ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 6, December 2013

Kurdistan Engineering Colleges and Using of Artificial Neural Network for Knowledge Representation in learning process

Dr. Farhad Bilal Baha'addin
Faculty of Eng. and Applied Scie. School of Engineering, Water Resources Eng. Depart. Duhok University

Abstract: *The research presents the architecture and describes the functionality of a Web-based Artificial network (WBANN), which uses Neural Network for Learning & teaching Process. . Neural Network are a type of hybrid rules integrating symbolic rules with Neural-computing. The use of Neural Network s as the knowledge representation basis of the artificial neural network results in a number of advantages. Part of the functionality of the artificial neural network is controlled by a neural -based inference engine. Apart from that, the system consists of four other components: the domain knowledge, containing the structure of the domain and the educational content, the user modeling component, which records information concerning the user, the pedagogical model, which encompasses knowledge regarding the various pedagogical decisions, and the supervisor unit that controls the functionality of the whole system. The system focuses on teaching & learning Internet technologies.*

Index terms: **Artificial Neural Network, Learning process, knowledge representation.**

I. INTRODUCTION

An artificial neural network (ANN), often just called a "neural network" (NN), is a mathematical model or computational model based on biological neural networks. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data [1].

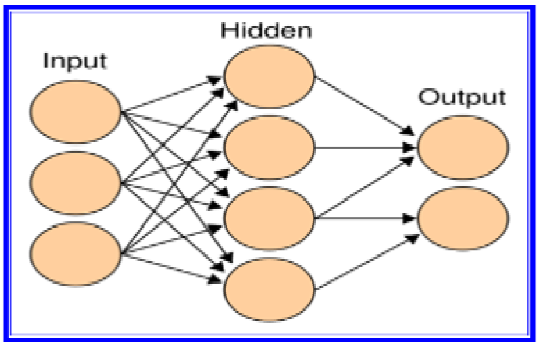

Fig (1): represent of neural network system

A neural network is an interconnected group of nodes, akin to the vast network of neurons in the human brain. Component based representation of a neural network. This kind of more general representation is used by some neural network software. There is no precise agreed-upon definition among researchers as to what a neural network is, but most would agree that it involves a network of simple processing elements (neurons), which can exhibit complex global behavior, determined by the connections between the processing elements and element parameters. The original inspiration for the technique was from examination of the central nervous system and the neurons (and their axons, dendrites and synapses) which constitute one of its most significant information processing elements (see Neuroscience). In a neural network model, simple nodes (called variously "neurons", "PEs" ("processing elements") or "units") are connected together to form a network of nodes — hence the term "neural network." While a neural network does not have to be adaptive per se, its practical use comes with algorithms designed to alter the strength (weights) of the connections in the network to produce a desired signal flow [2].

These networks are also similar to the biological neural networks in the sense that functions are performed collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned (see also connectionism). Currently, the term Artificial Neural Network (ANN) tends to refer mostly to neural network models employed in statistics, cognitive psychology and artificial intelligence. Neural network models designed with emulation of the central nervous system (CNS) in mind are a subject of theoretical neuroscience (computational neuroscience) [3].

In modern software implementations of artificial neural networks the approach inspired by biology has more or less been abandoned for a more practical approach based on statistics and signal processing. In some of these systems neural networks, or parts of neural networks (such as artificial neurons) are used as components in larger systems that combine both adaptive and non-adaptive elements. While the more general approach of such adaptive systems is more suitable for real-world problem solving, it has far less to do with the traditional artificial intelligence connectionist models. What they do, however, have in common is the principle of non-linear, distributed, parallel and local processing and adaptation [4].

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 6, December 2013

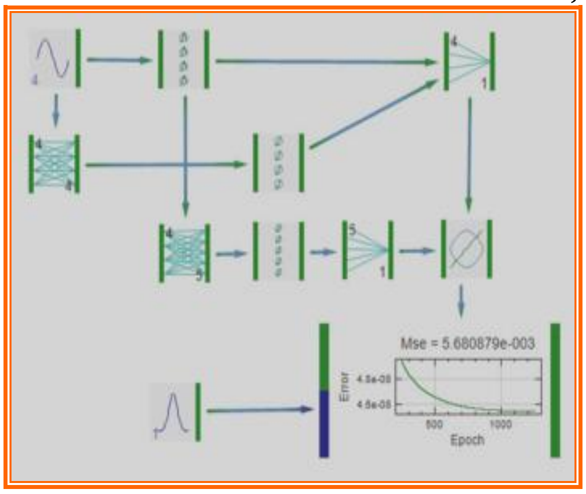

Fig (2): Show of the neural network in Electronic Systems

II. ARTIFICIAL NEURAL SYSTEMS

A. Basic concepts

Artificial neural systems (ANS) tried to solve problems by modeling the human brain information processing. They represent a different programming paradigm, also known as *connectionism*. Connectionism models solutions to problems by training simulated neurons connected in a network. ANS may well be the ideal front end to expert systems that require massive amounts of sensor inputs and fast response [5].

