Improving the robustness of 'online agglomerative clustering method' based on kernel-induce distance measures

Daoqiang Zhang[1,2*], Songcan Chen[1,2], Keren Tan[1]

[1] *Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, P.R. China*

[2] *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, 100080, P.R. China*

Abstract

Recently, an online agglomerative clustering method called AddC (Guedalia etc, Neural Computation, 1999) was proposed for nonstationary data clustering. Although AddC possesses many good attributes, a vital problem of that method is its sensitivity to noises, which limits its use in real-word applications. In this paper, based on kernel-induced distance measures, a robust online clustering (ROC) algorithm is proposed to remedy the problem of AddC. Experimental results on artificial and benchmark data sets show that ROC has better clustering performances than AddC, while still inheriting advantages such as clustering data in a single pass and without knowing the exact number of clusters beforehand.

Keywords: Competitive learning; Robustness; Kernel-induced measure; Online clustering; Nonstationary

---

[*] Corresponding author. Tel.:+86-25-8489-2805.

E-mail address: daoqz@mail.com (D. Zhang), s.chen@nuaa.edu.cn (S. Chen), tankeren@hotmail.com (K. Tan).

1. Introduction

Clustering analysis is the process of grouping data (patterns) into clusters such that the patterns in a cluster are more similar to each other than to patterns in different clusters under some measure of distance or similarity [5]. Typically, clustering algorithms can be divided into two classes, batch and on-line. Batch algorithms process data off-line; hence the temporal structure of generating data is usually ignored. On the other hand, most of the existing on-line clustering methods assume stationarity of the data. When used to cluster nonstationary data, these methods fail to generate a good representation for given data. Here by "nonstationary", we mean that on a short time scale, it is pseudo-stationary, while on the long time scale, the process has a sequential property [3]. In our real-world and science discipline, there are a large amount of these nonstationary data, such as stock market indexes and video streams transferred across the Internet [2]. To effectively process these nonstationary data, an online clustering algorithm called AddC was proposed recently by Guedalia *et al* [3]. It has been reported that the AddC has great advantages over traditional methods such as the online k-means [5] and the EquiDistortion [7] when used to cluster nonstationary data. However, a main drawback of the AddC is very sensitive to noise and thus results in its lack of robustness, which limits its use in real-world applications.

Based on the well-established kernel method [1, 6], we propose a robust online clustering algorithm, *i.e.* ROC for clustering nonstationary data. The kernel method in the machine learning theory refers to increasing the computational power of linear methods by mapping the data into high-dimensional feature space and has shown its great power in a number of kernel-based learning machines, *e.g.* support vector machines (SVMs) [1] and kernel principal component analysis (KPCA) [6]. And in one of our previous works [8], a robust batch clustering algorithm has been developed for clustering incomplete data using the kernel method. In this letter, we generalize the algorithm in [8] to make it able to cluster on-line nonstationary data. We carry out several experiments to compare the performances between the proposed ROC algorithm and the existing AddC algorithm. Experimental results on artificial nonstationary data set and 13 benchmark data sets show that ROC achieves better performances than AddC in most cases.

The rest of this paper is organized as follows: we present the kernel-induced distance measures and whole ROC algorithm in section 2. In section 3, some experimental results are given, and finally in section 4, we conclude and give some directions for further research.

2. The proposed ROC algorithm

2.1 Kernel-induced distance measure

Suppose we are given an input set X, and a mapping function φ that maps $x_i \in X$ from the input space X to a new space F with higher or even infinite dimensions. The kernel function is defined as the inner product in the new space F:

$$K(x,y) = \langle \varphi(x), \varphi(y) \rangle \qquad (1)$$

where $x, y \in X$, and $\langle \cdot, \cdot \rangle$ is the inner product operation in the new space.

An important fact about kernel function is that it can be constructed without knowing the concrete form of φ [1]. Namely, the transform is defined implicitly. There are several typical kernel functions, *e.g.* the radial basis function (RBF) kernel: $K(x,y) = \exp(-\sum_i | x_i - y_i |^b / \sigma^2)$ $(0 < b \le 2)$ and the polynomial kernel (PK): $K(x,y) = (x^T y + 1)^d$. For all RBF kernels, $K(x,x) = 1, \forall x \in X$, and the RBF kernel will become the Gaussian kernel (GK) when *b*=2.

