Artificial Neural Network: A Soft Computing Application In Biological Sequence Analysis

Rabi Narayan Behera
Institute of Engineering & Management, Salt Lake, Kolkata-700 091, India

ABSTRACT

The huge amount of available biological expression level data sets all over the world requires automated techniques that help scientists to analyze, understand, and cluster protein structure prediction, multiple alignment of sequences, phylogenic inferences etc. Major hurdles at this point include computational complexity and reliability of the searching algorithms. artificial neural networks a Soft Computing Paradigm widely used in analyzing gene expression level data produced by microarray technology on the genomic scale, sequencing genes on the genomic scale, sequencing proteins and amino acids, etc because of its features such as strong capacity of nonlinear mapping, high accuracy for learning, and good robustness. ANN be trained from examples without the need for a thorough understanding of the task in hand, and able to show surprising generalization performance and predicting power In this paper we review some of the artificial neural networks and related pattern recognition techniques that have been used in this area.

General Terms: *Artificial Neural Networks, Architectures, Learning Algorithms, Topologies, Biological Sequence.*

I. INTRODUCTION

Advancement in the genomics arena like The Human Genome Project (HGP), Microarray Data Analysis can produce large amounts of expression level data by parallel processing. Microarray Data Analysis aims to measure mRNA levels in particular cells or tissues for many genes at once that relies on the hybridization properties of nucleic acids *on a genomic scale*. Microarrays have given a new direction towards the study of genome by allowing researchers to study the expression of thousands of genes simultaneously for the first time. It is being predicted that this is essential to understanding the role of genes in various biological functions.

The action of discovering a motif (a short DNA or protein sequence) of gene expression is closely related to correlating sequences of genes to specific biological functions, helps in understanding the role of genes in biological functions.

The most challenging area of Computational Biology is to study thousands of genes across diverse conditions simultaneously. New computational and data mining techniques need to be developed for low-cost, low precision (approximate), good solutions in order to properly comprehend and interpret expression.

Gene Expression Analysis includes Differential analysis/marker selection for searching genes that are differentially expressed in distinct phenotypes, Class prediction (supervised learning), Class discovery (unsupervised learning), Pathway analysis(search for sets of genes differentially expressed in distinct phenotypes) and data conversion tasks, such as filtering and normalizing, which are standard prerequisites for genomic data analysis.

Artificial Neural Network (ANNs) inspired by Biological Neural Network (BNNs) and Statistical Learning are widely used pattern recognition techniques in microarray data analysis. ANN can be trained to identify the relevant genes that are used to make this distinction which is also distinguishing between *complementary DNA* (cDNA) microarray data of two types of colorectal lesions.

We have surveyed a number of Artificial Neural Network (ANNs), some important learning algorithms and related techniques that help (have been used) in discovering or correlating patterns of expression data (e.g. microarray data) to biological function. We have given a chronological survey of calculating number of hidden neurons in neural network. In this paper you will find a summary of our survey along with basic concepts of ANNs and some important learning algorithms.

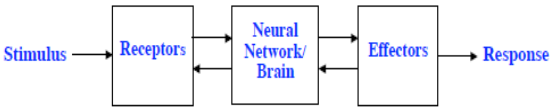
Artificial Neural Networks (ANNs) is unlike traditional computing (algorithmic approach) which require prior instruction to solve a problem, belong to the adaptive class of techniques in the *machine learning* arena. ANNS are used as a solution to various nonlinear complex problems; however, their success as an intelligent pattern recognition methodology has been most prominently advertised.

ANNs were inspired by the biological neural networks, whose performance is analogous to the most basic functions of human brains. Most models of ANNs are organized in the form of a number of processing units (also called artificial neurons, or simply neurons [1]), and a number of weighted connections (artificial synapses) between the neurons.

Though artificial neural networks are not an exact copy of biological human brain, it is important to begin with understanding fundamental concepts of Organization of the Nervous System and Brain.

