Towards Time Series Classification without Human Preprocessing

Patrick Schäfer

Zuse Institute Berlin, Berlin, Germany
patrick.schaefer@zib.de

Abstract. Similarity search is a core functionality in many data mining algorithms. Over the past decade these algorithms were designed to mostly work with human assistance to extract characteristic, aligned patterns of equal length and scaling. Human assistance is not cost-effective. We propose our *shotgun distance* similarity metric that extracts, scales, and aligns segments from a query to a sample time series. This simplifies the classification of time series as produced by sensors. A time series is classified based on its segments at varying lengths as part of our *shotgun ensemble classifier*. It improves the best published accuracies on case studies in the context of bioacoustics, human motion detection, spectrographs or personalized medicine. Finally, it performs better than state of the art on the official UCR classification benchmark.

1 Introduction

Time series result from recording data over time. The task of analyzing time series data is difficult as the data may be recorded at variable lengths, and are erroneous, extraneous and highly redundant due to repetitive (sub-)structures. Application areas include ECG [4] or EEG signals, human walking motions [6], or insect wing beats [23], for example. The classification of time series has gained increasing interest over the past decade [2, 7–9, 11, 14, 16, 20, 26]. It aims at assigning a class label to a time series. For this the features (the model) to distinguish between the class labels are trained based on a labeled train dataset. When an unlabeled query time series is recorded, the trained model is applied to determine the class of the query.

Empirical evaluation suggests that classifiers based on 1-nearest-neighbour Euclidean distance (ED) or dynamic time warping (DTW) are hard to beat [2, 7]. These methods calculate the distance between two entire time series to determine their similarity. To make these applicable a lot of time and effort has to be spent by a domain expert to filter the data and extract equal-length, equal-scale, and aligned patterns. Human assistance significantly eases the subsequent data mining task both in terms of the cost of the execution time and the complexity of the algorithm. However, human assistance comes at a high price and is often too time consuming [27]. There is an over-dependence on preprocessed time series data and only few algorithms exists that deal with the data 'as is'. These algorithms are based on matching time series by their structural similarity [8, 16].

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-08979-9_18

As traditional data mining algorithms are not easily applicable to raw datasets, international competitions were staged like identifying whale calls [10], human walking motions [10] and flying insects [23].

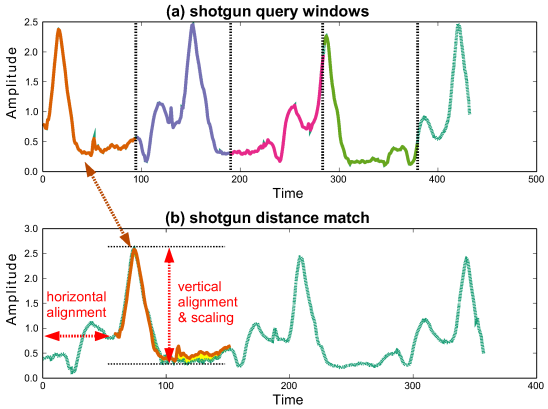


Fig. 1: Shotgun distance consists of segment extraction, horizontal and vertical alignment, and scaling.

Our work introduces a novel similarity metric for time series similarity search. It was applied as part of a contest [23]. *Shotgun distance* vertically and horizontally aligns time series segments (subsequences) of a query to a sample time series (Figure 1). Thereby it avoids preprocessing the data for alignment, scaling or length. This is achieved by breaking the query into disjoint subsequences of fixed length first. Next, each query subsequence is slid along a time series sample to find the best matching position in terms of minimizing a distance metric (horizontal alignment). These distances are aggregated. The sample that minimizes this aggregated distance is the 1-nearest-neighbor (1-NN) to a query and most similar. Normalization is applied prior to each distance computation, to provide the same vertical alignment and scaling of each subsequence. The *shotgun ensemble classifier* is based on an ensemble of 1-NN classifiers utilizing the shotgun distance at multiple subsequence lengths. Our contributions are as follows:

- Section 2 presents the motivation and related work on classifying time series.
- We introduce the shotgun distance that provides vertical scaling and horizontal alignment in Section 3.
- We present the shotgun ensemble classifier which represents a time series at multiple subsequences lengths in Section 3.4.
- Two pruning strategies are presented which significantly reduce the computational complexity by one order of magnitude in Section 3.5.
- Our shotgun ensemble classifier is significantly more accurate than state of the art on 5 case studies and the UCR benchmark datasets in Section 4.

2 Motivation & Related Work

The utility of the shotgun distance is related to the observation that a multitude of signals are composed of characteristic patterns (see Section 4.1). Consider human walking motions [6] as a concrete example. The data was captured by

recording the z-axis accelerometer values of either the right or the left toe. The difficulties in this dataset result from variable-length gait cycles, gait styles and pace due to different subjects throughout different activities. Figure 2 illustrates the walking motion of a subject, that is composed of 4 gait cycles. Classifying walking motions is difficult, as the samples are not preprocessed to have an approximate alignment, length, scale or number of gait cycles.

