ISSN No. 0976-5697

Volume 6, No. 5, May - June 2015

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

A Comparative Analysis on ID3 and Backpropagation Algorithms

C.R.kavitha
Sri Lakshmi Narasimha College of Pharmacy
Andhra Pradesh, India

Abstract: This article briefly explain a comparative study on ID3 classification and Backpropagation algorithm. Both are belong to learning paradigms, but ID3 gives the overview of learning through symbolic Artificial Intelligence algorithms, that is learning system based on the inductive paradigms, whereas Backpropagation describe learning through Neural Network algorithms. But both are belongs to Supervised Learning concept. Symbolist (Artificial Intelligence) and connectionist (Neural Network) approaches are learned from example; only differ in knowledge-base representation and in inductive mechanism. To explore these differences here we compare and identify the aspects of each system that may account for these performance differences.

Keywords: ID3, Backpropagation, Neural Network, Artificial Intelligence, Inductive

I. INTRODUCTION

Where does intelligence emerge? There are two important ways to answer this question in computational point of view. One is based on *symbolism* and the other is based on *connectionism*. The former approach models intelligence using symbols, while the latter using connections and associated weight [1]. Symbolic and connectionist (Neural network) learning strategies are receiving more attention. Comparative studies should qualify the advantages of system from each paradigm.

Symbolic and Connectionist learning methods often address similar tasks, but they may differ considerably in their processing assumptions. Thus there is impetus for investigating the relative advantages and limitations of systems of each paradigm [5].

A great variety of human experience can be described as learning; the term machine learning is sometimes obscure, research in machine learning has grown brisk in recent years. A task that has commonly been explored in each model is *Learning from examples (or tutored learning/Supervised learning)*: a tutor or a supervisor who classifies the training examples into classes. From a set of classified observation, a learning system abstracts a rule or mapping that facilitates classification of new observations [5].

The dominant approach in symbolism assumes that the knowledge base is a flat or tree structure set of concept and descriptions. Typically, each concept is a logical rule that defines class membership. In contrast, Connectionist methods assume a knowledge base of interconnected nodes, each of which computes a weighted sum of its inputs. External inputs are arithmetically combined and propagated through the network. This process terminates with the computation of external outputs that represent an objects classification. Learning alters weights so that classification correctness is improved [3].

II. ID3 (ITERATIVE DICHOTOMIZER 3)

ID3 stands for Iterative Dichotomizer (version) 3. J. Ross Quinlan originally developed ID3 at the University of Sydney. He first presented ID3 in 1975 in a book called Machine leaning, Volume.1 [7]. It learns objects classifications from labeled training examples. The basic algorithm is based on earlier research programs known as Concept Learner System (CLS). This system is also similar in many respects to the expert system architecture. ID3 is an implementation of the basic CLS algorithm with some modifications like the way in which the attributes are ordered for use in the classification process [2].

ID3 is a simple, but effective symbolic method for learning from example. It uses a tree representation to classify new unknown objects. To classify a set of instances, we start at the top of tree and answer the questions associated with the nodes in the tree until we reach a lead node, where the classification or decision is stored.

ID3 begins by choosing a random subset of the training instances. This subset is called the window. The procedure builds a decision tree that correctly classifies all instances in the window. The tree is then tested on the training instances outside the window. If all the instances are classified correctly, then the procedure halts. Otherwise, it adds some of the instances incorrectly classified to the window and repeats the process. This iterative strategy is empirically more efficient than considering all instances at once. In building a decision tree, ID3 select the feature which minimizes the entropy function and thus best discriminates among the training instances [1].

A. ID3 Algorithm:

1. Select a random subset W (called the 'Window') from the training set.
2. Build a decision tree for the current window.
   - Select the best feature which minimizes the entropy function H:

$$H = \sum_{i} -p_i \log p_i$$

Where p_i is the probability associated with i^{th} class. For a feature, the entropy is calculated for each value. The sum of the entropy weighted by the probability of each value is the entropy for that feature.

   - Categorize training instances into subsets by this feature.

© 2015-19, IJARCS All Rights Reserved

- Repeat this process recursively until each subset contains instances of one kind (class) or some statistical criterion is satisfied.
  3. Scan the entire training set for exceptions to the decision tree.
  4. If exceptions are found, inset some of them into W and repeat from step-2. The insertion may be dine either by replacing some of the existing instances in the window or by augmenting it with the new exceptions [1].

Example: Use the ID3 algorithm to build a decision tree for classifying the following objects.

Class	Size	Color	Surface
A	Small	Yellow	Smooth
A	Medium	Red	Smooth
A	Medium	Red	Smooth
A	Big	Red	Rough
B	Medium	Yellow	Smooth
B	Medium	Yellow	Smooth

Table: 1

a. Given:

No. of instances given : 6
No. of attributes given : 4 (Class, Size, Color, Surface)

b. Solution:

Using Natural Logarithm first calculate the Entropy for each attributes, by taking one attribute as the base (here take 'Class' attribute to calculate the entropy). The formula to find out the Entropy function is as follows:

$$H = \sum_i -p_i \log p_i$$

(i) Entropy of the attribute Size:

The attribute 'size' has three different sizes (Small, Medium, and Big) [1].

