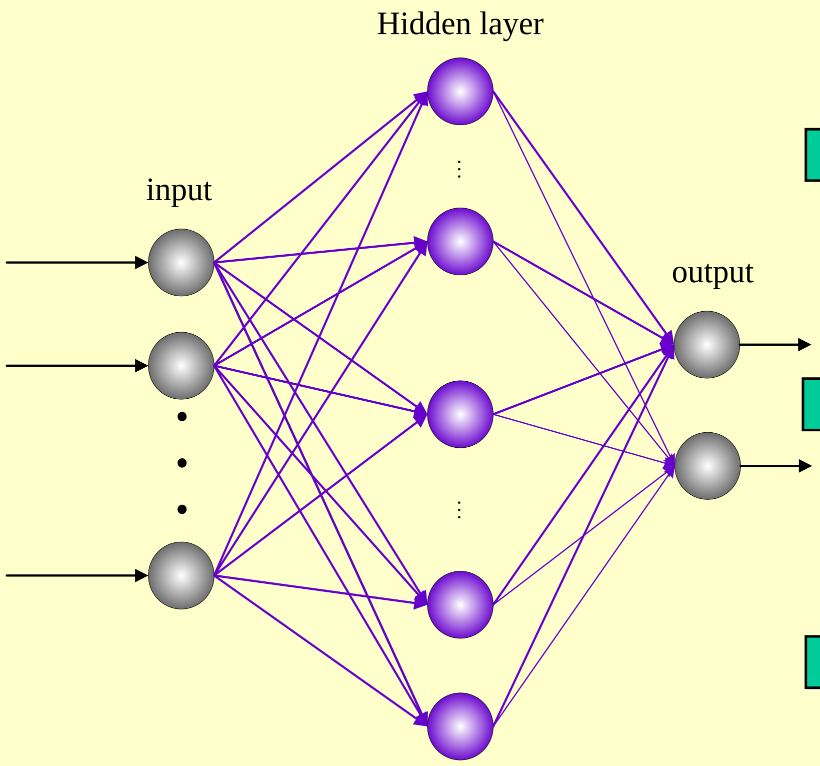
A Self-Adaptive Multi-Hierarchical Modular Neural Network for Complex Problems

