35

Forced Accretion and Assimilation Based on Self-Organizing Neural Network

Cheng-Yuan Liou and Wei-Chen Cheng
Department of Computer Science and Information Engineering
National Taiwan University, Taiwan
Republic of China

1. Introduction

The high level abstraction is developed layer after layer. This abstraction and generalization are important in language evolution and the study of mental processes. This chapter presents a self-organizing neural network based on cascading a series of layered perceptrons that resemble the development of the deep layers of the brain. This neural network is named 'SOM perceptron'. It maps all received patterns in a single class into a point in the deep layer space and maps different classes into different points. These widely separated class points can facilitate the abstract categorization and analogy operated in the mental process. Categorizing similar objects into small categories and different objects into different small categories can simplify a problem and reduce the mental load. Those small categories that contain similar objects will be successively combined into large categories to further reduce the load. This categorization process is a kind of divide and conquer process. This work emulates such process in successive layers. This design also resembles the formation of the corlica sensory representation. It preserves the topological relations among classes. It constructs a layered feed-forward network. Each layer attempts to split different classes and concentrate the same class patterns. It is trained using the discrimination differences between classes and trained independently layer after layer in a bottom-up manner. The class labels are not used in the training process to relieve the loading complexity of weights.

A bottom-up training method for the construction of the SOM perceptron Liou et al. (2000); Liou & Cheng (2008) is introduced in this chapter. This constructed SOM perceptron can be used as the mapping function to split different classes. It can be applied to split multiple classes.

The construction ideas are discussed in this section. There are four issues concerning the method. The first issue is the reduction of the number of hidden representations. The second one discusses a design method for the weights, such that the accomplishment of the SOM perceptron is guaranteed. The third one is why we prefer a bottom-up construction for the MLP and why we do not apply the BP to training it. The fourth issue discusses the different functions of front layers and rear layers of MLP under the BP training. These four issues will be presented in four subsections in this section. The method and architecture of the SOM perceptron are described in the next section. Experiments are included in the third section. Several related works are discussed in the last section.

We start with an introduction of the hidden representation. Let the set of all patterns be $X = \{\mathbf{x}^p, p = 1, \ldots, P\}$. Each pattern $\mathbf{x}^p$ is a D-dimensional column vector. The label function,

$C : R^D \rightarrow N$, maps each pattern, $\mathbf{x}^p$, to its class identity number, c_p. Let the set U_{c_i} contain all pattern pairs that belong to the same class c_i,

$$U_{c_i} = \{(\mathbf{x}^p, \mathbf{x}^q); C(\mathbf{x}^p) = C(\mathbf{x}^q) = c_i\}. \quad (1)$$

Let the set V_{c_i,c_j} contain all pattern pairs that belong to different classes,

$$V_{c_i,c_j} = \left\{(\mathbf{x}^p, \mathbf{x}^q); C(\mathbf{x}^p) = c_i, C(\mathbf{x}^q) = c_j, c_i \neq c_j\right\}. \quad (2)$$

Suppose there are L hidden layers in the network, $\{m = 1,2,\ldots,L\}$. Let the column vector $\mathbf{y}^{(p,m)}$ be the output vector of all neurons in the mth layer when the pattern $\mathbf{x}^p$ is fed to the input layer. $\mathbf{y}^{(p,m)}$ is the hidden (or internal) representation of $\mathbf{x}^p$ in the mth layer. We set $\mathbf{y}^{(p,0)} = \mathbf{x}^p$ for $m = 0$. Let n_m denote the total number of neurons in the mth layer. The collection of all hidden representations (HRs) of the mth layer is

$$\mathbf{Y}^m = \left\{\mathbf{y}^{(p,m)}, p = 1,\ldots,P\right\}.$$

A HR may be the same for different patterns, that is, a many-to-one mapping, $\mathbf{y}^{(p,m)} = \mathbf{y}^{(q,m)}$ for $\mathbf{x}^p \neq \mathbf{x}^q$. Let $\|\mathbf{Y}^m\|$ be the total number of distinct HRs in the set $\mathbf{Y}^m$.

1.1 First issue: reduced number of hidden representations

The HRs, $\mathbf{Y}^m$, have been studied in Liou & Yu (1995). All patterns have their HRs in each layer. These HRs are the output vectors of the hidden layers for all input patterns. They are all binary codes when the hard-limiting activation function is applied to all neurons. So, $\mathbf{y}^{(p,m)}$ is a binary code. The n_m decision hyperplanes of the neurons in the mth layer divide the n_{m-1} dimensional space of the preceding $(m-1)$th layer into nonoverlapped small decision areas and code these areas with binary codes. Let A^m be the collection of all areas, $A^m = \{\mathbf{a}_i^m, i = 1\ldots\|A^m\|\}$. The total number of these areas is less than or equal to $\Sigma_{k=0}^{n_{m-1}}\binom{n_m}{k}$, $\|A^m\| \leq \Sigma_{k=0}^{n_{m-1}}\binom{n_m}{k}$ Mirchandini & Cao (1989). Each area has a convex polyhedral shape.

Each of the codes, $\mathbf{y}^{(p,m)} \equiv \mathbf{a}_i^m$, represents all patterns (or HRs) contained in a single divided polyhedral area, $\{\mathbf{y}^{(p,m-1)}; \mathbf{y}^{(p,m-1)} \in \mathbf{a}_i^m\}$. This could be a many-to-one mapping. Each area is a small category containing many similar patterns. Note that certain areas may not contain any representation, $\|A^m\| \geq \|\mathbf{Y}^m\|$. According to the study in Liou & Yu (1995), the total number of HRs will be much reduced, generally, in a layer that is far from the input layer. This means

$$\left\|\mathbf{Y}^L\right\| \ll \ldots \ll \left\|\mathbf{Y}^2\right\| \ll \left\|\mathbf{Y}^1\right\| \ll P.$$

The reduced number in the mth layer, $\left\|Y^{m-1}\right\| - \|Y^m\|$, is the pattern complexity resolved in the mth layer. In many cases, $\left\|Y^{m-1}\right\| >> \|Y^m\|$. This reduction is very useful for the abstraction and isolation of the whole class patterns. Ideally, this number can be reduced to the number of classes, $\left\|\mathbf{Y}^L\right\| = \text{'}total\ number\ of\ classes\text{'} = \|C\|$. This makes the design possible for the SOM perceptron. Figure 1 illustrates the design idea.

1.2 Second issue: constructive weight design for SOM perceptron

The method in Liou & Yu (1994) provided a weight design for each layer, see program in web Liou (2000a). This design can be used in the SOM perceptron. This guarantees the accomplishment of the SOM perceptron. According to the design, the upper bound of

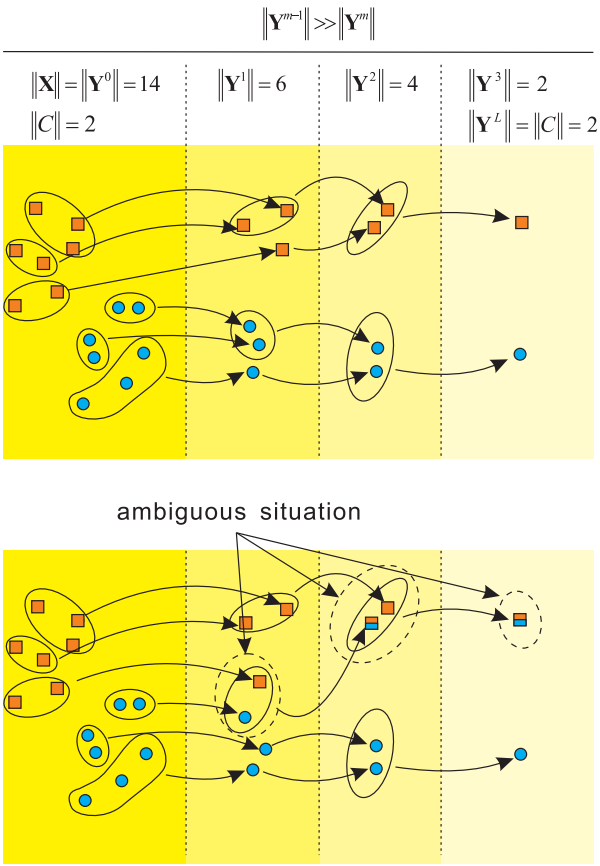


Fig. 1. The top diagram plots the reduction and the bottom plots an ambiguity representation in the second hidden layer.

