ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 5, May 2014

Study On Object Tracking Using Neural Network Functions

Harsha K[1], Kavith S N[2], Dr. S C Prasanna Kumar[3]

Student, Department of Information science, R. V. College of Engineering Bangalore, India[1]

Assistant Professor, Department of Information science, R. V. College of Engineering Bangalore, India[2]

Professor and HOD Department of Instrumentation Technology, R. V. College of Engineering Bangalore, India[3]

Abstract: In this article we present the properties of two types of neural networks: radial basis function networks and feed-forward network. In this paper, the advantages and disadvantages of the two types of neural network architectures are shown based on examples. The examples indicate approaches to be taken relative to the network model selection for applications

Keywords: tracking; neural network; hidden layer;

I. INTRODUCTION

Artificial neural networks (ANN) are distributed systems composed of several processing units called artificial neurons that calculate with certain mathematical functions. The design of ANNs was motivated by the structure of a real brain. Artificial neurons are the smallest unit in the neural network. These units are arranged in one or more layers and interconnected by a large number of connections. An artificial neural network (ANN) has the ability to learn from experiences, improving its performance and adapting to the changes in the environment. ANN approach is convenient when an analytical model is difficult to obtain.

Considering the potential of this approach, this paper aims to establish a comparison between Multilayer Feed-Forward network (FFN) and a Radial Basis Function Network (RBF) for object tracking. The RBF and FFN networks are usually used in the same kind of applications (nonlinear mapping approximation and pattern recognition), however their internal calculation structures are different. Traditional neural networks need to be well trained before it is applied for applications. Training data can be directly applied as network inputs, and the networks parameters, called "weights", are adjusted iteratively according with the differences between desired network behaviors and actual network behaviors.

Tracking is basic task for several applications of computer vision, e.g., video monitoring, robotics, target tracking, automated surveillance, and so on. The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video. Some of the important challenges encountered in visual tracking are non-rigid objects, complex object shapes, occlusion, scale/appearance change of the objects and real-time processing. RBF networks are trained by a two-step algorithm. In the first step, the center vectors of the RBF functions in the hidden layer are chosen. This step can be performed in several ways; centers can be randomly sampled from some set of examples, or they can be determined using k-means clustering. The second step simply fits a linear model with coefficients to the hidden layer's outputs with respect to some objective function. The FFN consists of neurons which can be divided into three classes: input neurons, hidden neurons and output neurons.

There are many training algorithms for FFN, we choose back propagation for both FFN and RBF training. Feature extraction and image segmentation methods appropriate to the radial basis neural network and feed forward network is employed for tracking the objects. The methods are chosen such that the computations required for the tracking of object in the training phase i.e. in the first frame of the video sequence is very low for both the RBF and FFN networks.

RBF object tracker uses an optimized k-means algorithm for image segmentation and mean shift method for tracking, FFN object tracker uses Monte Carlo Joint Probabilistic Data Association (JPDA) algorithm[12],[13] for detecting the object by matching the object features and the features in the frame so that they can be handled by ANN. Here the neural network is feed-forward neural network.

The paper is organized as follows. In the section II, the basic concepts of Radial Basis Function neural networks are introduced briefly. Section III presents the computational fundamentals of Feed-Forward neural networks.

In section IV, FFN networks and RBF networks are tested based on examples that are appropriate to the network model. Section V is the conclusion.

II. RADIAL BASIS FUNCTION NETWORKS

Fig. 1 shows the general form of RBF networks, with *N* inputs, *L* hidden units and *M* outputs.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 5, May 2014

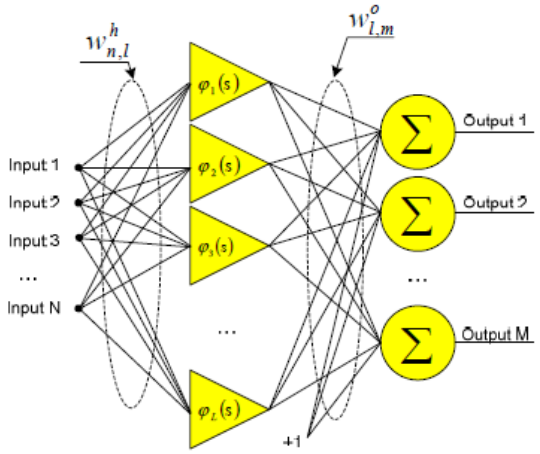


Fig. 1 RBF network: N inputs, L hidden units and M outputs.

