International Journal of Computational and Electronics Aspects in Engineering
Vol. 1, Issue. 1, Dec 2014, Revised in March 2020, pp 25-34.
https://doi.org/10.26706/ijceae.1.1.20141205

An Overview of Hopfield Network and Boltzmann Machine

Saratha Sathasivam[1]
saratha@usm.my

Abdu Masanawa Sagir[2]
ams13_mah013@student.usm.my

Universiti Sains Malaysia,
School of Mathematical
Sciences, Pinang, Malaysia.

Abstract— **Neural networks are dynamic systems in the learning and training phase of their operations. The two well-known and commonly used types of recurrent neural networks, Hopfield neural network and Boltzmann machine have different structures and characteristics. This study gives an overview of Hopfield network and Boltzmann machine in terms of architectures, learning algorithms, comparison between these two networks from several different aspects as well as their applications.**

Index Terms— Artificial neural network, Boltzmann machine, Hopfield network, learning algorithms

I. INTRODUCTION

Hecht-Nielsen [1], neural network is a computing system made up of a number of simple, highly interconnected processing elements (neurons, units, cells, or nodes), which process information by their dynamic state response to external inputs. Neural network resembles the brain in two aspects: knowledge which is acquired by the network from its environment through a learning process; and interneuron connection strengths, known as synaptic weights that are used to store the acquired knowledge, Haykin, [2]. Neural network describes by its pattern of connections between the neurons (called its architecture), its method of determining the weights on the connections (called its training, or learning algorithm) and its activation function, Fausett [3]. Neural networks offer the following characteristics and properties as having adaptive learning, fault tolerance via redundant information, real time operation and self-organization. Sathasivam [4], neural network can be grouped into two major categories: feed-forward networks (e.g. Single-layer Perceptron, Multilayer Perceptron, Radial Basis Function) and feedback networks (e.g. Hopfield Network, Boltzmann Machine, Kohonen's SOM).

Hopfield network is a form of recurrent neural network (single layer feedback network) proposed as an associative memory system (content addressable memory CAM). It is also called Associative memories or Hopfield Memories, because it has been able to store the correct pattern. In addition, Hopfield network is a tool for solving an optimization problem. Hopfield network is global network which means all input vectors produces activation. Energy function (Lyapunov function) used in their input activation, which becomes exactly minimum, when creating a pattern. In Hopfield network the weight W is a symmetric matrix with zero diagonal elements which are necessary conditions for the convergence of an asynchronous totally connected network. Hopfield network can be implemented in two modes: synchronous mode in which the weighted input sums of all neurons are calculated without updating the neurons. Then all neurons are set to their new value, according to the value of their weighted input sum. The asynchronous mode on the other hand is one in which one picks one neuron, calculates the weighted input sum and updates immediately. This can be done in a fixed order, or neurons can be fired at random, which is called asynchronous random updating [8]. Generally, Hopfield networks can be classified into two popular forms: discrete

Technical Article
First Online on – 30 Dec 2014, Revised on – 30 March 2020

Cite this article – Saratha Sathasivam and Abdu Masanawa Sagir, "An Overview of Hopfield Network and Boltzmann Machine", *International Journal of Computational and Electronics Aspects in Engineering*, RAME Publishers, vol. 1, issue 1, pp. 25-34, 2014, Revised in 2020.
https://doi.org/10.26706/ijceae.1.1.20141205

and continuous-time models. The original formulation of the discrete Hopfield network showed the usefulness of the network as content-addressable memory. Later extensions, Hopfield [5], for continuous-valued activations can be used either for pattern association or constrained optimization.

Boltzmann Machine is a recurrent neural network, whose behavior can be described statistically in terms of simple interactions between the units consist in that network. Ackley et al [6], described Boltzmann machine as an extension of Hopfield network. Boltzmann machine is classified as a stochastic neural network which consists of one layer of visible units (neurons) and one layer of hidden units (neurons). The visible neurons provide an interface between the network and its environment. Sathasivam [7], the connections between neurons are bidirectional and it is a symmetric network. This shows that information can flows in both directions during the training and also during the usage of the network and the weights are the same in both directions. During the training phase, the visible neurons are usually clamped whereas the hidden neurons will always operate freely.