the number of neurons in the mth layer required for solving a general-position two-class classification problem is

$$\left\lceil \frac{\|\mathbf{Y}^{m-1}\|}{n_{m-1}} \right\rceil \geq n_m.$$

For the number of neurons in the first hidden layer, n_1, the bound is $\left\lceil \frac{P}{D} \right\rceil \geq n_1$. With this weight design, the reduced number in the last layer L is guaranteed, that is $\|\mathbf{Y}^L\| = \|C\|$.
The method in Liou & Yu (1994) also showed a constructive weight design for the MLP. To illustrate this method, we show a general-position two-class classification problem. This problem can be solved with two hidden layers. This construction is very different from BP algorithms. It solves the complexity, $\Sigma_{k=0}^{n_{m-1}} \binom{n_m}{k}$, in the succeeding MLP layers.
Figure 2(a) illustrates the design for a two-class problem, $c_1 = 1$ and $c_2 = 2$, in a two dimensional space, $D = n_0 = 2$. In this D space, a center line of a strip, $\overline{\mathbf{x}^p \mathbf{x}^q}$, is allocated for two near patterns, $\mathbf{x}^p$ and $\mathbf{x}^q$, that are in a same class c_1, $\mathbf{x}^p \in c_1$ and $\mathbf{x}^q \in c_1$. We assume

that c_1 contains fewer number of patterns than that of c_2. Then, this center line is split into two parallel lines, line $\overline{a}$ and line $\overline{b}$. They are in the two opposite sides of this center line and parallel to the center line, $\overline{a} \| \overline{\mathbf{x}^p \mathbf{x}^q} \| \overline{b}$. For $\overline{a}$, pick a pattern $\mathbf{x}^r$, $\mathbf{x}^r \in c_2$, where $\mathbf{x}^r$ is closest to $\overline{\mathbf{x}^p \mathbf{x}^q}$. $\mathbf{x}^r$ and $\overline{a}$ are in the same side of $\overline{\mathbf{x}^p \mathbf{x}^q}$. Plot a parallel line $\overline{a}^r$, $\overline{a}^r \| \overline{\mathbf{x}^p \mathbf{x}^q}$, that passes the pattern $\mathbf{x}^r$. Pick a pattern $\mathbf{x}^s$, $\mathbf{x}^s \in c_1$, that is in between the two lines, $\overline{a}^r$ and $\overline{\mathbf{x}^p \mathbf{x}^q}$, and is the closest pattern to the line $\overline{a}^r$. Plot a parallel line, $\overline{a}^s$, $\overline{a}^s \| \overline{\mathbf{x}^p \mathbf{x}^q}$, that passes the pattern $\mathbf{x}^s$. Plot a decision border line, $\overline{a}^{rs}$, right in between the two parallel lines, $\overline{a}^r$ and $\overline{a}^s$. $\overline{a}^{rs}$ has wide margin between the pair $(\mathbf{x}^r, \mathbf{x}^s)$.

The two patterns, $\mathbf{x}^r$ and $\mathbf{x}^s$, serve as the margin-limiting stops of the region in between the two lines, $\overline{a}^r$ and $\overline{a}^s$. The decision border line $\overline{b}^{uv}$ for $\overline{b}$ can be accomplished in a similar way on the other side of $\overline{\mathbf{x}^p \mathbf{x}^q}$. These two decision lines are used as the two neurons in the mth layer. All patterns in between the two lines $\overline{a}^{rs}$ and $\overline{b}^{uv}$ belong to the same class c_1. They are well isolated from the patterns in the other class c_2. The stops $\mathbf{x}^r$ and $\mathbf{x}^s$ could be different from the support vectors in support vector machine (SVM) Boser et al. (1992).

The number of patterns in between the two lines $\overline{a}^{rs}$ and $\overline{b}^{uv}$ are one of the two factors of the two neurons. The other factor is the width between these two decision lines. These two factors are useful in the determination of the significance of these two neurons. Those neurons with large factors are preferable and will be preserved with high priority in many training operations. Small factor neurons will be eliminated occasionally.

An example of the typical decision regions is illustrated in Figure 2(b). The decision regions contain four bar-like strips. There exists physiological evidences on receptive fields, $D = 2$, for such bar-like strips, Daugman (1980); Dobbins et al. (1987).

There are many techniques to pick the center patterns $\mathbf{x}^p$ and $\mathbf{x}^q$ to build a strip. One way to do this is to select all patterns, $\{\mathbf{x}^p, \mathbf{x}^q, \mathbf{x}^r, \mathbf{x}^s, \mathbf{x}^u$ and $\mathbf{x}^v\}$, in a predefined neighborhood region. The size of this region can be tuned during the training process. One may include a penalty cost to set the borders $\overline{a}^{rs}$ and $\overline{b}^{uv}$ in a way similar to that for SVM.

Note that both the number of neurons and the number of layers in certain operations of the tiling algorithm [Mezard and Nadal, 1989] are highly sensitive to the setting of the origin, the absolute coordinates, of the patterns. Many other neural networks, such as SOM, are also sensitive to the setting of the origin. The relative distances among patterns are used in the method in Liou & Yu (1994). This relative distance gives the classification quality. To our knowledge, the performance of this method exceeds that of SVM. The relative distances will be used in this work.

Figure 2(c) illustrates an example for the general-position two-class classification problem. A single 'AND' function is used for a neuron in the second hidden layer to represent the patterns in one individual strip. A global 'OR' function is used in the output neuron to represent all patterns in class c_1 that are in all strips. To our knowledge, this is the simplest MLP architecture in many aspects. Any difficult isolated polyhedral region that is encompassed by several neurons can be included in this archietecture by adding an extra 'OR' weight connecting this region. The performance of this simple architecture exceeds that of tiling algorithm.

1.3 Third issue: front layers must be trained correctly to get perfect performance for MLP

The reason why we prefer a constructive way from the bottom layer is based on the work Liou & Yu (1995). It introduced a layered binary tree, named 'AIR' tree, that can trace the error neurons in a latent hidden layer that is far from the output layer and close to the input layer, see program in Liou (2000b). The error shows that certain mixed patterns from both

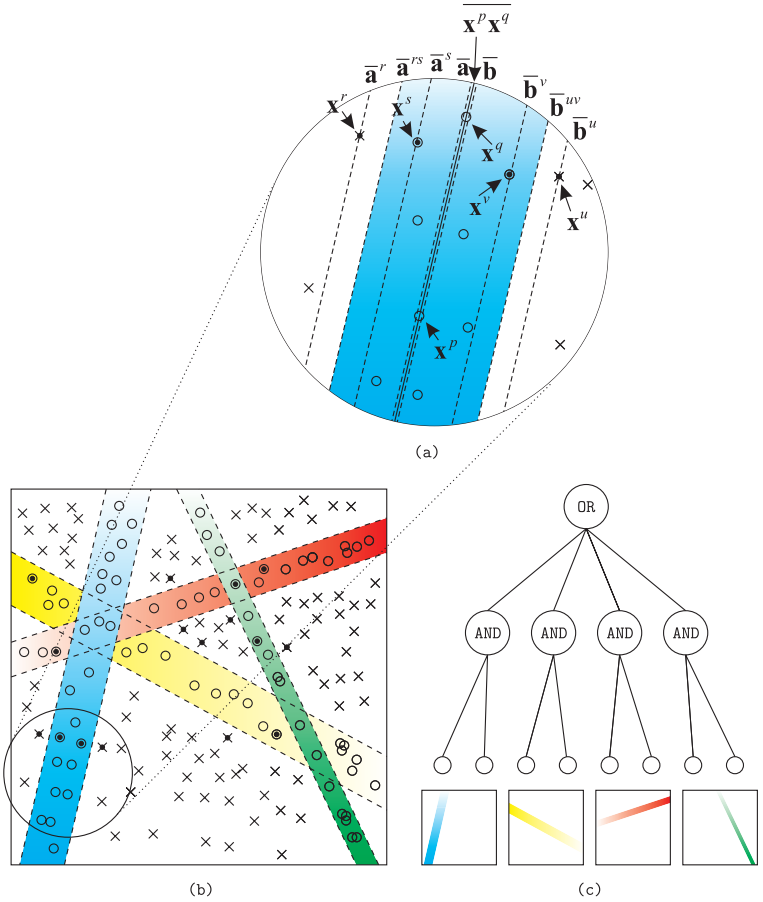


Fig. 2. The concept of the weight design method in Liou & Yu (1994).

classes are represented in a same code. This means that certain area code $\mathbf{y}^{(p,m)} \equiv \mathbf{a}_i^m \in A^m$ represents different class patterns, $\mathbf{y}^{(p,m)} = \mathbf{y}^{(q,m)}$ for $(\mathbf{x}^p, \mathbf{x}^q) \in V_{c_i,c_j}$. Any area that contains the mixed patterns, $(\mathbf{x}^p, \mathbf{x}^q) \in V_{c_i,c_j}$, is named 'ambiguity' area. An area that contains patterns belong to a single class is named 'unambiguity' area. The joint nodes of the tree are the HR codes. This tree exposes the latent error areas and their neurons. According to this tree, any BP algorithms cannot correct the latent error by adjusting the weights in its succeeding layers that near the output layer. The front layers must be trained correctly in order to send discriminated signals (HRs) to their succeeding layers. This suggests that one has to accomplish the MLP layer after layer in a bottom-up construction way.

