Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 72 (2009) 1283–1295
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Projective ART with buffers for the high dimensional space clustering and an application to discover stock associations ☆

Lian Liu[a], Lihong Huang[a,*], Mingyong Lai[b], Chaoqun Ma[c]

[a] College of Mathematics and Econometrics, Hunan University, Changsha, Hunan 410082, PR China
[b] School of Economics and Business, Hunan University, Changsha, Hunan 410082, PR China
[c] Management School, Hunan University, Changsha, Hunan 410082, PR China

Received 28 February 2006; received in revised form 18 July 2007; accepted 26 January 2008
Communicated by D.-S. Huang
Available online 4 March 2008

Abstract

Unlike to traditional hierarchical and partitional clustering algorithms which always fail to deal with very large databases, a neural network architecture, projective adaptive resonance theory (PART), is developed for the high dimensional space clustering. However, the success of the PART algorithm depends on both accurate parameters and satisfied orders of input data sets. These disadvantages prevent PART from being applied to realtime databases. In this paper, we propose an improved method, PART with buffer management, to overcome these disadvantages. The major contributions of our method are introducing a buffer management and a new similar degree function and buffer checkout process. The buffer management mechanism allows data sets not to be immediately clustered to one cluster. The purpose of the average similar degree is to successfully work with high similar noise data sets and partly achieve an order-independent objective without correct parameters. And the average similar degree has a good attribute, the parameter-tolerance. Namely, the clustering result does not depend on the precise choice of input parameters, and different parameter values have close clustering results including dimensions associated with clusters. The buffer checkout process can handle a huge amount of input data sets by a small buffer space. Also, simulations and comparisons in high dimensional spaces are reported, and an application by using our algorithm to find stock concurrence association rules is given finally.

Keywords: Neural network; Data mining; Clustering; Buffer management; PART; Bayes rule; Stock concurrence association

1. Introduction

Clustering is an unsupervised classification process to discover pattern hidden in a huge data set or high dimensional spaces. Many scholars have extensively explored clustering problems from different methodology perspectives [10,11,19]. And with widespread availability of the information age, a great number of very high dimensional data sets are more likely to be produced. As the noted statistician, Professor David L. Donoho, pointed out [9], "The coming century is surely the century of data". Traditional hierarchical and partitional clustering algorithms, however, always fail to deal with very large and high dimensional databases, especially in realtime situations such as text processing [18], video classification [13], biology networks [17,16] and social networks [12]. It is inefficient for most clustering algorithms to cluster in high dimensional spaces due to "the inherent sparsity of the data" [4,2,14].

Fortunately, PART, Projective Adaptive Resonance Theory [3], based on the ART [5] and PROCLUS [1], is very good at recognizing self-organizing patterns in arbitrary sequences [6–8,15]. Instead of considering in the total space, PART focuses on the subspace D in which a subset C of data points are very close to each other. The major contribution of PART is the selective output

☆ Research supported by the National Science Foundation of China (10371034, 70371028), the Specialized Research Fund for the Doctoral Program of Higher Education (20050532023) and "985 Project".
* Corresponding author. Tel./fax: +86 731 8823056.
E-mail address: lhhuang@hnu.cn (L. Huang).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.01.020

signaling mechanism. It allows the signal, generated in an input layer node, to be transmitted to a clustering layer node only when the signal is similar to the top-down weight between the two nodes. Therefore, it can successfully find projected clusters in high dimensional spaces.

Nevertheless, PART relies on some parameters, especially ρ (ρ presents the least similar degree of data sets placed in the same cluster). The clustering accuracy may be seriously degraded if an incorrect value is chosen. In realtime situations, it is rarely possible for users to supply accurate parameters. In most cases, it causes practical difficulty to apply PART to realtime data sets.

On the one hand, some high similar noise data sets maybe cause seriously inaccurate clustering results. These noise data sets may be correlated in several dimensions, and the number of such dimensions may be bigger than the ρ parameter. But in fact they do not belong to one cluster. In the paper [3], authors increase the ρ step by step to figure out the good result, but it is impossible in realtime situations since computers need too much time to find out the exact step value when the data are too large. The example simulations of high similar noise data sets are given in Section 4.1.

On the other hand, the dimensions of projected clusters are sensitive to the choice of the input parameter ρ. This property can be shown from the comparison in Section 4.2. As a simulation experiment introduced in paper [3], PART presents different clustering results of projected dimensions with different values of the parameter ρ. Although the number of data sets found in these experiments is basically consistent with that of real clusters, the projected dimensions associated with clusters are incomplete with regard to the real dimensions.

Sometimes, the projected dimensions associated to a cluster are more valuable than the number of data in the cluster. Users always hope to get the completely helpful information at one time. Apparently, PART cannot satisfy their appetite since PART requires the accurate parameter ρ in order to find complete projected dimensions associated to one cluster. In the paper [3], increasing step by step the parameter ρ could surely get a precise result including the projected dimensions. Nonetheless, in realtime situations we face hundreds of input data sets, so we do not have many time to deal with a huge number of data!

In this paper, in order to relieve PART from a parameter-laden dilemma, we propose a significant improvement, buffer management, that can neglect the noise data sets and achieve a parameter-free algorithm. The basic architecture of PART with buffer management is similar to that of the PART architecture. The simplified configuration of PART with buffer management is shown in Fig. 1.

In this architecture, there are an input-comparison field (F_1 layer), a clustering layer (F_2 layer), a reset subsystem and a sublayer hidden in F_1 layer which selectively sends signals to nodes in the F_2 layer. One node in the F_1 layer can be active to some F_2 nodes, but inactive to the other F_2 nodes. Two types of connections are existed between each node pair in the F_1 and F_2 layers. The connection from the F_1 layer to the F_2 layer is weighted by z_{ij}, the bottom-up value. While the top-down value, z_{ji}, denotes the weight of the top-down connection. Both weights should be modified in the control of two learning rules. And a winner-take-all paradigm is implemented so that the F_2 node with the largest net input becomes the candidate winner to get the input pattern. After that, an F_2 node is considered committed to learn some input patterns before, otherwise noncommitted. Only the committed nodes can accept signals from the F_1 layer, and noncommitted F_2 nodes cannot take signals. If all committed F_2 nodes are not suitable to learn the input pattern, we randomly select the

