Warships Formation Identification Based on Neural Network

Liye Tian*, Jinping Wu and Yongjie Wang
Navy Submarine Academy, Qingdao, Shandong, 266199, China
*new.tianli@163.com

Abstract. Submarine tactical software should be able to identify warships formation. Neural network based pattern recognition method is introduced in submarine tactical software to improve accuracy. This paper discussed how to transform sensor data into training samples by using Matlab, and how to choose training samples. Experiments indicate that GRNN network is capable of this work and suitable for actualizing.

1. Introduction
Submarine tactical software is researched, developed, applied for many years [1]. Tactical software is helpful for submarine officer to command and make decisions. Weapon controlling and intelligence processing also need help from tactical software. Years of progress and improvement make tactical software play an important role in warfare and training.

In tactical software, situation reasoning component and decision-making component are able to identify warships formation automatically. Known identification methods include template matching [2], domain knowledge [3], and neural network [4-5]. Mathematics based analytical method is also used. However, there are drawbacks to these methods. For example, constraint condition is too strict, fuzzy performance is poor, accuracy is low, identify result is not visible, etc [6-7]. Aiming at these problems, we suggest a RBF neural network based warships formation identification method. Experiments with common samples indicate a 100% accuracy of this method.

2. Artificial Neural Network
Artificial neural network, neural network for short, is an imitation of animal neural network behaviour. Essentially, neural network is a mathematical model, which is able to carry out distributed parallel computation [8]. By adjusting connecting weight, neural network could operate matrix. A great advantage of neural network is self-learning and self-adaption. It could automatically analysis the relationship between input and output data, after that, it could calculate a result of any new input data. The procedure of self-learning is called “training”, and the procedure of result calculating is called “simulation”. There are several kinds of neural network. Among all kinds network, RBF network can approximate any nonlinear function in any precision.

3. To Identify Warships Formation

3.1. Sample generation
Sensors such as radar, could get accurate positions of each ship in a warships formation. All the positions of warships formation constitute a data sample. The sample is used to train a RBF network, or the sample could be sent to a trained network to get simulating result. Essentially, one sample is one figure, which
describes locations of warships. However, as a matrix, neural network cannot accept figures directly. Before sending to neural network, figures need to be transformed into one dimension array.

Sensors get locations in form of longitude and latitude. Longitude and latitude can represent any point on the earth, so it’s not suitable to represent local warships formation. The formation is too small while the earth is too big, so calculation with longitude and latitude is not convenient. The focus should be a local sea area, and the only concern should be relative position of each ship. Some sensor possibly get height of the target, the height should be ignored. We suggest a method to focus on relative positions of warships formation, that is putting all ships in the formation on a 10×10 chessboard by relative positions. One edge of the chessboard represents X-axis, adjacent edge represents Y-axis. Black blank represents a ship, white blank represents a position without ship. Consequently, a 10×10 two-dimension array in computer could represents a warships formation as shown in Fig.1.

![Figure 1. Formation representation](image)

2-dimension array must be transformed into 1-dimension array before sending to neural network. Take out first column of the 2-dimension array, take out the second column to link to the first one. Then take out the third one to link to the second one, and so on. By this way, a 10×10 matrix could be transformed into an array with 100 element.

3.2. Training of RBF network

Before using, a RBF network must be trained by samples of every formation. By this way, RBF network can remember every formation. All samples used to train RBF constitute training set. It’s very important to setup a training set. It can’t be too small or too large. Samples in training set should be representativeness. Every formation’s sample need to be chosen into the set. Although RBF network is fuzzy, manual setting some error in training set is necessary. For example, in a row-formation constituted by ships, one ship is not in a straight line with others. However this is still a row-formation. If the training set is too small, the training will be less useful, RBF network may cannot remember each formation clearly. If the training set is too big, the over-fitting happens. Experiments indicate that RBF network doesn’t need too many samples to get a good result.

Samples from each kind of formation are manually set as following figure.
Figure 2. Samples of row, column, reverse-V, V formation

Transform these samples into 1-dimension array by the above-mentioned way, and setup the training set. In order to increase the amount of training set, every sample appears twice in the training set. While training a RBF network, Training target should be set to tell which formation the sample belongs to. Matlab R2010a is used to do the research and experiment. In Matlab, many kinds of RBF network are available, and GRNN network is chosen. GRNN network contains a competitive function, so that data processing and expressing are very convenient. When setup the training target, a sample index number is only required. Row is 1, column is 2, reverse-V is 3, V is 4. In Matlab, use the following instructions to transform index number into target:

T=ind2vec(target);
net=newgrnn(P,T);

Target means index, P is training set, T is target. Under this scale of training set, the training may be done in seconds by normal hardwares. Training parameters is set to default value by Matlab. After training, GRNN structure and parameters is shown in Fig.3. We pick up some samples from training set to test this GRNN, it shows that the GRNN network can identify training set 100% correctly.