Figure 3 illustrates an example tree for a two-class problem with the MLP network, $n_0 = 2$, $n_1 = 3$, $n_2 = 3$, $n_3 = 1$. There are 19 patterns in the class c_1 and 12 patterns in the class c_2, $P = 31$. In the input space, $n_0 = 2$, the three decision lines of the first hidden layer divide the input space into $\|A^1\| = 7$, $\|A^1\| = \Sigma_{k=0}^{n_0}\binom{n_1}{k}$, decision areas, $A^1 = \{\mathbf{a}_i^1, i = 1,2,3,5,6,7\}$. The input space is divided into 7 areas and coded with 7 binary codes shown in the bottom layer

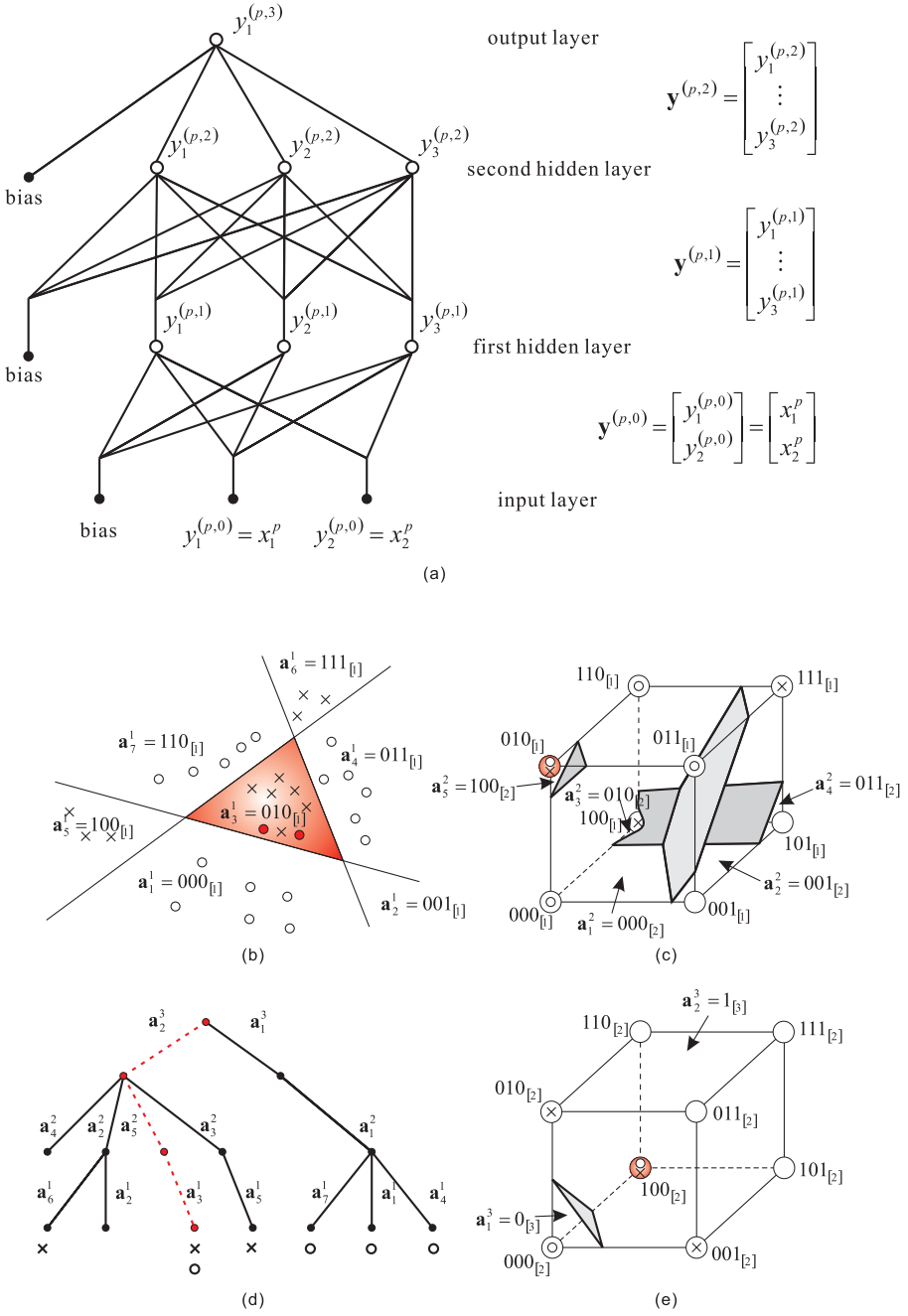


Fig. 3. The AIR tree for MLP Liou & Yu (1995).

of the tree. Note that these seven codes are symbols and are not binary numbers, for example $a_7^1 = 110_{[1]} \neq 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$. Each code represents all patterns in a single decision area. One of the areas is void, $\mathbf{a}_2^1$, and contains no pattern. No input pattern can activate this HR code. One area, $\mathbf{a}_3^1$, contains mixed patterns from both classes, this area is an ambiguity area. There exists one latent error and it can be traced by the tree, see the dash line in the figure. This kind error can be fixed locally using the patterns in the ambiguity area by adding extra neurons in this area. This error cannot be resolved by adjusting the weights in the layers n_2 and n_3 using any BP algorithms. When one deletes a significant neuron, many ambiguity areas will be generated. A less significant neuron will generate less number of ambiguity areas when it is deleted. One of two neurons can be pruned when both produce the same (or different) responses for all patterns. A neuron has the same response for all patterns can be pruned. One can build the tree for any training algorithms as an online monitor to visualize the latent contents. With this tree, one can see and understand how a MLP solve a problem. This tree can facilitate many applications.

Note that the latent errors cause the global mean square error (MSE) of the whole network stays significantly high constant for certain unpredictable period of time during the BP training. The BP may force the outputs of the error neurons to zero to suppress (or cover) the errors. This will do more harm than help to resolve the ambiguity areas. This is the major reason for slow learning and convergence to local minima of the BP algorithm. The presence of a high constant MSE is an important pointer of when we should fix the error neurons.

As for a single neuron, there exists at most one high constant MSE period Liou et al. (2005) Huang & Liou (2010) and the duration of that period is roughly proportion to the total number of boolean functions available near the origin point of input space.

1.4 Fourth issue: different functions of front layers and rear layers in MLP under BP training

The study in the work Liou & Yu (1995) further identifies the function of the front layers during the supervised BP training. It shows that categorization into different classes is the main function for those front layers. This means that the detailed identity of each class, the class label, is not used in the categorization. This suggests that the front layers can be successfully trained by using the discrimination differences between classes as the object function. The discrimination differences may come from parts of the distributed representation as discussed by Elman (1991).

Note that the most important conclusion of the work, Liou & Yu (1995), is that the MLP must be accomplished in a bottom-up manner in order to get perfect performance. It is hopeless to use the BP algorithm to get a global solution for a medium-sized MLP network. Any BP algorithm will converging to a local minimum solution. This work also shows that the MLP with perfect performance is a certain kind of " logical machine " that build a logical map for the events in the training dataset. This map can facilitate the comprehension of the events.

