Similarity Forest for Time Series Classification

Tomasz Górecki, Maciej Łuczak, and Paweł Piasecki

Abstract The idea of similarity forest comes from Sathe and Aggarwal [19] and is derived from random forest. Random forests, during already 20 years of existence, proved to be one of the most excellent methods, showing top performance across a vast array of domains, preserving simplicity, time efficiency, still being interpretable at the same time. However, its usage is limited to multidimensional data. Similarity forest does not require such representation – it is only needed to compute similarities between observations. Thus, it may be applied to data, for which multidimensional representation is not available. In this paper, we propose the implementation of similarity forest for time series classification. We investigate 2 distance measures: Euclidean and dynamic time warping (DTW) as the underlying measure for the algorithm. We compare the performance of similarity forest with 1-nearest neighbor and random forest on the UCR (University of California, Riverside) benchmark database. We show that similarity forest with DTW, taking into account mean ranks, outperforms other classifiers. The comparison is enriched with statistical analysis.

Keywords: time series, time series classification, random forest, similarity forest

Tomasz Górecki (✉)
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Uniwersytetu Poznańskiego 4, Poznań, Poland, e-mail: `tomasz.gorecki@amu.edu.pl`

Maciej Łuczak
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Uniwersytetu Poznańskiego 4, Poznań, Poland, e-mail: `maciej.luczak@amu.edu.pl`

Paweł Piasecki
Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Uniwersytetu Poznańskiego 4, Poznań, Poland, e-mail: `pawel.piasecki@amu.edu.pl`

P. Brito et al. (eds.), *Classification and Data Science in the Digital Age*,
Studies in Classification, Data Analysis, and Knowledge Organization,
https://doi.org/10.1007/978-3-031-09034-9_19

1 Introduction

Time series classification is a well-developing research field, that gained much attention from researchers and business during the last two decades apparently by the fact that more and more data around us seems to be located in the time domain – and thus fulfilling the definition of time series. Predictive maintenance [18], quality monitoring [22], stock market analysis [20] or sales forecasting [17] are just a few exemplar nowadays problems where time series are indeed present. The reason why we usually apply to time series different methods from regular (non-time series) data is the fact, that time series are ordered in time (or some other space with ordering) and it is beneficial to use the information conveyed by the ordering.

In recent years, one could observe many advances on the field of time series classification. In 2017, Bagnall et al. presented a comprehensive comparison of time series classification algorithms [2], showing that despite there are dozens of far more complex methods, 1-Nearest Neighbour (1NN) [6, 11] coupled with DTW [3] distance constitutes a good baseline. In fact, it has been outperformed by several classifiers, with Collective of Transformation Ensembles (COTE) [1] as the most efficient one. Furthermore, COTE was extended with Hierarchical Vote system, first to HIVE-COTE [13] and then finally to HIVE-COTE 2.0 [15] – a current state of the art classifier for time series. In general, the success of COTE-family classifiers is based on the observation, that in the case of time series it is highly beneficial to use different data representations. For example, HIVE-COTE 1.0 utilizes five ensembles based on different data transformation domains. However, a common criticism of such an approach is its time complexity. In the case of HIVE-COTE, it equals $O(n^2l^4)$, where n is a number of observations and l is a length of series. Another drawback, especially significant for practitioners is the complex structure of the model ensembles that makes it hard to use HIVE-COTE without spending a decent amount of time studying its components beforehand.

As an alternative to such complex models may be trying to achieve possibly slightly worse performance in favour of model simplicity and reduced computation time. A group of classifiers that seems to hold a great potential are those inspired by Random Forest (RF) [4]. This already 20-years old algorithm remains in the classifiers' forefront, showing extremely good performance and robustness across multiple domains. Fernandez-Delgado et al. [10] performed a comparison of 179 classifiers on 121 non-time series data sets originated from UCI Machine Learning Repository [9], concluding RF to be the most accurate one. Unfortunately, the usage of RF is essentially limited to multidimensional data, as they sample features from original space while creating each node of decision trees.

In this paper, we propose a method for extending RF to work with time series using similarity forests (SF). We significantly extend the applicability of the RF method to time series data. Furthermore, the approach even outperforms traditional classifiers for time series. The main goal of this paper is to enrich the pool of time series classifiers by Similarity Forest for time series classification. SF was initially proposed by Sathe and Aggarwal in 2017 [19], as a method extending Random Forests to deal with arbitrary data sets, provided that we are able to compute similarities between observations. We would like to implement and tune the method to time series data. We investigate the performance of the model using two distance measures (the algorithm's hyper-parameter): Euclidean and DTW. Also, a comparison with other selected time series classifiers is provided. We compare its performance against 1NN-ED, 1NN-DTW and RF.