➢The size **Small** belongs **to *class A*** and occurred only one time **(1/6)**

Class	Size	Color	Surface
A	Small	Yellow	Smooth

➢The size **Medium** belongs to both **class A** and **Class B** and occurred four times **(4/6)**

Class	Size	Color	Surface
A	Medium	Red	Smooth
A	Medium	Red	Smooth
B	Medium	Yellow	Smooth
B	Medium	Yellow	Smooth

➢The size **Big** belongs to **class A** and occurred only one time **(1/6)**

Class	Size	Color	Surface
A	Big	Red	Rough

$$H = \sum p_i \log p_i$$

$$Size: H = \frac{1}{6}\left(-\frac{1}{1}\log\frac{1}{1}\right) + \frac{4}{6}\left(-\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4}\right) + \frac{1}{6}\left(-\frac{1}{1}\log\frac{1}{1}\right)$$

$$Size: H = \frac{1}{6}\left(-\frac{1}{1} * (0.0000)\right) + \frac{4}{6}\left(-\frac{2}{4} * (-0.6931)\right) - \frac{2}{4} * (-0.6931)\Big) + \frac{1}{6}\left(-\frac{1}{1} * (0.0000)\right)$$

$$Size: H = \frac{1}{6}(0) + \frac{4}{6}(-0.5 * (0.6931) - 0.5 * (-0.6931)) + \frac{1}{6} * (0)$$

$$Size: H = 0 + \frac{4}{6}(0.34655) + 0.34655) + 0$$

$$Size: H = 0 + \frac{4}{6}(0.34655) + 0.34655) + 0$$

$$Size: H = 0 + \frac{4}{6}(0.6931) + 0$$

$$\boldsymbol{Size: H = 0.462}$$

(ii) Entropy of the attribute Color:

The attribute 'Color' has two different colors (Yellow and Red) [1].

Class	Size	Color	Surface
A	Small	Yellow	Smooth
A	Medium	Red	Smooth
A	Medium	Red	Smooth
A	Big	Red	Rough
B	Medium	Yellow	Smooth
B	Medium	Yellow	Smooth

* The color ***Red*** belongs to only ***Class A and*** occurred three times **(3/6)**

Class	Size	Color	Surface
A	Medium	Red	Smooth
A	Medium	Red	Smooth
A	Big	Red	Rough

* The color ***Yellow*** belongs to both ***class A*** and ***Class B*** and occurred three times **(3/6),** Out of three yellow color one belongs to class A (1/3) and two yellow color belongs to class B(2/3)

Class	Size	Color	Surface
A	Small	Yellow	Smooth
B	Medium	Yellow	Smooth
B	Medium	Yellow	Smooth

© 2015-19, IJARCS All Rights Reserved

$$H = \sum p_i \log p_i$$

$$Color: H = \frac{3}{6}\left(-\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right) + \frac{3}{6}\left(-\frac{3}{3}\log\frac{3}{3}\right)$$

$$Color: H = \frac{3}{6}\left(-\frac{2}{3} * (-0.4055) - \frac{1}{3} * (-1.0986)\right) + \frac{3}{6}\left(-\frac{3}{3} * (0)\right)$$

$$Color: H = \frac{3}{6}\,(0.2703 + 0.3662) + 0$$

$$Color: H = \frac{3}{6}\,(0.6365)$$

$$\boldsymbol{Color: H = 0.318}$$

(iii) Entropy of the attribute Surface:

The attribute surface has two kinds of surfaces (Smooth and Rough) [1].

➢The surface **Smooth** belongs to both **class A** and **Class B** and occurred five times **(5/6).** Out of five surfaces three belongs to **Class A (3/5)** and two belongs to **Class B (2/5).**

Class	Size	Color	Surface
A	Small	Yellow	Smooth
A	Medium	Red	Smooth
A	Medium	Red	Smooth
B	Medium	Yellow	Smooth
B	Medium	Yellow	Smooth

➢The surface **Rough** belongs to **only class A** and occurred only one time (1/6)

Class	Size	Color	Surface
A	Big	Red	Rough

$$H = \sum p_i \log p_i$$

$$Surface: H = \frac{5}{6}\left(-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5}\right) + \frac{1}{6}\left(-\frac{1}{1}\log\frac{1}{1}\right)$$

$$Surface: H = \frac{5}{6}\left(-\frac{3}{5} * (-0.5108) - \frac{2}{5} * (-0.9163)\right) + \frac{1}{6}\left(-\frac{1}{1} * (0.0000)\right)$$

$$Surface: H = \frac{5}{6}\left(-0.6 * (-0.5108) - 0.4 * (-0.9163)\right) + \frac{1}{6}(-1 * (0))$$

$$Surface: H = \frac{5}{6}\,(0.3064 + 0.3665) + 0$$

$$Surface: H = \frac{5}{6}\,(0.6729)$$

$$\boldsymbol{Surface: H = 0.560}$$

The entropies of the attribute are as follow:

Attributes	*Entropy values*
Size	**0.462**
Color	**0.318**
Surface	**0.560**

Table: 2

From the above table it's clear that the attribute *Color* is having minimum entropy of 0.318. So, select the attribute color as the first decision node (root node). This node has two different colors/branches: Red and Yellow. Under the branch Red, only *class A* object fall, and hence no further discrimination is needed. So, place it right side of the first decision node *color*. Under branch Yellow, we need another attribute to make further distinctions and it's placed left side of the first decision node *color*. So, we calculate the entropy for the other two attributes (*Size* and *Surface*) under this branch [1]:

