Knowledge and Information Systems (2000) 2: 115–129

Knowledge and Information Systems

Short Paper

FANNC: A Fast Adaptive Neural Network Classifier

Zhihua Zhou, Shifu Chen, Zhaoqian Chen

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, P. R. China

Abstract. In this paper, a fast adaptive neural network classifier named FANNC is proposed. FANNC exploits the advantages of both adaptive resonance theory and field theory. It needs only one-pass learning, and achieves not only high predictive accuracy but also fast learning speed. Besides, FANNC has incremental learning ability. When new instances are fed, it does not need to retrain the whole training set. Instead, it could learn the knowledge encoded in those instances through slightly adjusting the network topology when necessary, that is, adaptively appending one or two hidden units and corresponding connections to the existing network. This characteristic makes FANNC fit for real-time online learning tasks. Moreover, since the network architecture is adaptively set up, the disadvantage of manually determining the number of hidden units of most feed-forward neural networks is overcome. Benchmark tests show that FANNC is a preferable neural network classifier, which is superior to several other neural algorithms on both predictive accuracy and learning speed.

Keywords: Adaptive resonance theory; Fast learning; Field theory; Incremental learning; Machine learning; Neural networks

1. Introduction

Neural networks is an interdisciplinary research area. As an important component of artificial intelligence, it has become the common focus of artificial intelligence, cognitive science, psychology etc. Since the last decade, this area has attracted the attention of large numbers of researchers. The theoretical models, learning algorithms, practical applications and other aspects of artificial neural networks (ANNs) have been widely explored, and remarkable achievements have been attained. Those have richly proven that ANNs simulating the calculating ability

Received 10 Feb 1999
Revised 21 Jun 1999
Accepted 11 Oct 1999

of biological neural networks is superior to traditional algorithms in many ways, such as self-organization, self-adaptation, association and learning.

However, there are still serious limitations existing in most neural models at present. For example, as to associative memory (Kohonen, 1977), the memory volume is too small to avoid interference and degeneration of the stored information; as to backpropagation (Rumelhart et al, 1986), the number of hidden units is hard to determine, the training time cost is too large, online learning is impossible, and falling into local minima is often encountered. This has stimulated deeper research into theories and models of ANNs in this phase.

In this paper, a new neural learning model named FANNC, which organically exploits the advantages of both adaptive resonance theory (ART) (Grossberg, 1976) and field theory (Wasserman, 1993), is proposed. FANNC is designed to deal with classification tasks. It needs only one-pass learning, and achieves not only high predictive accuracy but also fast learning speed. The learning course of FANNC is performed in an incremental style. When new instances are fed, it does not retrain the whole training set as most feed-forward algorithms do. Instead, it could learn the knowledge encoded in the instances through slight adjustment of the topology, that is, adaptively appending one or two units and corresponding connections to the existing network when necessary. Furthermore, since the network architecture is adaptively set up, the disadvantage of manually determining the number of hidden units of most feed-forward networks is overcome. Benchmark tests show that FANNC is superior to several other algorithms on both predictive accuracy and time cost.

The rest of this paper is organized as follows. In Section 2, we summarize ART and field theory; in Section 3, we present the FANNC model; in Section 4, we report on benchmark tests and comparisons with FANNC; finally, in Section 5, we conclude and indicate some issues for future work.

2. Background

2.1. Adaptive Resonance Theory

ART is an important family of competitive neural learning models. Its memory mode is very similar to that of the biological case, and memory capacity can increase while the learning patterns increase. It can perform real-time online learning, and can work under a nonstationary world.

The classical adaptive resonance theory model ART1 (Carpenter and Grossberg, 1988) is depicted in Fig. 1, where F_1 and F_2 respectively denote the feature representation field and the category representation field. They store the active patterns of short-term memory, while the bottom-up and top-down paths between them store the contents of the adaptive long-term memory.

It has been mathematically proved (Carpenter and Grossberg, 1987a) that an ART1 architecture is capable of stably learning a recognition code in response to an arbitrary sequence of binary input patterns until it utilizes its full memory capacity. This architecture encodes a new input pattern by changing the connective weights leading from F_1 to F_2, while the paths leading from F_2 to F_1 play the role of learned expectations. Through the focus control set by top-down expectation to bottom-up information, ART1 is able not only to avoid learned knowledge being covered by new information, but also to extend learned knowledge consistently.

Since ART1, a series of models based upon ART have been developed, such as

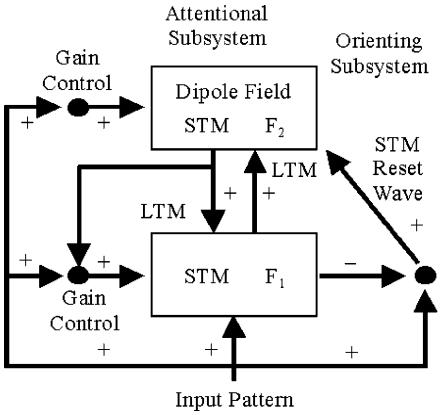

Fig. 1. Classical adaptive resonance theory model ART1.

ART2, ART3 (Carpenter and Grossberg, 1987b, 1990), ARTMAP, Fuzzy ART, Fuzzy ARTMAP (Carpenter et al, 1991a, b, 1992) and FTART (Zhou et al, 1999), which have made ART become an important family of neural models.

