Development of Bayesian Neural Network Using Breeder Genetic Programming to Acoustic Radar Pattern Identification

Samaher Hussein Ali
Department of Computer Science, University of Babylone

Abstract

The Evolutionary Algorithms have been used for neural networks in two main ways: (i) to optimize the network architecture in terms of hidden layers and number of neurons in each layers, and (ii) to train the weights of fixed architecture. While most previous work focuses on only one of these two options, this paper investigative Evolutionary approach called Breeder Genetic Programming (BGP) in which the architecture and the weights are optimized simultaneously. The genotype of each network is represented as a tree whose depth and width are dynamically adapted to the particular application by specifically defined genetic operators. The weights are trained by Gaussian approximation. The fitness function has been chosen as a function increasing when the mean square error at the end of the training and the number of epochs needed for learning are increase. In this paper, I have fined optimal Bayesian neural network using Breeder genetic programming to classify or identify acoustic radar Patterns. The results demonstrate that the method is capable of successfully identifying the different acoustic radar patterns.

Key words: Bayesian Neural Network, Breeder Genetic Programming, Gaussian Approximation, SODAR Pattern Identification

الخلاصة

تستخدم الخوارزميات التطورية في الشبكات العصبية بطريقتين اساسيتين:(أ)لايجاد امثل معمارية شبكة من حيث الطبقات المخفية وعدد الخلايا في كل طبقة (ب) لتدريب الاوزان لمعمارية شبكة ثابتة. بينما اغلب الاعمال السابقة تركز على واحدة فقط من تلك النقطتين فان هذا البحث يتفحص طريقة تطورية تدعى بالبرمجة الجينية التوليدية لايجاد امثل معمارية واوزان الى الشبكة العصبية بصورة متزامنة.حيث يمثل الفرد لكل شبكة كشجرة يعدل عمقها وعرضها اوتماتكياً للتطبيق العملي بتعريف عمليات جينية محددة. دالة الصلاحية التي تم اختيارها هنا هي دالة متزايدة مع تزايد كلاً من متوسط مربع الخطا في نهاية التدريب الشبكة وعدد الدورات التي نحتاجها للتدريب. في هذا البحث تم ايجاد شبكة بيزين العصبية المثلى باستخدام البرمجة الجينية التوليدية لتصنيف او تشخيص انماط الرادار الصوتي. و تبين النتائج بان الطريقة قادرة على تشخيص انماط الرادار الصوتي المختلفة بنجاح.

1. Introduction

There are two basic techniques to explore the atmospheric environment, (1) direct measurement and (2) remote sensing. The most commonly used artificial direct techniques are radiosonde, instrumented tower, microwave refractometer, and airborne systems.

Remote sensing techniques are comparatively recent and can be operated around-the-clock without human intervention. This can again be divided into two categories: passive and active. Radiometer, satellite imagery, etc. are examples of passive remote sensing techniques. On the other hand, active measurement involves the transmission and reception of electromagnetic, light, and acoustic energy. The three commonly used detection and ranging systems ("dars") are: (1) radar; (2) Lidar; and (3) acoustic radar (SODAR), corresponding to the three types of radiation used (Reeves & Janza, 1975).

SODAR (or acoustic radar) plays a very significant role in probing of the lower planetary boundary layer (LPBL). This is because the interaction of sound waves with the lower atmosphere is stronger as compared to the electromagnetic spectrum. It can be designed at a reasonable cost and is capable of providing the three-dimensional (3-D) (height, time, and intensity) view of the lower atmospheric microstructures, appearing during different time, month, and season, over a particular region (Singal , 1988).

Potential use of SODAR data can be made, provided it can be interpreted and identified correctly. So far, the task of identification of SODAR-recorded observations has been completely dependent on the knowledge, experience, and expertise of researchers working with the system and its recorded observations for a reasonable period of time. This restricts the utility of the data to a limited number of persons having expertise in the field. Utilization of the full potential of SODAR observations on a global basis calls for the existence of computer-based techniques, developed preferably by incorporating human expertise.

Construction multilayer neural networks involves difficult optimization problems, i.e. finding a network architecture appropriate for the application at hand and finding an optimal set of weight values for the network to solve the problem. Evaluation algorithms have been used for solving both optimization problems. In weight optimization, the set of weights is represented as a chromosome and a genetic search is applied on the encoded representation to find a set of weights that best fits the training data. Some encouraging results have been reported which are comparable with conventional learning algorithms. In architecture optimization, the topology of the networks is encoded as a chromosome and some genetic operators are applied to find an architecture which fits best the specified task according to some explicit design criteria.