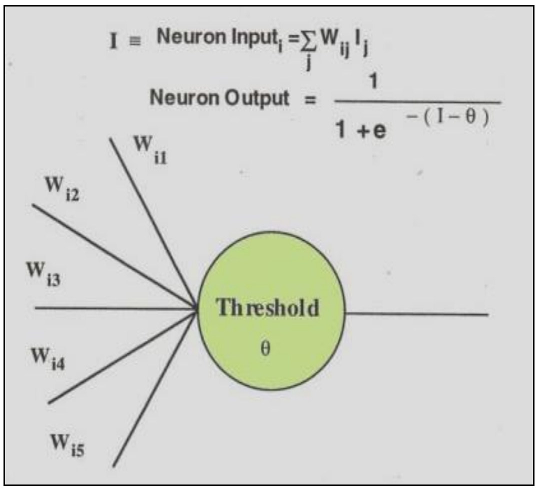
I = Neuron Input$_i = \sum_j W_{ij} I_j$
Neuron Output = $\frac{1}{1+e^{-(I-\theta)}}$
W_{i1}
W_{i2}
W_{i3}
W_{i4}
W_{i5}
Threshold
θ

Fig (3): A neuron processing element

ANS uses simple processing elements connected in parallel, called neurons. A neuron may have multiple inputs but only one output. The key to the functioning of an ANS is the weights associated with each element (refer Fig. 3). Each element (neuron) has an associated threshold value θ, subtracted from the input I. The input signals of the neuron are multiplied by the weights and summed to yield the total neuron input, equal to 1. The output is given by the formula in Fig. 3 and it is called an activation function [6].

B. ANS advantages

Since ANSes are modeled after current brain theories, information is represented by *weights*. The distributed representation of information is similar to a *hologram*, where the lines of the hologram diffract a laser beam passing through in order to reconstruct the stored image [7]. Advantages of ANS over conventional computers are fault tolerant storage. Removing portions of the net will only determine a degradation of the quality in the stored data; graceful degradation of the quality of the stored image, in proportion to the amount of net removed; data is stored in the form of *associative memory* (i.e. where partial data is sufficient to recall the complete stored information); nets can *extrapolate* and *interpolate* from their stored information. A net can be trained to seek significant features / relationships in the data. Subsequently, the net may suggest relationships with the new data. Nets display *plasticity*. After the removal of a part of the neurons in the net, the net can be retrained to its original skill level if sufficient neurons are left; ANSes are simple to build and cheap to maintain, mainly because of plasticity. However, for number intensive tasks or algorithmic solutions, ANSes are *not* a good choice [8].

C. ANS timeline

The origin of artificial neural systems dates back to 1943 (refer paragraph A). However, a milestone in the connectionist AI approach was 1961, with Rosenblatt's definition of perceptron, a new type of artificial neuron. The perceptron consisted of two layers of neurons, and a basic learning algorithm. It displayed remarkable capabilities for learning and pattern recognition. However in his model, the weights had to be set manually. In 1969, a new milestone occurs with the publication of the book Perceptions by Minksy and Papert. The book showed the limitations of the perceptrons as general computing machines, proving that a perceptron could not recognize the exclusive OR logic function. The book has had a negative impact on the ANS research. Eventually, new methods of representing symbolic AI information by frames became popular in the 1970s, but research only continued on a small scale [9].