This paper focuses on the review work involving the study of Discrete Hopfield network and Boltzmann machine with learning. The paper is organized as follows: section 2 explains briefly an overview of artificial neural networks. Section 3 describes the learning algorithms, similarities and dissimilarities between Hopfield network and Boltzmann machine, this lead us to section 4 which described the applications of these two types of neural networks. Section 5 concludes the remarks.

II. An Overview of Artificial Neural Networks

Cochocki [8], the origin of artificial neural network may be traced back to as early as 1943, when McCulloch and Pitts wrote their landmark paper "A Logical Calculus of Ideas Immanent in Nervous Activity". Hamadneh, et.al [9], since then scientists, engineers, physicists, and biologists have been studying ways to mathematically model the human brain's ability to accept inputs from many, completely different types of sensors, analyze that data, decide on a course of action, trigger the proper response in its operational appendages, and learn from the results of that response. Galushkin [10], artificial neural network is a simplified mathematical model of the human brain. It is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It can be implemented by both electric elements and computer software. It is a parallel distributed processor with large numbers of connections; it is an information processing system that has certain performance characters in common with biological neural networks. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems.

Artificial neural networks, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. Pranob et al [11], ANN is a system loosely modeled on the human brain. The field goes by many names, such as connectionism; parallel distributed processing, neurocomputing, natural intelligent systems and machine learning algorithms. It is an attempt to simulate within specialized hardware or sophisticated software, the multiple layers of simple processing elements called neurons. Each neuron is linked to certain of its neighbors with varying coefficients of connectivity that represent the strengths of these connections.

According to Genevieve et al [12], Neural networks can be used in so many areas such include:

- Signal processing: suppress line noise, with adaptive echo canceling, blind source separation
- Control: e.g. backing up a truck: cab position, rear position, and match with the dock get converted to steering instructions. Manufacturing plants for controlling automated machines.

- Siemens successfully uses neural networks for process automation in basic industries, e.g., in rolling mill control more than 100 neural networks do their job, 24 hours a day
- Robotics - navigation, vision recognition
- Pattern recognition, i.e. recognizing handwritten characters, e.g. the current version of Apple's Newton uses a neural net
- Medicine, i.e. storing medical records based on case information
- Speech production: reading text aloud (NETtalk)
- Speech recognition
- Vision: face recognition , edge detection, visual search engines
- Business,e.g.. rules for mortgage decisions are extracted from past decisions made by experienced evaluators, resulting in a network that has a high level of agreement with human experts.
- Financial Applications: time series analysis, stock market prediction
- Data Compression: speech signal, image, e.g. faces
- Game Playing: backgammon, chess, go e.t.c.

A. Types of Applications in Neural Networks

Also, according to [12], there are different types of applications in neural network, such include:

Machine learning:

- Having a computer program itself from a set of examples so you don't have to program it yourself. Neural networks that learn from a set of examples.
- Optimization: given a set of constraints and a cost function, how do you find an optimal solution? E.g. traveling salesman problem.
- Classification: grouping patterns into classes: for e.g. handwritten characters into letters.
- Associative memory: recalling a memory based on a partial match.
- Regression: function mapping

Cognitive science:

- Modelling higher level reasoning: language and problem solving
- Modelling lower level reasoning: vision, audition speech recognition and speech generation

Neurobiology: Modelling models of how the brain works.

- neuron-level
- higher levels: vision, hearing, etc. Overlaps with cognitive folks.

Mathematics:

- Nonparametric statistical analysis and regression.

Philosophy:

- Can human souls/behavior be explained in terms of symbols, or does it require something lower level, like a neurally based model?