Optimization of neural network architectures or finding a minimal network for particular applications is important because the speed and accuracy of learning and performance are dependent on the network complexity, i.e. the type and number of units and connections, and the connectivity of units. For example, a network having a large number of adjustable connections tends to converge fast, but it usually leads to over fitting of the training data. On the other hand, a small network will achieve a good generalization if it converges; it needs, however, generally a large amount of training time. Therefore, the size of the network should be as small possible, but sufficiently large to ensure an accurate fitting of the training set (Smieja, 1993).

Koza (Koza, 1992) provides an alternative approach to representing neural networks, under the framework of so-called genetic programming, which enables modification not only of the weights but also of the architecture for a neural network. However, this method provides neither a general method for representing an arbitrary feed forward network, nor a mechanism for finding a network of minimal complexity.

This paper describes a new genetic programming method, called Breeder Genetic Programming (BGP) that employs the fitness function as a function increasing when the number of epoch's (EP) needed for learning and mean square error of the network (MSE) increase. The weights are trained not by back propagation, but by Bayesian method.

In this manner, 90 data points were obtained from the charts for each category. These were used as the input to the network. The six acoustic radar pattern categories correspond to the output classes.

2. Representing Neural Networks as Trees

Multilayer feedforward neural networks are networks of simple processing elements, called neurons or units, organized in layers. The external inputs are presented to the input layer and feedforward via one or more layers of hidden units to the output layer. There is no connection between units in the same layer.

A commonly adopted architecture involves full connectivity between neighboring layers only. We allow both partial connectivity and direct connections between non-neighboring layers, since this is important for finding a parsimonious architecture.

Specifically, this architecture allows for some of input units to be connected directly to output units.

For the genetic optimization, we represent a feedforward network as a set of m trees (Neal & Zhang, 2003). Each corresponding to one output unit. Figure 1 shows a genotype to phenotype mapping example. A genotype is depicted that procedures the shown phenotype. There are 3 nodes, input nodes, one hidden, one output node, and seven connection definitions, one of which is recurrent. The second disabled, so the connection that it specifies (between nodes 2 and 4) is not expressed in the phenotype.

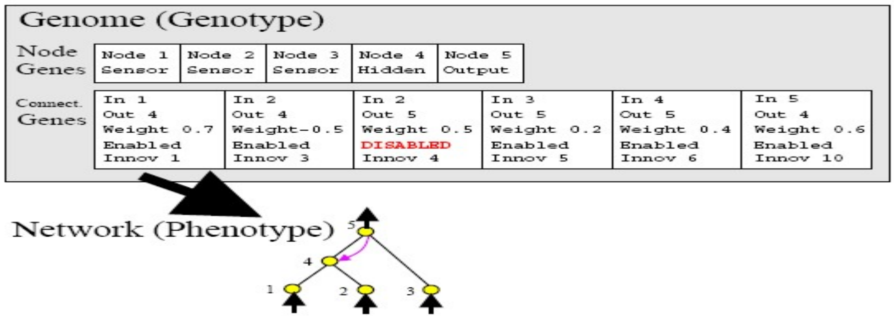

Genome (Genotype)
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Hidden
Node 5 Output
Connect. Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight-0.5 Enabled Innov 3
In 2 Out 5 Weight 0.5 DISABLED Innov 4
In 3 Out 5 Weight 0.2 Enabled Innov 5
In 4 Out 5 Weight 0.4 Enabled Innov 6
In 5 Out 4 Weight 0.6 Enabled Innov 10
Network (Phenotype)
5
4
1
2
3

Fig 1. A genotype to phenotype mapping example

3. Breeder Genetic Programming (BGP)

For the evolution of optimal neural networks we use the concepts based on the Breeder Genetic Algorithm (BGA). While the usual genetic algorithms model a natural evolution, the BGA model a rational selection performed by human breeders.

BGA represents a class of random optimization techniques gleaned from the science of population genetics, which have proved their ability to solve hard optimization problems with continuous parameters. BGA which can be seen as a recombination between Evoluation Strategies (ES) and Genetic Algorithm (GA) , uses truncation selection which is very similar to the (u ,λ) strategy in ESs (this strategy called also the comma strategy denoted by (μ, λ) the parents, after generating offspring, die off and are not taken into account to form the next generation) and the search process is mainly driven by recombination making BGAs very similar to GAs. It has been proven that BGAs can solve problems more efficiently than GAs due to the theoretical faster convergence to the optimum and they can, like GAs, be easily written in a parallel form (Falco, Cioppa, Balio & Tarantino , 2000).

