Designing A Decision Support System For Weaning Using Ann – A Preliminary Study

[1] Kavitha C, [2] Hemalatha K.L

[1] Dept. Computer science & Engineering, KNSIT, Banglore, INDIA
[1] kc98cs@gmail.com
[2,] Dept. of Information science & Engineering
SKIT, Banglore, INDIA
[2]hema.sjcit@gmail.com

Abstract— Patients require mechanical ventilator support when the ventilatory and/or gas exchange capabilities of their own respiratory system fail. Mechanical ventilation can have life threatening complications, it should be discontinued at the earliest possible time. The process of discontinuing mechanical ventilation termed "weaning" is one of the most challenging problems in intensive care and it account for a considerable proportion of the work load of staff in an ICU. The discontinuation of mechanical ventilation needs to be carefully timed. Premature discontinuation place severe stress on the respiratory and cardiovascular system, which can impede the patient's recovery. Unnecessary delay in discontinuation can lead to a host of complication. There is a need of research and to develop a standard and accurate approach to the problem. Here is a project proposed, using ANN as a decision supportive system, which will standardize the procedure of predicting weaning decision.

Keywords—Weaning, ANN, mechanical ventilator

I. INTRODUCTION

The process of discontinuation or withdrawal of mechanical ventilation machine from ventilator dependent patients is called WEANING. The process of weaning is gradual, which takes days to months. As the conditions that warranted placing the patient on ventilator stabilize and begin to resolve, attention should be placed on removing the ventilator as quickly as possible. Although this process often is termed "ventilator weaning".

Because mechanical ventilation can have life-threading complication, it should be discontinued at the earliest possible time. The process of discontinuing mechanical ventilation, termed weaning, is one of the most challenging problems in intensive care, and it accounts for a considerable proportion of the workload of staff in intensive care unit. When mechanical ventilation is discontinued, up to 25% of patients have respiratory distress, severe enough to necessitate the reinstitution of ventilator support.

There is no standard procedure for predicting a weaning or discontinuation from mechanical ventilation machine. Clinicians are inaccurate in predicting weaning decisions. The process of discontinuation of mechanical ventilation needs to be carefully timed. Premature discontinuation places severe stress on the respiratory and cardiovascular system, which will affect the early recovery of patients.

Unnecessary delays in discontinuation can lead to a host of complication causes pneumonia or airway trauma (infection), lose of money (cost) and even death. Decisions about timing that are based solely on expert clinical judgment are frequently erroneous.

In the current situation no proper methodology exists for weaning. Weaning decision is completely dependent on

1: Trail & error basis and

2: On clinician's expertise.

Any clinician irrespective of their expertise can utilize this project. Artificial Neural Network (ANN) is a technology that helps to develop decision supportive system, which will standardize the procedure of predicting weaning decision. Genetic Algorithm is used to train ANN to achieve high accuracy.

II RELATED WORK

ANN also referred to as "parallel distributed processing system". This is because all the nodes at the same layer process operation at the same time and are independent of each other and the output from each node is distributed to all the nodes of the next layer. The input nodes are passive since they just replicate the information whatever they possess and pass them to the hidden nodes. The hidden nodes are active, which extract the input features and maps them into other domain that is made availed to the outer layer. The outer node is active as well. So the hidden layer and the output layer do all the necessary computations. The links between the layers carry the signal from one node to the next node with the aid of connection weights, which can amplify or

diminish the signal strength. These weights are modifiable using learning algorithm to obtain the desired output[1].

Active node function

Considering the value of the bias input equal to one, at active nodes all the input signals multiplied by their corresponding weights are summed up and the result fed to the limiter, Sigmoid.

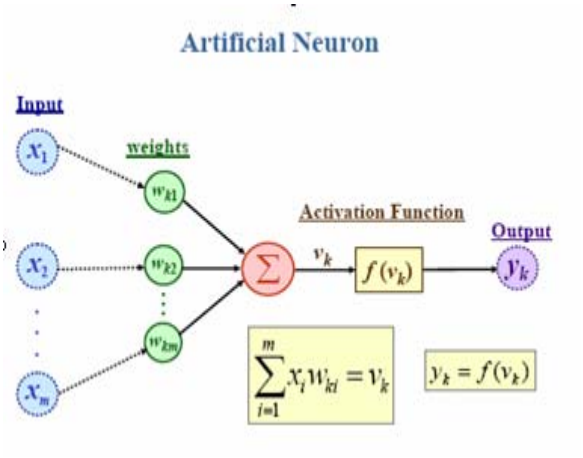

Fig 1. Artificial Neuron Model

The reasons for selection of sigmoid

- It has biological basis, since the firing frequency of biological neurons as a function of excitation follows a sigmoid characteristic. In simple it represents the mathematical model of biological neuron activity.
- Unlike the Step & the Ramp function, Sigmoid has smooth threshold for bounding the output response and exhibits a graceful balance between the linear and non-linear characteristics[2].