B. Hopfield Networks

Hopfield began with the system formulation proposed by McCulloch and Pitts, where the basic network consists of a set of neurons which compute the weighted sum of the inputs, then set the output to zero or one depending on their relation to a set threshold value, [8]. Hopfield neural networks have the following characteristics:

- They are primarily used as associative memories => content addressable memory.
- They can also be used to solve optimization problems.
- They have the same number of neurons as the dimension of the input and output pattern.
- The inputs to the network are provided as a set of initial conditions at the neuron outputs.
- The Hopfield network (by default) contains feedback, connecting the output of each neuron to the input of all the other neurons. Normally, self-feedback around neurons is omitted.
- The memory storage function is performed using an adaptation of Hebb's learning rule.
- There is a (relatively tight) limit on the number of training points which can be stored in the network's memory.

- In addition to the patterns which are stored by the user, there are extra patterns in memory, including:
  - Inverse patterns - patterns which are the inverse of the user-defined patterns, and
  - Spurious patterns - random patterns, the number of which increases with the number of stored patterns.
- Hopfield networks can accept continuous variables or binary variables (binary only considered here).
- Binary networks can utilize either [0, 1] or [1, -1]. [1, -1] is used here due to the resulting simplification in analysis.
- Hopfield networks can be either discrete time (using difference equations) or continuous time (using differential equations).

The Hopfield network can be made to operate in either discrete or continuous mode. Fig. 1.below is the Hopfield network Architecture:

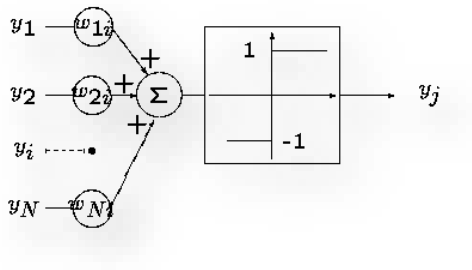

Fig. 1. Hopfield network architecture

But in our work, we would consider the discrete Hopfield network only.

In his first description, Hopfield [13], used binary input vectors. To store a set of binary patterns s(p), p = 1,…,P, where $s(p) = \left(s_i(p), \dots, s_i(p), \dots, s_n(p)\right)$, the weight matrix $W = \{w_{ij}\}$ is given by:

$$w_{ij} = \sum_p [2s_i(p) - 1][2s_j(p) - 1] \qquad (1)$$

$$for\ i \neq j, \text{and}\ w_{ii} = 0$$

where w_{ij} - the strength of the connection weight from neuron j to neuron i

s_i - the output of i^{th} unit

s_j - the state of unit j

Other descriptions, Hopfield [14], allows for bipolar inputs. To store a set of binary patterns s(p), p = 1,…,P, where $s(p) = \left(s_i(p), \dots, s_i(p), \dots, s_n(p)\right)$, the weight matrix $W = \{w_{ij}\}$ is given by:

$$w_{ij} = \sum_p s_i(p)s_j(p) \quad for\ i \neq j, \qquad (2)$$

$$\text{and}\ w_{ii} = 0$$

The energy function (or Lyapunov) for discrete Hopfield net is given by:

$$E = -0.5 \sum_{i \neq j} \sum_j y_i y_j w_{ij} - \sum_i x_i y_i + \sum_j \theta_i y_i \qquad (3)$$

θ_i - the threshold of unit i

If the activation of the net changes by an amount Δy_i, the energy changes by an amount

$$\Delta E = -\left[\sum_j y_j w_j + x_i - \theta_i\right] \Delta y_i \qquad (4)$$

Δy_i - activation of the net.

C. Boltzmann Machine

The Boltzmann machine, introduced by [6], is a novel approach to connectionist models using a distributed knowledge representation and a massively parallel network of simple stochastic computing elements. The computing elements are considered as logic units having two discrete states, viz. 'on' or 'off'. The units are connected to each other. With each connection, a connection strength is associated representing a quantitative measure of the hypothesis that the two connected units are both 'on'. Choosing the appropriate connections and their strengths is very difficult since different items may give rise to conflicting connection strengths. This major problem can be overcome by "learning" the Boltzmann machine the appropriate connection strengths. By applying a learning algorithm, the connection strengths are adjusted over a number of iterations (learning cycles) to adapt themselves to a given set of learning examples (e.g. the set of items N) that are subsequently clamped into a subset of the vertices of the Boltz-mann machine (i.e. the environmental vertices).