In the original AddC algorithm, a Euclidean norm is adopted as the distance measure. Here we develop a novel kernel-induced distance $d(x,y)$ defined as follows

$$\begin{aligned} d(x,y)^2 &= \|\varphi(x)-\varphi(y)\|^2 \\ &= \varphi(x)^T\varphi(x) - 2\varphi(x)^T\varphi(y) + \varphi(y)^T\varphi(y) \\ &= K(x,x) - 2K(x,y) + K(y,y) \end{aligned} \tag{2}$$

The above distance $d(x,y)$ in the feature space corresponds exactly to a class of new non-Euclidian distances or measures in the original space with varying kernels. It has been proved in [8] that the measures based on the RBF kernels including GK are all robust but the measure induced by the PK is not according to Huber M-estimator theory [4]. In summary, different kernels can induce different distance measures with different properties and thus can induce different clustering algorithms. In this letter, we only consider the Gaussian kernel for the simplicity of representation. From the above discussions, it is obvious that when the Gaussian kernel is used in Eq. (2), the distance $d(x,y)$ can be simplified as $d(x,y)^2 = 2 - 2K(x,y)$.

2.2 Proposed ROC algorithm

In this section, we are in a position to present the robust online clustering (ROC) algorithm based on the kernel-induced distance measure introduced in the last section. Similar to the AddC algorithm, the ROC algorithm can also be divided into three main steps. For a new arriving data point, we first look for and update the winner among the prototypes using the kernel-induced measure. Then we merge the two closest prototypes in order to obtain a redundant prototype for future learning. At last, we remove all clusters with negligible data points. The detailed description of the proposed algorithm is as follows (Note that we follow the notation used in [3]). There is a parameter controlling the scale of the desired solution in the ROC, which is denoted by $N_{\max}$, a possible maximum number of prototypes available. The final number of prototypes may be less than $N_{\max}$ because of the removing of clusters in the last step. Apparently, different values for $N_{\max}$ may result in clustering results with different scales. Generally the value of $N_{\max}$ is given in advance.

The proposed ROC algorithm:

Step 1: Set the threshold ε, a parameter used to control the final number of clusters, and initialize the system with zero prototypes: $N = 0$.

Step 2: Input a new data point x. The prototype closest to the data point is defined as the winner: *winner* = $\arg\min_{1 \le i \le K}(d(x, y_i))$, where $d(x, y_i)$ is defined in Eq. (2). Update the *winner prototype* y_{winner} and its weight c_{winner} as follows:

$$c_{winner} = c_{winner} + K(x, y_{winner})\,;\;\; y_{winner} = y_{winner} + \frac{x - y_{winner}}{c_{winner}} \tag{3}$$

where the kernel function *K*(*x*, *y*) as defined in Eq. (1).

Step 3: If $N < N_{\max}$, then $N = N + 1$, set $\delta = N$, and go to *Step* 5.

Step 4: Find the two closest prototypes: $\{\gamma, \delta\} = \arg\min_{\substack{1 \le \gamma \le \delta \le K \\ \gamma \ne \delta}}(d(y_\gamma, y_\delta))$. Merge the two prototypes using the following equation:

$$y_\gamma = \frac{y_\gamma c_\gamma + y_\delta c_\delta}{c_\gamma + c_\delta}\,;\;\; c_\gamma = c_\gamma + c_\delta \tag{4}$$

Step 5: Initialize the prototype y_γ with the new data x and set its weight to zero: $y_\gamma = x$; $c_\gamma = 0$.

Step 6: while there remains data to be clustered, go to *Step* 2.

Step 7: Post-processing: Remove all clusters with a negligible weight, *i.e.* $c_\alpha < \varepsilon$.