Shotgun distance reduces the cost-ineffective human assistance by vertically aligning and horizontally scaling the query to a sample time series. It is an analogy to *Shotgun Sequencing* [24], the process of breaking up a sequence into numerous small segments which are resembled based on overlaps. Figure 2 (bottom) illustrates the result of this process. The distance between the 4 gait cycles in the query and the sample are minimized, even though these differ in scale, have a variable length, and a phase-shift and noise occur. The quality of the shotgun distance is subject to two parameters (Figure 1):

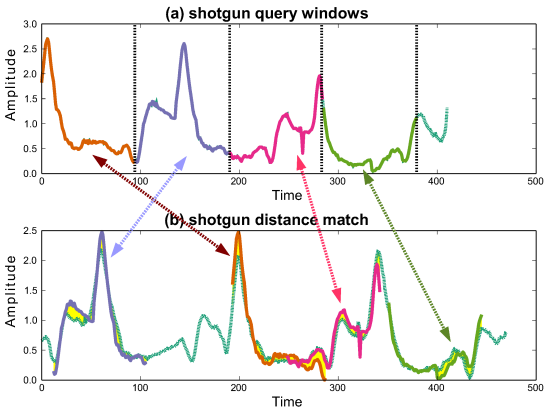


Fig. 2: Matching the gait cycles in the query to the sample is complicated due to different amplitudes, phase-shifts, variable lengths and noise.

1. *horizontal alignment* using the *window* length: an integer parameter which is limited by the length of the longest query.
2. *vertical alignment* using the *mean*: a Boolean parameter which defines if the mean should be subtracted prior to the distance calculations. The standard deviation is always normed to 1 to obtain the same scaling. Surprisingly, the mean normalization has not been considered to be a parameter before.

The *window length* parameter controls the length of the segments and depends on the length of the characteristic patterns in the dataset. Furthermore, it regulates how much information on the ordering of the values within the time series is incorporated into the matching-process. For long window lengths the whole query will be treated as a single pattern. This mostly happens with signals which were preprocessed by a human for alignment and length. In contrast, human motions contain repetitive gait cycles. Aligning any gait cycle in the query to any gait cycle in the sample is equivalent. Thus, the ordering information is less relevant, resulting in a window length that should be roughly equal to one gait cycle.

2.1 Related Work

Time series similarity search is a complex task for a computer. It is non trivial to extract a general statistical model from time series as these may show varying statistical properties with time. Classical machine learning algorithms degenerate due to the high dimensionality of the time series and noise [12]. Approaches can be characterized by (a) they try to find a similarity metric that resembles our intuition of similarity in combination with 1-NN classification (*shape-based*) or (b) they transform the data into an alternative data space to make existing data mining algorithms applicable (*structure-based*) [2, 14, 25]. The UCR time series classification datasets [11] have been established for reference [2, 7, 11, 14, 16]. *Shape-based* techniques include 1-NN Euclidean Distance (ED), or 1-NN DTW [17, 19] and are used as the reference [7]. However, shape-based techniques fail to classify noisy or long data. *Structure-based* techniques [2, 8, 14, 16, 18, 26] are based on data mining algorithms such as SVMs, decision trees, or random forests in combination with feature extraction. Feature extraction techniques include DFT [1], PLA [5], SFA [21], SAX [13], or Shapelets. By transforming time series data into an alternative space (i.e. using functional data analysis) the performance of classifiers can be improved [2]. However, the authors failed to show a significant improvement over 1-NN DTW. Shapelets classifiers [16, 18, 26] extract representative variable-length subsequences (called *shapelets*). A decision tree is build using these shapelets within the nodes of the tree and distance threshold for branching. One algorithms deals with classification on raw data [8]. The shotgun classifier is inspired by shotgun sequencing introduced to find an alignment of two DNA or protein sequences [24]. Shotgun sequencing was used to find the horizontal displacements of steel coils [15]. To find the horizontal displacement the authors use the median on the differences of the calculated starting positions for every pair of subsequences.

3 Shotgun Distance

3.1 Definitions

A time series consists of a sequence of real values:

$$T = (t_1, \ldots, t_n) \tag{1}$$

This time series is split into subsequences (time segments) using a windowing function.