Our approach differs from the BGA in that we use variable size of chromosomes, a characteristic of Genetic Programming (GP). Thus we call the method Breeder Genetic Programming (BGP). BGP also differs from usual GP. While GP uses proportional selection combined with crossover as main operator, BGP uses truncation selection combined with crossover plus Gaussian approximation.

The BGP evolutionary learning algorithm is summarized in eight steps as explain below the algorithm maintains a population A consisting of M individuals A_i of variable size. Each individual represents a neural network. The networks of initial population, A(0), are generated with a random number of layers. The receptive field of each neural unit and its width are also chosen randomly. The (g+1) population, A(g+1), is created from A(g) in three steps: **selection**, **Gaussian approximation**, and **mating**.

The selection step starts from a population of A(g) individuals. Only the M% elements showing the best fitness are chosen to give origin to the individuals of the next generation (i.e., are accepted into the mating pool B(g)). Usually truncation ratio lies in rang [10% to 50%]. The fitness function has been chosen as a function increasing when the mean square error at the end of the training and the number of epochs needed for learning are increase.

After selection, each individual in B(g) undergoes a Bayesian neural network where the weight's of the network are adapted by Gaussian approximation. This results in the revised mate set B(g). The mating phase repeatedly selects two random parent individuals in B(g) to mate and generate one offspring in the new population A(g+1) by applying crossover operators, until the population size amount to M. Notice that not only the size of individuals in one population may be different,

$$|A_i(g)| \neq |A_j(g)|,\ i \neq j \text{ and } j \in \{1,,M\} \qquad (1)$$

But the size of same individual of subsequent population may also different,

$$|A_i(g+1)| \neq |A_j(g)|,\ i \in \{1,,M\} \qquad (2)$$

A new population is generated repeatedly until an acceptable solution is found or the variance of the fitness V(g) falls below a specified limit value V_{min} (i.e., the procedure terminates if)

$$V(g) = \frac{1}{M} \sum_{i=1}^{M} \left(F_i(g) - \bar{F}(g) \right)^2 \leq V_{\min} \qquad (3)$$

Where $\bar{F}(g)$ the average fitness of the individuals in A(g), the algorithm also stops if a specified number of generations, $g_{max,}$ is carried out .

4. Bayesian Network

A belief network represents a joint probability distribution over a set of variables(Antala, Fannesa & Timmermanb, 2003).We assume that these are discrete variables, partitioned into three sets: set of inputs, output, and intermediate variables, respectively. The model consists of a qualitative part (a directed graph) and quantitative parts (dependency models). The vertices of the graph represent the random variables and the edges define the independency relations (each variable is independent of its non descendants given its parents (Bouckaert, 2005)). There is a probabilistic dependency model for each variable that describes its dependency on its parents.

Let U = {x1, …, xn}, n ≥ 1 be a set of variables. A Bayesian network B over a set of variables U is a network structure B_S, which is a directed acyclic graph (DAG) over U and a set of probability tables $B_P = \{p(u|pa(u))|u \in U\}$, where pa(u) is the set of parents of u in B_S. A Bayesian network represents probability distributions (Bouckaert,2005).

$$P(U) = \Pi_{u \in U}\, p(u|pa(u)|) \qquad (4)$$

4.1. The Bayesian Solution

The Bayesian viewpoint allows the following advantages to be gained by simple application of the rules of probability theory:

A. By viewing the product of learning as an ensemble of probable classifiers, we can take our uncertainty into account when making predictions. This improves the quality of predictions.

B. By viewing the regularizer with regularization constant α as defining a prior probability , we can solve the overfitting problem, i.e., the problem of setting hyper parameters such as α that control the "complexity" of the model with probability theory we can effectively infer the appropriate value of the regularization constant α from the data. This removes the need to waste resources on cross-validation, which is traditionally used to set such hyper parameters. This is a especially important advantage of the Bayesian approach when there are many such regularization constants.

