Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 7, pp. 18307-18328, July, 2017

ISSN: 0976-3031

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

ARTIFICIAL NEURAL NETWORK

Waleed Khalid Shihab and Sivaram Prasad R

Department of Commerce and Business Administration
Acharya Nagarjuna University

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0807.0485

ARTICLE INFO

Article History:

Received 05th April, 2017
Received in revised form 21st May, 2017
Accepted 06th June, 2017
Published online 28th July, 2017

Key Words:

Network architecture, layer of neuron, activation function, Application of neural network, learning

ABSTRACT

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. 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 (neurones) working in unison to solve specific problems. ANNs, 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 neurones. This is true of ANNs as well.

INTRODUCTION

How the brain works

How to teach a computer? You can either write a fixed program-or you can enable the computer to learn on its own. Living beings do not have any programmer writing a program for developing their skills, which then only has to be executed. They learn by themselves-without the previous knowledge from external impressions-and thus can solve problems better than any computer today. What qualities are needed to achieve such a behaviour for devices like computers? Can such cognition be adapted from biology? History, development, decline and resurgence of a wide approach to solve problems.(David kriesel,2005 p. 3)

'The computer hasn't proved anything yet,' angry Garry Kasparov, the world chess champion, said after his defeat in New York in May 1997. 'If we were playing a real competitive match, I would tear down Deep Blue into pieces.' But Kasparov's efforts to downplay the significance of his defeat in the sixgame match was futile. The fact that Kasparov-probably the greatest chess player the world has seen-was beaten by a computer marked a turning point in the quest for intelligent machines. (Michael Negnevitsky, 2005, p.165) Vincent Cheung, Kevin Cannons numerous advances have been made in developing intelligent system some inspired by biological neural networks. Researches from many scientific disciplines are designing artificial neural networks (A"s) to solve a variety of problems in pattern recognition, prediction, optimization, associative memory, and control. Conventional approaches have been proposed for solving these problems.

Although successful applications can be found in certain well-constrained environments, none is flexible enough to perform well outside its domain. ANNs provide exciting alternatives, and many applications could benefit from using them ("Ani1 K. Jaina, Jianchang Mao, K.M. Mohiuddin:p 31).

A neural network is an interconnected assembly of simple processing elements, *units* or *nodes,* whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter unit connection strengths, or *weights,* obtained by a process of adaptation to, or *learning* from, a set of training patterns.

Artificial neural networks (ANN) have been developed as generalizations of mathematical models of biological nervous systems. A first wave of interest in neural networks (also known as *connectionist models* or *parallel distributed processing*) emerged after the introduction of simplified neurons by McCulloch and Pitts (1943). The basic processing elements of neural networks are called *artificial neurons*, or *simply neurons* or *nodes*. In a simplified mathematical model of

Corresponding author:* **Waleed Khalid Shihab
Department of Commerce and Business Administration Acharya Nagarjuna University

the neuron, the effects of the synapses are represented by connection weights that modulate the effect of the associated input signals, and the nonlinear characteristic exhibited by neurons is represented by a transfer function. The neuron impulse is then computed as the weighted sum of the input signals, transformed by the transfer function. The learning capability of an artificial neuron is achieved by adjusting the weights in accordance to the chosen learning algorithm. (Ajith Abraham)

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function). Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron the sum of previously weighted inputs and bias is passing trough activation function that is also called transfer function. (Andrej Krenker, Janez Bešter and Andrej Kos, 2011, p1)

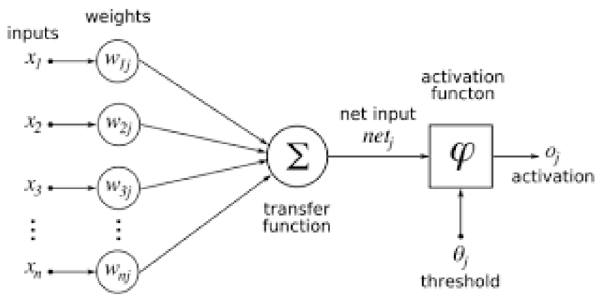


Fig 1 Working principle of an artificial neuron

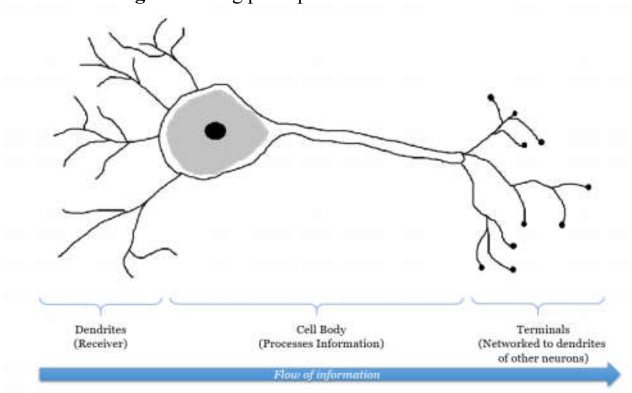

Dendrites (Receiver)
Cell Body (Processes Information)
Terminals (Networked to dendrites of other neurons)
Flow of information

Fig 2 Working principle of an biological neuron

(www.google.co.in)

History of artificial neural network

(www.psych.utoronto.ca: N. Yadav *et al*; 2015) the study of the human brain is thousands of years old. With the advent of modern electronics, it was only natural to try to harness this thinking process. The first step toward artificial neural networks came in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They modeled a simple neural network with electrical circuits.

Reinforcing this concept of neurons and how they work was a book written by Donald Hebb. The *Organization of Behavior* was written in 1949. It pointed out that neural pathways are strengthened each time that they are used.

As computers advanced into their infancy of the 1950s, it became possible to begin to model the rudiments of these theories concerning human thought. Nathanial Rochester from the IBM research laboratories led the first effort to simulate a neural network. That first attempt failed. But later attempts were successful. It was during this time that traditional computing began to flower and, as it did, the emphasis in computing left the neural research in the background.

Yet, throughout this time, advocates of "thinking machines" continued to argue their cases. In 1956 the Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks. One of the outcomes of this process was to stimulate research in both the intelligent side, AI, as it is known throughout the industry, and in the much lower level neural processing part of the brain.

In the years following the Dartmouth Project, John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes. Also, Frank Rosenblatt, a neuro-biologist of Cornell, began work on the Perceptron. He was intrigued with the operation of the eye of a fly. Much of the processing which tells a fly to flee is done in its eye. The Perceptron, which resulted from this research, was built in hardware and is the oldest neural network still in use today. A single-layer perceptron was found to be useful in classifying a continuous-valued set of inputs into one of two classes. The perceptron computes a weighted sum of the inputs, subtracts a threshold, and passes one of two possible values out as the result. Unfortunately, the perceptron is limited and was proven as such during the "disillusioned years" in Marvin Minsky and Seymour Papert's 1969 book *Perceptrons*.

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple ADAptive LINear Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter which eliminates echoes on phone lines. This neural network is still in commercial use.

Unfortunately, these earlier successes caused people to exaggerate the potential of neural networks, particularly in light of the limitation in the electronics then available. This excessive hype, which flowed out of the academic and technical worlds, infected the general literature of the time. Disappointment set in as promises were unfilled. Also, a fear set in as writers began to ponder what effect "thinking machines" would have on man. Asimov's series on robots revealed the effects on man's morals and values when machines where capable of doing all of mankind's work. Other writers created more sinister computers, such as HAL from the movie 2001.

These fears, combined with unfulfilled, outrageous claims, caused respected voices to critique the neural network research. The result was to halt much of the funding. This period of stunted growth lasted through 1981.

In 1982 several events caused a renewed interest. John Hopfield of Caltech presented a paper to the national Academy of Sciences. Hopfield's approach was not to simply model brains but to create useful devices. With clarity and mathematical analysis, he showed how such networks could work and what they could do. Yet, Hopfield's biggest asset was his charisma. He was articulate, likeable, and a champion of a dormant technology.

At the same time, another event occurred. A conference was held in Kyoto, Japan. This conference was the US-Japan Joint Conference on Cooperative/Competitive Neural Networks. Japan subsequently announced their Fifth Generation effort. US periodicals picked up that story, generating a worry that the US could be left behind. Soon funding was flowing once again.

By 1985 the American Institute of Physics began what has become an annual meeting-Neural Networks for Computing. By 1987, the Institute of Electrical and Electronic Engineer's (IEEE) first International Conference on Neural Networks drew more than 1,800 attendees.

By 1989 at the Neural Networks for Defense meeting Bernard Widrow told his audience that they were engaged in World War IV, "World War III never happened," where the battlefields are world trade and manufacturing. The 1990 US Department of Defense Small Business Innovation Research Program named 16 topics which specifically targeted neural networks with an additional 13 mentioning the possible use of neural networks.

Today, neural networks discussions are occurring everywhere. Their promise seems very bright as nature itself is the proof that this kind of thing works. Yet, its future, indeed the very key to the whole technology, lies in hardware development. Currently most neural network development is simply proving that the principal works. This research is developing neural networks that, due to processing limitations, take weeks to learn. To take these prototypes out of the lab and put them into use requires specialized chips. Companies are working on three types of neuro chips - digital, analog, and optical. Some companies are working on creating a "silicon compiler" to generate a neural network Application Specific Integrated Circuit (ASIC). These ASICs and neuron-like digital chips appear to be the wave of the near future. Ultimately, optical chips look very promising. Yet, it may be years before optical chips see the light of day in commercial applications.

1943 McCulloch and Pitts proposed the McCulloch-Pitts neuron model

1949 Hebb published his book *The Organization of Behavior*, in which the Hebbian learning rule was proposed.

1958 Rosenblatt introduced the simple single layer networks now called Perceptrons.

1969 Minsky and Papert's book *Perceptrons* demonstrated the limitation of single layer perceptrons, and almost the whole field went into hibernation.

1982 Hopfield published a series of papers on Hopfield networks.

1982 Kohonen developed the Self-Organising Maps that now bear his name.

1986 The Back-Propagation learning algorithm for Multi-Layer Perceptrons was rediscovered

and the whole field took off again.

1990s The sub-field of Radial Basis Function Networks was developed.

2000s The power of Ensembles of Neural Networks and Support Vector Machinesbecomes apparent.(John A. Bullinaria, 2004).

Organization of Nns Based on Their Functional Characteristics

Functional characteristics	Structure
Pattern recognition	MLP, Hopfield, kohonen, PNN
Associative memory	Hopfield, recurrent MLP,kohonen
Optimization	Hopfield, ART, CNN
Function approximation	MLP, CMAC,FLN, RBF
Modelling and control	MLP, recurrent MLP, CMAC,FLN, FPN
Image processing	CNN, Hopfield
Classification (including clustering)	MLP, Kkohonen,RBF,ART,PNN

(Magali R. G. Meireles, Paulo E. M. Almeida,*2003)*

The objectives of artificial neural network

As you read these words you are using a complex biological neural network. You have a highly interconnected set of some 1011 neurons to facilitate your reading, breathing, motion and thinking. Each of your biological neurons, a rich assembly of tissue and chemistry, has the complexity, if not the speed, of a microprocessor. Some of your neural structure was with you at birth. Other parts have been established by experience. Scientists have only just begun to understand how biological neural networks operate. It is generally understood that all biological neural functions, including memory, are stored in the neurons and in the connections between them. Learning is viewed as the establishment of new connections between neurons or the modification of existing connections. This leads to the following question: Although we have only a rudimentary understanding of biological neural networks, is it possible to construct a small set of simple artificial "neurons" and perhaps train them to serve a useful function? The answer is "yes." As will explain hereafter.(15)

Why artificial neural network

The long course of evolution has given the human brain many desirable characteristics not present invon Neumann or modern parallel computers.

These include

- massive parallelism,
- distributed representation and computation,
- learning ability,
- generalization ability,
- adaptively,
- inherent contextual information processing,
- fault tolerance, and
- low energy consumption.(Ani1 K. Jain, Jianchang Mao, K.M. Mohiuddin)

Neural networks are often used for statistical analysis and data modelling, in which their role is perceived as an alternative to standard nonlinear regression or cluster analysis techniques (Cheng & Titterington 1994). Thus, they are typically used in problems that may be couched in terms of classification, or forecasting. Some examples include image and speech recognition, textual character recognition, and domains of human expertise such as medical diagnosis, geological survey for oil, and financial market indicator prediction. This type of problem also falls within the domain of classical artificial intelligence (AI) so that engineers and computer scientists see neural nets as offering a style of *parallel distributed computing,* thereby providing an alternative to the conventional algorithmic techniques that have dominated in machine intelligence. This is a theme pursued further in the final chapter but, by way of a brief explanation of this term now, the parallelism refers to the fact that each node is conceived of as operating independently and concurrently (in parallel with) the others, and the "knowledge" in the network is distributed over the entire set of weights, rather than focused in a few memory locations as in a conventional computer. The practitioners in this area do not concern themselves with biological realism and are often motivated by the ease of implementing solutions in digital hardware or the efficiency and accuracy of particular techniques. Haykin (1994) gives a comprehensive survey of many neural network techniques from an engineering perspective.