Note that the proposed ROC algorithm is similar to the AddC algorithm, but there are two major differences: one is that the ROC uses a kernel-induced distance measure to replace the original Euclidean norm in AddC, and different kernel functions can induce different kinds of distance measures and thus different clustering algorithms in ROC. The other is the update of the weight c_{winner}, it is updated with the kernel $K(x, y_{winner})$ between the data x and the winner y_{winner} rather than simply set to constant 1 as in the *whole* clustering course of the AddC. Possibly, it is a very point that makes the AddC sensitive to noise and outliers. For example, when we take the Gaussian RBF kernel, $K(x, y_{winner})$ will approach to 1 for x near to y_{winner}; on the other hand, $K(x, y_{winner})$ will approach to 0 for x far away from y_{winner}, *i.e.* an outlier. As a result, the outlier does not cause too much effect on the prototype y_{winner}. It is the distance information used in the weight of the winner prototype that is expected to make ROC more robust to noise or outlier than AddC.

3. Simulation results

3.1 Artificial data set

We use the same dataset as in [3] in the experiment. At first glance, *i.e.* when viewed at low resolution, the dataset contains 3 clusters, as shown in Fig. 1(a). However, when viewed at high resolution, the same dataset has 9 clusters, as shown in Fig. 1(b). When the dataset is not corrupted by noise, both AddC and ROC can correctly clustering the data under different scales by setting the parameter $N_{\max}$ with corresponding different scales. We use the Gaussian kernel with σ = 1 for ROC. Fig. 1(a) and (b) show the results of AddC and ROC with $N_{\max}$ = 4 and 10 respectively. For more values of $N_{\max}$, the results are in Table 1. It can be seen that ROC has equivalent performance with AddC without noises.

Fig. 1(c), (e) and (g) show the clustering results with $N_{\max}$ = 4 when corrupted by 10%, 20% and 30% random noises respectively, and Fig. 1(d), (f) and (h) show the clustering results with $N_{\max}$ = 10 when corrupted by 10%, 20% and 30% random noises respectively. Table 1 shows the number of clusters found by AddC and ROC respectively under different N_{max} values when corrupted by 20% random noises.

According to Fig.1, AddC is very sensitive to the added noises and its performance deteriorates greatly as the level of noises increases. However, ROC achieves nearly the same result as in Fig. 1(a) when $N_{\max} = 4$ under all levels of noises. That is, ROC successfully eliminates the disturbance of noises at low resolution. When $N_{\max} = 10$, ROC both find three clusters under 10% and 20% added noises respectively, unlike the nine clusters in Fig. 1(b) without noises. We guess that the random noises have destroyed the fine structures of the dataset, and thus affected the final clustering at high resolution. And ROC begin to fail under 30% added random noises for $N_{\max} = 10$, as shown in Fig. 1(h), where a noise cluster, besides the normal data clusters, is discovered by ROC.

3.2 Benchmark data sets

In the following experiment, we test the performances of AddC and ROC on 13 benchmark data sets [1]. For each data set, we only use the first group of training patterns for clustering. Table 2 shows the total number of patterns and misclassified number of patterns by AddC and ROC respectively. In the ROC algorithm, the Gaussian kernel with $\sigma = 5$ is used for all the 13 data sets.

According to Table 2, for 'banana', 'flare-solar' and 'image' data sets, the clustering performances of the AddC are all a little superior to those of ROC. While for the rest data sets, ROC has all the same or better performance than AddC. And such a contrast is especially distinct for ''heart', 'twonorm' and 'waveform' data sets, where ROC has 2-4 times less misclassified numbers of patterns than AddC. The corresponding classification errors of ROC on ''heart', 'twonorm' and 'waveform' data sets are 21.18%, 11.25% and 19.75% respectively. Remember that there is only one pass in the clustering process of the ROC algorithm and hence the execution speed is very fast. The above performance of ROC is very competitive.

[1] available at: http://web.rsise.anu.edu.au/~raetsch/data/index.html

4. Conclusions

Base on a kernel-induced measure, a robust online clustering algorithm named ROC is proposed in this paper to breakthrough the limit of the AddC algorithm. We compare experimentally the performances of ROC and AddC on the artificial and benchmark data sets. Experimental results show that ROC is more robust to noises and has less classification errors than AddC in most cases.