The Boltzmann machine is important because it is one of the first neural networks to demonstrate learning of latent

variables (hidden units). Boltzmann machine learning was at first slow to simulate, but the contrastive divergence algorithm of Geoff Hinton [15], allows models such as Boltzmann machines and Products Experts to be trained much faster. A Boltzmann machine, like a Hopfield network, is a network of units with an "energy" defined for the network. It also has binary units, but unlike Hopfield nets, Boltzmann machine units are stochastic.

The global energy, E, in a Boltzmann machine is identical in form to that of a Hopfield network:

$$\mathrm{E} = -\left(\sum_{i<j} W_{ij} S_i S_j + \sum_i \theta_i S_i\right) \qquad (5)$$

Where W_{ij} is the connection strength between unit j and unit i.

θ_i is the bias of unit I in the global energy function. ($-\theta_i$ is the activation threshold for the unit.)

S_i is the state, $S_i \in \{0,1\}$ of unit i.

The connections in a Boltzmann machine have two restrictions:

$W_{ii} = 0, \forall i$ (No unit has a connection with itself)

$W_{ij} = W_{ji}\ \forall i, j$ (All connections are symmetric)

Often the weights are represented in matrix form with a symmetric matrix $\boldsymbol{W}$, with zeros along the diagonal.

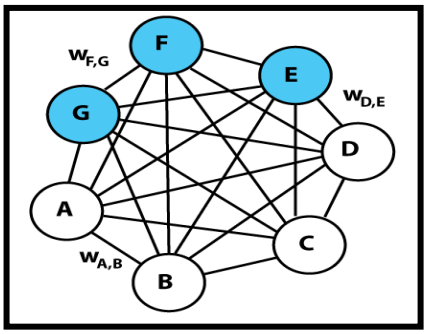

Fig. 2. Structure of representing Boltzmann machine

The structural representation of Boltzmann machine is shown in fig. 1. In this example there are 3 hidden units and 4 visible units.

In terms of energy minimization, the procedures are as follows:

(i) The Boltzmann machine works by picking a hidden unit at random - say unit j, and flipping the state of neuron j from S_j to $-S_j$ at temperature T (during the annealing cycle) with probability

$$Prob\ (S_j \to -S_j = \frac{1}{1 + e^{\left(\frac{-\Delta E}{T}\right)}} \qquad (6)$$

where ΔE_j is the energy change resulting from such a flip. The temperature parameter T is important for the method of simulated annealing. If the spinning procedure is applied continually to the units, the units will change state and it will ensure the relative probability of two global states is determined exclusively by their energy difference when the system reached the thermal equilibrium and follows a Boltzmann distribution:

$$\frac{P_\alpha}{P_\beta} = e^{-\left(\frac{E_\alpha - E_\beta}{T}\right)} \qquad (7)$$

where P_α is the probability of being in the $\alpha - th$ global state, and E_α is the energy of that state.

(ii) This summation runs over both visible and hidden units. The condition $j \neq i$ implies no self-feedback. W_{ji} is the weight from unit i to unit j. An external threshold θ_j applied to unit j is provided as usual (a weight of $-\theta_j$ from a unit with a fixed output of 1).

(iii) If the flipping procedure is applied repeatedly to the units, the net will reach thermal equilibrium. At thermal equilibrium, the units will change state, but the probability of finding the network in any particular state remains constant and obeys the Boltzmann distribution.

(iv) To find a stable configuration that is suited to the problem at hand, Boltzmann learning proceeds by first operating the net at high temperature, and gradually lowering it until it reaches thermal equilibrium at a series of temperatures, as prescribed by the simulated annealing procedure.

Other researchers have written on Hopfield network and Boltzmann machine such as Sathasivam [6], Zeng [17], Joya. et al [18], Gillblad [19], Barra et al [20] and Huang [21].

III. Materials And Methods

The authorized existing works involves in both softcopy and hardcopy were reviewed. The review work involves

learning algorithms of Hopfield network and Boltzmann machine, similarities, dissimilarities and applications of the two networks.