Neuroscientists and psychologists are interested in nets as computational models of the animal brain developed by abstracting what are believed to be those properties of real nervous tissue that are essential for information processing. The artificial neurons that connectionist models use are often extremely simplified versions of their biological counterparts and many neuroscientists are sceptical about the ultimate power of these impoverished models, insisting that more detail is necessary to explain the brain's function. Only time will tell but, by drawing on knowledge about how real neurons are interconnected as local "circuits", substantial inroads have been made in modelling brain functionality. A good introduction to this programme of *computational neuroscience* is given by Churchland & Sejnowski (1992).

Finally, physicists and mathematicians are drawn to the study of networks from an interest in nonlinear dynamical systems, statistical mechanics and automata theory.1

It is the job of applied mathematicians to discover and formalize the properties of new systems using tools previously employed in other areas of science. For example, there are strong links between a certain type of net (the Hopfield net-see Ch. 7) and magnetic systems known as spin glasses. The full mathematical apparatus for exploring these links is developed (alongside a series of concise summaries) by Amit (1989).

All these groups are asking different questions: neuroscientists want to know how animal brains work, engineers and computer scientists want to build intelligent machines and mathematicians want to understand the fundamental properties of networks as complex systems. Another (perhaps the largest) group of people are to be found in a variety of industrial and commercial areas and use neural networks to model and analyze large, poorly understood datasets that arise naturally in their workplace. It is therefore important to understand an author's perspective when reading the literature. Their common focal point is, however, neural networks and is potentially the basis for close collaboration. For example, biologists can usefully learn from computer scientists what computations are necessary to enable animals to solve particular problems, while engineers can make use of the solutions nature has devised so that they may be applied in an act of "reverse engineering".

Building Blocks of Artifical Neural Network

The basic building blocks of the artificial neural network are

1. Network architecture
2. Setting the weights
3. Activation function(52)

Network architectures

The basic architecture consists of three types of neuron layers: input, hidden, and output layers.(Ajith Abraham) ANNs may be described as a network of interconnected neurons sometimes called nodes) (C. W. Dawson, Robert Wilby,2001) (Cameron M. Zealand, Donald H. Burn, Slobodan P. Simonvic,1999) (Muriel Gevrey, Ioannis Dimopoulos, Sovan Lek, 2003) (Xin Yao,1997) (J¨urgen Schmidhuber,2014) (kaastra,M boyd, 1996), (Bimal K. Bose,2007), (Kishan Mehrotra, Chilukuri K. Mohan,1997) (xin Yao and Yong Liu) (J. M. Ben´ıtez, J. L. Castro, and I. Requena,1997) (Ajith Abraham) (Jocelyn Sietsma and Robert J.F.Dow,1991) ANN models attempt to achieve good performance through dense interconnections of simple computational elements. In this respect, the architectures of ANNs are based on the present understanding of the biological nervous systems. ANNs offer valuable characteristics unavailable together elsewhere. First, they infer solutions from data without prior knowledge of the regularities in the data; they extract the regularities empirically. Second, these networks learn the similarities among patterns directly from instances or examples of them. ANNs can modify their behaviour in response to the environment (i.e. shown a set of inputs with corresponding desired outputs, they self adjust to produce consistent responses). Third, ANNs can generalize from previous examples to new ones. Generalization is useful because real-world data are noisy, distorted, and often incomplete. Fourth, ANNs are also very good at the abstraction of essential characteristics from inputs containing irrelevant data. Fifth, they are non-linear, that is, they can solve some complex problems more accurately than linear techniques do. Finally, ANNs are highly parallel. They contain many identical, independent operations that can be executed simultaneously, often making them faster than alternative methods.

The biases and connection weights of the ANN are evaluated through an optimization process that starts by splitting the dataset into two parts: the training dataset and the validation dataset. Training data are used by the ANN to learn how the system behaves, a process which ultimately results in the specification of biases and weights. Validation data are used to assess the performance of the ANN in making predictions. We used 66% (corresponding to 1522 observations)of the dataset to train the ANN, and the remaining part (784 observations) for validation.(B. van Maanen1,2, G. Coco1, K. R. Bryan2, and B. G. Ruessink3,2010)

Layer of Neurons

A single-*layer* network of S neurons is shown in Figure 2.7. Note that each of the R inputs is connected to each of the neurons and that the weight matrix now has S rows.

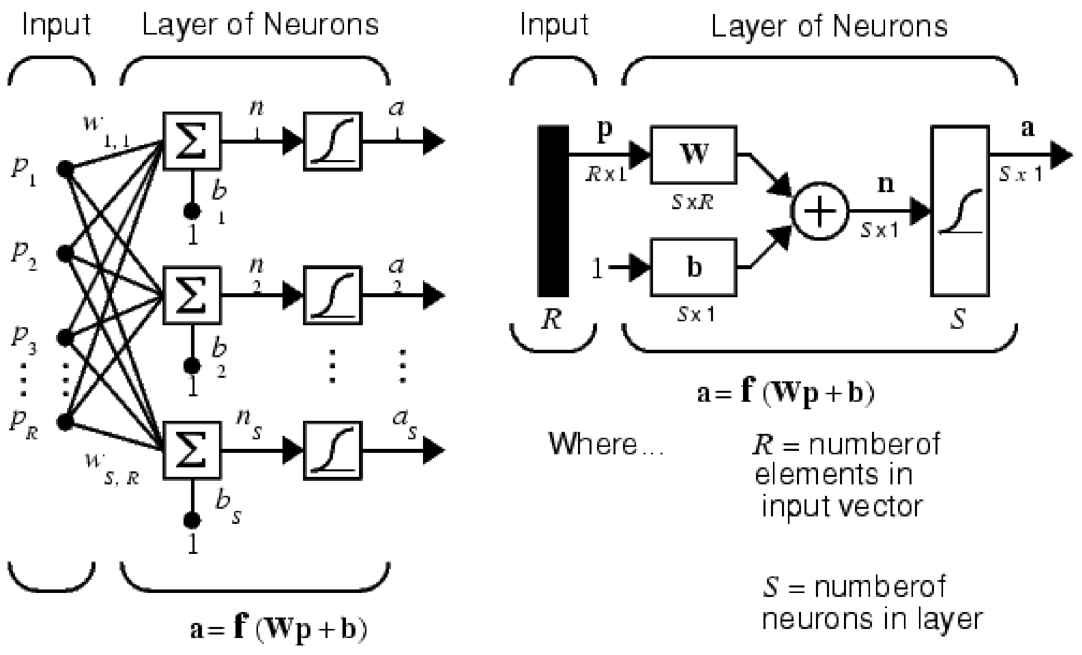


Figure 7 Layer of *S* Neurons

(encrypted.google.com)

The layer includes the weight matrix, the summers, the bias vector, the transfer function boxes and the output vector . Some authors refer to the inputs as another layer, but we will not do that here.

Each element of the input vector p is connected to each neuron through the weight matrix W. Each neuron has a bias bi, a summer, a transfer function f and an output ai.

Taken together, the outputs form the output vector a. (Miloslav Vilimek, 2014)

It is common for the number of inputs to a layer to be different from the number of neurons (i.e., R ≠ S).

You might ask if all the neurons in a layer must have the same transfer function. The answer is no; you can define a single (composite) layer of neurons having different transfer functions by combining two of the networks.

shown above in parallel. Both networks would have the same inputs, and each network would create some of the outputs.

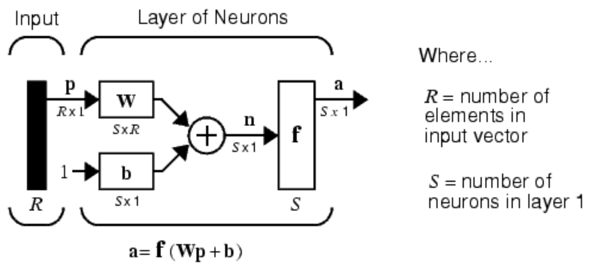
The input vector elements enter the network through the weight matrix **W**:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

(encrypted.google.com)

As noted previously, the row indices of the elements of matrix W indicate the destination neuron associated with that weight, while the column indices indicate the source of the input for that weight.

Thus, the indices in *w*3,2 say that this weight represents the connection *to* the third neuron *from* the second source. Fortunately, the *S*-neuron, *R*-input, one-layer network also can be drawn in abbreviated notation, as shown in Figure



Layer of Neurons, Abbreviated Notation

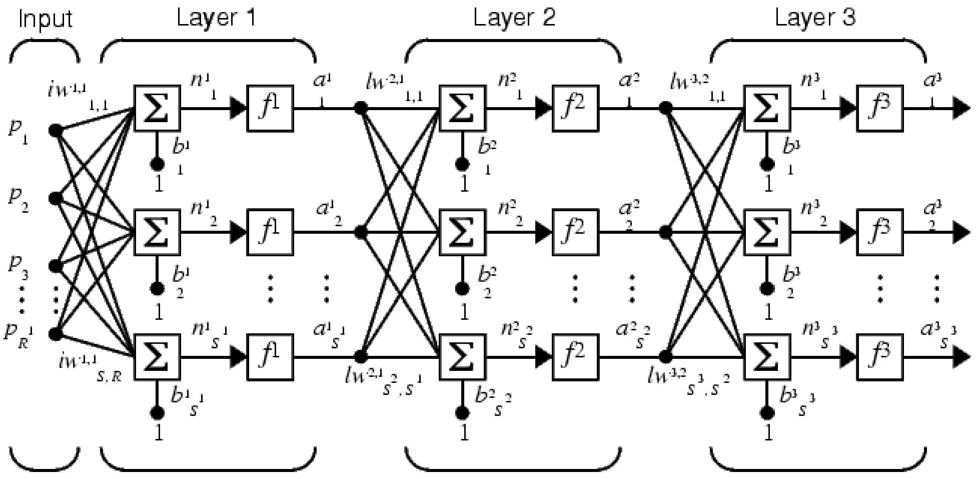
Here again, the symbols below the variables tell you that for this layer, p is a vector of length *R,* W is an *S* × *R* matrix, and a and b are vectors of length. As defined previously, the layer includes the weight matrix, the summation and multiplication operations, the bias vector b, the transfer function boxes and the output vector.

Multiple Layers of Neurons

Now consider a network with several layers. Each layer has its own weight matrix W, its own bias vector b, a net input vector n and an output vector a. We need to introduce some additional notation to distinguish between these layers. We will use superscripts to identify the layers. Specifically, we append the number of the layer as a *superscript* to the names for each of these variables. Thus, the weight matrix for the first layer is written as W1 and the weight matrix for the second layer is written as W2. This notation is used in the three-layer network shown in Figure

Three-Layer Network

As shown, there are *R* inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. As noted, different layers can have different numbers of neurons.

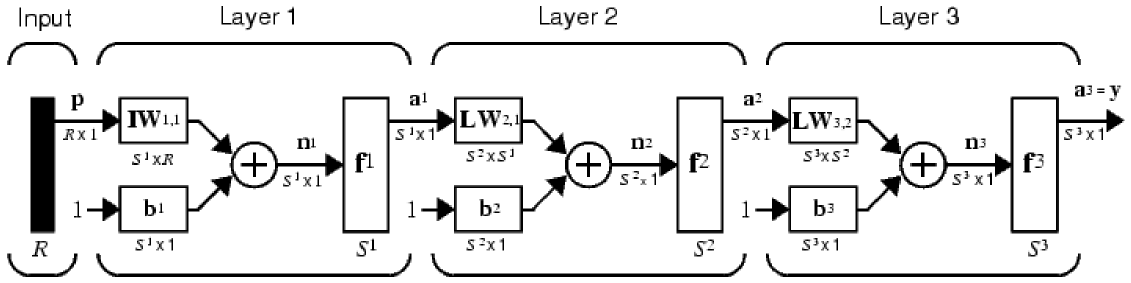


$$a^1 = f^1(IW^{1,1}p + b^1) \quad a^2 = f^2(LW^{2,1}a^1 + b^2) \quad a^3 = f^3(LW^{3,2}a^2 + b^3)$$

$$a^3 = f^3(LW^{3,2}f^2(LW^{2,1}f^1(IW^{1,1}p + b^1) + b^2) + b^3)$$

(encrypted.google.com)

The outputs of layers one and two are the inputs for layers two and three. Thus layer 2 can be viewed as a one-layer network with $R=S^1$ inputs, $S=S^2$ neurons, and an $S^{2\times}S^1$ weight matrix $\mathbf{W}^2$. The input to layer 2 is a^1, and the output is a^2 . A layer whose output is the network output is called an *output layer*. The other layers are called *hidden layers*. The network shown above has an output layer (layer 3) and two hidden layers (layers 1 and 2). The same three-layer network discussed previously also can be drawn using our abbreviated notation, as shown in Figure



$$a^1 = f^1(IW^{1,1}p + b^1) \quad a^2 = f^2(LW^{2,1}a^1 + b^2) \quad a^3 = f^3(LW^{3,2}a^2 + b^3)$$

$$a^3 = f^3(LW^{3,2}f^2(LW^{2,1}f^1(IW^{1,1}p + b^1) + b^2) + b^3 = y$$

(encrypted.google.com)

Three-Layer Network, Abbreviated Notation

Multilayer networks are more powerful than single-layer networks. For instance, a two-layer network having a sigmoid first layer and a linear second layer can be trained to approximate most functions arbitrarily well. Single layer networks cannot do this.