The rest of the paper is structured as follows. In Section 2, we provide details of similarity forest and we give more details about random forests. Additionally, we discuss how similarity forest is related to random forest. Section 3 describes data sets that we used and the comparison methodology. The corresponding results are presented in Section 4. Finally, in Section 5 we give a brief summary of our research.

2 Classification Methods Used in Comparison

In the paper, we compare the standard random forest and the similarity forest with the distance measure: ED (Euclidian distance) and DTW (dynamic time warping distance). As benchmark methods, we also use the nearest neighbor method (1NN) with distance measure ED and DTW. 1NN-ED and 1NN-DTW are very common classification methods for time series classification [2]. For a review of these methods refer to [14].

2.1 General Method of Random Forest Construction

Random forest consists of random decision trees. For the construction of a random forest we usually take decision trees as simple as possible — without special criteria for stopping, pruning, etc.

When building a decision tree, we start at a node N, which contains the entire data set (bootstrap sample). Then, according to an established criterion, we split the node N into two subnodes N_1 and N_2. In each subnode there are data subsets of the data set from node N. We make this split in a way that is optimal for a given split method. In each node, we write down how the split occurred. Then, proceeding recursively, we split next nodes into subnodes until the stop criterion occurs. In our case we take the simplest such criterion, namely we stop the split of a given node when only elements of the same class are included in a node. We call such a node a leaf and assign it a label which elements of the node (leaf) have.

Having built a tree, we can now use it (in the testing phase) to classify a new observation. We pass this observation through the trained tree — starting from the node N selecting each time one of the subnodes, according to the condition stored in the node. We do this until we reach one of the leaves, and then we assign the test observation to the class of the leaf.

Now, constructing the random forest, we collect a certain number of decision trees, train them independently according to the above method and, in the test phase, use each of the trees to test new observation. Thus, each tree assigns a label to the test observation. The final label (for the entire forest) we construct by voting, we choose the most frequently appearing label among the decision trees.

2.2 Classical Random Forest

To create a (classical) random tree and a random forest [4], we proceed as described above using the following node split method:

To obtain split conditions for a single tree, we select randomly a certain number of features ($\sqrt{k}$ for classification, k — number of features), and for each feature we create a feature vector (column, variable) made of all elements of the data set (bootstrap sample). For a given feature vector (variable), we determine a threshold vector. First, we sort values of the feature vector (uniquely — without repeating values). Let us name this sorted feature vector as $\boldsymbol{V} = (V_1, V_2, \dots)$. Then we take the values of the split as means of successive values of the vector $\boldsymbol{V}$:

$$v_i = \frac{V_i + V_{i+1}}{2} \quad i = 1, 2, \dots. \tag{1}$$

Each splitting value divides the data set in node N into two subsets — the one (left) in which we have elements with feature values smaller than v_i and the second (right) with other elements. Then we check the quality of such a split.

The splitting point is chosen such that it minimizes the Gini index of the children nodes. If p_1, $p_2 \dots p_c$ are the fractions of data points belonging to the c different classes in node N, then the Gini index of that node is given by: $G(N) = 1 - \sum_{i=1}^{c} p_i^2$.

Then, if the node N is split into two children nodes N_1 and N_2, with n_1 and n_2 points, respectively, the Gini quality of the children nodes is given by:

$$GQ(N_1, N_2) = \frac{n_1 G(N_1) + n_2 G(N_2)}{n_1 + n_2}.$$

Quality of the split is given by: $GQ(N) = G(N) - GQ(N_1, N_2)$.

2.3 Similarity Forest

The similarity forest [19] differs from the ordinary (classical) random forest only in the way we split nodes of trees. Instead of selecting a certain number of features, we select randomly a pair of elements e_1, e_2 with different classes. Then, for each element e of the subset of elements in a given node, we calculate the difference of the squared distances to the elements e_1 and e_2:

$$w(e) = d(e, e_1)^2 - d(e, e_2)^2,$$

where d is any fixed distance measure of the elements of the data set. We sort the vector $\boldsymbol{w}$ uniquely (without duplicates) creating the vector $\boldsymbol{V}$ and continue as for the classical decision tree. We calculate values of the split v_i (1), calculate the quality of the node split using the Gini index (2.2) and choose the best split. In the learning phase, we remember in each node how the optimal split occurred (elements e_1, e_2, $w(e)$). In the learning phase, in each node we write down the optimal split — elements e_1, e_2, and value $w(e)$).

2.4 Random Forest vs Similarity Forest

The difference between a classical random tree and a similarity tree is that instead of selecting $\sqrt{k}$ of the features, we select only one pair of elements e_1, e_2. Generally, we have much fewer possible node splits, which has a very good effect on the computation time.

