A Weighted Local Mean-Based k-Nearest Neighbors Classifier for Time Series

Tuan Minh Tran, Xuan-May Thi Le, Vo Thanh Vinh, Hien T. Nguyen, Tuan M. Nguyen
Faculty of Information Technology, Ton Duc Thang University
19 Nguyen Huu Tho St., Tan Phong Ward, District 7, Ho Chi Minh City, Vietnam
tmtuan1307@gmail.com, xuanmay@outlook.com, vothanhvinh@tdt.edu.vn, hien@tdt.edu.vn, nmtuan@tdt.edu.vn

ABSTRACT
We propose a Weighted Local Mean-based *k* Nearest Neighbors (WLM*k*-NN) classifier for time series. The proposed method is different from Local Mean-based *k*-Nearest Neighbors (LM*k*-NN) in that it assigns a weight for each element of time series when calculating local mean vectors. Indeed, our proposed method determines the local mean vectors more efficiently than LM*k*-NN does by using some weights which are obtained from the distances of a query instance to its *k* nearest instances in each class. Our experiments were conducted over 85 time series datasets in UCR Time Series Classification Archive and compared with some baseline methods. Experimental results show that our method outperforms *k*-NCN, LM*k*-NN, LM*k*-NCN, and the widely used classifier *k*-NN in many time series datasets.

CCS Concepts
• Computing methodologies → Supervised learning by classification.

Keywords
Classification; time series; *k*-nearest neighbors; LM*k*-NN; WLM*k*-NN; *k*-NCN; NCN.

1. INTRODUCTION
Time series data exist in many fields including economics, finance, or medicine. Among many time series mining tasks, classification has been emerged as an important problem as it is the key of pattern recognition. Due to the high dimensional characteristic of time series, it is really hard to find an efficient approach for this kind of data. Among popular classification methods such as *k*-Nearest Neighbors, Support Vector Machine, Artificial Neural Network, the classic approach *k*-Nearest Neighbors (*k*-NN) especially for 1-NN (1-Nearest Neighbor) has been widely considered as hard to be beaten on time series classification [2], [4], [9], [10].

The 1-NN is based on distances from a query instance to all training instances in the training set to find the nearest instance, then assigns the label of the nearest one to it. This method is based on only one nearest instance to determine the label; therefore, it might bring more risks in classification. Local Mean-based k-Nearest Neighbor (LM*k*-NN) [7] was proposed to improve *k*-NN; in which, it attempts to overcome the negative effect of outliers in a small sample training set by calculating the local mean vector of *k*-nearest instances in each class. Another method to improve *k*-NN is the *k*-Nearest Centroid Neighbors (*k*-NCN) [8] which is based on a method called Nearest Centroid Neighbor (NCN) [1]. The *k*-NCN has been evaluated as an effective method on datasets with different sample sizes [5], [8]. A combination of LM*k*-NN and *k*-NCN, namely, Local Mean-Based *k*-Nearest Centroid Neighbor LM*k*-NCN) [5] was proposed to take advances of two previous methods. LM*k*-NCN is robust to outliers and effective in the small sample size case with the NCN concept behind. Note that LM*k*-NN, *k*-NCN and LM*k*-NCN have not been tested on time series data yet, while k-NN is a baseline on this kind of data.

In this work, we propose a method for time series classification, namely, Weighted Local Mean-based *k*-Nearest Neighbors (WLM*k*-NN). The difference between our method and LM*k*-NN is that we determine the local mean vectors more efficient by adding some weights based on the distances from an unlabeled instance to the *k* nearest instances in each class. Experimental results show that WLM*k*-NN outperforms *k*-NN, LM*k*-NN, *k*-NCN, LM*k*-NCN in 85 time series datasets.

The rest of this paper is organized as follows: Session 2 gives definitions and notations, Session 3 describes the previous methods, Session 4 shows our proposed method, followed by a set of experiments in Session 5, and Session 6 concludes the work.