Figure 3. GRNN structure and parameters

4. Simulation and Result

A trained GRNN network can be used to warships online identification. Test samples are shown in Fig.4. Transform test samples into array-form by the above method. Use the following Matlab instructions to send it to GRNN network:

Ytest=sim(net,sampletest);

Net is the trained GRNN network, sampletest is new sample. GRNN network will give a simulation result automatically. The result is an index number which means a possible formation.

Some nonstandard samples are set manually to test GRNN network. In this way, GRNN’s fuzzy performance can be tested. Following samples can be recognized by human easily, they are row, column, reverse-V.
Table 1. Experiment results.

| Test Sample | Formation | Simulation Result |
|-------------|-----------|-------------------|
|             | RBF       | GRNN              | PNN              |
| 1           | 4         | 1                 | 1                |
| 2           | 4         | 2                 | 2                |
| 3           | 4         | 3                 | 1                |
| Accuracy    | 0%        | 100%              | 67%              |

Experiment results are listed in Table 1. It indicates that trained GRNN network can identify these test samples correctly. For contrast, a normal RBF and PNN network are also tested. PNN gets a low accuracy while RBF cannot identify correctly at all. This is because the training set is too small. The RBF and PNN structures are shown in Fig.5. GRNN network shows a nice fuzzy performance. More nonstandard samples are tested, most samples can be identified correctly. Those samples too different from standard ones cannot be identified correctly. Actually, warships usually maintain correct position in the formation, so those samples are seldom seen. Generally speaking, the method based on neural network can be used in tactical software.

5. Problems and future work
Research in this paper, use GRNN network, which is a kind of RBF network. Theoretically, other kind neural networks are able to do this work also. What is the actual situation, how’s the capability, is it suitable to be taken into practice, these question need further research.

Research in this paper only consider 4 kinds of warships formations, row, column, reverse-V, V. More kinds of formation need to be added before taken into practice, such as wedge, reverse-wedge, circle, semi-circle, etc. More kinds of formation bring bigger training set. Some training samples are similar, for example, semi-circle and wedge. It’s more difficult for neural network to identify warships formation. The influence exerted by this need further research.

The size of chessboard is set to 10×10, what happens if we set to 20×20, 30×30? The describing ability will increase a lot. Similarity between different samples decreases a lot. It helps to identify
different formations. Nevertheless, computation complexity will increase and the performance of the neural network will decrease. What is the best size of chessboard, this need further research.

Sensor get positions of warships, and the positions are transformed into locations on chessboard. This is the foundation of our research. How to transform these information, if there is a better method, need further research.

Research in this paper is done with Matlab, trained GRNN network need to be exported into an independent software module. So that tactical software could call it. Or, while developing, program a neural network without Matlab.

In our research, training parameters are set to default value by Matlab. In future work, other parameters should be tested to find out more efficient ones.

References
[1] Huang W.B., Lin S.L., Lu M.H. (1995) A brief of submarine tactical software research and development. Submarine Academic Research, 1.
[2] Qiao B., Xiao B. (2006) Identification of combat warship formation based on template matching. Computer Simulation, 23: 4-6.
[3] Leng H.P., Guan Q.Y., Wu X.F. (2013) Recognition of naval fleet line type formation based on domain knowledge. Ship Science and Technology, 35: 103-106.
[4] Wen X., Li X., Zhang X.W. (2015) The Application of MATLAB Neural Network. National Defence Industry Press, Beijing.
[5] Tian L.Y., Wang D.M., Bao K. (2017) Submarine tactical software formation identification method. Submarine Academic Research, 35: 39-41.
[6] Dong S.Q., Song W. (2010) Identification algorithm of combat warship formation based on similarity between spatial directions. Fire Control & Command Control, 35: 167-169.
[7] Zhang Y. Research on pattern recognition methods based on swarm intelligence. Dalian university of technology, 2007.
[8] Zhu K., Wang Z. (2010) Master Matlab neural network. Electronic Industry Press, Beijing.