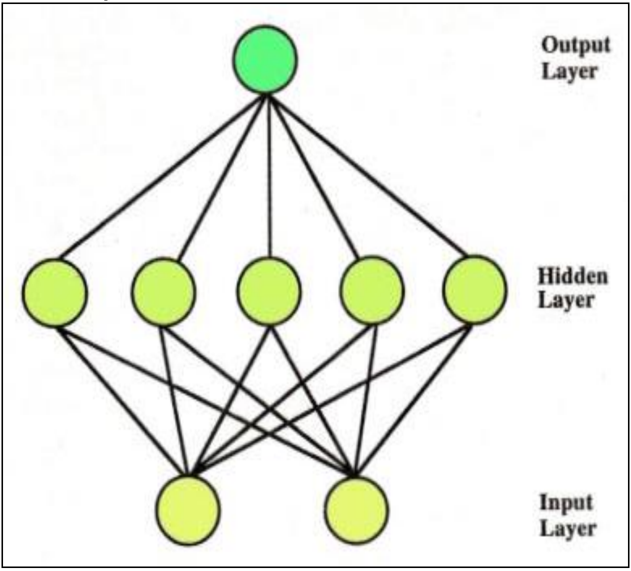

Fig (4): A back propagation net

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 6, December 2013

In 1982 though, Hopfield's work brought a revival of ANS by introducing the *two-layer Hopfield Net* and demonstrating that ANS *could* solve a wide variety of problems. The *back-propagation* net, commonly implemented as a three-layer net, is an example of ANS that can solve the XOR problem (refer Fig. 4). Other types of ANS emerged, such as the *counter-propagation* net, invented by Hecht- Nielsen in 1986. It was also proven that a three-layer network with *n* inputs and *2n+1* neurons in the hidden layer could map *any* continuous function.

The present trend in expert systems *is* towards the artificial neural nets. Also, there are masses of commercially available expert systems and shells. Companies and existing firms have been organized to develop ANS technology and products. For example, Nestor-Write is a program by Nestor Corp., which can recognize handwritten input and convert it to text. ANS simulators and hardware accelerator boards (to speed up the learning process) are also marketed by a number of companies such as Texas Instruments, Synaptic, Neural Tech, etc.

III. MODELS

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f : X \rightarrow Y$. Each type of ANN model corresponds to a *class* of such functions [9].

A. The network in artificial neural network

The word *network* in the term 'artificial neural network' arises because the function *f*(*x*) is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the *nonlinear weighted sum*, where

$$f(x) = K\left(\sum_i w_i g_i(x)\right),$$

Where *K* is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.

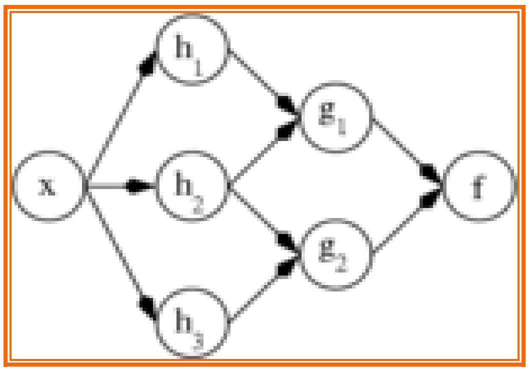

Fig (5):Show of network in artificial neural network ANN dependency graph

This figure depicts such a decomposition of *f*, with dependencies between variables indicated by arrows. These can be interpreted in two ways [10]. The first view is the functional view: the input *x* is transformed into a 3-dimensional vector *h*, which is then transformed into a 2-dimensional vector *g*, which is finally transformed into *f*. This view is most commonly encountered in the context of optimization.

The second view is the probabilistic view: the variable = *f*(*G*) depends upon the random variable *G* = *g*(*H*), which depends upon *H* = *h*(*X*), which depends upon the random variable *X*. This view is most commonly encountered in the context of graphical models.

The two views are largely equivalent. In either case, for this particular network architecture, the components of individual layers are independent of each other (e.g., the components of *g* are independent of each other given their input *h*). This naturally enables a degree of parallelism in the implementation [11].

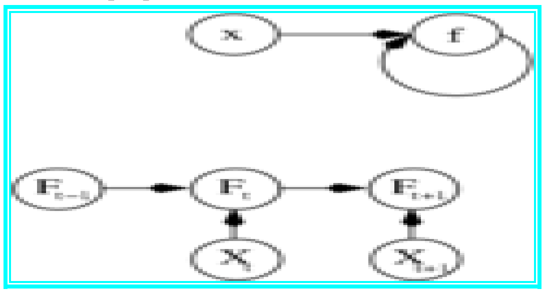

Fig (6):Show of ANN dependency graph Recurrent ANN dependency graph

Networks such as the previous one are commonly called feed-forward, because their graph is a directed acyclic graph. Networks with cycles are commonly called recurrent. Such networks are commonly depicted in the manner shown at the top of the figure, where *f* is shown as being dependent upon itself. However, there is an implied temporal dependence which is not shown. What this actually means in practice is that the value of *f* at some point in time *t* depends upon the values of *f* at zero or at one or more other points in time. The graphical model at the bottom of the figure illustrates the case: the value of *f* at time *t* only depends upon its last value[12].

B. Learning

However interesting such functions may be in themselves, what has attracted the most interest in neural networks is the possibility of *learning*, which in practice means the following: Given a specific *task* to solve, and a *class* of functions *F*, learning means using a set of *observations*, in order to find $f^* \in F$ which solves the task in an *optimal sense [13]*.