2.2. Field Theory

Field theory is named from CPM (coulomb potential model) (Bachmann et al, 1987). But some researchers had already investigated these kinds of model before Bachmann et al (Reilly et al, 1982). Those fruits are called field theory methods at present, as well as neural algorithms belonging to this kind developed later (Wasserman, 1993).

Field theory is a kind of relaxation model, i.e., its algorithm minimizes an energy function. In one of its many forms, the test instance is represented as a freely moving positive charge, analogous to a test charge in electrostatic theory. The test charge is allowed to move under the influence of the net electrostatic field created by the training charges, which represent training instances. Eventually it is captured by one of the training charges, pulled into its position, and assigned its classification.

Field theory is a kind of neural model that needs just one-pass learning. It can perform real-time supervised learning with fast speed, and no spurious responses will be produced regardless of the number of memories stored in the network. Hence, it is a kind of preferable heteroassociative pattern classifier fitting for tasks that need fast learning.

3. FANNC Model

3.1. Architecture

The FANNC network is composed of four layers of units. Figure 2 illustrates its architecture.

The activation function of hidden units is the *sigmoid* function, and *Gaussian* weights connect the input units with the second-layer units. FANNC uses the

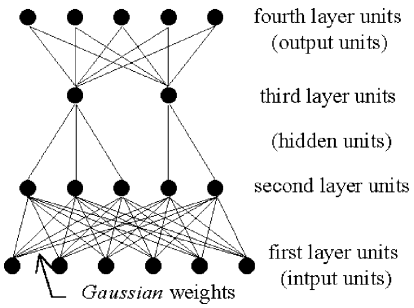

Fig. 2. The architecture of FANNC.

second-layer units to classify inputs internally and the third-layer units to classify outputs internally, and sets up associations between those two layers to implement supervised learning.

Except for the connections between the first- and second-layer units, all connections are bidirectional. The feedback connections, whose function is just to transmit feedback signal to implement competition and resonance, are always set to 1.0.

3.2. Learning Course

The initial network is composed of only input and output layers, whose unit number is respectively set to the number of components of the input and output vector. In particular, the unit number of hidden layers is zero. This is different from some other neural algorithms that configure hidden units before the start of the learning course. While instances are fed, FANNC will adaptively append hidden units so that the knowledge encoded in those instances could be learned. The unit-appending process will terminate after all the instances are fed. Thus, the topology of FANNC is always adaptively changing during the learning course.

When the first instance is fed, FANNC appends two hidden units to the network: one in the second layer and one in the third layer. Those two units are connected with each other; the feed-forward and feedback connections are all set to 1.0. The third-layer unit is connected with all the output units, the feed-forward connections are respectively set to the output components of current instance and the feedback connections are all set to 1.0. The second-layer unit is connected with all the input units through *Gaussian* weights. The responsive centers are respectively set to the input components of current instance, and the responsive characteristic widths are set to a default value.

FANNC introduces the notion of the attracting basin, which is proposed by field theory (Wasserman, 1993). An attracting basin is a region created by a training vector. If a test vector falls into the region, it will be captured by the training vector, that is, the former will be labeled with the classification of the latter. In FANNC each second-layer unit defines an attracting basin by responsive centers and responsive characteristic widths of *Gaussian* weights connecting with it. Thus, FANNC constructs its first attracting basin while the first instance is being fed. Also, it will add or move basins according to later instances.

Assuming that instances input to the input units are $A_k = (a_k^1, a_k^2, \ldots, a_k^n)$

($k = 1, 2, \ldots, m$), where k is the index of instance, and n is the number of input units. The value input to the second-layer unit j from the first-layer unit i is

$$bIn_{ij} = \exp - \left(\frac{a_i^k - \theta_{ij}}{\alpha_{ij}} \right)^2 \tag{1}$$

where θ_{ij} and α_{ij} are respectively the responsive center and the responsive characteristic width of the *Guassian* weight connecting unit i with unit j. The relationship between bIn_{ij} and other parameters satisfies

$$\begin{cases} a_i^k \to \theta_{ij} \Rightarrow bIn_{ij} \to 1 \\ |a_i^k - \theta_{ij}| \to \infty \Rightarrow bIn_{ij} \to 0 \end{cases} \tag{2}$$

Since the dynamical property of a *Guassian* weight is entirely determined by its responsive center and responsive characteristic width, learned knowledge can be encoded in the weight only through modifying those two parameters. Thus, during the training process, if the input pattern is located in an attracting basin that is determined by θ_{ij}'s and α_{ij}'s of (1), the basin will be slightly moved so that its center is closer to the input pattern. Else the nearest attracting basin will be found and modulated according to the relationship between the input pattern and the original basin, so that the basin could cover the input pattern.