The ANN architecture has to be designed with 4 Input nodes to take the input of patient parameters. This module reveals in detail about the neural network architecture proposed for the work along with its operation process flow. The input to the module is train set preprocessed patient parameters applied to ANN architecture initially with random weights and output is a value used to find the error.

Predictable Attributes

VE, VT, RR,NIF[3]

a.Minute Ventilation: VE (10-15ml/kg of ideal body weight)

b. VT (2-3ml/kg of ideal body weight)

c.Respiratory rate: RR (less than 25/minute)

d. Negative Inspiratory Force: NIF (-20 to -25cm H20) [4]

Initially all the connection strengths are multiplied with the corresponding inputs and the result of this becomes the input to the hidden layer processing nodes. The active hidden layer performs accumulation and mapping operation on these inputs. The mapping of the accumulated result is required so that it could remain a bound number.

Output of hidden layer nodes are multiplied with the random weights selected for the outer layer. This becomes the input to the output node, which is also active. Accumulation and mapping done finally at this node, the output from this becomes the output of a particular epoch (next generation).

The error function is defined to check the co-relation. If the error is not satisfying our required criteria (less than or equal to threshold), the learning rule will take charge of weight adjustment in right direction as to reduce the error. Now these adjusted weights become the connection strength for the next generation processing. The further operation is repeated as in the first generation till error satisfies our required criteria.

Training (learning) is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which network is embedded. Learning rule is a prescribed set of well-defined rules for the solution of learning.

Selection of Learning Algorithm:

Supervised learning is required for pattern matching and Generalized Delta Rule (GDR) used in so called Error Back Propagation Algorithm (EBPA) is the most popular of the supervised learning techniques used to train a feed forward multi-layered artificial neural network The EBPA uses gradient descent to achieve training by error correction, where network weights are adjusted to minimize error based on a measure of the difference

between desired and actual feed forward network output. Desired input/output behavior is given in the training set where the input & the target values {i, t} are predefined.

The GDR is a product learning rule for a feed-forward, multilayer structured neural network that uses the gradient descent to achieve training or learning by error correction. Network weights are adjusted to minimize an error based on a measure of the difference between the desired and actual feed forward network output.

When 'del' operator applied to a scalar, it becomes the gradient of that scalar function. Gradient is a vector, its magnitude gives the maximum rate of increment and direction gives the direction in which this maximum increment rate of scalar function can be found.

Initially the ANN weights are random. The train set patient parameters are applied at the input layer. The output of ANN is value which is compared with target value of the particular train set to find the error. The error is the difference of target output and ANN output.

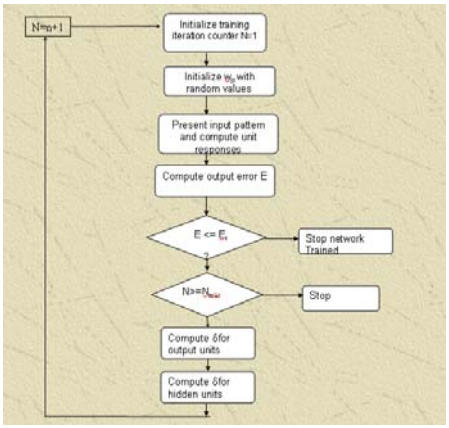

Fig 2. Training using Back Propagation Algorithm

In the back propagation paradigm, the connection weights are adjusted to reduce the output error. In the initial state, the network has a random set of connection weights. When a system starts with all connection weights equal, the network begins at a sort of local optimum, and will not converge to the global solution. In order for the network to learn, sets of inputs are presented to the system and sets of outputs are calculated. A difference between the actual outputs and desired outputs is calculated, which gives the value of error and the connection weights are modified in iterations to reduce this difference to a very low value. After inputs have been applied and the network output solution has been calculated, the estimated error contribution at the network output is calculated and its gradient needs to be determined. The calculations begin at the output layer of the network.

For the output layer, the change in connection weights can easily be calculated, since the error gradient of the output neurons is easily determined. With the introduction of hidden layers, the desired outputs of these hidden neurons (active nodes) become more difficult to estimate. In order to estimate the error gradient of a hidden neuron, the error signal from the output layer must be propagated backwards to all preceding layers. This process of calculating the error gradient is performed for all layers until the input layer is reached.

Generalized Delta Rule

The *Error signal* **e** at the output of an output node neuron (network output) is defined by, $e=(T_i-O_i)$

Where T_i and O_i denote respectively the desired (target) and the actual network output.