Assume we have a set of n independent training items (x1,y1)….(xn,yn), each of which gives the value of an output vector, yi associative with an "input" vector, xi. And the prior probability distribution to weights vector p(w), likelihood function that represent error mature function p(y/x,w) while the post probability to weights vector represent by Bayesian rule as the following :

$$p(w/(x_1,y_1),\ldots(x_n,y_n))=\frac{p(w)p((x_1,y_1),\ldots,(x_n,y_n)/w)}{p((x_1,y_1),\ldots,(x_n,y_n))} \qquad (5)$$

$$=\frac{p(w)p(y_1,\ldots,y_n/x_1,\ldots,x_n,w)}{p(y_1,\ldots,y_n/x_1,\ldots,x_n)}$$

$$=\frac{p(w)\Pi_{i=1}^{n}p(y_i/x_i,w)}{p(y_1,\ldots,y_n/x_1,\ldots,x_n)}$$

In the Bayesian approach to statistical prediction, one does not use a single "best" set of network weights, but rather integrates the prediction from all possible weight vectors over the posterior weight distribution, which combines information from the data with a prior bias toward more plausible weights. And assuming the addition input represents by x_{n+1} therefore the desired output y_{n+1} can be compute as following (Neal, 1992):

$$y_{n+1}=\int_{R^N} f(x_{n+1},w)p(w|(x_1,y_1),\ldots,(x_n,y_n))dw \qquad (6)$$

In the prior distribution and predictive distribution that represent by the above relation are present the optimal solution of the neural network problems but this techniques includes multi-dimensional integrations or multivariate integration and this types of integration represent the base of most difficulty particular that found in Bayesian approach specially in many application (Hedhab, 2006).

Therefore we can consider the training is the problem of neural networks while the multivariate integration is the problem of the Bayesian neural network, the analysis solutions are impossible therefore we need to search about the replacement methods to solve this integrations. And from these methods:-A.Direct Numerical Integration, B.Monte Carlo Simulation, C.Gaussian Approximation. In this paper, I will deal with Gaussian approximation to find the best weights of the network.

4.2. Gaussian Approximation

In this paper, I will deal only with neural networks used for regressions, through the method should be applicable to classification network as well. Assume we have a set of n independent training items (x1,y1)….(xn,yn), each of which gives the value of an output vector, yi associative with an "input" vector, xi. We also have the value of the input vector for an additional test item, x_{n+1}. The task is to predict the corresponding output vector, y_{n+1} (Neal, 1992).

A neural network of some given architecture and with weight vectors w defines a mapping from an input vector x to predicted output vector y given by

$$y = f(x,w) \qquad (7)$$

Assuming a Gaussian model, the conditional probability distribution for the output vector given the input vector based on this mapping will be as follows:

$$p(y/x,w)=(2\pi\sigma^2)^{-\frac{N}{2}}\exp\left(-\frac{|y-f(x,w)|^2}{2\sigma^2}\right) \qquad (8)$$

Where y is desired output, f(x,w) is actual output, N is dimension of the output vector, σ is the noise in output. The conventional maximum likelihood approach to neural network training is to used some gradient-based algorithm to find the weight vector, that maximize the degree of fit to the data, given by the log-likelihood L(w) , defined as:-

$$L(w) = \sum \log \; p(y_i / x_i, w) = -\sum_{i=0}^{N} \frac{\left\| y_i - f(x_i, w) \right\|^2}{2\sigma^2} + C \quad (9)$$

Where C does not depend on w. the output noise level σ can also estimated from the training data. The derivative of equation (10) represents maximum penalized likelihood estimation, defined as:-

$$L'(w) = -\frac{\|w\|^2}{2\omega^2} - \sum_{i=0}^{N} \frac{\left\| y_i - f(x_i, w) \right\|^2}{2\sigma^2} \quad (10)$$

Here, the constant ω relates to the expected scale of the weights, and might be set by program designer.In the Bayesian network the above equation represents the mount of error occurs between the desired and actual output of the network. In Gaussian probability distribution method I used this equation to adapts neural network weights and reaches to the optimal weights vector.