The basic computations in the RBF network above include:

(i) Input layer computation

At the input of hidden unit l, the input vector **x** is weighted by input weights $\boldsymbol{w}^h$:

$$S_l = [x_1 w^h_{1,l}, x_2 w^h_{2,l} \cdots x_n w^h_{n,l}] \quad (1)$$

Where: n is the index of input; l is the index of hidden units;
x_n is the n-th input; $w^h_{n,l}$ is the input weight between input n and hidden unit l.

(ii) Hidden layer computation

The output of hidden unit l is calculated by:

$$\varphi l(Sl) = exp\left(-\frac{\|Sl - Cl\|2}{\sigma l}\right) \quad (2)$$

Where: the activation function $\varphi_l(\bullet)$ for hidden unit l is normally chosen as Gaussian function; $\boldsymbol{c}_l$ is the center of hidden unit l and σ_l is the width of hidden unit l.

(iii) Output layer computation

The network output m is calculated by:

$$o_m = \sum_{l=1}^{L} \varphi_l(\mathbf{s}_l) w^o_{l,m} + w^o_{0,m} \quad (3)$$

Where: m is the index of output; $w^o_{l,m}$ is the output weight between hidden unit l and output unit m; $w^o_{0.m}$ is the bias weight of output unit m.

From basic computations (1), (2), and (3), one may notice that there are four types of parameters, input weight matrix $\boldsymbol{w}^h$, output weight matrix $\boldsymbol{w}^o$, center matrix $\boldsymbol{c}$ and width vector $\boldsymbol{\sigma}$. Normally, the input weights are all set as "1". The simple linear least squares (LS) method can only adjust the output weights and it performs for nonlinear cases.

Iteratively LS method [1] improves the nonlinear performance of output layer. First order gradient methods [2] can adjust output weights, widths and centers during the training process, but they often take long time for convergence and have limited search ability.

III. FEED-FORWARD NEURAL NETWORKS

The FFN consists of neurons which can be divided into three classes: input neurons, hidden neurons and output neurons.

In the following annotations the layer will separated by "-" and the hidden layer will be marked by "[ ]". An artificial neuron has -analog to his biological archetype - an activation function. Therefore we have chosen the symmetrically tangent hyperbolic sigmoid-function for the hidden and output neurons as proposed in [9]. The input neurons represent only the input value. Fig 2. Shows the basic architecture of FFN:

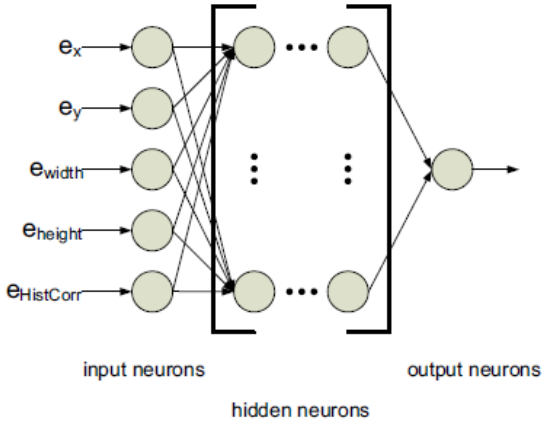


Fig 2. Architecture of a feed-forward neural network with regard to presented approach

The FFN consists of neurons which can be divided into three classes: input neurons, hidden neurons and output neurons.

This is depicted in Fig. 2. The connecting lines represent the weighted link of an output gate to the input gate of a neuron of the following layer. The output value O is defined by an activation function A. We chose the symmetrically tangent hyperbolic sigmoid-function for the hidden and output neurons as proposed in [8]. The parameters of A_j of a neuron j is the weighted sum of the output e_i of a neuron i of the former layer. Additional a bias Θ_j has to be considered. A formal description is depicted by Eq. 4.

$$O_j = A_j\left(-\Theta_j + \sum_i w_{j,i} \cdot e_{j,i}\right) \quad (4)$$

During the training the training-algorithm explores the weighting values $w_{j;i}$ of each connection and the bias values Θ_j to get the maximum performance. Each neuron in one layer has directed connections to the neurons of the subsequent layer. Multi-layer networks use a popular being *back-propagation*. The output values are compared with the correct answer to compute the value of some predefined error-function. The error is then fed back

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 5, May 2014

through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small.

