Indexing and classifying gigabytes of time series under time warping

Chang Wei Tan Geoffrey I. Webb François Petitjean
Faculty of IT, Monash University, Melbourne, Australia – firstname.lastname@monash.edu

Abstract

Time series classification maps time series to labels. The nearest neighbour algorithm (NN) using the Dynamic Time Warping (DTW) similarity measure is a leading algorithm for this task. NN compares each time series to be classified to every time series in the training database. With a training database of N time series of lengths L, each classification requires $\vartheta(N \cdot L^2)$ computations. The databases used in almost all prior research have been relatively small (with less than $10,000$ samples) and much of the research has focused on making DTW's complexity linear with L, leading to a runtime complexity of $O(N \cdot L)$. As we demonstrate with an example in remote sensing, real-world time series databases are now reaching the million-to-billion scale. This wealth of training data brings the promise of higher accuracy, but raises a significant challenge because N is becoming the limiting factor. As DTW is not a metric, indexing objects induced by its space is extremely challenging. We tackle this task in this paper. We develop TSI, a novel algorithm for Time Series Indexing which combines a hierarchy of K-means clustering with DTW-based lower-bounding. We show that, on large databases, TSI makes it possible to classify time series orders of magnitude faster than the state of the art.

Keywords: Time series classification, time series indexing, dynamic time warping

1 Introduction

The European Space Agency's Sentinel-2 satellites provide a full picture of Earth, every 5 days, at 10m resolution [2]. This and the corresponding NASA Landsat-8 programs introduce unprecedented opportunities to monitor the dynamics of any region of our planet over time and understand the constant flux that underpins the bigger picture of our world (more details at www.esa-sen2agri.org/SitePages/EOData.aspx). A high resolution satellite view of Houston city taken by the Sentinel-2A satellite, obtained from the Sentinels Scientific Data Hub (https://scihub.copernicus.eu/s2) is shown in Figure 1.

All images from these satellites are, by default, geometrically and radio-metrically corrected. Geometric corrections ensure that every pixel (x, y) always maps to the same geometric area. Radiometric corrections ensure that the spectral information is comparable from one image to the next of the series. This provides for each geographic coordinate on Earth, a time series of the "colours" that it underwent over the study period. One of the core tasks is to create temporal land-cover maps that describe the evolution of an area over time. This task is summarized in Figure 2: mapping the spectral evolution of a "pixel" (geographic coordinate) to a land-cover class such as "wheat crop", "broad-leaved tree" or "urban". Evolution is critical because, from space, all crops look the same; what makes it possible to correctly differentiate one from another is the temporal evolution (when the crop grows, when it is harvested etc.).

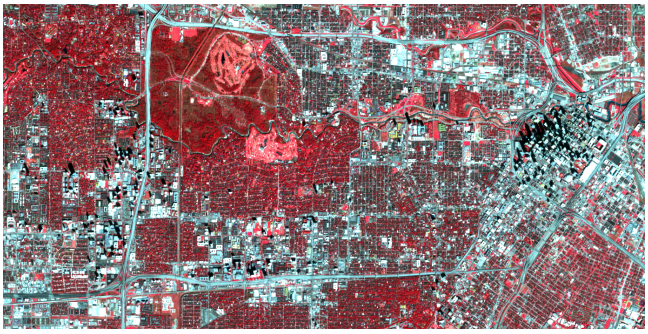

Figure 1: High-resolution image of Houston city near Westin Galleria taken by Sentinel-2A. © Copernicus Sentinel data [2016] for Sentinel data.

Quite simply, research into time series classification lags behind the demands of modern space imagery, which produce terabytes of data each day. Why? Most research into time series classification has addressed datasets that hold no more than 10 thousand time series [7]. In contrast, the Sentinel-2 satellite provides over 10 trillion time series, capturing Earth's land surfaces and coastal waters at resolutions of 10 to 60m [2]. Although much research has gone into classifying remote sensing images, few studies have analysed time series extracted from sequences of satellite images.

The go-to time series classification method in terms of accuracy for this type of task is Nearest Neighbour coupled with Dynamic Time Warping (NN-DTW) [25, 26]. This is for two main reasons: (1) many phenomena of interest – vegetation cycles, for instance – have a