The second-layer unit j computes its activation value according to

$$b_j = f \left(\sum_{i=1}^{n} bIn_{ij} - \theta_j \right) \tag{3}$$

where θ_j is the bias of unit j. f is the *sigmoid* function shown in (4):

$$f(u) = \frac{1}{1 + e^{-u}} \tag{4}$$

A leakage competition[1] is carried out among all the second-layer units. The outputs of the winners are transferred to related third-layer units. The activation value of the third-layer unit h is computed according to

$$c_h = f \left(\sum b_j v_{jh} - \theta_h \right) \tag{5}$$

where b_j is the activation value of the second-layer unit j, which is not only a winner in its competition but also connecting with unit h. v_{jh} is the feed-forward weight connecting unit j to unit h. Attention should be paid so that v_{jh} is always 1.0. θ_h is the bias of unit h. f is still the *sigmoid* function.

Next, a winner-take-all style competition is carried out among all the third-layer units. The output of the winner is transferred to the fourth-layer units. The activation value of the output unit l is computed according to

$$d_l = c_h w_{hl} \tag{6}$$

where d_l is the activation value of the output unit l. c_h is the activation value of the third-layer winner h. w_{hl} is the feed-forward weight connecting unit h to unit l.

If the network is not in training, a winner-take-all style competition is carried

[1] In leakage competition, if the activation value of a unit is greater than a certain threshold, it will be a winner. So, there may exist more than one winner at the same time.

out among the entire output units. Also, the classification represented by the winner is activated while the others are all inhibited. Else the adjusting phase is encountered as follows.

The error between real network output and expected output is computed. Here we use the average squared error as the measure:

$$Err = \frac{1}{q}\sum_{l=1}^{q}(d_l - d_l^k)^2 \tag{7}$$

where q is the number of the output units. d_l is the real output of the output unit l and d_l^k is its expected output.

If the error Err is in the allowable range, it means that current instance is covered by an existing attracting basin. In addition, the instance should be regarded as a more typical pattern of the center of the basin. Thus, the winning output unit gives out a stimulus signal and feeds it back layer by layer through feedback connections to the second-layer unit whose activation value is the maximum among those connecting with the winning third-layer unit. The responsive centers and responsive characteristic widths of the *Gaussian* weights connecting with the selected second-layer unit are adjusted according to (8) and (9). The effect of those two empirical equations is to move the center of the corresponding attracting basin closer to the input pattern.

$$\theta'_{ij} = \begin{cases} \dfrac{\theta_{ij} + 0.3\alpha_{ij} + a_i^k}{2} & a_i^k \in (-\infty, \theta_{ij} - 0.3\alpha_{ij}) \\ \theta_{ij} & a_i^k \in [\theta_{ij} - 0.3\alpha_{ij}, \theta_{ij} + 0.3\alpha_{ij}] \\ \dfrac{\theta_{ij} - 0.3\alpha_{ij} + a_i^k}{2} & a_i^k \in (\theta_{ij} + 0.3\alpha_{ij}, +\infty) \end{cases} \tag{8}$$

$$\alpha'_{ij} = \begin{cases} \dfrac{3\alpha_{ij} - (10a_i^k - 10\theta_{ij})}{6} & a_i^k \in (-\infty, \theta_{ij} - 0.3\alpha_{ij}) \\ \alpha_{ij} & a_i^k \in [\theta_{ij} - 0.3\alpha_{ij}, \theta_{ij} + 0.3\alpha_{ij}] \\ \dfrac{3\alpha_{ij} + 10a_i^k - 10\theta_{ij}}{6} & a_i^k \in (\theta_{ij} + 0.3\alpha_{ij}, +\infty) \end{cases} \tag{9}$$

If the error Err is beyond the allowable range, it means that the current instance is not covered by any existing attracting basins. We must find out whether we could cover this instance through slightly adjusting some basins, or whether we need to construct a new basin according to the instance. Furthermore, if it is the latter situation, we must find out whether we could exploit existing internal output classifications represented by existing third-layer units or not. Thus, the error between the expected network output and the characteristic third-layer unit output, which is represented by the feed-forward connections between the third-layer and output units, is computed according to

$$Errc_h = \frac{1}{q}\sum_{l=1}^{q}(w_{hl} - d_l^k)^2 \tag{10}$$

where $Errc_h$ is the characteristic error of the third-layer unit h. q is the number of output units. w_{hl} is the feed-forward weight connecting unit h to output unit l. d_l^k is the expected output of unit l.

The unit u whose characteristic error is the minimum among the entire third-layer units is selected. It satisfies

$$Errc_u = \underset{h=1}{\overset{p}{MIN}}(Errc_h) \tag{11}$$

where p is the number of third-layer units.

If the error $Errc_u$ is in the allowable range, it means that the internal output classification represented by unit u is applicable to the current instance. Also, it is the internal input classification represented by the second-layer units that should be adjusted. Thus, the unit t whose activation value is maximum among those connecting with unit u is selected. It satisfies

$$b_t = \underset{j=1}{\overset{o}{MAX}}(b_j) \tag{12}$$

where b_j is the second-layer unit activation value that was computed according to (3). o is the number of second-layer units connecting with unit u.

If unit t is a winner in its competition, it means that although the attracting basin represented by it could not cover current instance at present, it could be adjusted to do so. The strategy we used is to move the selected basin toward the input patterns, until the latter was covered. Thus, the responsive centers of the *Gaussian* weights connecting with unit t are modified according to

$$\theta'_{ij} = \theta_{ij} + \delta(a_i^k - \theta_{ij}) \qquad 0 < \delta < 1 \tag{13}$$

where θ_{ij} is the responsive center of the *Gaussian* weight connecting with unit t. a_i^k is the kth input component of the current instance. δ is the responsive center adjustment step. Experiments show that $\delta = 0.1$ could achieve preferable results.