The GDR tries to minimize the scalar *Error function* **E**, defined as the *mean square of an Error* signal **e** (because learning rule demands a differentiable function) given by, $E=[(T_i-O_i)^2]/2$ Gradient of *Error function* **E** is then computed and is denoted as ∇E**,** given by, $\nabla E = \partial E/\partial W_{ij}$

Where W_{ij} denotes the strength of interconnection from node **i** to node **j**.

Weight adjustment ΔW is then set proportional to ∇E, hence given by $\Delta W= - \eta * \nabla E$

Here negative of Gradient **E** (∇E) means the rate of decrement is maximum in negative direction, which implies decrement in error signal, thus serving the goal of training ANN.

'η' is the 'learning constant'. The effectiveness and convergence of the GDR depends significantly on the value of learning constant 'η'. However the optimum value of η depends on the problem being solved and there is no single learning constant value suitable for different training cases.

From chain rule, we obtain

$$\partial E/\partial W_{ij} = \partial E/\partial O_i * \partial O_i/\partial W_{ij}$$

From eqn (1), we obtain

$$\partial E/\partial O_i = -(T_i - Oi)$$

$\partial O/\partial W_{ij}$ gives the how the individual weights are going to affect the network output.

This affect is calculated by taking *Slope*.

.e., Slope = $\partial O_i/\partial E/\partial W_{ij} = -(T_i - O_i) * \text{Slope}$

Therefore from eqns (3) & (7)

$$\Delta W = \eta * (T_i - O_i) * \text{Slope}$$

- *Physical interpretation of Delta Rule*

In ANN, the main goal of learning rule is to minimize the *error*, which is the difference between the target and the observed network output. Weight training is usually formulated as minimization of an error function, such as the mean square error between the target and the actual outputs averaged over all by iteratively adjusting connection weights. Error exists because initially there is zero co-relation between the connection weights and the applied inputs, since the connection weight values are chosen to be random. So the error value is high initially.

- *Minimization of error*

In simplified form we can say, if error is high, it indicates that the current weights are very far from the required value and hence a large change in weights required. In other case if error is less, it indicates that the current weights are near to the required value and hence change in weights required for the next iteration is also small. Hence we can say,

$$\Delta W \propto (\text{Error})$$

Each and every weight is responsible for the output, but in different proportion, where the hidden layer weights (W_h) have less affect on the output as compared to the outer layer weights (W_o), which can be justified. So it seems logical to adjust the individual weights in accordance to their affect at the output in order to minimize the error, where the affect is calculated by taking slope, i.e., variation at the output w.r.t the variation in individual weights. Hence we can say,

$$\Delta W \propto (\text{Slope})$$

We have $\Delta W = \eta * (\text{Error}) * (\text{Slope})$ from eqn (a) and (b)

$$= \eta * (T_i - O_i) * (\text{Slope})$$

<u>ΔW for output layer (ΔW_o)</u>

Slope = $\partial O_i/\partial W_{o1}$;

$O_i \rightarrow f(\text{net})$;

$\text{net} \rightarrow f(W_o)$;

From chain rule, $\partial O_i/\partial W_{o1} = \partial O_i/\partial \text{net} * \partial \text{net}/\partial W_{o1}$

$$= f`(S) * \partial[X_1W_{o1} + X_2W_{o2} + \ldots]/\partial W_{o1}$$

$$= f`(S) * X_1$$

$$= \text{Slope}$$

$$\therefore \Delta W_o = \eta * (T_i - O_i) * f`(S) * X_i$$

In general

Hence $W_{new} = W_{old} + \Delta W_o$ for output layer

<u>ΔW for hidden layer (ΔW_h)</u>

Slope = $\partial O_i/\partial W_{h1}$;

$O_i \rightarrow f(\text{net})_o$;

$(\text{net})_o \rightarrow f(X_1)$;

From chain rule, $\partial O_i/\partial W_{h1} = \partial O_i/\partial (\text{net})_o * \partial (\text{net})_o/\partial X_1 * \partial X_1/\partial W_{h1}$

$$= f`_o(S) * W_{o1} * \partial[X_1]/W_{h1}$$

$\partial[X_1]/W_{h1} = \partial[X_1]/\partial(\text{net})_h * \partial(\text{net})_h/\partial W_{h1}$

$$= f`_h(S) * X_{11}$$

$$\therefore \partial O_i/\partial W_{h1} = f`_o(S) * W_{o1} * f`_h(S) * X_{11}$$

$$= (\text{Slope})_h * (\text{Slope})_o$$

$\therefore \Delta Wh = \eta * (T_i - O_i) * f`_o(S) * f`_h(S) * W_{oi} * X_i$ In general

Hence $W_{new} = W_{old} + \Delta W_h$ for hidden layer

- *GDR/EBPA Limitations*

The back propagation algorithm applies a gradient (steepest) descent approach to minimize an error function. In order to adjust the connection weights, this approach requires learning steps in weight space, the length (duration) of which corresponds proportionally to the chosen learning constant 'η'. If the steps are too small ($\eta << 1$), it may take longer to converge (to get minimum error) and may become trapped in a local minimum of error function where it is incapable of finding a global minimum, which is the drawback associated with GDR.