Definition 1. *Windowing: A time series $T = (t_1, \ldots, t_n)$ of length n is split into fixed-length windows $S_w(a) = (t_a, \ldots, t_{a+w-1})$ with length w and offset a in T. Two consecutive windows can* overlap *within an interval of $[0, w)$. Given the overlap, there are $\frac{(n-w)}{(w-overlap)}$ windows in T:*

$$windows(T, w, overlap) = \bigcup_{i=0}^{\frac{(n-w)}{(w-overlap)}} S_w(i \cdot (w - overlap) + 1) \tag{2}$$

Algorithm 1 The shotgun distance.

```
double ShotgunDistance(query,sample,W_LEN,MEAN_NORM)
(1) totalDist = 0.0
    // for each disjoint query window
(2) for q in disjoint_windows(query,W_LEN,MEAN_NORM)
(3)   qDist = MAX_VALUE
      // search for the position that minimizes the Euclidean distance
(4)   for s in sliding_windows(sample,W_LEN,MEAN_NORM)
(5)     qDist = min(qDist,EuclideanDist(q,s))
(6)   totalDist += qDist
(7) return totalDist
```

To *vertically* align two samples, the query window and the sample window are typically z-normalized by subtracting the mean and dividing by the standard deviation:

$$\hat{\omega}(T, w, overlap) = z\_norms(windows(T, w, overlap)) \tag{3}$$

However, the mean normalization is treated as a parameter of our model and can be enabled or disabled. For example, heart beats have to be compared using a common baseline but the pitch of a bird sound can be significant for the species. Commonly, the similarity of two time series is measured using a distance metric. The shotgun distance is a distance metric that minimizes the Euclidean distance between each disjoint window in the query Q and the sliding windows in a sample S. For example, each gait cycle is slid along a longer walking motion to find the best matching positions by minimizing the Euclidean distance.

Definition 2. *Shotgun distance: the shotgun distance $D_{shotgun}(Q, S)$ between a query Q and a sample S is given by aggregating the minimal Euclidean distance $D(Q_a, S_b)$ between each disjoint query window $Q_a \epsilon \hat{\omega}(Q, w, 0)$ and each offset b in S, represented by the sliding windows $S_b \epsilon \hat{\omega}(S, w, w-1)$:*

$$D_{shotgun}(Q, S) = \sum_{a=1}^{len(\hat{\omega}(Q,w,0))} \min\{D(Q_a, S_b) \mid S_b \epsilon \hat{\omega}(S, w, w-1)\} \tag{4}$$

This definition resembles the extraction of characteristic patterns (i.e. the gait cycles), and the scaling and aligning of the patterns. The latter provides invariance to the time ordering of the patterns and allows for comparing variable length time series. The shotgun distance is equal to the Euclidean distance for n equal to w.

3.2 Shotgun Distance Algorithm

The shotgun distance in Algorithm 1 makes use of the Euclidean distance, and can be tuned by the two parameters *window length* W_LEN and *mean* MEAN_NORM (the standard deviation of q and s is always normed to 1 regardless of MEAN_NORM). It first splits the query into disjoint windows (line 2)

and searches for the position in the sample that minimizes the Euclidean distance (line 4-5). Finally, the distances are accumulated for each query window (line 6).

Complexity: The computational complexity is quadratic in the length of the time series Q and S: for each query window, all sample windows are iterated and the Euclidean distance for each pair of windows is calculated. There are $\frac{|Q|}{w}$ disjoint query windows and $|S| - w + 1$ sliding windows for window length w:

$$T(\text{Shotgun Distance}) = O\left(\underbrace{\frac{|Q|}{w}}_{\text{disjoint windows}} \cdot w \cdot \underbrace{(|S| - w + 1)}_{\text{sliding windows}}\right) \tag{5}$$

$$\text{for } n = \max(|Q|, |S|) \Rightarrow O\left(n^2 - nw\right) \tag{6}$$

Note that for large window lengths $w \sim n$ this complexity is close to linear in n (like the Euclidean distance). For small window lengths $w \ll n$ the complexity is quadratic in n^2 (like DTW).

3.3 Shotgun Classifier

The shotgun classifier is based on 1-NN classification and the shotgun distance. Given a query, the *predict*-method (Algorithm 2) searches for the 1-NN to a query within the set of samples (line 3-5). Finally, the query is labeled by the class label of the 1-NN *nn*. The *fit*-method (Algorithm 2) uses leave-one-out cross-validation (lines 4–8) to obtain the parameters that maximize the accuracy on the train samples. The accuracies for all window lengths starting from the *maxLen* (the length of the longest time series) down to *minLen* (lines 2–7) are recorded. The *MEAN_NORM*-parameter is a Boolean parameter, which is constant for a whole dataset and not set per sample.

3.4 Shotgun Ensemble Classifier

By intuition every dataset is composed of substructures at multiple window lengths caused by different walking motions, heart beats, duration of vocals, length of shapes. For example, each human may have a different length of a gait cycle. Thus, each sample is represented by a set of window lengths.

