A Local Distribution Net for Data Clustering

Qiubao Ouyang, Furao Shen, and Jinxi Zhao

National Key Laboratory for Novel Software Technology,
Nanjing University, China

Abstract. In this paper, a local distribution neural network is proposed for data clustering. This competing network is designed for non-stationary and evolving environment. It represents data by means of neurons (ellipsoids) arranged on a topology map. The local distribution is stored in ellipsoids, while the global topology information is preserving in the relationship between adjacent ellipsoids. With a self-adapting threshold strategy and iteratively learning for information of local distribution, the algorithm is operated in an incremental and on-line way. During implementation, The adopted metric is an improved Mahalanobis distance which considers the local distribution and implies the anisotropy on different vector basis. Hence it can be interpreted as an incremental version of Gaussian mixture model. Experiments both on artificial data and real-world data are carried out to show the performance of the proposed method.

Keywords: self-organizing, covariance matrix, Mahalanobis distance, incremental learning, on-line learning.

1 Introduction

A self-organizing map (SOM) [1] was trained in an unsupervised learning way to produce a low-dimensional (typically two-dimensional) space. It used a neighborhood function between neurons to preserve the topological properties of the input space. However, the pre-determined neuron number and structure exposed itself to criticism. In the evolution of SOM, Teuvo Kohonen introduced another neural network named Adaptive-subspace self-organizing map (ASSOM) [2]. Different from SOM, each neuron of ASSOM was associated with a subspace. When placing a vector from data space, the projection error was obtained between subspaces (neurons) and vector, then the winning neuron was determined. After that, the extension combining the subspace and center bias was proposed. A principal components analysis self-organizing map (PCASOM) [3] was an alternative for this extension. It stored the covariance matrix, replacing the orthonormal vector basis, to represent the subspace. This was directly derived from statistic theory and has advantages in computation burden and reliability of the result. These two models contained more information in each neuron than SOM. But the critical defect in predefined size and structure was inherited. For this reason, ASSOM and PCASOM needed more priori knowledge before training procedure and could not handle the incremental learning.

P. Anthony, M. Ishizuka, and D. Lukose (Eds.): PRICAI 2012, LNAI 7458, pp. 411–422, 2012.

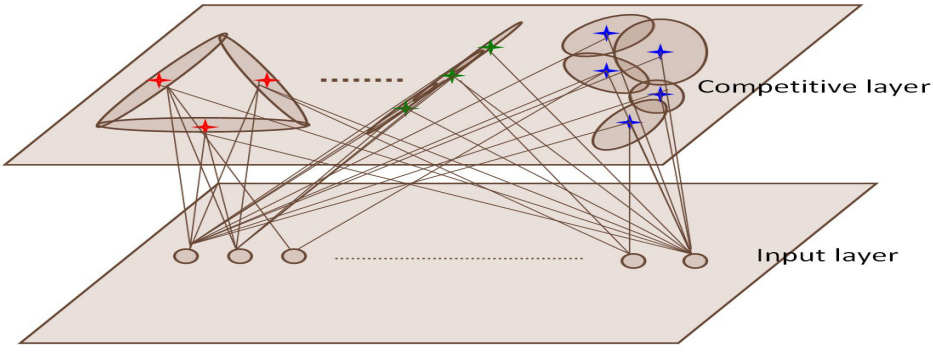


Fig. 1. Structure of Local-SOINN. The input layer accepts input data. The competitive layer dynamically generates and modifies ellipsoids (neurons) with the stimulation of the input data, and finally outputs ellipsoids to represent the initial data and cluster automatically.

A self-organizing incremental neural network (SOINN) [4] was adequate to realize the incremental learning just as its name implied. Incremental learning is a task to break away from the Stability-Plasticity Dilemma [7], which means how to adapt to new information without corrupting or forgetting previously learned information. Adopting a similarity threshold and a locally accumulated error-based insertion criterion, the system can grow incrementally and accommodate input data of non-stationary distribution. SOINN was a competition network and could deal with incremental and on-line learning task without prior knowledge such as how many classes exist.

In this paper, we propose an algorithm called local distribution learning self-organizing incremental neural network which combines the advantages of matrix learning and SOINN. Each neuron is associated with its mean vector and covariance matrix, thus the data distribution can be represented by a collection of local PCA units [9][10] or ellipsoids. Using a self-adapting threshold strategy, the model can be implemented in an incremental learning way even priori knowledge is utterly ignorant. Then we extend the Mahalanobis distance to a more general metric and give interpretation about the relation between Local-SOINN, local PCA and Gaussian Mixture Models. Some analysis of mathematical statistics are also presented to support the stability and availability of the proposed algorithm.