Attention should be paid to that the adjustment of the attracting basin is a resonant process, in which the second layer and the third layer of FANNC are respectively corresponding to the feature representation field and the category representation field of ART1 that is depicted in Fig. 1. The analog of the ART1's top-down learned expectation in FANNC is the winning state of the selected third-layer unit u, which is determined according to (11). The analog of ART1's bottom-up information in FANNC is the feed-forward value of the selected second-layer unit t, which is determined according to (12).

At the beginning of the adjustment, the attracting basin corresponding to unit t moves a step according to (13), and all the winning second-layer units transfer their activation values to the third-layer units. If unit u cannot win its competition, that is, the learned expectation is not matched with the bottom-up information, unit u will release a signal and feed it back to unit t through feedback connections. Then, unit t moves another step, and resonance occurs. This resonant process terminates only when the feed-forward value of unit t enables unit u to win its competition, that is, the bottom-up information matches with the learned expectation.

The adjustment of the attracting basin described above involves not only feedback signals but also iterative moving. However, since the center of the selected attracting basin becomes closer and closer to the input pattern while moving continues, the adjusting resonance is due to stabilize at a point where the input pattern is covered by the attracting basin. This stabilization property is an advantage that FANNC inherits from ART.

On the other hand, if unit t determined by (12) is not a winner in its competition, it means that there exists no attracting basin that could cover the

current instance through only slightly adjustment. However, the internal output classification represented by unit u determined by (11) could still be exploited while creating a new attracting basin. Thus, a new unit is appended to the second layer. It is connected not only with unit u but also all the input units. The feed-forward and feedback connections between the new unit and unit u are all set to 1.0. the responsive centers of the *Gaussian* weights connecting the new unit with input units are respectively set to the input components of the current instance, and the responsive widths are set to a default value.

If the error $Errc_u$ computed according to (11) is beyond the allowable range, it means that both the internal input classification and the internal output classification represented by existing hidden units are inadequate for current instance. Thus, two units are appended to the hidden layers: one in the second layer, the other in the third layer. The new second-layer unit is connected with all the input units. The responsive centers of the *Gaussian* weights are respectively set to the input components of current instance, and the responsive characteristic widths are set to a default value. The new third-layer unit is connected with all the output units. The feed-forward connections are respectively set to the output components of the current instance, and the feedback connections are set to 1.0. In addition, the two new units are connected with each other. All the feed-forward and feedback connections between them are set to 1.0.

The process of learning an instance fed to FANNC is accomplished by this. If there is an instance that has not been fed, FANNC starts to deal with it. Else the learning course terminates. Thus it can be seen that the learning of FANNC is performed in an incremental style, and the instances are fed in only one pass. Figure 3 shows the flow chart of the learning course.

4. Benchmark Tests and Comparisons

4.1. Methodology

We have performed two benchmark tests on FANNC. Both attained satisfactory results. In order to exhibit the superiority of FANNC, we compared it with several other neural algorithms, that is, Fuzzy ARTMAP (Carpenter et al, 1992), CPM (Bachmann et al, 1987) and SuperSAB (Tollenaere, 1990).

Fuzzy ARTMAP is a representative of ART. It combines the advantages of ARTMAP (Carpenter et al, 1991a) and Fuzzy ART (Carpenter et al, 1991b), and could perform incremental supervised learning in response to arbitrary sequences of analog or binary input vectors. In our experiments, the parameters of Fuzzy ARTMAP are set to values that were suggested by Carpenter et al, that is, the baseline vigilance $\bar{\rho}_a$ is always set to 0.95, and the training phase does not terminate until the training set accuracy achieves 100% (Carpenter et al, 1992).

CPM is a representative of field theory. It performs classification according to the similarity between training vectors and test vectors by supervised learning. The training speed of CPM is very fast. However, it is quite slow in predicting. The reason is that although CPM does not perform iterative training, its predicting is an iterative procedure. In our experiments, parameters of CPM are set to values that were suggested by Wasserman; that is, the accelerating step λ is always set to 0.15, and the perturbing constant e is set to 1.0×10^{-12} (Wasserman, 1993).

Considering that backpropagation is the most prevailing neural algorithm at present, an algorithm belonging to this kind is also included in the comparison.

