Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Improved competitive learning neural networks for network intrusion and fraud detection

John Zhong Lei [a], Ali A. Ghorbani [b,*]

[a] Vesta Corporation, Portland, OR 97229, USA
[b] Faculty of Computer Science, University of New brunswick, Fredericton, NB, Canada E3B 5A3

ARTICLE INFO

Available online 22 August 2011

Keywords:
Competitive learning
Fraud detection
Intrusion detection
Supervised/unsupervised clustering
Neural network

ABSTRACT

In this research, we propose two new clustering algorithms, the improved competitive learning network (ICLN) and the supervised improved competitive learning network (SICLN), for fraud detection and network intrusion detection. The ICLN is an unsupervised clustering algorithm, which applies new rules to the standard competitive learning neural network (SCLN). The network neurons in the ICLN are trained to represent the center of the data by a new reward-punishment update rule. This new update rule overcomes the instability of the SCLN. The SICLN is a supervised version of the ICLN. In the SICLN, the new supervised update rule uses the data labels to guide the training process to achieve a better clustering result. The SICLN can be applied to both labeled and unlabeled data and is highly tolerant to missing or delay labels. Furthermore, the SICLN is capable to reconstruct itself, thus is completely independent from the initial number of clusters.

To assess the proposed algorithms, we have performed experimental comparisons on both research data and real-world data in fraud detection and network intrusion detection. The results demonstrate that both the ICLN and the SICLN achieve high performance, and the SICLN outperforms traditional unsupervised clustering algorithms.

1. Introduction

Fraud detections and network intrusion detections are extremely critical to e-Commerce business. According to U.S. census bureau retail e-Commerce sales reports, e-Commerce in North America has continued to grow 20% or more each year. However, fraud costs e-Commerce companies in U.S. and Canada an overwhelming lost each year. With the recent growth of the e-Commerce, credit card fraud has become more prevalent. Based on the survey results in 2009, on average, 1.6% of orders proved to be fraudulent, which is about $3.3 billions. In addition to the direct losses made through fraudulent sales, fraud victims' trust in both the credit card and the retail company decreases, which further increases the loss. It is the intent of the companies and the credit card issuers to detect or prevent fraud as soon as possible. Network intrusions, on the other hand, attack e-Commerce companies from their back. Any down time of Web servers or leaks of business or customer information may cost huge loss.

Both the credit card fraud-detection and network intrusion detection domains present the following challenges to data mining:

- There are millions of transactions each day. Mining such massive amount of data requires highly efficient techniques.
- The data are highly skewed. There are many more good events than bad events. Typical accuracy-based mining techniques may generate highly accurate detectors by simply predicting all transactions legitimate but these detectors cannot detect fraud at all.
- Data labels are not immediately available. Frauds or intrusions usually are aware after they have already happened.
- It is hard to track users' behaviors. All types of users (good users, business, and fraudsters) change their behaviors quite often. Finding new or changing patterns is as important as recognizing old patterns.

In this research we propose two clustering algorithms for fraud detection and network intrusion detection: the improved competitive learning network (ICLN) [20] and the supervised improved competitive learning network (SICLN). The ICLN is an unsupervised clustering algorithm developed from the standard competitive learning network (SCLN) [15]. The SICLN is a supervised clustering algorithm derived from the ICLN. Our goal is to develop advance machine learning techniques to solve the practical challenges in network intrusion detections and fraud detections.

Fig. 1 is an example of a fraudulent event. If credit card information of a card holder is stolen and uses for online shopping, it will take a few days for this transaction to appear on the credit card statement, and take a few more days or a few months for the real card holder to know and report to the bank. It will take a few other days for the bank to sent a notice to the retail company. Usually

* Corresponding author.
E-mail addresses: john.lei@trustvesta.com (J.Z. Lei), ghorbani@unb.ca (A.A. Ghorbani).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.02.021

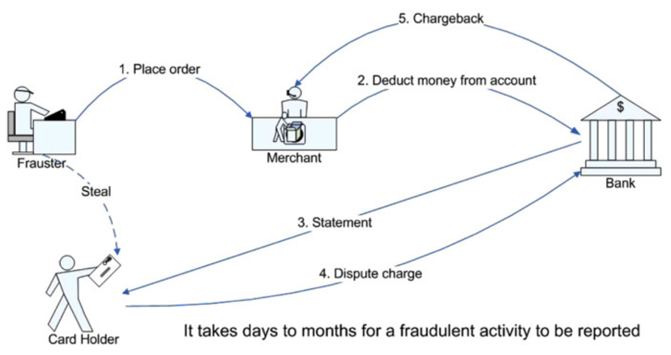

1. Place order
2. Deduct money from account
3. Statement
4. Dispute charge
5. Chargeback
Frauster
Merchant
Bank
Steal
Card Holder
It takes days to months for a fraudulent activity to be reported

Fig. 1. Fraud report procedure.

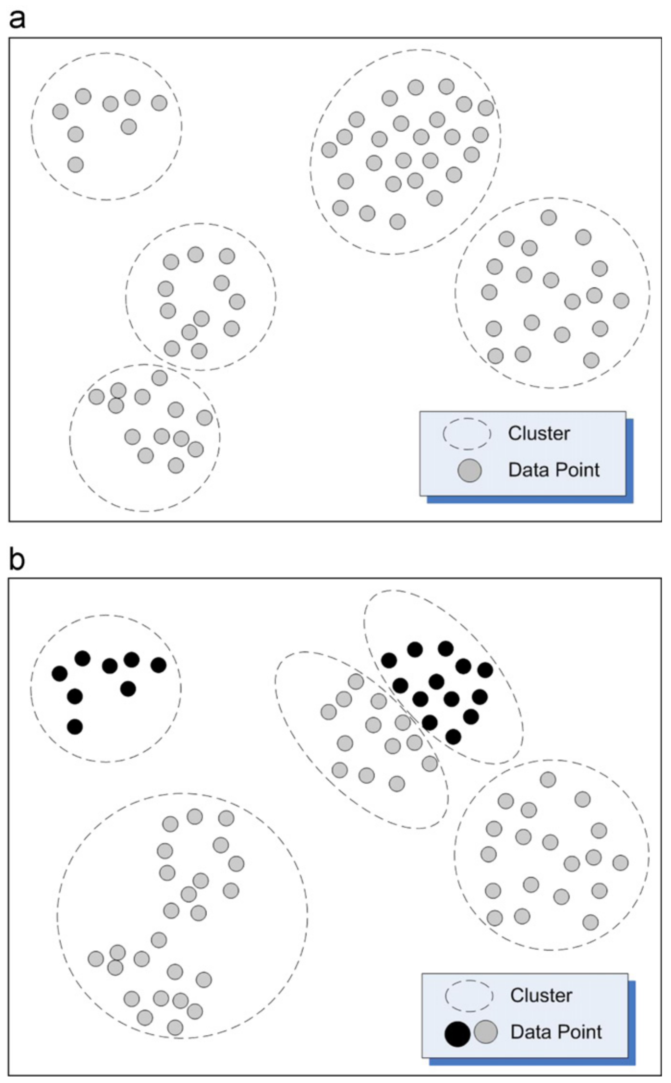


Fig. 2. Nature clustering result vs. desirable result. (a) Unsupervised clustering when the labels of data points are unknown or unused. (b) Desirable clustering when the labels of data points are known.

orders are considered as good before being reported as frauds. They are deemed to be fraud once the company receives fraud reports. The time gap from the date of an order to the date of the fraud report leads to mislabeled data. Mislabeled data could introduce noise to the supervised leaning. Waiting for a few months until most of the fraud reports are completed might reduce the mislabeled noise, but lose would have been happened and the patterns could have been changed at the time they are found.

The ability to learn from unlabeled data and deal with abnormal data makes clustering a good candidate for network intrusion detection and fraud detection. However, on the other hand, a clustering algorithm may not produce desirable clusters without additional information. Fig. 2 is an example. Clustering result illustrated in Fig. 2(a) is perfect in terms of unsupervised learning. The data points are grouped into clusters based on their natural similarities. However, the actual desirable clustering result is in Fig. 2(b) if we know the data labels. Guided by all or a portion of data labels, a clustering algorithm could achieve this desirable result. Based on the potential to combine the strength of classification and clustering, supervised clustering technique is therefore applied to our research.

2. Background

The techniques for fraud detection and intrusion detections fall into two categories: statistical techniques and data mining techniques. Traditional methods of network intrusion detection are based on the saved patterns of known events. They detect network intrusion by comparing the features of activities to the attack patterns provided by human experts. One of the main drawbacks of the traditional methods is that they cannot detect unknown intrusions. Moreover, human analysis becomes insufficient when the volume of the activities grows rapidly. This leads to the interest in data mining techniques for fraud detection and network intrusion detection [10,19].