A. Learning Algorithms

According to Rabunal [22], the ability to learn is a fundamental trait of intelligence. An adaptation process whereby synapses, weights of artificial neural networks, classifiers strengths, or some other set of adjustable parameters is automatically modified so that some objectives is more readily achieved. In other words, procedures for modifying the weights on the connection links in a neural network (also known as training algorithms, learning rules) are referred to as learning algorithms [3]. As described by Zhang [23], there are three types of learning paradigms: supervised learning, unsupervised learning and reinforcement learning. Supervised learning attempts to learn a concept for correctly labeling unseen examples, where the training examples are with labels. Unsupervised learning attempts to learn the structure of the underlying sources of examples, where the training examples are with no label. Reinforcement learning attempts to learn a mapping from states to actions, where the examples are with no labels but with delayed rewards that could be viewed as delayed labels.

Sathasivam et al [24], explains that training involves adjusting the weights on the interconnections in the network until the error which is the difference between the actual output and the target is small. The essence of a learning algorithm is the learning rule, i.e. a weight-updating rule which determines how connection it can be used to solve the problem at hand by merely presenting the inputs to the network. The corresponding value of the output is found virtually instantaneously as the inputs are propagated through the network.

1. Learning Algorithms of Hopfield Networks

There are various different learning rules that can be used to store information in the memory of the Hopfield Network. It is desirable for a learning rule to have both of the following two properties:

- *Local*: A learning rule is *local* if each weight is updated using information available to neurons on either side of the connection that is associated with that particular weight.
- *Incremental*: New patterns can be learned without using information from the old patterns that have been also used for training. That is, when a new pattern is used for training, the new values for the weights only depend on the old values and on the new pattern, Stokey [25].

There are various different learning rules that can be used to store information in the memory of Hopfield network networks (Hopfield perceptron rule, Pseudo-Inverse rule, Contrastive divergent rule, learning rule by Wan Abdullah method e.t.c.). But here only three of the learning rules would be discussed, they are: Hebbian, Wan Abdullah & Stokey learning rules.

a. Habbian Learning Rule

Hebb's postulate of learning is the oldest and most famous of all learning rules; it is named in honor of the neurophysiologist Hebb. Hebbian learning rule is a learning rule that specifies how much the weight of the connection between two units (nodes) should be increased or decreased in proportion to the product of their activation. According to Morris [26];

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased"

According to Cheung et al [27], this can be reformulated into:

i. If two neurons on either side of a connection are activated simultaneously, the strength of the connection is increased.

ii. If two neurons on either side of a connection are activated asynchronously, the connection is weakened or eliminated.

However, the basic Hopfield learning rule, contrasted with that of Hebb, is implemented in the following manner, when learning n binary patterns:

$$W_{ij} = \eta \sum_{n} x_i^{(n)} x_j^{(n)} \qquad (8)$$

To prevent the weights from growing with the number of patterns, η is often set to $\frac{1}{N}$

with the training data $D = \{x^{(n)} I\, n = 1, \ldots, N\}$

Algorithm for the Discrete Hopfield network (Fausett, 1994), are as follows:

Step 0: Initialize weights to store patterns (Use Hebb rule)

While activations of the net are not converged, do steps 1–7.

Step 1: For each input vector x, do steps 2 – 6

Step 2: Set initial activations of net equal to the external input vector x: $y_i = x_{i,}\ (i = 1, \ldots, n)$

Step 3: Do steps 4 – 6 for each unit Y_i (Units should be updated in random order)

Step 4: Compute net input $= x_i \sum_j y_j w_{ji}$

Step 5: Determine activation (output signal):

$$y_i = \begin{cases} 1 & if \quad > \theta \\ y_i & if \quad = \theta \\ 0 & if \quad < \theta \end{cases} \qquad (9)$$

Step 6: Broadcast the value of y_i to all other units. (This updates the activation vector)

Step 7: Test for convergence

Here, the threshold, θ_i, is usually taken to be zero. The order of update of the units is random, but each unit must be updated at the same average rate.

b. Wan Abdullah Method

The algorithm of this network is best described by Sathasivam [28], which is known as direct method. It consists of the following basic steps:

(i) Given a logic program, translate all the clauses in the logic program into basic Boolean algebraic form.