At this point the number of choices to be made in specifying a network may look overwhelming, so let us consider this topic. The problem is not as bad as it looks. First, recall that the number of inputs to the network and the number of outputs from the network are defined by external problem specifications.

So if there are four external variables to be used as inputs, there are four inputs to the network. Similarly, if there are to be seven outputs from the network, there must be seven neurons in the output layer. Finally, the desired characteristics of the output signal also help to select the transfer function for the output layer. If an output is to be either -1or 1, then a symmetrical hard limit transfer function should be used. Thus, the architecture of a single-layer network is almost completely determined by problem specifications, including the specific number of inputs and outputs and the particular output signal characteristic.

Now, what if we have more than two layers? Here the external problem does not tell you directly the number of neurons required in the hidden layers. In fact, there are few problems for which one can predict the optimal number of neurons needed in a hidden layer. This problem is an active area of research.

As for the number of layers, most practical neural networks have just two or three layers. Four or more layers are used rarely.

We should say something about the use of biases. One can choose neurons with or without biases. The bias gives the network an extra variable, and so you might expect that networks with biases would be more powerful than those without, and that is true. Note, for instance, that a neuron without a bias will always have a net input n of zero when the network inputs **p** are zero. This may not be desirable and can be avoided by the use of a bias. In later chapters we will omit a bias in some examples or demonstrations.

In some cases this is done simply to reduce the number of network parameters. With just two variables, we can plot system convergence in a two-dimensional plane. Three or more variables are difficult to display.

Weights Settings

The process of learning or training depends on setting the value for the weights. The process of modifying the weights in connection between network layers with the objective of achieving the expected outputs is called training a network. Learning is the internal process that takes place when a network is trained.(52)

learning an Artificial Neural Network

(Wulfram Gerstner) (Jure Zupan, 1994) (Jan Drchal, 2012) One of the most relevant features of artificial neural networks is their capability of learning from the presentation of samples (patterns), which expresses the system behaviour This is accomplished by presenting the network with a number of training samples (C.W. Dawson1 and R.L. Wilby,2001). Hence, after the network has learned the relationship between inputs and outputs, it can generalize solutions, meaning that the network can produce an output which is close to the expected (or desired) output of any given input values. Therefore, the training process of a neural network consists of applying the required ordinate steps for tuning the synaptic weights and thresholds of its neurons, in order to generalize the solutions produced by its outputs. The set of ordinate steps used for training the network is called learning algorithm. During its execution, the network will thus be able to extract discriminate features about the system being mapped from samples acquired from the system.

Learn the connection weights from a set of training examples and different network architectures required different learning algorithms()24 بلا رقم بعر updating network architecture and connection weights so that network can efficiently perform a task(Negin Yousefpour)

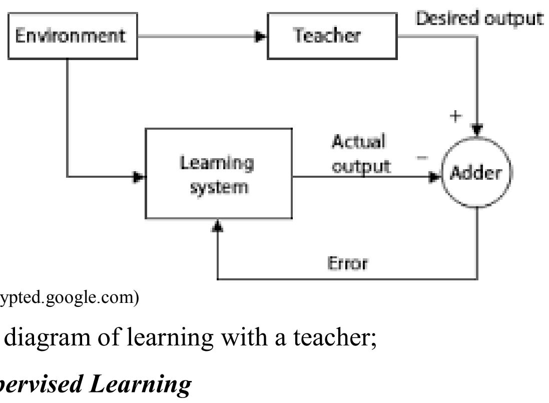
Usually, the complete set containing all available samples of the system behaviour is divided into two subsets, which are called training subset and test subset. The training subset, composed of 60-90 % of random samples from the complete set, will be used essentially in the learning process. On the other hand, the test subset, which is composed of 10-40 % from the complete sample set, will be used to verify if the network capabilities of generalizing solutions are within acceptable levels, thus allowing the validation of a given topology. Nonetheless, when dimensioning these subsets, statistical features of the data must also be considered. During the training process of artificial neural networks, each complete presentation of all the samples belonging to the training set, in order to adjust the synaptic weights and thresholds, will be called training epoch. Updating network architecture and connection weights so that network can efficiently perform a task (Margarita Sordo,2002)

Supervised Learning

We can call Learning with a teacher Figure shows a block diagram that illustrates this form of learning. (Simon Haykin,2009) which incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. (Oludele Awodele and Olawale Jegede,2009) methods require a dataset which contains “correct answers”.(TEGAN MAHARAJ, 2015)

The supervised learning strategy consists of having available the desired outputs for a given set of input signals; in other words, each training sample is composed of the input signals and their corresponding outputs. Henceforth, it requires a table with input/output data, also called attribute/value table, which represents the process and its behaviour. It is from this information that the neural structures will formulate “hypothesis” about the system being learned.

In this case, the application of supervised learning only depends on the availability of that attribute/value table, and it behaves as if a “coach” is teaching the network what is the correct response for each sample presented for its input. The synaptic weights and thresholds of the network are continually adjusted through the application of comparative actions, executed by the learning algorithm itself, that supervise the discrepancy between the produced outputs with respect to the desired outputs, using this difference on the adjustment procedure. The network is considered “trained” when this discrepancy is within an acceptable value range, taking into account the purposes of generalizing solutions. In fact, the supervised learning is a typical case of pure inductive inference, where the free variables of the network are adjusted by knowing a priori the desired outputs for the investigated system. Donald Hebb proposed the first supervised learning strategy in 1949, inspired by neuro physiological observations.



(encrypted.google.com)

Block diagram of learning with a teacher;

Unsupervised Learning

In unsupervised, or self-organized, learning, there is no external teacher or critic to oversee the learning process,(Simon Haykin,2009) in the case of unsupervised learning, no dependent variable is provided, and the resulting map of cases is based on the intrinsic similarities of the input data The most common unsupervised networks are Kohonen networks (Self Organizing Maps).(Maciej Szaleniec,2012) uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their emergent collective properties.(Oludele Awodele and Olawale Jegede, 2009)

Different from supervised learning, the application of an algorithm based on unsupervised learning does not require any knowledge of the respective desired outputs. Thus, the network needs to organize itself when there are existing particularities between the elements that compose the entire sample set, identifying subsets (or clusters) presenting similarities. The learning algorithm adjusts the synaptic weights and thresholds of the network in order to reflect these clusters within the network itself.

Alternatively, the network designer can specify (a priori) the maximum quantity of these possible clusters, using his/her knowledge about the problem.

Reinforcement Learning

There may or may not be a period of training on known examples at the beginning, and then new examples keep coming in, and the algorithm changes with the new information.(TEGAN MAHARAJ,2015)

Methods based on reinforcement learning are considered a variation of supervised learning techniques, since they continuously analyze the difference between the response produced by the network and the corresponding desired output The learning algorithms used on reinforcement learning adjusts the internal neural parameters relying on any qualitative or quantitative information acquired through the interaction with the system (environment) being mapped, using this information to evaluate the learning performance. The network learning process is usually done by trial and error because the only available response for a given input is whether it was satisfactory or unsatisfactory.

If satisfactory, the synaptic weights and thresholds are gradually incremented to reinforce (reward) this behavioural condition involved with the system. Several learning algorithms used by reinforcement learning are based on stochastic methods that probabilistically select the adjustment actions, considering finite set of possible solutions that can be rewarded if they have chances of generating satisfactory results. During the training process, the probabilities associated with action adjustment are modified to enhance the network performance This adjustment strategy has some similarities to some dynamic programming Techniques.

Offline Learning

In offline learning, also named batch learning, the adjustments on the weight vectors and thresholds of the network are performed after all the training set is presented, since each adjustment step takes into account the number of errors observed within the training samples with respect to the desired values for their outputs. Therefore, networks using offline learning requires, at least, one training epoch for executing one adjustment step on their weights and thresholds. Hence, all training samples must be available during the whole learning process.

Online Learning

Opposite to offline learning, in online learning, the adjustments on the weights and thresholds of the network are performed after presenting each training sample. Thus, after executing the adjustment step, the respective sample can be discarded. Online learning with this configuration is usually used when the behaviour of the system being mapped changes rapidly, thus the adoption of offline learning is almost impractical because the samples used at a given moment may no more represent the system behaviour in posterior moments. However, since patterns are presented one at a time, weight and threshold adjustment actions are well located and punctual, and they reflect a given behavioural circumstance of the system. Therefore, the network will begin to provide accurate responses after presenting a significant number of samples.

Activation Function

The activation function is used to calculate the output response of neuron. The sum of the weighted input signal is applied with an activation to obtain the response. Same activation functions are used for neurons in the same layer. The non-linear activation functions are used in a multilayer network. The output of a neuron in a neural network is between certain values (usually 0 and 1, or -1 and 1). In general, there are three types of activation functions they are the threshold function or step function, the pure linear function and sigmoid function.(52)

The back-propagation algorithm

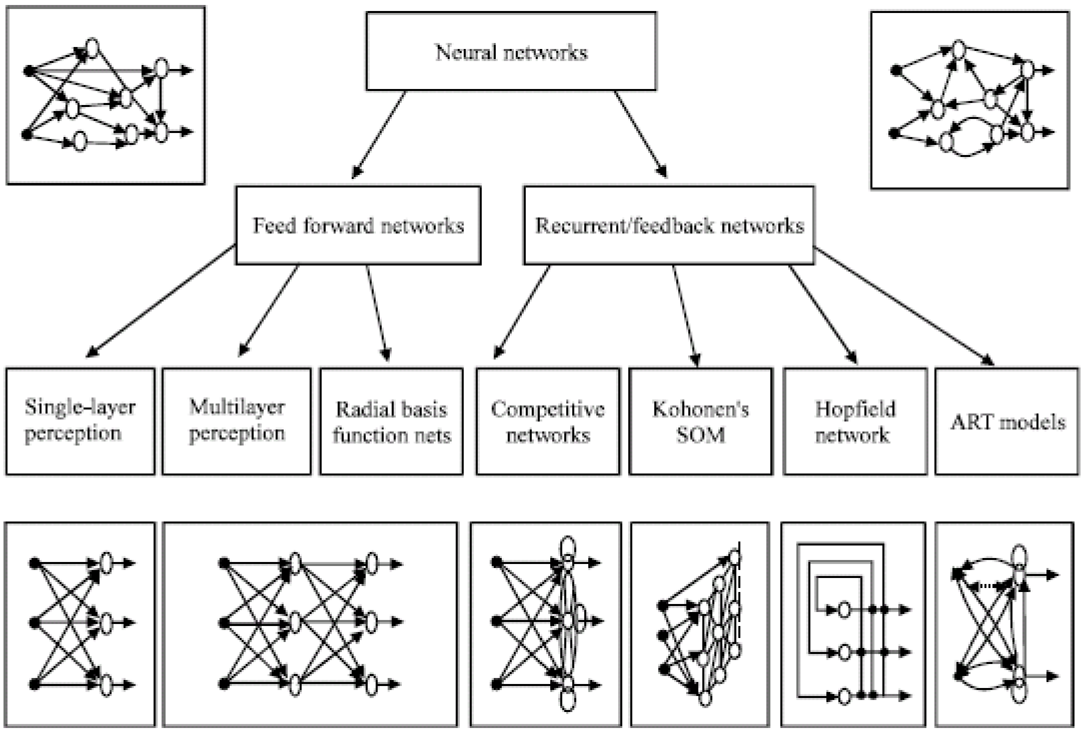
The back propagation algorithm is used in layered feed-forward ANNs. This means that the artificial neurons are organized in layers, and send their signals "forward", and then the errors are propagated backwards.بعد المبيه رقم)3(The training of ANNs by back propagation involves three stages (www.spring.com)or 4: (i) the feedforward of the input training pattern, (ii) the calculation and back propagation of the associated error, and (iii) the adjustment of the weights.(Z. Zhang*, K. Friedrich,2003)

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (**EW**). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the **EW**. (Oludele Awodele and Olawale Jegede,2009) learning algorithm adjusts connection weights until the system converges on a function that correctly reproduces the output.(15) The network "learns" from a series of "examples" that form the "training database".(18)(54)

A taxonomy of feed-forward and recurrent feedback network architectures.

What Can Neural Networks Be Used for?

Practically every non-mechanical task performed by animals requires the interaction of neural networks. Perception, recognition, memory, conscious thought, dreams, sensorimotor control-the list goes on.