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

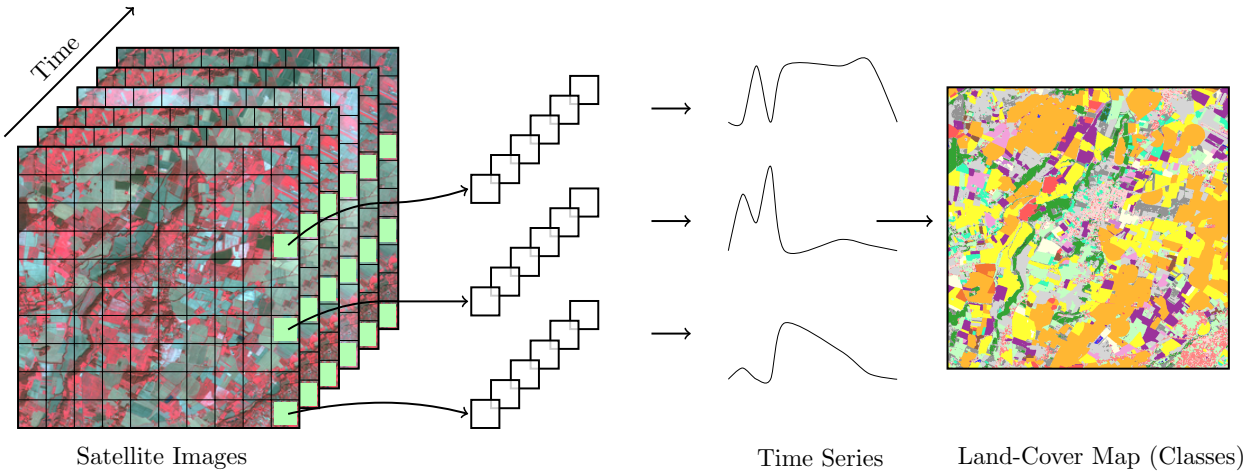

Figure 2: Production of a time series datasets from satellite image series.

periodic behaviour which can be slightly modulated by weather artifacts. These modulations result in distortions of canonical temporal profiles that are well handled by DTW [20]. (2) Time series are too short for Bag-of-word-type approaches [28, 29] to perform best.

NN-DTW cannot scale to the typical size of satellite datasets where it is common to have 100 million example time series [9, 10]. This is because to classify each query time series, we have to scan the entire 100 million training dataset. Even making the most of lower-bounding [12, 15], this is completely infeasible. Figure 3 illustrates this point: while all datasets of the standard archive of time series [7] can be classified in less than 30 minutes, creating a temporal land-cover map for just a city like Houston (16 million time series) assuming a bare minimum of 1 million training examples would take about a year to complete. To create a land-cover map of Texas (7 billion time series) with a reasonable training dataset of 100 million samples would require 30k years of computation.

With these motivations, this work tackles **Contract Time Series Classification**, where we would like to produce the most accurate classifier under a contracted time (obviously significantly smaller than running the NN-DTW). We propose a new algorithm that efficiently indexes the training database using a hierarchical K-means tree structure specifically designed for DTW. We will show that our algorithm reduces the time per query while retaining similar error to the state of the art, NN-DTW.

This paper is organized as follows. In section 2, we review some background and define the problem statement for our work. Then in section 3 we introduce and describe our approach. Section 4 shows the empirical evaluation for our approach. Lastly, section 5 offers some direction for our future work and we conclude our work in section 6.

2 Background and Motivation

2.1 Time Series Classification Many time series classification algorithms in the literature such as Shapelets [23, 33], 1-NN BOSS [28] and SAX-VSM [29] have been shown to be competitive (and sometimes superior) to the state of the art, NN-DTW.

Nonetheless, as explained in the introduction, classification of the Satellite Image Time Series (SITS) is better tackled by NN-DTW. NN-DTW has been shown to be extremely competitive for many other applications [4, 19, 20, 22, 24, 30, 31]. It has been argued that the widespread utility of NN-DTW is due to time series data having autocorrelated values, resulting in high apparent but low intrinsic dimensionality. Experimental comparison of DTW to most other highly cited distance measures on many datasets concluded that DTW almost always outperforms other measures [30].

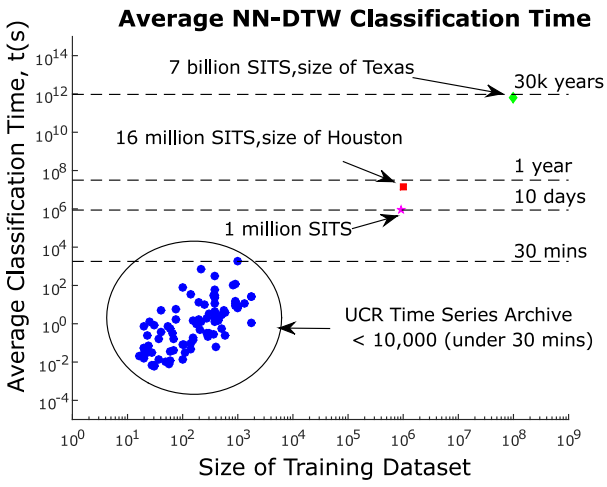

Figure 3: Average NN-DTW Classification Time on different datasets

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

2.2 Nearest Neighbour Search The nearest neighbour (NN) classifier (out of the context of time series) is one of the simplest classification algorithms but is nonetheless highly effective. It is a non-parametric, lazy learning algorithm and does not require any abstraction/training phase on the training dataset [27]. Its behaviour is also very interesting for large datasets, as its bias vanishes in the limit when the dataset size increases. k-NN classifier finds k nearest neighbours of the query in the training dataset and returns the dominating class as the class for the query sample [27].