Data mining based network intrusion detection techniques can be categorized into misuse detection and anomaly detection [19]. The misuse detection techniques build the patterns of the attacks by the supervised learning from the labeled data. The main drawback of the misuse detection techniques is that they cannot detect new attacks that have never occurred in the training data. On the other hand, the anomaly detection techniques establish normal usage patterns. They can detect the unseen intrusions by investigating their deviation from the normal patterns.

The artificial neural networks provide a number of advantages in fraud detection and network intrusion detection [5]. The applications of the neural network techniques includes both the misuse detection models and the anomaly detection models [18,25]. A multi-layer perceptron (MLP) was used in [13] for anomaly detection. A single hidden layer neural network was used and tested on the Defense Advanced Research Projects Agency (DARPA)1998 data. The MLP was applied in [22]. The back-propagation algorithm was used in the learning phase to adapt the weights of the neural network. As an unsupervised neural network, the self-organizing maps (SOM) has been applied in anomaly detection. It implicitly prepares itself to detect any aberrant network activity by learning to characterize the normal behaviors [25]. The SOM was also applied to perform the clustering of network traffic and to detect attacks in [14]. The SOM was designed to learn the characteristics of normal activities in [12]. The variations from normal activities provided an indication of a virus. The unsupervised niche clustering (UNC), a genetic niche technique for unsupervised clustering was applied to the intrusion detection in [21]. Each cluster evolved by the UNC was associated with a membership function that followed a Gaussian shape. Using the normal samples, the UNC generated clusters summarizing the normal space.

A hybrid artificial intelligent system is presented in [23]. An unsupervised neural model was embedded in a multi-agent system for network intrusion detection. Hybrid learning approaches [1,8] integrate different learning and adaptation techniques to overcome individual limitations. Combining the strength of two or multiple approaches could achieve high efficiency. A hybrid model of the SOM and the MLP was proposed in [5]. The SOM was combined with the feed-forward neural network to detect the dispersing and possibly collaborative attacks.

Traditional fraud detection approaches face the same problems as traditional methods of network intrusion detection. Fraud detection analysis is conducted by fraud specialist by comparing the fraud activities with normal transactions. Human analysis becomes insufficient as the volume of the transactions grows up rapidly. Moreover, traditional fraud detection is not receptive to new or changing patterns. Data mining based fraud detections are also categorized into misuse detection and anomaly detection. There have been many commercial data mining tools available. The following commercial tools are on the top level: SAS Enterprise Miner, SPSS Clementine, and IBM DB2 Intelligent Miner. These commercial tools can be used effectively for discovering patterns in data. There are less research reports on fraud detection than those on network intrusion detection despite their similarity. This is simply because financial data usually do not open to the public like KDD99 data for network intrusion detection and are hard to acquire. Decision trees are used in [6] for fraud detection. Their approach divided the large data set of labeled transactions into smaller subsets. Then it used decision tree to generate classifiers in parallel and combined the resultant base models by metal-earning [7] from the classifiers' behavior to generate a meta-classifier. Brause et al. [3] combined radial basis function network and rule-based information for credit card fraud detection.

3. Improved competitive learning network

The ICLN is developed from the SCLN. It overcomes the shortages of instability in the SCLN and converges faster than the SCLN. Therefore it obtains a better performance in terms of the computational time.

3.1. The limitation of SCLN

The SCLN consists of two layers of neurons: the distance measure layer and the competitive layer. The structure of SCLN is shown in Fig. 3. The distance measure layer consists of m weight vectors $W=\{w_1,w_2,\ldots,w_m\}$. When a training example is presented, the distance measure layer calculates the distance between the weight vectors and the training example. The distances calculated in the distance measure layer become the input of the competitive layer. The competitive layer finds out the closest weight vector of the training example. The output of the competitive layer is a $1\times m$ vector. Each bit of the output vector is either 0 or 1, representing the competitive result of the weight vectors. For example, if neuron w_j won the competition, output would be a $1\times m$ vector with $y(j)=1$ and $y(i)=0\ \forall i\neq j$. The winning weight vector w_j is then rewarded to be closer to the training sample. Every time the winning weight vector moves towards a particular sample. The other unwon weight vectors will remain unchanged. This process is repeated for all the training samples for many iterations. Eventually each of the weight vectors would converge to the centroid of a cluster.

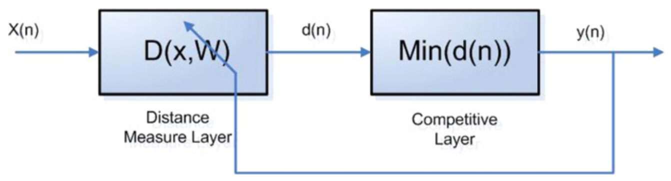


Fig. 3. The SCLN consists of two layers of neurons: the distance measure layer and the competitive layer.

The update rule of the SCLN is called "winer takes all". That means only one wining neuron updates itself each time when a training example is presented. The wining neuron would update itself to move closer to the training sample once it won the competition. The update is calculated by the standard competitive learning rule:

$$w_j(r+1)=w_j(r)+\eta(r)(x-w_j(r)) \tag{1}$$

where w_j is the weight vector of the winning neuron j, and η is the learning rate. Only one wining neuron updates itself once in a time. The essence of competitive learning is illustrated in Fig. 4.

The performance of the SCLN relies on the number of initial neurons and the value of their weight vectors. Once the number of output neurons is set, the number of clusters is also predetermined regardless of data distribution. On the other hand, different initial weight vectors may lead to different number of final clusters because the update function in Eq. (1) only changes the weight vector of the winning neuron toward its local nearby examples. Fig. 5 shows a scenario that reveals the limitations of the SCLN. In this example, two neurons are initialized close to one cluster. Both of them will stay in the clustering result since SCLN is a reward only algorithm. The SCLN clustering result of this example will be four clusters although only three clusters are expected as shown in Fig. 5(b).

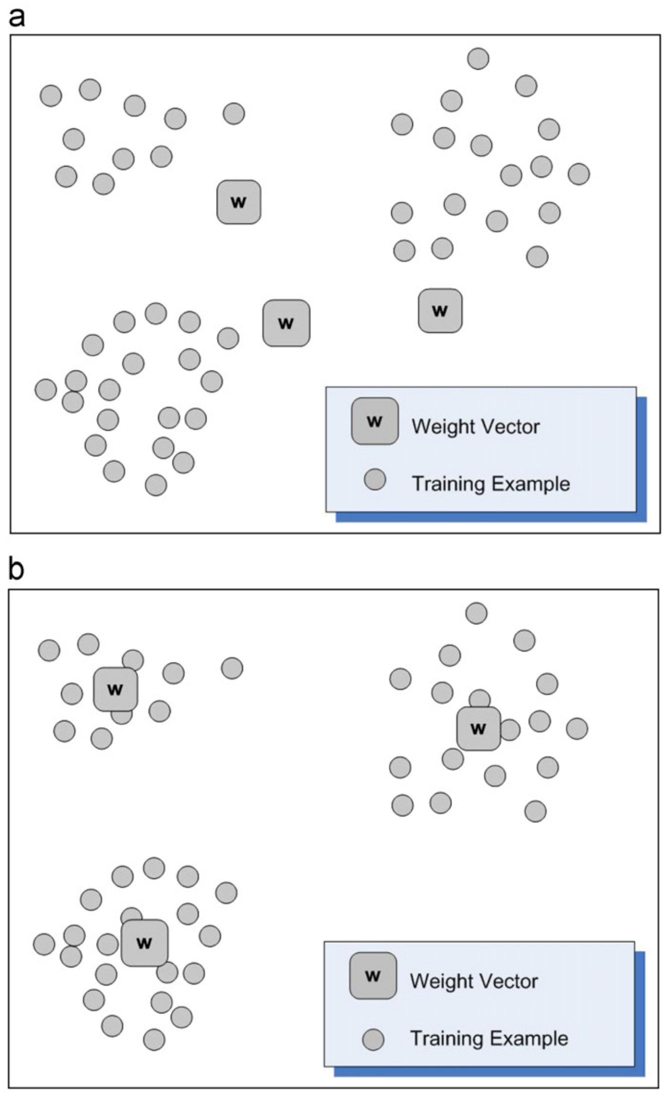


Fig. 4. The principle of the SCLN. (a) Initial weight vectors. (b) Clustering result.