(encrypted.google.com)

The desire to simulate some of these tasks has motivated the development of artificial neural networks. In this section, we present the reasons for studying neural networks from the viewpoint of the computational tasks for which they can be used. For each task, we identify performance measures that can be used to judge the degree of success of a neural network in performing the task. At a high level, the tasks performed using neural networks can be classified as those requiring *supervised* or *unsupervised* learning. In supervised learning, a teacher is available to indicate whether a system is performing correctly, or to indicate a desired response, or to validate the acceptability of a system's responses, or to indicate the amount of error in system performance. This is in contrast with unsupervised learning, where no teacher is available and learning must rely on guidance obtained heuristically by the system examining different sample data or the environment. A concrete example of supervised learning is provided by "classification" problems, whereas "clustering" provides an example of unsupervised learning. The distinction between supervised and unsupervised learning is illustrated in the following examples

EXAMPLE 1.3 (a) An archaeologist discovers a human skeleton and has to determine whether it belonged to a man or woman. In doing this, the archaeologist is guided by many past examples of male and female skeletons. Examination of these past examples (called the training set) allows the archaeologist to learn about the distinctions between male and female skeletons. This learning process is an example of supervised learning, and the result of the learning process can be applied to determine whether the newly discovered skeleton belongs to a man (b) In a different situation, the archaeologist has to determine whether a set of skeleton fragments belong to the same dinosaur species or need to be differentiated into different species. For this task, no previous data may be available to clearly identify the species for each skeleton fragment. The archaeologist has to determine whether the skeletons (that can be reconstructed from the fragments) are sufficiently similar to belong to the same species, or if the differences between these skeletons are large enough to warrant grouping them into different species. This is an unsupervised learning process, which involves estimating the magnitudes of differences between the skeletons. One archaeologist may believe the skeletons belong to different species, while another may disagree, and there is no absolute criterion to determine who is correct.

EXAMPLE 1.4 Consider the table in appendix B.l containing Fisher's iris data. This data consists of four measurements: the lengths and widths of sepals and petals of iris flowers. Class membership of each data vector is indicated in the fifth column of this table. This information is used in supervised learning. But if we remove the fifth column of the data, all we have is a set of 150 vectors (of widths and lengths of petals and sepals of iris flowers). To separate all 150 vectors into different groups of iris flowers, we would use procedures that depend only on the four values in each vector, and the relative proximity of different vectors. Such training is unsupervised because no *a priori* information is used regarding class membership, i.e., there is no teacher.

Classification: Classification, the assignment of each object to a specific "class" (one of many predetermined groups), is of fundamental importance in a number of areas ranging from image and speech recognition to the social sciences. We are provided with a "training set" consisting of sample patterns that are representative of all classes, along with class membership information for each pattern. Using the training set, we deduce rules for membership in each class and create a classifier, which can then be used to assign other patterns to their respective classes according to these rules. Neural networks have been used to classify samples, i.e., map input patterns to different classes. For instance, each output node can stand for one class. An input pattern is determined to belong to class / if the ith output node computes a higher value than all other output nodes when that input pattern is fed into the network. In some networks, an additional constraint is that the magnitude of that output node must exceed a minimal threshold, say 0.5.

For two-class problems, feed forward networks with a single output node are adequate.

If the node output has permissible values ranging from 0 to 1, for instance, a value close to 1 (say > 0.9) is considered to indicate one class, while a value close to 0 (say < 0.1) indicates the other class. Neural networks have been used successfully in a large number of practical classification tasks, such as the following.

1. Recognizing printed or handwritten characters
2. Classifying loan applications into credit-worthy and non-credit-worthy groups
3. Analyzing sonar and radar data to determine the nature of the source of a signal.

Clustering

Clustering requires grouping together objects that are similar to each other. In classification problems, the identification of classes is known beforehand, as is the membership of some samples in these classes. In clustering problems, on the other hand, all that is available is a set of samples and distance relationships that can be derived from the sample descriptions. For example, flowers may be clustered using features such as color and number of petals.

Most clustering mechanisms are based on some distance measure. Each object is represented by an ordered set (vector) of features. "Similar" objects are those that have nearly the same values for different features. Thus, one would like to group samples so as to minimize intra-cluster distances while maximizing inter-cluster distances, subject to constraints on the number of clusters that can be formed. One way to measure intra-cluster distance would be to find the average distance of different samples in a cluster from the cluster center. Similarly, inter-cluster distance could be measured using the distance between the centers of different clusters. Figure 1.17 depicts a problem in which prior clustering (into five clusters) is helpful in a classification pattern. The number of clusters depends on the problem, but should be as small as possible. Figure 1.18 shows three ways of clustering the same data, of which the first is preferable since it has neither too many nor too few clusters.
Some neural networks accomplish clustering by the following method. Initially, each node reacts randomly to the presentation of input samples. Nodes with higher outputs to an input sample learn to react even more strongly to that sample and to other input samples geographically near that sample. In this way, different nodes *specialize,* responding strongly to different clusters of input samples. This method is analogous to the statistical approach of *k-nearest neighbor* clustering, in which each sample is placed in the same cluster as the majority of its immediate neighbors.

Vector quantization: Neural networks have been used for compressing voluminous input data into a small number of weight vectors associated with nodes in the networks. Each input sample is associated with the nearest weight vector (with the smallest Euclidean distance). *Vector quantization* is the process of dividing up space into several connected regions (called "Voronoi regions"), a task similar to clustering. Each region is represented using a single vector (called a "codebook vector"). Every point in the input space belongs to one of these regions, and is mapped to the corresponding (nearest) codebook vector. The set of codebook vectors is a compressed form of the set of input data vectors, since many different input data vectors may be mapped to the same codebook vector. Figure 1.19 gives an example of such a division of two-dimensional space into Voronoi regions, called a Voronoi diagram (or "tessellation"). For two-dimensional input spaces, the boundaries of Voronoi regions are obtained by sketching the perpendicular bisectors of the lines joining neighboring codebook vectors.

Pattern association: In pattern association, another important task that can be performed by neural networks, the presentation of an input sample should trigger the generation of a specific output pattern. In *auto-association* or *associative memory tasks* (see figure 1.20), the input sample is presumed to be a corrupted, noisy, or partial version of the desired output pattern. In *hetew-association* (see figure 1.21), the output pattern may be any arbitrary pattern that is to be associated with a set of input patterns. An example of an auto-associative task is the generation of a complete (uncorrupted) image, such as a face, from a corrupted version. An example of hetero-association is the generation of a name when the image of a face is presented as input.

Function approximation

Many computational models can be described as functions mapping some numerical input vectors to numerical outputs. The outputs corresponding to some input vectors may be known from training data, but we may not know the mathematical function describing the actual process that generates the outputs from the input vectors. *Function approximation* is the task of learning or constructing a function that generates approximately the same outputs from input vectors as the process being modeled, based on available training data. Figure 1.22 illustrates that the same finite set of samples can be used to obtain many different functions, all of which perform reasonably well on the given set of points. Since ...... infinitely many functions exist that coincide for a finite set of points, additional criteria are necessary to decide which of these functions are *desirable.* Continuity and smoothness of the function are almost always required. Following established scientific practice, an important criterion is that of simplicity of the model, i.e., the neural network should have as few parameters as possible. These criteria sometimes oppose the *performance* criterion of minimizing error, as shown in figure 1.23. This set of samples contains one *outlier* whose behavior deviates significantly from other samples. Function *fo* passes through all the points in the graph and thus performs best; but *f\,* which misses the outlier, is a much simpler function and is preferable. The same is true in the example in figure 1.22, where the straight line (/0 performs reasonably well, although *fa* and *fa* perform best in that they have zero error. Among the latter, *fa* is certainly desirable because it is smoother and can be represented by a network with fewer parameters. Implicit in such comparisons is the assumption that the given samples themselves might contain some errors due to the method used in obtaining them, or due to environmental factors Function approximation can be performed using the networks described in chapters 3 and 4. Many industrial or manufacturing problems involve stabilizing the behavior of an object, or tracking the behavior of a moving object. These can also be viewed as function approximation problems in which the desired function is the time-varying behavior of the object in question.

Forecasting

There are many real-life problems in which future events must be predicted on the basis of past history. An example task is that of predicting the behavior of stock market indices. Weigend and Huberman (1990) observe that prediction hinges on two types of knowledge: knowledge of underlying laws, a very powerful and accurate means of prediction, and the discovery of strong empirical regularities in observations of a given system. However, laws underlying the behavior of a system are not easily discovered, and empirical regularities or periodicities are not always evident, and can often be masked by noise. Though perfect prediction is hardly ever possible, neural networks can be used to obtain reasonably good predictions in a number of cases. For instance, neural nets have succeeded in learning the 11-year cycle in sunspot data (cf. figure 1.24) without being told *a priori* about the existence of such a cycle [see Li *et al*. (1990); Weigend, Huberman, and Rumelhart(1990)].

Control applications

Many manufacturing and industrial applications have complex implicit relationships among inputs and outputs. *Control* addresses the task of determining the values for input variables in order to achieve desired values for output variables. This is also a function approximation problem, for which feedforward, recurrent, and some specialized neural networks have been used successfully. Adaptive control techniques have been developed for systems subject to large variations in parameter values, environmental conditions, and signal inputs. Neural networks can be employed in adaptive control systems to provide fast response, without requiring human intervention. Systems modeled in control problems may be *static* or *dynamic;* in the latter, the system may map inputs to outputs in a time-dependent manner, and the system's input-output mapping may change with time. A simple example of a static control system is one that maps input voltages into mechanical displacements of a robotic arm. Irrespective of the history of the system, an input voltage will always generate the same output displacement. By contrast, the inverted pendulum control task, where the behavior of a pendulum depends on time-dependent input variables such as velocity, is a dynamic control task. Neural networks have been used for two tasks associated with control; in both tasks, learning is supervised because the system's behavior dictates what the neural network is to accomplish. These tasks, illustrated in figure 1.25, are as follows.

System (forward) identification is the task of approximating the behavior of a system using a neural network or other learning method. If the system maps a set of input variables / to output variables 0, then forward identification is conducted by a feed forward neural network whose inputs correspond to / and outputs correspond to 0, trained to minimize an error measure that captures the difference between the network outputs and actual control system outputs.

Inverse identification is the task of learning the inverse of the behavior of a system, possibly using neural networks. For instance, given the amount of force applied to a robotic arm system, the system's behavior results in displacement by a certain amount.

The inverse problem in this case consists of determining the force required to produce a desired amount of displacement. This can be conducted using a feed forward network that uses components of *O* as its inputs and components of / as its outputs, using an error measure that captures the difference in actual system inputs (i) from the result *(N(S(i)))* of applying the actual system on those inputs, followed by the neural network. If the neural network has been successfully trained to perform inverse system identification, it can generate values for system inputs needed to obtain desired system outputs. If the system behavior changes with time, the same network that is used to generate in puts for the system may also be continually trained on-line, i.e., its weights are adjusted depending on the error measure *N(i) - N(S(N(i))),* the deviation between the network outputs *N(i)* and the result of applying the network to the system's output when applied to the network's outputs. If the system's input-output mapping is not completely known, or varies with time, a neural network may have to be continually trained to track changes in system behavior. A *feedback* control system is then appropriate, in which the error between actual and desired system output is used to modify the input signal, in order to produce the desired behavior. Feedforward neural networks (discussed in chapter 3) have been applied to many control problems, such as pole-balancing [Tolat and Widrow (1988)], robot arm control [Guez, Eilbert, and Kam (1988)], truck backing-up [Nguyen and Widrow (1990)], and inverse robot kinematics [Josin, Charney, and White (1988)]. These systems have been successful due to the following features.

1. Realization of fast decision making and control by parallel computation
2. Ability to adapt to a large number of parameters
3. Natural fault tolerance due to the distributed representation of information
4. Robustness to variations in parameters not modeled, due to the generalization properties of networks

Optimization: Many problems in business and scientific modeling can be represented as optimization problems in which the goal is to optimize (maximize or minimize) some functions, subject to some constraints. An example is the task of arranging components on a circuit board such that the total length of wires is minimized, with additional constraints that require certain components to be connected to certain others, as shown in figure 1.26. Some such problems can also be solved using neural networks. The best solutions for complex optimization problems are often obtained using networks with stochastic behavior in which network behavior has a significant random or probabilistic component. Chapter 7 discusses neural networks and stochastic algorithms for optimization

Search: Search problems and their solutions occupy a central place in Artificial Intelligence. Search problems are characterized by a set of states, a description of how transitions (or moves) may be made among different states, and methods of making moves that make it possible to reach a goal, possibly minimizing the number of moves or computational expense. Many problems discussed in preceding subsections (and solved using neural networks) can be viewed as search problems: for instance, each "state" in a neural network is a possible weight matrix, each "move" is the possible change in the weight matrix that may be made by the learning algorithm, and the "goal" state is one for which the mean squared error is at a local minimum.

Neural networks can also be applied to certain other search problems often solved using symbolic reasoning systems. In attempting to solve a search problem using neural networks, the first (and possibly the hardest) task is to obtain a suitable representation for the search space in terms of nodes and weights in a network. For instance, in a neural network to be used for game playing, the inputs to the network describe the current state of the board game, and the desired output pattern identifies the best possible move to be made. The weights in the network can be trained based on an evaluation of the quality of previous moves made by the network in response to various input patterns.