Despite its nice behaviour in the limit, k-NN has the major drawback of not being scalable: it has to compare the query sample to all samples in the training dataset, which is infeasible for most large datasets.

An efficient and popular method of scaling up NN classification on large datasets is to use the k-d tree structure, where k is the dimensionality of the Euclidean search space [5]. k-d tree allows faster retrieval of the nearest neighbour to the query sample in a short amount of time. It is very efficient in the Euclidean space. However, it is not applicable for DTW-induced space as k-d tree partitions each dimension of the data recursively while DTW makes associations across dimensions. Because of that, k-d tree will not be applicable to NN-DTW for time series classification.

A large amount of research has been done to scale up NN-DTW for time series classification such as early abandoning [13] and lower bounding [12, 15, 22]. As DTW has time complexity of $O(n^2)$, these methods speed up classification by minimizing DTW computations. In this work, we focus on approximate nearest neighbour search, where it is typical to be two or more orders of magnitude faster than linear search [17].

Approximate nearest neighbour search algorithms index the data in the training dataset in a systematic manner so that the query sample will be compared to only a few promising candidates from the training dataset within a given time. It may not return the actual nearest neighbour, but with sufficient data it is highly likely that the approximate nearest neighbour will be of the same class.

The Priority Search K-means Tree (PSKMT) algorithm [17] is a recent breakthrough approach to nearest neighbor search that performs a priority search in a hierarchy of K-means clusterings. It has high precision on large high dimensional datasets and is still one order of magnitude faster than previous approximate search algorithms [16, 17]. The algorithm outperforms existing approximate nearest neighbour algorithms on the 31 million sample SIFT dataset [17]. This algorithm offers an incredible opportunity for time series because, unlike k-d trees, it does not require that the data be tabular.

This paper adapts and extends PSKMT to time series creating a novel efficient time series classification algorithm. Such adaptation was not possible before because K-means clustering was ill-defined for the DTW-induced space. Hence, we leverage our recent work on clustering time series consistently for DTW [19, 21] using DTW Barycenter Averaging (DBA) and adapt PSKMT to make the most out of the available lower bound for DTW. DBA has been extensively studied in [18, 19, 21] and a proof of convergence can be found in [18, 19].

Our experiments demonstrate that our method makes it possible to classify large time series datasets two orders of magnitude faster than the state of the art, NN-DTW with LB_Keogh [11, 12].

2.3 Contract Time Series Classification In recent years there has been an increasing interest in using any-time algorithms for data mining [14, 32]. However, the variant known as contract algorithms have received less attention. Contract algorithms are a special type of any-time algorithms that require the amount of run-time to be determined prior to their activation. In other words, contract algorithms offer a trade-off between computation time and quality of results, but they are not interruptible.

Problem Statement Contract Time Series Classification: produce the most accurate time series classifier given (1) set constraints on computational resources available at classification time and (2) no constraints on computational resources at training time.

We assume that the computational resource constraint will be time, not space, and that it will be given to us in the form of the number of CPU cycles available for each query to classify. We assume that the constraint will be given as a positive integer L which is the number of time series to examine; for ease of exposition, we will report the result of our algorithm on different datasets at regular time intervals (i.e. with different L).

3 Our approach: DTW-indexing of time series for classification

Our algorithm, Time Series Indexing (TSI) is an adaptation of Priority Search K-means Tree (PSKMT) [17] to index time series embedded in a space induced by DTW. The general outline is as follows.

At **training time**, we construct a hierarchy of K-means clusterings over the training dataset; this prepares the indexing data structure that will allow fast querying at testing time. The K-means clustering is performed using DTW as the similarity measure for associating time series to their closest centroids. DBA [19, 21] is used to create and refine the centroids from

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

Algorithm 1: build_tree(D, K, I_{max}, w)

Input: D: Time series dataset
Input: K: Branching factor
Input: I_{max}: Maximum k-means iterations
Input: w: Warping window

```
if |D| ≤ K then create_leaf(D) ;
else
    P = kmeanspp(D, w);
    for iterations = 1 : I_max do      // k-means
        C = assign_to_centroids(P, D, w);
        for all C_i ∈ C do P_i = DBA(P_i, C_i, w) ;
    end
    /* recursively build tree                */
    for all C_i ∈ C do build_tree(C_i, K, I_max, w) ;
end
```

Algorithm 2: assign_to_centroids(P, D, w)

Input: P: Cluster centroids
Input: D: Time series dataset
Input: w: Warping window
Output: *cluster*: Clusters of time series

```
cluster = ∅;
for all S_i ∈ D do
    nearest_p = search_nearest_lb(S_i, P, w);
    cluster[nearest_p].add(S_i);
end
return cluster;
```

the associated time series in the expectation phase.

