A principal components analysis self-organizing neural network model and computational experiment

Jifu Nong
College of Science
Guangxi University for Nationalities
Nanning, China
njf93471@163.com

***Abstract*—We propose a new self-organizing neural model that performs principal components analysis. It is also related to the adaptive subspace self-organizing map (ASSOM) network, but its training equations are simpler. Experimental results are reported, which show that the new model has better performance than the ASSOM network.**

Keywords- Self-organization; Principal component analysis; Competitive learning; Pattern recognition

I. INTRODUCTION

The adaptive subspace self-organizing map (ASSOM) is an evolution of the self-organizing feature map (SOFM) proposed by Kohonen. These two networks are rooted in the concept of self-organization, which seems to explain several neural structures of the brain that perform invariant feature detection. The ASSOM was first presented as an invariant feature detector. This property has been further studied, and its relations with wavelets and Gabor filters have been reported. Furthermore, representation of observed data could be also achieved by using a vector basis. A probabilistic algorithm to learn vector basis has been given, while the ASSOM is an alternative for this task.

This network is an alternative to the standard principal component analysis (PCA) algorithms, as it looks for the most relevant features of the input data. An early neural network approach to the PCA problem can be found in Oja. Self organization has been used for this task before. Nevertheless, the ASSOM network is a completely new tool for pattern recognition. In general, unsupervised networks can be used to solve classification and clustering problems. The ASSOM has been successfully applied to the handwritten digit recognition problem which many neural network researchers have addressed. Also, it has been used for texture segmentation. This work is related with a supervised variant of the ASSOM, called Supervised Adaptive-Subspace Self-Organizing Map (SASSOM), first proposed by Ruiz del Solar and Ko¨ppen.

Finally, SOFM networks are adequate to create topographic maps, which are representations of the input space. This ability is inherited by the ASSOM network, which has been taken as a standard for comparison with other algorithms that make these maps.

II. THE ASSOM NETWORK

In this section we give a brief review of the original ASSOM. First of all, we define the orthogonal projection of a vector x on an orthonormal vector basis $B = \{b_h \mid h = 1, 2, \cdots, K\}$ as

$$\hat{x} = \left(\sum_{h=1}^{K} b_h b_h^T \right) x = \sum_{h=1}^{K} (b_h^T x) b_h \tag{1}$$

The vector x can be decomposed into two vectors, the orthogonal projection and the projection error, by the equation

$$x = \hat{x} + \tilde{x} \tag{2}$$

Every unit (say, i) of an ASSOMnetwork has a vector basis, Bi: The vector bases Bi are assumed to be orthonormal. They are orthonormalized frequently to meet this requirement. The aim is to approximate x by its orthogonal projection on the vector basis which gives the minimum projection error (winning neuron). This is achieved by changing the basis vectors of the winning neuron. A learning rate controls this change. The units form a lattice, like in the SOFM network. Hence the neighbouring neurons of the winning neuron also change their basis vectors according to a neighbourhood function which gives the degree of vicinity. It is expected that this strategy achieves a topological ordering of the units, i.e. neighbouring neurons span similar subspaces.

The input vectors are grouped into episodes in order to be presented to the network. So, an episode $S(t)$ has many time instants $t_p \in S(t)$, each with an input vector $x(t_p)$. The set of input vectors of an episode has to be recognized as one class, such that any member of this set and even an arbitrary linear combination of them should have the same winning neuron. The training process has the following steps:

(a) Winner lookup. The winning neuron c is computed according to this equation:

$$c = \arg\max_i \left(\sum_{t_p \in S(t)} \| \hat{x}^i(t_p) \|^2 \right) \tag{3}$$

(b) Learning. For every sample $x(t_p)$, $t_p \in S(t)$:

Published by Atlantis Press, Paris, France.

(b.1) Basis vectors rotation. The basis vectors are changed to approximate the sample more accurately:

$$b_h^i(t+1) = \left(I + \lambda(t) h_C^i(t) \frac{x(t_p) x(t_p)^T}{\| \hat{x}^i(t_p) \| \| x(t_p) \|} \right) b_h^i(t) \quad (4)$$

The learning rate $\lambda(t)$ and the neighbourhood function $h_C(t)$ may have either linear or exponential decays.

(b.2) Dissipation. For every component b_{hj}^i of every basis vector b_h^i:

$$b_{hj}^i{}' = \text{sgn}(b_{hj}^i) \max(0, | b_{hj}^i | - \varepsilon) \quad (5)$$

where

$$\varepsilon = \varepsilon_{hj}^i(t) = \delta | b_{hj}^i(t) - b_{hj}^i(t-1) | \quad (6)$$

The dissipation parameter δ has typically a small value. The dissipation step is needed to remove spurious small components, which may appear in the basis vectors.