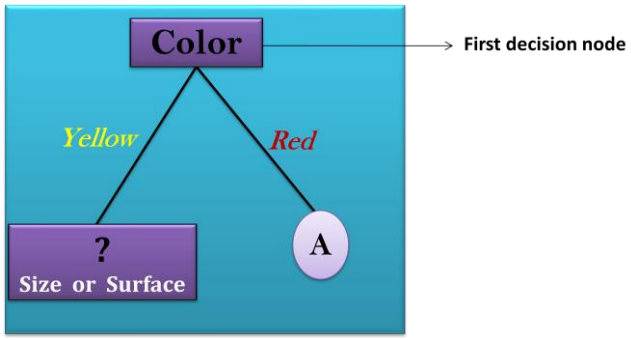

Figure 1 : Decision tree with first decision node

(i) Entropy of the attribute Size:

Based on the *color* yellow the *size* attribute has two kinds of size (Small and Medium). Totally there are three yellow colors, out this one belongs to ***class A(1/3)*** and two yellow colors belongs to ***class B(2/3).***

Class	Size	Color	Surface
A	Small	Yellow	Smooth
A	Medium	Red	Smooth
A	Medium	Red	Smooth
A	Big	Red	Rough
B	Medium	Yellow	Smooth
B	Medium	Yellow	Smooth

$$H = \sum p_i \log p_i$$

$$Size: H = \frac{1}{3}\left(-\frac{1}{1}\log\frac{1}{1}\right) + \frac{2}{3}\left(-\frac{2}{2}\log\frac{2}{2}\right)$$

$$Size: H = \frac{1}{3}\left(-\frac{1}{1} * 0\right) + \frac{2}{3}\left(-\frac{2}{2} * 0\right)$$

$$Size: H = \frac{1}{3}\,(0) + \frac{2}{3}\,(0)$$

$$Size: H = (0) + (0)$$

© 2015-19, IJARCS All Rights Reserved

$$\boldsymbol{Size: H = 0}$$

(ii) Entropy of the attribute Surface:

$$H = \sum p_i \log p_i$$

$$Surface: H = \frac{3}{3}\left(-\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right)$$

$$Surface: H = \frac{3}{3}\left(-\frac{2}{3} * (-0.4055)\ \frac{1}{3} * (-1.0986)\right)$$

$$Surface: H = \frac{3}{3}(0.2703 + 0.3662)$$

$$Surface: H = \frac{3}{3}(0.6365)$$

$$\boldsymbol{Surface: H = 0.6365}$$

The entropies of the attribute are as follow:

Attributes	*Entropy values*
Size	0
Surface	0.6365

Table: 3

This table shows that the attribute *size* is having minimum entropy of H = 0. Select this attribute as second decision node and the decision tree is created as follow:

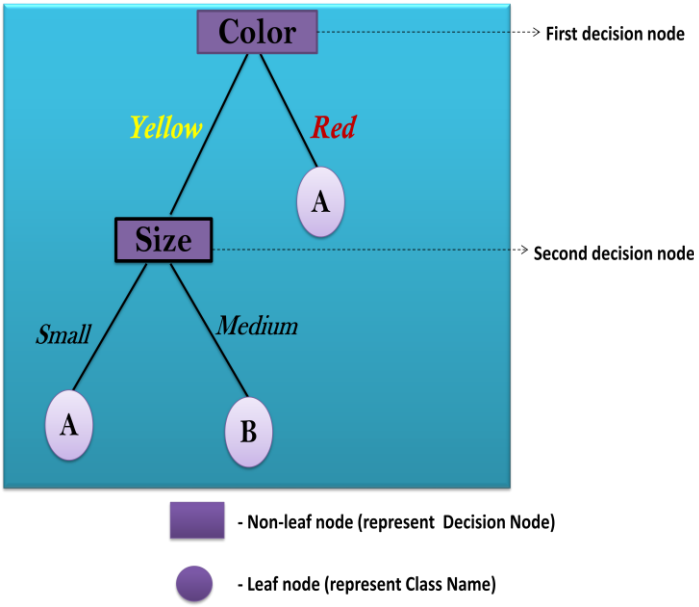

Figure 2: A Decision tree for classified objects

The classes created by ID3 are inductive that is, given a small set of training instances, the specific classes created by ID3 are expected to work for all future instances. The leaf nodes of the decision tree contain the class name, whereas a non-leaf node is a decision node. The decision node is an attribute test with each branch (to another decision tree) being a possible value of the attribute [7].

III. BACKPROPAGATION

The founders of Backpropagation are Rumelhart, Hinto and Williams, 1986. This network is the most well known and widely used among the current type of supervised Neural Network learning algorithm. The learning rule is known as Backpropagation, which is a kind of gradient descent techniques with backward error (gradient) propagation.