Using the *predictEnsemble*-method (Algorithm 2), a label is determined for the best window lengths. Using a constant parameter $factor \epsilon (0, 1]$ and the best accuracy *bestScore* obtained from the train samples, the best window lengths are given by: $correct > bestScore \cdot factor$ (line 3). Finally, the most frequent class label is chosen from the set of labels.

While it might seem that we add yet another parameter *factor*, the training of the shotgun ensemble classifier depends solely on the *factor* and *mean* parameters. The shotgun ensemble classifier model is derived from these two

Algorithm 2 The Shotgun Ensemble Classifier.

```
String predict(query,samples,W_LEN,MEAN_NORM)
(1) (dist, nn) = (MAX_VALUE, NULL)
(2) for sample in samples
(3)   D = ShotgunDistance(query,sample,W_LEN,MEAN_NORM)
(4)   if D < dist
(5)     (dist, nn) = (D, sample)
(6) return nn.label

String predictEnsemble(query,samples,bestScore,windows,MEAN_NORM)
    // stores for each window length a label
(1) windowLabels = []
    // determine the label for each window length
(2) for (correct, len) in windows
(3)   if (correct > bestScore*factor)
(4)     windowLabels[len] = predict(query,samples,len,MEAN_NORM)
(5) return most frequent label from windowLabels

[(int,int)] fit(samples,labels,MEAN_NORM)
(1) scores = []
    // search for best window lengths in parallel
(2) for len = maxLen down to minLen
(3)   correct = 0
(4)   for query in samples
(5)     nnLabel = predict(query,samples\{query},len,MEAN_NORM)
(6)     if (nnLabel==query.label) correct++
      // store scores for each window length
(7)   scores.push((correct, len))
(8) return scores
```

parameters using the *fit*-method, which returns the set of window scores. These scores are used as the model and to predict the label of an unlabeled query. In our experiments factors in between 0.95 to 1.0 were best throughout most datasets.

3.5 Pruning the Search Space

The rationale of search space pruning is to early abandon computations, as soon as these can not result in finding a new optimum. Previous work aims at stopping Euclidean distance calculations when the current distance exceeds the best distance found so far [16, 18, 26].

Early Abandoning: The purpose of the *ShotgunDistance*-method (Algorithm 3) is to accumulate the Euclidean distances for each query window. The Euclidean distance computations are pruned by reusing the best result *qDist* of the previous calculations (line 5). The *ShotgunDistance*-method is executed multiple times for each pair of query and sample. Passing the distance to the current calculation as *bestDist* allows for pruning calculations as soon as this *bestDist* is exceeded (line 7). Otherwise the sample is a new nearest-neighbor candidate and the distance is used to prune subsequent calls to *ShotgunDistance*. In the best case scenario, we have to compute the distance between one pair of time series and all other distance computations stop after one iteration of the for-loop in line 7.

Algorithm 3 Pruning techniques based on early abandoning.

```
double EuclideanDist(query,sample,bestDist)
(1) for i = 1 to len(query)
(2)   dist += (sample[i] - query[i])^2
(3)   if (dist > bestDist) return MAX_VALUE           // early abandoning
(4) return dist

double ShotgunDistance(query,sample,W_LEN,MEAN_NORM,bestDist)
(1) totalDist = 0
(2) for q in disjoint_windows(query,W_LEN,MEAN_NORM)
(3)   qDist = MAX_VALUE
(4)   for s in sliding_windows(sample,W_LEN,MEAN_NORM)
        // early abandoning
(5)     qDist=min(qDist,EuclideanDist(q,s,min(qDist,bestDist)))
(6)   totalDist += qDist
(7)   if (totalDist > bestDist) return MAX_VALUE      // window pruning
(8) return totalDist

String predict(q,samples,W_LEN,MEAN_NORM)
[...]
(2) for sample in samples
(3)   D = min(D,ShotgunDistance(q,sample,W_LEN,MEAN_NORM,D))
[...]
```

Algorithm 4 Use an upper bound on the current accuracy.

```
[(int,int)] fit(samples,labels,MEAN_NORM)
(1) scores = [], bestCorrect = 0
(2) for len = maxLen down to minLen
(3)   correct = 0
(4)   for q in [1..len(samples)]
(5)     nnLabel = predict(samples[q],samples\{samples[q]},len,MEAN_NORM)
(6)     if (nnLabel==samples[q].label) correct++
(7)     if (correct+(len(samples)-q)) < bestCorrect*factor
(8)       break
(9)   bestCorrect = max(bestCorrect,correct)
[...]
```

Upper Bound on Accuracy: While lower bounding on distance computations aims at reducing the complexity in the length n, we present a novel optimization that also aims at reducing the complexity in the number of samples N. For each window length, the best achievable accuracy at any point is given by:

$$\text{correct} \leq (\text{current correct} + \text{remaining samples}) = N \tag{7}$$

Thus, we do not need to obtain the exact accuracy for a window length in Algorithm 4 (lines 7–8), if the remaining samples will not result in finding a better accuracy (or at least within *factor* to the best accuracy).