The 'SIR' method in Liou et al. (2000) provides the categorization object function based on the discrimination differences between classes, see program in web Liou (2000c). The front layers can be trained layer after layer using this object function starting from the first hidden layer. Perfect categorization and production of right signals can be accomplished for each layer Liou & Cheng (2008). These front layers are served, suitably, as the SOM perceptron. The SOM perceptron will utilize the discrimination differences between classes to train the front layers. It will not use class labels in its training process. Labels are not used as the

supervised primitives. Rather, the categorization is accreted under the side direction of those discrimination differences. It carry the similar notion as that in Elman (1991).
The work Liou & Yu (1995) also identifies the function of rear layers that are near the output layer. It shows that labeling with class labels is the main function. These front and rear functions comprise the supervised BP for the MLP. We will include a labeling sector that contains several layers after the SOM perceptron. The object function for the labeling sector is the class labels. One can apply any wide margin techniques to train this sector.
We will use the discrimination differences between classes to train each front layer starting from the first hidden layer. Any ambiguity area should be resolved for each layer. All ambiguity errors can be traced using the AIR tree and fixed locally and independently. This can be done either by the weight design method Liou & Yu (1994) for those error neurons or by the retraining method Liou & Yu (1995). For a severe ambiguity area, one may insert additional neurons and train them using the HRs in the ambiguity area. Note that any added neuron will not destroy the unambiguity area.
A second hidden layer is added to the first hidden layer when the outputs of the first hidden layer cannot produce well isolated HRs for each class. When a hidden layer can produce well isolated signals for different classes, it will be served as the last front layer and as the output of the SOM perceptron. We expect that the number of reduced HRs of the last layer will be equal to the number of classes, $\left\|\mathbf{Y}^L\right\| = \|C\|$.

2. Method

Figure 4 illustrates the SOM perceptron and the labeling sector. The SOM perceptron consists of layered neurons.
The relative distance between two patterns will be used in this work. For the pair patterns in the same class, $(\mathbf{x}^p, \mathbf{x}^q) \in U_{c_i}$, each layer is trained by using the energy function Liou et al. (2000),Liou & Cheng (2008),

$$E^{att}(\mathbf{x}^p, \mathbf{x}^q) = \frac{1}{2}\left\|\mathbf{y}^{(p,m)} - \mathbf{y}^{(q,m)}\right\|^2, \tag{3}$$

to reduce the distance between their output vectors, $\left\|\mathbf{y}^{(p,m)} - \mathbf{y}^{(q,m)}\right\|$. For the pair, $(\mathbf{x}^p, \mathbf{x}^q) \in V_{c_i,c_j}$, each layer is trained by using the energy function,

$$E^{rep}(\mathbf{x}^p, \mathbf{x}^q) = \frac{-1}{2}\left\|\mathbf{y}^{(p,m)} - \mathbf{y}^{(q,m)}\right\|^2, \tag{4}$$

to increase the distance between their output vectors. The discrimination information between classes is implicitly used in these two energies. They comprise the self-organizing principle for evolving the HRs on each layer space. We expect that these energies can maximally utilize all inherent discrimination differences among class patterns to separate the classes. Note that the class labels are not used in these two object functions. The labels will be used only in the labeling sector.
The network is constructed layer after layer, starting from $L = 1$. A new hidden layer is added, $L^{new} = L^{old} + 1$, whenever L^{old} layers cannot accomplish the isolation. All weights of the trained L^{old} layers are fixed during the training of the added layer, $m = L^{old} + 1$.
The weight matrix which connects the output of the $(m-1)$th layer and the input of the mth layer, is denoted by W^m. The W^1 connects the input layer and the first hidden layer. In this paper, 'module W^m' is used for representing the weights of the mth layer. Applying

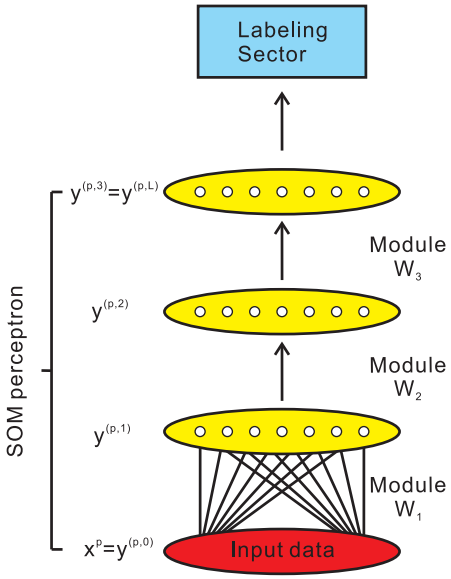


Fig. 4. The SOM perceptron and labeling sector.

the gradient descent method to the added layer, the two energies can be reduced efficiently during training iterations. The successfully trained network is used as the SOM perceptron to map the pattern, $\mathbf{x}^p$, to the output space, $\mathbf{y}^{(p,L)}$.

Algorithm

Each time a new hidden layer, $L^{new}=L^{old}+1$, is added, its weights are adjusted by the gradient descent method based on the energies (3) and (4). The weights of all trained layers, L^{old}, are fixed. Suppose there are two classes, $\{c_1=1, c_2=2\}$. The training algorithm is in below.

1. For each added layer W^m, W^m from W^1 to W^L
2. For limited epochs
3. Pick any two patterns in the same class, $\mathbf{x}^p$ and $\mathbf{x}^q$, which satisfy the following condition

$$(\mathbf{x}^p,\mathbf{x}^q)=\underset{\{(\mathbf{x}^i,\mathbf{x}^j)\in U_1 \text{ or } (\mathbf{x}^i,\mathbf{x}^j)\in U_2\}}{\arg\max}\left\|\mathbf{y}^{(i,m)}-\mathbf{y}^{(j,m)}\right\|^2. \tag{5}$$

   Among the pair patterns in the same class, either in U_1 or in U_2, the two patterns $(\mathbf{x}^p,\mathbf{x}^q)$ have the longest distance in the output space of the mth layer.

4. Find the pair patterns, $\mathbf{x}^r$ and $\mathbf{x}^s$ in different classes, which satisfy

$$(\mathbf{x}^r,\mathbf{x}^s)=\underset{(\mathbf{x}^i,\mathbf{x}^j)\in V_{1,2}}{\arg\min}\left\|\mathbf{y}^{(i,m)}-\mathbf{y}^{(j,m)}\right\|^2. \tag{6}$$

   The pair patterns $(\mathbf{x}^r,\mathbf{x}^s)$ have the shortest distance in the output space of the mth layer.

5. Adjust the module W^m by

$$\begin{aligned} \nabla W^m &\leftarrow \eta^{att}\frac{\partial E^{att}(\mathbf{x}^p,\mathbf{x}^q)}{\partial W^m}+\eta^{rep}\frac{\partial E^{rep}(\mathbf{x}^r,\mathbf{x}^s)}{\partial W^m} \\ W^m &\leftarrow W^m-\nabla W^m, \end{aligned} \tag{7}$$

where η^{att} and η^{rep} are learning rates.

The gradients of E^{att} and E^{rep} in (7) are

$$\begin{aligned} \frac{\partial E^{att}(\mathbf{x}^p,\mathbf{x}^q)}{\partial W^m} = &+\begin{bmatrix} \left(y_1^{(p,m)}-y_1^{(q,m)}\right)\left(1-\left(y_1^{(p,m)}\right)^2\right) \\ \vdots \\ \left(y_{n_m}^{(p,m)}-y_{n_m}^{(q,m)}\right)\left(1-\left(y_{n_m}^{(p,m)}\right)^2\right) \end{bmatrix}\left[y_1^{(p,m-1)},\ldots,y_{n_{m-1}}^{(p,m-1)},-1\right] \\ &-\begin{bmatrix} \left(y_1^{(p,m)}-y_1^{(q,m)}\right)\left(1-\left(y_1^{(q,m)}\right)^2\right) \\ \vdots \\ \left(y_{n_m}^{(p,m)}-y_{n_m}^{(q,m)}\right)\left(1-\left(y_{n_m}^{(q,m)}\right)^2\right) \end{bmatrix}\left[y_1^{(q,m-1)},\ldots,y_{n_{m-1}}^{(q,m-1)},-1\right] \end{aligned}$$

and

$$\begin{aligned} \frac{\partial E^{rep}(\mathbf{x}^p,\mathbf{x}^q)}{\partial W^m} = &-\begin{bmatrix} \left(y_1^{(p,m)}-y_1^{(q,m)}\right)\left(1-\left(y_1^{(p,m)}\right)^2\right) \\ \vdots \\ \left(y_{n_m}^{(p,m)}-y_{n_m}^{(q,m)}\right)\left(1-\left(y_{n_m}^{(p,m)}\right)^2\right) \end{bmatrix}\left[y_1^{(p,m-1)},\ldots,y_{n_{m-1}}^{(p,m-1)},-1\right] \\ &+\begin{bmatrix} \left(y_1^{(p,m)}-y_1^{(q,m)}\right)\left(1-\left(y_1^{(q,m)}\right)^2\right) \\ \vdots \\ \left(y_{n_m}^{(p,m)}-y_{n_m}^{(q,m)}\right)\left(1-\left(y_{n_m}^{(q,m)}\right)^2\right) \end{bmatrix}\left[y_1^{(q,m-1)},\ldots,y_{n_{m-1}}^{(q,m-1)},-1\right]. \end{aligned}$$

