The classified method for overlapping data

Boontee Kruatrachue*, Kulwarun Warunsin*, and Kritawan Siriboon*

*Computer Engineering Department, Faculty of Engineering,

King Mongkut's Institute of Technology Ladkrabang,

Chalongkrung Road, Bangkok 10520, Thailand

(Tel : +66-1-840-0405; E-mail: booontee@yahoo.com, kulwarun@hotmail.com, kritawan@ce.kmitl.ac.th)

Abstract: In this paper we introduce a new prototype based classifiers for overlapping data, where training pattern can be overlap on the feature space. The proposed classifier is based on the prototype from neural network classifier (NNC)[1] for overlap data. The method automatically chooses the initial center and two radiuses for each class. The center is used as a mean representative of training data for each class. The unclassified pattern is classified by measure distance from the class center. If the distance is in the lower (shorter radius) the unknown pattern has the high percentage of being in this class. If the distance is between the lower and upper (further radius), the pattern has the probability of being in this class or others. But if the distance is outside the upper, the pattern is not in this class. We borrow the words upper and lower from the rough set to represent the region of certainty [3]. The training algorithm to find number of cluster and their parameters (center, lower, upper) is presented. The clustering result is tested using patterns from Thai handwritten letter and the clustering result is very similar to human eyes clustering.

Keywords: overlapping data, classifier , supervised learning, Neural prototype

1. INTRODUCTION

A class is a collection of data object that are similar to one another within the same class and are dissimilar to the objects in other class. The process of grouping a set of physical or abstract objects into classes of similar objects is called classification. Classification is very close to clustering except that the classification is a supervised learning where classifier is trained with known class. Classification has wide applications including market or customer segmentation, pattern recognition, biological studies, spatial data analysis. Web document classification, and many others. Classification can be used as a standalone data mining tool to gain insight into the data distribution, or serve as a preprocessing step for other data mining algorithms operating on the detected clusters. Classification is a dynamic field of research in data mining. Many clustering algorithms have been develops. These can be categorized into hierarchical methods, partitioning methods, density-based methods, grid-based methods, and model-based methods [2].

The rest of the paper is organized as follows. In section 2 describe the NNC architecture. Section 3 we present our new prototype based classifiers for overlap data. Section 4, we report the experimental tested using patterns from Thai handwritten letter and the clustering result compare with human eyes clustering. Section 5 concludes with a summary and some directions for future research.

2. NNC ARCHITECTURE [1]

The neural network that implements the NNC is shown in Fig. 1. The input layer $L_A=(a_1, a_2, a_3,..., a_n)$ consists of n neurons, one for each dimension of the input pattern P. Each input neuron is connected to all the N prototype neurons in layer $L_B=(b_1, b_2, b_3,..., b_n)$. The connection weights from neurons in layer L_A to neurons in layer L_B are represented by:

$$W_1 = (w_1, w_2, ..., w_N)^T \tag{1}$$

Where $W_i = (w_{b_i a_1}, w_{b_i a_2}, ..., w_{b_i a_N})$ is the weight vector of the ith neuron in layer L_B.

A neuron in layer L_B representing only one pattern class. Each prototype neuron is added during learning, where there can be more than one neurons for a single class.

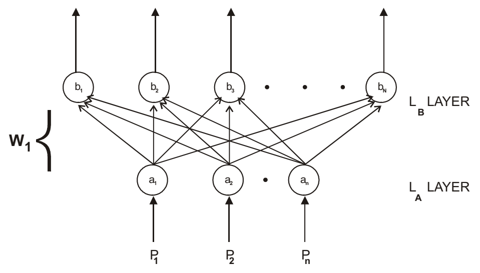

Fig. 1 NNC architecture.

In response to a pattern P_k the output of the NNC can be classified into one of the following class:

$$c_1, c_2, ...c_k, c_m \tag{2}$$

Where

c_k is the class of neuron in layer L_B of the NNC corresponding to class k,

m is the total number of classes in the training pattern.

