

Time Series Classification Challenge Experiments

Dave DeBarr

Microsoft Corporation
dave.debarr@microsoft.com

Jessica Lin

Computer Science Department
George Mason University
jessica@ise.gmu.edu

ABSTRACT

This paper describes a comparison of approaches for time series classification. Our comparisons included two different outlier removal methods (discords and reverse nearest neighbor), two different distance measures (Euclidean distance and dynamic time warping), and two different classification algorithms (k nearest neighbor and support vector machines). An algorithm for semi-supervised learning was evaluated as well. While dynamic time warping and support vector machines performed pretty well overall, there was no single approach that worked best for all data sets.

1. INTRODUCTION

Many data sources naturally generate either time series or sequence data. For example, an ElectroCardioGram (ECG) is used to identify “abnormal” heart rhythms. Electrodes are placed on the body to measure the average electrical voltage produced by the beating of the heart muscle. An electrocardiogram is often visualized as a line graph, where the “ x ” axis is time and the “ y ” axis is the average voltage measured by the electrodes. Spatial data may also be converted to a sequence of values. For example, the outline along the “top” of a word on a printed page can be used to recognize words. This representation can be visualized as a line graph, where the “ x ” axis is the distance from the beginning of the word and the “ y ” axis is the height of the corresponding point of the outline along the “top” of the word. Both an ECG and a word can be represented as a sequence of numeric values. For the ECG, the sequence is a series of electrical voltage values. For the word, the sequence is a series of height values.

Pattern recognition, also known as classification, is the process of mapping an input representation for an entity or relationship to an output category. Approaches to pattern recognition tasks vary by representation for the input data, similarity/distance measurement function, and pattern recognition technique. In this work, we evaluate outlier removal methods for pre-processing the training data, then evaluate a number of approaches for time series classification.

2. RELATED WORK AND BACKGROUND

The presence of outliers or mislabeled objects in the datasets can affect the accuracy of classification tasks. In this work we investigate two different outlier detection techniques. We show

how the removal of such outliers improves the classification accuracy.

2.1 Time Series Discord

In [4] we introduced the notion of time series discord. Simply stated, given a dataset, a discord is the object that has the largest nearest neighbor distance, compared to all other objects in the dataset. Similarly, k -discords are the k objects that have the largest nearest neighbor distances. Discord discovery can be useful for anomaly detection, since the discords represent the most-isolated data points in the dataset.

In [4] we proposed an efficient algorithm, HOT SAX, to find discords without having to compute the nearest neighbor distances for all the objects. Since the focus of this contest is on classification accuracy, we will only present a brief description of the algorithm, but note that outlier detection and removal by discord discovery can be performed efficiently with HOT SAX.

The basic idea is to convert the time series objects into discrete symbols [7], and from the symbols we can estimate how similar two objects are from each other. We can then use this information to identify the likely candidates for discords, and eliminate the ones that could not be the discords from further consideration. Our algorithm is able to prune off most of the search space and improves the running time drastically. We refer the interested reader to [4] for detailed description of the algorithm.

2.2 Outlier Detection by Reverse Nearest Neighbor

Given a query object, nearest neighbor (NN) search returns the object in the database that is the most similar to the query object (similarly, k -NN search [11] returns the k most similar objects to the query). Reverse nearest neighbor search, on the other hand, returns all objects in the database whose nearest neighbors are the query object [6,12,13,14]. Note that since the relation of NN is not symmetric, the NN of a query object might differ from its RNN(s). Figure 1 illustrates this idea. Object B being the nearest neighbor of object A does not automatically make it the reverse nearest neighbor of A , since A is not the nearest neighbor of B .

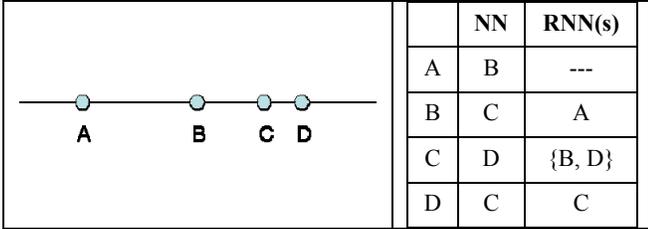


Figure 1: B is the nearest neighbor of A; therefore, the RNN of B contains A. However, the RNN of A does not contain B since A is not the nearest neighbor of B.