Figure 5 illustrates an example of the algorithm. In this figure, the intra-class pattern pairs are $U_1=\left\{\left(\mathbf{x}^1,\mathbf{x}^2\right),\left(\mathbf{x}^1,\mathbf{x}^3\right),\left(\mathbf{x}^2,\mathbf{x}^3\right)\right\}$ and $U_2=\left\{\left(\mathbf{x}^4,\mathbf{x}^5\right)\right\}$. The inter-class pattern pairs are $V_{1,2}=\left\{\left(\mathbf{x}^1,\mathbf{x}^4\right),\left(\mathbf{x}^1,\mathbf{x}^5\right),\left(\mathbf{x}^2,\mathbf{x}^4\right),\left(\mathbf{x}^2,\mathbf{x}^5\right),\left(\mathbf{x}^3,\mathbf{x}^4\right),\left(\mathbf{x}^3,\mathbf{x}^5\right)\right\}$. The intra-class pair has the maximal distance in the output space is

$$\left(\mathbf{x}^1,\mathbf{x}^2\right)=\underset{\{(\mathbf{x}^i,\mathbf{x}^j)\in U_1 \text{ or } (\mathbf{x}^i,\mathbf{x}^j)\in U_2\}}{\arg\max}\left\|\mathbf{y}^{(i,1)}-\mathbf{y}^{(j,1)}\right\|,$$

and the inter-class pair has the minimal distance is

$$\left(\mathbf{x}^3,\mathbf{x}^4\right)=\underset{\{(\mathbf{x}^i,\mathbf{x}^j)\in V_{1,2}\}}{\arg\min}\left\|\mathbf{y}^{(i,1)}-\mathbf{y}^{(j,1)}\right\|.$$

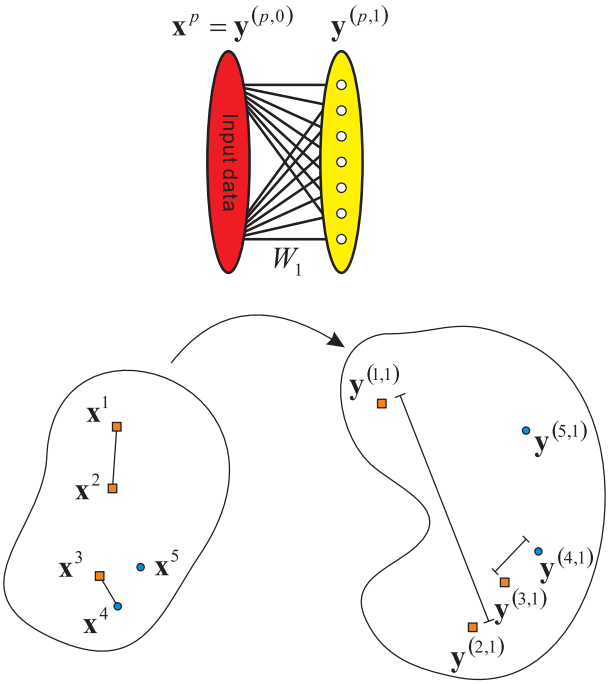


Fig. 5. A snapshoot of the first hidden layer of the algorithm.

The two energies are

$$E^{rep}\left(\mathbf{x}^3, \mathbf{x}^4\right) = \frac{-1}{2}\left\|\mathbf{y}^{(3,1)} - \mathbf{y}^{(4,1)}\right\|^2$$

and

$$E^{att}\left(\mathbf{x}^1, \mathbf{x}^2\right) = \frac{1}{2}\left\|\mathbf{y}^{(1,1)} - \mathbf{y}^{(2,1)}\right\|^2.$$

3. Experimental analysis

Two artificial datasets are used in the simulations. One is a two-class problem and the other is a three-class problem. Four real world datasets are also used in the simulations.

3.1 Two-class problem

Figure 6(b) plots the trained result for the two-class patterns, $c_i \in \{1, -1\}$, in the 2D plane, $n_0 = 2$. The border of these two-class patterns is $(x_1)^3 + 0.1x_1 = x_2$. Pattern points with the same color are in the same class. There are five neurons in each layer, $\{n_m = 5, m \in \{1, \ldots, L\}\}$. The SOM perceptron is trained layer after layer until it produces well isolated HRs for each class. We set the isolation condition for inter-class HRs as

$$\min_{(\mathbf{x}^p, \mathbf{x}^q) \in V_{1,2}} \left\|\mathbf{y}^{(p,L)} - \mathbf{y}^{(q,L)}\right\|^2 \approx 2^2 \times n_L, \tag{8}$$

and the condition for intra-class HRs as

$$\max_{\{(\mathbf{x}^p,\mathbf{x}^q)\in U_1 \text{or } (\mathbf{x}^p,\mathbf{x}^q)\in U_2\}} \left\|\mathbf{y}^{(p,L)} - \mathbf{y}^{(q,L)}\right\|^2 \approx 0. \tag{9}$$

The learning rates are $\eta^{att} = 0.01$ and $\eta^{rep} = 0.1$. This means that the repelling force is weighted stronger than the attractive force. The successful isolation is reached when $L = 2$. We set one neuron, $n_1^c = 1$, in the labeling sector as the output layer and use the class identities, $c_i \in \{1, -1\}$, to train this neuron. Figure 6(b) plots the trained result.
We also compare the result with those obtained by the MLP in Figure 6(a), and SVM in Figure 6(c). The MLP is a multilayer perceptron with two hidden layers, $n_1^{MLP} = n_2^{MLP} = 5$. This MLP is trained by the supervised BP. The Gaussian kernel, $K(\mathbf{u},\mathbf{v}) = \exp\left(\|\mathbf{u}-\mathbf{v}\|^2\right)$, is used in SVM Chang & Lin (2001).
The boundary in Figure 6(b) is much more close to the border than the result of the supervised MLP in Figure 6(a). Using the polynomial kernel, the boundary learned by SVM is also close to the border.

3.2 Multiple-class problem

Figure 7 plots the training patterns sampled from three classes separated by three ellipses, $c_i \in \{1,2,3\}$.
In this simulation, we train four SOM perceptrons with different number of neurons in each layer, $\{n_m = 5, n_m = 7, n_m = 9, n_m = 11\}$. Each layer is trained by 1000 epochs. The isolation conditions (8, 9) are used in this simulation to stop the addition of a new layer. The learning rates are $\eta^{att} = 0.01$ and $\eta^{rep} = 0.1$. The values of the isolation conditions for each layer

$$MinInterClass(m) = \min_{(\mathbf{x}^p,\mathbf{x}^q)\in\{V_{1,2},V_{1,3},V_{2,3}\}} \left\|y^{(p,m)} - y^{(q,m)}\right\|^2 \tag{10}$$

and

$$MaxIntraClass(m) = \max_{(\mathbf{x}^p,\mathbf{x}^q)\in\{U_1,U_2,U_3\}} \left\|y^{(p,m)} - y^{(q,m)}\right\|^2, \tag{11}$$

are recorded and plotted in the Figure 8.
When the isolation is reached, we set two layers in the labeling sector with $n_1^c = 2$ and $n_2^c = 3$ and use the class identities to train these two layers. In the layer $n_2^c = 3$, each neuron represents a single class.

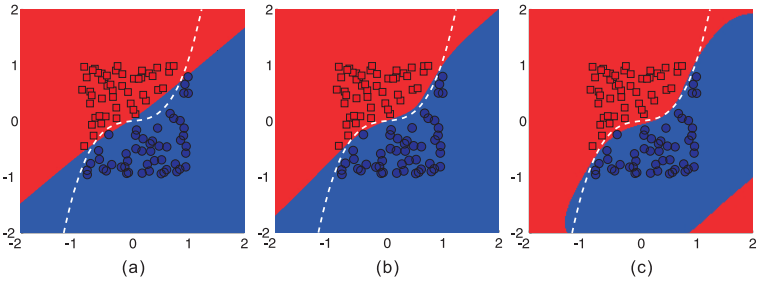


Fig. 6. The dash lines indicate the intrinsic border. (a) The trained result of MLP. (b) The result of SOM perceptron. (c) The result of SVM.