2 The Local-SOINN

2.1 Structure of Local-SOINN

The structure of Local-SOINN is similar to SOM, a competition network inspired by the study of biology. As Figure 1 show, there are two layers in Local-SOINN,

one is the input layer, and the other is competitive layer. In input layer, the number of neurons rest with the dimension of the input data. In competitive layer, the number of neurons is not predetermined, but automatically obtained with the training of specified data set.

The information of competitive layer is stored in two data structure: neuron set and adjacency list. Neuron set preserve the neuron information, each neuron is associated with a 3-tuple $< n, c, M >$, where n, c, M respectively represent the number, mean vector and covariance matrix of samples that assigned to this neuron. The neuron, visualized in the input space, can be described as an ellipsoid: $\{x|\sqrt{(x-c)^T M^{-1}(x-c)} < H\}$. Adjacency list save the topological relation among neurons. If two neurons (ellipsoids) overlap each other, a connection (edge) between the two neurons is created. Each connection is expressed by a 2-tuple $< id_1, id_2 >$, where id_1, id_2 respectively refer to the identity of the origin and destination neuron. These two data structure can alter dynamically with the proceeding of the proposed model, which are convenient for incremental and on-line learning.

In the next part, we detail the procedure of Local-SOINN model. The following notations are used throughout the paper except where otherwise noted. The input data set is $X = \{x_i|x_i \in R^d, i = 1, 2 \ldots t\}$, where d is the dimension of sample. The neuron set is $U = \{u^i|i = 1, 2, \ldots s\}$. Neuron u^i is associated with a 3-tuple $< n_i, c_i, M_i >$.

2.2 Algorithm of Local-SOINN

Figure 2 demonstrate the algorithm flow concisely. Initially, the neural network is empty; there is no neuron in the competitive layer. Then samples are fed into the network sequentially. When a new sample is input into the network, we obtain a set, of which the neurons contain that sample. If this set is empty, i.e., there is no neuron (ellipsoid) contain that sample, a new neuron is added in and the network get a geometrical growth. Else, acquire the winner neuron, update its information and the adjacency list, then detect whether satisfy the condition of merging with neighbors. When the result is expected to output, some post processing like denoising is implemented, and the available ellipsoids are clustered on the basis of neighbor relationship at last.

Winner Lookup and Geometrical Growth Criterion. The proposed model is based on competitive learning. The measure is an improved Mahalanobis distance. At the same time, we take a geometrical growth criterion to ensure the algorithm executed in an incremental manner. That is necessary because we can't provide the intact distribution of online-data in advance. Thus we should decide whether to create a new neuron. Assuming a new sample x access, the winner neuron should be determined. We use $D_i(x)$ to note the distance between x and neuron u^i. Set $S = \{u^i|D_i(x) < H\}$ means that sample x is located in these neurons.

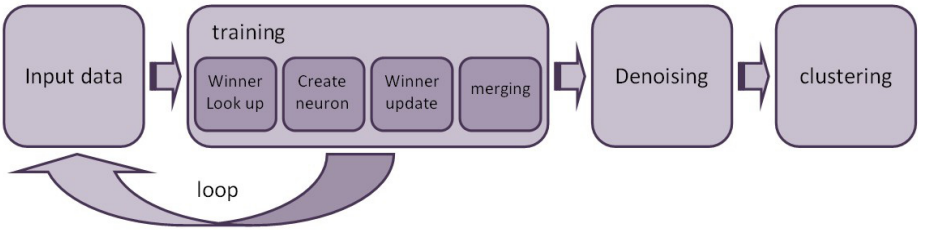


Fig. 2. Flow process of Local-SOINN

If $S = \emptyset$, a new neuron u is created and $U = U \bigcup \{u\}$. Neuron u is initialized with:

$$n = 2d, c = x, M = \frac{h^2}{d} I \tag{1}$$

where I is the identity matrix. The initialized neuron can be regarded as an ellipsoid include $2d$ samples. These samples have a h bias from x on d coordinate directions respectively. That can be represented as: $\{s|s = x \pm h\epsilon_i \text{ for } i = 1, 2, \ldots, d\}$ where $\epsilon_1 = (1, 0, 0, \ldots, 0), \epsilon_2 = (0, 1, 0, \ldots, 0), \cdots, \epsilon_d = (0, 0, 0, \ldots, 1)$.