The second important difference is that in the similarity tree we use any distance measure between elements of the data set. Therefore, we can use distance measures specific to a data set. For example, for time series we can use the DTW distance, much better suited for calculating the distance between time series, instead of the Euclidean distance.

3 Experimental Setup

We investigated the performance of similarity forest on UCR time series repository [7] (128 data sets). The latest update of the UCR database introduced several data sets with missing observations and uneven sample lengths. However, the repository includes a standardized version of the database without these impediments, and that is the version we used.

All data sets are split into a training and testing subset, and all parameter optimization is conducted on the training set only. We combined both parts and in the next step, we used 100 random train/test splits.

4 Results

The error rates for each classifier can be found on the accompanying website[1]. In the Table 1 we show a short summary of results, including a number of wins (draw is not counted as a win) and mean ranks. Taking into account mean ranks, SF-DTW is the best classifier, sightly ahead of RF (mean ranks correspondingly equal 2.64

[1] `https://github.com/ppias/similarity_forest_for_tsc`

Table 1 Number of wins (clearly wins) and mean ranks for examined methods.

Method	1NN-ED	1NN-DTW	RF	SF-ED	SF-DTW
Wins	12	28	**38**	10	31
Mean rank	3.59	2.89	2.69	3.19	**2.64**

and 2.89). Figure 1 demonstrates comparison of error rates and ranks for classifiers. These results lead to a conclusion that even though there is no clear winner, the top efficient distances are dominated by RF and SF-based classifiers. Figure 2 shows scatter plots of errors for pairs of classifiers.

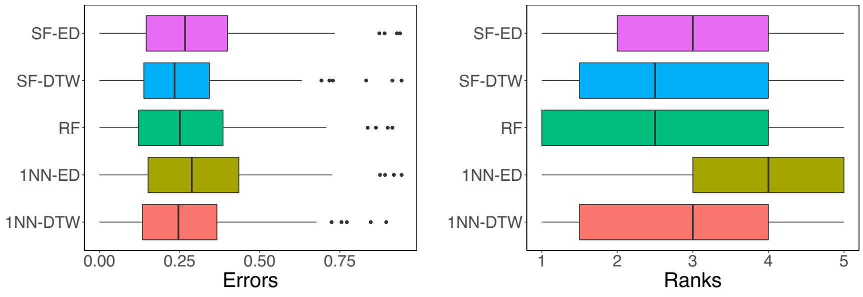

Fig. 1 Comparison of error rates and ranks.

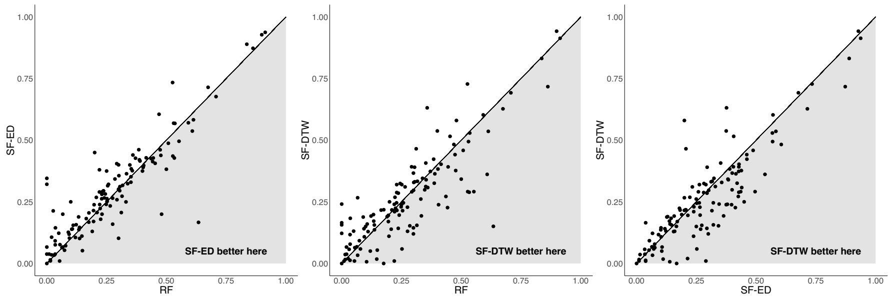

Fig. 2 Comparison of error rates.

To identify differences between the classifiers, we present a detailed statistical comparison. In the beginning, we test the null hypothesis that all classifiers perform the same and the observed differences are merely random. The Friedman test with Iman & Davenport extension is probably the most popular omnibus test, and it is usually a good choice when comparing different classifiers [12]. The p-value from this test is equal to 0. The obtained p-value indicates that we can safely reject the null hypothesis that all the algorithms perform the same. We can therefore proceed with the post-hoc tests in order to detect significant pairwise differences among all of the classifiers.

Demšar [8] proposes the use of the Nemenyi's test [16] that compares all the algorithms pair-wise. For a significance level α the test determines the critical difference (CD). If the difference between the average ranking of two algorithms is greater than CD the null hypothesis that the algorithms have the same performance is rejected. Additionally, Demšar [8] creates a plot to visually check the differences, the CD plot. In the plot, those algorithms that are not joined by a line can be regarded as different.

In our case, with a significance of $\alpha = 0.05$ any two algorithms with a difference in the mean rank above 0.54 will be regarded as non equal (Figure 3). We can see that we have three groups of methods. In the first group we have SF-DTW, RF and 1NN-DTW, in the second we have RF, 1NN-DTW and SF-ED and in the last group we have SF-ED and 1NN-ED. Unfortunately, groups are not disjoint. The first group is the group with the highest accuracy of classification. Hence, SF-DTW does not statistically outperform RF. However, we can recommend it over RF because of statistically the same quality and much better computational properties.

