in the past (cf. [5]). We close this gap by providing results on the most frequently used collection of time series [8]. We closely follow the experimental design of the exhaustive study in [3] by merging and shuffling the provided test and train datasets and performing a stratified cross-validation on 1-NN classification. (We use only the first batch of time series, because the second batch contains very large datasets that took 20 multi-cores over a month of computing power [3].) We report averaged classification rates on the same number of folds as in [3]. In this section we consider LTD using $\Omega_R = \mathbb{R}_{\geq 0} \times \mathbb{R}$ and use $\delta = 0.9$ for Ω_T throughout *all experiments* (no parameter tuning took place). The goal was simply to compensate for minor translational and dilational variations.

We want to compare the influence of different degrees of time warping flexibility. In that respect, ED and DTW represent two extrema in the spectrum of temporal transformations: while Euclidean distance deals neither with dilation nor translation, dynamic time warping supports any possible warping path (and is assumed to be the best measure on average in [11]). We expected LTD to settle down somewhere half way between both measures.

Table 2 gives the results. In the literature uniform scaling has been reported to perform superior for certain types of datasets [6, 7, 4] but to our surprise, LTD performs *consistently* comparable to DTW (and in some cases clearly better. Only with the synthetic control charts DTW seems better, but we will comment on this case at the end of the next subsection.) If LTD is at least as good as DTW, the model with the fewest assumptions

	#	ED	DTW	LTD
50words	5	0.58|0.01	0.61|0.01	0.62|0.01
Adiac	5	0.52|0.04	0.52|0.02	0.53|0.02
Beef	2	0.42|0.02	0.42|0.02	0.42|0.02
CBF	16	0.93|0.02	1.00|0.00	0.99|0.01
Coffee	2	0.88|0.03	0.86|0.05	**0.98|0.03**
ECG200	5	0.85|0.03	0.78|0.01	**0.85|0.02**
FaceFour	5	0.81|0.02	0.87|0.06	0.84|0.04
FISH	5	0.73|0.02	0.67|0.03	**0.77|0.01**
GunPoint	5	0.88|0.02	0.87|0.05	**0.99|0.01**
Lighting2	5	0.68|0.06	0.80|0.01	0.80|0.04
Lighting7	2	0.64|0.05	0.71|0.05	0.66|0.08
OliveOil	2	0.85|0.02	0.85|0.07	**0.90|0.01**
OSULeaf	5	0.52|0.03	0.58|0.04	0.57|0.03
SwedishLeaf	5	0.70|0.02	0.74|0.03	0.73|0.02
synth.control	5	0.86|0.01	**0.98|0.00**	0.91|0.00
Trace	5	0.66|0.03	0.99|0.01	0.99|0.01
TwoPatterns	5	0.67|0.01	1.00|0.00	0.99|0.00

Table 2: Mean accuracy and standard deviation of cross-validated 1NN-classifier (no. of folds in column #).

should be selected (Occam's Razor), that is, LTD.

6.2 Gain from Stiffness To provide insights into the performance of the similarity measures (rather than just 1-NN classification rates), we investigate the spread in the obtained distance values. Fig. 4 shows a histogram of the differences between the closest distances of the nearest series from a different and the same class. Positive differences indicate that a series of the same class is closest and thus the 1-NN prediction is correct. In the datasets shown DTW and LTD perform comparable, but one can see that LTD provides a wider spread for the distance values. This is due to the stiffness of LTD, whereas DTW can arbitrarily warp the temporal domain to compensate for deviations in the amplitude. If we interpret the difference of distances as a kind of confidence in the prediction, LTD is on average more confident than DTW.

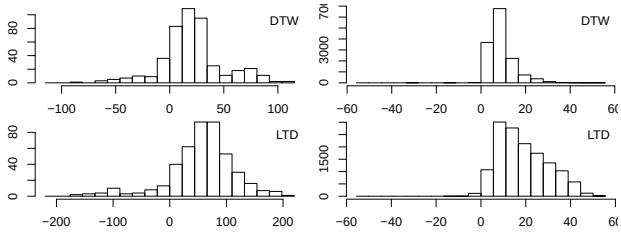

Figure 4: FaceFour (left) and CBF (right).