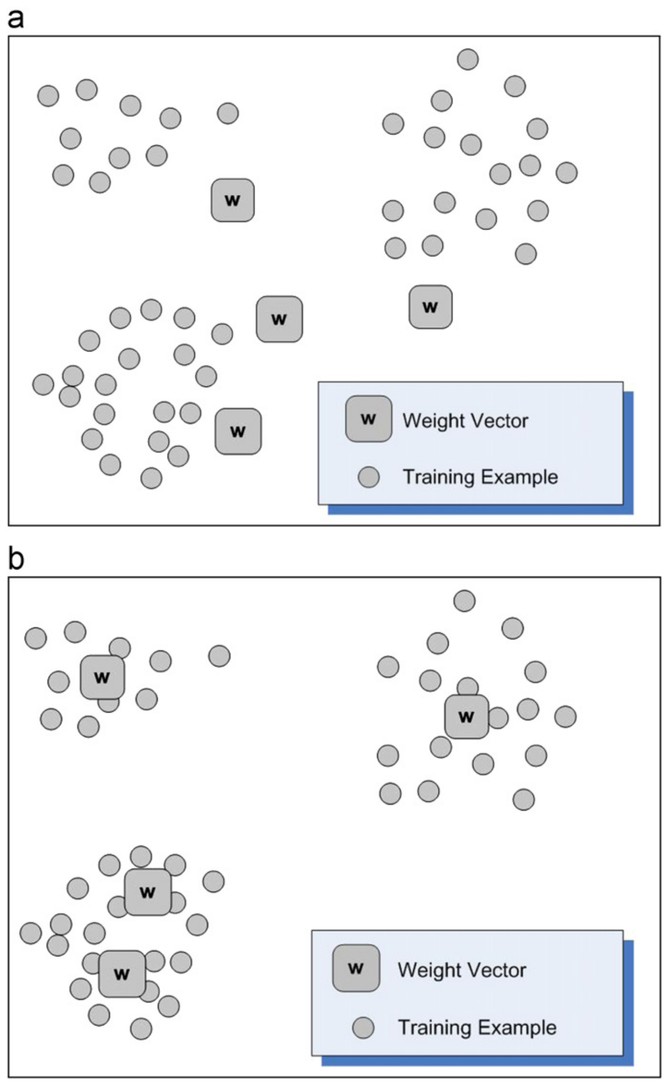


Fig. 5. The drawback of the SCLN. (a) Initial weight vectors. The performance of the SCLN depends heavily on the number of the initial neurons and their initial weight vectors. (b) Clustering result. The left lower cluster is separated into two groups since two weight vectors are initialized close to one cluster.

3.2. New update rules in ICLN

The ICLN changes the SCLN's reward-only rule to reward-punish rule. The winning neuron updates its weight vector by the same update rule in Eq. (1). This updated process is also called reward as the wining neuron is updated toward the training example. At the same time, the other neurons also update their weight vectors by

$$w_j(r+1)=w_j(r)-\eta_2(r)K(d(x,j))(x-w_j(r)) \tag{2}$$

where $K(d(x,j))$ is a kernel function in which $d(x,j)$ is the distance between neuron j and the input x, and η_2 is the learning rate. This update process is called punish as the neurons are updated to move away from the training example. There are various choices of the kernel function $K(d(x,j))$, such as the inverse distance, the triangular kernel, the quadratic kernel, and the Gaussian kernel [2]. A kernel function obtains the maximum value at zero distance, and the value decays as the distance increases. A good kernel function smooths and regulates the updated value.

The effect of the reward-punishment update rules is shown in Fig. 6. The two weight vectors at the left bottom of Fig. 5(a) compete against each other when applying ICLN. The punish rule pushes the losing weight vectors away from the cluster and one of them will finally be removed from the cluster. Furthermore, since the distance between the training example and all of the weight vectors are always calculated for the competition, using these values to apply punish rules to the losing weight vectors will accelerate the clustering process without additional calculations.

3.3. The ICLN algorithm

The ICLN algorithm is outlined in Fig. 7. The ICLN first initializes k neurons. Two methods can be used for the initialization. One is to assign the k neurons with k random training examples. The other is to assign the center point, i.e. the mean, of the whole training data to all k neurons. Center point initialization takes more iterations to converge than random one. The number of clusters k is usually set to a number bigger than the

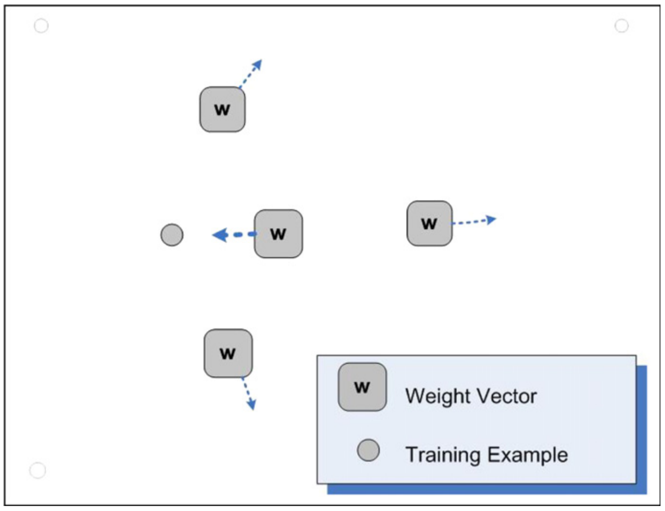


Fig. 6. The effect of the ICLN update rules.

```
Input: $X=\{x_1,x_2,\ldots,x_n\}$: the input dataset
Output: $W=\{w_1,w_2,\ldots,w_k\}$: the weight vectors
BEGIN
1. Randomly initialize the weight vectors $W=\{w_1,w_2,\ldots,w_k\}$
2. Setup the learning rates $\eta_1$ and $\eta_2$ for the winning neuron
   and the losing neurons, respectively. $0<\eta_2<\eta_1<1$.
3. Setup the minimum weight update value $T_\triangle$
4. Select Kernel function: $K(d(x_i,w_j))=e^{-d^2(x_i,w_j)}$
5. Setup maximum number of iterations $M_{epoch}$
repeat
  for $x_i \in X$ do
    for $w_j \in W$ do
      compute the distances: $d(x_i,w_j)=\| x_i-w_j \|$
    end for
    $w_{win}=w_a \quad if \quad a=arg\min_{m=1}^{k} d(w_m,x_i)$
    /*Update $w_{win}$*/
      $w_{win}=w_{win}+\eta_1(x_i-w_{win})$
    /*Update other weight vectors $w \in W \wedge w \neq w_{win}$ */
      $w=w-\eta_2 K(d(x_i,w))(x_i-w)$
  end for
until $|\triangle w|<T_\triangle \quad \forall w \in W$ or $(\#iteration > M_{epoch})$
Remove all weight vectors that have no associated input.
END
```

Fig. 7. Algorithm: the improved competitive learning network.

expected number of clusters because the ICLN can reduce the number of clusters but cannot add new cluster to the network.

The ICLN trains the initial weight vector $W=\{w_1,w_2,\ldots,w_k\}$ by randomly presenting the training examples $X=\{x_1,x_2,\ldots,x_n\}$. When a training example x is presented to the ICLN, the weight vectors compete to each other by comparing their distance, such as Euclidean distance, to x. Only one weight vector with the minimum distance to x wins the competition:

$$w_{win}(x)=w_j \quad \text{if } j=\arg\min_{i=1}^{k} d(w_i,x) \tag{3}$$

The wining weight vector is then updated by the reward function as in Eq. (1). At the same time, other weight vectors in the network are updated by the punishment function as in Eq. (2). The result of this reward-punishment rule is that the wining neuron is updated to be closer to the training example and the other neurons are updated away from the training example.

After the network finishes learning from x, another training example will be presented to the network. The network will then compete and learn from this new training example. The ICLN iterates the learning process until one of the stopping criteria are satisfied. One criterion is that the maximum update to weight vector W is less than a preset minimum update threshold:

$$\max_{i=1}^{k} \|w_i(r)-w_i(r-1)\| < T_\Delta \tag{4}$$

where $w_i(r)$ is the weight vector w_i in the current iteration, $w_i(r-1)$ is the weight vector w_i in the previous iteration, and T_Δ is the preset minimum update threshold. The other criterion is that it finishes the preset maximum number of iterations.

4. Supervised improved competitive learning network

The SICLN is a supervised clustering algorithm derived from the ICLN. When data labels are available, the SICLN uses them to guide the clustering procedure.

4.1. The objective function

The SICLN uses an objective function $Obj(X,W)$ to measure the quality of the clustering result. The purpose of the objective function is to minimize the impurity of the result clusters and keep a minimum number of clusters. The objective function is defined as

$$Obj(X,W)=\alpha\times Imp(X,W)+\beta\times Sct(X,W) \tag{5}$$

where α and β are the weights of impurity and scattering respectively, and $\alpha+\beta=1$.