(ii) Identify a neuron to each ground neuron.

(iii) Initialize all connections strengths to zero.

(iv) Derive a cost function that is associated with the negation of all the clauses, such that $\frac{1}{2}(1 + S_x)$ represents the logical value of a neuron X, where S_x is the neuron corresponding to X. The value of S_x is define in such a way that it carries the values of 1 if X is true and -1 if false. Negation (neuron X does not occur) is represented by multiplication whereas a disjunction connective is represented by addition.

(v) Obtain the values of connection strengths by comparing the cost function with the energy, H.

(vi) Let the neural networks evolve until minimum energy is reached. Check whether the solution obtained is a global solution.

c. The Storkey Learning Rule

This rule was introduced by [25] in 1997 and is both local and incremental. Storkey also showed that a Hopfield network trained using this rule has a greater capacity than a corresponding network trained using the Hebbian rule. The weight matrix of an attractor neural network is said to follow the Storkey learning rule if it obeys:

$$W_{ij}^{\ v} = W_{ij}^{\ v-1} + \frac{1}{n}\xi_i^v \xi_j^v - \frac{1}{n}\xi_i^v h_{ji}^v - \frac{1}{n}\xi_i^v h_{ij}^v \qquad (10)$$

where

$$h_{ij}^v = \sum_{k=1, k\neq 1, j}^{n} W_{ik}^{v-1}\ \xi_k^v \qquad (11)$$

is a form of local field at neuron i.

This learning rule is local, since the synapses take into account only neurons at their sides. The rule makes use of more information from the patterns and weights than the generalized Hebbian rule, due to the effect of the local field.

2. Learning Algorithms of Boltzmann Machine

According to [6], the goal of Boltzmann learning is to produce a neural network that categorizes input patterns according to a Boltzmann distribution. Two assumptions are made:

(a) Each environmental vector persists long enough for the network to reach thermal equilibrium;

(b) There is no structure in the sequence in which environmental vectors are clamped to the visible unit of the network. Boltzmann machine learning procedures are as follow:

i. Initialization: set weights to random numbers in [–1,1]

ii. Clamping Phase: Present the net with the mapping it is supposed to learn by clamping input and output units to patterns. For each pattern, perform simulated annealing on the hidden units at a sequence $T_0, T_1, \dots, T_{end}$ final of temperatures. At the final temperature, collect statistics to estimate the correlations

$$P_{ji}^{+} = \langle S_j S_i \rangle^{+}, j \neq i \quad (12)$$

iii. Free- Running Phase: Repeat the calculations performed in step 2, but at this time clamp only the input units. Hence, at the final temperature, estimate the correlations

$$P_{ji}^{-} = \langle S_j S_i \rangle^{-}, j \neq i \quad (13)$$

iv. Updating of Weights: update them using the learning rule

$$\Delta W_{ji} = \eta\left(P_{ji}^{+} - P_{ji}^{-}\right) \quad (14)$$

where η is a learning rate parameter, and it depends on the temperature $T\left(\eta = \frac{\varepsilon}{T}\right)$

v. Iterate until Convergence: Iterate steps 2 to 4 until the learning procedure converges with no more changes taking place in the synaptic weights W_{ji} for all j, i

B. Similarities and Dissimilarities between Hopfield Network and Boltzmann Machine.

TABLE I

SIMILARITIES

▪ Outputs must be binary (or bipolar) in nature, i.e. continuous values are not supported.
▪ Weights are symmetric. I.e. weight from neuron u_1 to u_2 is the same as from u_2 to u_1 ($W_{ij} = W_{ji}$ for $i \neq j$)
▪ No self connections i.e. $W_{ii} = 0$
▪ Asynchronous update of neuron activations
▪ Energy function for a Boltzmann machine has the same form as for Hopfield networks