Reporter：WangQiuWan

Xi’an University of Science and Technology

1 Introduction

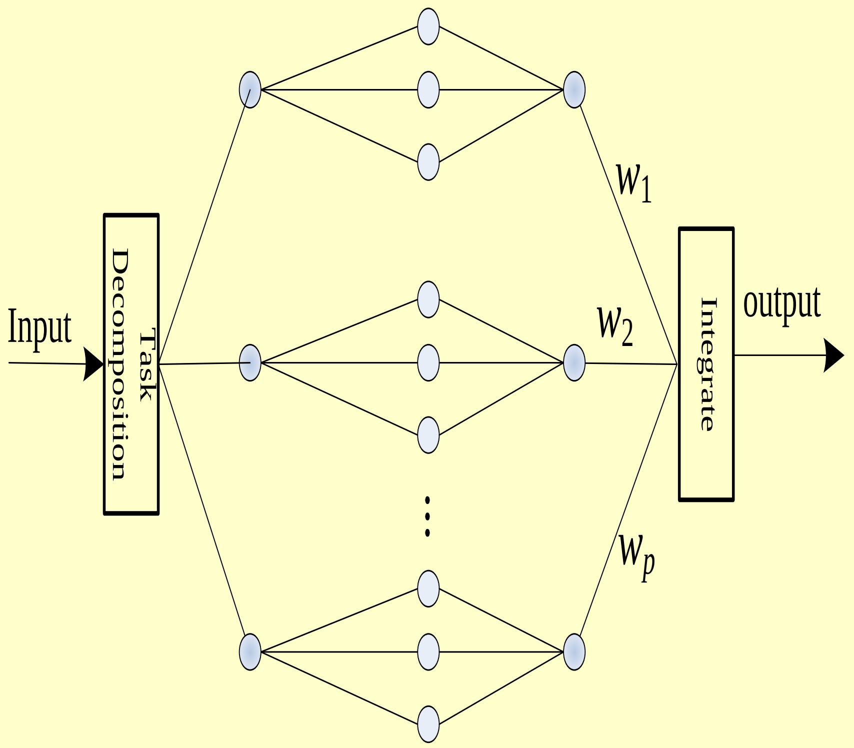


Convergence speed is very slowly

Difficult to converge to global minimum point

Poor scalability and difficult in proliferating knowledge

1 Introduction



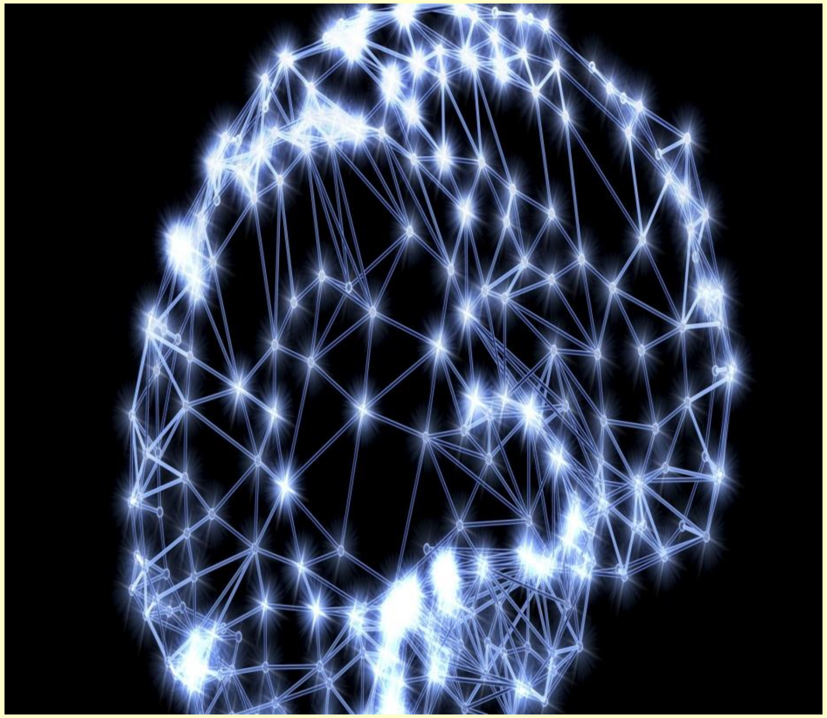
Advantages

- Efficiency
- Simplicity of structure
- Ease of evaluation
- Fault tolerance
- Better extendibility
- Robustness

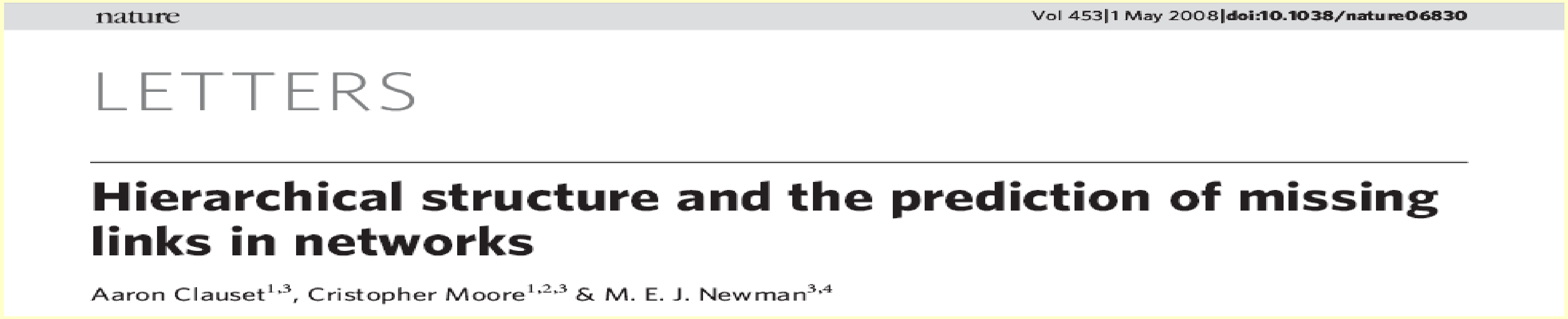
Disadvantage

- Difficult in module division
- Reduce the MNN's learning accuracy

1 Introduction



The brain networks demonstrate the property of hierarchical modularity, within each module there will be a set of sub-modules, and within each sub-module a set of sub-sub-modules