Else if $S \neq \emptyset$, we select a winner neuron from set S. The winner neuron u^i is the neuron which satisfy: $u^i = \underset{u^i \in S}{\operatorname{argmin}} D_i(x)$. After finding out the winner neuron, we update the relevant record of neuron set and adjacency list.

Iterative Update of Winner Neuron. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n samples, where $x_i \in R^d, i = 1 \ldots n$, belong to the same neuron u. Let c and M be the mean vector and covariance matrix of this data set.In case of online learning, the data are discarded after being learned. Thus, the computation of c and M must be in a recursive way. If x_{n+1} is the new sample assigned to neuron u, the relation between the old mean of dataset and the new can be present as follow:

$$c_{new} = c_{old} + (x_{n+1} - c_{old})/(n + 1) \tag{2}$$

Likewise, the covariance matrix M can be written in the form of recursive relation:

$$\begin{aligned} M_{new} = M_{old} + [n(x_{n+1} - c_{old})(x_{n+1} - c_{old})^T \\ - (n + 1)M_{old}]/(n + 1)^2 \end{aligned} \tag{3}$$

After updating the neuron information, we handle the update of neighbourship. According to the definition of set S, we know any two neurons of S overlap each other. Thus, we add new connections $\{< u_i, u_j > | u_i \in S \wedge u_j \in S \wedge i \neq j\}$ into the adjacency list.

Merging Strategy. Since the new neurons can be automatically added into the network, there may be some number of redundant neurons. The redundant neurons are the neurons that close to each other and having some common principal components. The merging strategy for these redundant neurons should be included in the learning algorithm in order to reduce these redundant neurons. At merging stage, two neurons will merge if two conditions are satisfied: two neurons are connected by an edge; the volume of the combined neuron is not more than the sum of the two neurons that prior to the combination. Considering the combination of neuron $u_1 < n_1, c_1, M_1 >$ and $u_2 < n_2, c_2, M_2 >$, the formula is stated as follow:

$$\begin{aligned} n_{new} &= n_1 + n_2 \\ c_{new} &= (n_1 * c_1 + n_2 * c_2)/n_{new} \\ M_{new} &= \tfrac{n_1}{n_{new}}(M_1 + (c_{new} - c_1)(c_{new} - c_1)^T) \\ &\quad + \tfrac{n_2}{n_{new}}(M_2 + (c_{new} - c_2)(c_{new} - c_2)^T) \end{aligned} \tag{4}$$

Meanwhile, the adjacency list should be updated. We use E_1 and E_2 to signify neighbors of u_1 and u_2 respectively. Then we carry out formula $E_{new} = E_1 \bigcup E_2$ to get the neighbor set of new neuron.

Denoising Procedure. The initial dataset may involve in the pollution of noise. Many neurons are created for these noises. They occupy the network and consume a great quantity of resource. In that case, a denoiser is absolutely necessary. Since the dominated area and sample number of a neuron dominate are acquired, a measure based on density is implemented. As stated above, an ellipsoid Ω is defined as: $\{x|\sqrt{(x-c)^T M^{-1}(x-c)} < H\}$. If M is nonsingular, the volume of this ellipsoid can be figured out using formula:

$$Volume(\Omega) = 2^{[\frac{d+1}{2}]}\pi^{[\frac{d}{2}]}\left(\prod_{i=0}^{[d/2]-1}\frac{1}{d-2*i}\right)\sqrt{|M|}H^d \tag{5}$$

After obtaining the density of each ellipsoid, we can determine a threshold, such as the average density, to discriminate noise neurons from normal neuron.

Clustering Based on the Topotaxy of Neurons. The result of proposed model can ideally simulate the distribution of original data. More neurons are produced at complicated and diversified area. On the contrary, less neurons are generated to represent the simple and unitary local region. The neighbourship relation reflects the topological relationship among local areas. It gives an expression of the distribution at holistic level. With the expression of the whole distribution and the description of local information, our algorithm precisely reconstructs the original data in a condensed way.

