

A multi-measure nearest neighbor algorithm for time series classification

Original

A multi-measure nearest neighbor algorithm for time series classification / Fabris, F., Drago, I., Varejão, F.M.. - ELETTRONICO. - 5290:(2008), pp. 153-162. (11th Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008 Lisbon, Portugal 2008) [10.1007/978-3-540-88309-8_16].

Availability:

This version is available at: 11583/2659199 since: 2016-12-13T22:17:23Z

Publisher:

Springer

Published

DOI:10.1007/978-3-540-88309-8_16

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

A Multi-Measure Nearest Neighbor Algorithm for Time Series Classification

Fábio Fabris, Idilio Drago, and Flávio M. Varejão

Federal University of Espírito Santo, Computer Science Department,
Goiabeiras, 29060900, Vitória-ES, Brazil
{ffabris, idrago, fvarejao}@inf.ufes.br

Abstract. In this paper, we have evaluated some techniques for the time series classification problem. Many distance measures have been proposed as an alternative to the Euclidean Distance in the Nearest Neighbor Classifier. To verify the assumption that the combination of various similarity measures may produce a more accurate classifier, we have proposed an algorithm to combine several measures based on weights. We have carried out a set of experiments to verify the hypothesis that the new algorithm is better than the classical ones. Our results show an improvement over the well-established Nearest-Neighbor with DTW (Dynamic Time Warping), but in general, they were obtained combining few measures in each problem used in the experimental evaluation.

Key words: Data Mining, Machine Learning, Time Series Classification, Multi-Measure Classifier

1 Introduction

A time series is a sequence of data points taken at regular intervals. Supervised classification is defined as the task of assigning a label to cases, based on the information learned from examples with known labels. Special algorithms are needed when temporal features represent the examples. The Nearest Neighbor Algorithm with some specialized measure has been the main approach to deal with this kind of classification problem. Several measures have been proposed to this task throughout the years. [1, 2] describe some of them without any practical accuracy evaluation. [3] evaluates empirically a set of measures and shows poor classification results in a couple of time series classification problems. [4] shows an extensive experimental evaluation about DTW and concludes that the measure is superior to the basic Euclidean.

In this work, we evaluated if combining measures can improve classification accuracy. We proposed a new heuristic based on weights to combine measures in the nearest neighbor decision rule and carried out a set of experiments to verify our algorithm.

In the next section, we briefly describe the Nearest Neighbor Algorithm and some similarity measures commonly used. Next, we describe the algorithm proposed to search the best weights combination. In Section 4, we showed the test-

ing method, the data sets and our experimental results. Finally, in Section 5, we pointed out our conclusions.

2 1NN Algorithm and Similarity Measures

The traditional approach to classify a given time series uses the Nearest Neighbor Algorithm (1NN) with some similarity measure suitable for temporal data. In this section, we present the nearest neighbor decision rule and several similarity measures that are able to work with time series.

2.1 1NN Algorithm

In the context of classification, the 1NN Algorithm was first introduced by [5]. The algorithm idea rests in the fact that close samples (given some similarity measure) should belong to the same class if they are well distributed and there is enough correlation between the time series and the classes.

We can define formally the 1NN Algorithm given $S = \{(\mathbf{x}_1, \theta_1), \dots, (\mathbf{x}_n, \theta_n)\}$ as a set of pairs containing n time series \mathbf{x}_i and their classes θ_i . For the sake of our discussion, $\theta_i \in \{\theta^1, \dots, \theta^c\}$. We wish to ascertain the class of a new sample when introduced its correspondent \mathbf{x} . We call $\mathbf{x}' \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ the nearest neighbor of \mathbf{x} if

$$\delta(\mathbf{x}', \mathbf{x}) = \min \delta(\mathbf{x}_i, \mathbf{x}) \quad i = 1, 2, \dots, n, \quad (1)$$

where $\delta(\mathbf{y}, \mathbf{z})$ measures the similarity between two time series \mathbf{y} and \mathbf{z} . The θ of the new sample is set to its nearest neighbor class. If more than one such neighbor exist, θ is set to the most frequent class.