If an input pattern P causes neuron b_k to fire, and b_k belong to class c_k. Then P is classified as class c_k:

In the context of this work, training the NNC means ensuring that its can correctly classify any training input pattern $P^k = \{p^k_1, p^k_2, ..., p^k_n\}$. This can be achieved by doing the following:

- creating new prototype neuron in layer L_B for any unclassified P,

- adjusting the connection weights of the nearest neurons according to the Euclidean distance in equation (3), if the distance < that neuron threshold and the class of neuron is the same as the class of pattern P, Euclidean distance from P to neuron b_j is

$$b_{jT} = \sqrt{\sum_{s=1}^{n} (w_{b_j a_s} - p_s)^2} \qquad (3)$$

$w_{b_j a_s}$ is the connection weight between neuron b_j in L_B and neuron a_s in L_A

The weights are adjust to average the new pattern P into the nearest neurons as follow:

$$W_{b_j a_{i_{new}}} = \frac{W_{b_j a_{i_{old}}} + p_i}{M+1} \quad \forall i = 1,2,\ldots,n \qquad (4)$$

M is a popularity measure corresponding to the number of patterns

- For any neurons that has different class from pattern P and the distance between P and those neurons < that neurons threshold, reduce the firing threshold conditions of those neurons in order to exclude the pattern P from them.

A flowchart for training NNC is described in Fig. 2.

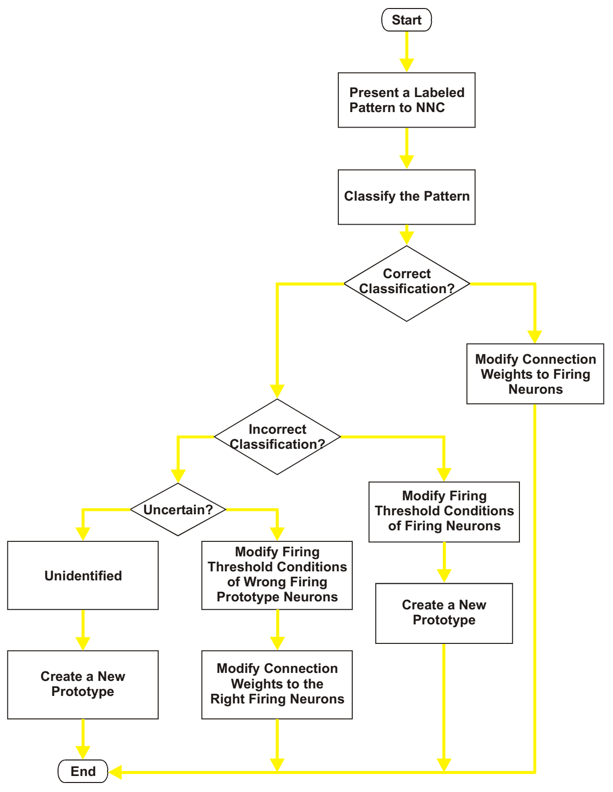
Start
Present a Labeled Pattern to NNC
Classify the Pattern
Correct Classification?
Modify Connection Weights to Firing Neurons
Incorrect Classification?
Uncertain?
Modify Firing Threshold Conditions of Firing Neurons
Unidentified
Modify Firing Threshold Conditions of Wrong Firing Prototype Neurons
Create a New Prototype
Create a New Prototype
Modify Connection Weights to the Right Firing Neurons
End

Fig. 2 NNC training algorithm.

3. PROTOTYPE BASED CLASSIFIERS FOR OVERLAP DATA

The NNC classifier described in previous section will keep reducing threshold and create new prototype neuron if the training Patterns has overlap as shown in Fig 3. These training loops start by the threshold reduction of wrong classified pattern. Then, any the pattern that used to be classified by that neuron become unclassified and finally new neuron is created for the wrong classified pattern. The final network will have large number of neuron, one for each training pattern in the highly overlap area.