Using the information of neighbourship relation among neurons, the neurons can be clustered into different classes. Assuming the neurons and connections are vertices and edges in a undirected graph, the task can be described as a problem of graph partition. A graph G is an ordered pair $< V, E >$ comprising a set V of vertices or nodes together with a set E of edges. A partition of V is obtained regarded to the connective relationship $\sim$ among vertices. The mathematical form can be expressed as followed:

$$V/\sim = \{V_1, V_2, \ldots, V_k\} \tag{6}$$

where V_i is an equivalence class, for $i = 1, 2, \ldots, k$. Each equivalence class represent one cluster and the entire data contain k clusters in all.

3 Discussion and Implementation Issues

3.1 Extension of Mahalanobis Distance

Mahalanobis distance [13] is named from P. C. Mahalanobis. It is defined as follow:

$$D_M(x) = \sqrt{(x-c)^T M^{-1}(x-c)} \tag{7}$$

where M is a $d \times d$ covariance matrix and assumed to be nonsingular, c is the mean vector. Mahalanobis distance is an extension of Euclidean distance. Let the M be unit matrix, Mahalanobis distance degenerates into Euclidean distance. Further more, the Mahalanobis distance takes into account the correlation in the data, since it is calculated using the inverse of the covariance matrix of the data set.

When the investigated data are measured over a large number of variables (high dimensions), they can contain much redundant or correlated information. This so-called multicollinearity in the data leads to a singular or nearly singular covariance matrix that cannot be inverted [13]. In that case, the Mahalanobis distance is disabled and the proposed model get into trouble. So we should expand the definition of Mahalanobis distance.

As Singular Value Decomposition (SVD) formulate, a real symmetric positive semi-definite matrix M can be decomposed:

$$M = E^T \Sigma E \tag{8}$$

where E is orthogonal matrix and $\Sigma = diag(\lambda_1, \lambda_2, \ldots, \lambda_d), \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k > 0 = \lambda_{k+1} = \ldots = \lambda_d$. if $k < d$, the covariance matrix M is singular, the Mahalanobis distance is impracticable. Moreover, the noise and machine error may give rise to minor components which reflected in the smaller eigenvalues and corresponding eigenvectors. Similar to PCA, we can get a truncation of all eigenvalues, the minor eigenvalues are considered as noise or machine error and are

cut off. Thus, predetermining a scaling factor ρ, we can obtain an approximate formula:

$$D_M^2(x) = \underbrace{(E(x-c))^T \begin{pmatrix} \frac{1}{\lambda_1} \cdots 0 & 0 \cdots 0 \\ \vdots \ddots \vdots & \vdots \ddots \vdots \\ 0 \cdots \frac{1}{\lambda_t} & 0 \cdots 0 \\ 0 \cdots 0 & 0 \cdots 0 \\ \vdots \ddots \vdots & \vdots \ddots \vdots \\ 0 \cdots 0 & 0 \cdots 0 \end{pmatrix} (E(x-c))}_{\text{An approximation of Mahalanobis Distance}} + \underbrace{\frac{1}{\zeta}\|(x-c) - \sum_{i=1}^{t} e_i^T(x-c)e_i\|}_{Reconstructionerror} \tag{9}$$

where $t = \underset{1 \le t \le d}{\operatorname{argmin}} \sum_{i=1}^{t} \lambda_i \ge \rho \sum_{i=1}^{d} \lambda_i, \zeta = \min(\lambda_{t+1}, (1-\rho)\sum_{i=1}^{d} \lambda_i)$. It indicate that our distance measurement take two facts together, one is the Mahalanobis distance and the other is reconstruction error. Formula (9) can be showed in a more expressive form:

$$D_M^2(x) = \sum_{i=1}^{t} \frac{(e_i^T(x-c))^2}{\lambda_i} + \frac{1}{\zeta} \sum_{i=t+1}^{d} (e_i^T(x-c))^2 \tag{10}$$

this is just to make clear about the anisotropy of different directions. Bias on minor components bring large contribution to the distance. On the contrary, bias on principal components bring small contribution. That makes a lot more sense than Euclidean distance.

3.2 Statistical Interpretation

Gaussian distribution has a density function:

$$f(x) = (2\pi)^{-d/2}|M|^{-1/2} exp^{-\frac{1}{2}(x-c)^T M^{-1}(x-c)} \tag{11}$$

The central limit theorem claim that the probability density for the sum of d independent and identically distributed variables tend to a Gaussian distribution. It is one of the most important and useful distribution. In recent decades, it is widely applied to data mining to simulate the input distribution. Gaussian Mixture Models (GMM) is one implementation of these applications. Mixture models are able to approximate arbitrarily complex probability density functions. This fact make them an excellent choice for representing distribution of complex input data [14].