(c) Orthonormalization. Orthonormalize every basis. It is not needed to execute this step in every iteration. This is because the learning step usually gives near orthogonal vector bases.

The objective function to minimize considered by Kohonen to obtain Eq. (4) is the average expected spatially weighted normalized squared projection error over the episodes [10]:

$$E = \int \sum_{t_p \in S(t)} \sum_i h_C^i(t_p) \frac{\| \tilde{x}(t_p) \|^2}{\| x(t_p) \|^2} p(X) dX \quad (7)$$

where $p(X)$ is the joint probability density for the samples $x(t_p), t_p \in S(t)$, that produce the Cartesian product set X, and dX means a volume differential in the Cartesian product space of the $x(t_p)$. The Robbins-Monro stochastic approximation is used to minimize Eq. (7), which leads to Eq. (4). Our aim here is to propose a new model, which learns vector subspaces, like the ASSOM, but has a broader capability to represent the input distribution.

III. THE PCASOM MODEL

A. Neuron weights updating

The ASSOM network stores the basis vectors in its units. The weight update involves a rather complicated transformation to obtain a vector basis that is similar to the older one, but closer to the present input samples. Here we propose an alternative way to store the information: we use the covariance matrix. The covariance matrix of an input vector x is defined as

$$R = E[(x - E(x))(x - E(x))^T] \quad (8)$$

where $E(\cdot)$ is the mathematical expectation operator and we suppose that all the components of x are real random variables.

If we have M input samples, $x_1, x_2, \cdots, x_M$, we can make the following approximation:

$$R \approx R^I = \frac{1}{M-1} \sum_{i=1}^{M} (x_i - E(x))(x_i - (x))^T \quad (9)$$

Note that this is the best approximation that we can obtain with this information (it is an unbiased estimator with minimum variance). Now, if we obtain N new input samples, $x_{M+1}, x_{M+2}, \cdots, x_{M+N}$, we may write:

$$M = \frac{1}{N} \sum_{i=M+1}^{M+N} (x_i - E(x))(x_i - E(x))^T \quad (10)$$

$$R \approx R^{II} = \frac{1}{M+N-1} ((M-1)R^I + NM) \quad (11)$$

Both R^I and R^{II} are approximations of R, but R^{II} is more accurate because it takes into account the $N+M$ input samples. Eq. (11) is a method to accumulate the new information to the old information. Now we need to approximate the expectation of the input vector $E(x)$.

Every processing unit of our model stores approximations to the matrix R and the vector $E(x)$. They will use the above algorithm to update these approximations. So, in the time instant t the unit i stores the matrix $R_i(t)$ and the vector $e_i(t)$.

B. Competition among neurons

When an input sample is presented to a self-organizing map, a competition is hold among the neurons. Here we use an adaptation of the ASSOM approach. Remember that the orthogonal projection of a vector x on an orthonormal vector basis $B = \{b_h \mid h = 1, 2, \cdots, K\}$ is given by Eq. (1). The vector x can be decomposed in two vectors, the orthogonal projection and the projection error, by the Eq. (2).

Every unit (say, i) of our network has an associated vector basis $B^i(t)$ at every time instant t: It is formed by the K eigenvectors corresponding to the K largest eigenvalues of $R_i(t)$. This is rooted in the principal components analysis (PCA). Note that $B^i(t)$ must be orthonormalized in order to the system to operate correctly. The difference among input vectors $x^i(t), i = 1, 2, \cdots, N$ and estimated means are projected onto the vector bases of all the neurons. The neuron *c* that has the minimum sum of projection errors is the winner.

$$c = \arg \left(\sum_{i=1}^{N} \| x^i(t) - e_j(t) - Orth(x^i(t)), B^i(t) \|^2 \right) \quad (12)$$

where $\text{Orth}(x, B)$ is the orthogonal projection of vector x on basis B. So, we are looking for the neuron which best represents the inputs $x^i(t)$.

C. Network topology

We consider a topology that defines which neurons are neighbors. This means that our model is a computational map. Typically the neurons form a rectangular lattice. When a neuron c wins the competition it is updated. Its neighbors are also updated, according to the degree of neighborhood π_{ic} between winning neuron c and its neighbor i:

$$\pi_{ic}(t) = \exp\left(-\frac{d_{i,c}^2}{2\sigma^2(t)}\right) \quad (13)$$

In Eq. (13), $d_{j,i}$ is the distance between winning neuron i and neuron j; and π_{ji} is a unimodal function of the lateral distance d_{ij}, called neighborhood function, with $\sigma(t) \to 0$ as $t \to \infty$. The value $\sigma(t)$ controls the neighborhood size.