Reverse nearest neighbor queries have potential applications in business planning, targeted marketing, etc. For example, a supermarket chain planning to open a new branch can determine the optimal location by identifying its potential customers in terms of distance. Note that the problem is not to identify customers who live within a certain radius from the store which can be determined by range queries, because such customers might actually live closer to another supermarket in the area. Rather, the objective is to find the customers for whom the new store will be the closest one. Ideally, the new location should be one that would attract the most customers. The RNN queries can answer exactly this question.

In addition, the RNN count of an object can potentially capture the notion of its *influence* or *importance* [6]. For the supermarket example, the location with the highest RNN count (i.e. the most potential customers) can be said to be of significant influence. Nanopoulos et al [8] proposed an algorithm based on $RkNN$ to identify anomalies in data. The idea is that, given an appropriate value of k , objects that do not have any $RkNN$ are considered anomalies [8,14]. To the best of our knowledge, there is no empirical validation on this idea. In this work we investigate the effectiveness of the $RkNN$ -based outlier detection algorithm, and compare the results with the discord-based outlier detection algorithm.

3. PREPROCESSING – OUTLIER REMOVAL

The datasets we used are from the UCR Clustering/Classification page [J]. Each dataset consists of a training set and a test set. Below we discuss our heuristics for identification and removal of outliers/anomalies from the training set.

3.1 Time Series Discord

To identify discords, we process each class individually. The discord algorithm is applied to each class, rather than to the entire dataset. As suggested in Figure 2, if we apply the discord algorithm globally, the first discord (O_8) can be identified successfully, but the second one (O_7) will be missed. The reason is that the nearest neighbor distances for O_{11} , O_{12} , O_{13} , and O_{14} are all larger than the nearest neighbor distance for O_7 .

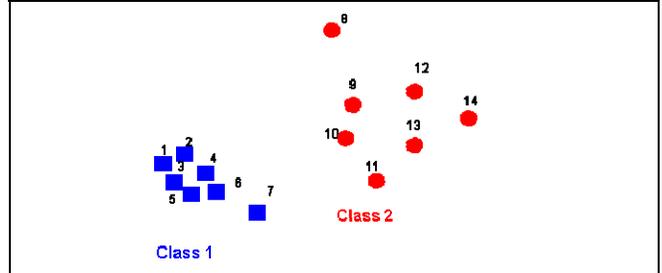


Figure 1: Two clusters of data with one global outlier (8) and one local outlier (7). The discord discovery algorithm must be applied to each class individually, otherwise we will not be able to identify the second discord.

The steps are outlined as follows:

1. The discord for each class is identified.
2. For each discord, compute the ratio of its nearest neighbor distance to the *average* nearest neighbor distance of all other objects in the same class.
3. Remove the discord with the largest ratio from Step 2.
4. Compute the predictive accuracy using leave-one-out cross validation. If the accuracy is worse than the accuracy before any removal, or if we have reached the user-defined maximum number of discords, then stop.
5. Re-compute the discord from the class which the previous object was removed. Go to Step 2.

3.2 $RkNN$ -based Outlier Detection

The steps for the $RkNN$ -based approach are similar to the discord algorithm, except that instead of processing each class separately, the $RkNN$ -based algorithm identifies all potential outliers at once. Therefore, we need to run the algorithm only once in the beginning, for any desirable k . For the dataset shown in Figure 2, if we set $k = 2$, then the only objects that do not belong to *any* other objects' 2-NN sets are O_7 and O_8 . In our training step, we try different values of k starting from 2, and remove all the objects with zero $RkNN$ count. We stop the process when no object has zero $RkNN$ count (typically when $k = 5$ or 6) or when we reach the user-defined maximum value of k . Note that as k increases, the number of such objects returned decreases. In the next section we compare and analyze the results from both outlier detection approaches.

3.3 Training Results

After identifying and removing the outliers, we validate the results and show the actual classification accuracy on the test sets in Table 1. The first column of numbers is the original accuracy computed before any outlier removal. The next two columns are the accuracy for the discord-based and the $RkNN$ -based outlier detection algorithms, respectively. The bolded text indicates that the removal of outliers by the corresponding method improves the accuracy, whereas the shaded cell indicates that the accuracy deteriorates after the removal. To get an idea of how effective our algorithms perform, we repeat the classification task using various parameters examined during the training process, and record the best possible accuracy (shown as the second number, if the actual accuracy is not the best).