TABLE II

DISSIMILARITIES

▪ Hopfield cannot have hidden units.	▪ A Boltzmann machine can have hidden units.
▪ Hopfield network is a deterministic recurrent neural network.	▪ Boltzmann machine is a stochastic network.
▪ Hopfield nets are typically used for pattern completion tasks and as such will undergo unsupervised learning (at least for autoassociators).	▪ Boltzmann machines can be configured to solve and generalize for supervised learning problems too.
▪ Phase Storage and Phase Retrieval	▪ Clamping Phase and Free-Running Phase

IV. APPLICATIONS

A. Applications of Hopfield Networks

The Travelling Salesman Problem (TSP) is a well known problem which can be solved using Hopfield Networks. The TSP solution with Hopfield Networks is based on the uniqueness constraint. This is, each city must be visited once and only once while trying to minimize the travelling distance. Hu [29], the travelling salesman problem (TSP) is a classical problem in combinational optimization. The task is to find the shortest possible tour through a set of N cities that passes through each city exactly once. It is generally believed that the computational power needed to solve it grows exponentially with the number of the cities.

Du [30], the Hopfield network can be used as an effective interface between analog and digital devices, where the input signals to the network are analog and the output signals are discrete values. Associative memory is a major application of the Hopfield network, the fixed points in the network energy function are used to store feature patterns. When a noisy or incomplete pattern is presented to the trained network, a pattern in the memory is retrieved. This property is most useful for pattern recognition or pattern completion.

B. Applications of Boltzmann Machine

Boltzmann Machine in stock market trend prediction, character recognition, Face recognition, Internet Application, Cancer Detection, Loan Application, Decision making and many more, [6]. Aarts [31], applications of Boltzmann machine include in future computer architectures, knowledge representation in intelligent systems, modeling of neural behaviours of the brain, use for combinatorial optimization problems.

V. CONCLUSION

This paper has been dedicated to review of Discrete Hopfield Network and Boltzmann machine with learning. We have analyzed the Hopfield network can perform some of the functions of memory recall in a manner analogous to the brain functions. The Hopfield neural network (HNN) is applicable for solving optimization or mathematical programming (MP) problems. The major advantage of HNN is in its structure can be realized on an electronic circuit, possibly on a VLSI (very large-scale integration) circuit, for an on-line solver with a parallel-distributed process. On the other hand, Boltzmann machine is an extension of a Hopfield network, have several applications benefit from characteristics of this network such include in stock market trend prediction, character recognition, modeling of neural behaviours of the brain, use for combinatorial optimization problems, Cancer Detection and Loan Application to mentioned few.

ACKNOWLEDGMENT

This research was supported by FRGS grant (203/PMATHS/6711368) offered by Ministry of Higher Education and Universiti Sains Malaysia.

REFERENCES

[1] Hecht-Nielsen, R. The Basic introduction to Neural networks, 1989. Retrived on 17th September, 2014 @ http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

[2] Haykin S. Neural Networks: A comprehensive foundation. Prentice Hall, Inc, USA. ISBN 0780334949, 9780780334946, 1999.

[3] Fausett, L.V. Fundamentals of Neural networks: Architectures, algorithms and applications. Prentice-Hall,Inc, New Jersey, U.S.A, 1994. ISBN 0-13-334186-0

[4] Sathasivam, S., & Abdullah, W. A. T. W. *Logic learning in Hopfield networks*. *arXiv preprint arXiv:0804.4075*, 2008.

[5] Hopfield, J. J., & Tank, D. W. "Neural" computation of decisions in optimization problems. *Biological cybernetics*, *52*(3), 141-152, 1985.

[6] Ackley, D.H., Hinton, G.E. and Sejnowski, T.J. A learning algorithm for Boltzmann machines, Cognitive Science (9), 1985, 147-169.

[7] Sathasivam, S. and Abdullah W. *Logic mining in neural network: Reverse analysis method*. Computing, 91(2): 2011, P 119-133.

[8] Cochocki A. and Unbehauen R. *Neural Networks for Optimization and Signal Processing*. John Wiley & Sons, Inc., 1993.

[9] Hamadneh N. S., Sathasivam S. L., Tilahun and Choon O. H. *Learning logic programming in radial basis function network via genetic algorithm*. J. Appl. Sci., 12(9): 2012, P 840-847.