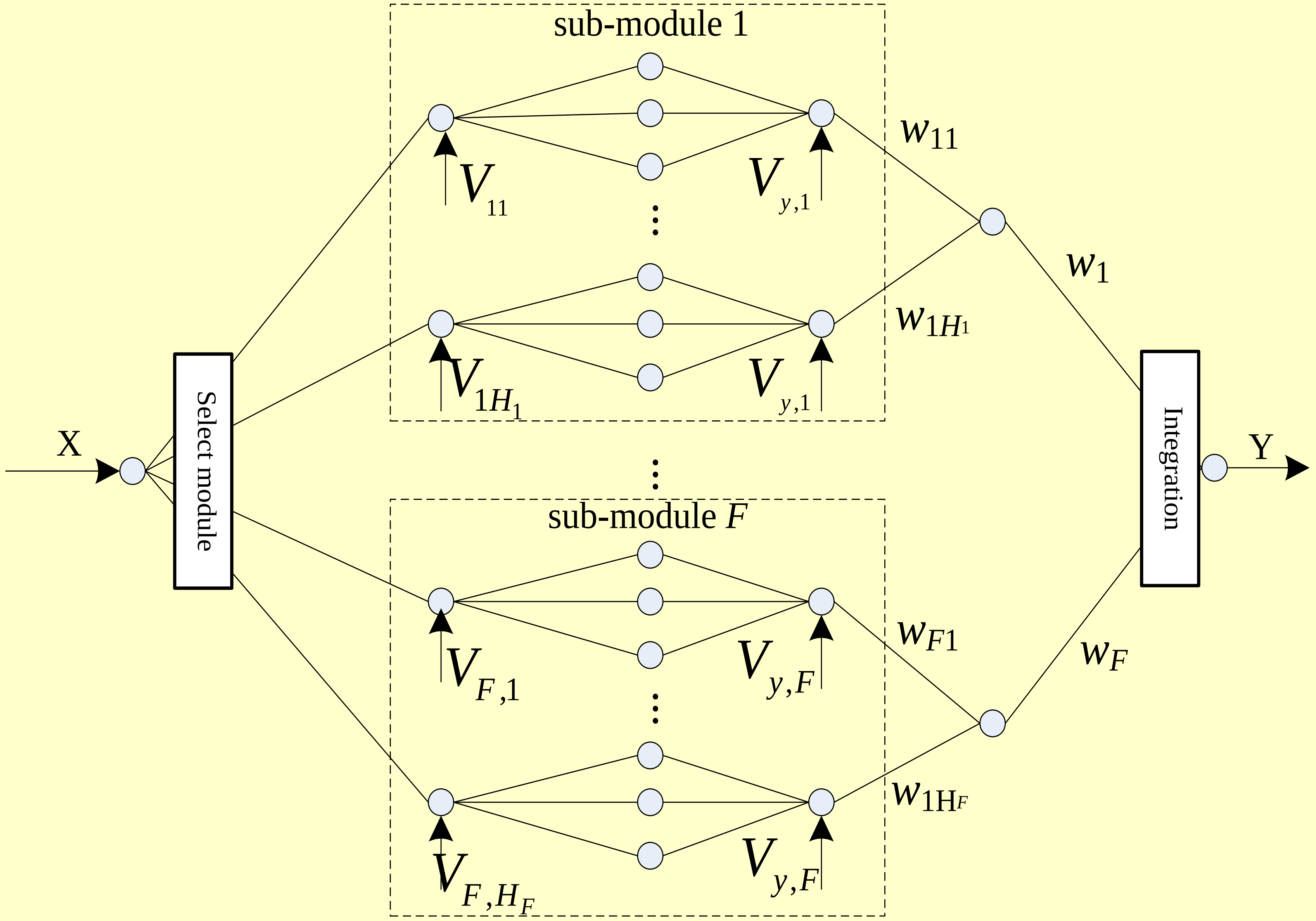
nature Vol 453|1 May 2008|doi:10.1038/nature06830