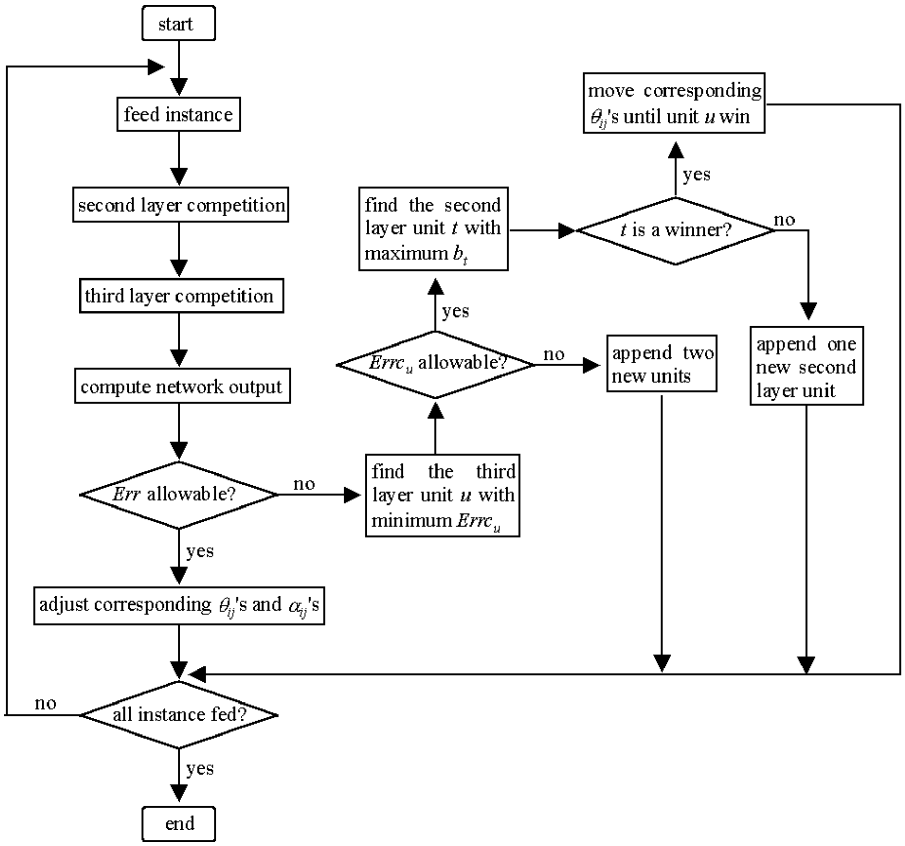

Fig. 3. Flow chart of the learning course of FANNC.

The algorithm we selected is SuperSAB, which is one of the fastest variations of backpropagation. Tollenaere (1990) reported that it is 10–100 times faster than standard BP (Rumelhart et al, 1986). In our experiments, the parameters of SuperSAB are set to values that were suggested by Wasserman; that is, the maximum weight step η_{ij} is always set to 10, and the weight increase factor η_{up} and weight decrease factor η_{down} are respectively set to 1.05 and 0.2 (Wasserman, 1993). In order to avoid overfitting, the training process is terminated while there is no test set accuracy improvement in five consecutive epochs.

The machine used in our experiments is a Pentium MMX 200 MHz with 32 MB RAM. The parameters of FANNC are set as follows. The default responsive characteristic width α_{ij} of *Gaussian* weight is set to 0.25. the bias of the hidden layer unit is set to 0.3. the responsive center adjustment step δ is set to 0.1. The leakage competition threshold is set to 0.8 and the maximum permissible error is set to 0.11.

We compare the training set accuracy, test set accuracy and time cost of the selected algorithms corresponding to different training set size. Considering that although CPM has a fast training speed, it is quite slow in predicting, and only comparing the training time cost is unfair to other algorithms. Therefore, not only

the training time cost but also the test time cost is included in our comparison items.

Attention should be paid to the fact that the storage costs of these algorithms are different, owing to the different number of hidden units. CPM is nearly proportional to the size of the training set; Fuzzy ARTMAP or FANNC is adaptively increased; while SuperSAB is a constant. However, we do not include storage cost in our comparison items. The reason is that we believe it is not a problem for present computers to store a neural network, no matter how big the network is. While the most important thing concerns the predictive accuracy and time cost, we do not want relatively unimportant items to distract our attention.

4.2. Circle-in-the-Square

Circle-in-the-square was specified as a benchmark test problem for neural network performance evaluation in the DARPA ANNT (artificial neural network technology) program (Wilensky, 1990). It requires a learning system to identify which points of a square lie inside and which lie outside a circle whose area equals half that of the square. Wilensky examined the performance of $2 - n - 1$ backpropagation systems on this problem. He studied systems whose hidden unit number ranged from 5 to 100, and found that systems with 20–40 hidden units could achieve relatively preferable predictive accuracy. In order to avoid overfitting. Wilensky stopped the training process when the training set accuracy reached 90%. He found that about 5000 epochs were necessary for backpropagation to reach this point, and the test set accuracy was approximately 90% in this condition.

In our experiments, the training set size ranges from 100 to 1000, and the test set is composed of 10,000 instances generated through uniformly varying the coordinate $x, y \in [0, 1]$ in steps of 0.01. Three networks are trained with SuperSAB because the number of hidden units of backpropagation type algorithms is hard to be determined. The hidden unit numbers of those three networks are respectively set to 25, 30 and 35 based upon the experimental results of Wilensky. The data of SuperSAB cited below belongs to the network with the best test set accuracy. Our strategy to avoid overfitting, which is quite different from that of Wilensky, is to stop the training process while there is no test set accuracy improvement in five consecutive epochs. The reason is that we want to find out what is the highest test set accuracy that SuperSAB could achieve.