This entails defining a cost function $C : F \rightarrow \mathbb{R}$ such that, for the optimal solution f^*, $C(f^*) \leq C(f) \; \forall f \in F$ (no solution has a cost less than the cost of the optimal solution).

The cost function *C* is an important concept in learning, as it is a measure of how far away we are from an optimal solution to the problem that we want to solve. Learning algorithms search through the solution space in order to find a function that has the smallest possible cost[14].

For applications where the solution is dependent on some data, the cost must necessarily be a *function of the observations*; otherwise we would not be modeling anything related to the data. It is frequently defined as a statistic to which only approximations can be made. As a simple example consider the problem of finding the model *f* which minimizes $C = E\left[(f(x) - y)^2\right]$, for data pairs (*x*,*y*) drawn from some distribution $\mathcal{D}$. In practical situations we would only have *N* samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the true data distribution.

When $N \to \infty$ some form of online learning must be used, where the cost is partially minimized as each new example is seen. While online learning is often used when $\mathcal{D}$ is fixed, it is most useful in the case where the distribution changes slowly over time. In neural network methods, some form of online learning is frequently also used for finite datasets [15].

C. Choosing a cost function

While it is possible to arbitrarily define some ad hoc cost function, frequently a particular cost will be used either because it has desirable properties (such as convexity) or because it arises naturally from a particular formulation of the problem (i.e., In a probabilistic formulation the posterior probability of the model can be used as an inverse cost). Ultimately, the cost function will depend on the task we wish to perform. The three main categories of learning tasks are overviewed below [16].

D. Learning paradigms

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning. Usually any given type of network architecture can be employed in any of those tasks.

E. Supervised learning

In supervised learning, we are given a set of example pairs $(x, y), x \in X, y \in Y$ and the aim is to find a function $f : X \to Y$ in the allowed class of functions that matches the examples. In other words, we wish to *infer* the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain [17].

A commonly used cost is the mean-squared error which tries to minimize the average squared error between the network's output, f(x), and the target value y over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called Multi-Layer Perceptions, one obtains the common and well-known back-propagation algorithm for training neural networks. Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition). This can be thought of as learning with a "teacher," in the form of a function that provides continuous feedback on the quality of solutions obtained thus far[18].

F. Unsupervised learning

In unsupervised learning we are given some data *x*, and the cost function to be minimized can be any function of the data *x* and the network's output, *f*.

The cost function is dependent on the task (what we are trying to model) and our *a priori* assumptions (the implicit properties of our model, its parameters and the observed variables).

As a trivial example, consider the model $f(x) = a$, where a is a constant and the cost $C = E[(x - f(x))^2]$. Minimizing this cost will give us a value of *a* that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: For example in compression it could be related to the mutual information between x and y. In statistical modeling, it could be related to the posterior probability of the model given the data. (Note that in both of those examples those quantities would be maximized rather than minimized)[19].

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

G. Reinforcement learning

In reinforcement learning, data *x* is usually not given, but generated by an agent's interactions with the environment. At each point in time *t*, the agent performs an action y_t and the environment generates an observation x_t and an instantaneous cost c_t, according to some (usually unknown) dynamics. The aim is to discover a *policy* for selecting actions that minimizes some measure of a long-term cost, i.e. the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated [20].

More formally, the environment is modeled as a Markov decision process (MDP) with states $s_1, \ldots, s_n \in S$ and actions $a_1, \ldots, a_m \in A$ with the following probability distributions: the instantaneous cost distribution $P(c_t \mid s_t)$, the observation distribution $P(x_t \mid s_t)$ and the transition $P(s_{t+1} \mid$

s_t,a_t), while a policy is defined as conditional distribution over actions given the observations. Taken together, the two define a Markov chain (MC). The aim is to discover the policy that minimizes the cost, i.e. the MC for which the cost is minimal.

ANNs are frequently used in reinforcement learning as part of the overall algorithm. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

H. Learning algorithms

Training a neural network model essentially means selecting one model from the set of allowed models (or, in a Bayesian framework, determining a distribution over the set of allowed models) that minimizes the cost criterion. There are numerous algorithms available for training neural network models; most of them can be viewed as a straightforward application of optimization theory and Statistical estimation.

Most of the algorithms used in training artificial neural networks are employing some form of gradient descent. This is done by simply taking the derivative of the cost function with respect to the network parameters and then changing those parameters in a gradient-related direction.