Next, we demonstrate that the high flexibility of DTW may actually be harmful. To underline the practical relevance, we provide two artificial examples that are motivated by experiences we made with real data. Both cases consists of 100 cases (of length 100) from two equiprobable classes. One may think of the different classes as a regular machine run (green example in Fig. 5) and a defective run (red example), where some moving part of a machine was slowed down or got stuck for a certain period of time. (The red and green examples are shifted for better visibility.) The goal is to distinguish both cases from each other. We have generated two versions of both datasets, in the first the series are already aligned (no variation in offset, only moderately different temporal scaling), the other additionally contains some variation in the starting point (temporal offset). Ten examples from the (aligned) datasets are also shown in Fig. 5 (black).

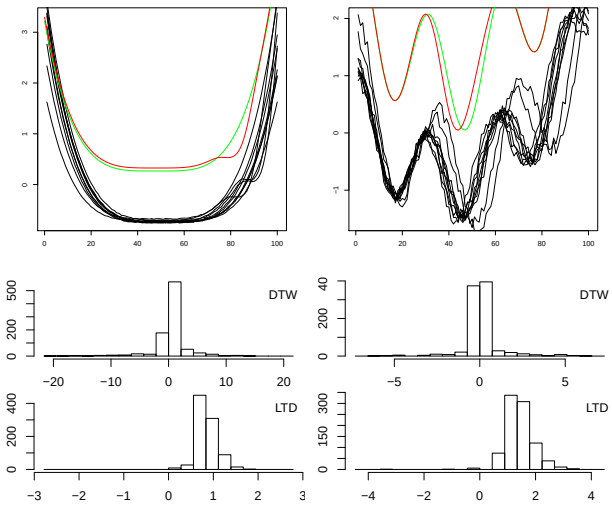

Figure 5: Two example datasets (left: jam, right: warp).

Table 3 shows the results on both datasets for a 10-fold cross-validation that was carried out in the same way as in the previous section. Especially in case of the *warp* dataset it becomes very difficult to distinguish the series visually, because the local distortion is quite small and superimposed by different scaling factors. For the aligned case, Euclidean distance does a better job than DTW. For the additional temporal offset, ED and DTW perform only a little better than guessing.

Figure 5 also shows the same type of histograms as Fig. 4. In both cases, the restricted warping capabilities of LTD allow for a finer distinction between the series. The warping flexibility of DTW is (mis-) used as a build-in smoothing filter, as illustrated in Fig. 6. The red series contains a bump (similar to jam dataset) which is important to distinguish series from the blue class (straight line). Then, under the warping path on the right of Fig. 6, both series become identical. DTW *smoothes* the bump *away* by time warping (and as this

set	#	ED	DTW	LTD
JAM	10	0.83 ± 0.08	0.74 ± 0.08	1.00 ± 0.00
WARP	10	0.76 ± 0.08	0.53 ± 0.05	0.99 ± 0.03
JAM2	10	0.67 ± 0.07	0.62 ± 0.05	0.91 ± 0.07
WARP2	10	0.56 ± 0.06	0.49 ± 0.05	0.94 ± 0.05

Table 3: Cross-validated mean accuracy (and standard deviation) for Euclidean (ED), DTW and LTD distance. JAM/WARP as in Fig. 5, JAM2/WARP2 with additional random offset.

is a local effect, constrained DTW will do so as well). This smoothing capability may be considered as a nice feature (and may be the reason why no smoothing is usually applied when comparing time series measures), but as it is *implicitly* contained in DTW we cannot control it very well. Explicitly smoothing the series and sticking to a linear temporal transformation may be considered as a more principled approach. Applying a smoothing filter may increase LTD performance (rising the results for synthetic control to 0.95 ± 0.01).