Computations related with the single neuron include:

i) Net computation

$$net = \sum_{n=1}^{N} x_n w_n + w_0 \quad (5)$$

Fig. 3 shows the basic unit of traditional neural networks, with *N* inputs and *M* outputs.

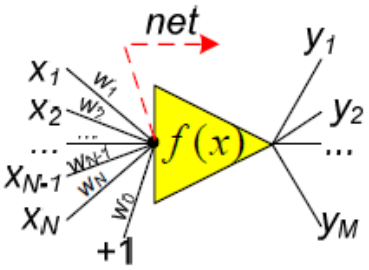


Fig 3. Single neuron: N inputs and M outputs

Where: n is the index of inputs and weights, from 1 to N; w_n is the weight on input x_n; w_0 is the bias weight.

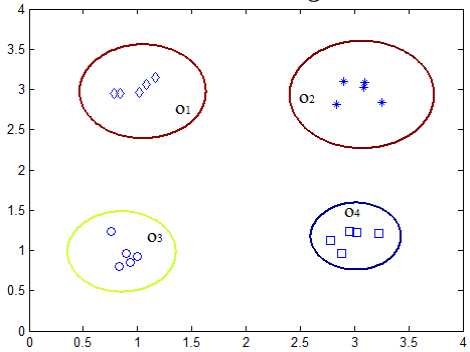
ii) Output computation

$$y_m = f(net) = \tanh(net) \quad (6)$$

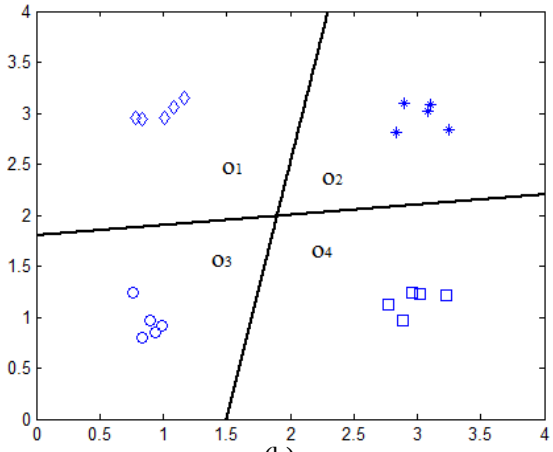
Where: y_m is the output of the neuron; $f(\bullet)$ is the activation function and normally chosen as sigmoidal shape. For more neurons interconnected together, the two basic computations (5) and (6) for each neuron remain the same; while the only difference is that the inputs of a neuron could be provided by either the outputs of neurons from previous layers or network inputs. Weight values are the only type of parameters and can be updated by learning algorithms. Based on error back propagation procedure, various gradient algorithms are developed for traditional neural network learning.

IV. COMPARISON

In this section the two neural network models are compared with respect to their approach to segment data and the tracking of objects with two different videos. The comparative data segmentation of RBF network and FFN network can be seen below in the Fig 4:



(a)



(b)

Fig 4. Different classification mechanisms for pattern classification in 2D space: (a) RBF network; (b) FFN network

In the simple two-dimension case as shown in Fig. 2, the RBF network in Fig. 4a separates the four clusters by circles or ellipses; while the FFN network does the separation by lines (Fig. 4b).

i) Evaluation of Different Architectures of Feed-Forward Neural Networks

Several architecture models for the ANN were analyzed differing in the number of hidden layers and the neurons per layer. The different network architectures which we evaluated are listed on the left in Tab. I. The corresponding number of failures of each architecture is listed on the right. The networks 4-[6]-1 and 4-1 provided the highest accuracy.

TABLE I
DIFFERENT NETWORK TYPES AGAINST THEIR NUMBER OF FAILURES IN VIDEO

ANN	Number of failures
4-1	89
4-[6]-1	77
4-[13]-1	102
4-[2]-1	127
4-[4]-[2]-1	463

Fig 5. below Shows the snapshot of the video for which the FFN network with different architectures was tested

Fig 5. Scene of football

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 5, May 2014

ii) Evaluation of Radial basis function Neural Networks

A study on how the classification performance of RBFN classifier varies against the number of hidden neurons presented, similar to the FFN network above.

The first frame of typical tracking scenario is used for training, while the second frame is used for testing. In order to calculate the classification accuracy of the object Equation (7) is used:

$$\text{Object Detection Rate (\%)} = (A/B)\times 100 \quad (7)$$

Where A is the number of correctly detected object pixels; B is the number of object pixels extracted manually. The number of hidden neurons was varied from 8 to 11 and the performance was analyzed and reported in Table II for a typical target.