Neural networks have been successfully applied to broad spectrum of data-intensive applications. The list below is based on real-world success stories. It will give you an overview of the scope of problems that NeuroIntelligence can address.

Financial

Stock Market Prediction
Credit Worthiness
Credit Rating
Bankruptcy Prediction
Property Appraisal
Fraud Detection
Price Forecasts
Economic Indicator Forecasts

Medical

Medical Diagnosis
Detection and Evaluation of Medical Phenomena
Patient's Length of Stay Forecasts
Treatment Cost Estimation

Sales and Marketing

Sales Forecasting
Targeted Marketing
Service Usage Forecasting
Retail Margins Forecasting

Industrial

Process Control
Quality Control
Temperature and Force Prediction

Operational Analysis

Retail Inventories Optimization
Scheduling Optimization
Managerial Decision Making
Cash Flow Forecasting

Science

Pattern Recognition
Recipes and Chemical Formulation Optimization
Chemical Compound Identification
Physical System Modelling
Ecosystem Evaluation
Polymer Identification
Recognizing Genes
Botanical Classification
Signal Processing: Neural Filtering
Biological Systems Analysis
Ground Level Ozone Prognosis
Odor Analysis and Identification

HR Management

Employee Selection and Hiring
Employee Retention
Staff Scheduling
Personnel Profiling

Energy

Electrical Load Forecasting
Energy Demand Forecasting
Short and Long-Term Load Estimation
Predicting Gas/Coal Index Prices
Power Control Systems
Hydro Dam Monitoring

Educational

Teaching Neural Networks
Neural Network Research
College Application Screening
Predict Student Performance

Other

Sports Betting
Making Horse and Dog Racing Picks
Quantitative Weather Forecasting
Games Development
Optimization Problems, Routing
Agricultural Production Estimates
(http://www.alyuda.com)

Network Models

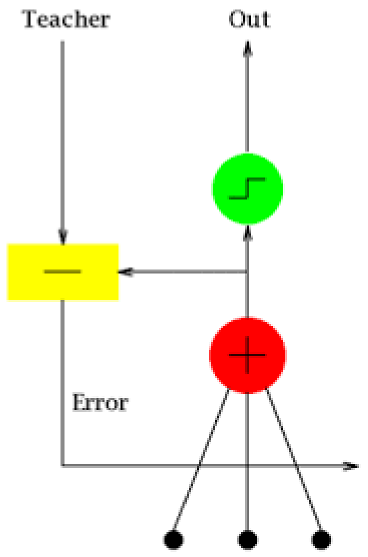
Generally, NNWs can be classified as feedforward and feedback or recurrent types. In the feedforward class, the signals flow only in the forward direction (see Figs. 3 and 4), whereas in recurrent neural network (RNN), the signals can flow in forward as well as backward or lateral direction (see Fig. 5). A network can be defined as static or dynamic, depending on whether it emulates static or dynamic system. A NNW is characterized by input-output mapping property. For static mapping, the feedforward NNWs are important, whereas for dynamic or temporal mapping, the RNNs are important [1]. The detailed description of all the NNW topologies is beyond the scope of this paper. Only a fewtopologies that are most relevant for power electronic systems will be discussed. Currently, around 90% of NNW applications use feedforward architecture, particularly the backpropagation network is most popular. (Bimal K. Bose*, 2007*)

Addaline

ADALINE (Adaptive Lineardi Neuron or later Adaptive Linear Element) is an early single-layer artificial neural network and the name of the physical device that implemented this network. The network uses memistors. It was developed by Professor Bernard Widrow and his graduate student Ted Hoff at Stanford University in 1960. It is based on the McCulloch-Pitts neuron. It consists of a weight, a bias and a summation function.

The difference between Adaline and the standard (McCulloch-Pitts) perceptron is that in the learning phase, the weights are adjusted according to the weighted sum of the inputs (the net). In the standard perceptron, the net is passed to the activation (transfer) function and the function's output is used for adjusting the weights.

There also exists an extension known as Madaline



Definition

Adaline is a single layer neural network with multiple nodes where each node accepts multiple inputs and generates one output.

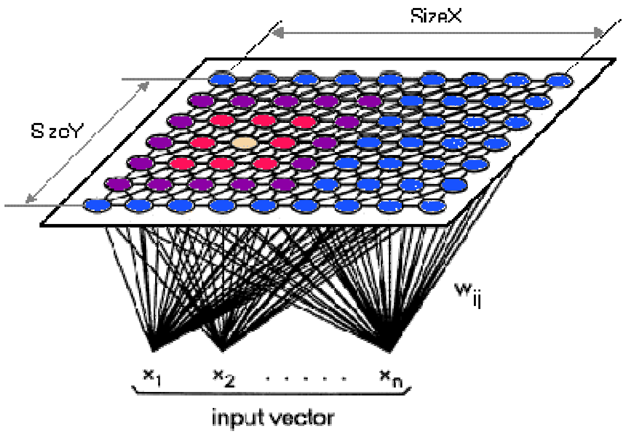
Madaline

MADALINE (Many ADALINE is a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecturefor classification that uses ADALINE units in its hidden and output layers, i.e. its activation function is the sign function.[2] The three-layer network uses memistors. Three different training algorithms for MADALINE networks, which cannot be learned using backpropagation because the sign function is not differentiable, have been suggested, called Rule I, Rule II and Rule III. The first of these dates back to 1962 and cannot adapt the weights of the hidden-output connection.] The second training algorithm improved on Rule I and was described in 1988.[1] The third "Rule" applied to a modified network with sigmoid activations instead of signum; it was later found to be equivalent to backpropagation. The Rule II training algorithm is based on a principle called "minimal disturbance". It proceeds by looping over training examples, then for each example, it:

- finds the hidden layer unit (ADALINE classifier) with the lowest confidence in its prediction,
- tentatively flips the sign of the unit,
- accepts or rejects the change based on whether the network's error is reduced,
- stops when the error is zero.(en.wikipedia.org)

Kohonen's Self-Organizing Maps(Som)

T he self-organizing algorithm of Kohonen is well known for its ability to map an input space with h a neural network. According to multiple observations, self organization seems to be an essential feature of the brain.(Pierre D emartines Francois Blayo,1992) Kohonen Self-Organizing Maps (or just Self-Organizing Maps, or SOMs for short), are a type of neural network. They were developed in 1982 by Tuevo Kohonen, a professor emeritus of the Academy of Finland [Wiki-01]. Self-Organizing Maps are aptly named. “Self-Organizing” is because no supervision is required. SOMs learn on their own through unsupervised competitive learning. “Maps” is because they attempt to *map* their weights to conform to the given input data. The nodes in a SOM network attempt to become like the inputs presented to them. In this sense, this is how they learn. They can also be called “Feature Maps”, as in Self-Organizing Feature Maps. Retaining principle 'features' of the input data is a fundamental principle of SOMs, and one of the things that makes them so valuable. Specifically, the topological relationships between input data are preserved when mapped to a SOM network. This has a pragmatic value of representing complex data.(Shyam M. Guthikonda,2005) SOM creates topologically ordered mappings between input data and processing elements of the map (Markus Törmä) (John A. Bullinaria, 2004)We shall concentrate on the particular kind of SOM known as a ***Kohonen Network***. This SOM has a feed-forward structure with a single computational layer arranged in rows and columns. Each neuron is fully connected to all the source nodes in the input layer:



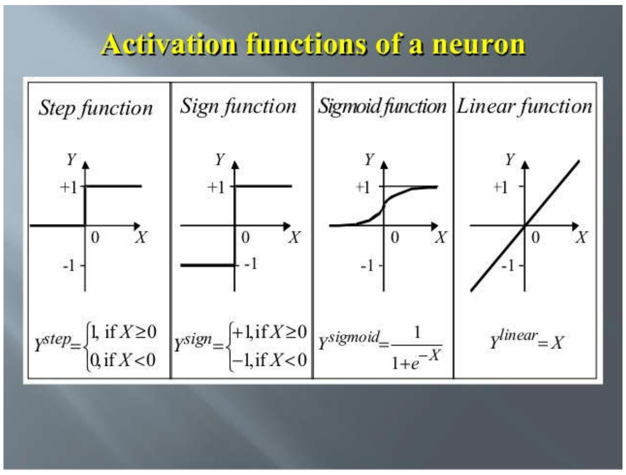
Clearly, a one dimensional map will just have a single row (or a single column) in the computational layer. (encrypted.google.com)

The behaviour of one neuron is independent of the behaviours of other neurons in its neighbourhood within the layer. We are now going to take into consideration that physical arrangement (the topology) of these nodes. Nodes that are ”close” together are going to interact differently than nodes that are ”far” apart.

In the brain, neurons tend to cluster in groups. The interactions of the neurons within the group are much greater than those outside the group. The Kohonen self-organizing maps are neural networks that try to mimic this feature in a simple way. (K. Ming Leung, 2017) Kohonen's self-organizing map (SOM) is an abstract mathematical model of topographic mapping from the (visual) sensors to the cerebral cortex. Modelling and analyzing the mapping are important to understanding how the brain perceives, encodes, recognizes and processes the patterns it receives and thus, if somewhat indirectly, are beneficial to machine-based pattern recognition.(Hujun Yin)

Like most artificial neural networks, the SOM has two modes of operation

1. At the beginning a map is built during the training process. Then the neural network organizes itself using the competitive process. A large number of input vectors must be given to the network. Preferably as much vectors representing the kind of vectors expected during the second phase as possible. Otherwise, all input vectors ought to be administered several times.
2. Each new input vector should be quickly given a location on the map during the mapping process. Then the vector is automatically classified or categorized. There will be one single winning neuron. It is the neuron whose weight vector lies closest to the input vector. (This can be simply determined by calculating the Euclidean distance between input vectors and weight vector.) (Miroslav Pich)


Activation functions of a neuron
Step function
Sign function
Sigmoid function
Linear function
Y
+1
-1
0
X
$Y^{step} = \begin{cases} 1, \text{if } X \ge 0 \\ 0, \text{if } X < 0 \end{cases}$
$Y^{sign} = \begin{cases} +1, \text{if } X \ge 0 \\ -1, \text{if } X < 0 \end{cases}$
$Y^{sigmoid} = \frac{1}{1+e^{-X}}$
$Y^{linear} = X$

(encrypted.google.com)

Overall SOM Algorithm

Training

- Select output layer topology
- Train weights connecting inputs to outputs
- Topology is used, in conjunction with current mapping
- of inputs to outputs, to define which weights will be
- updated
- Distance measure using the topology is reduced over
- time; reduces the number of weights that get updated
- per iteration
- Learning rate is reduced over time
- Testing
- Use weights from training(kphpnen3)

Basic Idea

Three important processes in the formation of the map:

1. Competition: Each neuron computes value of a discriminant function. The neuron with the largest value wins the competition. This is reminiscent of *long-range inhibition* in the brain.
2. Cooperation: The winning neuron determines the spatial location of a topological neighborhood for cooperation of excited neurons. This corresponds to *short-range excitation.*
3. Synaptic Adaptation: Enable the *excited* neurons to increase their values of the discriminant function in relation to the input patterns.(Joschka Boedecker,2015)

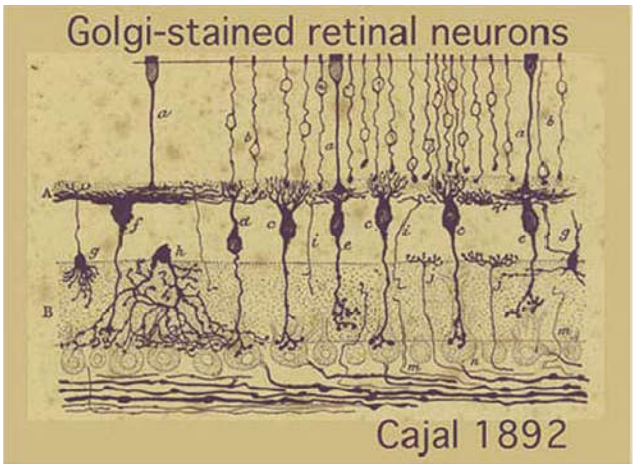
Competitive Process

Denote m-dimensional input pattern by: x = [x1, x2, . . .,xm]T
Synaptic weight vector of neuron *j*: wj = [wj1, wj2, . . .,wjm]T, j = 1, 2, . . ., l
Discriminant function is the inner product between neuron *j*'s weight vector and the input vector: $w^t_j x$
Determine index i(**x**) as: i(x) = argmin j x wj , j A

Biological Background: Lateral Inhibition and Hebbian Learning

Human visual perception and the brain make up the most complex cognition system and the most complex of all biological organs. Visual inputs contribute to over 90% of the total information (from all sensors) entering the brain. Nature and our living environment are constantly shaping our perception and cognition systems. Physiologically, human and indeed other animal visual (and other perception) systems have been evolved to so many different types of eyes and mammalian visual pathways for different purposes and conditions. For example, many insects have compound eyes, which have good temporal resolution and are more directionally sensitive and at the same time make them smaller and group them into a single structure - giving insects a better ability to detect spatial patterns and movement in order to escape from predators. Compound eyes have good time resolving power. Human eyes need 0.05 second to identify objects, while compound eyes need only 0.01 second. That is, they are good at recognizing (fast) moving objects. Eyes of large animals including humans have evolved to single-chambered 'camera lens' eyes, which have excellent angle resolution and are capable of seeing distant objects. Camera eyes have great space resolving power: high spatial resolution for good details of objects and patterns, and long depth resolution for both very near and very far objects. They also have brilliant sensitivities for light intensity - over 20 billion times (that is, 206 dB) range (the brightest to the darkest) - compared with most digital cameras, which are below 16-bit resolution (30 dB).