At **testing time**, we maintain three priority queues. The first two queues store the potential branches to explore once exploration of the current branch is complete. The first stores those for which full DTW to the query has been calculated and the second stores those for which only lower bound have been computed. The third queue stores the nearest neighbours.

Since we prune off DTW with lower bound (LB), having 2 priority queues ensures that we always traverse from the actual closest branch without having to compute the full DTW distance for all potential branches. We start by descending the tree from the root to the first leaf, at each internal node following the branch closest to the query but pushing the alternatives to the priority queues. Those alternatives that can be excluded just on the lower bound go to the second queue while those for which the full DTW distance is computed go to the first queue. We explore the leaf, and then proceed to the closest branch that was not explored on the path to that leaf (stored as the head of the first queue). We continue in this cycle, stopping when we have exhausted our "contracted time" (or have explored the full tree).

Algorithm 1 presents the algorithm for building the tree. The root node contains all the training data. The algorithm recursively clusters the data associated to each node into K clusters. A branch is formed leading to each of the K child nodes, each child node associated with the data in one cluster. The branch is labelled with the DTW average of the time series in the node to which it leads. The data associated with each child node is then clustered into K sub-clusters; and so on recursively. All nodes are labelled with the majority class of the data associated with them. This allows the algorithm to give a plausible class prediction even when we have not yet reach a leaf node. Every recursion is initialized using the standard non-deterministic K-means++ algorithm [3]. The recursion stops when a node contains K or fewer time series.

To further speed up the clustering process, LB_Keogh is used with NN-DTW in the *search_nearest_lb* sub-routine to assign each time series to the nearest cluster centroid, as described in Algorithm 2. Our algorithm is not limited to just LB_Keogh. It can be used with any DTW lower bounding functions such as LB_Improved [15], depending on the application. In this work, we chose to use LB_Keogh because in general, it performs well for most time series datasets.

Algorithm 3 describes the tree search algorithm. The tree is searched by first traversing down the tree to the first leaf node, outlined in Algorithm 4. At each level of the tree, the algorithm proceeds with the nearest branch to the query time series. To efficiently search for the nearest branch, we first sort all the branches in ascending lower-bound distance to the query, then use NN-DTW to find the closest branch. This further minimizes DTW computations. The unexplored branches where DTW have been computed will be enqueued into the DTW priority queue while the remaining ones will be enqueued into the LB priority queue. These priority queues are implemented as min-heaps [6], with standard enqueue and dequeue functions.

When the query reaches a leaf node, the algorithm searches for the nearest time series using the same method as searching for the nearest branch. The k nearest time series found in the search so far are kept in the nearest neighbour priority queue, implemented as a max-heap. A max-heap allows faster retrieval of the k^{th} nearest neighbour from the query.

After exploring the leaf node, the algorithm proceeds to the next branch by dequeuing the DTW priority queue. Both DTW and LB priority queues are compared to ensure that the closest branch to the query is

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

Algorithm 3: search_tree(T, Q, L, w)

Input: T: Hierarchical k-means tree
Input: Q: Query time series
Input: L: Number of time series to examine
Input: w: Warping window
Output: kNN: k nearest neighbours

Initialize priority queues & seen=0
$W = envelope(Q, w)$;
$traverse\_tree(T, Q, W, PQs, L, seen, w)$;
while $(PQs\ not\ empty)$ && $(seen < L)$ **do**
    /* find nearest branch */
    while $lb\_PQ.top < dtw\_PQ.top$ **do**
        $lb\_branch = lb\_PQ.dequeue$;
        $d = DTW(Q, lb\_branch.Centroid, w)$;
        $dtw\_PQ.enqueue(lb\_branch)$;
    end
    if $dtw\_PQ\ not\ empty$ **then**
        $T = dtw\_PQ.dequeue$;
        $traverse\_tree(T, Q, W, PQs, L, seen, w)$;
    end
end
$kNN = nn\_PQ.getAllData$;
return kNN;

Algorithm 4: traverse$(T, Q, W, PQs, L, seen, w)$

Input: T, Hierarchical k-means tree
Input: Q, Query time series
Input: W, Envelope for query time series
Input: PQs, dtw_PQ, lb_PQ and nn_PQ
Input: L, Number of time series to examine
Input: $seen$, Time series seen so far
Input: w, Warping window