2.2 Similarity Measures

In order to classify an unlabeled time series using the 1NN Algorithm, we must use some kind of similarity measure $\delta(\mathbf{y}, \mathbf{z})$. Many measures have been proposed throughout the years. In this section, we shall expose a brief description of the most used ones together with some well-known remarks about them.

Euclidean Distance Probably the first choice of measurement between two time series \mathbf{y} and \mathbf{z} , both with size d , is to consider the series as a vector in a d -dimensional space and ascertain the Euclidean Distance between them. [6] defines the distance as

$$\delta(\mathbf{y}, \mathbf{z}) = [(\mathbf{y} - \mathbf{z})^t(\mathbf{y} - \mathbf{z})]^{1/2} = \left[\sum_{i=1}^d (y_i - z_i)^2 \right]^{1/2}. \quad (2)$$

There are several drawbacks using Euclidean Distance we have to consider. For instance, the distance is very sensitive to noise, translations, scaling, and small phase differences [7]. To solve vertical translation and scaling problems, an accepted solution is to normalize the curves before measuring [3]. [6] shows many other similarity measures, for example Manhattan Distance, however the same noise, scaling, translation and phase difference problems apply to them.

Edit Distance The Edit Distance [8] is defined as the minimum set of operations required to change a string into another. The allowed operations are characters deletion, insertion, and substitution. In order to use the Edit Distance, we first have to discretize the data to create a finite set of symbols. [9] defines the Edit Distance of two strings \mathbf{y} and \mathbf{z} as follows:

$$\mathcal{D}(i, 0) = s_w \times i \quad (3)$$

$$\mathcal{D}(0, j) = s_w \times j \quad (4)$$

$$\mathcal{D}(i, j) = \min \begin{cases} \mathcal{D}(i, j-1) + d_w \\ \mathcal{D}(i-1, j) + d_w \\ \mathcal{D}(i-1, j-1) + t(i, j) \end{cases} \quad (5)$$

where $\mathcal{D}(i, j)$ means the minimum operation cost needed to change the first i symbols of the string \mathbf{y} into the first j symbols of the string \mathbf{z} , $t(i, j)$ equals m_w (matching weight) if $y_i = z_j$ and s_w (substitution weight) otherwise, and finally d_w means the character insertion or deletion weight.

In particular, we are interested in the original Edit Distance formulation, with $m_w = 0$ and $s_w = d_w = 1$. The Edit Distance has two main advantages over Euclidean Distance: the character deletion and insertion remove distortions in the series and correct local misalignment; the discretizing process acts as a noise filter.

A common variation of the Edit Distance ($s_w = 1$, $d_w = \infty$, and $m_w = 0$) is the Hamming Distance [10], defined as the number of positions with different symbols in two strings. This variation does not keep the desirable property of distortion removal. The noise filter, however, is still present since the data must be discretized before Hamming Distance calculation.

Another common variation is the *Longest Common Subsequence* (LCSS [9]), defined as the size of the longest ordered sequence of symbols (not necessarily contiguous) appearing in both sequences \mathbf{y} and \mathbf{z} . We can define LCSS as

$$\mathcal{L}(i, j) = \begin{cases} \mathcal{L}(i-1, j-1) + 1 & y_i = z_i \\ \mathcal{L}(i-1, j) & \mathcal{L}(i-1, j) \geq \mathcal{L}(i, j-1) \\ \mathcal{L}(i, j-1) & \text{otherwise} \end{cases} \quad (6)$$

where $\mathcal{L}(i, 0) = \mathcal{L}(0, j) = 0$, and use the LCSS as a similarity measure by the relation

$$\delta(\mathbf{y}, \mathbf{z}) = d - \mathcal{L}(d, d) \quad (7)$$

where d is the series size. If we consider the Edit Distance formulation with $s_w = \infty$, $d_w = 1$, and $m_w = 0$, [11] shows the following relation

$$\mathcal{D}'(d, d) = 2[d - \mathcal{L}(d, d)] \quad (8)$$

where $\mathcal{D}'(d, d)$ is the Edit Distance particular case.