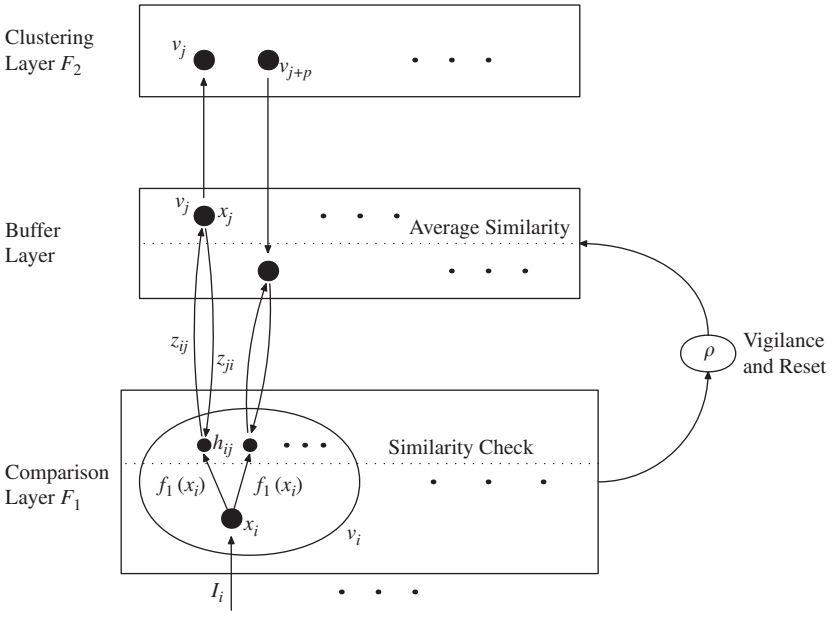

Fig. 1. The architecture of BPART.

winner node from noncommitted F_2 nodes. A winner node is a chosen committed F_2 node which has finally accept the input data set, or the input pattern. Moreover, we add an outlier cluster, C_{outlier}, which collects data sets clustered not to any F_2 nodes. Therefore, the outlier cluster serves as a trash can which does not learn any input pattern. Meanwhile, the reset mechanism decides whether the candidate will learn the input pattern. The purpose of the reset mechanism is to determine the similar degree of data sets in a cluster.

The principal contributions of BPART[1] are as follows:

1. Buffer management (the buffer layer). It manages the nodes whose similar degrees are less than the average similar degree but more than the parameter ρ.
2. New similar degree function avg_j of the cluster v_j. It is 0 at the beginning. When a input data set is ready to be grouped into one candidate cluster, BPART calculates the new average similar degree of this candidate cluster. If the average similar degree is more than the similar degree of the input data set, it is pushed into the buffer. Otherwise, the input data set is grouped into this candidate cluster.
3. Buffer checkout. We check whether similar degree of any pushed data set in the buffer is more than or equal to the new average degree. If yes, we pull this data set to the cluster. If the buffer is full, BPART will discard the data set whose similar degree is smallest in the buffer to the outlier cluster.

The remainder of this paper is organized as follows. In Section 2, we describe the BPART architecture, the STM and LTM equations, the vigilance and reset mechanism, and the buffer management mechanism. In Section 3, we provide the BPART algorithm and pseudocodes in detail. Examples and simulations in high dimensional spaces are presented in Section 4. In Section 5, we apply BPART to discover stock concurrence association rules in 100 stock transactions of the Hang Seng Composite Index of HongKong. Conclusions and Acknowledgements are provided in Sections 6 and 7, respectively.

2. PART with buffer management

2.1. Basic architecture of BPART

In the basic BPART architecture as shown in Fig. 1, v_i $(i=1,\ldots,m)$ stands for nodes in the F_1 layer, whose activation is x_i; v_j $(j=m+1,\ldots,m+n)$ denotes jth-node in the F_2 layer, whose activation is x_j. v_j is also a cluster who learns the input pattern. The number of input data sets clustered to v_j is n_j. And the range of activation x_i is W_i. Avg_j is the average similar degree of the cluster v_j. And an F_2 node is called committed if it has learned some input patterns before, otherwise noncommitted.

The bottom-up weight from v_i to v_j is z_{ij}, and the top-down weight from v_j to v_i is z_{ji}. According to paper [3], the term h_{ij} is defined as the selective output signal from node v_i in the F_1 layer to a committed node v_j in the F_2 layer as follows:

$$h_{ij} = h(x_i, z_{ij}, z_{ji}) = h_\sigma(d(f_1(x_i), z_{ji}))l(z_{ij}), \tag{1}$$

where

$$d(a,b) = |a-b|/(e+|b|), \tag{2}$$

for given constants σ and θ, $h_\sigma(a,b)$ is given by

$$h_\sigma(a,b) = \begin{cases} 1 & \text{if } d(a,b) \leqslant \sigma, \\ 0 & \text{if } d(a,b) > \sigma, \end{cases} \tag{3}$$

and $l(z_{ij})$ is given by

$$l(z_{ij}) = \begin{cases} 1 & \text{if } z_{ij} > \theta, \\ 0 & \text{if } z_{ij} \leqslant \theta. \end{cases} \tag{4}$$

If $h_{ij}=1$, we consider that v_i is active to v_j; otherwise, v_i is inactive to v_j. We also simply take function f_1 as PART, $f_1(x_i)=x_i$.

2.2. STM equations

The STM activation x_i and x_j of any F_1 node v_i and any F_2 node v_j obey a membrane equation, respectively:

$$\varepsilon \frac{\mathrm{d}x_i}{\mathrm{d}t} = -x_i + I_i \tag{5}$$

and

$$\varepsilon \frac{\mathrm{d}x_j}{\mathrm{d}t} = -x_j + (1 - Ax_j)J_j^+ - (B + Cxj)J_j^-, \tag{6}$$

where

$$0 < \varepsilon \ll 1, \tag{7}$$

$$J_j^+ = g(x_j) + T_j, \tag{8}$$

$$J_j^- = \sum_{k \neq j, v_k \in F_2} g(x_k), \tag{9}$$

$$g(x_i) = \begin{cases} 0, & x_i < \theta, \\ 1 & \text{otherwise.} \end{cases} \tag{10}$$

Here, I_i is the input data set. A, B and C are all constants. Particularly, A, B and C are assigned to 1, 0, 1, respectively. Input J_j^+ adds a positive feedback signal $g(x_j)$ from v_j to itself in addition to the bottom-up adaptive filter input T_j.

Regarding T_j, PART simply uses the equation $T_j = \sum z_{ij}h_{ij}$ which neglects the important factor of volumes of one cluster. In other words, a big cluster has more chances to learn patterns than a small cluster even though two clusters have a same number of projected dimensions in response to one input data set. Therefore, in contrast to

[1] BPART is short for PART with buffer management in the rest of this paper without special hints.