2. DEFINITIONS AND NOTATIONS
- **Definition 1:** A time series X is a sequence of real numbers collected at regular intervals over a period of time: $X = x_1, x_2, \dots, x_n$. The periods of collecting are equals, so they are unimportant. As a consequence, we can consider a time series as an *n*-dimensional instance in a metric space.
- **Definition 2:** Euclidean Distance (ED) between two time series $X = (x_1, x_2, \dots, x_n)$, $Y = (y_1, y_2, \dots, y_n)$ is given by Eq. (1).

$$ED(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

- **Definition 3:** Given a set of M time series $\{X_i\}_{i=1}^{M}$, where each $X_i = x_{i1}, x_{i2}, \dots, x_{in}$, are time series of length n. The mean vector of S, denoted as $\bar{X} = (\overline{x_1}, \overline{x_2}, \dots, \overline{x_n})$, where each $\overline{x_j}$ is defined as in Eq. (2).

ICMLC 2017, February 24-26, 2017, Singapore, Singapore

DOI: http://dx.doi.org/10.1145/3055635.3056594

$$\overline{x_j} = (\sum_{i=1}^{M} x_{ij})/M \quad (2)$$

The weighted mean vector of S denoted as $\overline{X_w} = (\overline{x_{w1}}, \overline{x_{w2}}, .., \overline{x_{wn}})$, where each $\overline{x_{wj}}$ is defined as in Eq. (3).

$$\overline{x_j} = \frac{\sum_{i=1}^{M}(w_j \times x_{ij})}{\sum_{i=1}^{M} w_i} \quad (3)$$

The notations used in this paper are summarized in Table 1.

Table 1. Notations

Notation	Meaning
ED(X,Y)	Euclidean Distance between two time series *X*, *Y*
k-NN	*k*-Nearest Neighbors
1-NN	1-Nearest Neighbor
NCN	Nearest Centroid Neighbor
k-NCN	*k*-Nearest CentroidNeighbor
LM*k*-NN	Local Mean-based *k*-Nearest Neighbors
WLM*k*-NN	Weighted Local Mean-based *k*-Nearest Neighbors (proposed method)
T_L	A training set $T_L = \{T_i\}_{i=1}^{M}$, where *L* is the number of classes, *M* is the training set's size, and each t_i is a time series of length *n*
T_L^i	A subset of $T_L$which contains time series of class *i* with $i = \overline{1,L}$

3. RELATED WORK

3.1 The LMk-NN Classifier

Local Mean-based *k*-Nearest Neighbor (LM*k*-NN) is a simple and robust classifier in small sample size cases [5], [7]. The target of LM*k*-NN is to overcome the negative effect of the outliers existing in a training set. The rationale behind this method is the local mean vectors which are obtained from *k* nearest neighbors of a query instance in each class. The query instance is then assigned to the class which contains the local mean vector closest to it. The outline of this method is as follows:

Given a training set T_L and a subset $T_L^i (\, i = \overline{1,L})$ of T_L which contains all instances in class *i*. The label of a query instance *u* is determined by the following steps:

1. For each T_L^i, find *k* nearest neighbors of *u* from T_L^i say $\{u_{ij}\}_{j=1}^{k}$, and then put them into KNN_i set $i = \overline{1,L}$.
2. For each $KNN_i = \{u_{ij}\}_{j=1}^{k}$ obtained from Step 1, find the local mean vector $\overline{u_i}$ for class *i* according to Eq. (2). Thus, each class *i* has one local mean vector $\overline{u_i}$.
3. Calculate distances from *u* to each local mean vector $\overline{u_i}$ and then assign the label of its nearest local mean vector to *u*.

Note that LM*k*-NN is equivalent to 1-NN when *k* = 1. LM*k*-NN was successfully on classification of ordinary data but it has not been tested on time series yet. In this work, we extend LM*k*-NN to create a method called WLM*k*-NN.

3.2 The k-NCN Classifier

The *k*-NCN is based on NCN, which attempts to find the neighbors distributed as geometrically around the query instance as possible. In order to understand *k*-NCN, we first introduce NCN [1], [5]. Given a set of *M* instances: $S = \{X_i\}_{i=1}^{M}$, and a query instance *u*, NCN is found by the following steps:

1. Initialize set $U = \emptyset$.
2. Find the 1st nearest neighbor of *u* from *S*. This instance, denoted as u_1 is considered as the 1st nearest centroid. Add u_1 to *U* and remove it from *S*.
3. For each remaining instance in S, say X_j, calculate each mean vector $\overline{v_j}$ (using Eq. (2)) from the set $U \cup X_j$; then select the instances X_j where $\overline{v_j}$ is the closest to u, denoted as u_j, add u_j to *U* and remove it from *S*; repeat this step until *j* reaches a user-defined value *k*.