Dynamic Time Warping Dynamic Time Warping (DTW) is often used to classify continuous time series [12,4]. The goal is to calculate the minimum

distance (often Euclidean) between two series after aligning them. The concept is similar to Edit Distance's but while the Edit Distance only outputs the number of operations made in a reduced set of symbols, DTW outputs the actual distance between the aligned curves. Therefore, it appears that DTW is more prone to deal with time series because no information is lost. On the other hand, we probably do not get the desirable noise reduction from discretization.

The recurrence relation to calculate DTW Distance of two series \mathbf{y} and \mathbf{z} can be written as

$$DTW(i, j) = \gamma(y_i, z_j) + \min \begin{cases} DTW(i, j - 1) \\ DTW(i - 1, j) \\ DTW(i - 1, j - 1) \end{cases} \quad (9)$$

where $DTW(i, j)$ is the aligned distance of the first i points in \mathbf{y} and the first j points in \mathbf{z} . $\gamma(y_i, z_j)$ is the coordinate distance of points y_i and z_j .

DTW has an integer non-negative parameter r (also known as *warping window*) that fixes the greatest modular difference between i and j . If $r = 0$ we are merely calculating the Euclidean Distance of two series. Other values of r limit the maximum possible distortion allowed to match the series.

The value of r affects the classification performance. If r is too big, the alignment could distort too much the series involved, resulting in poor classification performance (see [4]). We can extend the concept of r to Edit Distance and its variations as well. We have considered r as a parameter of these measures during the experimental evaluation.

Transformation based measures A common approach to measure time series distance requires transforming the series to a new domain and then perform calculation. The Edit Distance, for instance, does exactly that by changing continuous time series to simple discrete strings of symbols. We can use various mathematical transformations to change the series format before measuring.

The Discrete Fourier Transform (DFT) and the Discrete Wavelet Transform (DWT) have been used frequently as a pre-processing step in measure calculation. The amplitude of Fourier coefficients, for example, has the attractive property of been invariant under shifts. Moreover, for some kind of time series, they concentrate major signal energy in few low frequency coefficients [1].

We can generate the Fourier coefficients using the well-known relation [13]

$$X_k = \frac{1}{\sqrt{d}} \sum_{t=0}^{d-1} x_t e^{\frac{-i2\pi tk}{d}} \quad k = 0, 1, \dots, d - 1, \quad (10)$$

and use the k first coefficients distance as a similarity measure. In this case, k is a parameter to tune and small values will filter high frequencies components (probably noise). In the same way, the Discrete Wavelet Transform can be used as the pre-processing step. The following relation calculates the DWT coefficients

$$c_{j,k} = \frac{1}{d} \sum_{t=0}^{d-1} x_t \psi_{j,k}(t), \quad (11)$$

where the Mother Wavelet $\psi_{j,k}(t)$ will be scaled and translated by the following relation

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \quad (12)$$

where j is a scale factor, k determines the translation and several basis are available, for instance, Haar and Daubechies basis [14]. In this case, we can only use the first j scales in the measuring in order to obtain a noise free distance between two time series.

Discretization Process When the measure needs to convert series to strings, a discretizing step is needed. The usual method is defining a (fixed) number of intervals and mapping the real numbers to some of them. The intervals limits are chosen equally spaced to divide the values in a homogeneous way.

This method has however a weakness in the boundaries: when two values are near to the limits they may be mapped to different symbols. For instance, the Edit Distance will be increased in that situation, when the series in fact may be very similar.

An alternative is matching points based on proximity [11]. In this case, two points are considered the same symbol if they are closer than a limit β . The value of β must be adjusted, and will deeply affect the classification accuracy.

3 Combining Measures

In the previous section, we have defined several similarity measures. When it comes to classification using the 1NN Algorithm, the traditional approach is to select one measure at a time. We want to evaluate if the use of several measures will extract more useful information from data, improving the overall classification accuracy.