if $N\ is\ Leaf$ **then**
    $S = T.getAllData$;
    $lb\_distance = sort\_with\_lb(W, S)$;
    for $all\ S_i \in S$ **do**
        $worst\_d = nn\_PQ.firstDistance$;
        if $lb\_distance_i < worst\_d$ **then**
            $d = DTW(Q, S_i, w)$;
            if $d < worst\_d$ **then**
                $nn\_PQ.enqueue(S_i, d)$;
        end
        if $++seen == L$ **then** break;
    end
else
    $C = T.getChildren$;
    $dtw\_flag = [false, ...false]$;
    $best\_so\_far = \inf$;
    $distances = sort\_with\_lb(W, C)$;
    for $all\ C_i \in C$ **do**
        if $distances_i < best\_so\_far$ **then**
            $distances_i = DTW(Q, C_i, w)$;
            $dtw\_flag_i = true$;
            if $distances_i < best\_so\_far$ **then**
                $best\_so\_far = distances_i$;
                $C_q = C_i$;
            end
        end
    end
    for $all\ C_i \in C\ except\ C_q$ **do**
        if dtw_flag_i **then**
            $dtw\_PQ.enqueue(C_i, d_i)$;
        else $lb\_PQ.enqueue(C_i, d_i)$;
    end
    $traverse\_tree(C_q, Q, W, PQs, L, seen, w)$;
end

in the DTW priority queue: if the head of the LB priority queue is smaller than the head of the DTW priority queue, we dequeue that branch, compute its DTW distance and enqueue it into the DTW priority queue. The algorithm stops searching the tree when it has seen at least L time series from the leaf nodes. Here, L can also represent the "contracted" classification time.

Our source code has been uploaded to Github[1].

4 Empirical Evaluation

In this section, we comparatively assess the performance of our algorithm for large-scale time series classification against the state-of-the-art method:

- **LB_Keogh NN-DTW**: This is the state-of-the-art approach of doing NN-DTW [11, 12] and serves as the baseline algorithm in this work. Time series are taken one by one: if their lower-bound distance to the query is greater than the best-so-far neighbour, then the time series is pruned and the method proceeds to the next series, else its actual DTW distance with the query is computed and it becomes the best-so-far neighbour if it is closer than the current best-so-far. Because the pruning power depends on how close are the neighbours found early in the search, we report the average results over 10 different shuffling of the data.
- **Time Series Indexing (TSI)**: This is our proposed method described in Section 3. As we use the non-deterministic K-means++ [3] initialization, we report the average results over 10 different runs.

[1] https://github.com/ChangWeiTan/TSI

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

Since our task is contract classification, we provide – for all methods – their results at different time intervals. It is worth noting that the results for all methods tend to the full NN-DTW as the time constraint tends to infinity. Our experimental datasets and results have been uploaded to [1].

Our experiments are divided into two parts:

A. First, we begin with a real world case study to show the practical utility of our technique.

B. Then, we assess the performance of the different methods on a diverse range of datasets. We show that our approach is more accurate than the conventional approach under a contracted time.

A. Satellite Image Time Series (SITS) Classification As motivated in the introduction, the new-generation Earth Observation (EO) satellites have begun imaging Earth frequently, completely and in high-resolution. This introduces unprecedented opportunities to monitor the dynamics of any regions on our planet over time and revealing the constant flux that underpins the bigger picture of our world.

To the best of our knowledge, there has been little prior research into the classification of time series extracted from satellite image series – also called Satellite Image Time Series (SITS). The ability to monitor and classify these time series will have strong impact in many domains especially in the agriculture industry, marine applications and for environment monitoring. In prior work [20], we showed that DTW is a good measure for such time series, because of the non-linear distortions of prototypical ground surfaces, i.e., the fact that two neighbouring surfaces might have slightly different growth rates, although belonging to the same class of crop/tree.

In this work, we used 46 geometrically and radiometrically corrected images taken by FORMOSAT-2 satellite. These images are corrected such that every pixel corresponds to the same geographic area on Earth. Each of these images consists of 1 million pixels and each pixel represents a geographic area of 64m^2, resulting in an area of 64km^2 per image. Each geographic area (x, y) ($\sim$pixel) in the image forms a time series with a length of 46, creating a dataset with 1 million time series. The series have been manually collected by experts in geoscience by a mix of photo-interpretation, ground campaigns and urban databases. All time series thus have a label about their temporal class such as "wheat", "maize", "broad-leaved tree", etc. The formation of Satellite Image Time Series is illustrated in Figure 2, labelled with their temporal classes, represented in different colours.

Table 1: Properties of 1 Million SITS dataset

SITS 1 Million	
Length	46
Size of Dataset	1,000,000
Number of classes	24
Warping window size	4
1-NNDTW Error-Rate (10 fold cv)	0.168

To ensure reproducibility of our results and encourage researchers to work on large time series datasets, we have obtained permission from the CesBIO and French Space Agency to make our satellite dataset available online in [1]; note that this is a very high-cost dataset (images are worth more than USD100,000 and collecting the labels required months of work) which we hope will be a significant motivation to the field. As there are no pre-defined train/test sets for this dataset, we used 10-fold cross validation results. In this experiment, a warping window of 4 was used: this is aligned with the phenology of observed phenomena for which similar stages of growth cannot be distant by more than about a month [20]. The properties of this dataset that will be used in this case study are shown in Table 1.