Evolutionary methods simulated annealing, and expectation-maximization and non-parametric methods are among other commonly used methods for training neural networks. See also machine learning.

Temporal perceptual learning relies on finding temporal relationships in sensory signal streams. In an environment, statistically salient temporal correlations can be found by monitoring the arrival times of sensory signals. This is done by the perceptual network.

Employing artificial neural networks

Perhaps the greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism which 'learns' from observed data. However, using them is not so straightforward and a relatively good understanding of the underlying theory is essential [21].

- Choice of model: This will depend on the data representation and the application. Overly complex models tend to lead to problems with learning.
- Learning algorithm: There are numerous tradeoffs between learning algorithms. Almost any algorithm will work well with the *correct hyper parameters* for training on a particular fixed dataset. However selecting and tuning an algorithm for training on unseen data requires a significant amount of experimentation.
- Robustness: If the model, cost function and learning algorithm are selected appropriately the resulting ANN can be extremely robust.

With the correct implementation ANNs can be used naturally in online learning and large dataset applications. Their simple implementation and the existence of mostly local dependencies exhibited in the structure allows for fast, parallel implementations in hardware.

IV. APPLICATIONS

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical [22].

Real life applications

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind source separation and compression.

Application areas include system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications (automated trading systems), data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

V. NEURAL NETWORK SOFTWARE

Neural network software is used to simulate research, develop and apply artificial neural networks, biological neural networks and in some cases a wider array of adaptive systems. See also logistic regression.

Types of neural networks Feed forward neural network Main article: Feed-forward neural network

The feed forward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

Radial basis function (RBF) network

Radial Basis Functions are powerful techniques for interpolation in multidimensional space. A RBF is a function which has built into a distance criterion with respect to a centre. Radial basis functions have been applied in the area of neural networks where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in Multi-Layer Perceptions. RBF networks have two layers of processing: In the first, input is mapped onto each RBF in the 'hidden' layer. The RBF chosen is usually a Gaussian. In regression problems the output layer is then a linear combination of hidden layer values representing mean predicted output. The interpretation of this output layer value is the same as a regression model in statistics. In classification problems the output layer is typically a sigmoid function of a linear combination of hidden layer values, representing a

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 6, December 2013

posterior probability. Performance in both cases is often improved by shrinkage techniques, known as ridge regression in classical statistics and known to correspond to a prior belief in small parameter values (and therefore smooth output functions) in a Bayesian framework.

RBF networks have the advantage of not suffering from local minima in the same way as Multi-Layer Perceptions. This is because the only parameters that are adjusted in the learning process are the linear mapping from hidden layer to output layer. Linearity ensures that the error surface is quadratic and therefore has a single easily found minimum. In regression problems this can be found in one matrix operation. In classification problems the fixed non-linearity introduced by the sigmoid output function is most efficiently dealt with using iteratively re-weighted least squares.

RBF networks have the disadvantage of requiring good coverage of the input space by radial basis functions. RBF centers are determined with reference to the distribution of the input data, but without reference to the prediction task. As a result, representational resources may be wasted on areas of the input space that are irrelevant to the learning task. A common solution is to associate each data point with its own centre, although this can make the linear system to be solved in the final layer rather large, and requires shrinkage techniques to avoid over fitting.

Associating each input datum with an RBF leads naturally to kernel methods such as Support Vector Machines and Gaussian Processes (the RBF is the kernel function). All three approaches use a non-linear kernel function to project the input data into a space where the learning problem can be solved using a linear model. Like Gaussian Processes, and unlike SVMs, RBF networks are typically trained in a Maximum Likelihood framework by maximizing the probability (minimizing the error) of the data under the model. SVMs take a different approach to avoiding over fitting by maximizing instead a margin. RBF networks are outperformed in most classification applications by SVMs. In regression applications they can be competitive when the dimensionality of the input space is relatively small [23].

Kohonen self-organizing network

The self-organizing map (SOM) invented by Teuvo Kohonen performs a form of unsupervised learning. A set of artificial neurons learn to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space and the SOM will attempt to preserve these.

Recurrent network

Contrary to feed-forward networks, recurrent neural networks (RNs) are models with bi-directional data flow. While a feed-forward network propagates data linearly from input to output, RNs also propagate data from later processing stages to earlier stages [24].

Simple recurrent network

A *simple recurrent network* (SRN) is a variation on the Multi-Layer Perception, sometimes called an "Elman network" due to its invention by Jeff Elman. A three-layer network is used, with the addition of a set of "context units" in the input layer. There are connections from the middle (hidden) layer to these context units fixed with a weight of one. At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule (usually back-propagation) 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 is beyond the power of a standard Multi-Layer Perception [25].