Base on NCN, the outline of *k*-NCN is given as follows:

1. Find *k* centroids that are nearest to an unlabeled instance *u* from a training set T_L to get a set of *k* instances *U* as the above description of NCN.
2. Assign the unlabeled instance *u* to the class with a majority of votes in *U*.

3.3 The LMk-NCN Classifier

LM*k*-NCN is a combination of LM*k*-NN and *k*-NCN so it can deal with outliers and produce effective result in different sample sizes. The outline of LM*k*-NCN is given as follows:

Given a training set T_L and a subset $T_L^i (i = \overline{1,L})$ of T_L which contains all instances in class *i*. The label of a query instance *u* is determined by the following steps:

1. For each T_L^i, find *k* nearest centroids neighbors (*k*-NCN) of *u* from T_L^i, say $KNCN_i = \{u_{ij}\}_{j=1}^{k}$.
2. For each $KNCN_i = \{u_{ij}\}_{j=1}^{k}$obtained from Step 1, find the local mean vector $\overline{u_i}$ for class *i* according to Eq. (2). Thus, each class *i* has one local mean vector $\overline{u_i}$.
3. Calculate distances from *u* to each local mean vector $\overline{u_i}$, and then assign the label of its nearest local mean vector to *u*.

4. PROPOSED METHOD

4.1 The Weighted Local Mean-Based k-Nearest Neighbor

The LMk-NN finds mean instances by calculating mean vectors of k nearest instances in each class. By this way, it can overcome negative effects of outliers. Our proposed method finds a mean instance by using weighted mean vectors where weights are computed based on distances from a query instance to its *k* nearest instances in each class. Therefore, our method can deal better with outliers. The outline of WLM*k*-NN is as follows:

Given a training set T_L and a subset T_L^i. $(i = \overline{1,L})$ of T_L which contains all instances in class *i*. The label of a query instance *u* is determined by the following steps:

1. For each T_L^i, find *k*nearest neighbors of u from T_L^i, say $\{u_{ij}\}_{j=1}^{k}$, and then put them into KNN_i set ($i = \overline{1,L}$).

2. For each $KNN_i = \{u_{ij}\}_{j=1}^{k}$ obtained from Step 1, find the weighted local mean vector $\overline{u_i}$ for class i according to Eq. (3) in which the weights w_{ij} is defined as in Eq. (4).

$$w_{ij} = \frac{1}{ED(u, u_{ij})} \quad (4)$$

   Thus, each class i has one weighted local mean vector $\overline{u_i}$.

3. Calculate distances from u to each weighted local mean vector $\overline{u_i}$, and then a assign the label of its nearest weighted local mean vector to u.

Note that WLM*k*-NN, LM*k*-NN and 1-NN are the same when $k = 1$.

4.2 Difference between WLM*k*-NN and LM*k*-NN

The difference between WLM*k*-NN and LM*k*-NN is in the Step 2 of both algorithms when we calculate the local mean vectors. LM*k*-NN simply finds a local mean vector by calculating an ordinary arithmetic mean, whereas we use a weighted arithmetic mean.

Table 2. 14 points for example and distances between them to the unlabeled instance *u*(6, 7)

Name	*x*	*y*	Class	ED distance to *u*
D1	3	1	A	6.71
D2	2	3	A	5.66
D3	3	3	A	5
D4	2	1	A	7.2
D5	3	2	A	5.8
D6	1	1	A	7.81
D7	0	1	A	8.5
D8	1	0	A	8.6
D9	0	0	A	9.2
D10	9	8	B	2.2
D11	8	9	B	3.6
D12	8	9	B	2.8
D13	13	15	B	10.6
D14	14	14	B	10.6

Table 3. The local mean vectors of class *A* and *B* when applying LM*k*-NN, label *A* is assigned to *u*

	x	*y*	ED distance to *u*
Local mean vector of class A	2.8	2	**5.9363**
Local mean vector of class B	10.4	11	5.9464

For example, in a 2-dimensional space *Oxy* given a training set T_2 which has 14 instances and two classes *A*, *B* as shown in Table 2 and an unlabeled instance *u*(6, 7) which should be more likely classified as class *B*.