Table 1. Error rate before and after outlier removal

Dataset	Original	Discord/best possible	RkNN/best possible
50words	0.2527	0.2527 / 0.2505	0.2527 / 0.2505
Adiac	0.4194	0.4194	0.4194
Beef	0.33	0.40 / 0.33	0.33
CBF	0.0233	0.06 / 0.0233	0.06/0.0233
Coffee	0	0	0
ECG200	0.1900	0.16	0.1800
FaceAll	0.2154	0.2154	0.2154 / 0.1811
FaceFour	0.1591	0.1591	0.1591
GunPoint	0.04	0.04 / 0.0267	0.04 / 0.033
Lighting2	0.1311	0.1311	0.1311
Lighting7	0.2329	0.2329 / 0.2191	0.2740 / 0.2329
OSULeaf	0.4463	0.4380	0.4463
OliveOil	0.1333	0.1333	0.1333
Swedish	0.2192	0.2192	0.2192 / 0.2176
Trace	0.01	0.01	0.01
2Patterns	0.0008	0.0008	0.0008 / 0.0005
Fish	0.1657	0.1600	0.1543
Synthetic	0.0367	0.0367	0.0367
Wafer	0.0092	0.0092	0.0117 / 0.0092

The discord-based approach seems to yield better accuracy than the RkNN-based approach. This result is not surprising, as the latter removes anomalies in batches, whereas the former does it one at a time. One way to improve the flexibility of the RkNN-based approach is to treat the “outliers” returned from the algorithm as candidates only, rather than removing them right away. Once we get the set of candidates, we can then follow the paradigm of the discord-based approach and remove them one at a time until the accuracy deteriorates. We will defer the investigation of such refinement for future research.

In general, the discord-based outlier detection algorithm seems to be a more desirable approach, since it’s more robust, efficient, and produces better results. However, one unique advantage of the RkNN-based approach is that it is not limited to supervised learning, where the class labels are known.

We conclude this section with the observation that while outlier removal can improve classification accuracy, the “best possible” values in both approaches suggest that, the heuristics for the training process can be improved since our algorithms did not always find the parameters that result in the best accuracy.

4. CLASSIFICATION APPROACH

In formulating different approaches for classification, we evaluated two distance measures and two classification algorithms. To evaluate the use of information about the test set to help build classifiers, we also tried a semi-supervised learning approach.

4.1 Distance Measures

In pattern classification, similarity (or distance) between a new observation and previously observed examples, is used to reason

about the possible classification of the new example. Similarity, or (inversely) distance, between sequences can be measured in a variety of ways. Possible measures of similarity (or distance) include: Euclidean distance, cosine distance, Dynamic Time Warping (DTW) distance, cosine similarity, and Radial Basis Function (RBF) similarity. Euclidean distance is defined as the square root of the sum of squared differences between two sequences. For example, if vector[1] = [3 5] and vector[2] = [6 1], then the Euclidean distance between the two is $\sqrt{(3-6)^2+(5-1)^2} = 5$. The geometric interpretation of this function is the distance between two points in a Euclidean space. Values for the Euclidean distance function are non-negative.

Dynamic Time Warping (DTW) is used to allow flexible matching between sequences. DTW is capable of measuring distance between sequences of different lengths, as well as sequences of the same length. DTW is useful for matching sequences where one sequence is simply a shifted version of another sequence. Dynamic programming is used to compute the minimum square root of the sum of squared differences between points in two sequences [10]. The DTW distance was normalized by the length of the “warping” path used for comparison. A constrained version of DTW was used to speed up DTW computations. The number of points that can match against any given point in another sequence is constrained to reduce the number of squared difference calculations. Values for the DTW distance function are also non-negative.