Therefore, a way for selecting or combining measures is necessary. We have created a weighting algorithm to assign a real number for each measure. In our approach, the nearest neighbor is defined as a weighted sum of all measures. We have proposed a new heuristic to search for the best weights, based on a method for the feature selection problem. In this context, [15] proposed the following strategy: divides the interval $[0, 1]$ in k equally spaced sub-intervals, uses the middle of the interval as starting point in the search, moves through space replacing weights by their next larger or smaller values in a greedy way, and uses the estimated error rate in the training set as evaluation function. The search stops when successive moves do not produce better results.

In some situations, this approach may generate redundant weights. For example, if $k = 10$ and we have two measures (or features), the combinations $(0.2, 0.8)$ and $(0.1, 0.4)$ produce the same classification results. We have proposed a new approach to avoid generating redundant values and to reduce the search space size: the sum of all values must be always 1. This restriction has led us to the following number of possible weights:

$$C_{m+k-1}^k = \binom{m+k-1}{k}, \quad (13)$$

where m is the number of measures. For small values of k and m , we can check exhaustively all combinations. In the experimental evaluation presented in the next section, we have $m = 8$ and $k = 10$, so the exhaustive enumeration is feasible.

Algorithm 1 shows the steps to generate all C_{m+k-1}^k weights combinations. It enumerates all possibilities using the lexicographic order and estimates the error rate for each distribution using only the training set and the *leave-one-out* procedure [16].

Algorithm 1 Searches for the combination of weights that maximize the classification performance using several measures.

Require: S , training examples;
 M , similarity measures;
 d , the step for weight distribution ($1/k$).

Ensure: \mathbf{P} , array of size m containing the best weight distribution found.

- 1: $se \leftarrow sd \leftarrow \infty$ { se - smallest error found. sd - average distance between samples on the best combination.}
- 2: $Q_0 \leftarrow 1$ e $Q_{1\dots m-1} \leftarrow 0$ {Initializes the weight array for evaluation.}
- 3: **repeat**
- 4: $e \leftarrow ascertain_error(\mathbf{Q}, M, S)$ {Evaluates the current weight distribution.}
- 5: $dist \leftarrow ascertain_distance(\mathbf{Q}, M, S)$
- 6: **if** $(e < se) \vee [(e = se) \wedge (dist < sd)]$ **then**
- 7: Updates \mathbf{P} , se and sd ;
- 8: **end if**
- 9: $i \leftarrow m - 1$ { i determines if there are more combinations to evaluate.}
- 10: **repeat**
- 11: $i \leftarrow i - 1$
- 12: **until** $[(i \geq 0) \wedge (Q_i = 0)]$
- 13: **if** $i \geq 0$ **then** {There are more valid configurations yet.}
- 14: $tmp \leftarrow Q_{m-1}$
- 15: $Q_{m-1} \leftarrow 0$
- 16: $Q_{i+1} \leftarrow tmp + d$
- 17: $Q_i \leftarrow Q_i - d$
- 18: **end if**
- 19: **until** $[(i \geq 0)]$

As an example of Algorithm 1 steps, suppose $d = 0.5$ and 3 similarity measures. The algorithm will generate weights as showed in Table 1 and use the *ascertain_error* function to estimate the classification error for each combination.

The algorithm solves ties by preferring combinations where the average distance from samples to their nearest neighbors is minimal (line 6 and variable sd), i.e. we are looking for a selection that keeps neighbors as close as possible.

Another important remark about this algorithm is that it does not tune each measure during the search. It is well known in the literature that some

Table 1. Weights sequence generated by Algorithm 1 when $d = 0.5$ and there are 3 metrics.

Step	Q_0	Q_1	Q_2	Step	Q_0	Q_1	Q_2
0	1.0	0.0	0.0	3	0.0	1.0	0.0
1	0.5	0.5	0.0	4	0.0	0.5	0.5
2	0.5	0.0	0.5	5	0.0	0.0	1.0

measure parameters affect the classification accuracy - for instance, see [4] where the authors present such evaluation, about the parameter *warping window* in DTW. We have chosen to work only with the best individual measure setup to restrict the search space size in Algorithm 1. In the experimental evaluation, we have adjusted each measure separately with the training set before running the algorithm.