The degree of neighborhood and the learning rate are combined to control the updating of the neurons. For every neuron i we have

$$e_i(t+1) = e_i(t) + \eta_e(t)\pi_{ic}(t)\left[\frac{1}{N}\sum_{i=1}^{N} x_i - e_i(t)\right] \quad (14)$$

$$R_i^*(t+1) = \frac{1}{N}\sum_{j=1}^{N}(x_j - e_i(t+1))(x_j - e_i(t+1))^T \quad (15)$$

$$R_i(t+1) = R_i(t) + \eta_R(t)\pi_{ic}(t)(R_i^*(t+1) - R_i(t)) \quad (16)$$

The learning process is divided into two phases, like the standard self-organizing map algorithms: the ordering phase and the convergence phase. It is during the ordering phase when the topological ordering of the neurons takes place.

The convergence phase is required principally for the fine-tuning of the computational map. The learning rate is maintained at a small, constant value during this phase.

IV. COMPUTATIONAL EXPERIMENTS

A. Convergence speed experiment

First we have considered the following architecture: a 6×6 lattice with two basis vectors in each neuron. We have trained this network to adapt to three subspaces of R^{10}, each of dimension two. Our goal has been to measure the convergence speed and the final projection error obtained with the ASSOM and the PCASOM. For every episode of the ASSOM, we select randomly one of the three subspaces, and then build 20 samples, which belong to this subspace. The samples have been built by multiplying each of the two basis vectors of the subspace by a random quantity in the open interval (0,1). For PCASOM, we build the samples in the same way, but without grouping them in episodes. In order to make a useful comparison, we choose the same parameters for both models (when possible). Hence, both the ordering phase and the convergence phase take one half of the episodes. The neighborhood function has been always a Gaussian with linear decay rate in the ordering phase (with the neighborhood width s ranging from $\sigma = 4$ to $\sigma \cong 0$). In the convergence phase we fix $\sigma = 0.04$. For the ASSOM model we used a initial learning rate $\lambda(0) = 0.5$, and a dissipation parameter $\delta = 0.2$. For the PCASOM model we used initial learning rates $\eta_e = 1$ and $\eta_R = 1$.

We have selected a performance measure of that is invariant to scalings, rotations and translations of the input data. The relative error for an input vector x is defined as

$$E_\tau(x) = \frac{\min_i \| \tilde{x}^i \|}{\| x \|} \quad (17)$$

where the index *i* runs over all the units of the network, i.e. the projection error norm for the winning neuron, divided by the norm of the input vector. We select this criterion because the winning neuron has the best approximation to the input vector x of the network. Thus, the different algorithms are compared with this measure.

We have run 30000 iterations of the ASSOM and 20000 iterations of PCASOM. The result can be seen in Fig. 1.

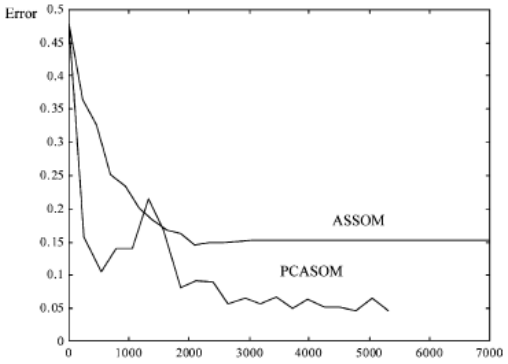

Figure 1. ASSOM vs. PCASOM.

B. Separation capability experiment

As we have said, the PCASOM model has the capability to separate clusters which have the same orientation, but different centroids. We have selected some distributions to demonstrate this ability.

In order to be able to plot the results, which can be seen in Figs.1, we have used *K*=1,*D*=2, as in the previous discussion of the differences between the classic PCA and the PCASOM. We have used mixtures of several Gaussian distributions.

In all the cases we have obtained 500 samples from every Gaussian distribution, i.e. every cluster has 500 samples. These clusters have been selected so that the orientation of some of them is the same. Furthermore, some of the clusters are very close. In fact, there are a few cases of non-separable clusters. We have used elongated clusters so that the neurons could learn their directions. There are clusters with different elongations in the same distribution. The clusters have been marked with gray ellipses. Please note that the used Gaussian distributions do not end sharply, unlike the ellipses plotted.

We have run 20000 iterations with each distribution. We have used only one sample per episode, i.e. there has not

been any grouping by episodes. The parameter selection has been the same as the previous experiment.