In order to reduce number of neuron in the overlap area, each neuron has two thresholds (radius), shorter and the longer one as shown in Fig 3. The shorter radius is to define the perimeter of the neuron in the same way as the threshold in the NNC. The longer one is to define the conditional perimeter where any pattern located in this area can be the same or different class as this neuron. Hence the pattern in the shorter radius can be classified but the pattern between shorter and longer radius is uncertain but has some probability to be in the neuron class. But the pattern outside the longer radius is certainly not in the same cluster (neuron). These shorter and longer radiuses are analogous to the idea of lower and upper in the rough set [3].

From Fig. 3 A1 and A2 is area of class 1 and 2. B1 is area of probability data of class 1 and other, B2 is area of probability data of class 2 and other, C is area of data class 1 and class 2.

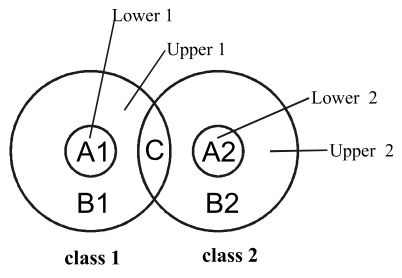

Fig. 3 area of lower and upper in class 1 and class 2.

The algorithm start by initial lower equal to upper radius and first data is the center of class (neuron). Then each training pattern is used to adjust the neuron parameter (center, lower, upper). The adjustment split into two cases depends on whether the pattern and the neuron have the different class (Fig. 4), or same class (Fig. 5). The distance used here is the same Euclidean distance as NNC.

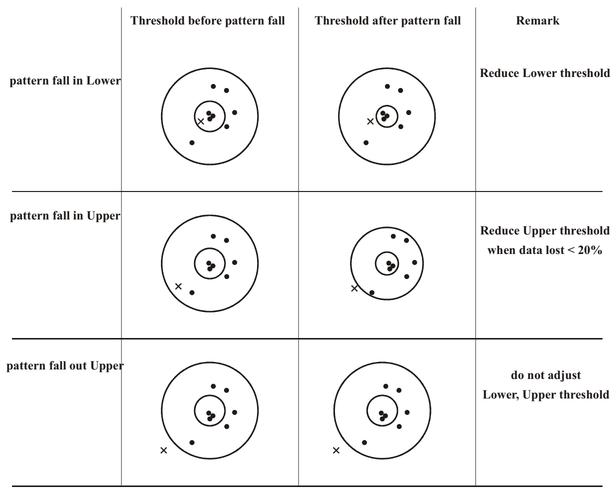
Threshold before pattern fall
Threshold after pattern fall
Remark
pattern fall in Lower
Reduce Lower threshold
pattern fall in Upper
Reduce Upper threshold when data lost < 20%
pattern fall out Upper
do not adjust Lower, Upper threshold

Fig. 4 adjustment of radius when training data is different class as the train cluster.

For each training pattern, the distance is calculated for each neuron. For all neurons that have different class from the pattern, the radius adjustments depend on location of the pattern as shown in fig 4. If the pattern is in lower (shorter radius) then reduce the lower to exclude that pattern out of the lower region. If the pattern is between upper and lower, we don't need to change any radius. But in order to maintain high class probability of the upper region, the upper is reduce to exclude the wrong pattern if at least 80 % of the pattern that used to be in the upper still remain. Certainly, if the pattern is outside the upper, no change to the radius. The weight of the neuron is unchanged from the pattern of different class.

For the neuron that has the same class with the pattern, only the closet one is consider, others neurons are unchanged. The radius adjustment is as shown in fig 5. If the pattern is in the lower area no change to the radius since the correct pattern is already in the neuron lower area. If the pattern is between lower and upper, still the lower and upper is unchanged. Here the lower is not expanding to include the pattern because there may be other class pattern that will be in the lower. If the pattern is outside the upper area of the closet neuron with the same class, the upper is expanded to include that pattern if the distance is < 1.2 upper. Otherwise new neuron is constructed at that training pattern with the lower equal to the upper. And the upper extend to the center of the closet neuron.