4.2 Classification Algorithms

Pattern recognition is the process of mapping an input representation for an entity or relationship to an output category. A functional model is used to map observed inputs to output categories. A great deal of model construction techniques have been developed for this purpose: including decision trees, rule induction, Bayesian networks, memory-based reasoning, Support Vector Machines (SVMs), neural networks, and genetic algorithms. Memory-based reasoning methods, such as nearest neighbor, have been successfully used for classification of sequence data [3]. Kernel based methods, such as SVMs, also rely on measuring pairwise similarity between pairs of objects.

The nearest neighbor classification algorithm works by computing the distance between the sequence to be classified and each member of the training set. The classification of the sequence to be classified is predicted to be the same as the classification of the nearest training set member. A common variation of this algorithm predicts the classification of the test sequence to be the most common classification found among the “k” nearest neighbors in the training set.

Support Vector Machines (SMVs) are commonly used for constructing classification models for high-dimensional representations. The observations from the training set which best define the decision boundary between classes are chosen as support vectors (they support/define the decision boundary). A non-zero weight is assigned to each support vector. The classification of a test sequence is predicted by computing a weighted sum of similarity function output between the new sequence and the support vectors for each class. The predicted class is the class with the largest sum, relative to a bias (offset) for the decision boundary. Kernel functions are used to measure the similarity between sequences. Common kernel functions for

this purpose include the cosine similarity function and the Gaussian RBF.

4.3 Semi-Supervised Learning

In supervised learning, training examples with known classification labels are used to construct a model to label test examples with unknown classification labels. Semi-supervised learning on the other hand involves using both the training examples with known classification labels and the test examples with unknown classification labels to construct a model to label the examples with unknown classification labels. While it's possible to construct a model for labeling future test examples, it's also possible to construct a model that focuses on the current test examples. Vapnik referred to this as transduction [15], suggesting that it's better to focus on the simpler problem of classifying the current test examples rather than trying to construct a model to future test examples.

The spectral representation used for these experiments is a combination of approaches previously used by others [2, 9]. The testing set is concatenated to the end of the training set for computing the spectral representation of both the training and testing set sequences. The Laplacian matrix of the spectral graph for the combined set is computed as:

$$\text{Laplacian} = \text{Inverse}(\text{Degree}) * (\text{Degree} - \text{Adjacency})$$

The "adjacency" matrix was computed as the sum of the k=3 nearest neighbor similarity matrix and its transpose. Each entry in the k=3 nearest neighbor affinity matrix is computed as follows. $\text{Affinity}[i,j] = 0$ if sequence 'j' is not one of the 3 nearest neighbors for sequence 'i'; otherwise, $\text{Affinity}[i,j]$ is computed as:

$$\frac{\text{Sim}(\text{Seq}[i], \text{Seq}[j])}{\text{Sim}(\text{Seq}[i], \text{Neighbor}[1]) + \text{Sim}(\text{Seq}[i], \text{Neighbor}[2]) + \text{Sim}(\text{Seq}[i], \text{Neighbor}[3])}$$

... where $\text{Sim}()$ is the cosine similarity function. The diagonal "degree" is then computed as $\text{Degree}[i,i] = 0$ if 'i' is not equal to 'j'; otherwise $\text{Degree}[i,i] = \text{Sum}(\text{Affinity}[i,*])$ values. The rows of the matrix formed by the eigenvectors of the spectral decomposition of the Laplacian matrix are then used as the new training and test set sequences. The eigenvector associated with the smallest eigenvalue is removed, because it is considered to provide little value in distinguishing groups of similar observations. The eigenvectors associated with the largest eigenvalues are removed for the same reason. The number of eigenvectors to be retained is determined by leave-one-out cross validation of the training set. The first rows of the matrix formed from the remaining eigenvectors represent the training set sequences, while the last rows represent the testing set sequences.

5. EMPIRICAL EVALUATION

Ten fold cross validation was used to compare all combinations of similarity measurement and classification algorithm, along with variations of parameters. For dynamic time warping, we varied the Sakoe-Chiba bandwidth from 1 to 10. For the nearest neighbor classification algorithm, we varied the number of neighbors from 1 to 10. For the radial basis functions, we varied kernel widths based on the 10th, 30th, 50th, 70th, and 90th percentiles of a sample of observed distances for the training data. For the support vector machines, we varied the

complexity penalty based on the size of the training set; i.e. varying nu from 1% to 16% of the size.