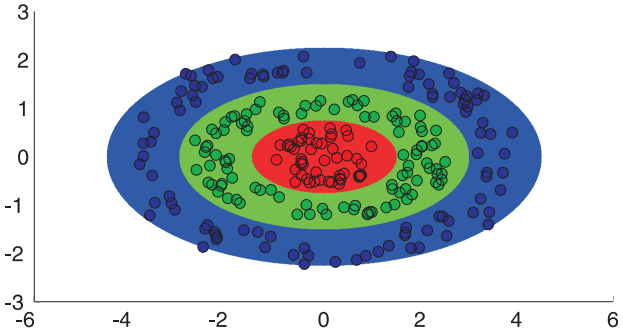


Fig. 7. The patterns in three classes.

We employ the SOM to visualize the output signals, $\mathbf{y}^{(j,m)}$, of each layer, to see the isolation of classes. The neurons of the SOM are placed on the regular points, see Figure 9. The SOM consists of 10×10 neurons.

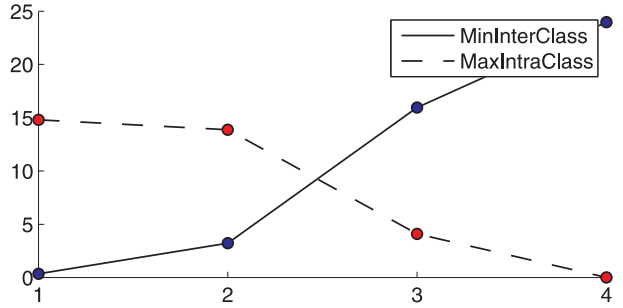


Fig. 8. Recorded isolation conditions, MinInterClass (10) and MaxIntraClass (11), for each layer, $m = 1,2,3,4$.

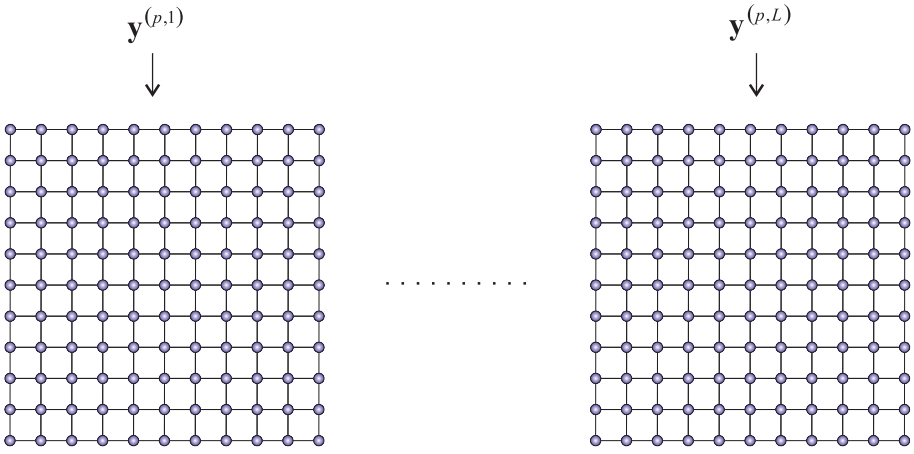


Fig. 9. The SOM used for visualization.

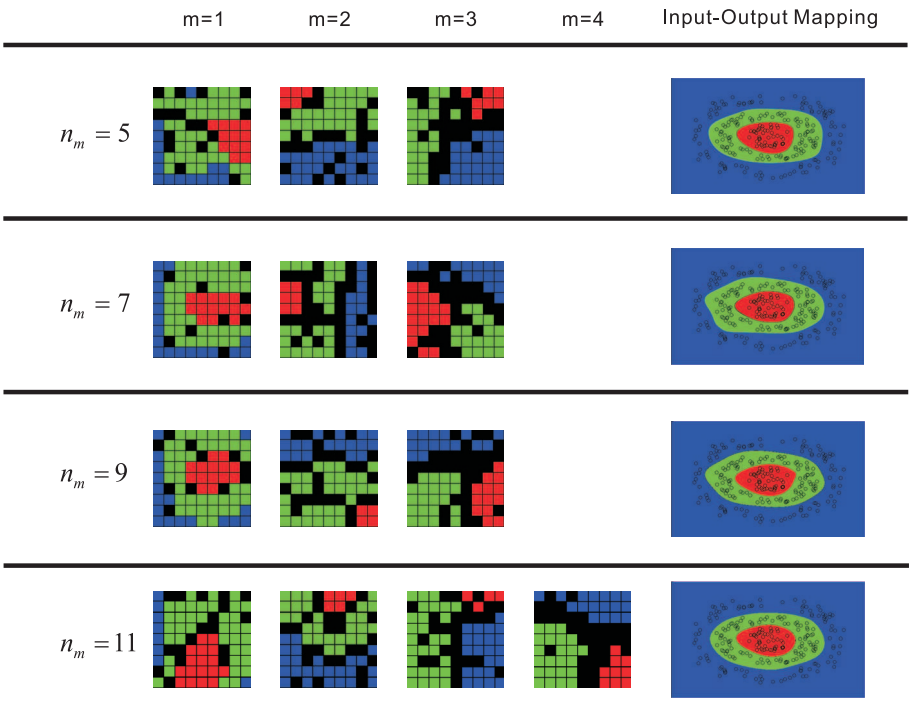


Fig. 10. The results of using the SOM to visualize the isolation of the output vectors of each layer. The images on the right column display the mapping relation between input space and output space of the labeling sector. Each neuron of the layer $n_2^c = 3$ indicates a single class (one color) in the input pattern space.

Figure 10 plots the SOM results for each layer. Each pixel denotes a SOM neuron. The pattern color is marked on its winner neuron. This figure shows that well isolated signals are gradually accomplished in the last few layers. The output signals of the last layer have three concentrated points in the SOM.

3.3 Real datasets

The Sonar Dataset, Wine Dataset, Ionosphere data and promoter gene sequences will be used in these experiments. Four machine learning techniques, k-NN (k-nearest neighbors algorithm), SOM perceptron, MLP and SVM, are compared using the 10-fold cross-validation. The dataset is randomly split into ten partitions, nine of them are used in the training process and the rest one is used in the testing process. The results are the average of the 10-fold cross-validation. The settings in the labeling sector for all datasets are listed in Table 1. The parameters of SVM are the cost C for the error tolerance and the gamma γ in the Gaussian kernel. Parameter k indicates the number of neighboring cells in the k-NN algorithm. The values of C, γ, and k, are optimized using an inner 10-fold cross-validation procedure. The settings that produce the lowest errors are used to learn the models with the whole set of training data. The MLP has two hidden layers. All parameters are listed in Table 1 and Table 2. The values of the input patterns are normalized within the range $[-1,1]$.

	k-NN	SOMP
		(n_m, n_1^c, n_2^c)
Sonar	(3,1,1,3,3)	(35,5,1)
	(3,1,1,1,1)	
Wine	(3,15,13,11,19)	(10,5,3)
	(15,19,13,15,11)	
Ionosphere	(1,3,1,1,1)	(10,5,1)
	(1,11,1,1,1)	
Promoters	(3,5,1,3,1)	(100,40,1)
	(3,3,3,3,3)	

Table 1. Parameters in k-NN and SOM perceptron

Table 3 and Table 4 show that the SOM perceptron is competitive and practicable in real world applications. The column 'SOMP' contains the results of SOM perceptron. As we expected, the training accuracy of the SOM perceptron is perfect 100%. In all our experiments the number of layers L is always small.

4. Summary

The topological form and correspondence on the L layer space, $\mathbf{Y}^L$, may resemble the fine organization of the corlica sensory representation, Homunculus Penfield & Rasmussen (1950) Kohonen (1982). There exists one and only one representation point on outer cortex for a sensory input.

A high dimensional combined representation point may indicate and serve as the state transition for two consecutive states and as the state transformation between two kinds of sensory inputs. For example, the transition between successive two states can be represented by a combined pattern in some sense similar to the bi-directional associative model Kosko (1988).