What information do eyes extract from the retina or sensory cells? Visual information is processed in both the retina and brain, but it is widely believed



A cross-sectional diagram of the retina, drawn by Santiago Ram´on y Cajal (1853-1934) and verified that most processing is done in the retina, such as extracting lines, angles, curves, contrasts, colours, and motion. The retina then encodes the information and sends through optic nerves and optic chiasma, where some left and right nerves are crossed, to the brain cortex in the left and/or right hemispheres. The retina is a complex neural network. Figure 1 shows a drawing of the cross section of the retina. The human retina has over 100 million photosensitive cells (combining rods and cones), processing in parallel the raw images, codes and renders to just over one million optic nerves, to be transmitted in turn to the brain cortex.(Miroslav Pich)

Structure of a SOM

The structure of a SOM is fairly simple, and is best understood with the use of an illustration such as Figure

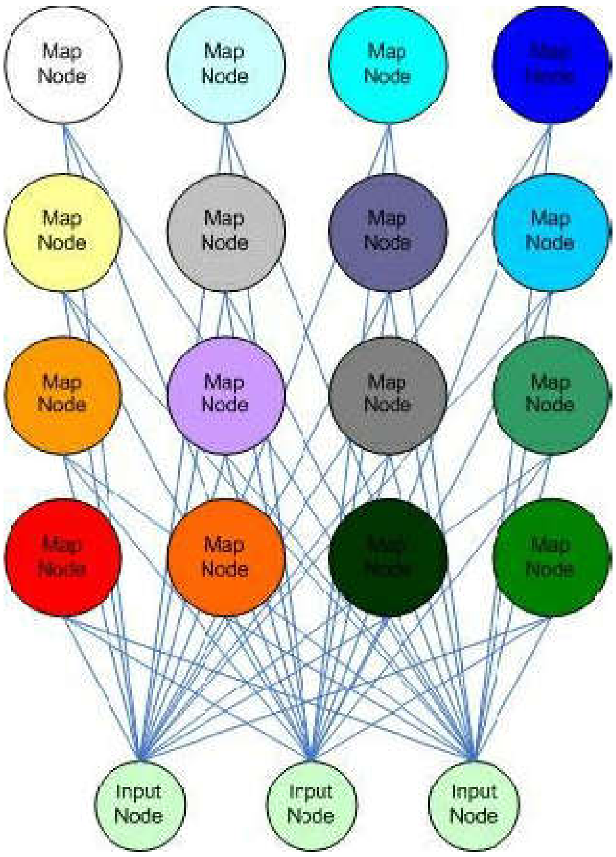


Figure 3 is a 4x4 SOM network (4 nodes down, 4 nodes across). It is easy to overlook this structure as being trivial, but there are a few key things to notice. First, each map node is connected to each input node. For this small 4x4 node network, that is 4x4x3=48 connections. Secondly, notice that *map nodes are not connected to each other*. The nodes are organized in this manner, as a 2-D grid makes it easy to visualize the results. This representation is also useful when the SOM algorithm is used. In this configuration, each map node has a unique (i,j) coordinate. This makes it easy to reference a node in the network, and to calculate the distances between nodes. Because of the connections only to the input nodes, the map nodes are oblivious as to what values their neighbors have. A map node will only update its' weights (explained next) based on what the input vector tells it.

The following relationships describe what a node essentially is:

1. *network⊂mapNode⊂ float weights* [*numWeights*]
2. *inputVectors⊂inputVector⊂ float weights*[*numWeights*]

1 says that the network (the 4x4 grid above) contains map nodes. A single map node contains an array of floats, or its' weights. numWeights will become more apparent during application discussion. The only other common item that a map node should contain is it's (i,j) position in the network. 2 says that the collection of input vectors (or input nodes) contains individual input vectors. Each input vector contains an array of floats, or its' weights. Note that numWeights is the same for both weight vectors. The weight vectors must be the same for map nodes and input vectors or the algorithm will not work.(Shyam M. Guthikonda,2015)

Adaptive Resonance Theory (ART)

What is (ART)

- ART stand for “adaptive resonance theory “ invented by Stephen grossberg in 1976.ART represents a family of neural network.
- ART encompasses a wide variety of neural network. The basic ART system is an supervised system.
- The term” resonance” refers to resonant state of neural network in which category prototype vector matches close enough to the current input vector. ART matching leads to this resonant state which permits learning .the network learns only in its resonant state.
- ART neural networks are capable of developing stable cluster of arbitrary sequence of input patterns by self organizing .
- ART-1 can cluster binary input vectors.
- ART-2 can cluster real-valued input vectors.

ART systems are well suited to problems that require online learning of large and evolving database.(Rc Chakraborty,2010) A cognitive and neural theory of how the brain can quickly learn and stably remember and recognize, objects, sounds, events, etc. from a stream of continuous stimuli.(Davide Bacciu)

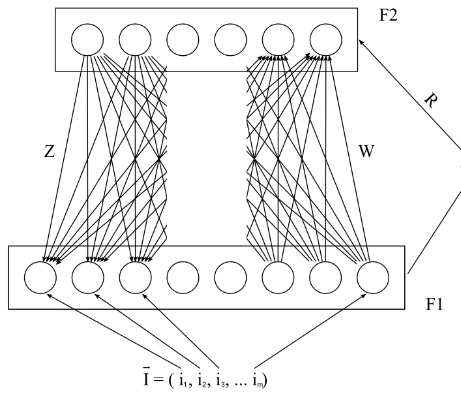
The primary intuition behind the ART model is that object identification and recognition generally occur as a result of the interaction of 'top-down' observer expectations with 'bottom-up' sensory information. The model postulates that 'top-down' expectations take the form of a memory template or prototype that is then compared with the actual features of an object as detected by the senses. This comparison gives rise to a measure of category belongingness. As long as this difference between sensation and expectation does not exceed a set threshold called the 'vigilance parameter', the sensed object will be considered a member of the expected class.(art 5) Essentially, ART (Adaptive Resonance Theory) models incorporate new data by checking for similarity between this new data and data already learned; “memory”. If there is a close enough match, the new data is learned.

Otherwise, this new data is stored as a “new memory”. (Michael Byrd,2008) An ART system consists of two subsystems, an attentional subsystem and an orienting subsystem. The stabilization of learning and activation occurs in the attentional subsystem by matching bottom-up input activation and top-down expectation. The orienting subsystem controls the attentional subsystem when a mismatch occurs in the attentional subsystem. In other words, the orienting subsystem works like a novelty detector. (T. Tanaka and A. Weitzenfeld,) In the second part of this research project it is tried to clarify how ART networks behave and why they are so complicated. In the treatment of this network a distinction is made between the ART network whieh handles binary valued input patterns (Rc Chakraborty,2010) and ART networks which handles analog valued inputs (Gail A. Carpenter and Stephen Grossberg,2002). Both networks have shown to be able to cluster a set of input patterns into a stable grouping in which output nodes are direct accessible. This is done by self organisation according to a clustering algorithm in which the neural network is critical to itself, which means that it will always "ask itself" whether a pattern and a prototype which are associated indeed fit in order to accept.(Freriks, L.W.; Cluitmans, P.J.M.; van Gils, M.J.,1992)

Learning

The basic ART system is an unsupervised learning model. It typically consists of a comparison field and a recognition field composed of neurons, a vigilance parameter (threshold of recognition), and a reset module. The comparison field takes an input vector (a one-dimensional array of values) and transfers it to its best match in the recognition field. Its best match is the single neuron whose set of weights (weight vector) most closely matches the input vector. Each recognition field neuron outputs a negative signal (proportional to that neuron’s quality of match to the input vector) to each of the other recognition field neurons and thus inhibits their output. In this way the recognition field exhibits lateral inhibition, allowing each neuron in it to represent a category to which input vectors are classified. Many ART applications use *fast learning*, whereby adaptive weights converge to equilibrium in response to each input pattern. Fast learning enables a system to adapt quickly to inputs that occur rarely but that may require immediate accurate recall. Remembering details of an exciting movie is a typical example of learning on one trial. Fast learning creates memories that depend upon the order of input presentation. Many ART applications exploit this feature to improve accuracy by voting across several trained networks, with voters providing a measure of confidence in each prediction.(Gail A. Carpenter and Stephen Grossberg,2002)

After the input vector is classified, the reset module compares the strength of the recognition match to the vigilance parameter. If the vigilance parameter is overcome (i.e. the input vector is within the normal range seen on previous input vectors), training commences: the weights of the winning recognition neuron are adjusted towards the features of the input vector. Otherwise, if the match level is below the vigilance parameter (i.e. the input vector's match is outside the normal expected range for that neuron) the winning recognition neuron is inhibited and a search procedure is carried out. In this search procedure, recognition neurons are disabled one by one by the reset function until the vigilance parameter is overcome by a recognition match. In particular, at each cycle of the search procedure the most active recognition neuron is selected and then switched off if its activation is below the vigilance parameter (note that it thus releases the remaining recognition neurons from its inhibition). If no committed recognition neuron’s match overcomes the vigilance parameter, then an uncommitted neuron is committed and its weights are adjusted towards matching the input vector. The vigilance parameter has considerable influence on the system: higher vigilance produces highly detailed memories (many, fine-grained categories), while lower vigilance results in more general memories (fewer, more-general categories).(en.wikipedia.org)



(encrypted.google.com)

Training

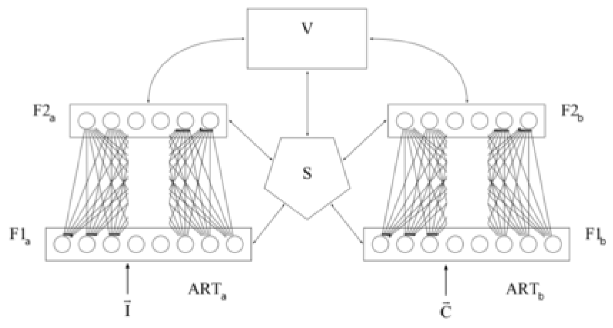
There are two basic methods of training ART-based neural networks: slow and fast. In the slow learning method, the degree of training of the recognition neuron’s weights towards the input vector is calculated to continuous values with differential equations and is thus dependent on the length of time the input vector is presented. With fast learning, algebraic equations are used to calculate degree of weight adjustments to be made, and binary values are used. While fast learning is effective and efficient for a variety of tasks, the slow learning method is more biologically plausible and can be used with continuous-time networks (i.e. when the input vector can vary continuously).(art 5)

Types of ART

The term "ART network" describes a general class of network system architectures rather than one particular instantiation of a system. What all ART network systems share in common is a set of functional principles [CARP3], all of which must be met by the organization of the network. (Richard B. Wells)

ART 1 is the simplest variety of ART networks, accepting only binary inputs. **ART 2** extends network capabilities to support continuous inputs. (art8) **ART 2-A** is a streamlined form of ART-2 with a drastically accelerated runtime, and with qualitative results being only rarely inferior to the full ART-2 implementation. **ART 3** builds on ART-2 by simulating rudimentary neurotransmitter regulation of synaptic activity by incorporating simulated sodium (Na+) and calcium (Ca2+) ion concentrations into the system's equations, which results in a more physiologically realistic means of partially inhibiting categories that trigger mismatch resets.

Fuzzy ART implements fuzzy logic into ART's pattern recognition, thus enhancing generalizability. An optional (and very useful) feature of fuzzy ART is complement coding, a means of incorporating the absence of features into pattern classifications, which goes a long way towards preventing inefficient and unnecessary category proliferation.



ARTMAP Overview

ARTMAP, also known as **Predictive ART**, combines two slightly modified ART-1 or ART-2 units into a supervised learning structure where the first unit takes the input data and the second unit takes the correct output data, then used to make the minimum possible adjustment of the vigilance parameter in the first unit in order to make the correct classification.