6.3 Normalization It is common practice to normalize time series before applying the distance measure. With LTD this normalization is embedded into the regression distance, which has the advantage that we can switch between different kinds of normalization without re-running the preprocessing and that we open grounds for notions of similarity that are not accessible by z-normalization.

As a first example, we use a real-world meteorological dataset, where the wind strength is used to label air pressure time series: the class information indicates whether the subsequent hours are calm our stormy. In the first experiment we normalized the time series and applied 1-NN classification. Then we repeated the experiments on the raw data and Table 4 shows the change in accuracy. While there is an improvement with all measures, the improvement is three times larger when only shift or only scale is considered. The explanation is simple: the absolute increase of the air pressure curve is

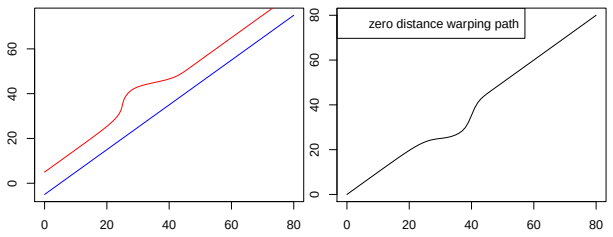

Figure 6: DTW explains bumps away by warping: Under the warping path (right) the blue and red series become identical.

ED	DTW	LTD			
		$\mathbb{R} \times \mathbb{R}$	$\mathbb{R}_+ \times \mathbb{R}$	$\mathbb{R} \times \{0\}$	$\{1\} \times \mathbb{R}$
0.03	0.03	0.03	0.02	0.11	0.09

Table 4: Average improvement in mean accuracy (of 10-fold cross-validation) when skipping z-normalization of the weather dataset. The four rightmost columns show results for LTD with Ω_R shown in the column header.

a good indicator for upcoming storm, the absolute value (measured in mbar), however, is not important. After normalization, the original slope of, say, a moderately increasing and steeply increasing air pressure curve is no longer distinguishable for the distance measure. When amplitude shift *and* scaling is allowed, we obtain similar results as with z-normalization. But if we forbid scaling (column $\Omega_R = \{1\} \times \mathbb{R}$) two air pressure curves have to be close in their original slope to match well.

We use the CinC_ECG_torso dataset from the UCR repository as a second example. This is a real-world dataset where ECG data was measured at multiple position of the torso. The data is not perfectly aligned and additionally, depending on where the series was measured, the peaks of the series may be inverted. While under z-normalization inverted series are dissimilar they become similar under regression distance with a possibly negative scaling factor. If the train dataset, however, contains many examples (normal and inverted), even ED does a good job because it just picks the correctly scaled example from the training set. Therefore, we took only the first 200 examples from the test set plus the 40 examples from the training set (240 in total) and again performed a stratified *30-fold* cross-validation. That is, the training sets consist only of $\frac{240}{30} = 8$ examples (per fold), *only 2 from each class*. Table 5 shows a superior performance of LTD with $\Omega_R = \mathbb{R}^2$.

set	#	ED	DTW	LTD
CinC torso	30	0.59 ± 0.07	0.64 ± 0.03	0.73 ± 0.04

Table 5: Cross-validated mean accuracy (and standard deviation) for Euclidean (ED), DTW and LTD for 30-fold cross-validation on 240 CinC samples.

Remember that such notions of similarity cannot be achieved by ED/DTW on preprocessed data, because the actual distance depends on the *pair of compared series* (while preprocessing operates on single series).