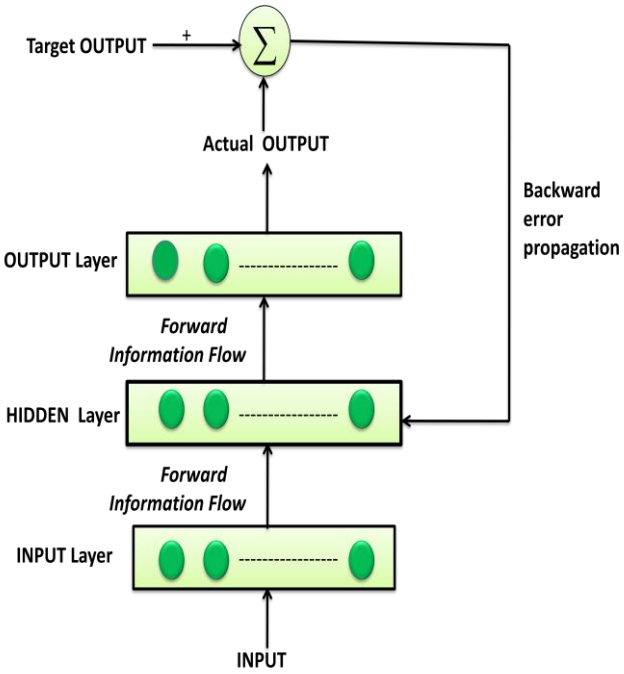

Figure 3: The Backpropagation Network

The name 'Backpropagation' comes from the fact that the error (gradient) of hidden units are derived from propagating backward the errors associated with output units since the target values for the hidden units are not given. The activation function chosen is the sigmoid function, which compresses the output value into the range between 0 and 1 [1]. It's a three layered (input, output and hidden layer) with feed-forward structure. The first layer of the neuron is the input layer and the third layer is the output layer. The middle layer is called the hidden layer, because it is the only layer that does not communicate with the external environment either by taking in the external input or sending out the system output.

The Backpropagation model could have more than one hidden layer, but the hidden layers have a hierarchical structure a lower level communicates only to its immediate upper level [4]. The training instance set for the network must be presented many times in order for the interconnection weights between the neurons to settle into a state for correct classification of input patterns. While the network can recognize patterns similar to those they have learned, they do not have the ability to recognize new patterns. In order to recognize new patterns, the network needs to be retrained with these patterns along with previously know patterns. If only new patterns are provided for retraining, then old patterns may be forgotten. In this way the learning is not incremental over time [1].

A. The Backpropagation Algorithm:

- ***Weight Initialization***

Set all weights and node thresholds to small random numbers. Note that the node threshold is the negative of the weight from the bias unit (whose activation level is fixed at 1).

© 2015-19, IJARCS All Rights Reserved

- ***Calculation of Activation:***

a) The activation level of an input unit is determined by the instance presented to the network.

b) The activation level O_j of a hidden and output unit is determined by

$$O_j = F(\sum W_{ji} O_i - \Theta_j)$$

Where

W_{ji} – the weight from an input O_i

Θ_i - the node threshold

F - Sigmoid Function

$$F(a) = \frac{1}{1+e^{-a}}$$

- ***Weight Training:***

a) Start at the output units and work backward to the hidden layers respectively. Adjust weight by

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}$$

where

$W_{ji}(t)$ – Weight from unit *i* to unit *j* at time t (or) t^{th} iteration.

ΔW_{ji} - the weight adjustment.

b) The weight change is computed by

$$\Delta W_{ji} = \eta \delta_j O_i$$

where

η - a trial-independent learning rate ($0 < \eta < 1$, e.g., 0.3)

δ_j - the error gradient at unit *j*

Convergence is sometimes faster by adding a momentum term:

$$W_{ji}(t+1) = W_{ji}(t) + \eta \delta_j O_i + \alpha[W_{ji}(t) - W_{ji}(t-1)]$$

Where $0 < \alpha < 1$

c) The error gradient is given by

* For the output units

$$\delta_j = O_j(1 - O_j)(T_j - O_j)$$

where

T_j - the desired (target) output activation

O_j - the actual output activation at output unit *j* .

* For the hidden units

$$\delta_j = O_j(1 - O_j)\sum \delta_k W_{kj}$$

where

δ_k – the error gradient unit *k* to which a connection points from hidden unit *j*.

d) Repeat iterations until convergence in terms of the selected error criterion. An iteration includes presenting an instance, calculating activation, and modifying weights.

B. Limitations of Backpropagation:

1. Learning is not incremental:

The network can recognize only old patterns and doesn't have the ability to recognize new patterns. While retraining it forgot old patterns and recognize only new.

2. Local minima:

The Backpropagation network is Prone to local minima, just like any other gradient descent algorithm the below figure shows that Backpropagation searches on the error surface along the gradient (steepest descent) in order to minimize the error criterion [1].

Backpropagation uses a gradient-descent procedure and follows the curve of an error surface with weight updates moving it in the direction of steepest descent. For simple two-layer networks(without a hidden layer), the error surface is bowl shaped and using gradient-descent to minimize error is not a problem, the network will always find an errorless solution (At the bottom of the bowl). Such errorless solutions are called '*Global Minima'*. When an extra hidden layer is added to solve more difficult problems, the possibility arises for complex error surfaces which contain many minima. Since some minima are deeper than others it is possible that gradient descent will not find a global minimum. The network may fall into *Local Minima* which represent suboptimal solutions. It's difficult to avoid local minima while training a Backpropagation in some cases. But in practice the more hidden units you have in a network then less likely you are to meet a local minimum during training [6].