Fuzzy ARTMAP is merely ARTMAP using fuzzy ART units, resulting in a corresponding increase in efficacy.((encrypted.google.com)

Criticism

It has been noted that results of Fuzzy ART and ART 1 depend critically upon the order in which the training data are processed. The effect can be reduced to some extent by using a slower learning rate, but is present regardless ofthe size of the input data set. Hence Fuzzy ART and ART 1 estimates do not possess the statistical property of consistency.(ART 5)

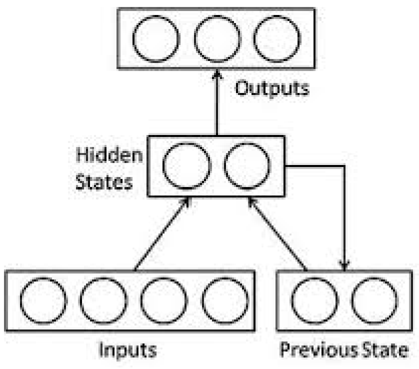
Real-Time Recurrent Network

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behaviour. Unlike feed forward, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition. (en.wikipedia.org)

Recurrent neural networks (RNNs) are a subclass of ANNs which allow units to form a cyclic graph. This allows the network to store an internal state and consequently process sequences of inputs and thus perform temporal tasks. Since RNNs accumulate previous activity in the activation of units that form directed cycles, there is no need to feed the network with history of previous inputs and outputs like in TDNN. Rather than approximating a function, RNNs try to model a process. Therefore, prediction of future process output can be described (Karol Kuna, 2015)

Simple recurrent network

A strict feed forward architecture does not maintain a short-term memory. Any memory effects are due to the way past inputs are re-presented to the network (as for the tapped delay line). A simple recurrent network (SRN has activation feedback which embodies short-term memory. A state layer is updated not only with the external input of the network but also with activation from the previous forward propagation. The feed- back is modified by a set of weights as to enable automatic adaptation through learning (e.g. back propagation) (Mikael Bod´en,*2001)*



(encrypted.google.com)

Simple recurrent network

Recurrent Neural Networks (RNN) are nonlinear or linear dynamic systems. They can be simulated in software on computers or implemented in hardware (analog or digital). A first property that can be used to distinguish RNN in two distinct groups is the representation of time in the system. We have:

- continuous-time systems
- discrete-time systems

A second property is the representation of signals in the system. The signals can be

- real-valued
- quantized

A system can be real-valued if implemented in analog hardware. All digital implementations use quantized values since values are stored in a finite number of bits. However, a digital implementation is often analyzed as if it were real-valued in case that the error introduced by the quantization is too small to be noticed (for example when floating point numbers are used).

The above properties of the system do not say much about the intended application of the RNN. All possible applications of RNN can be grouped into two broad categories. Recurrent neural networks can be used as

Associative memories

Sequence mapping systems

Recurrent neural networks used as sequence mapping system:

Recurrent neural networks used as sequence mapping systems are operated by supplying an input *sequence*, which consists of different input patterns at each time step (in case of a discrete time system), or a time-varying input pattern over time (in case of a continuous-time system). At each time instant, an output is generated which depends on previous activity of the system and on the current input pattern. The entire output *sequence* generated over time is considered the result of the computation. The class of sequence mapping systems is interesting for practical applications in sequence recognition, generation or prediction and it will be examined in the next chapter. Sequence mapping neural networks are nearly always implemented in software or clocked digital hardware (both have a discrete representation of time).

This report will focus on recurrent neural networks used as sequence mapping systems. Using the common ways of implementation these networks are discrete-time systems. Therefore, all treatment of recurrent neural networks in the next chapters will be restricted to discrete-time systems. Some examples of recurrent neural networks used as associative memories will be given now.

Recurrent neural network used as associative memories

Recurrent neural networks used as associative memories are operated by applying a fixed input pattern (that does not change over time). Then the network is operated according to a set of equations describing the network dynamics. Internal signals and the network outputs will change over time. Under certain conditions (and waiting for a sufficient time interval), the network .settles.. This means the system's output has converged to some *static pattern* which is considered the result of the computation performed by the system. This result is some association made by the system in response to the input, hence the name associative memories. The difference with sequence mapping systems lies in supplying a static input to the network (not a sequence) and only using the final output values of the network as a result (and not the output sequence over time). So both input and output are static patterns whereas for the sequence mapping systems, both input and output are sequences.

These recurrent neural network architectures were proposed to create associative content addressable memories. They were used in Artificial Intelligence research and they contributed to research about the way the (human) brain works. Associative memories are sometimes implemented in analog hardware, but generally for research purposes a software implementation is favoured because it is more convenient and flexible. Examples of these architectures are the Brain-State-in-a-Box neural network and the Hopfield network. The Hopfield network model was later on extended with neurons that operate in a stochastic manner (using theory from the field of statistical mechanics) which are called Boltzmann machines.

Applications of discrete-time recurrent neural networks

Sequence mapping recurrent neural networks can be used in a number of different ways. The most common uses or task types are:

1. Modeling the input-output behavior of a dynamic system: In this case the network is trained with known examples of input-output behavior of a .black-box. Dynamic system. The trained network is then used to predict dynamic system output, given an input sequence.
2. Time-Series prediction: In this case the network is trained with an output data sequence of some .black-box. Process. The trained network is then used to predict the future (unknown) process outputs, given a sequence of process outputs up to the current time.
3. Sequence classification: The network is trained with a data sequence that belongs to one class wi out of N possible classes. The target value for the network is a coded
4. Representation of the class wi. Then, the network can be used to assign a classification to a new data sequence.
5. *Feature extraction:* The network is trained with a sequence of raw data vectors as input and a smaller feature sequence as a target. The network can learn to transform the.low level. raw data vectors into .higher-level. features. These features can then be used as input to any subsequent classification procedure.
6. *Modeling a finite state machine:* The network is trained with example sequences of input and output of a .black box. Finite state machine. The trained network is used to identify the state machine or predict its output, given an input sequence.(Esko O. Dijk,1999)

Stability, Controllability and Observability

Since one can think about recurrent networks in terms of their properties as dynamical systems, it is natural to ask about their stability, controllability and observability:

Stability concerns the boundedness over time of the network outputs, and the response of the network outputs to small changes (e.g., to the network inputs or weights).

Controllability is concerned with whether it is possible to control the dynamic behaviour. A recurrent neural network is said to be controllable if an initial state is steerable to any desired state within a finite number of time steps.

Observability is concerned with whether it is possible to observe the results of the control applied. A recurrent network is said to be observable if the state of the network can be determined from a finite set of input/output measurements.
A rigorous treatment of these issues is way beyond the scope of this module!

Recurrent Neural Network Architectures

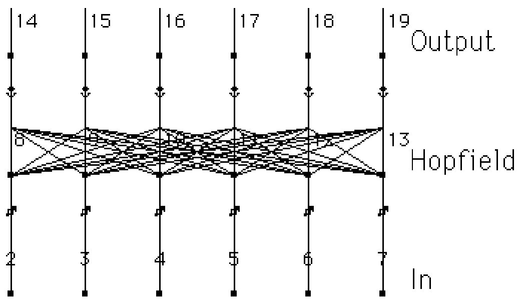
The fundamental feature of a *Recurrent Neural Network (RNN)* is that the network contains at least one *feed-back connection*, so the activations can flow round in a loop. That enables the networks to do *temporal processing* and *learn sequences*, e.g., perform sequence recognition/reproduction or temporal association/prediction. Recurrent neural network architectures can have many different forms. One common type consists of a standard Multi-Layer Perceptron (MLP) plus added loops. These can exploit the powerful non-linear mapping capabilities of the MLP, and also have some form of ***memory***. Others have more uniform structures, potentially with every neuron connected to all the others, and may also have stochastic activation functions.

For simple architectures and deterministic activation functions, learning can be achieved using similar gradient descent procedures to those leading to the back-propagation algorithm for feed-forward networks. When the activations are stochastic, simulated annealing approaches may be more appropriate. The following will look at a few of the most important types and features of recurrent networks.(John A. Bullinaria, 2015)

Hopfield Network

John Hopfield first presented his cross-bar associative network in 1982 at the National Academy of Sciences. In honor of Hopfield's success and his championing of neural networks in general, this network paradigm is usually referred to as a Hopfield Network. The network can be conceptualized in terms of its energy and the physics of dynamic systems. A processing element in the Hopfield layer, will change state only if the overall "energy" of the state space is reduced. In other words, the state of a processing element will vary depending whether the change will reduce the overall "frustration level" of the network. Primary applications for this sort of network have included associative, or content-addressable, memories and a whole set of optimization problems, such as the combinatoric best route for a traveling salesman.

The Figure 5.3.1 outlines a basic Hopfield network. The original network had each processing element operate in a binary format. This is where the elements compute the weighted sum of the inputs and quantize the output to a zero or one. These restrictions were later relaxed, in that the paradigm can use a sigmoid based transfer function for finer class distinction. Hopfield himself showed that the resulting network is equivalent to the original network designed in 1982.



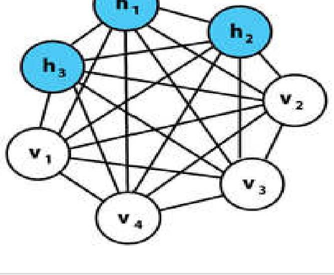
A Hopfield Network Example

The Hopfield network uses three layers; an input buffer, a Hopfield layer, and an output layer. Each layer has the same number of processing elements. The inputs of the Hopfield layer are connected to the outputs of the corresponding processing elements in the input buffer layer through variable connection weights. The outputs of the Hopfield layer are connected back to the inputs of every other processing element except itself. They are also connected to the corresponding elements in the output layer. In normal recall operation, the network applies the data from the input layer through the learned connection weights to the Hopfield layer. The Hopfield layer oscillates until some fixed number of cycles have been completed, and the current state of that layer is passed on to the output layer. This state matches a pattern already programmed into the network.

The learning of a Hopfield network requires that a training pattern be impressed on both the input and output layers simultaneously. The recursive nature of the Hopfield layer provides a means of adjusting all of the connection weights. The learning rule is the Hopfield Law, where connections are increased when both the input and output of an Hopfield element are the same and the connection weights are decreased if the output does not match the input. Obviously, any non-binary implementation of the network must have a threshold mechanism in the transfer function, or matching input-output pairs could be too rare to train the network properly. The Hopfield network has two major limitations when used as a content addressable memory. First, the number of patterns that can be stored and accurately recalled is severely limited. If too many patterns are stored, the network may converge to a novel spurious pattern different from all programmed patterns. Or, it may not converge at all. The storage capacity limit for the network is approximately fifteen percent of the number of processing elements in the Hopfield layer. The second limitation of the paradigm is that the Hopfield layer may become unstable if the common patterns it shares are too similar. Here an example pattern is considered unstable if it is applied at time zero and the network converges to some other pattern from the training set. This problem can be minimized by modifying the pattern set to be more orthogonal with each other. (Rome Laboratory, 1992)

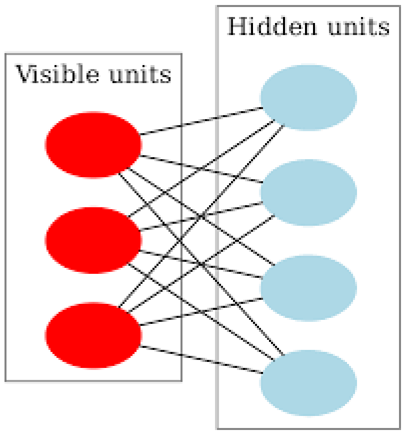
Boltzmann Machine

A Boltzmann machine is a type of stochastic recurrent neural network and Markov Random Field invented by Geoffrey Hinton and Terry Sejnowski in 1985. Boltzmann machines can be seen as the stochastic, generative counterpart of Hopfield nets. They were one of the first examples of a neural network capable of learning internal representations, and are able to represent and (given sufficient time) solve difficult combinatoric problems. They are theoretically intriguing because of the locality and Hebbian nature of their training algorithm, and because of their parallelism and the resemblance of their dynamics to simple physical processes. Due to a number of issues discussed below, Boltzmann machines with unconstrained connectivity have not proven useful for practical problems in machine learning or inference, but if the connectivity is properly constrained, the learning can be made efficient enough to be useful for practical problems. (en.wikipedia.org)



A graphical representation of an example Boltzmann machine. Each undirected edge represents dependency. In this example there are 3 hidden units and 4 visible units. This is not a restricted Boltzmann machine.

The stochastic neurons of the Boltzmann machine are partitioned into two functional groups, *visible* and *hidden*, as depicted in Fig. 11.5. The visible neurons8 provide an interface between the network and the environment in which it operates. During the training phase of the network, the visible neurons are all *clamped* onto specific states determined by the environment .The hidden neurons, on the other hand, always operate freely; they are used to explain underlying constraints contained in the environmental input vectors. The hidden neurons accomplish this task by capturing higher-order



Architectural graph of Botzmann machine statistical correlations in the clamping vectors. The network described here represents a special case of the Boltzmann machine. It may be viewed as an unsupervised-learning procedure for modeling a probability distribution that is specified by clamping patterns onto the visible neurons with appropriate probabilities. By so doing, the network can perform *pattern completion*. Specifically, when a partial information-bearing vector is clamped onto a subset of the visible neurons, the network performs completion on the remaining visible neurons, provided that it has learned the training distribution properly.