4 Experimental Evaluation

The utility of the shotgun ensemble classifier is underlined by case studies and the UCR time series classification benchmark datasets [11]. Each dataset is split into two subsets: *train* and *test*. By the use of the same train/test splits the results are

Dataset	DTW	Best Rival	Shotgun Ensemble	Classifier
Personalized Medicine	62.8%	92.4% [8]	99.3%	*factor* : 1, mean:true
Walking Motions	66.2%	91% [26]	96.9%	*factor* : 0.95, mean:true
Spectrographs	71.3%	72.6% [26]	80.7%	*factor* : 0.95, mean:true
Bio Acoustics	-	93.29% [23]	92.38%	*factor* : 1, mean:true
Astronomy	90.7%	93.68% [18]	95.3%	*factor* : 0.97, mean:true

Table 1: Test accuracies on the case studies.

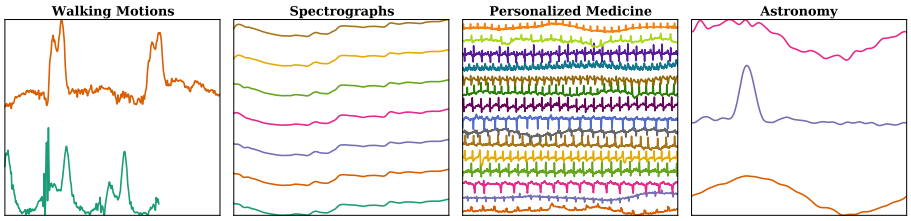


Fig. 3: A sample representing each class of the case studies: abnormal and normal walking motions, wheat spectrographs, ECG signals, and starlight curves.

comparable those previously published [2, 3, 7, 16, 18]. **In all experiments we optimized the parameters of the classifiers based on the train dataset. The optimal set of parameters is then used on the test dataset.** Our web page [22] contains a spreadsheet with all raw numbers and source codes. All benchmarks were performed on a shared memory machine running Linux with 8 Quad-Core AMD Opteron 8358 SE and Java JDK x64 1.7.

4.1 Case Studies

Astronomy / Scalability: We test the scalability using the largest dataset in the UCR time series archive [11]. It contains three types of star objects: *Eclipsed Binaries*, *Cepheids* and *RR Lyrae Variables*. The *Cepheids* and *RR Lyrae Variables* have a similar shape and are hard to separate (Figure 3 top and bottom). To test the scalability of the shotgun *fit*-method, we iteratively doubled the number of samples from 100 to 1000, each of length 1024, and measured the pruning strategies presented in this paper. Figure 4 shows that the time of the *brute force* algorithm grows quadratically to approximately 9 hours for 1000 samples. *Early abandon-*

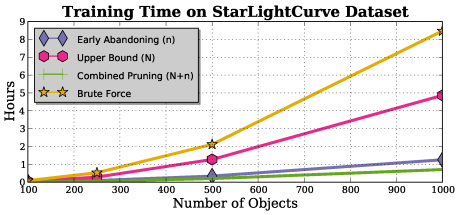


Fig. 4: The time required to execute the shotgun *fit*-method.

ing reduces this by a factor of 7 and in combination with the *upper bound* by a factor of 12 to only 41 minutes. Both pruning strategies combined significantly reduce the run-time for training when the number of samples is increased.

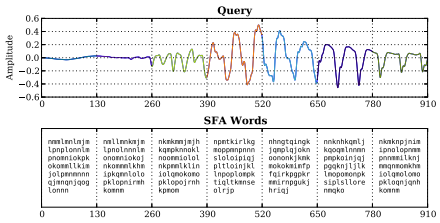
To the best of our knowledge, the highest reported test accuracy is 93.68% [18] with 52 minutes for training and 1-NN DTW scores 90.7%. The test accuracy of our shotgun ensemble classifier is 95.3% (Table 1).

Personalized Medicine: The BIDMC Congestive Heart Failure Database [4] consists of ECG recordings of 15 subjects, which suffer from severe congestive heart failures (Figure 3). The recordings contain noisy or extraneous data, when the recordings started before the machine was connected to the patient. ECG signals show a high level of redundancy due to repetitive heart beats but even a single patient can have multiple different heart beats. To deal with these distortions a classifier has to be invariant to amplitude, uniform scaling, phase shifts and occlusion. The total size of this dataset is equal to 9 million data points (10 hours sampled at at 250 Hz). We used the train/test split in [8], which selected 150 minutes for training and 450 minutes for testing and search for individual patient heart beats (15 distinct classes). There are 600 samples for training at length 3750 and 600 samples for testing at length 11250.