PART, we have the formula by Bayes rules:

$$T_j = T_{j|x_{i_1}i_2\ldots i_p} = \frac{P(x_{i_1}x_{i_2}\ldots x_{i_p}|v_j)P(v_j)}{P(x_{i_1}x_{i_2}\ldots x_{i_p})} = \frac{P(x_{i_1}|v_j)P(x_{i_2}|v_j)\ldots P(x_{i_p}|v_j)P(v_j)}{P(x_{i_1})P(x_{i_2})\ldots P(x_{i_p})}, \quad (11)$$

where

$$p = \sum_i h_{ij}, \quad (12)$$

$$P(x_i|v_j) = 1 - \frac{|x_i - z_{ji}|}{W_i}, \quad (13)$$

$$P(v_j) = \frac{n_j}{\sum_k n_k}, \quad (14)$$

$$P(x_i) = \sigma. \quad (15)$$

The F_2 layer makes a decision by the winner-take-all paradigm:

$$f_2(x_j) = \begin{cases} 1 & \text{if node } v_j \text{ is a winner,} \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

2.3. Buffer management mechanism

Similar to PART, the candidate winner node v_j is chosen by: Let $\Gamma = T_k : F_2$ node v_k is committed but not reset on the current trial. Then the node v_j is a candidate winner if $\Gamma \neq \emptyset$ and $T_j = \max \Gamma$.

For a candidate winner node v_j, if its similar degree is more than or equal to the avg_j, or if $\Gamma = \emptyset$ and node v_j is the next noncommitted node in F_2 layer, then the candidate winner node v_j becomes the winner; otherwise, the candidate winner node v_j is reset.

We calculate the new avg_j as follows:

$$avg_j = \begin{cases} x & \text{if } avg_j \text{ is zero,} \\ avg_j * \dfrac{(n_j - 1) * (n_j - 2)/2}{n_j * (n_j - 1)/2} & \\ \quad + x * \dfrac{n_j - 1}{n_j * (n_j - 1)/2} & \text{otherwise,} \end{cases} \quad (17)$$

where x is the similar degree of the current input data set with regard to the candidate winner node v_j.

We could see that with the increasing number of input data sets, n_j, in one cluster, the effect of sparse noise data sets on avg_j is very small. Hence, the stable avg_j can reflect maximum projected dimensions associated with one cluster in spite of the inaccurate choice ρ.

Eq. (17) could be simplified as follows:

$$avg_j = \begin{cases} x & \text{if } avg_j \text{ is zero,} \\ avg_j * \dfrac{n_j - 2}{n_j} + x * \dfrac{2}{n_j} & \text{otherwise.} \end{cases} \quad (18)$$

When a candidate winner node becomes a winner, we check whether the similar degree of any buffer data set is more than or equal to the new similar average degree. If yes, we pull the buffer data set to the winner node. And if the buffer is full, BPART will pull one buffer data set, whose similar degree is smallest, to the outlier.

2.4. LTM equations

The LTM bottom-up weight equation is described by

$$\delta \frac{dz_{ij}}{dt} = f_2(x_j)\left[(1 - z_{ij})Lh(x_i, z_{ij}, z_{ji}) - z_{ij}\sum_{k \neq i, x_k \in F_1} h(x_k, z_{kj}, z_{jk})\right]. \quad (19)$$

The LTM top-down weight equation is presented by

$$\frac{dz_{ji}}{dt} = f_2(x_j)[-z_{ji} + f_1(x_i)] \quad \text{if } v_j \text{ is committed,} \quad (20)$$

$$\delta \frac{dz_{ji}}{dt} = f_2(x_j)[-z_{ji} + f_1(x_i)] \quad \text{if } v_j \text{ is noncommitted,} \quad (21)$$

where $0 < \varepsilon \ll \delta \ll 1$, and L is a positive constant. When a noncommitted F_2 node v_j is selected as the winner node, all F_1 nodes are immediately active to the winner.

2.5. Vigilance and reset

Similar to PART algorithm, the vigilance parameter ρ in BPART also decides the similar degree of data sets in the same cluster in F_2 layer. If a candidate winner F_2 node v_j does not meet the vigilance requirement, or the similar degree r_j is less than the vigilance parameter ρ, BPART will reset v_j not to become a winner during the current trial; otherwise, the winner node is ready to accept the input data set.

We reset the winner v_j if and only if

$$r_j < \rho, \quad (22)$$

where

$$r_j = \sum_i h_{ij}. \quad (23)$$

Here $\rho \in \{1, 2, \ldots, m\}$ is a vigilance parameter. It is easy to know that avg_j is more than or equal to ρ.

3. Algorithms and pseudocodes

3.1. Algorithm trees

1. Initialization
   Initialize parameters $L, \varepsilon, \alpha, \theta, \sigma, \rho, m, n$. m is the number of dimensions in input data sets, n is the expected maximum number of clusters.
   Set all F_2 nodes as being noncommitted.
   Set all $avg_j = 0$, $j = 1, \ldots, n$.

2. For the input data set S, do the following Steps 2(a)–(f), until users stop inputting data.
   (a) Compute h_{ij} for all F_1 nodes v_i and committed F_2 nodes v_j. Compute T_j for all committed F_2 nodes v_j. If all F_2 nodes are noncommitted, randomly select one noncommitted F_2 node as the candidate winner, then go to 2(c).
   (b) Select the candidate winner F_2 node v_j. If no candidate winner F_2 node can be selected, put the input data set into the outlier cluster and then continue to do Step 2.
   (c) Compute r_j.
   (d) If $r_j \geqslant \rho$, go to Step 2(e); otherwise reset the candidate winner v_j and go back to Step 2(a).
   (e) If $r_j \geqslant avg_j$, go to Step 2(f); otherwise push this input data set into the buffer, and renew the avg_j for the candidate winner node v_j. If the buffer is full, pull the buffer data set, whose similar degree is smallest, to the outlier. Then go to Step 2.
   (f) Set the candidate winner v_j as the winner and the committed node, and update the bottom-up and top-down weights for the winner node v_j, and renew the avg_j for the winner node v_j. Then check whether the similar degree of any buffer data set is more than or equal to the new average degree. If yes, put this buffer data set to the winner cluster.
3. Each committed F_2 node v_j represents a projected cluster C_j and the projected dimension set is D_j.

The algorithm block diagram is in Fig. 2. As presented above, the difference between BPART and PART is the buffer management which avoids a step-by-step increasing procedure. Every increasing procedure need completely deal with all input data sets. It is impossible to run in realtime situations because of the time complexity of PART. Although extra resources of the buffer management such as time and space are needed, we could know that extra resources we need are very limited through experiments in Section 4.