To complete the Bayesian formulation of the problem, a prior distribution for the network weights is required. To determine the weights vector we take random samples and then find the probability distribution that its represent by Gaussian distribution as follows:-

$$p(w) = (2\pi\omega^2)^{\frac{N}{2}} \exp(-\frac{\|w\|^2}{2\omega^2}) \quad (11)$$

5. The Proposed Methodology

The objective of this paper is present a method using breeder genetic o programming to develop Bayesian neural network to acoustic radar pattern identification, this search concerns itself with find the optimal network architecture in terms of hidden layers and number of neurons in each layers, and to train the weights of fixed architecture. Then use this network to identify the acoustic radar patterns. The main steps of the proposed method are explained in the following procedure. And these steps are discussed in detail below.

```
**Procedure of the proposed methodology**
Input: pattern data set is array of features for each pattern.
Output: optimal network (structure, connection weights) to identify acoustic radar
patterns.
Set population size, truncation ratio, learning factor, momentum term, noise level
value.
Set MaxGen of BGP, $EP_{max}$ of Bayesian neural network.
DO for each pattern in the dataset (initialize)
        Randomly generate M networks.
        Set a random values to number of neurons in each layers in the rang [0-100]
and connection
        weights in the rang[0,1].End for
Set Gen←1
Do for each network in the population
        Evaluate the fitness function according to f(x) as explain in eq(13)
        Select upper M% network into mating pool B(gen).
        Train each network in mating pool by Bayesian neural network as explain in paragraph(5.3.)
        Set the results in mating pool B(gen).
  End for
 Repeat
Do for each individual in the population
    Keep the best individual
    Select subpopulation of parents;
    Perform crossover between parents.
    Evaluate the offspring.
    Modify the population from A(t+1) to A(t) by replace the worst fit network in A(t+1)by the best
    in A(t)
  End for
Gen ← Gen+1
Until (Gen > MaxGen)
Return the best architecture and weights for Bayesian neural network that satisfies the conditions.
End.
```

5.1. Genetic Operators

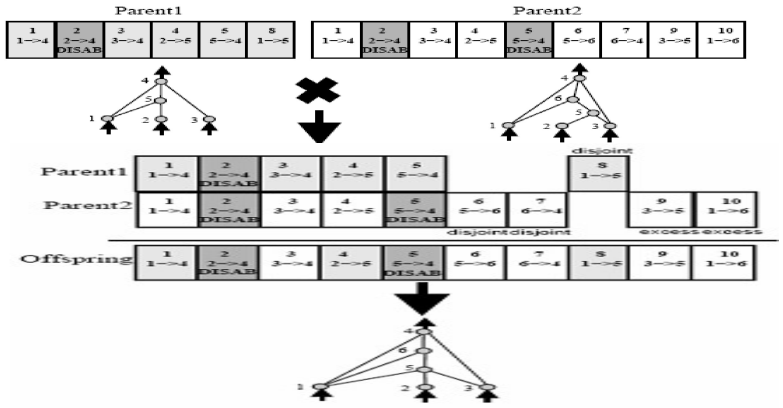
The weights of a network are trained by applying a Gaussian approximation to each of the individuals accepted by truncation selection. Given a chromosome Si of the network, Bayesian network finds a better chromosome by repeatedly applying the Gaussian approximation and adapting the weight until there is no weight form found having better fitness in each sweep through the individual.

After that the crossover operator adapts the size and shape of the network architecture. A crossover operation stars by choosing at random two parent individuals from the mating pool B(g). Actual crossover of two individuals, i and j, is done on their genotypical representations Si and Sj. The nodes in the tree are numbered according to depth – first search order and crossover sites Ci and Cj are chosen at random with the following conditions:

$$1 \le Ci \le size(Si) \quad \text{and} \quad 1 \le Cj \le size(Sj) \qquad (12)$$

Here, the length of an individual, size (Si), is defined as the total number of units and weights.

Given the crossover points, the subtrees of two parent individuals, Si and Sj are exchanged to form one offspring. The label of the nodes Ci and Cj must belong to the same class as explain below:-



5.2. Fitness Function

To evaluate the goodness of an individual, the network is trained with a fixed number of patterns and then evaluated according to determined parameters. The parameters which seen to better describe the goodness of a network configuration are the mean square error MSE at the end of the training and the number of epochs EP needed for learning. Clearly it is desirable to attain for both the parameters MSE and EP value low as possible: in fact, a neural network must learn as fast as possible (same value of EP), and with a good approximation of the output desired. It is necessary to establish an upper limit EP_{max} to the number of epochs EP units by the network for the training , thus $0 \leq EP \leq EP_{max}$. Moreover, it is desirable MSE be as close as possible to a small value e_{min} ($0 \leq e_{min} \leq MSE$) which represent the minimum error required (below e_{min} we can say the network has learned its task).

$$F(x) = \frac{EP}{EP_{\max}} + MSE \qquad (13)$$

Please note that, according to the above formula, F(x) can be lower than 1 if and only if the learning phase is such that minimum error required is reached be artificial neural network within a number of epochs lower than EP_{max} (Quagliarella, Periaux, Poloni.& Winter, 1998).