We have used four neurons in all the networks, so that the number of units matches the number of clusters, and only one basis vector per neuron. As we can see, every neuron finds a cluster, with its estimated mean in the centroid of the cluster (marked with 'o'), and the principal component direction along the orientation of the cluster (marked with a straight line).

C. UCI benchmark databases experiment

Our final set of experiments is devoted to the comparison of the ability to adapt to complex multidimensional data of the ASSOM and the PCASOM. We have used some classification tasks for this purpose. Several standard benchmark databases have been selected from the UCI Repository of Machine Learning Databases and Domain Theories (Murphy, 2001).

The experiments with the 'Supervised' title (Tables 1) have been designed as follows. First, the training subset has been split by classes. Then the samples of each class have been presented to a different network for training, i.e. we have one network per class. Finally, after the networks have been trained, the samples of the test set have been presented to all the networks, one by one. The network which had the neuron with the less projection error has been declared as the winning network. If the winning network corresponds with the class of the test sample, we count it as a successful classification.

We have use the same network architecture with both ASSOM and PCASOM: a 4×4 rectangular lattice, with two basis vectors per neuron. The neighbourhood function has been always a Gaussian. We have selected a linear decay rate of the neighbourhood width σ in the ordering phase. In the convergence phase we fix $\sigma = 0.04$.

The parameters of both models have been optimized in order to make a useful comparison. The optimal values for the parameters that we have used have been the following. For the ASSOM model: initial learning rate $\lambda(0) = 0.45$, initial neighbourhood width $\sigma(0) = 4$, dissipation parameter $\delta = 0.2$.

The optimal values for the parameters that we have used have been the following. For the ASSOM model: initial learning rate $\lambda(0) = 0.45$, initial neighbourhood width $\sigma(0) = 4$, dissipation parameter $\delta = 0.2$.

We can see that the PCASOM model outperforms ASSOM in the Glass database (with supervised learning) and in the BalanceScale, Ionosphere, Segmentation and Yeast databases (with non-supervised learning). It must be noted that these results are achieved even with 20000 epochs, when the computational effort is much less than ASSOM.

TABLE I. CLASSIFICATION PERFORMANCE RESULTS (SUPERVISED LEARNING)

Episodes	ASSOM (%)		PCASOM (%)	
episode size	10000× 10	20000× 20	20000× 1	60000× 1
BalanceScale	88.46	90.06	68.27	68.97
Contraceptive	57.14	60.41	59.05	58.64
Glass	15.24	25.71	40.84	40.95
Haberman	78.95	81.58	17.76	65.79
Ionosphere	39.89	40.11	28.57	27.43
PimaIndians	72.14	37.76	68.49	67.71
Segmentation	87.62	94.29	48.23	87.62
Yeast	10.38	11.49	11.59	11.83

V. CONCLUSIONS

We have proposed a new self-organizing neural model that performs PCA. It is also related to the ASSOM network, but its training equations are much simpler, and its input representation capability is broader. Furthermore, the grouping of the input samples into episodes is no longer needed. Experimental results have been reported, with an application to classification. A performance measure has been defined in order to make meaningful comparisons. Experiments show that the new model has better performance than the ASSOM network with less computations in a number of benchmark problems.

ACKNOWLEDGMENT

The work was supported by Guangxi department of education scientific research project under Grand No. 201204LX083.

REFERENCES

[1] De Sa, V. R., and Ballard, D. H.. Category learning through, multimodality sensing. Neural Computation, 1998, 10(5), 1097–1117.

[2] Fleuret, F., and Brunet, E.. DEA: an architecture for goal planning and classification. Neural Computation, 2000, 12(9), 1987–2008.

[3] Friedman, J. H.. Exploratory projection pursuit. Journal of American Statistical Association, 1987, 82(397), 249–266.

[4] Fukushima, K.. Self-organization of shift-invariant receptive fields. Neural Networks, 1999, 12(6), 791–802.

[5] Higuchi, I., and Eguchi, S.. The influence function of principal component analysis by self-organizing rule. Neural Computation, 1998, 10(6), 1435–1444.

[6] Hyvarinen, A.. Survey on independent component analysis. Neural Computing Surveys, 1999, 2, 94–128.

[7] Kohonen, T.. The self-organizing map. Proceedings of the IEEE, 1990, 8, 1464–1480.

[8] Kohonen, T.. Emergence of invariant-feature detectors in the adaptive-subspace SOM. Biological Cybernetics, 1996, 75, 281–291.

[9] Kohonen, T., Kaski, S., and Lappalainen, H.. Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM. Neural Computation, 1997, 9(6), 1321–1344.

[10] Lewicki, M. S., and Sejnowski, T. J.. Learning overcomplete representations. Neural Computation, 2000, 12(2), 337–365.