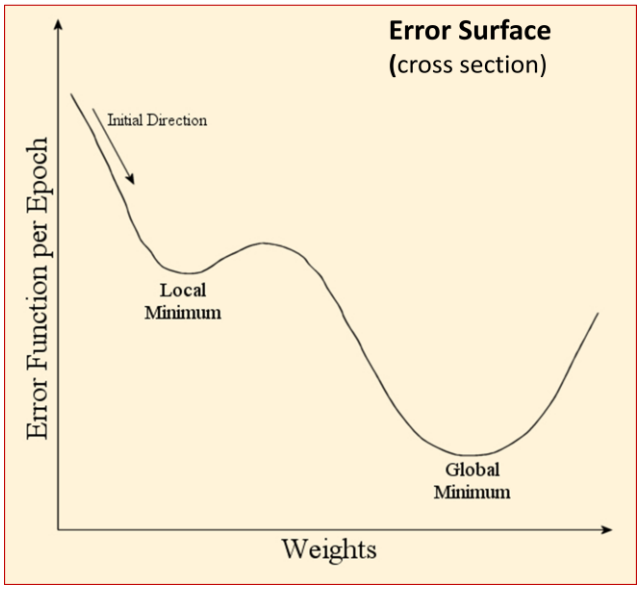

Figure 4: Search on the error surface along the gradient

3. Network [10] paralysis occurs when the weights are adjusted to very large values during training large weights can force most of the units to operate at extreme values, in a region where the derivative of the activation function is very small.

4. A multilayer neural network requires many repeated presentation of the input patterns, for which the weights need to be adjusted before the network is able to settle down into an optimal solution [10].

C. Derivation of Backpropagation learning rule:

Here, [1] we show how to derive the Backpropagation learning rule given by

$$\Delta W_{ji} = \eta \delta_j O_i$$

If unit j is an output unit, then its δ_j is calculated by

$$\delta_j = (T_j - O_j)F'(net_j)$$

where

$$net_j = \sum_i W_{ji} O_i$$

© 2015-19, IJARCS All Rights Reserved

F is a sigmoid function, and

$$O_j = F\left(net_j\right) = F\left(\sum_i W_{ji} O_i\right)$$

if unit *j* is a hidden unit, then its δ_j is given by

$$\delta_j = F'_j\left(net_j\right) \sum_k \delta_k W_{ki}$$

the Backpropagation procedure minimizes the error criterion

$$E = \frac{1}{2} \sum_j \left(T_j - O_j\right)^2$$

Gradient descent yields

$$\Delta W_{ji} = -\eta \left(\partial E / \partial W_{ji}\right)$$

by using the chain rule, we obtain

$$\partial E / \partial W_{ji} = \left(\partial E / \partial W_j\right) / \left(\partial O_j / \partial W_{ji}\right)$$

in the case when unit *j* is an output unit,

$$\partial E / \partial O_j = -\left(T_j - O_j\right)$$

and

$$\partial O_j / \partial W_{ji} = F'_j\left(net_j\right) O_j$$

Thus

$$\begin{aligned} \partial E / \partial W_{ji} &= \left(\partial E / \partial O_j\right)\left(\partial O_j / \partial W_{ji}\right) \\ &= -\left(T_j - O_j\right) F'_j\left(net_j\right) O_i \\ &= -\delta_j O_i \end{aligned}$$

So, we obtain

$$\Delta W_{ji} = \eta \delta_j O_i$$

When unit *j* is hidden units, T_j is not given. Applying the chain rule gives

$$\partial E / \partial O_j = \sum_k \left(\partial E / \partial O_k\right)\left(\partial O_k / \partial O_j\right)$$

The output of unit *k* is given by

$$O_k = F\left(\sum_j W_{kj} O_j\right)$$

thus, the term $\partial O_k / \partial O_j$ can be transformed by

$$\partial O_k / \partial O_j = F'\left(net_k\right) W_{kj}$$

As a result

$$\begin{aligned} \partial E / \partial O_j &= \sum_k \left(\partial E / \partial O_k\right)\left(\partial O_k / \partial O_j\right) \\ &= -\sum_k \left(T_k - O_k\right) F'\left(net_k\right) W_{kj} \\ &= -\sum_k \delta_k W_{kj} \end{aligned}$$

Thus

$$\begin{aligned} \partial E / \partial W_{ji} &= \left(\partial E / \partial O_j\right)\left(\partial O_j / \partial W_{ji}\right) \\ &= -\left(\sum_k \delta_k W_{kj}\right) F'_j\left(net_j\right) O_i \\ &= -\delta_j O_i \end{aligned}$$

Thus we obtain

$$\Delta W_{ji} = \eta \delta_j O_i$$

D. Backpropagation and XOR:

As we know the truth table for XOR gate is as follow:

X	Y	Z
1	1	0
1	0	1
0	1	1
0	0	0

Table 4: Truth table for XOR gate

Implement a Backpropagation network to simulate the exclusive-or function as shown in the following figure. Here we have taken the first pattern x=1, y=1 and z=0 for training the instances, that is z= XOR(x, y).