[10] Galushkin, A.T. Neural Networks theory. Springer, Berlin, Germany. xx,396, 2007, p.176 illus. ISBN 978-3-540-48125-6

[11] Pranob, K., Charles, H.K., Rajesh-Kumar, C., Nikhita, N., Santhosh, R., Harish, V., & Swathi, M. Artificial Neural Network Based Image Compression using Levenberg – Marquardt Algorith. *International Journal of Modern Engineering Research (IJMER)*, 1(2), 2012, pp. 482 – 489.

[12] Genevieve, O., Schraudolph, N. and Cummins F. Lecture Notes on Neural Networks, (1999). Willamette University, Portland. Retrieved on 2nd October, 2014 @ http://www.willamette.edu/~gorr/classes/cs449/intro.html

[13] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 1982, 2554-2558.

[14] Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, *81*(10), 1984, 3088-3092.

[15] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye. A fast learning algorithm for deep belief nets. Neural Computation, 18(7):1527–1554, 2006.

[16] Sathasivam, S. Improving Logic Programming in Hopfield Network with Sign Constrained. *SSJ*, *1*(2), 2, 2009.

[17] Zeng, X., & Martinez, T. R. Improving the performance of the Hopfield network by using a relaxation rate. In *Artificial Neural Nets and Genetic Algorithms*, 1999, January, pp. 67-72. Springer Vienna.

[18] Joya, G., Atencia, M. A., & Sandoval, F. Hopfield neural networks for optimization: study of the different dynamics. *Neurocomputing*, *43*(1), 2002, 219-237.

[19] Glilblad, D. Feedback Networks and Hopfield Networks, 2008. Retrieved on 30thSeptember, 2014 @ www.csc.kth.se/utbildning/kth/kurser/DD2432/ann14/.../07-hopfield.pdf

[20] Barra, A., Bernacchia, A., Santucci, E., & Contucci, P. On the equivalence of Hopfield networks and Boltzmann Machines. *Neural Networks*, 2012, *34*, 1-9.

[21] Huang, H. Reconstructing the Hopfield network as an inverse Ising problem. *Physical Review E*, *81*(3), 036104. 2010.

[22] Rabunal, J. R., & Dorado, J. (Eds.). *Artificial neural networks in real-life applications*. IGI Global. 2006.

[23] Zhang, M. L., & Zhou, Z. H. Adapting RBF neural networks to multi-instance learning. *Neural Processing Letters*, *23*(1), 2006, 1-26.

[24] Sathasivam, S., Hamadneh, N., & Choon, O. H. Comparing Neural Networks: Hopfield Network and RBF Network. *Applied Mathematical Sciences*, *5*(69), 2011, 3439-3452.

[25] Storkey, A. "Increasing the capacity of a Hopfield network without sacrificing functionality." Artificial Neural Networks – ICANN'97, 1997: 451-456.

[26] Morris, R. G. DO Hebb: The Organization of Behavior, Wiley: New York; 1949. *Brain research bulletin*, *50*(5-6), 1998, 437-437.

[27] Cheung, K. F., Atlas, L. E., & Marks II, R. J. Synchronous vs asynchronous behavior of Hopfield's CAM neural net. *Applied Optics*, *26*(22), 1987, 4808-4813.

[28] Sathasivam, S., & Fen, N. P. Developing Agent Based Modelling for Doing Logic Programming in Hopfield Network. *Applied Mathematical Sciences*, *7*(1), 2013, 23-35.

[29] Hu, J. E., & Siy, P. The ordering-oriented Hopfield network. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference*, Vol. 7, 1994, pp. 4693-4698). IEEE.

[30] Du, K. L., & Swamy, M. N. *Neural networks in a softcomputig framework*, 2006, pp. 207 – 215. London: Springer.

[31] Aarts, E. H., & Korst, J. H. Boltzmann machines and their applications. In *PARLE Parallel Architectures and Languages Europe* , 1987, January), pp. 34-50. Springer Berlin Heidelberg.