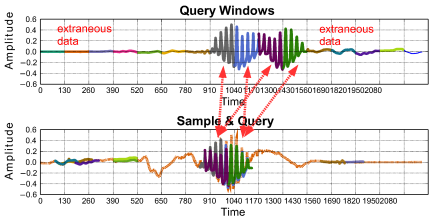
To the best of our knowledge, the best rivalling approach reported a test accuracy of 92.4% [8] and 1-NN DTW scores 62.8%. The shotgun ensemble classifier obtains a much higher test accuracy of 99.3%. This is a result of the design of the shotgun distance: ECG signals are composed of recurring patterns, which are distorted by all kinds of noise. To obtain this score, training the shotgun ensemble classifier took roughly 2 days as all window lengths have to be evaluated. Prediction on the 600 test samples took roughly 1.5 hours in total.

Human Walking Motions: The CMU [6] contains walking motions of 4 subjects. Each motion was categorized by the labels *normal walk* and *abnormal walk* (Figure 3). The data were captured by recording the z-axis accelerometer values of either the right or the left toe. The difficulties in this dataset result from variable-length gait cycles, gait styles and pace due to different subjects throughout different activities including stops and turns. To deal with these distortions, a classifier needs to be invariant to amplitudes, uniform scaling, phase shifts and occlusions. To make our results comparable to [26], we used the data provided by their first segmentation approach. The dataset contains 40 samples for training and 228 samples for testing, each of variable lengths with an upper limit of 500 points. We search for normal or abnormal walking patterns. Training the shotgun ensemble classifier took less than a minute. This results in a test classification accuracy of 96.9%, due to the repetitive nature of the data. The accuracy is significantly higher than that of the best rivalling approach in [26] with an accuracy of 91% or 1-NN DTW that scores 66.2%.

Spectrographs: *Wheat* [26] is dataset of 775 spectrographs of wheat samples grown in Canada. The data is split into 49 samples of length 1050 for training



(a) The query is cut into windows, and the SFA representations are calculated.

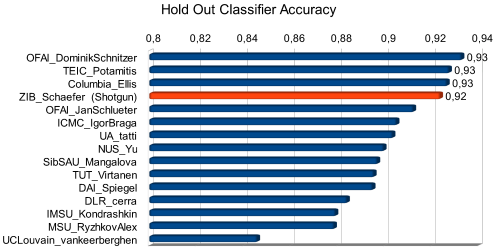


(b) An insect passes the laser multiple times, causing an *echo*. The shotgun classifier aligns these two signals.

Fig. 5

and 726 samples of length 1050 for testing. The dataset contains different wheat types like *Canada Western Red Spring, Soft White Spring* or *Canada Western Red Winter* (Figure 3). The class labels define the year in which the wheat was grown. This makes the classification problem much more difficult, as the same wheat types in different years belong to different classes. The best rivalling approach [26] reported a test accuracy of 72.6% on this dataset and 1-NN DTW obtains a test accuracy of 72.6%. Our shotgun ensemble classifier obtains a significantly higher test accuracy of 80.69%.

Bioacoustics: Producers set up traps in the field that lure and capture pests, in order to detect and count these. Manual inspection of traps is a procedure that is costly and error prone. Repeated inspection must be carried manually, sometimes in areas that are not easily accessible. A novel recording device has been recently introduced [23]. The core idea is to embed a device in an insect trap to record the fluctuations of light received by a photoreceptor as an insect passes a laser beam and partially occludes light. The samples were recorded at 16 kHz and are 1s long but the insect motion within each sample is typically only a few hundredths of a second long. The bandwidth between 0.2-4 kHz is most characteristic. The dataset D1 was collected from 5 insects, namely: *Aedes aegypti male*, *Fruit flies mixed sex*, *Culex quinquefasciatus female*, *Culex tarsalis female*, *Culex tarsalis male,* and consists of 5000 recordings for testing and 500 for training.To connect time series analysis with bioacoustics, we use Symbolic



(a) Classifier accuracies on the test dataset. ZIB_Schaefer represents the shotgun classifier.

Fig. 6

Fourier Approximation (SFA) [21]. Its symbolic and thus compact representation of a time series has shown to be capable of exact similarity search and to index terabyte-sized datasets. Our workflow consists of feature extraction and feature matching. SFA is applied to extract features, which are then passed to the shotgun classifier.

The solution utilizes SFA to reduce noise by the use of low pass filtering and quantization. The SFA transformation results in a character string (see Figure 5a). Each symbol represents an interval in the frequency domain. In particular, noise is generated by the angle and the speed of an insect passing the photoreceptor. This affects the recorded intensities. SFA's noise reduction accounts for these differences in the intensity. By introducing the shotgun classifier for feature matching, we obtain invariance to the time of the insect passage. Shotgun distance further deals with outliers like multiple insects passing the laser within a short time frame (see Figure 5b). For the sake of brevity, the interested reader is referred to [20] for details on the approach and SFA. Using a small window length of 130 and a small number of SFA symbols of 22 performed best. Our approach scored within 1 percentage point of the best approach applied in the contest (Figure 6a). This proves that time series analysis is applicable to computational bioacoustics.