The impurity of the whole result is the weighted average of the impurity of each cluster:

$$Imp(X,W)=\frac{\sum_{i=1}^{k}|w_i|\times Imp(X,w_i)}{n} \tag{6}$$

where n is the count of the data set X and $|w_i|$ is the count of the cluster members of w_i. One common choice of the impurity function is the misclassification rate. If a cluster contains members that are labeled as classes $\{c_1,c_2,\ldots,c_i\}$, the misclassification rate of this cluster is the percentage of members that are not labeled as the dominate class. The dominate class of a cluster is defined as the most frequent class of its members. For a data set $X=\{x_1,\ldots,x_n\}$ which are labeled as classes $C=\{c_1,\ldots,c_t\}$, c_j is the dominate class of w_i if the count of members of w_i belong to class c_j is more than those belong to any other class

$$DomC(w)=c_j$$

$$\text{if } \#(x\in c_j)>\#(x\notin c_j;x\in w_i) \quad \forall x\in w$$

where # denotes the count. Similarly, the second dominate class $Dom_2C(w)$ is the class that has more members than other classes except the dominate class. The misclassification rate of cluster of weight vector w_i is computed as

$$Misrate(w_i)=\frac{\#(x\notin DomC(w_i);x\in w_i)}{|w_i|} \tag{7}$$

where $|w_i|$ is the count of the members of w_i.

When misclassification rate is chosen as the impurity function, according to Eqs. (6) and (7), the impurity is calculated as

$$Imp(X,W)=\frac{\sum_{i=1}^{k}|w_i|\times\dfrac{\#(x\notin DomC(w_i))}{|w_i|}}{n} =\frac{\sum_{i=1}^{k}\#(x\notin DomC(w_i))}{n} \tag{8}$$

An alternative to misclassification rate is the GINI impurity measure. The GINI impurity measure was first used in classification and regression trees (CART) [4] and has been widely used to determine the purity of split in decision trees. The GINI of a weight vector w_i is computed as

$$Gini(w_i)=1-\sum_{j=1}^{t}\left(\frac{\#(x\in c_j;x\in w_i)}{|w_i|}\right)^2 \tag{9}$$

where c_j is the class of members of w_i from $j=1;2;\ldots;t$, and $|w_i|$ is the size of w_i. A smaller GINI indicates a lower impurity. The GINI value reaches its maximum if the members are equal population in each class. By contrast, GINI value is 0 if all members belong to one class. When GINI is chosen, according to Eqs. (6) and (9), the impurity is calculated as

$$Imp(X,W)=\frac{\sum_{i=1}^{k}|w_i|\times\left(1-\sum_{j=1}^{t}\left(\dfrac{\#(x\in c_j;x\in w_i)}{|w_i|}\right)^2\right)}{n} \tag{10}$$

The second part of the objective function is the scattering. A simple choice of the scattering function is to compare the number of clusters and the number of data points:

$$Sct(X,W)=\sqrt{\frac{k-t}{n}} \tag{11}$$

where t represents the number of classes of the data set X, n is the number of data points, and k is the number of clusters. A bigger scattering indicates a wider spread clustering. When the number of clusters equals to the number of data points, scattering reach its maximum. By contrast, if the number of clusters is the same as the number of classes, scattering is 0. An alternative choice is to use the size of each cluster:

$$Sct(X,W)=\sqrt{\sum_{i=1}^{k}\frac{1}{|w_i|}\times(k-t)} \tag{12}$$

Scattering is bigger when the variance of the size of the clusters are bigger. Eliminating small size clusters will minimize this clustering function.

For fraud detection and intrusion detection, we chose misclassification function as the impurity function since it is easy to set up by business goal. We choose the first scattering function because the alternative choice intent to remove small size clusters but fraud and intrusion are usually in small size clusters. Combining Eqs. (5), (8) and (11), the following objective function is chosen to evaluate the quality of a clustering result.

$$Obj(X,W)=\alpha\times\frac{\sum_{i=1}^{k}\#(x\notin DomC(w_i))}{n}+\beta\times\sqrt{\frac{k-t}{n}} \tag{13}$$

where α is the weight of impurity, and β is the weight of scattering. A smaller $Obj(X,W)$ indicates a better clustering result. Minimizing $Obj(X,W)$ attains the best result. It means to minimize both impurity and scattering. However, impurity and scattering usually conflict to each other. Decrease of either one leads to increase of the other.

4.2. The SICLN algorithm

The SICLN is outlined in Fig. 8. It first initializes k neurons. The initialization methods of the SICLN are the same as those of the ICLN. It is not as important because the network will be reconstructed in training.

The SICLN also labels the weight vectors with their member data points. In the SICLN, a weight vector is labeled to a class with the biggest population of its cluster members. If all members are in "unknown", this weight vector will be labeled as "unknown". Fig. 9 illustrates how the initial weight vectors are labeled. w_1 and w_5 are labeled as "Black" because their black point members are more than gray point members. w_2 and w_4 are labeled as "Gray" because gray points of their members are more than black points. w_3 is labeled as "unknown" because all of its members are missing label. w_6 is labeled as "unknown" because it has no data member.

The learning step of the SICLN is a revised version of the ICLN. The output neurons compete to be active. Since labels are available, The SICLN uses the labels to update the weight vectors. In the new rule, only neurons with the same class as the training example or "unknown" has the right to compete to win. Neurons that are labeled as different class will lose. When a neuron won the competition, its weight vector would be rewarded by the same update rule as Eq. (1) in the ICLN. The punishment update is also the same as the ICLN in Eq. (2).

```
Input: $X = \{x_1, x_2, \ldots, x_n\}$: the input dataset
Input: $L(X) = \{L_{x_1}, L_{x_2}, \ldots, L_{x_n}\}$: the label of the dataset
Input: $C = \{c_1, c_2, \ldots, c_t\}$: the classes of the dataset
Output: $W = \{w_1, w_2, \ldots, w_k\}$: the weight vectors
Begin
1 //Initialize//
  1.1 Randomly initialize the weight vectors $W = \{w_1, w_2, \ldots, w_k\}$
  1.2 Setup other parameters such as learning rate, Kernel function,
      objective thresholds, iteration limit
2 //Training//
repeat
  2.1 Identify members of $W$
  2.2 Label $W$ with their dominate class.
  2.3 //Learning//
  for $\forall x \in X$ do
      2.3.1 Look for wining weight vector $w_{win}$ regarding input $x$
      2.3.2 Update $w_{win}$ with reward function
      2.3.3 Update other $w \neq w_{win}$ with punishment function
  end for
  2.4 If objective threshold is satisfied, stop
  2.5 Split weight vector $w \in W$ if estimate objective function value
      is higher than before splitting. Construct new vectors $W = W_{new}$
  2.6 //Learning (same as 2.3)//
  for $\forall x \in X$ do
      2.6.1 Look for wining weight vector $w_{win}$ regarding input $x$
      2.6.2 Update $w_{win}$ with reward function
      2.6.3 Update other $w \neq w_{win}$ with punishment function
  end for
  2.7 Merge weight vector $w_i$ and $w_j$ if estimate objective function value
      is higher than before merging and they are the same class and closer
      to each other than any other weigh vectors. Construct new vectors
      $W = W_{new}$
until Object function is satisfied or iteration exceeds limit
3 Remove all weight vectors that have no member data point.
4 Output $W = \{w_1, w_2, \cdots, w_k\}$
End
```

Fig. 8. Algorithm outline: supervised improved learning competitive network.

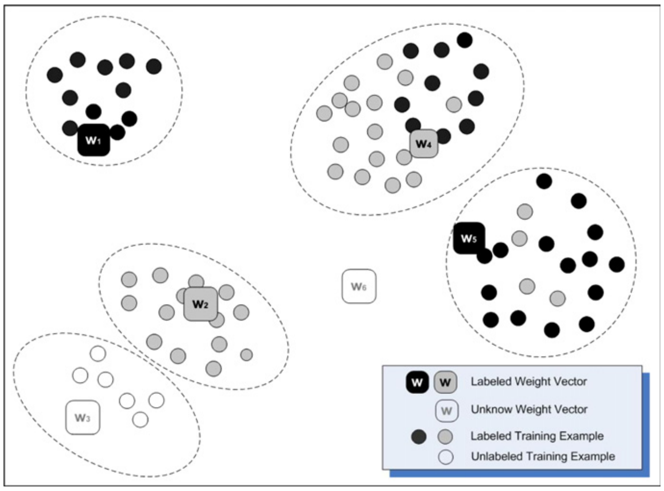


Fig. 9. The SICLN labels the weight vectors with their member data points.