4.1 RBM and SOM perceptron

The HRs developed by the restricted Boltzmann machine (RBM) Salakhutdinov et al. (2007) and the Boltzmann machine Ackley et al. (1985); Liou & Lin (1989) are very different from those developed in the SOM perceptron. RBM is also constructed in a bottom-up manner. Each individual layer is developed freely and evolved independently. The labels are not used in RBM. The discrimination differences between classes are not used directly for the

	SVM		MLP	
	C	gamma	n_1^{MLP}	n_2^{MLP}
Sonar	$(2^5,2^1,2^5,2^3,2^7)$	$(2^{-3},2^{-1},2^{-3},2^{-3},2^{-5})$	30	10
	$(2^7,2^7,2^3,2^3,2^5)$	$(2^{-5},2^{-5},2^{-1},2^{-3},2^{-3})$		
Wine	$(2^{-1},2^1,2^{-1},2^3,2^1)$	$(2^{-1},2^{-1},2^{-1},2^{-9},2^{-3})$	20	5
	$(2^5,2^{-1},2^1,2^{-1},2^1)$	$(2^{-5},2^{-1},2^{-1},2^{-1},2^{-3})$		
Ionosphere	$(2^3,2^3,2^3,2^3,2^1)$	$(2^{-1},2^{-1},2^{-1},2^{-1},2^{-5})$	20	5
	$(2^5,2^{-1},2^1,2^{-1},2^1)$	$(2^{-3},2^{-5},2^{-1},2^{-3},2^{-1})$		
Promoters	$(2^1,2^1,2^1,2^3,2^1)$	$(2^{-9},2^{-11},2^{-7},2^{-13},2^{-7})$	20	5
	$(2^5,2^1,2^3,2^1,2^5)$	$(2^{-15},2^{-9},2^{-11},2^{-9},2^{-11})$		

Table 2. Parameters in SVM and MLP algorithms

Training Accuracy				
	k-NN	SOMP	MLP	SVM
Sonar	95.46%	100.00%	98.24%	100.00%
Wine	97.88%	100.00%	100.00%	99.56%
Ionosphere	97.69%	100.00%	99.46%	99.05%
Promoters	93.72%	100.00%	100.00%	100.00%

Table 3. The training accuracy on real dataset.

development of the RBM HRs. It is expected that the HRs developed in the RBM hidden layers can support the visual patterns to tolerate noisy patterns and variations of patterns. When the number of hidden neurons in a hidden layer is much less than that of visual patterns, the representation capacity is low and the HRs tend to encode those patterns with the complexity of each neuron layer. This encoding scheme has been extensively discussed in Ackley et al. (1985); Liou & Lin (1989). This kind encoding has been used to explain the mechanism of the RBM.

When the number of neurons is much larger than that of visual patterns, the representation capacity is high and there are so many alternative HRs for the visual representations. There exist so many alternative HRs for the visual support. There is no specific preferred HRs by the RBM. The SOM perceptron imposes the repelling energy to split different classes and the attraction energy to concentrate a class directly. It seeks those widely separated HRs to support and isolate the visual patterns.

4.2 Hebbian learning

It is generally accepted that the supervised BP algorithm is not biological plausible. The SIR learning Liou et al. (2000), weights degrading and enhancing mechanism for classes, keeps the Hebbian form for a single layer. It is biological possible. As for the Hebbian form, in the gradient formulas, $\frac{\partial E^{att}(\mathbf{x}^p,\mathbf{x}^q)}{\partial W^m}$ and $\frac{\partial E^{rep}(\mathbf{x}^p,\mathbf{x}^q)}{\partial W^m}$, the terms

$$\left(1-\left(y_i^{(p,m)}\right)^2\right) \text{ and } \left(1-\left(y_i^{(q,m)}\right)^2\right)$$

have nonnegative values. These two terms are the derivatives of the activation function $f(x)=\tanh(x)$. If we substitute 1 for these terms, the learning rule becomes

$$w_{ij}^m(n+1) \longleftarrow w_{ij}^m(n)+\eta\left(y_i^{(p,m)}(n)-y_i^{(q,m)}(n)\right)\left(y_j^{(p,m-1)}(n)-y_j^{(q,m-1)}(n)\right), \qquad (12)$$

that is the Hebbian learning with 'bi-patterns'. The modification strength is proportional to the difference between two HR patterns,

$$\left(y_j^{(p,m-1)}(n)-y_j^{(q,m-1)}(n)\right),$$

Testing Accuracy				
	k-NN	SOMP	MLP	SVM
Sonar	82.74%	86.60%	84.14%	88.00%
Wine	97.78%	98.33%	97.78%	98.30%
Ionosphere	85.17%	90.60%	88.32%	94.87%
Promoters	72.64%	86.55%	85.73%	89.36%

Table 4. The testing accuracy on real dataset.

and the difference between their postsynaptic responses (output vectors),

$$\left(y_i^{(p,m)}(n) - y_i^{(q,m)}(n)\right).$$

The increase of the strength of a synapse, $w_{ij}^m(n+1)$, is proportional to such differences on both sides of that synapse synchronously. To compare, a popular Hebbian learning form is

$$w_{ij}(n+1) \longleftarrow w_{ij}(n) + \eta y_i(n) x_j(n), \tag{13}$$

where $y_i(n)$ is the postsynaptic response and $x_j(n)$ is the presynaptic input pattern.
Another interesting form called the covariance hypothesis was introduced in Sejnowski (1977). According to this hypothesis, the learning applied to the synaptic weight w_{ij} is defined by

$$w_{ij}(n+1) \longleftarrow w_{ij}(n) + \eta\left(y_i(n) - \bar{y}(n)\right)\left(x_j(n) - \bar{x}(n)\right) \tag{14}$$

where $\bar{x}(n)$ and $\bar{y}(n)$ denote the time-averaged values of x_i and y_i, respectively. Comparing Equation (13) and Equation (14), the differences between them are the presynaptic and postsynaptic reference thresholds, which determine the sign of synaptic modification. In Equation (12), instead of the time-averaged references, the presynaptic signal and the postsynaptic signal use the other signals as the references.

4.3 Relation with support vector machine

The support vector machine employs the Mercer kernel to map patterns to a high dimensional space. Usually, the class information is not used in the design of the mapping function. The outcome of the mapping relies on the choice and the setting of the kernel function. The SOM perceptron is an adjustable mapping function to transform the patterns to a high dimensional space. It can be used as the Mercer kernel to map patterns to a space with highly separated representations.
In SVM, a multi-class classification task (polychotomy) can be decomposed into a set of simpler two-class classification tasks (dichotomies). Each dichotomy is implemented using one such machine independently. The outputs of these dichotomizes are reconstructed in classification. Advanced techniques have been developed for decomposition of poychotomy into dichotomies and reconstruction of their outputs. The SOM perceptron attempts to simultaneously divide the whole representation space for all classes. It uses the internal space of the layer perceptrons where each dichotomy (hyperplane) learns in a way dependent on each other. This learning will exhaust the hidden layer space and maximize the utility of all neurons to accomplish highly separated representations in that layer. Such representations have large margins and facilitate the operations of error correction.

4.4 Relation with mutual information learning

The proposed method is based on the maximization of the representations distances among different class representations and the minimization of the distances among the same class representations. There is a network Becker & Hinton (1992) with two modules. It has a different goal. It maximizes the mutual information, $I\left(\mathbf{y}^{(p,L)};\mathbf{y}^{(q,L)}\right)$, where $\mathbf{y}^{(p,L)}$ and $\mathbf{y}^{(q,L)}$ are the output vectors corresponding to the input patterns $\mathbf{x}^p$ and $\mathbf{x}^q$. This mutual information is defined as

$$I = \frac{1}{2}\log\frac{Var\left(\mathbf{y}^{(p,L)} + \mathbf{y}^{(q,L)}\right)}{Var\left(\mathbf{y}^{(p,L)} - \mathbf{y}^{(q,L)}\right)}$$