4.2 UCR Classification Benchmark Datasets

We used a standardized benchmark [11] to evaluate the classifiers. Each dataset consists of a train and a test subset. The shotgun ensemble classifier is compared to state of the art time series classifiers like shapelets [16], fast shapelets [18], 1-NN classifiers using Euclidean distance or dynamic time warping (DTW) with the optimal warping window, support vector machines (SVM) with a quadratic and cubic kernel, and a tree based ensemble method (random forest). We followed the setup in [16, 18]. The authors in [2] used data transformations (i.e. functional data analysis) to improve classifiers performance. However, they failed to show a significant improvement over 1-NN DTW. We omit data transformations prior to classification for the sake of brevity. The scatter plots in Figure 7 show a pairwise comparison of each classifier with the shotgun classifier. Each dot represents the test accuracies of the two classifiers on one concrete dataset. Dots below the diagonal line indicate that the shotgun ensemble classifier is more accurate.

The shotgun ensemble classifier is better than fast shapelets and shapelets on the majority of the datasets. We conclude that this is a result of the sensitivity to the overfitting of the shapelet classifiers and the decision tree in particular, where the difference between the train and test accuracy makes up for up to 50 percentage-points. In contrast the shotgun classifier is more robust towards overfitting (see web page [22]). 1-NN DTW is the established benchmark classifier [7]. Our shotgun ensemble classifier is better than 1-NN DTW or 1-NN Euclidean distance on the majority of datasets by a large margin in terms of accuracy. DTW provides invariance to local scaling (time warping). Shotgun distance does not explicitly provide this invariance. However, the results imply that either (a) most UCR datasets do not require local scaling or (b) the shotgun

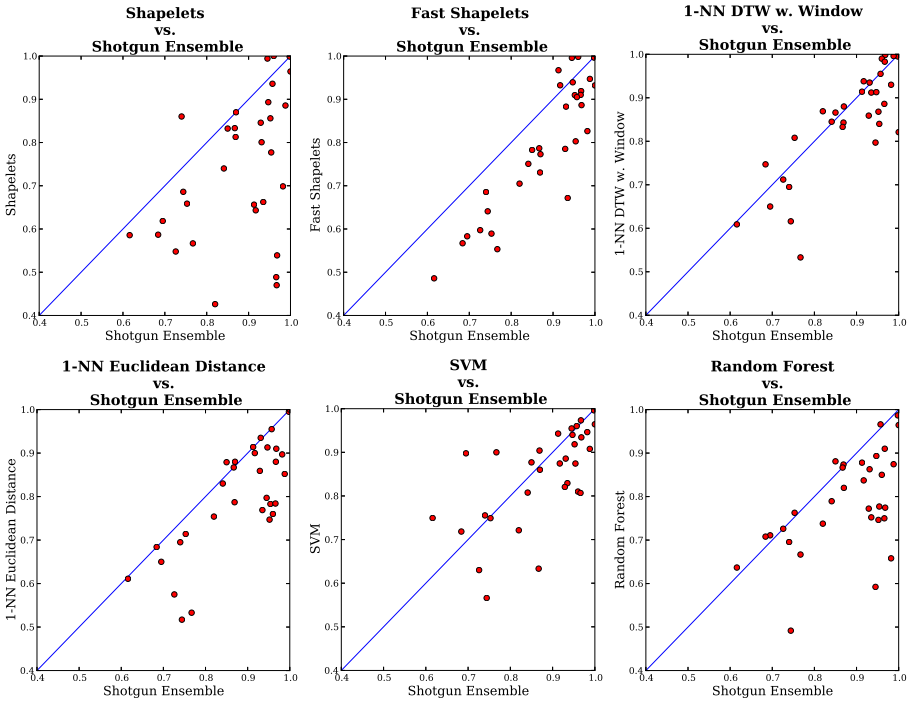


Fig. 7: Accuracy of the shotgun ensemble classifier vs. rivalling approaches.

distance provides some local scaling invariance. This will be part of future work. The shotgun distance is equal to Euclidean distance, if the window length is equal to the query length. Thus, the shotgun ensemble classifier performs better than the 1-NN Euclidean distance. SVMs and the shotgun ensemble classifier complement each other quite well as one classifier is good on a dataset in which the other performs badly. So, at least for datasets which were preprocessed for approximate alignment and fixed length, the choice of the classifier depends on the dataset. When comparing the shotgun ensemble classifier with random forests, the results suggest that the former is more accurate by a large margin. Note that the UCR datasets were preprocessed for approximate alignment and length. Still our shotgun classifier performs significantly better than rivalling state of the art classifiers on a majority of datasets.