In a *fully recurrent network,* every neuron receives inputs from every other neuron in the network. These networks are not arranged in layers. Usually only a subset of the neurons receive external inputs in addition to the inputs from all the other neurons, and another disjunctive subset of neurons report their output externally as well as sending it to all the neurons. These distinctive inputs and outputs perform the function of the input and output layers of a feed-forward or simple recurrent network, and also join all the other neurons in the recurrent processing.

Hopfield network

The Hopfield network is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield in 1982, this network guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable (or associative) memory, resistant to connection alteration [26].

Echo state network

The echo state network (ESN) is a recurrent neural network with a sparsely connected random hidden layer. The weights of output neurons are the only part of the network that can change and be learned. ESN are good to (re)produce temporal patterns.

Long short term memory network

The Long short term memory is an artificial neural net structure that unlike traditional RNNs doesn't have the problem of vanishing gradients. It can therefore use long delays and can handle signals that have a mix of low and high frequency components.

Stochastic neural networks

A stochastic neural network differs from a typical neural network because it introduces random variations into the network. In a probabilistic view of neural networks, such random variations can be viewed as a form of statistical sampling, such as Monte Carlo sampling.

Boltzmann machine

The Boltzmann machine can be thought of as a noisy Hopfield network. Invented by Geoff Hinton and Terry Sejnowski in 1985, the Boltzmann machine is important because it is one of the first neural networks to demonstrate learning of latent variables (hidden units). Boltzmann machine learning was at first slow to simulate, but the contrastive divergence algorithm of Geoff Hinton (circa 2000) allows models such as Boltzmann machines and products of experts to be trained much faster.

Modular neural networks

Biological studies have shown that the human brain functions not as a single massive network, but as a collection of small networks. This realization gave birth to the concept of modular neural networks, in which several small networks cooperate or compete to solve problems.

Committee of machines

A committee of machines (CoM) is a collection of different neural networks that together "vote" on a given example. This generally gives a much better result compared to other neural network models. In fact in many cases, starting with the same architecture and training but using different initial random weights gives vastly different networks. A CoM tends to stabilize the result.

The CoM is similar to the general learning bagging method, except that the necessary variety of machines in the committee is obtained by training from different random starting weights rather than training on different randomly selected subsets of the training data.

Associative neural network (ASNN)

The ASNN is an extension of the committee of machines that goes beyond a simple/weighted average of different models. ASNN represents a combination of an ensemble of feed-forward neural networks and the k-nearest neighbor technique (kNN). It uses the correlation between ensemble responses as a measure of distance amid the analyzed cases for the kNN. This corrects the bias of the neural network ensemble. An associative neural network has a memory that can coincide with the training set. If new data becomes available, the network instantly improves its predictive ability and provides data approximation (self-learn the data) without a need to retrain the ensemble. Another important feature of ASNN is the possibility to interpret neural network results by analysis of correlations between data cases in the space of models. The method is demonstrated at http://www.vcclab.org/lab/asnn, where you can either use it online or download it [27].

Other types of networks

These special networks do not fit in any of the previous categories.

Holographic associative memory

Holographic associative memory represents a family of analog, correlation-based, associative, stimulus-response memories, where information is mapped onto the phase orientation of complex numbers operating. Jean-Claude Perez from IBM proposes in 1990 years a neural network model entitled *"Fractal Chaos"* providing holographic-like emergent properties [1][2][3].

Instantaneously trained networks

Instantaneously trained neural networks (ITNNs) were inspired by the phenomenon of short-term learning that seems to occur instantaneously. In these networks the weights of the hidden and the output layers are mapped directly from the training vector data. Ordinarily, they work on binary data, but versions for continuous data that require small additional processing are also available.

Spiking neural networks

Spiking neural networks (SNNs) are models which explicitly take into account the timing of inputs. The network input and output are usually represented as series of spikes (delta function or more complex shapes). SNNs have an advantage of being able to process information in the time domain (signals that vary over time). They are often implemented as recurrent networks. SNNs are also a form of pulse computer.

Networks of spiking neurons — and the temporal correlations of neural assemblies in such networks — have been used to model figure/ground separation and region linking in the visual system.

Gerstner and Kistler have a freely available online textbook on Spiking Neuron Models.

Spiking neural networks with axonal conduction delays exhibit polychronization, and hence could have a potentially unlimited memory capacity.[*citation needed*]

In June 2005 IBM announced construction of a Gene supercomputer dedicated to the simulation of a large recurrent spiking neural network.