In this paper, only the Gaussian kernel is used for the simulations. Furthermore, we fix the kernel parameter σ to 5 in the real data experiments here for simplicity. The classification errors of ROC can actually be further reduced if we optimize the parameter, which will be the ongoing and future research. Moreover, other type of kernels such as the polynomial kernel can also be exploited and investigated.

Acknowledgements

This work was supported in part by the National Science Foundations of China and of Jiangsu under Grant Nos. 60271017 and BK2002092 respectively.

References

[1] N. Cristianini, J.S. Taylor, An Introduction to SVMs and Other Kernel-based Learning Methods, Cambridge University Press, 2000.

[2] D. Deng, N. Kasabov, On-line pattern analysis by evolving self-organizing maps, Neurocomputing 51 (2003) 87-103.

[3] I.D. Guedalia, M. London, M. Werman, An on-line agglomerative clustering method for nonstationary data, Neural Computation 11 (1999) 521-540.

[4] P J. Huber, Robust statistics, Wiley, New York, 1981.

[5] A.K. Jain, R.C. Dubes, Algorithms for clustering data. Englewood Cliffs NJ: Prentice Hall, 1988.

[6] B. Schölkopf, A. Smola, K.R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (5) (1998) 1299-1319.

[7] N. Ueda, R. Nakano, A new competitive learning approach based on an equidistortion principle for designing optimal vector quantizers, Neural Networks, 7 (8) (1994) 1211-1227.

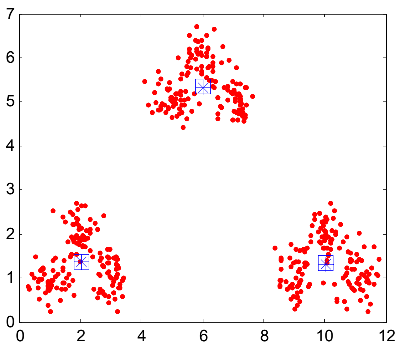
[8] D. Zhang, S. Chen, Clustering incomplete data using kernel-based fuzzy c-means algorithm, Neural Processing Letters 18 (2003) 155-162.

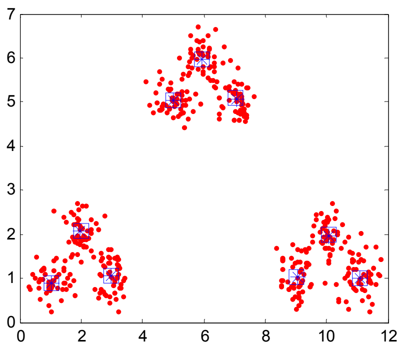
Figure captions

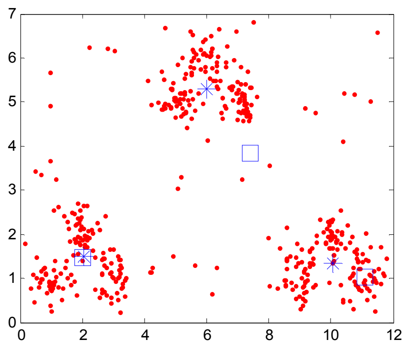
Fig. 1 Performance of AddC and ROC in clustering nonstationary data: (a) $N_{max} = 4$ without noise, (b) $N_{max} = 10$ without noise, (c) $N_{max} = 4$ with 10% noises, (d) $N_{max} = 10$ with 10% noises, (e) $N_{max} = 4$ with 20% noises, (f) $N_{max} = 10$ with 20% noises, (g) $N_{max} = 4$ with 30% noises, (h) $N_{max} = 10$ with 30% noises.

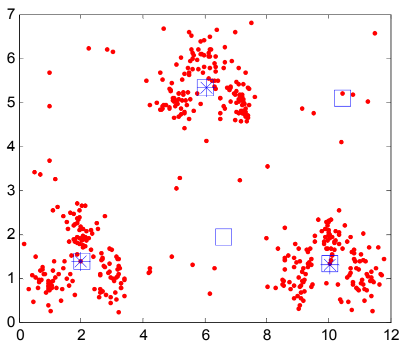
•(dot)-data points, □(square)-prototypes of AddC, *(star)-prototypes of ROC.