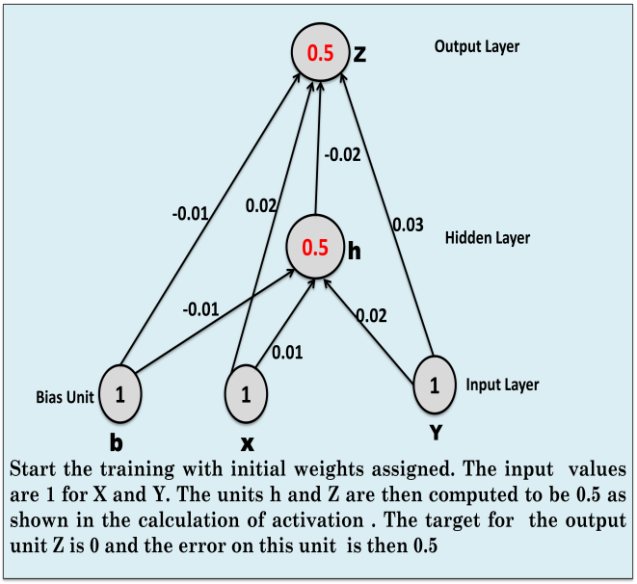

Figure 5: A three layered Backpropagation network for learning the XOR function (training process)

In this figure the circles represent neurons or units or nodes that are extremely simple analog computing devices. The numbers within the circles represent the activation values of the units. The main nodes are arranged in layers. In this case there are three layers, the input layer that contains the values for x,y and z. The hidden layer is so named because the network can be regarded as a black box with inputs and outputs that can be seen but the hidden units can't be seen. There is another unit present in this network is called as 'bias unit' whose values are always 1. The lines connecting

The circles represent weights and the number beside a weight is the value of the weight. Mostly Backpropagation networks only have connections within adjacent layers, however this one has two extra connections that go directly from the input units to the output unit. In some cases, like these XOR input-output connections make training the network much faster [7].

- ***Weight Initialization:***

The weights are initialized randomly as follows
$W_{zx} = 0.02$, $W_{zy} = 0.03$, $W_{zh} = -0.02$,
$W_{hx} = 0.01$, $W_{hy} = 0.02$, $W_{zb} = -0.01$,
$W_{hb} = -0.1$

- ***Calculation of Activation:***

© 2015-19, IJARCS All Rights Reserved

The function which is used to compute the value of a neuron can be called the '*Activation function or Squashing function or Transfer function'* [7]. The activation function chosen is the Sigmoid Function

$$F(a) = \frac{1}{1+e^{-a}}$$

This function compresses output value into the range between 0 and 1. The sigmoid function is advantageous in that it can accommodate large signals without saturation while allowing the passing of small signals without excessive attenuation. It's a smooth function so that gradients can be calculated, which are required for a gradient descent search [1].

1. Input layer (x,y):

Here the Activation of Input layers x and y are assigned to 1.

X = 1 and y= 1

2. Hidden layer (h):

The first step of this computation is to look at each lower level unit and the bias unit that is connected to the hidden unit. For each of these connections, find the value of the unit and multiply by the weight and sum (a) all the result [7]. The calculations give:

1 * -0.01 = -0.01
1 * 0.01 = 0.01
1 * 0.02 = 0.02

Sum (a) = 0.02

$$F(a) = \frac{1}{1+e^{-a}}$$

$$= \frac{1}{1+e^{-0.02}}$$

$$= \frac{1}{1+0.9802}$$

$$= \frac{1}{1.9802}$$

Activation function for Hidden layer = **0.505**

3. Output layer (z):

Calculation of activation function for the output unit z is:

1 * -0.01 = -0.01
1 * 0.02 = 0.02
1 * 0.03` = 0.03
0.505 * -0.02 = -0.0101

Sum (a) = 0.0299

$$F(a) = \frac{1}{1+e^{-a}}$$

$$= \frac{1}{1+e^{-0.0299}}$$

$$= \frac{1}{1+0.9705}$$

$$= \frac{1}{1.9705}$$

Activation function for Output layer = **0.508**

- ***Weight Training:***

(i) First calculate the error gradient for 'output layer (z)'

$$\delta_j = O_j(1 - O_j)(T_j - O_j)$$

$$\delta_z = 0.508(1 - 0.508)(0 - 0.508)$$

$$\delta_z = 0.508(1 - 0.508)(0 - 0.508)$$

$$\delta_z = 0.508 * (0.492 * -0.508)$$

$$\delta_z = 0.508 * (-0.2499)$$

$$\delta_z = -0.127$$

(ii) Second compute the weight change/weight adjustment of output layer(z) as follow and assume learning rate $\eta = 0.3$

$$\Delta W_{ji} = \eta \delta_j O_i$$

$$\Delta W_{zx} = 0.3 * (-0.127) * 1$$
$$\Delta W_{zx} = -0.038$$

$$\Delta W_{zy} = 0.3 * (-0.127) * 1$$
$$\Delta W_{zy} = -0.038$$

$$\Delta W_{zb} = 0.3 * (-0.127) * 1$$
$$\Delta W_{zb} = -0.038$$

$$\Delta W_{zh} = 0.3 * (-0.127) * 0.505$$
$$\Delta W_{zh} = 0.3 * -0.06414$$
$$\Delta W_{zh} = -0.019242$$