Dynamic neural networks

Dynamic neural networks not only deal with nonlinear multivariate behavior, but also include (learning of) time-dependent behavior such as various transient phenomena and delay effects [28].

Cascading neural networks

Cascade-Correlation is an architecture and learning algorithm developed by Scott Fahlman and Christian Lebiere. Instead of just adjusting the weights in a network of fixed topology, Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden units one by one, creating a multi-layer structure. Once a new hidden unit has been added to the network, its input-side weights are frozen. This unit then becomes a permanent feature-detector in the network, available for producing

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 6, December 2013

outputs or for creating other, more complex feature detectors. The Cascade-Correlation architecture has several advantages over existing algorithms: it learns very quickly, the network determines its own size and topology, it retains the structures it has built even if the training set changes, and it requires no back-propagation of error signals through the connections of the network. See: Cascade correlation algorithm.

Neuro-fuzzy networks

A neuro-fuzzy network is a fuzzy inference system in the body of an artificial neural network. Depending on the *FIS* type, there are several layers that simulate the processes involved in a *fuzzy inference* like fuzzification, inference, aggregation and defuzzification. Embedding an *FIS* in a general structure of an *ANN* has the benefit of using available *ANN* training methods to find the parameters of a fuzzy system.

Compositional pattern-producing networks

Compositional pattern-producing networks (CPPNs) are a variation of ANNs which differ in their set of activation functions and how they are applied. While typical ANNs often contain only sigmoid functions (and sometimes Gaussian functions), CPPNs can include both types of functions and many others. Furthermore, unlike typical ANNs, CPPNs are applied across the entire space of possible inputs so that they can represent a complete image. Since they are compositions of functions, CPPNs in effect encode images at infinite resolution and can be sampled for a particular display at whatever resolution is optimal.

One-shot associative memory

This type of network can add new patterns without the need for re-training. It is done by creating a specific memory structure, which assigns each new pattern to an orthogonal plane using adjacently connected hierarchical arrays [4]. The network offers real-time pattern recognition and high scalability; it however requires parallel processing and is thus best suited for platforms such as Wireless sensor networks (WSN), Grid computing, and GPGPUs.

Theoretical properties Computational power

The multi-layer perceptron (MLP) is a universal function approximate, as proven by the Cybenko theorem. However, the proof is not constructive regarding the number of neurons required or the *settings of the weights*.

Work by HavaSiegelmann and Eduardo D. Sontag has provided a proof that a specific recurrent architecture with rational valued weights (as opposed to the commonly used floating point approximations) has the full power of a Universal Turing Machine. They have further shown that the use of irrational values for weights results in a machine with trans-Turing power.

Capacity

Artificial neural network models have a property called 'capacity', which roughly corresponds to their ability to model any given function. It is related to the amount of information that can be stored in the network and to the notion of complexity.

Convergence

Nothing can be said in general about convergence since it depends on a number of factors. Firstly, there may exist many local minima. This depends on the cost function and the model. Secondly, the optimization method used might not be guaranteed to converge when far away from a local minimum. Thirdly, for a very large amount of data or parameters, some methods become impractical. In general, it has been found that theoretical guarantees regarding convergence are an unreliable guide to practical application.

Generalization and statistics

In applications where the goal is to create a system that generalizes well in unseen examples, the problem of overtraining has emerged. This arises in over complex or over specified systems when the capacity of the network significantly exceeds the needed free parameters. There are two schools of thought for avoiding this problem: The first is to use cross-validation and similar techniques to check for the presence of overtraining and optimally select hyper-parameters such as to minimize the generalization error. The second is to use some form of regularization. This is a concept that emerges naturally in a probabilistic (Bayesian) framework, where the regularization can be performed by selecting a larger prior probability over simpler models; but also in statistical learning theory, where the goal is to minimize over two quantities: the 'empirical risk' and the 'structural risk', which roughly correspond to the error over the training set and the predicted error in unseen data due to over fitting [29].

Confidence analysis of a neural network

Supervised neural networks that use an MSE cost function can use formal statistical methods to determine the confidence of the trained model. The MSE on a validation set can be used as an estimate for variance. This value can then be used to calculate the confidence interval of the output of the network, assuming a normal distribution. A confidence analysis made this way is statistically valid as long as the output probability distribution stays the same and the network is not modified.

By assigning soft ax activation function on the output layer of the neural network (or a soft-max component in a component-based neural network) for categorical target variables, the outputs can be interpreted as posterior probabilities. This is very useful in classification as it gives a certainty measure on classifications. The soft-max activation function:

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{c} e^{x_j}}$$

Dynamic properties

This article or section is in need of attention from an expert on the subject. Wiki Project Technology or the Technology Portal may be able to help recruit one. If a more appropriate Wiki Project or portal exists, please adjust this template accordingly.