The case study was conducted by varying the contract time from the least to the more permissive, i.e. with more and more time allowed in the contract to perform the classification of each query until we have gone through the whole training dataset. We record the error for each query as we go through the whole training dataset. Note that at smaller time intervals, TSI was not able to predict the error because the algorithm has not reached a leaf node. In this case, we predict the error using the majority class of the time series set in the nearest branch explored to date.

We present our results in Figure 4 where the x-axis is in log-scale. The first element to note is that our algorithm, TSI is significantly better than the state of the art; having its curve consistently under the state of the art. Second, we can see that if we had a 'contract' to classify each time series in no more than 1ms, then TSI would obtain error rate of 0.195 as opposed to 0.374. The same observation holds for 0.1ms (error of 0.283 vs 0.5), 10ms (error of 0.178 vs 0.287) and 100ms (error of 0.17 vs 0.220).

This is a very important result: imagine if you were satisfied with an average error-rate of 0.2, then using our approach would classify each query within 1ms, as opposed to 500ms for the state-of-the-art approach. Having a million time series to classify, this would translate to our approach, TSI finishing in 17 minutes as opposed to 138 hours for LB_Keogh NN-DTW; a 500

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

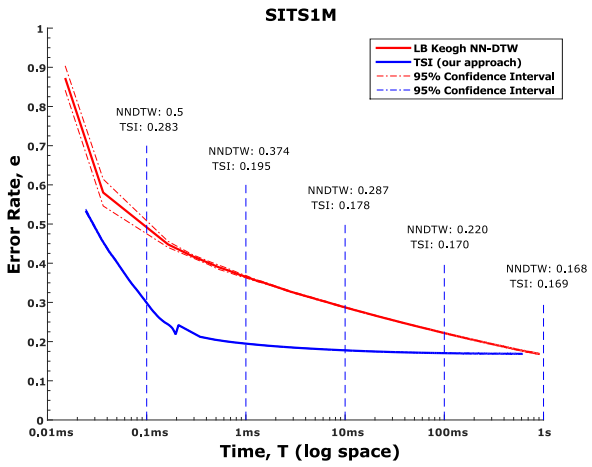

Figure 4: Comparative results on our 1 million SITS showing the average error rate per query as we go through the whole training dataset

times speed-up. It is only when getting to the far right of the curve that the overheads of TSI – linked to the exploration of the tree and maintenance of the priority queues – would become disadvantageous. This makes sense because if the contract is that you have the time to explore all of the training set, then you might as well just do that rather than using our approximate search.

Another point to note in Figure 4 is the slight jump in error for TSI at approximately 0.2ms. This is when the first leaf is expanded and the first class of an actual example is used in place of the majority class of a traversed branch.

B. Contract Time series classification To show that our algorithm, TSI can classify more accurately within a contracted time, we run a statistical comparison of classifiers [8] on all the datasets from the standard UCR time series archive[2] [7] plus our SITS dataset, all together 85 datasets. We use the train/test split from [7] and warping window size reported in [7] for the time series archive. Note that, the datasets from the standard time series archive are relatively small, ranging from training size of 20 to 1,800 [7]. The different classifiers are compared using the Wilcoxon Signed-Ranks Test described in [8]. Wilcoxon Signed-Ranks Test is a test which ranks the differences in performance of two classifiers for each dataset [8]. We want to assess if 85 datasets is a large enough sample to show that our algorithm is statistically different.

[2]We exclude `ElectricDevices`, as most series are identical under time warping, thus preventing any clustering.

Similar to the methodology in our case study, we record the error for each query at different time intervals, for every dataset and algorithm. To make the comparison fair for different datasets with different training size, we align the time intervals on the time of processing the whole dataset with LB_Keogh NN-DTW. We consider 6 time intervals from 1% to 50% (of the time it takes LB_Keogh NN-DTW to process one query). Similarly at smaller time intervals, we use the majority class of the nearest branch to predict the error. Let us take an example to ensure that our methodology is clear. We run LB_Keogh NN-DTW for a dataset and found that it takes, on average, 1s to classify a query. We then study the error-rate of TSI at 10ms(1%), 100ms(10%), etc. up to 500ms(50%).