5.3. Bayesian Neural Network Training

The Bayesian neural network learning algorithm is a form of supervised learning used to train mainly feed forward neural networks to perform many tasks (Bill, 2003). It's used sigmoid function to determine activation of their neurons and it uses the MSE as a measure to determine convergence of network outputs toward the desired output. Figure 2 explains the structure of Bayesian neural network.

Bayesian neural network consists of three kinds of layers (Zurada, 1997):-

- **Input Layer (i)**: - This layer is accountable for received network inputs and is distributed on hidden layer neurons.
- **Hidden Layer (j)**:- This layer does extraction information about distribution features within the sum of training examples and it's sent to a neighboring hidden layer or output layer with dependency weights (Salah, 2003).
- **Output Layer (k)**: - This layer is responsible for receiving stimulus pattern code from the hidden layers with dependency weights and finding the actual output of the network. The output of each layers can be computed as following:-

$$h_j = f\left(\sum_{i=1}^{n} w_{ij} x_i\right) \qquad (14)$$

$$y_k = f\left(\sum^{l} w_{jk} h_j\right) \qquad (15)$$

Where h_j is the output of hidden node, y_k is the output of output node, f() represents a nonlinear contentious function increasing monotonically and differential contentious that called activation function and in this paper, I will deal with sigmoid function.

$$f(x) = \frac{1}{1+\exp(-\lambda net)} \qquad (16)$$

Let λ=1 and net represents as:-

$$net = \sum_{i=1}^{n} x_i w_{ij} \qquad (17)$$

$$net = \sum_{j=1}^{l} h_j w_{jk} \qquad (18)$$

How you can determine the weights of the Bayesian neural network?

At the first, we generate random weights of the network between 0 and 1 to the output and hidden layers. Then calculate weights through prior probability distribution that represents a Gaussian approximation as the following:-

$$p(w_{ij}) = (2\pi\omega^2)^{-\frac{L}{2}} \exp\left(-\frac{w_{ij}^{\;2}}{2\omega^2}\right) \qquad (19)$$

$$p(w_{jk}) = (2\pi\omega^2)^{-\frac{L}{2}} \exp\left(-\frac{w_{jk}^{\;2}}{2\omega^2}\right) \qquad (20)$$

Where L is number of output nodes (i.e., in this paper, L=6).

At the above I explains the Bayesian neural network is feedforward network therefore its needs to desired output and this value use to training the network by compute the magnitude of different between the actual output and desired output of the network, as explain above Bayesian network depend on compute probability in its works therefore we must compute the condition probability distribution function and there derivative, where the derivative of probability function represents maximum magnitude of weights that given the best output of network and the following equation represent the derivative of Gaussian approximation to likelihood function:

$$l'_k(d_k / x_k, w_{jk}) = -\frac{w_{jk}^{\;2}}{2\omega^2} - \frac{[d_k - f(x_k, w_{jk})]^2}{2\sigma^2} \qquad (21)$$

Where d_k is desired output, $f(x_k,w_{jk})$ is actual output, σ the noise level in output.

I use the probability function in compute the error occurs of all nodes in output layer. If the error occurs in hidden layer then the require value in hidden nodes unknown therefore we use the error founds in output nodes.

$$l'_j = h_j(1 - h_j)\sum_k (l'_k * w_{jk}) \qquad (22)$$

Adjust weights between input layer and hidden layer as follows:-

$$\Delta w_{ij} = \eta l'_j x_i + \alpha \Delta w'_{ij} \qquad (23)$$

Where Δw_{ij} represents the different between the input layer and hidden layer weights, η learning rate is a value between [0.1, 0.3], $\Delta w'_{ij}$ represents the different between current and prior weight. α momentum rate, l'_j error in node j. The new weight become:-

$$w_{ij} = \Delta w'_{ij} + w''_{ij} \qquad (24)$$

Where w''_{ij} is prior weight. Adjust weights between hidden layer and output layer as follows:-

$$\Delta w_{jk} = \eta l'_k h_j + \alpha \Delta w'_{jk} \qquad (25)$$

Where Δw_{jk} represents the different between the output layer and hidden layer weights, η learning rate is a value between [0.1, 0.3], $\Delta w'_{ij}$ represents the different between current and prior weight. α momentum rate, l'_k error in node k, h_j represents output of hidden layer. The new weight become

$$w_{jk} = \Delta w'_{jk} + w''_{jk} \qquad (26)$$

This paper uses the MSE as a measure to determine convergence of network outputs toward the desired output.