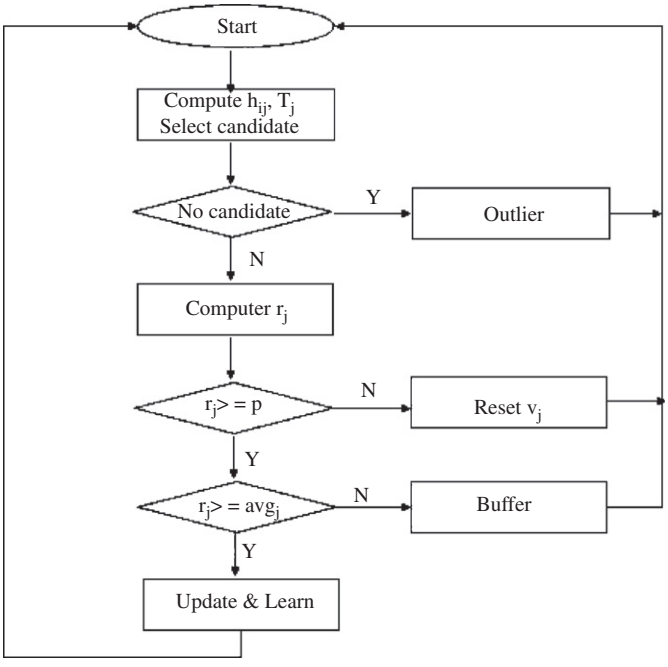

Fig. 2. Algorithm block diagram of the BPART algorithm.

3.2. Detailed phases and pseudocodes

In this section, we detail each phase as follows.

3.2.1. Computation of h_{ij} and T_j

From Eq. (1) we obtain

$$h_{ij} = h_\sigma(I_i, z_{ji}) l(z_{ij}). \tag{24}$$

The pseudocodes of computation of this equation are in Fig. 3.

3.2.2. Selection of the winner

We select the candidate winner according to rules introduced in Section 2.2. The number of clusters in the F_2 layer is up to n.

The pseudocodes of the selection procedure are in Fig. 4.

3.2.3. Vigilance and reset

The algorithm resets the candidate winner v_j if and only if

$$\sum_i h_{ij} < \rho \tag{25}$$

for a vigilance parameter $\rho \in \{1, 2, \ldots, m\}$.

```
procedure ComputationHandT;
begin
 {m: the number of dimensions in the input data set,
 n: the expected maximum number of clusters in F2,
 h: the selective output signal,
 vj: one of F2 layer nodes, every one is a cluster,
 zij, zji: the bottom-up and top-down weights,
 Tj: the bottom-up adaptive filter input.}


for i←1 to m do
  for j←1 to n do
  hij←0; {initialize hij}

for j←1 to n do
  if (vj is committed but not reset) then
  for i←1 to m do
    if (d(Ii,zji) ≤ σ and l(zij) > θ) then hij←1
     else hij←0
{θ is 0 or a small number, σ is the distance
parameter.}

for j←1 to n do
  T[j]←0;{initialize Tj}

for j←1 to n do
 if (vj is committed) then Tj ← ∑ zijhij

end;
```

Fig. 3. Computation of h_{ij} and T_j.

```
procedure Selection;
begin
 {Twin: the candidate winner,
 I: the current input data set,
 C_outlier: the outlier cluster,
 n_twin: the number of data sets in the v_twin cluster.}

Twin←0; T[Twin]←0;
for j←1 to n do
  if T[j]>T[Twin] then Twin←j;
{Select the candidate winner.}

IF Twin=0 then {If no candidate winner is selected, there are two
possible alternatives: one, no F2 node can be selected, put the
data set into the outlier cluster; two, all F1 nodes are
noncommitted, randomly select one uncommitted as the winner node.}
begin
  j←0;
  repeat
    j←j+1;
  until (j>n) or (v_j=false);
end;

IF (Twin=0) and (j>n) then C_outlier ←I
 {No F2 node can be selected, put the data set into the outlier cluster.}

IF (Twin=0) and (j<=n) then
begin
   v_twin Committed←True;
   n_twin ← n_twin+1;
   Exit the cluster procedure of this input data set, begin to cluster next input data set
end;

end;
```

Fig. 4. Selection of the winner.

```
procedure VigilanceandReset;
begin
{r,ρ: the similar degree.}

IF Twin>0 then r← ∑_i h_ij;
IF r<ρ then
  begin
    v_j reset←True;
    Goto "Selection of the winner";
  end;

end;
```

Fig. 5. Vigilance and reset.

The pseudocodes of the vigilance and reset mechanism are in Fig. 5.

3.2.4. Buffer management

We push input data sets, whose similar degree is less than avg_j, into the buffer. Meanwhile, we detect the fullness of the buffer. If the buffer is full of data, we clean the buffer data set whose similar degree is smallest and put this data set to the outlier cluster.

For the convenience of management, every buffer data set adds two elements, indicators for the candidate winner and similar degree, respectively.

The pseudocodes of the buffer management are in Figs. 6 and 7.

3.2.5. Learning

If the similar degree of a candidate winner is more than or equal to the average similar degree, the candidate winner becomes the winner.

The winner learning formulae obey from Eqs. (19) to (21). According to Eq. (19), for a committed winning F_2 node v_j, we have

$$z_{ij}^{\text{new}} = \begin{cases} L/(L-1+|X|) & \text{if } v_i \text{ is active to } v_j, \\ 0 & \text{otherwise,} \end{cases} \tag{26}$$

$$z_{ji}^{\text{new}} = (1-\alpha)z_{ji}^{\text{old}} + \alpha I_i, \tag{27}$$

where $|X|$ is the number of F_1 nodes which are active to the F_2 node v_j, and α is the learning rate, $0 \leqslant \alpha \leqslant 1$. And L is a positive constant.