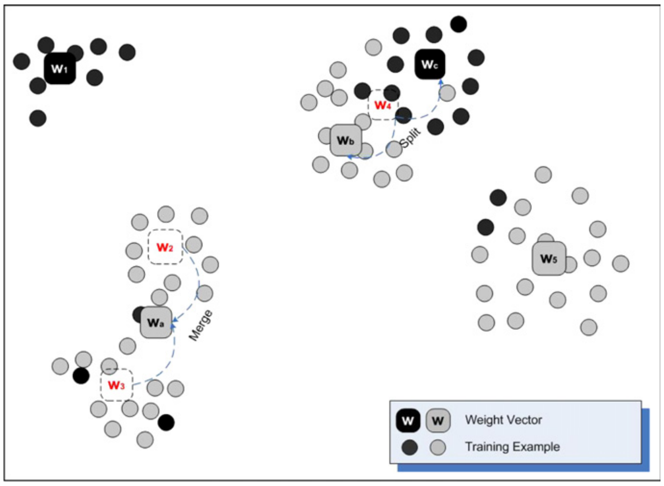


Fig. 10. The reconstruction process of the SICLN.

In the SICLN, when a labeled training example is presented to the network, only the neurons of the same class or "Unknown" class are able to win to get the reward. However, if an unlabeled training example is presented to the network, all neurons in the network have the ability to compete to get reward or punish. In this case, the learning step of the SICLN becomes the same as that of the ICLN. If all training examples in the data set are unlabeled, all train data and weight vectors belong to "unknown" class. At this point, the SICLN becomes an ICLN.

After the learning step, the SICLN will reconstruct a new network based on the trained network. In the reconstruction step, a neuron is split into two new neurons if it contains many members belonging to other classes. On the other hand, two neighboring neurons are merged into one if they belong to the same class. Fig. 10 illustrates the reconstruction step of the SICLN.

The split process starts from the clusters with the maximum impurity values. An estimated after split impurity is between the after split impurity and the best possible impurity. For example, if weight vector w_s is split to two vectors w_{s_1} and w_{s_2}, the after split impurity is

$$Imp_{after}(X,W) = \frac{\sum_{i=1}^{k+1} \#(x \notin DomC(w_i))}{n}$$

The best possible impurity is

$$Imp_{best}(X,W) = \frac{\sum_{i=1,i \neq s}^{k-1} \#(x \notin DomC(w_i))}{n} + \frac{\#(x \notin DomCw_s, x \notin Dom_2C(w_s))}{n}$$

The estimated value of the impurity after split is

$$\widehat{Imp}(X,W) = Imp_{after}(X,W) + \gamma \times (Imp_{best}(X,W) - Imp_{after}(X,W))$$

where γ is an estimate factor $0<\gamma<1$. The Scattering value after split is

$$\widehat{Sct}(X,W) = \sqrt{\frac{k+1-t}{n}}$$

The estimated objective function value is

$$\widehat{Obj}(X,W) = \alpha\, \widehat{Imp}(X,W) + \beta\, \widehat{Sct}(X,W)$$

If the estimated objective value is smaller than the objective value before split, the weight vector will be split into two. The median point of members of the dominate class and the median point of the members of the second dominate class of the neuron are selected to be the new neurons.

The merge process looks for the closest same class weight vectors as the candidates. To find out two weight vectors that are closer to each other than to any other weight vectors, we use the *mutual neighbor distance* [17]. The *mutual neighbor distance* is

$$MND(w_i,w_j) = NN(w_i,w_j) + NN(w_j,w_i) \tag{14}$$

where $NN(w_i,w_j)$ is the neighbor number of neuron w_j with respect to neuron w_i.

If $MND(w_1,w_2)=2$ (e.g. $NN(w_1,w_2)=1$ and $NN(w_2,w_1)=1$), w_1 and w_2 are closer to each other than to any other weight vectors. w_i and w_j are merged if they meet the following conditions: (1) $C(w_i)=C(w_j)$, (2) $MND(w_i,w_j)=2$, (3) $\widehat{Obj}(X,W_s) < Obj(X,W)$. The new neuron takes the mean of w_i and w_j as its weight vector.

The reconstruction step creates a "new" network by splitting or merging the weight vectors, driven by the objective function. This "new" network will replace the old one to continue the learning step. This learn—reconstruct iteration—repeats until one of the following stopping criteria is satisfied: (1) the objective function value satisfies the minimum threshold; (2) the training reach the maximum number of iterations.

4.3. The SICLN vs. the ICLN

While ICLN has the capability to cluster data in its nature groups. The SICLN uses labels to guide the clustering process. The ICLN groups data into clusters by gathering closer data points into the same group. As a supervised clustering algorithm, the SICLN minimizes the impurity of the groups and the number of groups. Fig. 11 shows the improvement from the ICLN to the SICLN. The result of the ICLN is in Fig. 11(a). The data are identified in their nature groups without looking at the data labels. Weight vectors w_2 and w_3 become the cluster center of two groups of data at the left bottom although both groups belong to the same class. On the other hand, weight vector w_4 represents the group of data on the right upper, which contains data of two classes. The result of the SICLN applying to the same data is in Fig. 11(b). Weight vector w_4 is split into w_b and w_c, which represent the centers of two groups of data with different classes. Therefore, the purity of the clustering result is higher than that of the ICLN. At the same time, the SICLN attempts to result in less clusters while keeping the same level of purity. Weight vectors x_2 and w_3 are merged to w_a. The new weight vector w_a becomes the center data group w_2+w_3, which belongs to the same class.

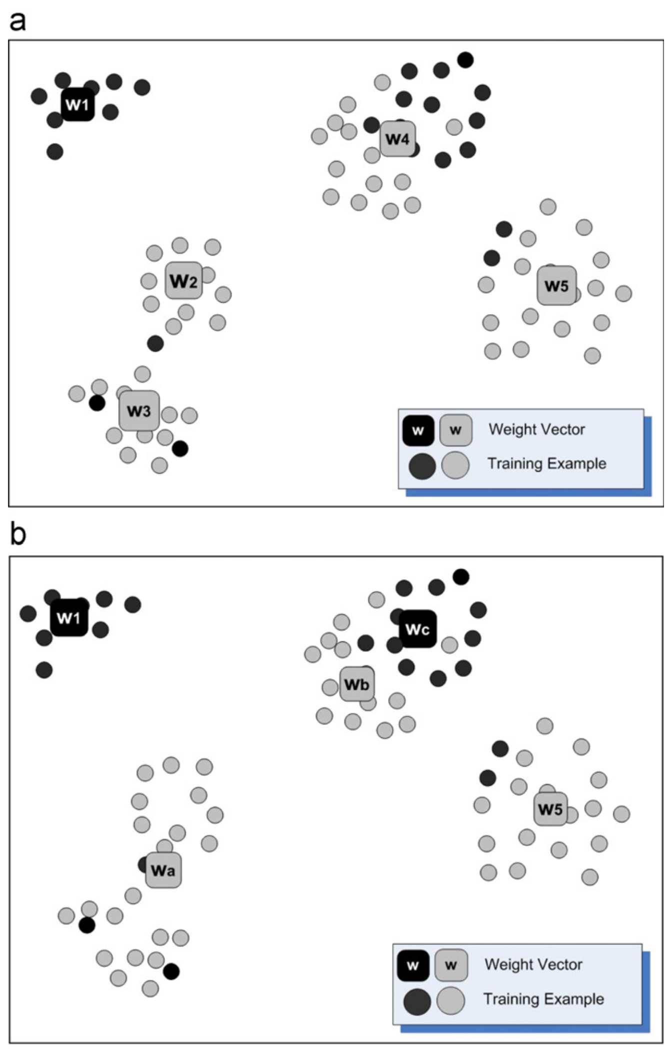


Fig. 11. The ICLN vs. the SICLN. (a) ICLN: cluster data in its nature groups. Same class data at the left bottom are cluster into two groups represented by weight vectors w_2 and w_3. Data of different classes in the middle are clustered into one group represented by weight vector w_4. (b) SICLN: optimize the purity of the clusters and the number clusters. Weight vector w_4 in (a) is split into w_b and w_c to maximize the purity of the clustering. Weight vectors w_2 and w_3 are merged to w_a to minimize the number of clusters.

5. Experimental comparisons

In this section, we compare the performance of the SICLN and the ICLN with the *k*-means and SOM on three data sets: the Iris data, the KDD 1999 data, and the Vesta transaction data.

5.1. Evaluation metrics

The outputs of a prediction or detection model fall into four categories: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). TP and TN are correct prediction or detection while FP and FN are incorrect prediction or detection. The evaluation metrics includes: accuracy, precision, recall, and receiver operating characteristic (ROC) curve. They are calculated as

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

The three metrics describe the percentage of correct prediction. However, none of them alone can represent the performance of an algorithm. A high accuracy may not represent a better result because the cost of incorrect predictions of positive data and negative data are usually different. A high accuracy result may have a low recall. A high recall may not indicate a good result either because the recall can easily increase by decreasing the precision. A high precision result may come with a very low recall. ROC curve [11] is a graphical plot of the TP vs. FP rate as the threshold of the classification varies. It illustrates the trade-off between TP rate and FP rate.