LETTERS

Hierarchical structure and the prediction of missing links in networks

Aaron Clauset[1,3], Cristopher Moore[1,2,3] & M. E. J. Newman[3,4]

2 BMNN Structure



Design the structure of BMNN

➡ **How to divide the sub-modules and sub-sub-modules**

➡ **How to select different sub-sub-modules from the different sub-modules to learning the input samples**

➡ **How to integrate the learning results of the sub-sub-modules**

1 Divide the sub-modules

Given sample set S={($\boldsymbol{X}_k$), k=1, …,N}

K-means of S

Clustering center is {$\mathbf{C}_1$,$\mathbf{C}_2$,..,$\mathbf{C}_F$}

Establishing the target set S into F fuzzy sets

According to the result of the fuzzy sets, the whole module can be divided into F sub-modules

$$f_{ik}=\begin{cases}\exp\left(\dfrac{-\left\|\mathbf{X}_k-\mathbf{C}_i\right\|^2}{\left(\dfrac{\mathbf{C}_i-\mathbf{C}_{i-1}}{2}\right)^2}\right), \mathbf{X}_k\le\mathbf{C}_i\\[2ex] \exp\left(\dfrac{-\left\|\mathbf{X}_k-\mathbf{C}_i\right\|^2}{\left(\dfrac{\mathbf{C}_{i+1}-\mathbf{C}_i}{2}\right)^2}\right), \mathbf{X}_k>\mathbf{C}_i\end{cases} \qquad \textbf{(1)}$$

2 Divide the sub-sub-modules

For each target fuzzy set, fuzzy clustering the input sample ***X***

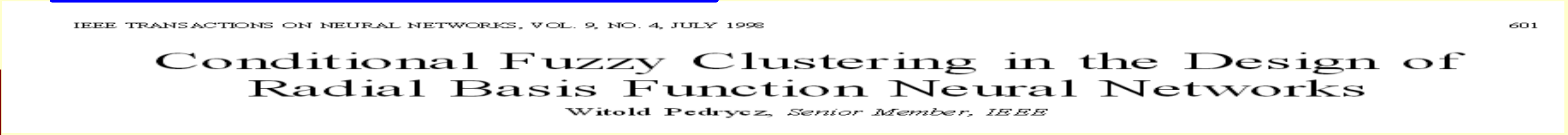
The method can divide the whole input sample set into H_T sample subsets

The results of the condition fuzzy clustering can establish sub-sub-modules for each sample subset, then there will be H_T sub-sub-modules in BMNN

$$\mathrm{V}_{ij} = \frac{\sum_{k=1}^{N} \left(\mathrm{u}_{ijk}\right)^2 \mathrm{x}_k}{\sum_{k=1}^{N} \left(\mathrm{u}_{ijk}\right)^2} \qquad \textbf{(2)}$$

$$\mathrm{u}_{ijk} = \frac{f_{ik}}{\sum_{m=1}^{H_i} \left(\frac{\left\| \mathrm{X}_k - \mathrm{V}_{ij} \right\|}{\left\| \mathrm{X}_k - \mathrm{V}_{im} \right\|} \right)^2} \qquad \textbf{(3)}$$

$$H_T = \sum_{i=1}^{F} H_i \qquad \textbf{(4)}$$



IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 9, NO. 4, JULY 1998 601

Conditional Fuzzy Clustering in the Design of Radial Basis Function Neural Networks

Witold Pedrycz, *Senior Member, IEEE*

3 Select sub-module and sub-sub-module

The sub-module selection is to determine which sub-module will be selected to process the subtask. According to the classification method described above, there exist some affiliation between the training set and sub-modules, therefore, the sub-module selection is to determine the likelihood of a given input sample belongs to the sub-module or sub-sub-module.