For a noncommitted winner v_j and F_1 node v_i, we obtain

$$z_{ij}^{\text{new}} = L/(L-1+m), \tag{28}$$

$$z_{ji}^{\text{new}} = I_i. \tag{29}$$

```
procedure PushingIntoBuffer;
begin
 {avg_twin: the average similar degree of the cluster v_twin,
 round[avg_twin]: the round number of avg_twin,
 buffnum: the current number of buffer,
 buffmax: the maximum number of buffer,
 B: the array of data sets in the buffer.}

if r < round[avg_twin] then
  Begin
     BuffNum ← BuffNum + 1;
     B ← B + I;
     B_buffNum[m + 1] ←Twin;
     B_buffNum[m + 2] ←r;
     if BuffNum >= BuffMax then CleanBuffer;
  End;

end;
```

Fig. 6. Pushing into buffer.

```
procedure CleanBuffer;
begin
{clNo: the indicator of data sets to be cleaned,
 clgap: the biggest similar gap,
 abs(x): the absolute value of x.}

clNo←0; clgap←0;
 for I←1 to BuffNum do
 if abs(B[I,m+2] - avg_B[I,m+1]) > clgap then
    begin
       clgap←abs(B[I,m+2] - round(avg_B[I,m+1]));
       clNo←I;
    end;

 n_outlier ← n_outlier + 1;
 B_clNo ← B_buffNum;
 BuffNum ← BuffNum − 1;

end;
```

Fig. 7. Cleaning the buffer.

3.2.6. Computation of avg_j and the buffer checkout

At the end of BPART, we compute avg_j according to Eq. (18). We should keep avg_j stable at first. In other words, n_j initial value is more than 0, for example 50 in this paper.

When an input data set into a cluster v_j in F_2 layer, since avg_j is renewed, we must check whether the similar degree of any buffer data set is more than or equal to the new average degree, if yes, we pull this buffer data set to the cluster.

The pseudocodes of the buffer checkout are in Fig. 8.

4. Examples and simulations

4.1. Examples for PART disadvantages

In this section, we give several simple examples to illustrate that high similar noise data sets deteriorate the clustering result.

On one hand, in Figs. 9 and 10, the similar degree of the noise data set, *3 2 1 6 5 0*, is more than the vigilance parameter ρ. So according to PART, the noise data set has to be clustered to the projected cluster. The result reduces the valuable information, such as the projected dimensions. While Fig. 11 is the best clustering result. The noise data set is put into the outlier cluster, and the desired clustering result in Fig. 11 preserves the maximum valuable information, in which the desired result has the 5-dimension similar degree comparing to 2-dimension of the bad results in Figs. 9 and 10.

On the other hand, as Professor Jain says [10], "ART nets are order-dependent, that is, different partitions are obtained for different orders in which the data sets are presented to the net". Figs. 9 and 10 have the identical input data but small order difference. As a result, clustering results are different from each other.

4.2. Simulations in high dimensional spaces

For the purpose of illustration, we design a few sets of data, which form clusters only in subspaces. The high dimensional spaces are generated by the method introduced by Aggarwal [1]. To show advantages of BPART, we take a small modification to the method as follows.

The coordinates of clustered data sets and noise data sets (outliers) all span in the range [0, 100]. There are totally 5%

```
procedure CheckBuffer;
begin
I:=0;
While I<BuffNum do
  begin
   I:=I+1;
   if B[i,m+1]<>TWin then Continue;
   {B[i,m+1] denotes the candidate winner of I_i, Continue is miss the
rest of this loop, restart this loop.}

   if (d(I_i,z_ji) ≤ σ and l(z_ij) > θ) then h_ij←1
     else h_ij←0;

   r← Σ_i h_ij;

   if r>=Round(avg_twin) then
     begin
      update z_ij and z_ji
      n_j ← n_j + 1;
      B_i ← B_BuffNum
      BuffNum ← BuffNum − 1;
      I ← I − 1;
     end;
  end;

end;
```

Fig. 8. Check the buffer.

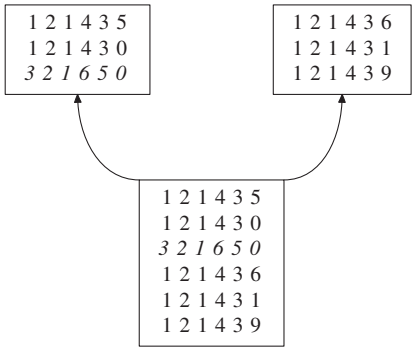

Fig. 9. The high similar noise data, *3 2 1 6 5 0*, disturb the clustering result, where $\rho = 2$, $\sigma = 0.1$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.1$, $\theta = 0$.

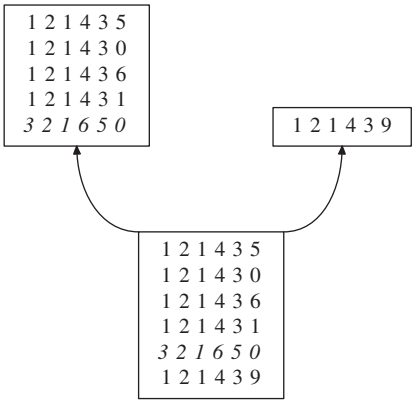

Fig. 10. The high similar noise data, *3 2 1 6 5 0*, deteriorate the cluster, where $\rho = 2$, $\sigma = 0.1$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.1$, $\theta = 0$.

noise data sets and are distributed uniformly random throughout the entire space. The centers of all clusters are obtained by using k uniformly distributed data sets in a d-dimensional space. The number of dimensions associated with a cluster is generated by the Poisson distribution. Every two clusters have some overlap dimensions. The data sets for the cluster v_j are generated as follows: the coordinates of data sets on the nonprojected dimensions are random, while the coordinates of one projected dimension j obey the normal distribution with the mean at range of the respective $\pm 5\%$ of the anchor data set.

The modification to this method is that we put some noise data sets into input files. Those noise data sets have a characteristic: they share partial dimensions (more than or equal to the degree of ρ) of one cluster, but they are completely different at the other dimensions of the cluster.

We design the experiment file which has 10,000 data sets and five clusters in a 20-dimensional space and all five clusters have the seven projected dimensions in different subspaces. Table 1 shows the projected dimensions and the number of data sets of each cluster.

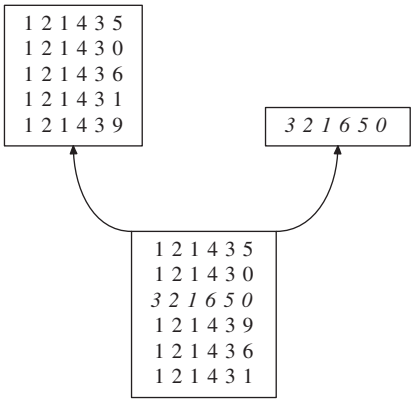

Fig. 11. The best clustering result with high similar noise data, *3 2 1 6 5 0*, deteriorate the cluster, where $\rho = 2$, $\sigma = 0.1$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.1$.