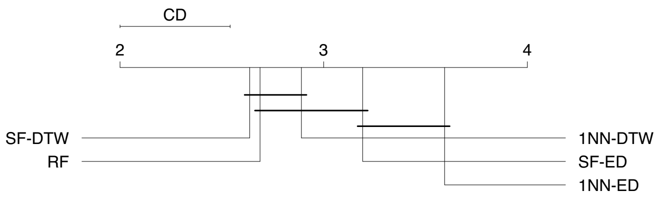

Fig. 3 Critical difference plot.

5 Conclusions

Our contribution is to implement similarity forest for time series classification using two distance measures: Euclidean and DTW. Comparison based on the recently updated UCR data repository (128 data sets) was presented. We showed that SF-DTW outperforms other classifiers, including 1NN-DTW which has been considered as a strong baseline hard to beat for years. The statistical comparison showed, that RF and SF-DTW are statistically insignificantly different, however taking into account mean ranks the latter one is the best one.

There are many improvements that could be applied to the implementation that we propose. For example, we could test other distance measures such as LCSS [21] or ERP [5] that were successfully used in time series tasks. Another idea could be to investigate the usage of boosting algorithm.

Acknowledgements The research work was supported by grant No. 2018/31/N/ST6/01209 of the National Science Centre.

References

1. Bagnall, A., Lines, J., Hills, J., Bostrom A.: Time-series classification with COTE: The collective of transformation-based ensembles. IEEE Trans. on Knowl. and Data Eng. **27**, 2522–2535 (2015)
2. Bagnall, A., Lines, J., Bostrom, A., Large J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. and Knowl. Discov. **31**, 606–660 (2017)
3. Berndt, D. J., Clifford, J.: Using dynamic time warping to find patterns in time series. Proc. of the 3rd Int. Conf. on Knowl. Discov. and Data Min., pp. 359–370 (1994)
4. Brieman, L.: Random forests. J. Mach. Learn. Arch. **45**, 5–32 (2001)
5. Chen, L., Ng, R.: On the marriage of L_p-norms and edit distance. Proc. of the 30th Int. Conf. on Very Large Data Bases **30**, pp. 792–803 (2004)
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. on Inf. Theor. **13**, 21–27 (1967)
7. Dau, H.A., Keogh, E., Kamgar, K., Yeh, Chin-Chia M., Zhu, Y.,Gharghabi, S., Ratanamahatana, C.A., Yanping, C., Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., Hexagon-ML: The UCR time series classification archive (2019) `https://www.cs.ucr.edu/\string~eamonn/time\_series\_data\_2018`
8. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. of Mach. Learn. Res. **7**, 1–30 (2006).
9. Du,a D., Graff, C.: UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`
10. Fernandez-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems?. J. of Mach. Learn. Res. **15**, 3133–3181 (2014)
11. Fix, E, Hodges, J. L.: Discriminatory analysis: nonparametric discrimination, consistency properties. Techn. Rep. **4**, (1951)
12. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental Analysis of Power. Inf. Sci. **180**, 2044–2064 (2010)
13. Lines, J., Taylor S., Bagnall, A.: HIVE-COTE: The hierarchical vote collective of transformation based ensembles for time series classification. IEEE Int. Conf. on Data Min., pp. 1041–1046 (2016)
14. Maharaj, E. A., D'Urso, P., Caiado, J.: Time Series Clustering and Classification. Chapman and Hall/CRC. (2019)
15. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.: HIVE-COTE 2.0: a new meta ensemble for time series classification. (2021) `https://arxiv.org/abs/2104.07551`
16. Nemenyi, P.:Distribution-free multiple comparisons. PhD thesis at Princeton University (1963)
17. Pavlyshenko, B. M.: Machine-learning models for sales time series forecasting. Data **4**, 15 (2019)
18. Rastogi, V., Srivastava, S., Mishra, M., Thukral, R.: Predictive maintenance for SME in industry 4.0. 2020 Glob. Smart Ind. Conf., pp. 382–390 (2020)
19. Sathe, S., Aggarwal, C. C.: Similarity forests. Proc. of the 23rd ACM SIGKDD, pp. 395–403 (2017)
20. Tang, J., Chen, X.: Stock market prediction based on historic prices and news titles. Proc. of the 2018 Int. Conf. on Mach. Learn. Techn., pp. 29–34 (2018)
21. Vlachos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. Proc. 18th Int. Conf. on Data Eng., pp. 673–684 (2002)
22. Wuest, T., Irgens, C., Thoben, K. D.: An approach to quality monitoring in manufacturing using supervised machine learning on product state data. J. of Int. Man. **25**, 1167–1180 (2014)