(iii) Error gradient for ' Hidden layer (h)'

$$\delta_h = O_j(1 - O_j) \sum \delta_k W_{kj}$$

$$\delta_h = 0.5 * (1 - 0.5) * -0.127 * -0.019242)$$

$$\delta_h = 0.5 * (1 - 0.5) * 0.002443734$$

$$\delta_h = 0.5 * (1 - 0.5) * 0.002443734$$

$$\delta_h = 0.0006109335$$

(iv) weight change/weight adjustment of hidden layer(h) as follow and assume learning rate $\eta = 0.3$

$$\Delta W_{ji} = \eta \delta_j O_i$$

$$\Delta W_{hx} = 0.3 * 0.0006109335 * 1$$
$$\Delta W_{hx} = 0.00018328005$$

$$\Delta W_{hy} = 0.3 * 0.0006109335 * 1$$
$$\Delta W_{hy} = 0.00018328005$$

$$\Delta W_{hb} = 0.3 * 0.0006109335 * 1$$
$$\Delta W_{hb} = 0.00018328005$$

© 2015-19, IJARCS All Rights Reserved

(v) Weight change/weight adjust formula for a weight is as follow:

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}$$

The new weights will [7] be:

$$W_{zx} = W_{zx}(t) + \Delta W_{zx}$$
$$W_{zx} = 0.02 + (-0.038)$$
W_{zx} = -0.018

$$W_{zy} = W_{zy}(t) + \Delta W_{zy}$$
$$W_{zy} = 0.03 + (-0.038)$$
W_{zy} = -0.008

$$W_{zb} = W_{zb}(t) + \Delta W_{zb}$$
$$W_{zb} = -0.01 + (-0.038)$$
W_{zb} = -0.048

$$W_{zh} = W_{zh}(t) + \Delta W_{zh}$$
$$W_{zh} = -0.02 + (-0.038)$$
W_{zh} = -0.039242

$$W_{hx} = W_{hx}(t) + \Delta W_{hx}$$
W_{hx} =0.01 + 0.00018328005
W_{hx} =0.01018328005

$$W_{hy} = W_{hy}(t) + \Delta W_{hy}$$
W_{hy} =0.02 + 0.00018328005
W_{hy} =0.02018328005

$$W_{hb} = W_{hb}(t) + \Delta W_{hb}$$
W_{hb} = −0.01 + 0.00018328005
$$W_{hb} = -0.00981672$$

The rest of the weight adjustments are omitted. Note that the threshold (which is the negative of the weight from the bias unit) is adjusted likewise. It takes much iteration like this before the learning process stops. The following set of final weights gives the mean squared error of less than 0.01 values [1].

W_{zx} = 4.98, W_{zy} = 4.98, W_{zh} = -11.30, W_{hx} = 5.62, W_{hy} = 5.62, W_{zb} = -2.16, W_{hb} = -8.83

(vi) Training process*:*

In [7] this example the network weights are initialized randomly and the training process will try to adjust the weights to get correct output. A small recall of the working process of the Backpropagation is as follow:

a. First, put one of the patterns to be learned on the input units.
b. Second, find the values for the hidden and output unit.
c. Third, find out how large the error is on the output unit.
d. Fourth, use one of the back-propagation formulas to adjust the weights leading into the output unit. (the ideas is try to make the answer little bit closer to the right answer).
e. Fifth, use another formula to find out errors for the hidden unit.
f. Last one is, adjust the weights leading into the hidden layer unit via another formula. Repeat steps from 1 to 6 for the second, third and fourth XOR patterns.
g. Last one is, adjust the weights leading into the hidden layer unit via another formula. Repeat steps from 1 to 6 for the second, third and fourth XOR patterns.
h. Iteration:

Even though all these changes to the weights the answers will only be a little closer to the rights answer and the whole process has to be repeated many times. Each time all the patterns in the problem have been used once we will call that an *iteration*, often other people call this an *epoch*.

The [7] unfortunate problem with Backpropagation is that the learning rate is too large the training can fails as it did in the case when $\eta = 0.3$.

IV. COMPARATIVE ANALYSIS ON ID3 AND BACKPROPAGATION

Table 5

Sr.No.	***Principle Concepts***	***ID3***	***Backpropagation***	***References***
1.	*Learning Method:* Any interventions that are deliberately undertaken to assist the process of learning at individual, team or organizational level.	Supervised Learning/ Error based learning / Monitor Learning / Inductive Learning/ Tutor Learning/	Supervised Learning / Error based learning / Monitor Learning / Inductive Learning/ Tutor Learning/	[1], [5]
2.	*Learning Community*	Learning from example	Learning from example	[4]
3.	*Learning Mode:* Through the manner a program learned.	Non-incremental	Non-incremental	[1], [5]
4.	*Learning System:* A software applications or web-based technology used to plan, implemented and assess a specific learning process.	Concept Learning System(CLS)	Neural Network System	[1], [2]
5.	*Learning Algorithm:* A branch of artificial intelligence in which a computer generates rules underlying or based on raw data that has been fed into it. Basically both are belong to Machine Learning algorithm.	Artificial Intelligence Learning Algorithm	Neural Network Learning Algorithm	[1]
6.	*Learning Rule:* Algorithm for determining the connection strength to ensure learning.	Decision Tree	Widrow-Hoff Delta Rule (or) Error Correction Learning Rule (or) Generalized delta rule	[9]