Table 1 shows the test results. It reveals that the training set accuracy of FANNC, Fuzzy ARTMAP and CPM are all 100%, superior to that of SuperSAB. As to the most important factor of neural networks, that is, the test set accuracy, FANNC is the best among all the four algorithms. It achieves higher than 90% while only a few training instances are fed, and achieves the highest accuracy among the four algorithms no matter what size the training set is. As to the training time cost, CPM is the lowest. However, its test time cost is the largest. The training time cost of FANNC is comparable to that of CPM, far better than Fuzzy ARTMAP and SuperSAB. The test time cost of FANNC, Fuzzy ARTMAP and SuperSAB are comparable, all far less than that of CPM. Attention should be paid to the fact that the time cost depends on the size of training set and test set. Our data is achieved through the configuration we used in the experiments.

From the summary of the experimental results given above, we can say that FANNC is the best algorithm in this benchmark test. It achieves the highest

Table 1. Test results of circle-in-the-square

	Training set size	Training set accuracy	Test set accuracy	Training time (seconds)	Test time (seconds)
FANNC	100	100%	92.0%	246	8
	200	100%	94.0%	569	9
	500	100%	97.2%	1,425	9
	1000	100%	98.1%	2,690	10
Fuzzy ARTMAP	100	100%	88.7%	502	9
	200	100%	91.1%	1,278	9
	500	100%	93.8%	4,633	10
	1000	100%	95.3%	8,271	10
CPM	100	100%	71.4%	219	19,574
	200	100%	80.5%	437	20,496
	500	100%	87.2%	1,098	21,300
	1000	100%	90.3%	2,194	21,521
SuperSAB	100	85.5%	79.3%	3,269	9
	200	89.4%	84.7%	7,011	9
	500	95.1%	92.5%	16,573	9
	1000	98.6%	96.2%	32,861	10

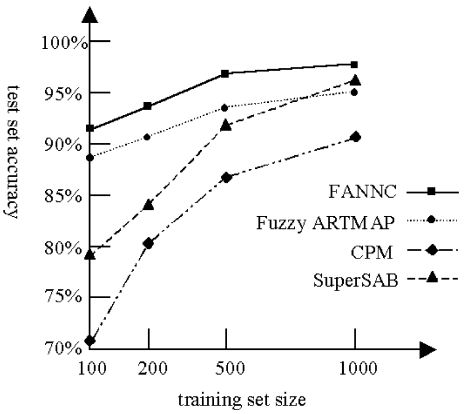

Fig. 4. Comparison of test set accuracy of circle-in-the square.

accuracy with the lowest time cost (considering the sum of training time cost and test time cost). As to the other three algorithms, we believe that the order should be Fuzzy ARTMAP, SuperSAB and CPM. The rank of SuperSAB is higher than that of CPM because when the difference of time cost is not remarkable we order algorithms according to their test set accuracy.

Figure 4 compares the test set accuracy of the four algorithms. Attention should be paid to the fact that the SuperSAB data belongs to the best one selected from three networks. Figure 5 shows the test results of FANNC while 100–1000 training instances are fed. In those figures, black and white points respectively represent instances that are identified to be in the circle and out of the circle by FANNC.