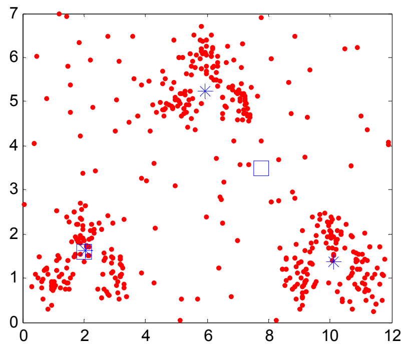
List of Tables

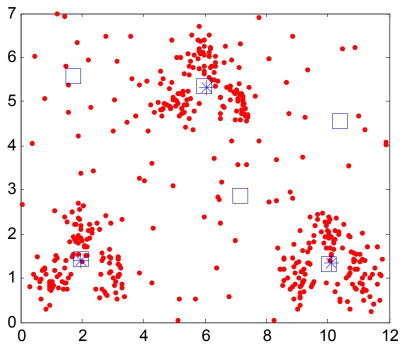

(a)

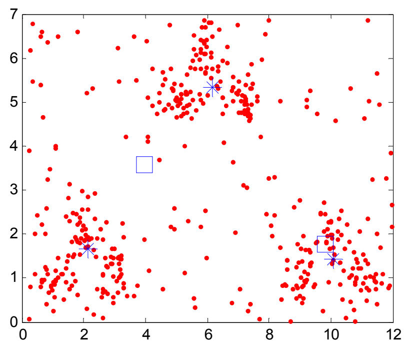

(b)


(c)


(d)


(e)


(f)


(g)

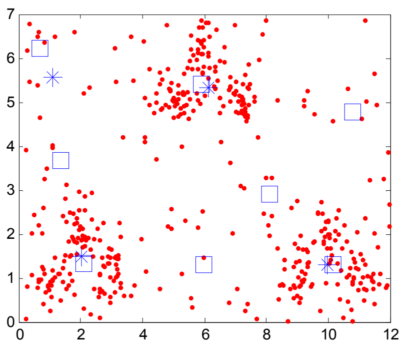

(h)

Fig. 1 Performance of AddC and ROC in clustering nonstationary data: (a) N_{max} = 4 without noise, (b) N_{max} = 10 without noise, (c) N_{max} = 4 with 10% noises, (d) N_{max} = 10 with 10% noises, (e) N_{max} = 4 with 20% noises, (f) N_{max} = 10 with 20% noises, (g) N_{max} = 4 with 30% noises, (h) N_{max} = 10 with 30% noises.
•(dot)-data points, □(square)-prototypes of AddC, *(star)-prototypes of ROC.

Table 1 Number of clusters found by AddC and ROC respectively under different N_{max} values

N_{max}		1	2	3	4	5	6	7	8	9	10
Without noise	AddC	1	1	2	3	4	5	6	7	8	9
	ROC	1	1	2	3	4	5	6	7	8	9
With 20% noise	AddC	1	1	2	2	3	4	5	6	7	6
	ROC	1	1	2	3	3	3	3	3	3	3

Table 2 Clustering performances of AddC and ROC on 13 benchmark data sets[1]

Data Set	Total number of patterns	Misclassified number of AddC	Misclassified number of ROC [2]
banana	400	**<u>137</u>**	138
breast-cancer	200	53	**<u>46</u>**
flare-solar	666	**<u>307</u>**	315
diabetis	468	165	165
german	700	217	217
heart	170	74	**<u>36</u>**
image	1300	**<u>566</u>**	568
ringnorm	400	194	194
splice	1000	484	**<u>424</u>**
thyroid	140	34	34
titanic	150	39	39
twonorm	400	194	**<u>45</u>**
waveform	400	183	**<u>79</u>**

[1] Data sets available from: http://web.rsise.anu.edu.au/~raetsch/data/index.html

[2] The kernel function used in ROC algorithm is Gaussian kernel with σ = 5 for all data sets