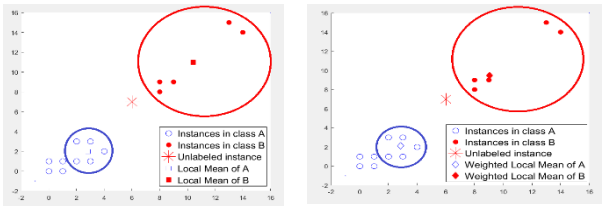
As we can see in Table 2, D13 and D14 are outliers of class *B*. When applying LM*k*-NN with $k = 5$ on T_2 to classify *u*, the *k* nearest neighbors of *u* in class *A* is D1, D2, D3, D4, D5 and the *k* nearest neighbors of *u* in class *B* is D10, D11, D12, D13, D14. So D13 and D14 affect to the local mean vector of class *B*. Table 3 shows local mean vectors of class A and B as well as their distances to *u*. As a result, it would assign label A to *u* because the local mean vector of class A is closer to *u* than that of class B.

By using the proposed method WLM*k*-NN with $k = 5$ on T_2 to classify *u*, D13 and D14 are also selected as *k* nearest neighbors of *u* in class *B*. However, when we calculate the weighted local mean vector of class *B*, the weights given to D13, D14 are now much less than the weights given to the other instances because their distances to *u* are much farther. The weighted local mean vectors of class *A* and *B* as well as their distances to *u* are presented in Table 4. The weighted local mean vector of class *B* is much closer to *u* than that of *B*, so it would assign label *B* to *u* as we expected.

Table 4. The weighted local mean vector of class *A* and *B* when applying WLM*k*-NN, label *B* is assigned to *u*

	x	*y*	ED distance to *u*
Weighted local mean vector of class A	2.9	2.1	5.8
Weighted local mean vector of class B	9.0	9.5	**3.9103**

In Fig. 1, we illustrate different results in determining local mean vectors and weighted local mean vectors of LM*k*-NN and WLM*k*-NN respectively.



(a) (b)

Figure 1. (a) Local mean vector of LM*k*-NN, (b) Weighted local mean vector of WLM*k*-NN.

5. EXPERIMENTS

This section shows experiments to compare the proposed method WLM*k*-NN with previous methods: LM*k*-NN, *k*-NCN, LM*k*-NCN, *k*-NN. In the case of *k*-NN, it has been proven that 1-NN is the best choice of time series classification, so we compare our method with this classifier in the case of $k = 1$.

5.1 Datasets

The experiments were conducted over 85 datasets from UCR Time Series Classification Archive [3]. Each dataset includes a training set and a testing set from different domains such as medicine, engineering, astronomy, or signal processing. Due to limited space, we do not show the properties of each datasets. Interested reader may refer to them in [3], some properties of some sample datasets is presented in Table 5.

Table 5. Some sample datasets used in the experiments, please refer to [3] for a full list of datasets.

Dataset	Adiac	ECG5000	Haptics	Wafer
Train size	390	500	155	1000
Test size	391	4500	308	6164
Length	176	140	1092	152
No. of classes	37	5	5	2
Type	IMAGE	ECG	MOTION	SENSOR

5.2 Comparing WLMk-NN with LMk-NN

In this experiment, we compare WLM*k*-NN with LM*k*-NN in different values of $k = \overline{4,9}$. As seen in Fig. 2, WLM*k*-NN outperforms LM*k*-NN in many datasets with different values of *k*. Especially with *k* = 9, it wins LM*k*-NN in 62 datasets, lose in only 11 datasets, and draws in 12 datasets.In the other values of *k* the proposed method is also better than LM*k*-NN in a vast majority of datasets. Details of the comparison is shown in Fig. 3 in which we show a column chart where each column presents subtraction of accuracy rates of these two methods. In general, the more columns exist in the positive area of the charts, the better our method.