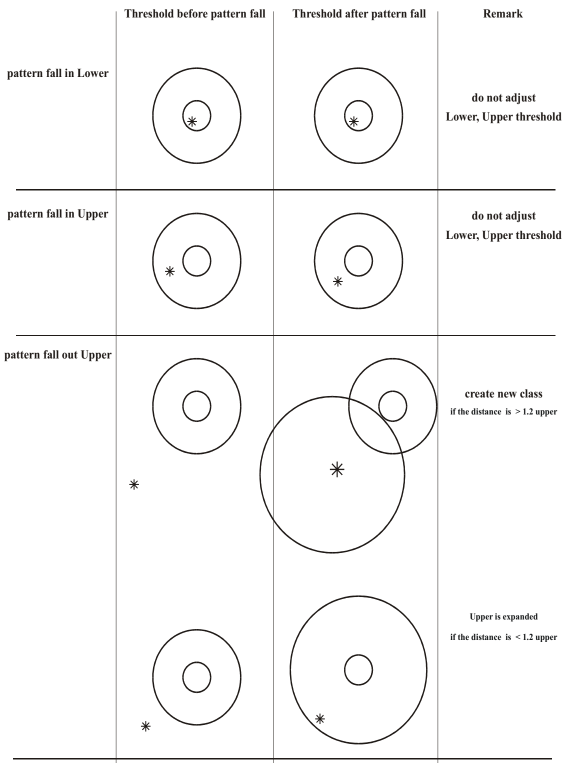

Fig. 5 adjustment of radius when training data is the same class as the train cluster.

As shown in Fig. 6, the distance between a training pattern and each neuron is calculated in order to reduce the lower or upper radius of each neuron with different class to the pattern if the wrong pattern is in the lower or upper region. For the neuron with the same pattern only the closet neuron weight is changed by average the new pattern with its weight. But if the pattern is not in any lower or upper region of any neuron with the same class, the new neuron node is created at the pattern. The training is repeated for each training pattern until all patterns are tested. If any neuron has radius change or new neurons are created, the training process is repeated until no change in the network.

```
CONSTANT SMALL  is value for adjust radius.

While ( (radius reduce) || (new neuron) ) {
   for each pattern p in the training patterns{
      read pattern p;
      for(node=0; node<total neuron; node++){
        o    compute distance = Euclidean distance (weight[node], p);
        o    if (class of p == class [node]){
                  if(distance < lower[node])
                         mark fall flag for adjust weight;
                  else if (distance < upper[node]){
                         mark fall flag for adjust weight;
                          if (distance < (1.2 * upper[node]))
                             upper[node] = distance – SMALL;
                                                          // expand upper
                          else
                             clear fall flag for create new node
                  }
             }// if (class of p == class [node])
        o    if (class of p != class [node]){
                  if(distance < lower[node])
                         lower[node] = distance - SMALL;
                                                          //reduce lower
                  else if (distance < upper[node]){
                         if (reduce upper[node] at least 80 % of the pattern)
                              upper[node] = distance – SMALL;
                  }
             }// if (class of p != class [node])
      }// for node
      if (pattern not fall in any node with the same class)
             create new node (p);
      else
             average pattern and weight
             in the nearest node with the same class;

   }// for each pattern p
}// While ( (radius reduce) || (new neuron) )
```

Fig. 6 Algorithm for overlap classifier.

4. EXAMPLE OF EXPERIMENT

We test new algorithm with clustering of 198 patterns from 6 letter THAI handwritten letter. Each pattern has only 2 dimension features in order to create some overlap as shown in Fig. 7. The clustering result of data into 6 clusters as shown in Fig. 9. It is hard to compare the classification results for overlap data since the classification accuracy depend on how much data overlap, especially for large number of class and patterns. Hence, in this paper we intend to use small number of classes (6 letter) and only 198 patterns in order to compare the clustering results with human approximate cluster.

As expected, the NNC create higher number of neurons in the overlap area. On the contrary overlap classifier create less number of neuron but with uncertainty ring region (between upper and lower).