if the distance of $\boldsymbol{X}_k$ is close to the center of sample set $\boldsymbol{V}_{ij}$, then the likelihood of the X_k belongs to NET_{ij} is large. The relative distance is adopted to measure the likelihood of $\boldsymbol{X}_k$ belongs to NET_{ij}.

$$J_i = \sum_{j}^{H_i} w_{ij} d_{ij} \qquad (5)$$

$$\sum_{j=1}^{H_i} w_{ij} = 1 \qquad w_{ij} \in [0,1] \qquad d_{ij} = \left\| \boldsymbol{X}_k - \boldsymbol{V}_{ij} \right\| / da_{ij} \qquad da_{ij} = \frac{1}{N_{ij}} \sum_{m=1}^{N_{ij}} \left\| \boldsymbol{X}_m - \boldsymbol{V}_{ij} \right\|$$

3 Construction of The BMNN Structure

Minimizing the performance index function J_i with Lagrange multiplier method can solve w_{ij}

$$w_{ij} = \begin{cases} 1 & , when\ d_{ij} = 0. \\ \dfrac{\left(\dfrac{1}{d_{ij}}\right)}{\sum_{j=1}^{H_i}\left(\dfrac{1}{d_{ij}}\right)} & , otherwise. \end{cases} \qquad \textbf{(6)}$$

where i=1,…,F, j=1,…, H_i. Obviously, if d_{ij} is larger then w_{ij} is smaller, while the likelihood of $\boldsymbol{X}_k$ belongs to NET_{ij} would be smaller. If d_{ij} is smaller then w_{ij} is larger, while the likelihood of $\boldsymbol{X}_k$ belongs to NET_{ij} would be larger. Thus, a sub-sub-module will be selected from each sub-module to processing $\boldsymbol{X}_k$, and the output of each sub-module is the output of the NET_{is}.

3 Construction of The BMNN Structure

However, this is only a primary selection, not all NET_{is} (i=1,…,F) are suitable for take part in the learning process. Therefore, the sub-sub-modules which are selected but not suitable for processing $\boldsymbol{X}_k$ must be filtered. The method of filtering sub-sub-modules are the same way as the previous sub-module selection method. Establish the performance index function for the selected sub-sub-modules:

$$J = \sum_{i}^{F} w_i d_i \qquad \textbf{(7)}$$

$$\sum_{i=1}^{F} w_i = 1 \quad w_i \in [0,1] \quad d_i = \left\| \boldsymbol{X}_k - \mathrm{C}_i \right\| / da_i \quad da_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \left\| \boldsymbol{X}_j - \mathrm{C}_i \right\|$$

da_i is average distance between the NET_i training samples, d_i is the relative distance measure that $\boldsymbol{X}_k$ to NET_i, N_i is the training sample number of NET_i, $\mathbf{C}_i$ is the cluster center of the training samples that belongs to NET_i.

3 Construction of The BMNN Structure

Minimizing the performance index function *J*, use Lagrange multiplier method can figure out w_i as

$$w_i = \begin{cases} 1, & when \;\; d_i = 0. \\ \dfrac{\left(\dfrac{1}{d_i}\right)}{\sum_{i=1}^{F}\left(\dfrac{1}{d_i}\right)}, & otherwise. \end{cases} \qquad \textbf{(8)}$$

Because each sub-module has only one sub-sub-module, NET_{is}, to learn the training sample $\boldsymbol{X}_k$, therefore, the w_i is actually the membership degree of $\boldsymbol{X}_k$ for NET_{is}. Taking into account the overlapping among the training samples, a threshold *K* can be set, the sub-sub-module which satisfies the condition $w_i \geq K$ will be selected to learning $\boldsymbol{X}_k$. According to the above described selection method, for a given input sample $\boldsymbol{X}_k$, there will be varying amounts of sub-sub-modules involved in learning with different distributions positions of the $\boldsymbol{X}_k$ and different value of *K*.