Table II: Object Detection Rate for first frame (training) and Object Detection Rate for second frame (testing), as the number of hidden neurons (k) increases.

k/ Object detection rate	training (First frame)	testing (Second frame)
8	74.43%	70.56%
9	76.23%	71.44%
10	76.50%	72.47%
11	77.71%	73.87%

Fig 6. From left to right and up to down shows tracking result for proposed method (dashed red), Mean shift method (dash-dot blue) and ground truth (solid green)

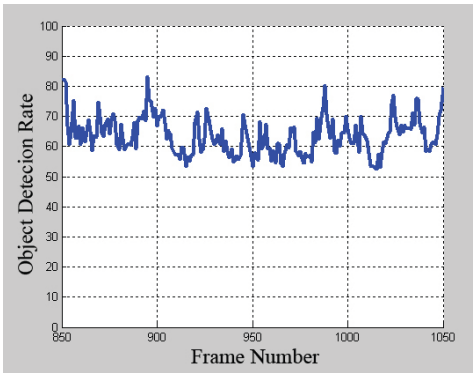


Fig 7. Object detection rate of proposed approach for sequence shown in Fig. 6.

V. CONCLUSION

The paper presents the comparison of FFN networks and RBF networks based on two examples. According with the comparing results and properties of two types of neural networks, the conclusions below can be made to provide suggestions for network model selection:

- RBF network shown, can successfully handle partial occlusion of object but for handling full occlusion of object the proposed method can be combined with prediction methods
- RBF is not as efficient as FFN network with regard to multiple object tracking

REFERENCES

[1] B. M. Wilamowski, "Modified EBP Algorithm with Instant Training of the Hidden Layer," Proceedings of Industrial Electronic Conference (IECON'97), New Orleans, November 9-14, 1997, pp. 1097-1101.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[3] N. B. Karayiannis, "Reformulated Radial Basis Neural Networks Trained by Gradient Descent," IEEE Trans. Neural Networks, vol. 10, issue 3, pp. 657-671, Aug. 2002.

[4] Asvadi, A.; Karami-Mollaie, M.; Baleghi, Y.; Seyyedi-Andi, H.; , "Improved Object Tracking Using Radial Basis Function Neural Networks," Machine Vision and Image Processing(MVIP), 2011 7th Iranian , vol., no., pp.1-5, 16-17 Nov. 2011

[5] A. Yilmaz, O. Javed, M. Shah, Object Tracking: A Survey, ACM Computing Surveys (CSUR) 38 (4, Article 13).

[6] D. E. Rumelhart, G. E Hinton and R. J. Williams, "Learning Representations by Back-Propagating Errors," Nature, vol. 323, pp.533-536, 1986.

[7] Y. Bar-Shalom and E. Tse, "Tracking in a cluttered environment with probabilistic data association," Automatica, vol. 1, no. 1, pp. 451–460, 1975.

[8] A. Bugeau, P. Pérez, "Detection And Segmentation Of Moving Objects In Complex Scenes," Computer Vision And Image Understanding, Vol. 113, No. 4, pp. 459–476, 2009.

[9] T. M. Mitchell, Machine Learning. Mcgraw-Hill Higher Education, 1997, chapter 4.6.2 Representational Power of Feedforward Networks, page:105.

[10] R.V. Babu, S. Suresh and A. Makur, "Online adaptive radial basis function networks for robust object tracking," Computer Vision And Image Understanding, Vol. 114, No. 3, pp. 297–310, 2010.

[11] J. Ahamed, M.N. Jafri, J. Ahamad, M.I. Khan, "Design and implementation of a neural network for real-time object tracking," in Proceedings of World Academy of Science, Engineering and Technology, vol. 6, 2005, pp. 209–212.

[12] Y.M. Cheung, "K*-Means: A new generalized k-means clustering algorithm," Pattern Recognition Letters, Vol. 24, No. 15, pp. 2883–2893, 2003.

[13] R. Karlsson and F. Gustafsson, "Monte Carlo data association for multiple target tracking," in IEE Target tracking: Algorithms and applications, Netherland, Oct 2001.

[14] J. Vermaak, S.J. Godsill, and P. Perez, "Monte Carlo filtering for multi-target tracking and data assoication," IEEE Trans. Aerospace and Systems, vol. 41, no. 1, pp. 309–332, January 2005.