5.2. Iris data

Iris data is a data set with 150 random samples of flowers from the iris species setosa, versicolor, and virginica. From each species there are 50 observations for sepal length, sepal width, petal length, and petal width in cm. The data set contains three classes. Each class refers to a type of iris plant. In the three classes, one class is linearly separable from the other two classes; the latter are not linearly separable from each other.

We initialized the number of clusters of SICLN, the ICLN and k-means to 5. Since the number of neurons in the SOM had to form a rectangle, we select 6 as its initial number, e.g. 3×2. Fig. 12 shows the performance comparison of these algorithms. The SICLN outperforms the others in accuracy. The performance of the ICLN and the SOM are almost identical. The k-means has the lowest accuracy.

To test SICLN's capability of deal with missing labels, we masked some of the labels. We randomly masked 100%, 70%, 50%, 30%, and 20% labels in the iris data set and applied the SICLN on them. Fig. 13 shows performance results. The SICLN become the ICLN when 100% labels are missing. The result is exactly the same as the ICLN. The performance of the SICLN become better with the increase of the available labels. When there are enough labeled data to guide the clustering procedures (less than 20% labels were missing for the case of Iris data), the SICLN reached its highest performance.

In another experiment we tested the SICLN's ability of adapting to different initial number of weight vectors. The SICLN was initialized to 1, 3, 5, 8, 10, and 15 neurons. The SICLN consistently resulted in five clusters.

5.3. Network intrusion detection: KDD-99 data

The KDD-99 data set was used for the Third International Knowledge Discovery and Data Mining Tools Competition. This data set was acquired from the 1998 DARPA intrusion detection evaluation program. There were 4,898,431 connection records, of which 3,925,650 were attacks. Each data point is a network connection, which is represented by 41 features, including the basic features of the individual connections, the content features suggested by the domain knowledge, and the traffic features computed using a 2-s time window [16]. Each connection is labeled as "normal" or a particular type of the attacks: neptune, smurf, ipsweep, or back DoS. The nature of these attacks are described in [24,26]. From this data set, 501,000 records were chosen in our experiment. The selected connections were further split into the training set and the test set, containing 101,000 and 400,000 connections, respectively.

The performance comparison is shown in Fig. 14. The SICLN is better than the other algorithms in three evaluation metrics. The ROC curves are illustrated in Fig. 15. It shows that all of them have good performance. In addition, the results of the SICLN shows its

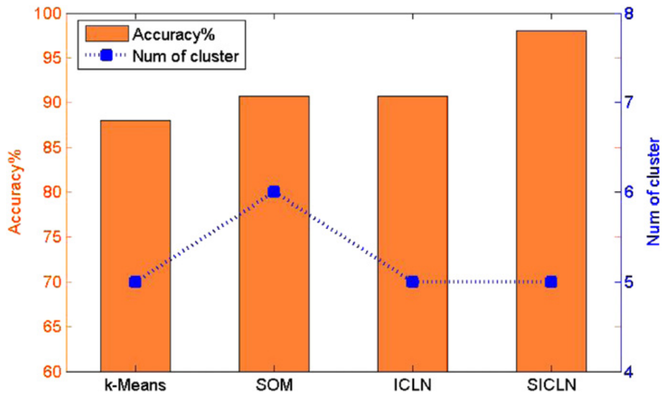


Fig. 12. Performance comparison on the Iris data.

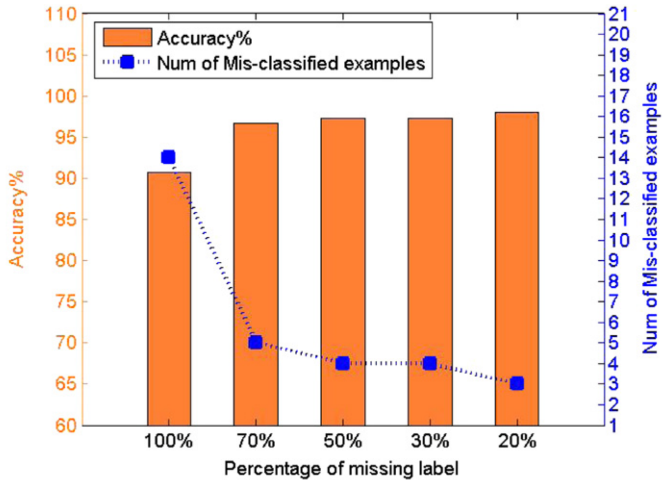
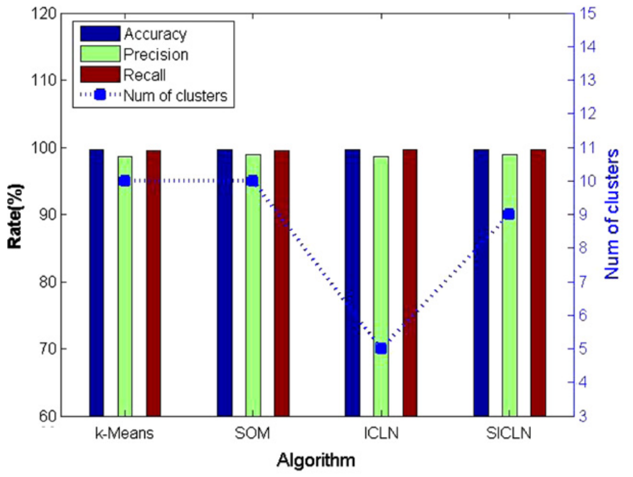
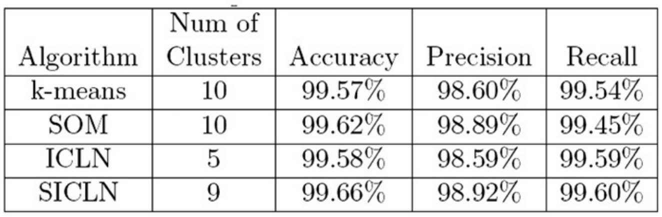


Fig. 13. Performance of the SICLN on Iris data with missing labels.





Algorithm	Num of Clusters	Accuracy	Precision	Recall
k-means	10	99.57%	98.60%	99.54%
SOM	10	99.62%	98.89%	99.45%
ICLN	5	99.58%	98.59%	99.59%
SICLN	9	99.66%	98.92%	99.60%

Fig. 14. Performance comparison on KDD99 data.

capability to distinguish small population classes when we brake down the results into individual class level as shown in Table 1. Attack type neptune and ipsweep have only 0.03% and 0.91% of the population in the data set, and they are similar to each other. Although these neptune and ipsweep connections are detected as attacks in all algorithms, *k*-means, the SOM, and the ICLN are not able to distinguish these two attack types from each other. The SICLN outperformed *k*-means, SOM, and ICLN in terms of misclassification rate. The capability of knowing the types of the attack bring better automatic solutions or treatments. The clustering detail also shows that the SICLN has the capability to distinguish clusters of small population.

We also masked some of the labels to test whether the SICLN can deal with missing labeled data. The results show that when all label are missing, SICLN becomes an ICLN. The performance reaches the highest point as more than 70% labels are available. We tested the SICLN's capability to adapt to different initial number of weight vectors as well. Starting from the initialized number of 1, 5, 10, 15, 20, or 30 neurons, the SICLN consistently converged to 10 clusters.

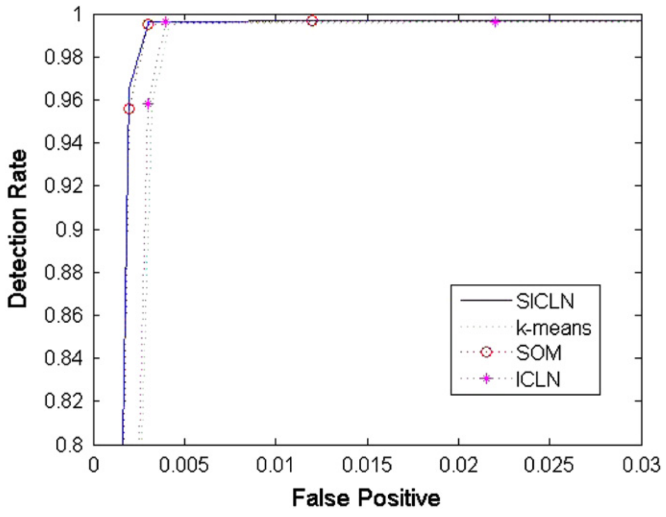


Fig. 15. ROC curves of SICLN, *k*-means, SOM, and ICLN on KDD-99 data.

Table 1
Misclassify rate on individual class.