4 Integration the output of sub-modules

Suppose the input sample is $\boldsymbol{X}_k$, let $\boldsymbol{w}=\{w_1,\ldots, w_F\}$, if $w_i<k$ then let $w_i=0(i=1,\ldots,F)$, and normalize the $\boldsymbol{w}$, then the output of the BMNN is

$$Y=\sum_{i=1}^{C} w_i y_i \qquad \textbf{(9)}$$

where y_i is the output of NET_i, w_i is the weight value of NET_i, which is the ith component of the normalized $\boldsymbol{w}$. The unselected sub-module's weights w_i=0, so it make no contribution to the output of the BMNN, and the total output of the BMNN is the weighted sum of the selected sub-modules.

5 Self-adaptively construction the structure of sub-modules

In BMNN, each sub-modules is a RBF network, how to construct an appropriate structure of RBF network according to the learning task is a difficult problem, Wilamowski improved the Levenberg-Marquardt(LM) learning algorithm, and Yu proposed an ErrCor algorithm which is an incremental design of RBF networks. The basic idea of the ErrCor algorithm is to use RBF units with kernel function (1) to create a peak/valley shape to compensate for the largest error in the error surface at the beginning of each iteration, and it is able to design the most compact RBF structure.

$$\square_{k+1} = \square_k - \left(\mathbf{Q}_k + \mu_k \mathbf{I}\right)^{-1} \mathbf{g}_k \qquad \mathbf{Q} = \sum_{p=1}^{P} \mathbf{q}_p ; \mathbf{q}_p = \mathbf{j}_p^T \mathbf{j}_p$$

$$\mathbf{g} = \sum_{p=1}^{P} \boldsymbol{\eta}_p ; \boldsymbol{\eta}_p = \mathbf{j}_p^T e_p \qquad e_p = y_p - o_p \qquad \mathbf{j}_{p,n} = \frac{\partial e_p}{\partial \square_n}$$

3 Construction of The BMNN Structure

In BMNN, For a given training sample X_p, considering the RBF network parameters $w_h, c_{h,i}$ and σ_h, the elements of the Jacobian row can be organized as

$$\mathbf{j}_{p,n} = [\frac{\partial e_p}{\partial w_0}, \frac{\partial e_p}{\partial w_1} L \frac{\partial e_p}{\partial w_h} L \frac{\partial e_p}{\partial w_h}, \frac{\partial e_p}{\partial c_{1,1}} L \frac{\partial e_p}{\partial c_{1,i}} L$$

$$\frac{\partial e_p}{\partial c_{1,I}} L \frac{\partial e_p}{\partial c_{h,1}} L \frac{\partial e_p}{\partial c_{h,i}} L \frac{\partial e_p}{\partial c_{h,I}} L \frac{\partial e_p}{\partial c_{H,1}} L \qquad (10)$$

$$\frac{\partial e_p}{\partial c_{H,i}} L \frac{\partial e_p}{\partial c_{H,i}}, \frac{\partial e_p}{\partial \sigma_1} L \frac{\partial e_p}{\partial \sigma_h} L \frac{\partial e_p}{\partial \sigma_H}]$$

5 Self-adaptively construction the structure of sub-modules

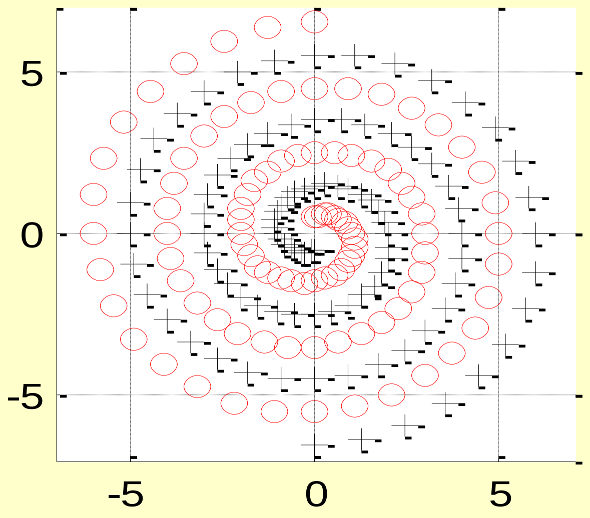
In BMNN, Integrating the above formulas, with differential chain rule, the Jacobian row elements for X_p in (22) can be rewritten as