© 2015-19, IJARCS All Rights Reserved

7.	*Representation Concept:* Illustrating the network model.	Represented through '*Symbolism'*	Represented through '*Connectionism'*	[5]
8.	*Backtracking:* To return by the same route by which one has come.	Backtracking is not applicable. The algorithm uses a greedy search, to picks the best attribute and never looks back to reconsider earlier choices.	The name "Backpropagation" comes from the fact that the errors (Gradient) of hidden units are derived from propagation backward the errors associated with output units.	[1], [8]
9.	*Weight Adjustment* A small positive (or) negative number assigned to a parameter associated with a connection from one neuron to another are adjusted to get correct or closer to target output.	Not applicable	Backpropagation adjust weights, so as to improve the match between actual and ideal output. The weights on the connections encode the knowledge of a network. It uses a highly parallel, distributed control and can learn to adjust itself automatically.	[1], [3]
10.	*Network Model:* Semantic description of the arrangement of a network, including its nodes and connecting lines.	Top-down tree structure with divide and conquer approach	Multi-layered feed-forward Neural Network	[4], [7]
11..	*Activation Function:* Describe the output behavior of a neuron (or) function which is used to compute the value of a neuron. Also called as Squashing function (or) transfer function.	Entropy Function $H = \sum p_i \log p_i$	Sigmoid Function $F(a) = \frac{1}{1+e^{-a}}$	[1]
12.	*Statistical/ Stopping Criterion:*	Statistical criterion can be applied to stop the tree from growing as long as most of the instances are classified correctly.	3 types of stopping criterion: 1. Based on the error to be minimized. 2. Based on the gradient. 3. Based on cross-validation performance.	[1]
13.	*Amount of training data*	ID3 perform worse on small amount training sets. And it may be related to the problem of small disjunction.	For small amount of training data Backpropagation is a better choice.	[11]
14.	*Training Process*	ID3 processes the data only once.	Backpropagation repeatedly process the data until one of the stopping criteria is met.	[12]
15.	*Human Interpretability of the acquired rules.*	Symbolic learning can produce interpretable rules.	Networks of weights are harder to interpret.	[12]
16.	*Utility of attributes*	ID3 is a '*Monothetic* 'classification, that is learning considers the utility of a single attribute at a time. And this may lead ID3 to weak.	Backpropagation is '*Polythetic'*, in that the values of multiple attributes are simultaneously considered.	[3]
17.	*Information Processing*	Decision tree style	Brain-Style / Neurally-inspired	[1], [12]
18.	*Level of abstraction*	Cognitive Learning (at conscious level)	Perceptual Learning (at Subconscious level)	[1]

V. CONCLUSION

In this article we have formulated two important learning algorithms. The main purpose of this review article is to provide readers with good introduction and comparison about both algorithms. The fundamental differences between the ID3 and Backpropagation will help the beginners of machine learning to understand the learning paradigm easily. Future investigation on these algorithms will hopefully give a best solution to the relative strengths and weakness of the symbolic and connectionist approaches to machine learning.

VI. ACKNOWLEDGEMENT

I would like to thank my parents and my brother for their support for completing this journal. Last, and most obvious but not least, I thank the IJARCS for their valuable guidance to rectify mistakes in my article.

VII. REFERENCES

[1]. LiMin Fu, Neural Networks in Computer Intelligence, 2003, Tata McGraw Hill Edition, pp. 3-94.

[2]. Dan W. Patterson, Introduction to Artificial Intelligence and Expert Systems, 2004, Prentice Hall of India Private Limited, pp. 401-414.

[3]. Dogulas H.Fisher and Kathleen B.McKusick "An Empirical Comparison of ID3 and Back-Propagation", International Joint Conference on Artificial Intelligence, Aug. 1989, pp.788-793, doi:August 1989(Article in a Conference Proceeding)

© 2015-19, IJARCS All Rights Reserved

[4]. James R. Noan, "Computer System that learn: An Empirical study of the effect of noise on the performance of three classification methods", Expert System with Applications, vol. 23, July 2002, pp.39-47, doi:July 2002 (Article in a Journal).

[5]. Douglas Fisher, Kathleen McKusick, Raymond Mooney, Jude W.Shalik, Geoffrey Towell, "Proceedings of the Sixth International Workshop on Machine Learning" Morgan Kaufmann Publishers, vol.6, June 1989, pp.169-173.

[6]. Simon Dennis ,http://staff.itee.uq.edu.au/janetw/cmc/chapters/BackProp/index2.html.

[7]. Donald R.Tveter, http://www.dontveter.com/pbr/public2.html.

[8]. Douglas D.Dankel II, http://www.cise.ufl.edu/~ddd/cap6635/Fall-97/short-papers/2.htm.

[9]. Kiyoshi Kawaguchi, http://wwwold.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node22.html.

[10]. Saduf, Mohd Arif Wani, "Comparative study of Back Propagation Learning Algorithms for Neural Networks", International Journal of Advanced Research in Computer Science and Software Engineering, vol.3, Dec. 2013, pp.1151-1156, doi: December 2013(Article in a Journal).

[11]. Y.h.VAVCOUVER, www.ece.ubc.ca/~yingh/review/shawlik91.html.

[12]. Raymond Mooney, Jude Shavlik, Geoffrey Towell, Alan Gove, "An Experimental Comparison of Symbolic and Connectionist Learning Algorithms", IJCAI'89 Proceedings of the 11th International Joint conference on Artificial Intelligence, vol.1, Aug.1989, pp.775-780, doi:1989-08-20(Article in a Conference Proceeding).

© 2015-19, IJARCS All Rights Reserved