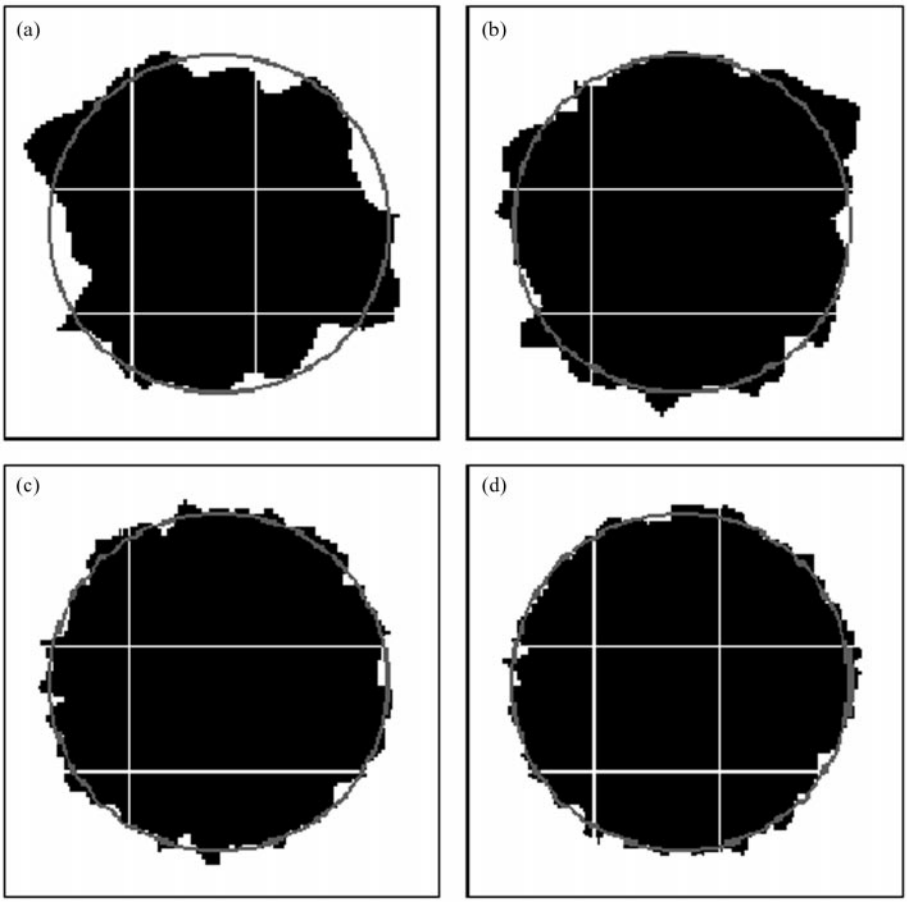

Fig. 5. Test results of FANNC while 100–1000 training instances are fed. The test set is composed of 10,000 instances generated through uniformly varying the coordinate $x, y \in [0, 1]$ in steps of 0.01. (a) 100 training instances, test set accuracy 92.0%. (b) 200 training instances, test set accuracy 94.0%. (c) 500 training instances, test set accuracy 97.2%. (d) 1000 training instances, test set accuracy 98.1%.

4.3. Telling-Two-Spirals-Apart

Telling-two-spirals-apart is a neural network benchmark test proposed by Wieland (Lang and Witbrock, 1989). Each of the two spirals of this task makes three complete turns in the plane, with 32 points per turn plus an endpoint. Thus, there are 194 training instances in sum. The training instances (x^k, y^k, b^k) $(k = 1, 2, \ldots, 194)$ are generated according to the following equations, where $n = 1, 2, \ldots, 97$, x^k and y^k respectively represent the coordinate x and y of the training point, $b^k = 1$ means it is a white point, and $b^k = 0$ means it is a black point.

$$x^{2n-1} = r_n \sin \alpha_n + 0.5 = 1 - x^{2n} \tag{14}$$

$$x^{2n} = 0.5 - r_n \sin \alpha_n = 1 - x^{2n-1} \tag{15}$$

$$y^{2n-1} = r_n \cos \alpha_n + 0.5 = 1 - y^{2n} \tag{16}$$

Table 2. Test results of telling-two-spirals-apart

	Training set accuracy	Test set accuracy	Training time (seconds)	Test time (seconds)
FANNC	100%	100%	523	1
Fuzzy ARTMAP	100%	99.1%	537	1
CPM	100%	62.9%	435	4179
SuperSAB	95.5%	94.7%	6923	1

$$y^{2n} = 0.5 - r_n \cos \alpha_n = 1 - y^{2n-1} \tag{17}$$

$$b^{2n-1} = 1 \tag{18}$$

$$b^{2n} = 0 \tag{19}$$

$$r_n = 0.4 \left(\frac{105 - n}{104} \right) \tag{20}$$

$$\alpha_n = \frac{\pi(n-1)}{16} \tag{21}$$

This benchmark test is a quite difficult task. Land and Witbrock (1989) reported that it is unable to accomplish this task using a standard backpropagation network with connections from one layer to the next. In order to solve this problem, they crafted a special 2-5-5-5-1 architecture with shortcut connections, each node being connected to all nodes in all subsequent layers. They trained the network with vanilla backpropagation and QuickProp, and found that the number of epochs that were necessary for convergence were respectively 20,000 and 8000 or so.

In our experiments, the training set is composed of 194 instances that generated from (14)–(21) with $n = 1, \ldots, 97$, and the increasing step of n is 1. The test set is also derived from those equations. The only difference is that the increasing step of n is 0.1. Thus, there are 1942 instances in the test set. The data of SuperSAB cited below belongs to a network that not only trained with SuperSAB but also had the same architecture as Lang and Witbrock's (1989).

Table 2 shows the test results. It reveals that the training set accuracy of FANNC, Fuzzy ARTMAP and CPM are all 100%, superior to that of SuperSAB. The performance of Fuzzy ARTMAP is quite similar to that of FANNC, although the latter is still slightly better than the former on both test set accuracy and time cost. The most amazing thing is that CPM only achieves a little more than 60% test set accuracy in this task. We believe that this is because the attracting basins created by CPM corresponding to training instances cannot effectively cover the whole instance space, so that many test instances fall into wrong basins under the compound effect of several basins. Since the test set accuracy of CPM is far inferior to that of the other algorithms, we have to rank it at the tail of the order list, in which FANNC, Fuzzy ARTMAP and SuperSAB have respectively occupied the first to third places.

Figure 6 shows the test results of FANNC and CPM. We do not report the images generated by Fuzzy ARTMAP and SuperSAB, whose test result is quite similar to that of FANNC, because it is very hard to distinguish them from the image generated by FANNC with the naked eye.