Take node of that, the formula of Mahalanobis distance and Gaussian density function all contain the covariance matrix M and mean vector c, which

respectively refer to the unbiased estimations of expectation and variance in mathematic statistics. Each neuron contains plenty of statistical information of the original data. It stores the first-moment and second-moment in the mean vector and covariance matrix. The information is more detailed than that in SOM or SOINN, in which the neurons only be associated with mean vector. Thus our model loses less information of original data and represents the input data more ideally. In the sense of statistics theory, the common ground between Mahalanobis distance and Gaussian distribution imply Local-SOINN can be interpreted as a special GMM.

In our model, the ellipsoid (neuron) has a tendency to expand. The ellipsoid explores the density around the area it dominates. It expand itself if the density around is dense. This property is made clear below.

If M is the covariance matrix of data, we can prove that λ_i is the variance on eigenvector e_i, for $i = 1, 2, \ldots, d$. This result is a posteriori estimate of variance for the distribution. From another point of view, assumed samples uniformly distribute in a ellipsoid Ω defined as $\{x|\sqrt{(x-c)^T M^{-1}(x-c)} < H\}$, the prior estimate of variance on eigenvectors of M can be represented as:

$$\begin{aligned} Var(i) &= \frac{1}{Volume(\Omega)} \int_{\Omega} (e_i^T x)^2 \mathrm{d}\Omega \\ &= \frac{H\lambda_i}{d+2} \quad \text{for } i = 1, 2, \ldots, d \end{aligned} \tag{12}$$

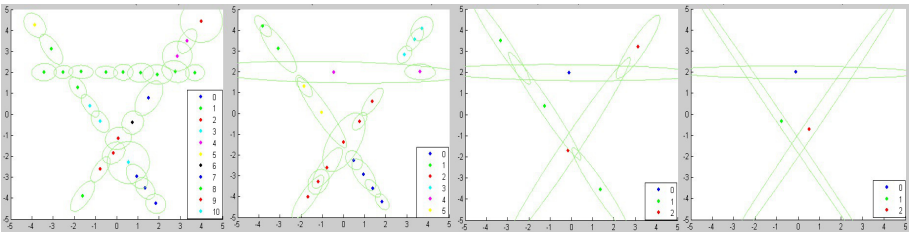
Comparing the two results of prior and posteriori estimate, the ellipsoid is in a dynamic balance if $\lambda_i == Var(i)$, i.e., $H == d + 2$. With $H > d + 2$, the newcome samples adjust the covariance matrix M and have a positive feedback on the size of ellipsoid Ω, and it's the same in reverse.

In the realization of proposed model, we set $H = (1 + 2 * 1.05^{1-n})(d+2)$. This strategy let the ellipsoid has a tendency of expansivity at preliminary stage. The ellipsoid automatically detects the density of around area, and expands itself if the periphery density is similar to the center. With the incremental of samples included in ellipsoid, H approach to $d + 2$, thus the ellipsoid stay a stable state. Anyway, this parameter has a slight effect to our model. There are various alternatives as long as satisfying $H \geq d + 2$ and $\lim_{n\to\infty} H = d + 2$.

4 Computational Experiments

4.1 Simulation with Artificial Data

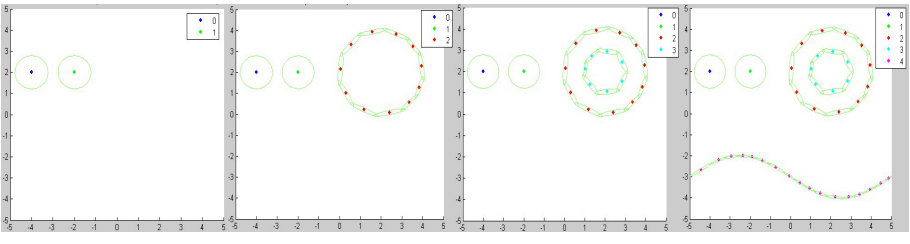
Automatically Clustering Capability Experiment. We test the experiment with a artificial data set which include three belt distributions which overlap each other. With $h^2 = 0.5$ and executed in a unsupervised and on-line model, Figure 3 illustrate local-SOINN can automatically cluster and simultaneously gain the number of classes without knowing the class number in advance. At earlier stage, Local-SOINN generates lots of ellipsoids as the same size as the initial ellipsoid. These ellipsoids just blanket the input data simply, not reflect



(a) $t = 200$ (b) $t = 600$ (c) $t = 1000$ (d) $t = 1500$

Fig. 3. Results at different iterations on the distribution of three overlapping belt. The near and similar neurons combine together gradually.

the correlation of data and local distribution. Then the size and boundary of ellipsoids are self-adjusting according to the input samples. The ellipsoids explore boundary and learn principal components of local area automatically. At the same time, the ellipsoids, which are close to each other and have similar principal components, may merge into a bigger ellipsoid to better display the principal components. Thus the proposed model uses as few ellipsoids as possible to describe the input data and report the categories number at last.