Although widely used, the back propagation algorithm has not escaped from drawbacks. The method of calculating weights doesn't seem to be biologically plausible, since neurons don't seem to adjust the efficacy of their synaptic weights by functioning in reverse direction. Thus the back propagation learning algorithm is not viewed by many as a learning process that emulates the biological world but as a method to design a network with learning.

Training of ANN using Genetic algorithm.

Evolutionary Computing is the collective name for a range of problem-solving techniques based on principles of biological evolution, such as natural selection and genetic inheritance. Genetic algorithm is a programming technique that mimics biological evolution as a problem-solving strategy. Genetic algorithms are a class of search algorithms modeled on the process of natural evolution. As the complexity of the search space increases, genetic algorithms present an increasingly attractive alternative to gradient based techniques such as back propagation. Advantage of genetic algorithms is their generality. Initially ANN weights are random. A chromosome is a set of weights of ANN. The population consists of chromosomes. The train set patient parameters are applied at input layer of ANN. The output of ANN is a value which is compared with the target value of train set to find the error.

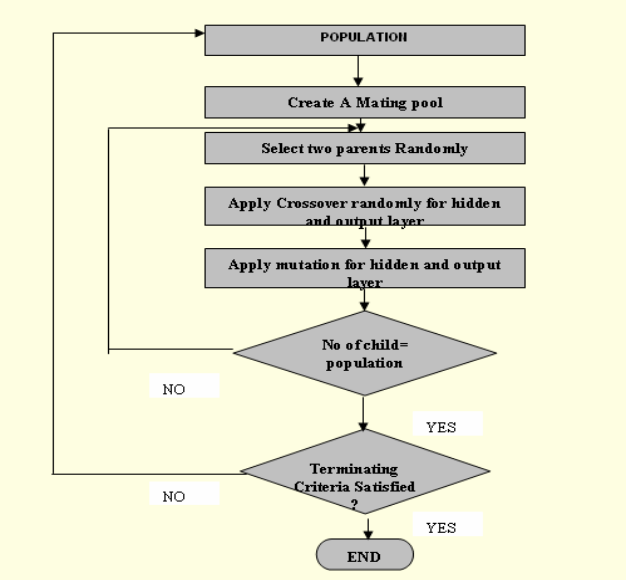

Fig 3. Training using Genetic Algorithm

Genetic Algorithm always evaluate from population, so they create the number of parents required to form the population. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem. Here the problem is selection of weights for proper decision. So a chromosome is a set of weights of ANN, the weights are randomly generated. Population consists of the chromosomes. The weights from the chromosome are assigned to ANN. The patient parameters are applied at the input layer of ANN. The output of ANN is a value which is compared with the target value of the particular train set patient parameter to find the error value. The error is the difference of ANN output and target value. The error values are stored and mean error is calculated. If the terminating criteria are met, i.e. chromosome error and mean error should be minimum, and then the fittest chromosome is selected as the optimal set of weights.

If the terminating criteria is not met the fitness of the particular chromosome is calculated as fitness=1/error. All the chromosomes should have a fitness value i.e. all the chromosomes should be applied to ANN to calculate error and fitness, otherwise next chromosome is applied to ANN architecture. .After all the chromosomes have the fitness value, they are arranged in the increasing order of fitness. The weakest chromosomes are replaced by the fittest one. Arranging chromosomes in increasing order and replacing weakest by fittest one forms the mating pool. Two chromosomes from mating pool are chosen at random, the genetic

operator crossover is applied to hidden layer and output layer followed by mutation being applied to hidden layer and output layer. The process repeats until the number of children is equal to the population. Once the number of child equal to population is obtained, this will form the new generation.

III CONCLUSIONS

In this paper, we proposed a prediction system for weaning using back propagation algorithm and genetic algorithm. The train set patient parameters are linear normalized and applied as inputs to ANN. Initially random weights are used in ANN. Gradient descent algorithm is used as learning rule to train the weights. The train set and test set patient parameters are applied and verified. To improve accuracy Genetic algorithm is used as learning rule. ANN trained by genetic algorithm gives decision of weaning mechanical ventilator from the patient accurately. When the patient parameters are applied to ANN, if the output is 1, then weaning can be done, if the output is 0, then weaning is not suggested.

REFERENCES

[1] An Introduction to Neural Networks,by James . Anderson,_MIT Press, 1995.
[2] Artificial Neural Networks by Schalkoff, Tata_cGraw-Hill Education, 1997
[3] AARC Clinical Practice Guidelines, 1993.
[4] Clinical Practice standards of and recommendations from Critical Care and ulmonary Medicine. Approval of Medical Director.