Table 1
Dimensions and numbers of data sets of INPUT clusters in a 20-dimensional space

Cluster no.	Projected dimensions	Number
1	3,4,7,9,14,16,17	2139
2	3,4,7,12,13,14,17	2328
3	4,6,11,13,14,17,19	1824
4	4,7,9,13,14,16,17	1573
5	3,4,9,12,14,16,17	1636
Outliers	–	500

Table 2
Dimensions and numbers of data sets of OUTPUT clusters for the input file by PART, where $\rho = 5$, $\sigma = 0.17$

Found cluster no.	Projected dimensions	Number
1	3,4,7,9,14,16,17	2139
2	3,4,7,12,13,14,17	2327
3	4,6,11,13,17	1825
4	4,7,9,13,14,16,17	1573
5	4,9,14,16,17	1637
Outliers	–	498

Table 3
Dimensions and numbers of data sets of OUTPUT clusters for the input file by BPART, where $\rho = 5$, $\sigma = 0.17$

Found cluster no.	Projected dimensions	Number
1	3,4,7,9,14,16,17	2139
2	3,4,7,12,13,14,17	2327
3	4,6,11,13,14,17,19	1824
4	4,7,9,13,14,16,17	1573
5	3,4,9,12,14,16,17	1636
Outliers	–	501

Table 4
Dimensions and numbers of data sets of OUTPUT clusters for the input file by PART, where $\rho = 4$, $\sigma = 0.17$

Found cluster no.	Projected dimensions	Number
1	3,9,14,17	2140
2	4,12,14,17	2328
3	4,11,13,17	1827
4	4,9,13,17	1574
5	4,9,14,16,17	1637
Outliers	–	493

Table 5
Dimensions and numbers of data sets of OUTPUT clusters for the input file by BPART, where $\rho = 4$, $\sigma = 0.17$

Found cluster no.	Projected dimensions	Number
1	3,4,7,9,14,16,17	2139
2	3,4,7,12,13,14,17	2327
3	4,6,11,13,14,17,19	1824
4	4,7,9,13,14,16,17	1573
5	3,4,9,12,14,16,17	1636
Outliers	–	501

Table 6
Dimensions and numbers of data sets of OUTPUT clusters for the input file by PART, where $\rho = 3$, $\sigma = 0.17$

Found cluster no.	Projected dimensions	Number
1	7,9,17	2142
2	3,7,12	2323
3	4,13,17	1831
4	7,9,13	1574
5	9,16,17	1644
Outliers	–	475

Table 7
Dimensions and numbers of data sets of OUTPUT clusters for the input file by BPART, where $\rho = 3$, $\sigma = 0.17$

Found cluster no.	Projected dimensions	Number
1	3,4,7,9,14,16,17	2139
2	3,4,7,12,13,14,17	2327
3	4,6,11,13,14,17,19	1824
4	4,7,9,13,14,16,17	1573
5	3,4,9,12,14,16,17	1636
Outliers	–	501

We report the experiment result of both BPART and PART in each file. In this experiment, BPART emphasizes independence of the vigilance parameter ρ. BPART algorithm obtains the full projected dimensions by the different parameters ρ. Tables 2 and 3 show the two simulation results with $\rho = 5$, $\sigma = 0.17$; and Tables 4 and 5 show the two simulation results with $\rho = 4$, $\sigma = 0.17$; and Tables 6 and 7 show the two simulation results with $\rho = 4$, $\sigma = 0.17$. (In this paper we only focus on ρ, so we choose a small σ.)

From the results and comparisons, we could understand the advantage of BPART, that found dimensions are independent on the accurate choice of ρ. In Tables 2, 4 and 6, with the decreasing ρ, the found dimensions are also decreasing. While in Tables 3, 5 and 7, BPART can still find full projected dimensions. We call the advantage ρ-toleration.

Due to the ρ-toleration, users do not have to choose a precise ρ. So they could choose a rough parameter to obtain a close result. In fact, choosing an accurate ρ is a hard job for an user (even an experienced user).

4.3. Scalability results

Due to the buffer management mechanism of BPART, readers may ask whether the run time and extra space is sustainable. So in the following we will discuss the dependency on run time and extra buffer space with different percentages of outliers, different total number of input data sets and various capacities of the buffer space.

4.3.1. Dependency of run time

Firstly, we design a series of input data files with 20,000 data sets in a 20-dimension space. They are clustered into five clusters.

Simulations are performed on a 677-MHz Intel Celeron CPU with 256M of memory and Windows XP sp2.

As noticed in previous sections, the output of PART is seriously influenced by the choice of ρ and high similar noise data sets. Hence, we compare the run times of PART and BPART only in input files which eliminate such noise data sets, and the ρ is set to 5. Because of random outliers, each run time reported in this section is averaged over five times.

Fig. 12 shows the time dependency of simulation results of BPART and PART on different percentages of random outliers. From this figure, we could conclude three points:

1. The run times of both BPART and PART are increasing with the growth of percentages of random outliers.
2. Although the run time of BPART is more than that of PART at each column, the difference is small. The biggest difference is 23.5% (at 25% percentage), and the average of difference of run time is 14.2%.
3. The rate of the run time to the outlier percentage is gradually diminishing.

The other simulation about the dependency of run time on the number of input data sets is in Fig. 13. In these input files, not only the total number of data sets but also the percentages of random outliers are increasing. The total number is from 10,000 to 47,500, while the percentage of outliers is from 5% to 80%. The run times of BPART and PART at each number and percentage are also consistent with our conclusion.

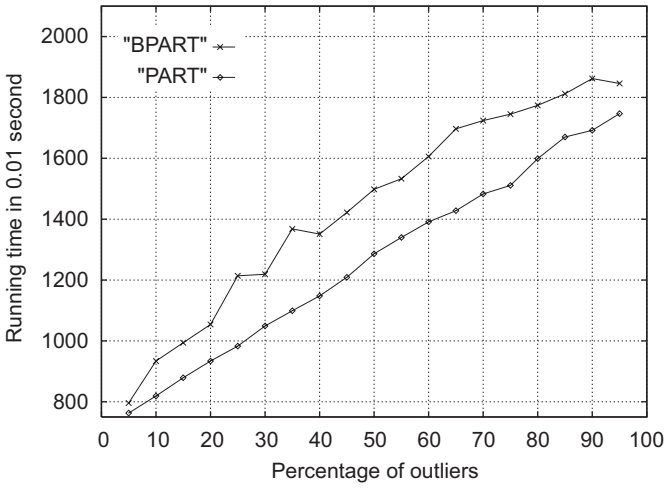

Fig. 12. Scalability with different percentages of outliers, where $\rho = 5$, $\sigma = 0.17$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.1$, $\theta = 0$.