5 Conclusion

The time series classification task is complicated by noise, dropouts, subtle distinctions, variable lengths or extraneous data. The shotgun distance is a novel distance measure based on the characteristic patterns in time series. Shotgun distance utilizes time segments which are vertically and horizontally aligned and

scaled between the query and a sample, and thereby simplifying the preprocessing. Based on an ensemble of 1-nearest-neighbor classifiers the shotgun ensemble classifier is presented. To deal with the increased complexity, two pruning strategies for the length and the number of time series are presented. This reduces the computational complexity by one order of magnitude. The experimental evaluation shows that the shotgun ensemble classifier performs better than rivalling methods in the context of computational bioacoustics, human motion detection, spectrographs, astronomy, or personalized medicine. This is underlined by the best classification accuracy on the UCR time series classification datasets.

Acknowledgements. The author would like to thank C. Eichert-Schäfer, F. Schintke, F. Wende and S. Dreßler for their comments and the dataset owners.

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. Foundations of Data Organization and Algorithms 730, 69–84 (1993)
2. Bagnall, A., Davis, L.M., Hills, J., Lines, J.: Transformation Based Ensembles for Time Series Classification. In: SDM. vol. 12, pp. 307–318. SIAM (2012)
3. Batista, G., Wang, X., Keogh, E.J.: A Complexity-Invariant Distance Measure for Time Series. In: SDM. vol. 11, pp. 699–710. SIAM / Omnipress (2011)
4. BIDMC: `http://www.physionet.org/physiobank/database/chfdb/`
5. Chen, Q., Chen, L., Lian, X., Liu, Y., Yu, J.X.: Indexable PLA for Efficient Similarity Search. In: VLDB. pp. 435–446. ACM (2007)
6. CMU Graphics Lab Motion Capture Database: `http://mocap.cs.cmu.edu/`
7. Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., Keogh, E.: Querying and mining of time series data: experimental comparison of representations and distance measures. Proceedings of the VLDB Endowment 1(2), 1542–1552 (2008)
8. Hu, B., Chen, Y., Keogh, E.: Time Series Classification under More Realistic Assumptions. In: SDM. pp. 578–586. SIAM (2013)
9. Jeong, Y., Jeong, M.K., Omitaomu, O.A.: Weighted dynamic time warping for time series classification. Pattern Recognition 44(9), 2231–2240 (2011)
10. Kaggle: Go from Big Data to Big Analytics: `https://www.kaggle.com`
11. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: UCR Time Series Classification/Clustering Homepage, `http://www.cs.ucr.edu/~eamonn/time_series_data`
12. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. In: KDD. pp. 102–111. ACM (2002)
13. Lin, J., Keogh, E.J., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Mining and Knowledge Discovery 15(2) (2007)
14. Lin, J., Khade, R., Li, Y.: Rotation-invariant similarity in time series using bag-of-patterns representation. J. Intell. Inf. Syst. 39(2), 287–315 (2012)
15. Lipowsky, C., Dranischnikow, E., Göttler, H., Gottron, T.: Alignment of Noisy and Uniformly Scaled Time Series. In: DEXA. pp. 675–688. Springer (2009)
16. Mueen, A., Keogh, E.J., Young, N.: Logical-shapelets: an expressive primitive for time series classification. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1154–1162. KDD '11, ACM (2011)

17. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. pp. 262–270. ACM, ACM (2012)
18. Rakthanmanon, T., Keogh, E.: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In: SDM. pp. 668–676. SIAM (2013)
19. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust., Speech, Signal Processing (1), 43–49 (1978)
20. Schäfer, P., Dreßler, S.: Shooting Audio Recordings of Insects with SFA. In: to appear, AmiBio Workshop, Bonn, Germany (2013)
21. Schäfer, P., Högqvist, M.: SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In: Rundensteiner, E.A., Markl, V., Manolescu, I., Amer-Yahia, S., Naumann, F., Ari, I. (eds.) EDBT. pp. 516–527. ACM (2012)
22. Shotgun Distance Webpage: `http://www.zib.de/patrick.schaefer/shotgun`
23. UCR Insect Contest: (2012), `http://www.cs.ucr.edu/~eamonn/CE`
24. Venter, J.C., Others: The Sequence of the Human Genome. Science 291(5507), 1304–1351 (2001)
25. Warren Liao, T.: Clustering of time series data—a survey. Pattern Recognition 38(11), 1857–1874 (2005)
26. Ye, L., Keogh, E.J.: Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. DMKD 22(1-2), 149–182 (2011)
27. Zhang, S., Zhang, C., Yang, Q.: Data Preparation for Data Mining. Applied Artificial Intelligence 17(5-6), 375–381 (2003)