6.4 Lower Bounding The effectiveness of lower bounding obviously depends on the dataset, that is, the time series (the inherent pruning power [4]) and the order in which they are processed. The earlier we find a

c	sb	t
5	97%	39%
10	91%	21%
20	83%	26%
all	64%	36%

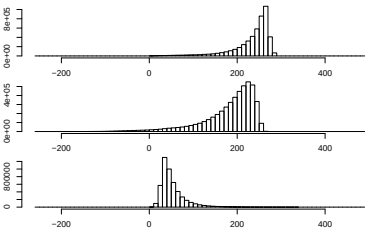

Table 6: Effect of pruning. Left: block size (c), skipped blocks (sb), runtime (t) relative to naive algorithm. Right: Histogram of distances (top), lower bounds (mid), delta between distance and lower bound.

close pair, the more we benefit from lower bounds. In [11] various suggestions were made to find a good guess heuristically. We tested many datasets, but discuss only results on the *50words* dataset. Table 6 presents the results for a 2-fold cross-validation *without* any of these optimizations only. The length of a series is 270, with $\delta = 0.9$ we consider a starting point in [1,27] and endpoint in [243,270]. We can see from the last line of Table 6 that 64% of the cases were pruned if just one lower bound is calculated. For the other lines, Corollary 5.2 was used to subdivide the range of possible start/end points in chunks of a certain size (given in first column). That is, for a chunk size of $c = 10$, we calculate lower bounds for blocks of starting-points ($[0, 10[, [10, 20[, [20, 30]$) combined with blocks of endpoints ($[243, 253[, [253, 263[, [263, 270]$). (Alternatively, a recursive bi-partitioning may be applied.) While the calculation of the lower bound is $O(n \log n)$, the naive algorithm used to find the best start/endpoint combination per block is $O(c^2 \cdot n)$. Therefore we can trade lower bound calculations against naive calculations (per block). In case of a block size of $c = 10$ the best trade-off is achieved in this example.

7 Conclusions

From the obtained results we draw the following conclusion. First, regarding time warping, it seems that the higher flexibility of DTW is seldomly needed. It may even be harmful because DTW has some built-in noise cancelling that is always applied – whether it is helpful or not (cf. 6.2). Applying a smoothing filter separately and stick to linear transformations might be a more principled approach.

Second, the regression approach with positive scaling factor behaves comparable to z-normalization, but regression offers several variations (e.g. negative scaling factor) that can give an advantage in specific applications (unachievable by preprocessing). Switching between these variations is easy and, more importantly, subsequent steps (the proposed bounding) can also cope with the variations. Probably none of these variations beat the default case with positive scaling factor over a broad range of data sets, but we connected properties of the application with the properties of the used regression model and therefore the choice can be driven by background knowledge about the application.

Third, although the concept of LTD is simple, its calculation takes some time. But after a rough alignment only limited dilational or translational effects are to be expected. The presented lower bound, that covers amplitude and time transformations, greatly helps to reduce the computational effort further.

References

[1] T. Argyros and C. Ermopoulos. Efficient subsequence matching in time series databases under time and amplitude transformations. In *IEEE Data Mining*, pages 481–484. IEEE Comput. Soc, 2003.

[2] D. J. Berndt and J. Clifford. Finding Patterns in Time Series: A Dynamic Programming Approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248. MITP, 1996.

[3] 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. In *Proc. VLDB Endow.*, volume 1, pages 1542–1552, Aug. 2008.

[4] A. W.-C. Fu, E. Keogh, L. Y. H. Lau, C. A. Ratanamahatana, and R. C.-W. Wong. Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921, Mar. 2007.

[5] B. Hu, Y. Chen, and E. Keogh. Time Series Classification under More Realistic Assumptions. In *SIAM Int. Conf. Data Mining*, number 1, pages 578–586, 2013.

[6] E. Keogh. Efficiently Finding Arbitrarily Scaled Patterns in Massive Time Series Databases. In *Knowledge Discovery in Databases (PKDD)*, pages 253–265, 2003.

[7] E. Keogh and T. Palpanas. Indexing large human-motion databases. In *Proc. Int. Conf. Very Large Databases*, pages 780–791, 2004.

[8] E. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage, 2011.

[9] J. Listgarten, R. M. Neal, S. T. Roweis, and A. Emili. Multiple Alignment of Continuous Time Series. In *Advances in Neural Information Processing Systems*, 2004.

[10] C. G. Nevill-Manning and I. W. Witten. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, Sept. 1997.

[11] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. *SIGKDD Knowledge Discovery and Data Mining*, page 262, 2012.