Various techniques originally developed for studying disordered magnetic systems (i.e. the spin glass) have been successfully applied to simple neural network architectures, such as the Hopfield network. Influential work by E. Gardner and B. Derrida has revealed many interesting properties about perceptions with real-valued synaptic weights, while later work by W. Krauth and M. Mezard has extended these principles to binary-valued synapses.

REFERENCES

[1] Bar-Yam, Yaneer (2003). Dynamics of Complex Systems, Chapter 2.

[2] Bar-Yam, Yaneer (2003). Dynamics of Complex Systems, Chapter 3.

[3] Bar-Yam, Yaneer (2005). Making Things Work. Please see Chapter 3 .

[4] Bhadeshia H. K. D. H. (1992). "Neural Networks in Materials Science". ISIJ International 39: 966–979. doi: 10.2355/isijinternational.39.966.

[5] http://www.cit.gu.edu.au/~noran/cit_6105.

[6] George F. Luger, William A. Stubblefield (1993) Artificial Intelligence - Structures and Strategies for Complex Problem Solving, Benjamin-Cummings, Albuquerque, ISBN 0-8053-4780-1.

[7] Stephen L. Gallant, Connectionist Expert Systems, Comm of the ACM, Feb 1998.

[8] Buchanan, Feigenbaum, Dendral and META-Dendral: Their Applications Dimension, Artificial Intelligence, 1978. AI Magazine, IEEE Journal, AI Expert publications, AAAI web site papers.

[9] Bhagat, P.M. (2005) Pattern Recognition in Industry, Elsevier. ISBN 0-08-044538-1.

[10] Bishop, C.M. (1995) Neural Networks for Pattern Recognition, Oxford: Oxford University Press. ISBN 0-19-853849-9 (hardback) or ISBN 0-19-853864-2 (paperback).

[11] Cybenko, G.V. (1989). Approximation by Superposition's of a sigmoidal function, Mathematics of Control, Signals and Systems, Vol. 2 pp. 303-314. electronic version.

[12] Duda, R.O., Hart, P.E., Stork, D.G. (2001) Pattern classification (2nd edition), Wiley, ISBN 0-471-05669-3.

[13] Egmont-Petersen, M., de Ridder, D., Handels, H. (2002). "Image processing with neural networks - a review". Pattern Recognition 35: 2279–2301. doi:10.1016/S0031-3203(01)00178-9

[14] .Gurney, K. (1997) an Introduction to Neural Networks London: Routledge. ISBN 1-85728-673-1 (hardback) or ISBN 1-85728-503-4 (paperback)

[15] Haykin, S. (1999) Neural Networks: A Comprehensive Foundation, Prentice Hall, ISBN 0-13-273350-1.

[16] Fahlman, S, Lebiere, C (1991). The Cascade-Correlation Learning Architecture, created for National Science Foundation, Contract Number EET-8716324, and Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under Contract F33615-87-C-1499. Electronic version.

[17] Hertz, J., Palmer, R.G., Krogh. A.S. (1990) Introduction to the theory of neural computation, Perseus Books. ISBN 0-201-51560-1.

[18] Lawrence, Jeanette (1994) Introduction to Neural Networks, California Scientific Software Press. ISBN 1-883157-00-5.

[19] Masters, Timothy (1994) Signal and Image Processing with Neural Networks, John Wiley & Sons, Inc. ISBN 0-471-04963-8.

[20] Ness, Erik. 2005. SPIDA-Web. Conservation in Practice 6(1):35-36. On the use of artificial neural networks in species taxonomy.

[21] Ripley, Brian D. (1996) Pattern Recognition and Neural Networks, Cambridge.

[22] Siegelmann, H.T. and Sontag, E.D. (1994). Analog computation via neural networks, Theoretical Computer Science, v. 131, no. 2, pp. 331-360. Electronic version.

[23] Smith, Murray (1993) Neural Networks for Statistical Modeling, Van Nostrand Reinhold, ISBN 0-442-01310-8.

[24] Wasserman, Philip (1993) Advanced Methods in Neural Computing, Van Nostrand Reinhold, ISBN 0-442-00461-3.

AUTHOR BIOGRAPHY

Dr.Farhad Bilal was born in Erbil, IRAQ, 1950. Received master (MSc) of condition monitoring program, from University of Swansea, U.K., 1984. And received Ph.D. from U.S.M. Work as Lecturer in faculty of Engineering, university of Duhok, Kurdistan Region, IRAQ.