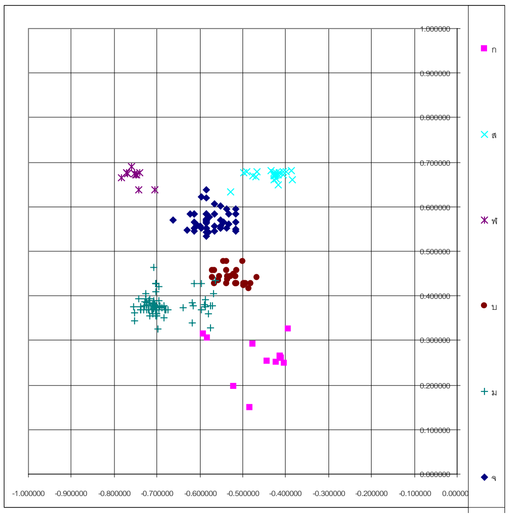

Fig. 7 Sample two feature THAI alphabet 198 data.

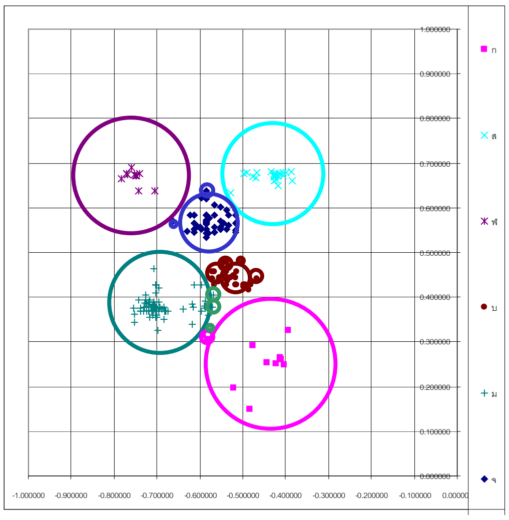

Fig. 8 Result is the NNC classifier described in section 2.

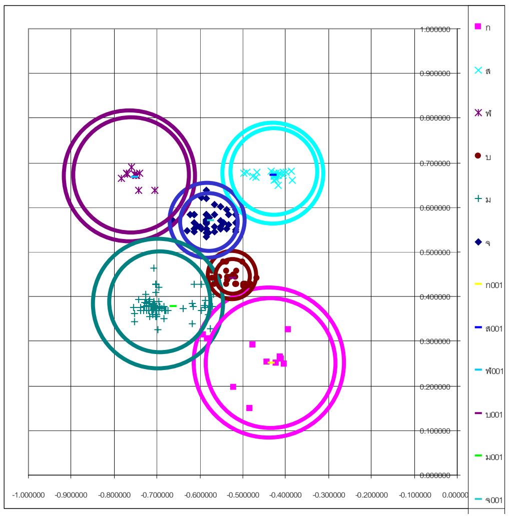

Fig. 9 Result of prototype based classifiers
for overlap data.

5. CONCLUSIONS

In this paper, we propose a prototype based classifiers and algorithm for overlap data. Our algorithm is an extension of neural classifier, NNC [1] where the concept of class boundary is extend to have lower and upper boundary instead of just one radius to represent class boundary. This clustering has the same fast training time as neural classifier [1] which is much faster than normal gradient descent based back propagation neural classifier. Example has been provided that demonstrates the ability of overlap classifier in continuous classification problems with overlapping data. The technique used to set and modify the threshold conditions of its prototype is described in this paper. Future work improves algorithm for efficient and effective cluster analysis in large databases. Active themes for research focus on scalability of clustering methods, the effectiveness of methods for clustering high-dimensional techniques.

REFERENCES

[1] M.A. ABOU-NASR and M.A. SID-AHMED, Fast learning and efficient memory utilization with a prototype based neural classifier., Pattern Recognition, Vol.28, No.4, pp. 581-593, 1995.

[2] Jiawei Han and Micheline Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann, pp. 335-393, 2001.

[3] Sarah Coppock, Lawrence Mazlack, "Rough Sets Used In The Measurement Of Similarity Of Mixed Mode Data," 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS 2003) Proceedings, Chicago, pp. 197-201, July, 2003.