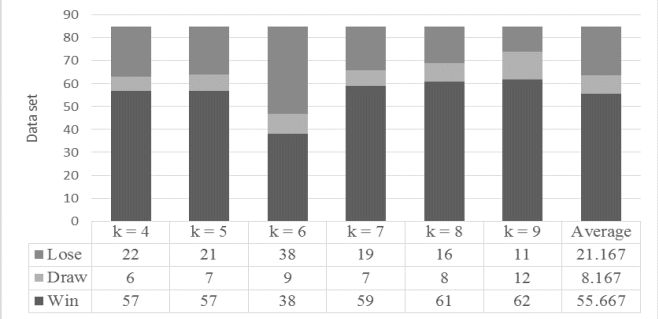


Figure 2. Stacked column chart comparing WLM*k*-NN with LM*k*-NN, *Win*: number of datasets in which the accuracy of WLM*k*-NN is better than that of LM*k*-NN, *Lose*: number of datasets in which the accuracy of WLMk-NN worse than that of LMk-NN, *Draw*: number of datasets in which WLM*k*-NN and LM*k*-NN produce the same accuracy.

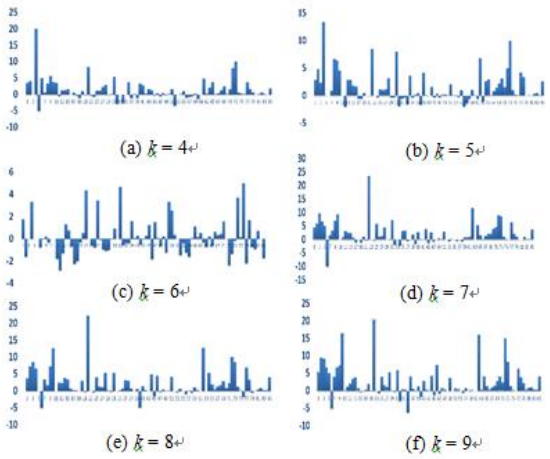


Figure 3. Comparing accuracy of WLM*k*-NN and LM*k*-NN, the Y-axis values in these charts are calculated by subtracting the accuracy rates of WLM*k*-NN and those of LM*k*-NN. The higher positive values in the Y-axis, the better WLM*k*-NN.

5.3 Comparing WLMk-NN with 1-NN

In this subsection, we compare WLM*k*-NN with 1-NN, a common used algorithm in the field of time series data. As from Fig. 4 WLM*k*-NN outperforms 1-NN in many datasets with different values of *k*. Especially with *k* = 9, it wins LM*k*-NN in 53 datasets, lose in 29 datasets, and draws in 3 datasets. The other values of *k* also show that WLM*k*-NN is better than 1-NN in many datasets. Details of the comparison is shown in Fig. 5 in which we show a column chart where each column presents the subtraction of accuracy rates of the two methods. In general, the more columns deviate in the positive area of the charts, the better our method.

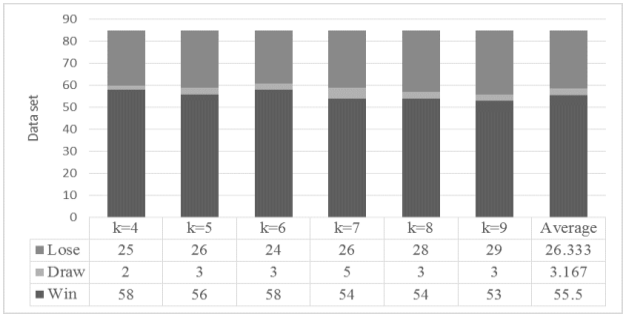


Figure 4. Stacked column chart comparing WLM*k*-NN and 1-NN.