(a) t=40000 (b) t=60000 (c) t=80000 (d) t=100000

Fig. 4. In class-incremental environment, Local-SOINN can learn the new class incrementally

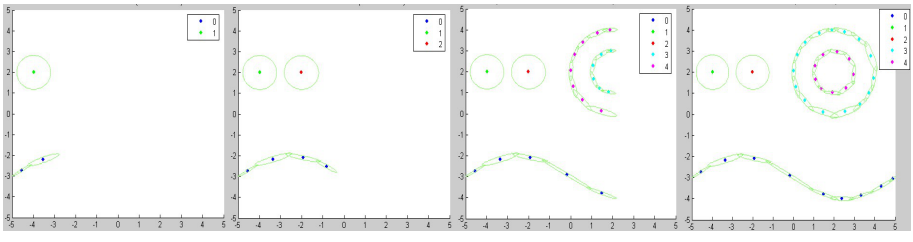
Incremental and On-Line Learning Ability Experiment. In this section, we adopt the artificial dataset used in [4][15]. The original data set include five clusters: two Gaussian distributions, two concentric rings and a sinusoidal curve. In addition, we add 10% noise to estimate the denoising effect. It includes two sub-experiments which are executed in two different data environments.

We first execute experiment in a class-incremental way: first, all the samples of the first Gaussian distribution are input into the net randomly, and then the samples of the second Gaussian distribution, then the big ring and small ring

sequentially, the sinusoidal curve at last. Figure 4 demonstrate the incremental learning ability of Local-SOINN. It can learn the new classes without corrupting or forgetting previously learned classes.

Then, our algorithm is implemented in a more complicated environment: sample incremental and class incremental. New samples and new classes are added to the input data incrementally, the distribution of input data will change in different periods. Figure 5 illustrate that Local-SOINN can effectively simulate the data in incremental way.

Note that, experiments in this section are all run under a on-line model with $h^2 = 0.1$, the samples are discarded after learned. Three results illustrate Local-SOINN can cluster automatically under the complex non-stationary environment even the data is polluted by noise.



(a) t=25000 (b) t=50000 (c) t=75000 (d) t=100000

Fig. 5. In more complicated environment (sample-incremental and class-incremental), Local-SOINN automatically adds in new neurons to simulate the input data

4.2 Simulation Using Real-World Data

Our final set of experiments is devoted to the comparison of the ability to adapt to complex multidimensional data of the ASSOM, PCASOM and Local-SOINN. The real data used are selected from the UCI Repository of Machine Learning Databases and Domain Theories.

In all these experiments, we have split the complete sample set into two disjoint subsets. The training subset has been presented to the networks in the training phase, while the test subset has been used to measure the classification performance of the trained networks. Each of the two subsets have 50% of the samples.

The experiments have been designed as follows. All the samples of the training subset have been presented to a unique network (unsupervised learning). When the training has finished, we have computed the winning neuron for all the samples of the training subset. For every neuron, we have computed its receptive field, i.e. the set of training samples for which this neuron is the winner. Each neuron has been assigned to the class with the most training samples in its receptive field. Finally, we have presented the test samples to the network, one by one. If the winning neuron corresponds with the class of the test sample, we count it as a successful classification. Otherwise, we count it as a classification failure.