4 Experimental Evaluation

Our goal is to determine if combining measures in the classification is better than using always the same one. We have compared the new classifier against a “classical option”. Normally the Euclidean Distance is used in the baseline classifier when someone wants to advocate the utility of a novel measure, however many works already had showed various weakness related with this measure [7]. We have chosen DTW because it is certainly better than Euclidean Distance, when implemented with *warping window* and tuned with a sufficiently large training set (see [4]).

We want to ascertain if Algorithm 1 yields an improved performance over 1NN-DTW in general, and not only in a special problem. According [17], what we need is a technique to compare two classifiers on multiple data sets. The *Wilcoxon Signed-Rank Test* is suitable for this task, if a random set of time series classification problems is available. The test ranks the difference between the two algorithms and computes the rank sums where each algorithm wins. Under the *null hypothesis*, the algorithms are equal and the sums will be the same. If the data indicate low probability of the result, we can reject the *null hypothesis*. A complete description about that test can be found in [18].

4.1 Data Sets

In order to compare the accuracy of two algorithms on multiple domains, we need a random sample of time series classification problems. There is a great difficulty in obtaining such sample - for instance, [3] shows how the absence of database benchmarks causes erroneous conclusions in data mining community.

We have used 20 problems available in [19], probably the biggest public repository of time series classification data. Even with a reasonable number of examples (in this case, different classification problems), the lack of randomness is

a serious drawback to this kind of experimental essay, as described in [20]. Although this weakness reduces the quality of our results, we have used a very safe statistic method (according [17]) and the best sample available, so our conclusions are supported at least by good evidences.

All data sets in the repository were normalized to standard deviation 1 and average 0 in order to mitigate vertical translations and scales issues. Each data set were originally split in two disjoint files for training and independent test. We have merged the two files, randomized their examples, and estimated the error rate by cross-validation, as described in the next section.

4.2 Experimental Results

In order to apply the *Wilcoxon Signed-Rank Test* we have to estimate the error rate for each domain in the experimental sample. We have used the recommended approach in [20]: we have conducted a 10-fold cross-validation, where for each fold only the training set was used to tune the measures and to estimate the weight distribution that minimizes the error rate. In both cases, the *leave-one-out* procedure was used. We have ascertained the round errors with the test sets and the final error rate was the average across 10 repetitions. As noted in [17], individual variances are not needed, since it comes from the different (and ideally independent) problems in the sample.

Table 2 summarizes the results. It shows 1NN-DTW and Algorithm 1 results and marks (boldface) the best result for each problem. The errors for both algorithms were obtained using the same random split during the cross-validation. For this table, the *Wilcoxon Signed-Rank Test* reports *p-value* equal to 2.2% and we were able to reject the null hypothesis that the algorithms have the same accuracy, with 5% significance level.

Table 2. Comparison of error rates for 1NN-DTW and measures combined as in the Algorithm 1.

Database	1NN-DTW	Comb. Measures	Database	1NN-DTW	Comb. Measures
<i>50 Words</i>	20.87	17.12	<i>Adiac</i>	33.17	33.30
<i>Beef</i>	43.33	33.33	<i>CBF</i>	0.00	0.00
<i>Coffee</i>	0.00	1.67	<i>ECG200</i>	11.00	1.00
<i>Face/All</i>	2.70	1.40	<i>Face/Four</i>	6.36	1.82
<i>Fish</i>	14.57	8.86	<i>Gun/Point</i>	2.00	3.00
<i>Lighting 2</i>	13.01	11.35	<i>Lighting 7</i>	22.38	22.43
<i>Olive Oil</i>	11.67	11.67	<i>OSU Leaf</i>	26.22	13.15
<i>S. Leaf</i>	12.89	8.27	<i>S. Control</i>	1.00	0.67
<i>Trace</i>	0.50	0.50	<i>Two Pat.</i>	0.00	0.00
<i>Wafer</i>	0.50	0.60	<i>Yoga</i>	15.67	13.00

We now must point out some details about the results in Table 2. Firstly, Algorithm 1 searches exhaustively for the best weight distribution, so the combi-

nations where just one measure receives all weights (all others with zero weight) were checked. We expected at least the same results for both algorithms, because the 1NN-DTW was a special case tested by Algorithm 1, but in 5 problems 1NN-DTW has got better results. This might have occurred in some cases due to the small data set size - for instance, in *Coffee* database we had about 50 examples for round training, almost all measures alone got 100% of correct classification, and the final error difference (1.67%) was caused by 1 mistake. The small database size prevented us from finding the true best measure. In other cases, the weight distribution was very unstable across the cross-validation rounds and probably Algorithm 1 has suffered from overfitting.