$$MSE = \frac{1}{2}\sum_{i=0}^{N}\sum_{j=0}^{L}[d_j(i) - y_j(i)]^2 \qquad (27)$$

Where N is number of training patterns, L is number of classes.

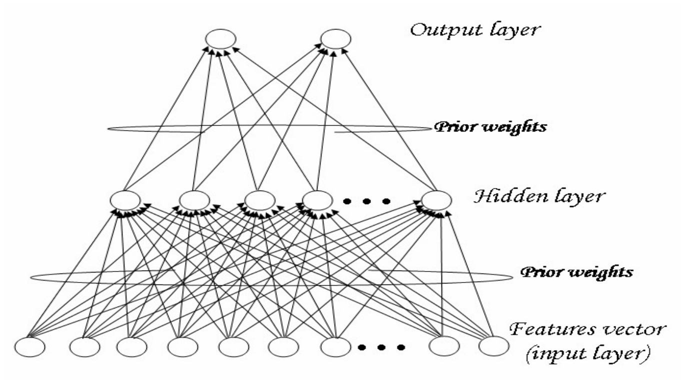


Fig 2. The Structure of Bayesian Neural Network

6. Implementation and Results

In this paper, 90 data points were obtained from the charts for each category. These were used as the input features. The six pattern categories correspond to the output classes (i.e., output classes are convective boundary layer, temperature inversion layer with flat top, temperature inversion layer with small spikes, temperature inversion layer with tall spikes, rising inversion layer, rising inversion layer with capping a convective boundary layer).

A three-layered Bayesian neural network, with breeder genetic programming, was used for classifying the patterns, random sets of 20 sequential observations, from the total 90 data points in each pattern category, constituted a training pattern with 20 attributes (input nodes) for that class. Six nodes were involved at the output layer. Various truncation selection ratio were used (10%, 20%, 30%, 40%, 50%) each one presents different network in shape and size (number of node in hidden layer) and also in training time. The selection ratio Q% refers to random class wise selection of Q% training data (a sequence of 20 points in the chart) from the entire dataset. The remaining (100-Q) % data constitute the test set in each case.

6.1. Case Study Number One

In this case study uses the following parameters of breeder genetic programming (BGP):-Population size= 50, Truncation ratio=10%, No. of generation= 50. And the following parameters of the Bayesian neural network:- Max No. of epochs =20000, Learning rate= 0.2, Momentum term=0.09, Noise level value= 0.15. The best weights and architecture achieved is 20-6-6, the value of fitness function=0.992068, and the network enables from identification all other patterns after 18795 epochs with MSE is 0.052318.

6.2. Case Study Number Two

In this case study uses the following parameters of breeder genetic programming (BGP):-Population size= 30, Truncation ratio=20%, No. of generation= 50. And the following parameters of the Bayesian neural network:- Max No. of epochs =3000, Learning rate= 0.2, Momentum term=0.05, Noise level value= 0.17. The best weights and architecture achieved is 20-10-6, the value of fitness function=0.80561, and the network enables from identification all other patterns after 2405 epochs with MSE is 0.00395.

6.3. Case Study Number Three

In this case study uses the following parameters of breeder genetic programming (BGP):-Population size= 100, Truncation ratio=30%, No. of generation= 50. And the following parameters of the Bayesian neural network:- Max No. of epochs =7000, Learning rate= 0.11, Momentum term=0.09, Noise level value= 0.18.The best weights and architecture achieved is 20-13-6, the value of fitness function=0.78501 and the network enables from identification all other patterns after 3100 epochs with MSE is 0.342153.

6.4. Case Study Number Four

In this case study uses the following parameters of breeder genetic programming (BGP):-Population size= 100, Truncation ratio=40%, No. of generation= 50. And the following parameters of the Bayesian neural network:- Max No. of epochs =10000, Learning rate= 0.25, Momentum term=0.04, Noise level value= 0.13. The best weights and architecture achieved is 20-8-6, the value of fitness function=0.65681, and the network enables from identification all other patterns after 5641 epochs with MSE is 0.09271.