where *Var* is the variance over the responses of the training samples. This network is plotted in Figure 11(a). The goal of this network is to make the output $\mathbf{y}^{(p,L)}$ and $\mathbf{y}^{(q,L)}$ of the two modules to agree closely (i.e., to have a small value in the denominator $Var\left(\mathbf{y}^{(p,L)}-\mathbf{y}^{(q,L)}\right)$) for a closely related pair of input patterns $\mathbf{x}^p$ and $\mathbf{x}^q$. This goal is similar to, in some sense, the attraction energy for the same class patterns. In the same time, the two modules cannot just produce constant output that is unaffected by the input patterns, otherwise, they convey no information. The outputs of these two modules should vary as the inputs are varied. This constant situation is prevented by a large value in the numerator, $Var\left(\mathbf{y}^{(p,L)}+\mathbf{y}^{(q,L)}\right)$. Since there are two more classes in SOM perceptron, this kind prevention is not necessary. We will ignore the discussion on this numerator.
When we replace this two-module network with a single-module network as shown in Figure 11(b) and confine the output responses in a hypercube space. We then train this network to minimize the object function

$$I' = \frac{1}{2}\log Var\left(\mathbf{y}^{(p,L)}-\mathbf{y}^{(q,L)}\right)$$

for patterns in the same class that are closely related pairs. Conversely, we train the network to maximize this function for different class patterns. We could obtain similar results for the same class patterns as those obtained by the SOM perceptron.
We briefly explain the similarity of the two goals. The object function I' will weight frequent patterns. In our experiments all patterns have equal appearance (uniform probability distribution). Suppose that the mean value of the vector $\left(\mathbf{y}^{(p,m)}-\mathbf{y}^{(q,m)}\right)$ is zero. Assume each pattern in $\left\{\mathbf{x}^1,\mathbf{x}^2,\ldots,\mathbf{x}^P\right\}$ has its own representation, $\mathbf{y}^{(p,m)} \neq \mathbf{y}^{(q,m)}$ for $p \neq q$. Assume $\mathbf{y}^{(p,L)}-\mathbf{y}^{(q,L)}$ has equal probability of appearance, P^{-1}. Then

$$I' = \frac{1}{2}\{-2E^{rep}\} - \log P.$$

The information function, I', is similar to the repelling energy and the attraction energy. This shows that the two energies are agree with the mutual information to a certain extent. Note that the assumption on the equal probability of $\mathbf{y}^{(p,L)}-\mathbf{y}^{(q,L)}$ is not precise.

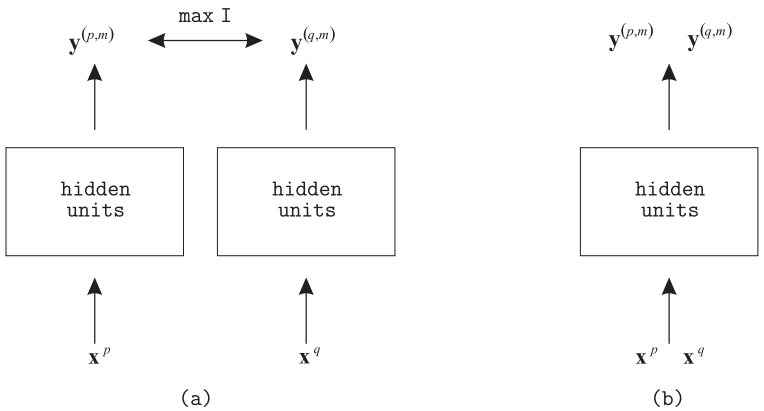


Fig. 11. (a) The two-module network. (b) The single-module network.

When we use

$$E^{rep} = -\frac{1}{2}\sum_{p=1}^{P}\sum_{q=1}^{P}\sum_{i=1}^{n_m}\left(y_i^{(p,m)} - y_i^{(q,m)}\right)^2,$$

the SIR method tends to maximize (or minimize) the variance of each neuron's output difference, $Var\left(y_i^{(p,m)} - y_i^{(q,m)}\right)$, evenly for all pairs of different class patterns (or same class patterns). All neurons will be devoted to these class patterns. All neurons are sensitive to these patterns only. Any unknown pattern will be included in one of these patterns' representations. In other words, these representations exhaust the pattern space.

4.4.1 Acknowledgement
This work is supported by National Science Council.

5. References

Ackley, D., Hinton, G. & Sejnowski, T. (1985). A learning algorithm for boltzmann machines, *Cognitive Science* 9: 147–169.

Becker, S. & Hinton, G. (1992). A self-organizing neural network that discovers surfaces in random-dot stereograms, *Nature* 355: 161–163.

Boser, B., Guyon, I. & Vapnik, V. (1992). A training algorithm for optimal margin classifiers, *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152.

Chang, C.-C. & Lin, C.-J. (2001). Libsvm : a library for support vector machines, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Daugman, J. (1980). Two-dimensional spectral analysis of cortical receptive field profiles, *Vision research* 20: 847–856.

Dobbins, A., Zucker, S. & Cynader, M. (1987). Endstopped neurons in the visual cortex as a substrate for calculating curvature, *Nature* 329: 438–441.

Elman, J. (1991). Distributed representations, simple recurrent networks, and grammatical structure, *Machine Learning* 7: 195–225.

Huang, J.-C. & Liou, C.-Y. (2010). Solution space of perceptron, *International conference on Artificial Neural Networks (ICANN)*.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics* 43: 59–69.

Kosko, B. (1988). Bidirectional associative memories, *IEEE Transactions on Systems, Man, and Cybernetics* 18: 49–60.

Liou, C.-Y. (2000a). NN Chapter 3, http://red.csie.ntu.edu.tw/NN/Classinfo/code/COPLANARITY.M.

Liou, C.-Y. (2000b). NN Chapter 4, http://red.csie.ntu.edu.tw/NN/Demo/hamming_tree.html.

Liou, C.-Y. (2000c). NN Chapter 5, http://red.csie.ntu.edu.tw/NN/Classinfo/code/SIR_METHOD.M.

Liou, C.-Y., Chen, H.-T. & Huang, J.-C. (2000). Separation of internal representations of the hidden layer, *Proceedings of the 2000 International Computer Symposium*, pp. 26–34.

Liou, C.-Y. & Cheng, W.-C. (2008). Resolving hidden representations, *Lecture Notes in Computer Science*, Vol. 4985, Part II, Springer, Heidelberg, pp. 254–263.

Liou, C.-Y., Huang, J.-C. & Kuo, Y.-T. (2005). Geometrical perspective on learning behavior, *Journal of Information Science and Engineering* 21: 721–732.

Liou, C.-Y. & Lin, S.-L. (1989). The other variant boltzmann machine, *Proceedings of the International Joint Conference on Neural Networks*, Washington, D. C., USA, pp. 449–454.

Liou, C.-Y. & Yu, W.-J. (1994). Initializing the weights in multilayer network with quadratic sigmoid function, *Proceedings of the International Conference on Neural Information Processing*, pp. 1387–1392.

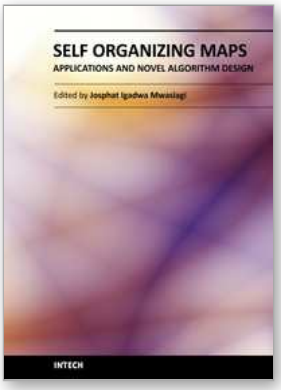
Liou, C.-Y. & Yu, W.-J. (1995). Ambiguous binary representation in multilayer neural network, *Proceedings of International Conference on Neural Networks*, Vol. 1, pp. 379–384.

Mirchandini, G. & Cao, W. (1989). On hidden nodes in neural nets, *IEEE Transactions on Circuits and Systems* 36: 661–664.

Penfield, W. & Rasmussen, T. (1950). *The cerebral cortex of man*, The Macmillan Company, New York.

Salakhutdinov, R., Mnih, A. & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering, *Proceedings of the 24th international conference on Machine learning*, pp. 791–798.

Sejnowski, T. (1977). Storing covariance with nonlinearly interacting neurons, *Journal of mathematical biology* 4: 303–321.



Self Organizing Maps - Applications and Novel Algorithm Design
Edited by Dr Josphat Igadwa Mwasiagi

ISBN 978-953-307-546-4
Hard cover, 702 pages
Publisher InTech
Published online 21, January, 2011
Published in print edition January, 2011

Kohonen Self Organizing Maps (SOM) has found application in practical all fields, especially those which tend to handle high dimensional data. SOM can be used for the clustering of genes in the medical field, the study of multi-media and web based contents and in the transportation industry, just to name a few. Apart from the aforementioned areas this book also covers the study of complex data found in meteorological and remotely sensed images acquired using satellite sensing. Data management and envelopment analysis has also been covered. The application of SOM in mechanical and manufacturing engineering forms another important area of this book. The final section of this book, addresses the design and application of novel variants of SOM algorithms.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821