In 8 out of 11 problems where Algorithm 1 was better, the weights were very concentrated on measures derived from Edit Distance. These measures are much related to DTW. In those problems, probably the noise filter makes the difference.

5 Conclusions

In this work, we presented some measures typically used in time series classification problems. To verify if using different measures together improves the classification accuracy, we proposed an algorithm that combines several measures by assigning weights to them. We conducted a set of experiments and we compared the accuracy of the new algorithm against 1NN-DTW. The *Wilcoxon Signed-Rank Test* reported results with 5% significance level. Our results show that DTW is certainly the first good choice in this kind of problems, but another measures, specially the string ones, must be checked. In some cases, combining metrics brought a good accuracy gain, but in most cases the weights were concentrated in the few best metrics to each problem.

Acknowledgments. We would like to thank CNPq (Grant N^o 620165/2006-5) for the financial support given to the research from which this work originated, and Dr. Eamonn Keogh from University of California - Riverside for providing the data used in the experimental evaluation.

References

1. Antunes, C.M., Oliveira, A.L.: Temporal Data Mining: An Overview. In: Proceedings of the Workshop on Temporal Data Mining, San Francisco, EUA (2001) Knowledge Discovery and Data Mining (KDD 01).
2. Savary, L.: Notion of Similarity in (Spatio-)Temporal Data Mining. In: ECAI'02 Workshop on Knowledge Discovery from (Spatio-)Temporal Data. (2002) 63–71
3. Keogh, E., Kasetty, S.: On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *Data Mining and Knowledge Discovery* **7**(4) (2003) 349–371

4. Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.A.: Fast Time Series Classification Using Numerosity Reduction. In: ICML '06: Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, ACM Press (2006) 1033–1040
5. Cover, T., Hart, P.E.: Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* **13**(1) (1967) 21–27
6. Devijver, P.A., Kittler, J.: Pattern Recognition: A Statistical Approach. Prentice Hall, London (1982)
7. Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K.: Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. *Proceedings of the 21th International Conference on Very Large Data Bases* (1995) 490–501
8. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* **10**(8) (1966) 707–710
9. Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York, NY, USA (1997)
10. Hamming, R.W.: Error Detecting and Error Correcting Codes. *Bell System Technical Journal* **29**(2) (1950) 147–160
11. Bozkaya, T., Yazdani, N., Özsoyoglu, M.: Matching and Indexing Sequences of Different Lengths. In: CIKM '97: Proceedings of the Sixth International Conference on Information and Knowledge Management, New York, NY, USA, ACM Press (1997) 128–135
12. Sakoe, H., Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **26**(1) (1978) 43–49
13. Agrawal, R., Faloutsos, C., Swami, A.: Efficient Similarity Search in Sequence Databases. *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms* (1993) 69–84
14. Daubechies, I.: Ten Lectures on Wavelets. CBMS-NSF Reg. Conf. Series in Applied Math. SIAM (1992)
15. Kohavi, R., Langley, P., Yun, Y.: The Utility of Feature Weighting in Nearest-Neighbor Algorithms. In: 9th European Conference on Machine Learning, Prague, Czech Republic, Springer-Verlag (1997)
16. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. 2nd edn. John Wiley and Sons, New York (2001)
17. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* **7**(1) (2006) 1–30
18. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. 2nd edn. Chapman & Hall/CRC (2000)
19. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: The UCR Time Series Classification/Clustering (2006) http://www.cs.ucr.edu/~eamonn/time_series_data.
20. Salzberg, S.L.: On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data Mining and Knowledge Discovery* **1**(3) (1997) 317–328