The spectral representation significantly improved performance on the OliveOil data set ("k" nearest neighbor with k=3), while support vector machines significantly improved performance on the Beef data set (Radial Basis Function with gamma = 95.9170 and nu = 0.1).

Table 2. Error Rate Improvements

Data Set	Before	After
Beef	0.33	0.00
OliveOil	0.1333	0.0678

For all other data sets, a k nearest neighbor classifier with dynamic time warping distance provided the best performance (as reported in table 1).

6. CONCLUSIONS AND FUTURE WORK

As noted in discussions of the "no free lunch" theorem [16], there is no single best approach for machine learning problems. This appears to be true for time series classification as well. In future work, we hope to focus on visualization of training set classification error in order to understand sources of classification error, in addition to evaluating ensemble methods for classification.

7. REFERENCES

- [1] D. Dasgupta and S. Forrest. Novelty Detection in Time Series Data Using Ideas from Immunology. In proceedings of the 8th Int'l Conference on Intelligent Systems. Denver, CO. Jun 24-26, 1999.
- [2] Joachims, T. "Transductive Learning via Spectral Graph Partitioning", Proceedings of the 20th International Conference on Machine Learning (ICML-2003), 2003.
- [3] Keogh, E. and Kasetty, S. "On the Need for Time Series Data Mining Benchmarks: a Survey and Empirical Demonstration", Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Canada, 2002.
- [4] Keogh, E., Lin, J. & Fu, A. HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In the 5th IEEE International Conference on Data Mining. New Orleans, LA. Nov 27-30.
- [5] Keogh, E., Xi, X., Wei, L. & Ratanamahatana, C. A. (2006). The UCR Time Series Classification/Clustering Homepage: http://www.cs.ucr.edu/~eamonn/time_series_data.
- [6] F. Korn and S. Muthukrishnan. Influence Sets Based on Reverse Nearest Neighbor Queries. In proceedings of the 19th ACM SIGMOD International Conference on Management of Data. Dallas, TX. May 14-19, 2000. pp. 201-212

- [7] J. Lin, E. Keogh, W. Li, and S. Lonardi. Experiencing SAX: A Novel Symbolic Representation of Time Series. *Data Mining and Knowledge Discovery*. 2007.
- [8] A. Nanopoulos, Y. Theodoridis, and Y. Manolopoulos. C2p: Clustering Based on Closest Pairs. In *proceedings of the 27th International Conference on Very Large Data Bases*. Rom Italy. Sept 11-14, 2001. pp. 331-340
- [9] Ng, A.Y. and Jordan, M.I. "On Spectral Clustering: Analysis and an Algorithm", *Advances in Neural Information Processing Systems (NIPS) 14*, edited by T. Dietterich, S. Becker, and Z. Ghahramani, 2002.
- [10] Ratanamahatana, C.A. and Keogh, E. "Everything You Know about Dynamic Time Warping is Wrong", *Third Workshop on Mining Temporal and Sequential Data*, in conjunction with the 10th ACM SIGKDD International Conference of Knowledge Discovery and Data Mining (KDD-2004), Seattle, 2004.
- [11] T. Seidl and H. P. Kriegel. Optimal Multi-Step k-Nearest Neighbor Search. In *proceedings of the 1998 SIGMOD International Conference on Management of Data*. Seattle, WA. June 2-4, 1998. pp. 154-165
- [12] A. Singh, H. Ferhatosmanoglu, and A. Tosun. High-Dimensional Reverse Nearest Neighbor Queries. In *proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management*. New Orleans, LA. Nov 2-8, 2003.
- [13] I. Stanoi, D. Agrawal, and A. E. Abbadi. Reverse Nearest Neighbor Queries for Dynamic Databases. In *proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. Dallas, TX. May 14, 2000. pp. 44-53
- [14] Y. Tao, D. Papadias, and X. Lian. Reverse knn Search in Arbitrary Dimensionality. In *proceedings of the 30th International Conference on Very Large Data Bases*. Toronto, Canada. Aug 31-Sept 3, 2004. pp. 744-755
- [15] Vapnik, V. "Estimating the Values of Function at Given Points", *Statistical Learning Theory*, Wiley-Interscience, New York, 1998.
- [16] No Free Lunch Theorems, <http://www.no-free-lunch.org/>, accessed June 16, 2007.