Using the error-rate at these time intervals, we calculate the difference in error-rate, d_i of LB_Keogh NN-DTW and TSI on the i-th out of the N datasets. We then rank the differences by their absolute values at each time intervals [8]. Average ranks are assigned in case of ties [8]. These ranks are then used to calculate R^+ and R^- using Equation 4.1 and 4.2 respectively [8]. R^+ represents the sum of ranks for the datasets where the second algorithm outperforms the first and R^- the opposite [8]. In our context, the first algorithm refers to LB_Keogh NN-DTW and the second refers to TSI.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{4.1}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{4.2}$$

With 85 datasets ($N = 85$), the critical value is calculated using Equation 4.3 that is distributed approximately normal [8]. To reject the null-hypothesis (where the two algorithms perform equally well) with $\alpha = 0.05$, the test statistic has to be less than the critical value, $z < -1.96$. The results are shown in the last column of Table 2 where we reject the null hypothesis highlighted in bold.

$$z = \frac{min(R^+, R^-) - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{4.3}$$

Table 2 shows the results of the statistical comparison where we report the average rankings of LB_Keogh NN-DTW and TSI across all datasets at the different time intervals and the Wilcoxon Test statistics. The average ranking allows us to identify the better performing classifier (emphasized in bold) if we reject the

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

Table 2: Wilcoxon Test Results

LB_Keogh NN-DTW vs TSI					
	Average Ranks		Wilcoxon Test Statistics		
Intervals	NN-DTW	TSI	R^+	R^-	z
1%	1.529	**1.471**	2034.5	1620.5	-0.907
10%	1.841	**1.159**	3449	206	**-7.105**
20%	1.871	**1.129**	3451	204	**-7.114**
30%	1.806	**1.194**	3219.5	435.5	**-6.099**
40%	1.741	**1.259**	2903	752	**-4.713**
50%	1.671	**1.329**	2616	1039	**-3.455**
Average	1.743	**1.257**			

null hypothesis. In this case, our algorithm, TSI performs more accurately than the state of the art under a contracted time. The results show that TSI is more accurate than LB_Keogh NN-DTW at all time intervals except for 1%, where we are unable to reject the null hypothesis. As the majority of datasets tested are small, the algorithms are unable to find even an approximate nearest neighbor at the 1% time interval, as a result of which they predict the majority class. Note that if the classes are very similar and the clusters do not well separate the classes, TSI can underperform LB_Keogh NN-DTW. This is, for example, the case for the `Computers` dataset; the associated plot is available at [1].

5 Optimizing the Number of Clusters, K

From the experiments, we observed that the number of clusters (branching factor), K is an important parameter that determines the convergence rate of the algorithm. In our experiments K was chosen to be 3. Although this is not the optimal K for the algorithm, we had shown that TSI still outperforms the state of the art if we just have 50% of the full 1-NN time.

However, we believe that for each dataset there exists an optimal K that allows the algorithm to converge faster to the full 1-NN error rate. This will be a trade-off between numerous factors. Larger K means that the length of each complete branch in the tree will be shorter and hence there will be fewer internal nodes to traverse to reach the leaf nodes that contain candidate nearest neighbors. However, it also means that at each internal node more DTW calculations must be performed to select the branch to follow.

There are many potential ways to optimize K. Here, we suggest an intuitive way to optimize K without compromising the error rate of the algorithm. We can vary K and record the average time per query to find the exact nearest neighbor at each K value. Then, we keep the K value that gives the minimum time per query.

6 Conclusion and Future Work

In this work, we have proposed the first algorithm to index DTW-induced space and showed that it is essential for the classification of time series when a large amount of data is available. We demonstrated that on a large remote sensing data where time series classification is critical, we are able to obtain the same accuracy up to 2 orders of magnitude faster than the state of the art, NN-DTW search; we can thus classify the entire 1 million dataset in 17 minutes instead of 5 days. This is extremely promising for larger remote sensing datasets that contain hundreds of millions of examples [9, 10].

Besides optimizing the branching factor, K, our future work will also include speeding up the search for the best warping window for large datasets and improved approaches for selecting a branch to follow. The current way of finding the best warping window, is to repeat LB_Keogh NN-DTW with different warping windows, which is computationally expensive for large datasets. With the fast error convergence rate of our method, we can find the best warping window for large datasets in a short amount of time.

7 Acknowledgement

The authors would like to thank the researchers from CESBIO (D. Ducrot, C.Marais-Sicre, O. Hagolle and M. Huc) for providing the land cover maps and the geometrically and radiometrically corrected FORMOSAT-2 images. This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-16-1-4023. This work was supported by the Australian Research Council under awards DE170100037 and DP140100087, and by the 2016 IBM Faculty Award (F. Petitjean). We would also like to thank Germain Forestier for his comments.

References

[1] *Additional material.* http://bit.ly/SDM2017.

[2] European Space Agency, *Sentinel 2 overview.* https://sentinel.esa.int/web/sentinel/missions/sentinel-2/overview.

[3] David Arthur and Sergei Vassilvitskii, *k-means++: The advantages of careful seeding*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[4] A Bagnall and J Lines, *An experimental evaluation of nearest neighbour time series classification. technical report# cmp-c14-01*, Department of Computing Sciences, University of East Anglia, Tech. Rep, (2014).