Table 1. Classification performance results (the best result is in bold font)

Episodes × episode size	ASSOM		PCASOM		Local-SOINN
	10000 × 10	20000 × 20	20000 × 1	60000 × 1	1-pass-throw
BalanceScale	59.9	57.7	66.3	68.3	**79.2**
Contraceptive	44.8	46.4	42.7	42.6	**48.2**
Glass	52.4	52.4	36.2	36.2	**69.8**
Haberman	73.5	73.8	73.7	71.7	**73.9**
Ionosphere	79.4	74.9	76.1	84.2	**86.9**
PimaIndiansDiabetes	66.4	60.2	65.1	63.5	**70.4**
Yeast	39.8	39.6	38.4	45.3	**52.3**

As stated in [3], the network architecture used in both ASSOM and PCASOM is: a 4x4 rectangular lattice, with two basis vectors per neuron. The neighbourhood function has been always a Gaussian. A linear decay rate of the neighbourhood width σ has been selected in the ordering phase. In the convergence phase $\sigma = 0.04$ was fixed. The structure of Local-SOINN is not predetermined; the competitive layer is empty in initial phase. We use cross validation method to choose the best h^2 in training phase of Local-SOINN.

In the experiments with ASSOM, it is mandatory that the samples are grouped into episodes. On the other hand, the PCASOM does not need any grouping, but samples are repeatedly inputed in PCASOM to obtain fairly acceptable performance. However, the process is 1-pass-throw in Local-SOINN; samples are discarded after trained.

From Table (1), we see that the Local-SOINN model outperforms ASSOM and PCASOM in all the databases. Meanwhile, the training mode determines that the iterations of Local-SOINN are far less than ASSOM and PCASOM.

5 Conclusion

We have proposed a self-organizing incremental neural network model based on the local distribution. The neurons of this network are associated with a mean vector and a covariance matrix, which imply wealthy information of local distribution and lay a solid foundation of statistical theory for Local-SOINN. Our model uses the improved Mahalanobis distance as its measurement. This distance implicitly incorporates the reconstruction error and the anisotropy on different eigenvectors. Further more, the information storage ensure the implementation of iterative update. The denoising and automatic clustering are added in post processing at last. All these specialties make Local-SOINN realize in a incremental and on-line way. Experiments show that the new model has preferable performance both on artificial and real-word data. The future works may include extension to manifold and attributive increment learning.

Acknowledgment. This work was supported in part by the Fund of the National Natural Science Foundation of China (Grant No. 60975047, 61021062), 973 Program (2010CB327903), and Jiangsu NSF grant (BK2009080, BK2011567).

References

1. Kohonen, T.: The self-organizing map. Proceedings of the IEEE 78, 1464–1480 (1990)
2. Kohonen, T.: The adaptive-subspace SOM (ASSOM) and its use for the implementation of invariant feature detection. In: Fogelman-Soulié, F., Galniari, P. (eds.) Proceedings of the ICANN 1995, International Conference on Artificial Neural Networks, Paris: EC2 and Cie, vol. 1, pp. 3–10 (1995)
3. López-Rubio, E., Muñoz-Pérez, J., Gómez-Ruiz, J.A.: A pricipal components analysis self-organizing map. Neural Networks 17, 261–270 (2004)
4. Shen, F., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Networks 19, 90–106 (2006)
5. Arnonkijpanich, B., Hammer, B., Hasenfuss, A.: Local Matrix Adaptation in Topographic Neural Maps
6. Yin, H., Huang, W.: Adaptive nonlinear manifolds and their applications to pattern recognition. Information Sciences 180, 2649–2662 (2010)
7. Carpenter, G.A., Grossberg, S.: The ART of adaptive pattern recognition by a self-organizing neural network. IEEE Computer 21, 77–88 (1988)
8. Kambhatla, N., Leen, T.K.: Dimension reduction by local principal component analysis. Neural Computation 9(7), 1493–1516 (1997)
9. Tipping, M.E., Bishop, C.M.: Mixtures of probabilistic principal component analyzers. Neural Computation 11(2), 443–482 (1999)
10. Weingessel, A., Hornik, K.: Local PCA algorithms. IEEE Trans. Neural Networks 11(6), 1242–1250 (2000)
11. Reilly, L., Cooper, L.N., Elbaum, C.: A Neural Model for Category Learning. Biological Cybernetics 45, 35–41 (1982)
12. Wasserman, P.D.: Advanced Methods in Neural Computing. Van Nostrand Reinhold, New York (1993)
13. Maesschalck, R., Jouan-Rimbaud, D., Massart, D.L.: The Mahalanobis distance. Chemometrics and Intelligent Laboratory Systems 50, 1–18 (2000)
14. Figueiredo, M.A.T., Jain, A.K.: Unsupervised Learning of Finite Mixture Models. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 381–396 (2002)
15. Shen, F., Hasegawa, O.: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks 20, 893–903 (2007)