6.5. Case Study Number Five

In this case study uses the following parameters of breeder genetic programming (BGP):- Population size= 100, Truncation ratio=50%, No. of generation= 50. And the following parameters of the Bayesian neural network:- Max No. of epochs =1000, Learning rate= 0.3, Momentum term=0.089, Noise level value= 0.18. The best weights and architecture achieved is 20-15-6, the value of fitness function=0.59468 and the network enables from identification all other patterns after 585 epochs with MSE is 0.00968.

7. Conclusions

This work presented an evolutionary method called breeder genetic programming for learning both the network architecture and the weights at the same time. The method uses trees to represent a feedforward network whose size and topology are dynamically adapted by genetic operators.

The use of the Bayesian neural network for identifying types of acoustic radar patterns is expected to overcome the above problems. Neural network models are inherently suitable in data-rich environments and can extract underlying relationships from the data domain. This reduces dependence on the human expert for identification process.

In all the cases study above the network enables from identification all the remaining data related to those classes in test stage. By looking of the results to all the cases study above, we can conclusion the following:-

When increasing the truncation ratio and number of neurons in hidden layer the ability of network to identification is increasing and the time of learning is decrease.

When comparing this work with the previous works, we found the following: this work using new type of fitness function depends on the number of epoch need to training neural network and it is not depended on the trail and error principle in determine the structure of network but suggest using breeder genetic programming as a tool to determine the structure and the optimal weight of the network.

The real challenge in this situation is to be able to providing the optimal Bayesian neural network by dynamically adapted the architecture and the weights of the network, I think the proposed method successes in satisfy this point. While the goal of this paper is identifying the different acoustic radar patterns, and from the results I can say the propose method also successes in satisfy that goal.

References

Reeves. R.G and Janza. F.J, "Manual of Remote Sensing: Theory Instruments and Techniques". Falls Church, VA: Amer. Soc. Photogramm., 1975.

Singal. S.P, “Acoustic sounding studies of the atmospheric boundary layer,” Meteorol. Service, Wellington, New Zealand, Tech. Rep. 30, 1988.

Smieja. F, " Neural Network Constructive Algorithms, Trading Generalization for Learning Efficiency?", Circuits Systems, and Signal Processing, pp 331-374, 1993.

Koza. J.R,"Genetic Proramming: On the Programming of Computers by Means of Natural Selection", 1992.

Site:http://citeseer.ist.psu.edu/cache/papers/cs/1832/http:zSzzSzwww.irsip.na.cnr.itzSz~hotgSzpaperszSzppsn.pdf/defalco96investigating.pdf

Neal. R.M and Zhang. J,"Classification for High Dimensional Problems Using Bayesian Neural Networks and Dirichlet Diffusion Trees",University of Toronto, NIPS , Feature Selection Workshop,2003.

Falco. I, Cioppa. A, Balio. R and Tarantino. E, "Investigating A Parallel Breeder Genetic Algorithm on The Inverse Aerodynamic Design", Naples-Italy, 2000.

Quagliarella. D, Periaux. J, Poloni. C.and Winter. G,"Genetic Algorithms and Evoluation Strategy in Engineering and Computer Science ", John Wiley and Sons, New Yourk, October, 1998.

Antala. P, Fannesa. G, Timmermanb. D, Moreaua . Y, and De Moora. B, "Bayesian applications of belief networks and multilayer perceptrons for ovarian tumor classification with rejection" Artificial Intelligence in Medicine, Vol 29, pp 39–60, 2003.

Bouckaert. R.R. "Bayesian Network Classifiers in Weka", University of Utrecht, April 21, 2005.

Neal. R, "Bayesian Training Of Back propagation Networks By The Hybrid Monte Carlo Method", Department of Computer Science, Toronto University, April, 1992.

Hedhab. K, "Bayesian Neural Network for Medical Image Classification", M.Sc. Thesis, Babylon University, 2006.

Bill.W,"Neural Networks and Error Back-Propagation Learning", 2003. Site:http:// www.cs.unew.edu.au\~billw\mldict.html

Zurada. J. M, "Introduction to Artificial Neural System", JAICO publishing House, Bombay, 1997

Salah..M, "Hybrid System By Using Wavelet Transform in Neural Networks for Image Classification ", M.Sc.Thesis, Babylon University, 2003.