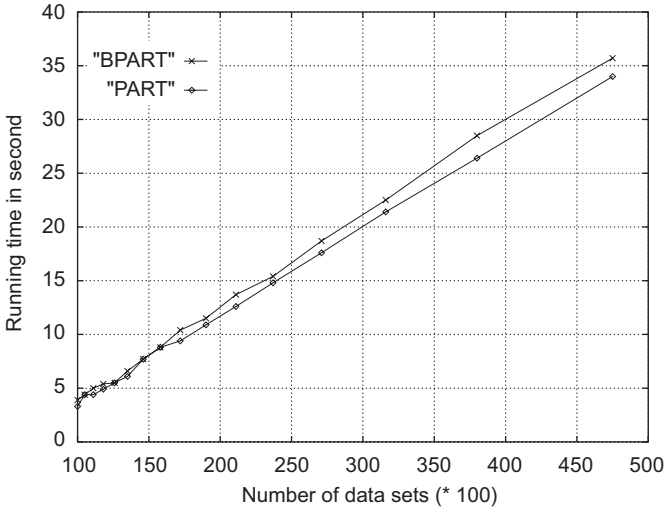

Fig. 13. Scalability with different total number of data sets, where $\rho = 5$, $\sigma = 0.17$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.1$, $\theta = 0$.

From Figs. 12 and 13, introducing the buffer management does not bring the great change of run time of BPART with a big number of data sets or different percentages of outliers. The extra run time of BPART is checking whether the similar degree of any buffer data set is more than or equal to the new average degree. However, the avg_j greatly reduces the chance that high similar noise data sets are grouped to a good cluster. With increasing number of input data sets grouped to clusters, the avg_j becomes a stable value.

4.3.2. Dependency of buffer spaces

In this section, through the comparison with different buffer spaces, we will find that the small capacity of the buffer can handle a big amount of input data sets.

Table 8
Scalability with capacity of buffer spaces on the first input file (20% outliers), where $\sigma = 0.17$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.17$, $\theta = 0$

ρ	2.5%	1%	5‰	2.5‰	1‰	PART
8	0	0	0	0	3	8
7	0	0	0	0	4	28
6	0	0	0	3	23	346
5	0	0	0	11	31	1487

Table 9
Scalability with different capacity of buffer spaces on the second input file (70% outliers), where $\sigma = 0.17$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.17$, $\theta = 0$

ρ	2.5%	1%	5‰	2.5‰	1‰	PART
8	0	0	0	0	7	63
7	0	0	0	0	11	254
6	0	0	0	14	35	1874
5	0	0	0	31	58	4481

We design two input files with 20,000 data sets in five clusters and a outlier cluster, each of which is projected between 10-dimension and 40-dimension subspace in a 100-dimension space. One input file contains 20% of outliers, and the other contains 70% of outliers. Tables 8 and 9 are the simulation results. A necessary explanation of this simulation is that we demonstrate different results with the decreasing capacity of the buffer and ρ. In Tables 8 and 9, we compare the number of data sets clustered to wrong clusters with different buffers to demonstrate BPART's benefits over PART. (The buffer space is 2.5% of the total number of data sets, 1%, 5‰, 2.5‰ and 1‰, respectively.)

From Tables 8 and 9, we could obtain a conclusion that the decreasing capacity of the buffer space has a small effect on the clustering result. In other words, a small buffer space can process a huge amount of input data sets. In fact, the good behavior of BPART is consistent with those algorithms reported in Section 2, and the main function of the buffer space is storing the high similar noise data sets. Although high similar data sets are very few, they can bring a jeopardous result if we cannot efficiently discard them.

5. An application

In decades, the problem of discovering association rules between items in a large database is a popular research field. Association rule mining finds interesting associations or correlation relationships among large set of data items.

In this section, we apply BPART to cluster stock concurrence association rules, which means that in most cases these stocks surprisingly rise or fall at same time. For example, the rise of stocks in the petroleum sector always brings about the fall of stocks of tourism and airline corporations. This concurrence association rule is good to predict the trend of certain stocks with the aid of other stocks. Thus, we are going to use data mining algorithm to discover stock concurrence association rules behind a wealth of stock transaction databases in order to predict the stock market change.

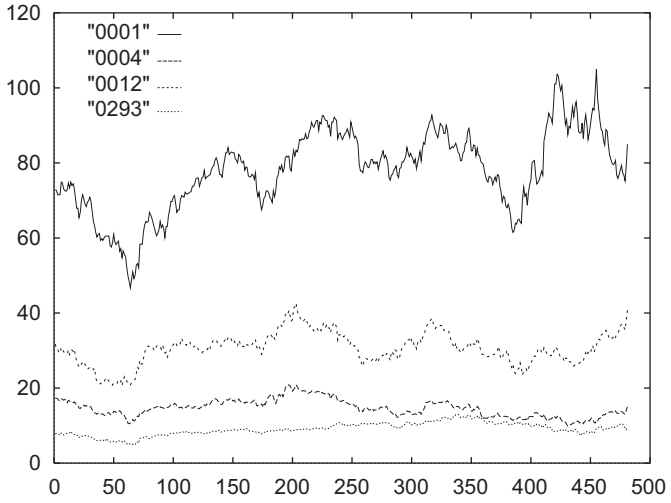

Fig. 14. The concurrence of rising–falling of four stocks in HSI.

In our study, we experiment with BPART in the database of The Hang Seng Composite Index Series (HSCI), launched on 3 October 2001. We select randomly 100 constituent stocks contributing to the HSCI Index. There are a total 481 transaction days and each transaction day contains all 100 constituent stocks. And we initialize parameters as follows: $\rho = 2$, $\sigma = 0.2$, $L = 2$, $\varepsilon = 1$, $\alpha = 0.1$ and $\theta = 0$.

After this experiment, we find that there are four out of 100 stocks which have concurrence associations. In detail, 00001 Cheung Kong, 00004 Wharf (Hldgs), 00012 Henderson Land and 00293 Cathay Pac Air are related (or concurrence) in 90 days out of 481 transaction days, and partly related in 105 days out of 481. "Partly related" means that two or three stocks are related in one transaction day. The result of stock concurrence associations is shown in Fig. 14. The ratio of concurrence of rise–fall is 195 $(90 + 105)$ to 481, or 0.4. Specially, we regard the unchangedness of stocks on one transaction day as rise.

When we use our result to predict "future" trends, surprising success is emerged: the four stocks are concurrence in 103 days of 243 transaction days dated from January 02, 2002 to December 31, 2002 and the ratio of concurrence is approximately 0.5 (103/243)!