Class	Popul. (%)	Misclassify rate (%)			
		k-Means	SOM	ICLN	SICLN
Normal	77.12	0.42	0.33	0.42	0.32
Neptune	0.03	100	100	100	0
Smurf	21.88	0	0	0	0
Ipsweep	0.91	7	7	7	3.2
Back	0.06	100	100	100	100

5.4. Fraud detection: credit card payment data

The data we used for this experiment is the fraud detection data from Vesta Corporation. The data contain credit card transactions of calling cards of a telecommunication company. Vesta corporation is an innovator and worldwide leader in virtual commerce with headquarter in Portland, Oregon, USA. The company is servicing most major US telecommunications carriers, including AT&T, T-Mobile, Cricket, Verizon, and Sprint.

Fig. 16 shows the data flow for fraud analysis. The on-line transaction processing (OLTP) servers transfer data to the on-line analytical processing (OLAP) data warehouses, on which data mining tasks for fraud detection and business analysis are performed. Front end OLTP data are integrated to data warehouse in analysis servers through the backup servers in a daily frequency. Data mining and analysis tasks are performed in analysis servers by risk management using Microsoft SQL and SAS Enterprise Miner.

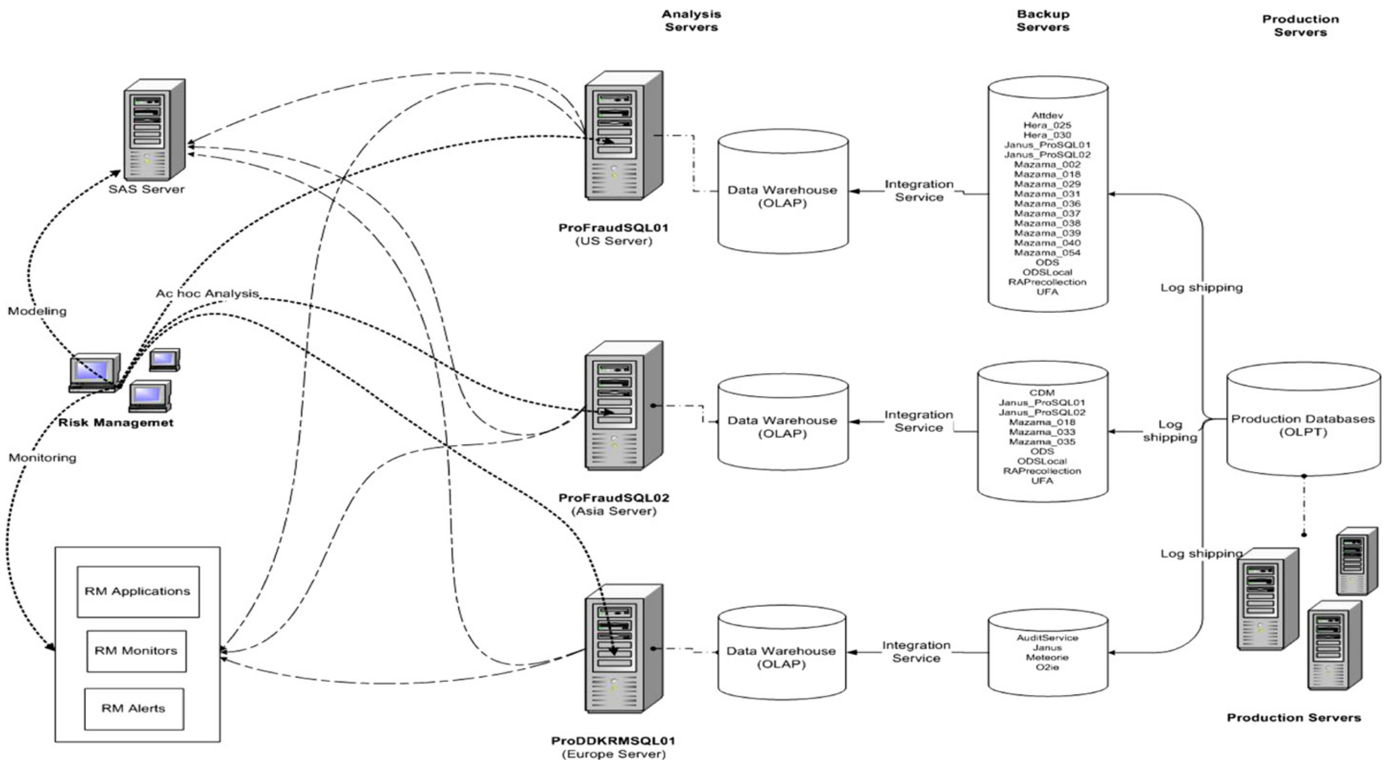


Fig. 16. Data flow of Vesta data for fraud analysis.

The data used in this experiment are a portion of credit card payment transactions of one of vesta's telecommunication partner. This data set contains three months transaction history of 206,541 credit cards, in which 204,078 are normal and 2463 are fraudulent. By combining business knowledge, simple statistics, and statistics measures, 21 variables were selected from the raw data.

As stated in Section 1, data in fraud detections are highly skewed. There are many more good events than fraudulent events. In fraud detection, recall rate is more important than the overall accuracy and precision. Accuracy alone cannot reflect the quality of the algorithms because simply predicting that all transactions are good events, although this is equivalent to not detecting fraud at all can still gets high accuracy. In this case, the native ratio of fraud against normal is around 1.2%. The accuracy can be 98.8% if simply guessing every transaction is normal. However, our goal is to detect as many frauds as possible, while keeping false positive rate at a certain acceptable level. Table 2 shows the experimental results on Vesta data. The recall rate of the SICLN is about 20% higher than the others. ROC curve is a better tool for performance comparison in this case. Fig. 17 shows the improvement of the SICLN by the comparison of the ROC curves of the SICLN, *k*-means, the SOM, and the ICLN.

5.5. Discussions

The small size and low dimension of the Iris data make it possible to provide visible learning process of the algorithms. KDD-99 data and Vesta data test the algorithms' scalability and capability of dealing with real-world data.

The SICLN outperforms the other algorithms on all of these three data sets. The improvement is more noticeable when data points of different classes are very close. Furthermore, the SICLN is completely independent from the initial number of clusters since its reconstruction step is able to rebuild the structure of itself based on the data labels. Meanwhile, The SICLN's capability of dealing with missing data is also demonstrated in these experiments. The clustering performance of the SICLN improves when available number of labels increases and it reaches the highest point when about 70% of data points are labeled. This feature makes the SICLN an ideal candidate of algorithms for fraud detection and network intrusion detection since there are always a certain amount of unlabeled and delay labeled data in these domains.

Table 2
Experimental result on Vesta data.

Algorithm	Number of clusters	Accuracy (%)	Recall (%)	Precision (%)
k-Means	12	97.4	60.7	25.7
SOM	13	97.8	54.8	27.8
ICLN	12	97.8	57.4	28.4
SICLN	12	97.4	79.1	28.8

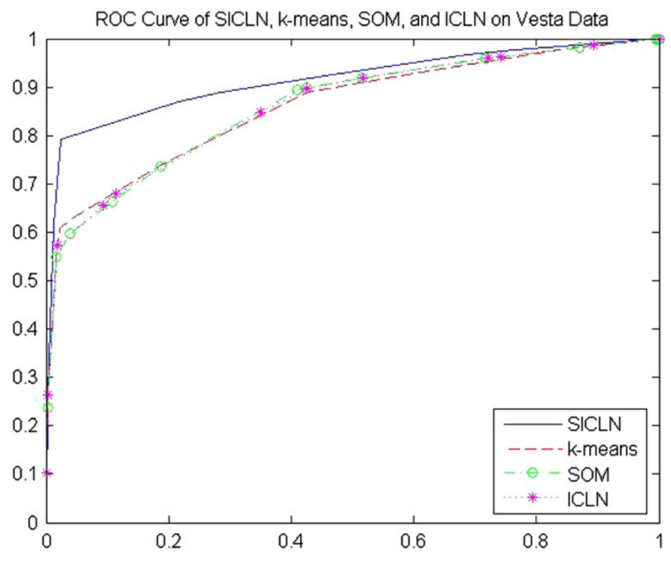


Fig. 17. ROC curves of SICLN, *k*-means, SOM, and ICLN.

6. Conclusion and future work

We have proposed and developed two clustering algorithms: (1) ICLN, an unsupervised clustering algorithm improving from the standard competitive learning neural network, and (2) The SICLN, a supervised clustering algorithm, which introduces supervised mechanism to the ICLN.

The ICLN improves the SCLN by modifying its update rule from the reward only rule to the reward-punishment rule. The new update rule increases the stability and speeds up the training process of the ICLN. Furthermore, the number of final clusters of the ICLN is independent from the number of initial network neurons since the redundant neurons will be finally excluded from the clusters by the punishment rule.