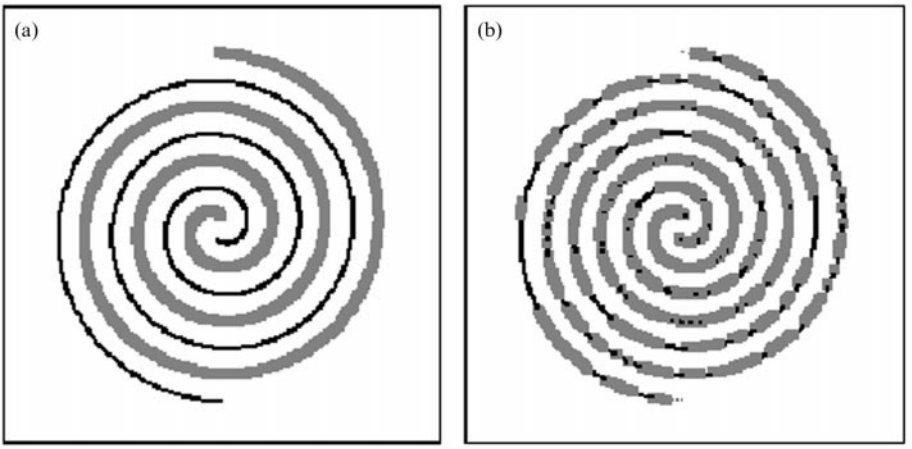

Fig. 6. Test results of FANNC and CPM. (a) Test result of FANNC, test set accuracy 100%. (b) Test result of CPM, test set accuracy 62.9%.

5. Conclusions and Future Work

This paper proposes a fast neural network classifier named FANNC, which exploits the advantages of both ART and field theory. It needs only one-pass learning, and achieves not only high predictive accuracy but also fast learning speed. Besides, FANNC has incremental learning ability, which makes it fit for real-time online learning tasks. Moreover, FANNC can adaptively set up its topology so that the disadvantage of manually determining the number of hidden units of most feed-forward neural models is overcome. Benchmark tests show that FANNC is a preferable neural network classifier, which is superior to Fuzzy ARTMAP, CPM and SuperSAB on both predictive accuracy and time cost (sum of training time and test time).

Until now, FANNC has only been applied to artificial problems such as circle-in-the-square and telling-two-spirals-apart. Although some of them are quite difficult, they are just toy tasks because the data sets are too small. In most real-world tasks, both the training set and the test set are very large, comprising tens of thousands or even millions of instances. In the near future, we plan to do further research to investigate the performance of FANNC facing such large-scale problems. If the result is satisfactory, we plan to develop some real-world neural network learning systems based upon FANNC. On the other hand, FANNC has not been applied to regression estimation tasks because it is specially designed for classification tasks. In the future, we plan to devise a neural model based upon FANNC to solve regression estimation problems with high predictive accuracy and fast learning speed.

Acknowledgements. The authors wish to thank Wenlong Wei and Dong Shao for their fruitful work. The comments and suggestions of the anonymous reviewers greatly improved this paper. The authors also wish to thank the associate editor, Dr Lipo Wang, for his kind help. The National Natural Science Foundation of P. R. China and the Natural Science Foundation of Jiangsu Province, P. R. China, supported this research.

References

Bachmann CM, Cooper L, Dembo A et al (1987) A relaxation model for memory with high storage density. Proceedings of the National Academy of Sciences, USA 21:7529–7531

Carpenter GA, Grossberg S (1987) A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Vision, Graphics, and Image Processing 37:54–115

Carpenter GA, Grossberg S (1987) ART2: self-organization of stable category recognition codes for analog input patterns. Applied Optics 26(23):4919–4930

Carpenter GA, Grossberg S (1988) The ART of adaptive pattern recognition by a self-organizing neural network. Computer 21(3):77–88.

Carpenter GA, Grossberg S (1990) ART3: self-organization of distributed pattern recognition codes in neural network hierarchies. In Proceedings of the international neural network conference (INNC '90), vol 2, Paris. Kluwer, Dordrecht, pp 801–804

Carpenter GA, Grossberg S, Reynolds J (1991) ARTMAP: a self-organizing neural network architecture for fast supervised learning and pattern recognition. In Proceedings of the international joint conference on neural networks, vol 1 (IJCNN '91), Seattle, WA. IEEE/INNS, pp 863–868

Carpenter GA, Grossberg S, Rosen D (1991) Fuzzy ART: an adaptive resonance algorithm for rapid, stable classification of analog patterns. In Proceedings of the international joint conference on neural networks, vol 2 (IJCNN '91), Seattle, WA. IEEE/INNS, pp 411–416

Carpenter GA, Grossberg S, Markuzon N et al (1992) Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Transactions on Neural Networks 3(5):698–713

Grossberg S (1976) Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. Biological Cybernetics 23:121–134

Kohonen T (1977) Associative memory: a system theoretical approach. Springer, New York

Lang KJ, Witbrock MJ (1989) Learning to tell two spirals apart. In Proceedings of the 1988 connectionist models summer school. Morgan Kaufmann, San Mateo, CA, pp 52–59

Reilly DL, Cooper LN, Elbaum C (1982) A neural model for category learning. Biological Cybernetics 45:35–41

Rumelhart D, Hinton G, Williams R (1986) Learning representation by backpropagating errors. Nature 323(9):533–536

Tollenaere T (1990) SuperSAB: fast adaptive backpropagation with good scaling properties. Neural Networks 3:561–573

Wasserman PD (1993) Advanced methods in neural computing. Van Nostrand Reinhold, New York

Wilensky G (1990) Analysis of neural network issues: scaling, enhanced nodal processing, comparison with standard classification. DARPA Neural Network Program Review (October), pp 29–30

Zhou Z, Chen Z, Chen S (1999) Research of field theory based adaptive resonance neural network. Journal of Software (forthcoming)

Correspondence and offprint requests to: Zhihua Zhou, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, P. R. China