Compared with PART, our algorithm initializes the important parameter ρ to 2, which is easily estimated and applied. And from this result above, any value more than 4

fails to find the concurrence of four stocks. Therefore, our algorithm over PART can obtain the good result without an accurate parameter ρ.

6. Conclusions

In this paper, we have presented an improved architecture, PART with Buffer, of the neural network for data mining, which can successfully deal with high similar noise data sets and has the *ρ-tolerance* attribute. At the same time, the time and space cost of BPART is very limited.

Our improvement is the introduction of a buffer layer and a new similar degree function, avg_j. Both the buffer and avg_j can keep away from the fatal clustering result impairment in the case of high similar noise data sets. And we could obtain close results including projected dimensions even if we choose an imprecise parameter ρ value. Besides the high-similar-noise-data-tolerance and ρ-tolerance, we could find that the run time of BPART is also as much as that of PART, while a small buffer space can satisfy the requirement of correct clustering. Therefore, because of above good features, PART with buffers is a promising methodology for the high dimensional space clustering.

Acknowledgments

We are very grateful to all anonymous referees and Editor in Chief and Associated Editor for their valuable comments and communications about the previous version. Their comments not only help us achieve a clear paper, but also guide our future work, especially in technological literature writing.

References

[1] C.C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, J.S. Park, Fast algorithms for projected clustering, in: Proceedings of SIGMOD'99, 1999, pp. 61–72.
[2] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbors" meaningful, in: Proceedings of the 7th International Conference on Database Theory (ICDT'99), 1999, pp. 217–235.
[3] Y. Cao, J. Wu, Projective ART for clustering data sets in high dimensional spaces, Neural Networks 15 (2002) 105–120.
[4] Y. Cao, J. Wu, Dynamics of projective adaptive resonance theory model: the foundation of PART algorithm, IEEE Trans. Neural Networks 15 (2004) 245–260.
[5] G.A. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Comput. Vision Graphics Image Process. 37 (1987) 54–115.
[6] G.A. Carpenter, S. Grossberg, ART2: self-organization of stable category recognition codes for analog input patterns, Appl. Opt. 26 (1987) 4919–4930.
[7] G.A. Carpenter, S. Grossberg, ART3: hierarchical search using chemical transmitters in self-organizing pattern recognition architecture, Neural Networks 4 (1990) 129–152.
[8] G.A. Carpenter, S. Grossberg, D.B. Rosen, ART2-A: an adaptive resonance algorithm for rapid category learning and recognition, Neural Networks 4 (1991) 493–504.
[9] D.L. Donoho, High dimensional data analysis: the curses and blessings of dimensionality, in: Presented at American Mathematics Society Conference: Math challenges of the 21st Century, Los Angeles, USA, 2000.
[10] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (1999) 264–323.
[11] T. Kohonen, Clustering, taxonomy, and topological maps of patterns, in: Proceedings of the 6ICPR of International Conference on Pattern Recognition, 1982, pp. 114–128.
[12] B. Li, S. Xu, J. Zhang, Enhancing clustering blog documents by utilizing author/reader comments, in: Proceedings of the 45th Annual Southeast Regional Conference, 2007, pp. 94–99.
[13] W. Lin, A. Hauptmann, Structuring continuous video recordings of everyday life using time-constrained clustering, in: IS&T/SPIE Symposium on Electronic Imaging (EI'06), 2006, pp. 6073–6085.
[14] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, ACM SIGKDD Explorations Newsl. 6 (2004) 90–105.
[15] H. Takahashi, H. Honda, Modified signal-to-noise: a new simple and practical gene filtering approach based on the concept of projective adaptive resonance theory (PART) filtering method, Bioinformatics 22 (2006) 1662–1664.
[16] H. Takahashi, T. Nemoto, T. Yoshida, H. Honda, T. Hasegawa, Cancer diagnosis marker extraction for soft tissue sarcomas based on gene expression profiling data by using projective adaptive resonance theory (PART) filtering method, BMC Bioinformatics 7 (2006) 399–410.
[17] H. Wang, D. Huang, A novel clustering analysis based on PCA and SOMs for gene expression patterns, in: Lecture Notes in Computer Science, vol. 3174, Springer, Berlin, 2004, pp. 476–481.
[18] S. Xu, J. Zhang, A hybrid parallel Web document clustering algorithm and its performance study, J. Supercomput. 30 (2004) 117–131.
[19] Z. Zhao, D. Huang, L. Guo, A novel clustering-neural tree for pattern classification, in: The 2004 International Joint Conference on Neural Networks (IJCNN2004), 2004, pp. 1303–1308.

Lian Liu received his B.S. degree in Information and Computational Science, M.S. degree in Applied Mathematics from Hunan University, China, in 2003 and 2006, respectively. He is currently a doctoral student at the University of Kentucky, USA. His research interests are information retrieval, privacy preserving data mining, and neural networks.

Lihong Huang was born in Hunan, China, in 1963. He received the B.S. degree in Mathematics in 1984 from Hunan Normal University, Changsha, China, and the M.S. degree and the Ph.D. degree in Applied Mathematics from Hunan University, Changsha, China, in 1988 and 1994, respectively.

From July 1988 to June 2000 he was with the Department of Applied Mathematics at Hunan University, where he was an associate professor of applied mathematics from July 1994 to May 1997. In June 1997 he became a professor and doctoral advisor of applied mathematics and Chair of the Department of Applied Mathematics. Since July 2000 he has been Dean of the College of Mathematics and Econometrics at Hunan University, Changsha, China.

He is the author or co-author of more than 200 journal papers, eight edited books. His research interests are in the areas of dynamics of neural networks, and qualitative theory of differential equations and difference equations.

Mingyong Lai was born in Jiangxi, China, in 1965. He received his Bachelor of Science degree in mathematics in 1986 from Hunan University, Changsha, China, and his Master's degree and Ph.D. in International business from Hunan University, Changsha, China, in 1990 and 1997, respectively. From July 1986 to June 1988 he was with the Department of Applied Mathematics at Hunan University, where he was an associate professor of applied mathematics. In June 1998 he became a professor and doctoral advisor of International Business and Chair of the Department of school of international business. Since July 2002 he has been Dean of the College of Economics and Business at Hunan University, Changsha, China. He is the author or co-author of more than 120 journal papers and five edited books. His research interests are in the areas of international business and logistic management quantitative economic analysis and applied economic modeling.

Chaoqun Ma is Professor and the Vice Dean of the College of Business Administration at Hunan University, China. His main research areas are data ming, financial enginnering and risk management.