$$\frac{\partial e_p}{\partial w_h} = -\varphi_h\left(X_p\right), \frac{\partial e_p}{\partial w_0} = -1 \qquad \frac{\partial e_p}{\partial c_{h,i}} = -\frac{2w_h\varphi_h\left(X_p\right)\left(x_{p,i} - c_{h,i}\right)}{\sigma_h} \qquad \frac{\partial e_p}{\partial \sigma_h} = -\frac{w_h\varphi_h\left(X_p\right)\left\|X_p - c_h\right\|^2}{\sigma_h^{\ 2}}$$

With these three formulas, all the elements of Jacobian row J_p for the given training sample X_p can be calculated. After applying all the patterns, quasi-Hiessian matrix Q and gradient vector g are obtained, so as to apply the update rule for parameter adjustment. It can self-adaptively construct the structure of the sub-modules according to the training samples from task decomposition layer by applying the aforementioned ErrCor algorithm.

1 Classification of two spiral problem

$$\begin{cases} \theta = i \times \pi / 16 \\ r = 6.5 \times (104 - i)/104 \\ x = r \times \sin(\theta) \\ y = r \times \cos(\theta) \end{cases} \qquad (17)$$



Training sample:

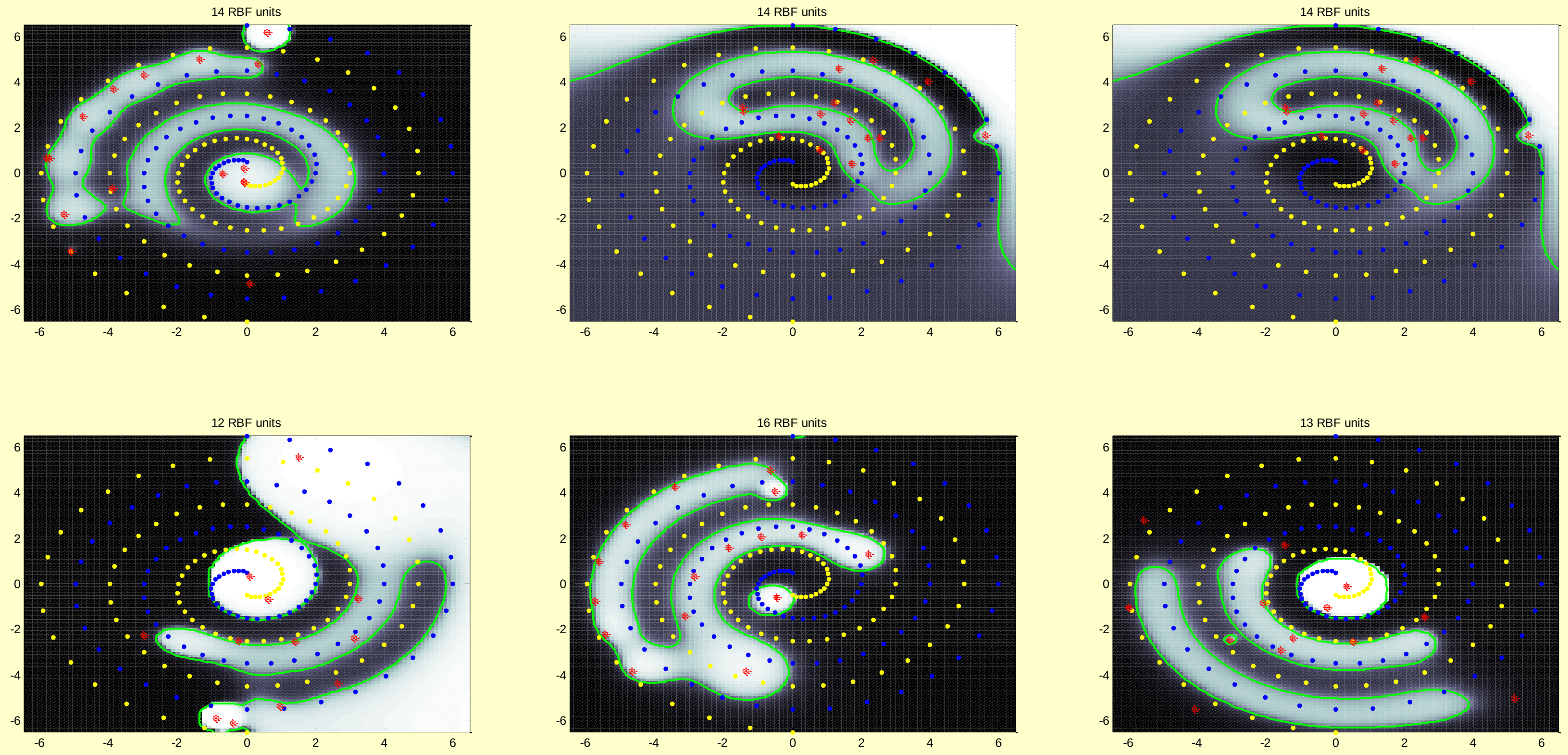
The total number of training sample is 388