The SICLN is a supervised clustering algorithm derived from the ICLN. The SICLN utilizes labeled data to improve the clustering results. The SICLN modifies the learning rule of the ICLN to train on both labeled and unlabeled data. Furthermore, the SICLN adds the reconstruction step to the ICLN to merge or split the existing weight vectors for the clustering task. The reconstruction step enables the SICLN to become completely independent from the number of initial clusters. An objective function which combines the purity and scattering of the clusters is used in the SICLN to optimize the misclassification rate and the number of clusters.

We compared the performance of the SICLN and the ICLN with the *k*-means and the SOM using three data sets: Iris data, KDD-99 data, and credit card payment data. The ICLN achieve similar accuracy as the other traditional unsupervised clustering algorithms. The SICLN outperforms the other algorithms in all three data set and exhibits the following advantages: (1) achieves low misclassification rate in solving classification problems; (2) is able to deal with both labeled and unlabeled data; (3) has the capability to achieve high performance even when part of data labels are missing; (4) is able to classify highly skew data; (5) has the capability to identify unseen patterns; (6) is completely independent from the initial number of clusters. The experimental comparison demonstrates the SICLN has excellent performance in solving classification problems using clustering approaches. The advantages list above recommend the SICLN could be an ideal algorithm for fraud detections and network intrusion detections.

The following are the future improvements and directions of this research:

- A better estimation method for the reconstruction step may improve the efficiency of the SICLN. If there is a more accurate method to estimate the objective function value, the SICLN could be able to converge faster to the final result.
- Further improvement may be done to avoid local optimization. Although the reconstruction step, the selection of learning rate, and the use of weight decay can reduce the chance of ending to local optimal point for the SICLN. The current SICLN does not guarantee avoiding local optimization. Further research may improve the SICLN from this prospect.

- SICLN has the potential to be modified to an incremental training algorithm although the current SICLN is designed for batch training. An incremental training approach will improve the fraud detection or network intrusion detection system to be an automatic adaptive system without or with small amount of human interaction.
- Introducing fuzzy logic [9] will be a potential big improvement to the SICLN. The difference between the fuzzy clustering and from the traditional clustering is that the output of the fuzzy clustering is the membership function that associates each data point to each cluster. Fuzzy result is helpful for fraud detections and network intrusion detections to specify the likelihood of an activity being a fraud or intrusion event. Knowing the possibility of an activity being a fraud or intrusion event can guide the system to perform proper reactions.

Acknowledgments

The authors graciously acknowledge the funding from the Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) and through Grant RGPN 227441 from the National Science and Engineering Research Council of Canada (NSERC) to Dr. Ghorbani. The data sets for fraud detection used in the experiments were collected in and provided by Vesta Corporation, United States.

References

[1] A. Abraham, E. Corchado, J.M. Corchado, Hybrid learning machines, Neurocomputing (13–15) (2009) 2729–2730.
[2] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning, Artificial Intelligence Review 11 (1–5) (1997) 11–73.
[3] R. Brause, T. Langsdorf, M. Hepp, Neural data mining for credit card fraud detection, in: ICTAI, 1999, pp. 103–106.
[4] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth International, 1984, pp. 21–28.
[5] J. Cannady, Artificial neural networks for misuse detection, in: Proceedings of the 1998 National Information Systems Security Conference, Arlington, VA, 1998, pp. 443–456.
[6] P.K. Chan, W. Fan, A.L. Prodromidis, S.J. Stolfo, Distributed data mining in credit card fraud detection, IEEE Intelligent Systems 14 (6) (1999) 67–74.
[7] P.K. Chan, S.J. Stolfo, Experiments in multistrategy learning by meta-learning, in: Proceedings of the Second International Conference on Information and Knowledge Management, 1993, pp. 314–323.
[8] E. Corchado, A. Abraham, A.C.P.L.F. de Carvalho, Hybrid intelligent algorithms and applications, Information Science (14) (2010) 2633–2634.
[9] E. Czogala, J. Leski, Fuzzy and Neuro-fuzzy Intelligent Systems, Physica-Verlag, Heidelberg, 2000, pp. 107–127.
[10] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, P. Tan, Data mining for network intrusion detection, in: Proceeding NSF Workshop on Next Generation Data Mining, 2002, pp. 21–30.
[11] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., Wiley-Interscience, 2000, pp. 517–601.
[12] K. Fox, R. Henning, J. Reed, R. Simonian, A neural network approach towards intrusion detection, in: Proceedings of the 13th National Computer Security Conference, 1990, pp. 125–134.
[13] A.K. Ghosh, A. Schwartzbard, A study in using neural networks for anomaly and misuse detection, in: Proceedings of USENIX Security Symposium, 1999, p. 12.
[14] L. Girardin, An eye on network intruder-administrator shootouts, in: Proceedings of the Workshop on Intrusion Detection and Network Monitoring, 1999, pp. 19–28.
[15] J. Han, M. Kamber, in: Data Mining: Concepts and Techniques, Morgan Kaufmann, 2000, pp. 335–385.
[16] S. Hettich, S.D. Bay, The UCI KDD archive, 1999, 〈http://kdd.ics.uci.edu〉, University of California Department of Information and Computer Science, Irvine, CA, 2004, p. 1.
[17] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Computing Surveys 31 (3) (1999) 264–323.
[18] S.C. Lee, D.V. Heinbuch, Training a neural-network based intrusion detector to recognize novel attacks, IEEE Transactions on Systems, Man & Cybernetics Part A Systems & Humans 31 (4) (2001) 294–299.
[19] W. Lee, S.J. Stolfo, Data mining approaches for intrusion detection, in: SSYM'98: Proceedings of the 7th Conference on USENIX Security Symposium, 1998, p. 6.
[20] J.Z. Lei, A. Ghorbani, Network intrusion detection using an improved competitive learning neural network, in: Second Annual Conference on Communication Networks and Services Research, 2004, pp. 190–197.
[21] E. Leon, O. Nasraoui, J. Gomez, Network intrusion detection using genetic clustering, in: Genetic and Evolutionary Computation, vol. 3103/2004, 2004, pp. 1312–1313.
[22] R.P. Lippmann, R.K. Cunningham, Improving intrusion detection performance using keyword selection and neural networks, Computer Networks 34 (4) (2000) 597–603.
[23] Á. Herrero, E. Corchado, M.A. Pellicer, A. Abraham, MOVIH-IDS: a mobile visualization hybrid intrusion detection system, Neurocomputing (13–15) (2009) 2775–2784.
[24] S. Mukkamala, A. Sung, A. Abraham, Cyber security challenges: designing efficient intrusion detection systems and antivirus tools, in: V. Rao Vemuri, V. Sree Hari Rao (Eds.), Enhancing Computer Security with Smart Technology, CRC Press, USA, 2005.
[25] B.C. Rhodes, J.A. Mahaffey, J.D. Cannady, Multiple self-organizing maps for intrusion detection, in: Proceedings of the 23rd National Information Systems Security Conference, 2000.
[26] E. Skoudis, Counter Hack: A Step-by-step Guide to Computer Attacks and Effective Defenses, 2002.

John Lei is a senior analytic scientist and an analytic platform manager at Vesta Corporation in Portland, Oregon, USA. His research focuses on the development of data mining, machine learning, and statistical modeling techniques for fraud detection and business intelligent in challenging real-world application contexts. John has an M.S. in Computer Science from the University of New Brunswick in Canada and an M.S. in Arts in Economics from Guangxi University in China. John received his B.S. in Computer Science from The University of Electronic Science and Technology of China in 1990.

Ali Ghorbani has held a variety of positions in academia for the past 29 years including heading up project and research groups and as department chair, director of computing services, director of extended learning and as assistant dean. He received his Ph.D. and Master's in Computer Science from the University of New Brunswick, and the George Washington University, Washington, DC, USA, respectively. Dr. Ghorbani currently serves as Dean of the Faculty of Computer Science. He holds UNB Research Scholar position. His current research focus is Web Intelligence, Network and Information Security, Complex Adaptive Systems, and Critical Infrastructure Protection. He authored more than 230 reports and research papers in journals and conference proceedings and has edited eight volumes. He served as General Chair and Program Chair/co-Chair for seven International Conferences, and organized over 10 International Workshops. He has also supervised more than 120 research associates, postdoctoral fellows, and undergraduate and graduate students. Dr. Ghorbani is the founding Director of Information Security Centre of Excellence at UNB. He is also the coordinator of the Privacy, Security and Trust (PST) network at UNB. Dr. Ghorbani is the co-Editor-In-Chief of Computational Intelligence, an international journal, and associate editor of the International Journal of Information Technology and Web Engineering and the ISC journal of Information Security. His book, Intrusion detection and Prevention Systems: Concepts and Techniques, published by Springer in October 2009.