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited

[5] Jon Louis Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM, 18 (1975), pp. 509–517.
[6] ———, *Programming pearls*, ACM, 1986.
[7] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, *The ucr time series classification archive*, 7 2015. www.cs.ucr.edu/~eamonn/time_series_data/.
[8] Janez Demšar, *Statistical comparisons of classifiers over multiple data sets*, Journal of Machine learning research, 7 (2006), pp. 1–30.
[9] J. Inglada, M. Arias, B. Tardy, O. Hagolle, S. Valero, D. Morin, G. Dedieu, G. Sepulcre, S. Bontemps, P. Defourny, and B. Koetz, *Assessment of an operational system for crop type map production using high temporal and spatial resolution satellite optical imagery*, Remote Sensing, 7 (2015), pp. 12356–12379.
[10] J. Inglada, A. Vincent, M. Arias, and C. Marais-Sicre, *Improved early crop type identification by joint use of high temporal resolution sar and optical image time series*, Remote Sensing, 8 (2016), p. 362.
[11] Eamonn Keogh, *Welcome to the lb_keogh homepage!* www.cs.ucr.edu/~eamonn/LB_Keogh.htm.
[12] E. Keogh and C.A. Ratanamahatana, *Exact indexing of dynamic time warping*, Knowledge and Information Systems, 7 (2005), pp. 358–386.
[13] Eamonn Keogh, Li Wei, Xiaopeng Xi, Michail Vlachos, Sang-Hee Lee, and Pavlos Protopapas, *Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures*, The VLDB JournalThe International Journal on Very Large Data Bases, 18 (2009), pp. 611–630.
[14] P. Kranen and T. Seidl, *Harnessing the strengths of anytime algorithms for constant data streams*, Data Mining and Knowledge Discovery, 19 (2009), pp. 245–260.
[15] Daniel Lemire, *Faster retrieval with a two-pass dynamic-time-warping lower bound*, Pattern recognition, 42 (2009), pp. 2169–2180.
[16] Marius Muja, *Flann-fast library for approximate nearest neighbors*. www.cs.ubc.ca/research/flann/.
[17] Marius Muja and David G Lowe, *Scalable nearest neighbor algorithms for high dimensional data*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 (2014), pp. 2227–2240.
[18] F. Petitjean, G. Forestier, GI Webb, AE Nicholson, Y. Chen, and E. Keogh, *Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm*, Knowledge and Information Systems, 47 (2016), pp. 1–26.
[19] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh, *Dynamic time warping averaging of time series allows faster and more accurate classification*, in 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 470–479.
[20] F. Petitjean, J. Inglada, and P. Gançarski, *Satellite image time series analysis under time warping*, IEEE transactions on geoscience and remote sensing, 50 (2012), pp. 3081–3095.
[21] François Petitjean, Alain Ketterlin, and Pierre Gançarski, *A global averaging method for dynamic time warping, with applications to clustering*, Pattern Recognition, 44 (2011), pp. 678–693.
[22] 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*, in SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2012, pp. 262–270.
[23] Thanawin Rakthanmanon and Eamonn Keogh, *Fast shapelets: A scalable algorithm for discovering time series shapelets*, in Proceedings of the 13th SIAM international conference on data mining, SIAM, 2013, pp. 668–676.
[24] C.A. Ratanamahatana and E. Keogh, *Making time-series classification more accurate using learned constraints*, SIAM, 2004.
[25] Hiroaki Sakoe and Seibi Chiba, *A dynamic programming approach to continuous speech recognition*, in Proceedings of the seventh international congress on acoustics, vol. 3, Budapest, Hungary, 1971, pp. 65–69.
[26] ———, *Dynamic programming algorithm optimization for spoken word recognition*, IEEE transactions on acoustics, speech, and signal processing, 26 (1978), pp. 43–49.
[27] C. Sammut and G.I. Webb, eds., *Encyclopedia of Machine Learning*, Springer, Berlin, 2010.
[28] P. Schäfer, *Scalable time series classification*, Data Mining and Knowledge Discovery, (2015), pp. 1–26.
[29] Pavel Senin and Sergey Malinchik, *Sax-vsm: Interpretable time series classification using sax and vector space model*, in 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1175–1180.
[30] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh, *Experimental comparison of representation methods and distance measures for time series data*, Data Mining and Knowledge Discovery, 26 (2013), pp. 275–309.
[31] X. Xi, E. Keogh, C. Shelton, L. Wei, and C.A. Ratanamahatana, *Fast time series classification using numerosity reduction*, in Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 1033–1040.
[32] Y. Yang, G.I. Webb, K. Korb, and K-M. Ting, *Classifying under computational resource constraints: Anytime classification using probabilistic estimators*, Machine Learning, 69 (2007), pp. 35–53.
[33] L. Ye and E. Keogh, *Time series shapelets: a new primitive for data mining*, in Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, 2009, pp. 947–956.

Copyright © by SIAM
Unauthorized reproduction of this article is prohibited