The primary goal of Boltzmann learning is to produce a neural network that correctly models input patterns according to the Boltzmann distribution. In applying this form of learning, two assumptions are made:

1. Each environmental input vector (pattern) persists long enough to permit the network to reach *thermal equilibrium*.
2. There is *no* structure in the sequential order in which the environmental vectors are clamped onto the visible units of the network.

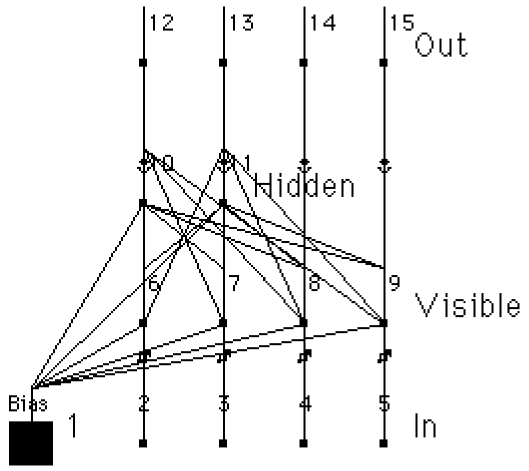
A particular set of synaptic weights is said to constitute a perfect model of the environmental structure if it leads to exactly the same probability distribution of the states of the visible units (when the network is running freely) as when these units are clamped by the environmental input vectors. In general, unless the number of hidden units is exponentially large compared with the number of visible units, it is impossible to achieve such a perfect model. If, however, the environment has a regular structure, and the network uses its hidden units to capture these regularities, it may achieve a good match to the environment with a manageable number of hidden units. (Simon Haykin, 2009)

Recirculation network

Recirculation networks were introduced by Geoffrey Hinton and James McClelland as a biologically plausible alternative to back-propagation networks. In a back propagation network, errors are passed backwards through the same connections that are used in the feed forward mechanism with an additional scaling by the derivative of the feed forward transfer function. This makes the back-propagation algorithm difficult to implement in electronic hardware.

In a recirculation network, data is processed in one direction only and learning is done using only local knowledge. In particular, the knowledge comes from the state of the processing element and the input value on the particular connection to be adapted. Recirculation networks use unsupervised learning so no desired output vector is required to be presented at the output layer. The network is auto-associative, where there are the same number of outputs as inputs.

This network has two layers between the input and output layers, called the visible and hidden layers. The purpose of the learning rule is to construct in the hidden layer an internal representation of the data presented at the visible layer. An important case of this is to compress the input data by using fewer processing elements in the hidden layer. In this case, the hidden representation can be considered a compressed version of the visible representation. The visible and hidden layers are fully connected to each other in both directions. Also, each element in both the hidden and visible layers are connected to a bias element. These connections have variable weights which learn in the same manner as the other variable weights in the network. (Rome Laboratory,1992)



An Example Recirculation Network.(encrypted.google.com)

The learning process for this network is similar to the bi-directional associative memory technique. Here, the input data is presented to the visible layer and passed on to the hidden layer. The hidden layer passes the incoming data back to the visible, which in turn passes the results back to the hidden layer and beyond to the output layer. It is the second pass through the hidden layer where learning occurs. In this manner the input data is recirculated through the network architecture. During training, the output of the hidden layer at the first pass is the encoded version of the input vector. The output of the visible layer on the next pass is the reconstruction of the original input vector from the encoded vector on the hidden layer. The aim of the learning is to reduce the error between the reconstructed vector and the input vector. This error is also reflected in the difference between the outputs of the hidden layer at the first and final passes since a good reconstruction will mean that the same values are passed to the hidden layer both times around. Learning seeks to reduce the reconstruction error at the hidden layer also. In most applications of the network, an input data signal is smoothed by compressing then reconstructing the input vector on the output layer. The network acts as a low bandpass filter whose transition point is controlled by the number of hidden nodes (Rome Laboratory, 1992)

Elman networks and Jordan networks

The following special case of the basic architecture above was employed by Jeff Elman. A three-layer network is used (arranged horizontally as x, y, and z in the illustration), with the addition of a set of "context units" (u in the illustration). There are connections from the middle (hidden) layer to these context units fixed with a weight of one.[9] At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that are beyond the power of a standard multilayer perceptron

Jordan networks, due to Michael I. Jordan, are similar to Elman networks. The context units are however fed from the output layer instead of the hidden layer. The context units in a Jordan network are also referred to as the state layer, and have a recurrent connection to themselves with no other nodes on this connection. (en.wikipedia.org)

References

John A. B. (2004) Introduction to Neural Networks.

Yadav, N .Yadav, A .Kumar, M Chapter 2, History of Neural Networks, http://www.springer.com/978-94-017-9815-0,springer.

Martin T. H, Howard B. D, Mark, H. Orlando De.J. Neural Network Design, second edition.

Kevin, G. (1997) An introduction to neural networks. London: UCL Press.

Oludele, A. and Olawale,J.(2009) Neural Networks and Its Application in Engineering, Proceedings of Informing Science & IT Education Conference (InSITE) Babcock University, Nigeria.

F, A. Ioannis, T. (2004) Feed-Forward Artificial Neural Networks, *Vanderbilt University,* MEDINFO.

F,C.christo, A.R, masri, E.M.nebot. (1996) An Integrated Pdf\ Neural Network Approach For Simulating Turbulent Reacting System, twenty-sixth symposium (international) on combustion\ the combustion institute, pp.43-48.

Rama Krishna,S. Prajneshu.(2008) Artificial Neural Network Methodology for Modelling and Forecasting Maize Crop Yield, Agricultural Economics Research Review, Vol. 21 January-June 2008 pp 5-10, New Delhi - 110 012.

Jocelyn,S . Robert J. F. D. (1991) Neural Networks That Generalize *Neural,¥etworks,* Vol. 4, pp. 67 79, 1991, orginal contribution.

Ajith. A,oArtificial Neural Networks, Oklahoma State University, Stillwater, OK, USA.

J. M. Ben´ıtez, J. L. C, I. Requena, (1997) Are Artificial Neural Networks Black Boxes? IEEE Transactions on Neural Networks, VOL. 8, NO. 5, SEPTEMBER 1997.

Xin, Y. yong, L. Towards designing artificial neural networks by evolution, the university of new south wales,Australian defence force academy, Canberra ACT Australian 2600 .

Kishan, M .chilukuri, M.(1997) Elements of Artificial Neural Nets, research Gate.

Bimal K. B, (2007) Neural Network Applications in Power Electronics and Motor Drives-An Introduction and Perspective, IEEE Transactions on Industrial Electronics, VOL. 54, NO. 1, FEBRUARY 2007.

Iebeling, k. Milton, B.(1996) desinging a neural network for forcasting and economic time series, Canada, neurocomputing 10 (1996)215-236,ELSEVIER.

J¨urgen, S. (2014) Deep Learning in Neural Networks: An Overview, Technical Report IDSIA-03-14 / arXiv: 1404.7828 v4 [cs.NE] (88 pages, 888 references, University of Lugano & SUPSI Galleria 2, 6928 Manno-Lugano, Switzerland.

Xin, Y. Yong, L. (1997) A New Evolutionary System for Evolving

Artificial Neural Networks, IEEE Transactions on Neural Networks, VOL. 8, NO. 3, MAY 1997.

Muriel, G. Ioannis, D. Sovan, L. (2003) Review and comparison of methods to study the contribution of variables in artificial neural network models, Ecological Modelling 160 (2003) 249_/264, Elsevier.

Cameron M. Z. Donald H. B. Slobodan P. S. (1999) Short term streamflow forecasting using artificial neural networks, Journal of Hydrology 214 (1999) 32-48, Elsevier.

Amir,A.(1991) learning algorithm for neural networks, California institute of technology, Pasadena, California.

Magali R. G. M. Paulo E. M. A. (2003) A Comprehensive Review for Industrial Applicability of Artificial Neural Networks, IEEE Transactions On Industrial Electronics, VOL. 50, NO. 3, JUNE 2003.

C.W. D. R.L. W. (2001) Hydrological modelling using artificial neural networks, Progress in Physical Geography 25,1 (2001) pp. 80-108.

Johann,G. Jure, Z. (1993) Neural Networks in Chemistry, *Angew. Chem. Int. Ed. Engl.* 1993, 32, 503-527.

Maciej, S. (2012) Prediction of enzyme activity with neural network models based on electronic and geometrical features of substrates, by Institute of Pharmacology Polish Academy of Sciences Pharmacological Reports 2012, 64, 761.781 ISSN 1734-1140.

Simon,H.(2009) Neural Networks and Learning Machines, Pearson Education, Inc: New Jersey.

Chao. Erihe. Artificial Neural Network and Fuzzy Logic in forecasting short-term Temperature, Telemark University College Faculty of Technology.

Krzysztof, P. (2008) Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes, Library of Congress Control Number: 2008926085, 2008 Springer-Verlag Berlin Heidelberg.

Z. Zhang, K. Friedrich. (2003) Artificial neural networks applied to polymer composites: a review, Composites Science and Technology 63 (2003) 2029-2044, composites science and technology, Elsevier.

Fakhri, k. Fundamentals of artificial intelligence, university of waterloo. Artificial neural networks.

MILOSLAV, V. (2014) An artificial neural network approach and sensitivity analysis in predicting skeletal muscle forces, *Acta of Bioengineering and Biomechanics, Vol. 16, No. 3, 2014,* Original paper, DOI: 10.5277/abb140314 .

Dave, A. George, M. (1992) Artificial Neural Networks Technology, Utica, New York 13502-4627, Kaman Sciences Corporation.

John, D. Eric, L. Jessica, Z. Anita, W. artificial neural networks.

A.D.Dongare. R.R.Kharde. Amit D.K. (2012) Introduction to Artificial Neural Network, International Journal of Engineering and Innovative Technology (IJEIT), Volume 2, Issue 1, July 2012, ISSN: 2277-3754 ISO 9001:2008 Certified.

Richard P. L. (1987) An Introduction' to Computing with Neural Nets, IEEE Assp Magazine.

Sovan, L. J.F. Gue´gan. (1990) Artificial neural networks as a tool in ecological modelling, an introduction, Ecological Modelling 120 (1999) 65-73, Ecological Modeling, Elsevier.

Jure, Z. (1994) Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them, Tarragona, Spain, Acta Chimica Slovenica 41/3/1994, pp. 327-352.

Dimitri P. B. Neuro-Dynamic Programming: An Overview Laboratory for Information and Decision Systems Massachusetts Institute of Technology Cambridge, MA 02139, USA.

D.C. Park, M.A. El-Sharkawi, R.J. Marks L.E. Atlas and M.J. Damborg. (1991) Electric Load Forecasting Using An Artificial Neural Network, University of Washington, "Electric load forecasting using an artificial neural network", IEEE Transactions on Power Engineering, vol.6, pp.442-449 (1991).

Guoqiang, Z. B. Eddy, P. Michael Y. Hu. (1998) Forecasting with artificial neural networks: The state of the art, *Graduate School of Management*, *Kent State University*, *Kent*, *Ohio* 44242-0001, *USA,* International Journal of Forecasting 14 (1998) 35-62,Else IER.

Simon, H. Neural Networks A Comprehensive Foundation, Second Edition, *McMaster University Hamilton, Ontario, Canada,* Prentice Hall.

C. W. Dawson. Robert, W (2001) Hydrological Modelling Using Artificial Neural Networks, Article in Progress in Physical Geography· March 2001, DOI: 10.1177/03091333010250 0104, research Gate.

Mohamed A.S, Mark B. J.Holger R. M. (2001) Artificial Neural Network Applications In Geotechnical Engineering, *Department of Civil and Environmental Engineering, Adelaide University,* Australian Geomechanics-March 2001.

B.S. Ahna,*, S.S. Chob, C.Y. Kim.(2000) The integrated methodology of rough set theory and artificial neural network for business failure prediction, Expert Systems with Applications 18 (2000) 65-74, Pergamon.

Adrian A. C .OFER, L. (2008) Annz: Estimating Photometric Redshifts Using Artificial Neural Networks, Institute of Astronomy, University of Cambridge, Cambridge CB3 0HA, UK, Draft Version February 2, 2008, Preprint typeset using Latex style emulateapj v. 2/10/04.

Iebeling, k.Milton B. (1996) Designing a neural network for forcasting financial and economic time series, Neurocomputing 10 (1996) 215-236, Elesevier.

Guoqiang, Z,. Michael Y. H. B. Eddy P. Daniel C. I. (1996) Artfcial neural networks in bankruptcy prediction: General framework and cross-validation analysis, European Journal of Operational Research 116 (1999) 16±32, Elsevier.

How to cite this article:

Waleed Khalid Shihab and Sivaram Prasad R.2017, Artificial Neural Network. *Int J Recent Sci Res.* 8(7), pp. 18307-18328.
DOI: http://dx.doi.org/10.24327/ijrsr.2017.0807.0485

****