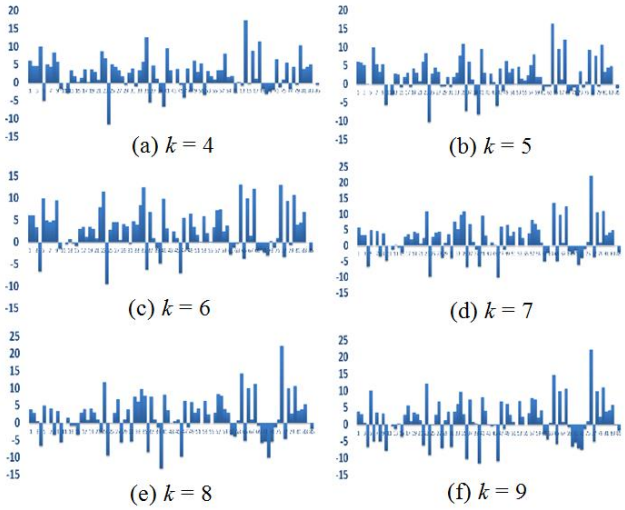


Figure 5. Comparing accuracy of WLM*k*-NN and 1-NN.

5.4 Comparing WLMk-NN with k-NCN

In this experiment, we compare WLM*k*-NN with *k*-NCN, in that we take values of $k = \overline{4,9}$. The summary comparison is shown in Fig. 6. As we can see, with *k* = 9, WLM*k*-NN wins *k*-NCN in 59 datasets, lose in 21 datasets, and draws in 5 datasets. The other values of *k* also show that WLM*k*-NN is better in many datasets. Details of the comparison are shown in Fig. 7 in which we show a column chart. In general, the more columns deviate in the positive area of the charts, the better our method.

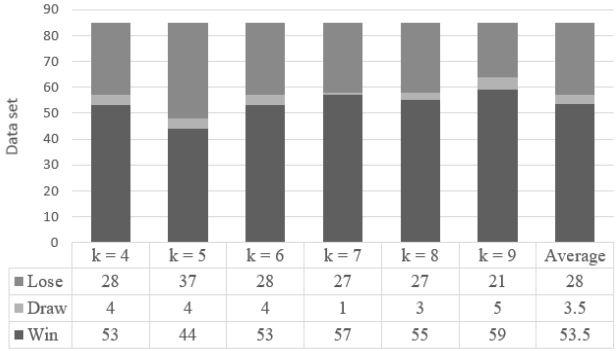


Figure 6. Stacked column chart comparing WLM*k*-NN with *k*-NCN.

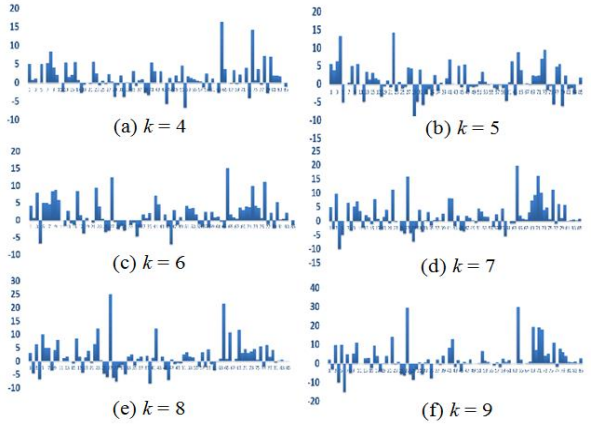


Figure 7. Comparing accuracy of WLM*k*-NN and *k*-NCN.

5.5 Comparing WLMk-NN with LMk-NCN

The last experiments is to compare WLM*k*-NN with LM*k*-NCN.

In Fig. 8, we show the summarized comparison which reveal that WLM*k*-NN outperform LM*k*-NCN with the winning rate average is 51.5 datasets. Especially with k = 9, WLM*k*-NN wins in 57 datasets, draws in 4 datasets and loses in 24 datasets. And the other values of k WLM*k*-NN also give performance better than those of LM*k*-NCN. We show the details of this comparison in Fig. 9 in which the column chart is also used.

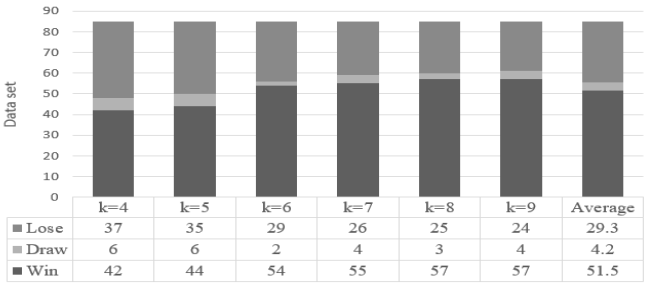


Figure 8. Stacked column chart comparing WLM*k*-NN and LM*k*-NCN.