Test sample:

X=-6.6:0.1:6.5 and y=-6.5:0.1:6.5.
The total number of test sample is 19881

4 Experiments

After the learning process, the hidden unites of each sub-module are 14, 14, 16, 12, 16, 13, respectively. For the same problem, the RBF-MLP networks required at least 74 RBF units to solve the two-spiral problem.



2 The real life data regression

This section compares BMNN with well-known algorithms on traditional benchmarks form various repositories. These are real life problems with many dimensions and with number of patterns from hundreds to thousands. Table 1 shows the specifications of the benchmark data sets. In our experiments, all of the inputs have been normalized into the range [-1,1] while the outputs have been normalized into [0,1].

Table 1. Specification of real life data sets

Real life problem	Train patterns	Test patterns	Input dimensions
Abalone	2000	2177	8
Delta ailerons	3000	4129	5
Delta elevators	4000	5517	6
Computer activity	4000	4192	8
Census	10000	12784	8
Bank domains	4500	3692	8
California housing	8000	1246	8

4 Experiments

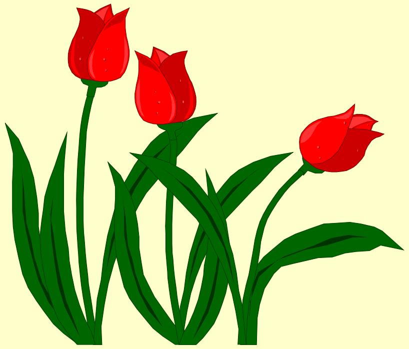
The comparison of root mean square error (RMSE) on several algorithms is shown in table 2. It can be noticed from table 2 that the test RMSE of proposed BMNN on all of the data set are smaller than the other algorithms. A comparison of training times for different algorithms on all of the data sets can be observed in Table 3.

Table 2. Test RMSE comparison of several algorithms

Real life problem	RAN	ErrCor	BMNN
Abalone	0.0978	0.0765	0.0501
Delta ailerons	0.0552	0.0431	0.0280
Delta elevators	0.0733	0.0573	0.0375
Computer activity	0.0649	0.0507	0.0328
Census	0.0905	0.0707	0.0461
Bank domains	0.0579	0.0452	0.0294
California housing	0.1434	0.1012	0.0655

Table 3. Training time comparison of several algorithms

Real life problem	RAN(s)	ErrCor(s)	BMNN(s)
Abalone	105.17	4.808	2.070
Delta ailerons	114.12	5.219	3.760
Delta elevators	131.46	5.997	3.902
Computer activity	120.94	5.519	3.153
Census	241.89	11.06	7.480
Bank domains	147.55	6.750	4.330
California housing	212.44	9.710	7.422



Thank you !