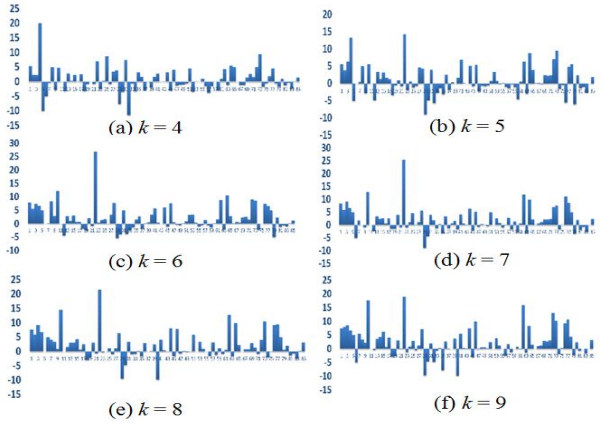


Figure 9. Comparing accuracy of WLM*k*-NN and LM*k*-NCN.

6. CONCLUSIONS

In this work, we proposed a method for time series classification called WLM*k*-NN which improves LM*k*-NN. The WLM*k*-NN is added some weights when we calculate the local mean vectors, therefore it can deal better with outliers than LM*k*-NN does. The new classification method is then applied on time series data. We compare our method with baselines including LM*k*-NN, LM*k*-NCN, *k*-NCN, and 1-NN. The experiments were conducted on 85 datasets from UCR Time Series Classification Archive. The experimental results show that our proposed method provides a better classifier for time series. These results also help us to conclude that 1-NN is not likely to be the best method for time series as many earlier work mentioned.

As for future work, we intend to adapt this method with a better distance measure Dynamic Time Warping (DTW) [6]. Although DTW is a better distance for time series compared to Euclidean Distance, computing the mean vector in this case is also a challenge.

7. ACKNOWLEDGMENTS

Two first authors contributed equally and should be considered as joint first authors. Corresponding authors: Hien T. Nguyen, Tuan M. Nguyen.

8. REFERENCES

[1] B. B. Chaudhuri. *A new definition of neighborhood of a point in multi-dimensional space*. Pattern Recognition Letters, 17(1):11–17, 1996.

[2] Y. Chen, B. Hu, E. J. Keogh, and G. E. A. P. A. Batista. *DTW-D: time series semi-supervised learning from a single example*. In The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013, pages 383–391, 2013.

[3] 11-14, 2013, pages 383–391, 2013. [3] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista. *The ucr time series classification archive*, July 2015. www.cs.ucr.edu/˜eamonn/time series data/.

[4] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. *Querying and mining of time series data: experimental comparison of representations and distance measure*s. PVLDB, 1(2):1542–1552, 2008.

[5] J. Gou, Z. Yi, L. Du, and T. Xiong. *A localmean-based k-nearest centroid neighbor classifier*. Comput. J., 55(9):1058–1071, 2012.

[6] E. J. Keogh and C. A. Ratanamahatana. *Exact indexing of dynamic time warping*. Knowl. Inf. Syst., 7(3):358–386, 2005.

[7] Y. Mitani and Y. Hamamoto. *A local mean-based nonparametric classifier*. Pattern Recognition Letters, 27(10):1151–1159, 2006.

[8] J. S. Sa ́nchez, F. Pla, and F. J. Ferri. *On the use of neighbourhood-based non-parametric classifiers*. Pattern Recognition Letters, 18(11-13):1179–1186, 1997.

[9] L. Wei and E. J. Keogh. Semi-supervised time series classification. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, USA, August 20-23, 2006, pages 748–753, 2006.

[10] X. Xi, E. J. Keogh, C. R. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In Machine Learning, *Proceedings of the Twenty-Third International Conference* (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006, pages 1033–1040, 2006.