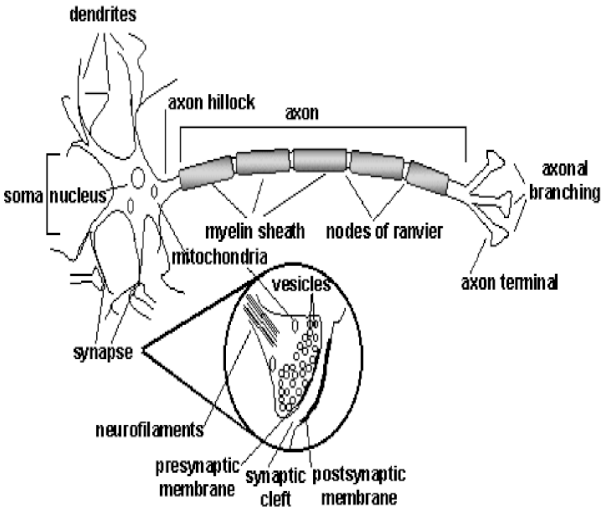
Nervous System:

The human nervous system can be broken down into three stages that may be represented in block diagram form as:



The receptors collect information from the environment (e.g. photons on the retina), The effectors generate interactions with the environment (e.g. activate muscles), The flow of information/activation is represented by arrows – feedforward and feedback. Naturally, this paper will be primarily concerned with how the neural network in the middle works.

Basic Components of Biological Neurons:



The majority of ***neurons*** encode their activations or outputs as a series of brief electrical pulses (i.e. spikes or action potentials). When sufficient input is received (i.e. a threshold is exceeded), the neuron generates an action potential or 'spike' (i.e. it 'fires').

That action potential is transmitted along the axon to other neurons, or to structures outside the nervous systems (e.g., muscles).

The neuron's ***cell body (soma)*** processes the incoming activations and converts them into output activations by summation of incoming signals (spatially and temporally).

The neuron's ***nucleus*** contains the genetic material in the form of DNA. This exists in most types of cells, not just neurons.

Dendrites are fibres which emanate from the cell body and provide the receptive zones that receive activation from other neurons. Signals from connected neurons are collected by the dendrites.

Axons are fibres acting as transmission lines that send activation to other neurons.

The junctions that allow signal transmission between the axons and dendrites are called ***synapses***. The process of transmission is by diffusion of chemicals called ***neurotransmitters*** across the synaptic cleft.

If sufficient input is not received (i.e. the threshold is not exceeded), the inputs quickly decay and no action potential is generated.

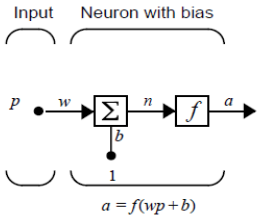
Timing is clearly important – input signals must arrive together strong inputs will generate more action potentials per unit time.

The process of building an ANN, similar to its biological inspiration, involves a learning episode (also called training). During learning episode, the Network requires a massive number of associative mappings to observe a sequence of recorded data, and adjusts the strength of its synapses according to a learning algorithm and based on the observed data, so that a particular input leads to a specific target output.

Training a neural network is the process of finding a set of weight and bias values so that for a given set of inputs, the outputs produced by the neural network are very close to some target values.

A neural network can be thought of as a complicated mathematical function that has various constants called weights and biases, which must be determined. The process of finding the set of weights and bias values that best match your existing data is called training the neural network. There are many ways to train a neural network.

Learning algorithms are generally divided into two types, supervised and unsupervised.

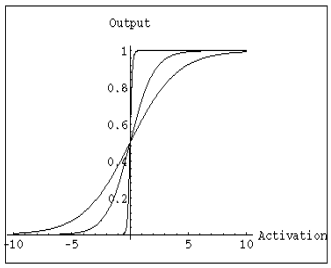


The scalar input *p* is transmitted through a connection that multiplies its strength by the scalar weight *w*, to form the product *wp*, again a scalar. *b* is a bias value allows you to shift the activation function to the left or right**,** which may be critical for successful learning. The activation function of a node defines the output of that node given an input or set of inputs. Neurons are "switches" that output a "1" when they are sufficiently Activated, and '0' when not. Here the weighted input *wp* + *b* is the argument of the activation function *f*, typically a step function or a sigmoid function, which takes the argument *n* and produces the scalar output *a*. The bias is much like a weight, except that it has a constant input of 1. [2]

Depending upon the problem variety of Activation function is used

1. Linear Activation function like step function
2. Nonlinear Activation function like sigmoid function

Using a nonlinear function which approximates a linear threshold allows a network to approximate nonlinear functions using only small number of nodes.



A list of common transfer function (activation functions) and their names can be found in following Table

Activation Functions	
Name	**Formula**
Identity	$f(x) = x$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Tanh	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Step	$f(x) = \begin{matrix} -1 & if\ x < 0 \\ 1 & if\ x \geq 0 \end{matrix}$

Training the network nothing but adjusting both *w* and *b* scalar parameters of the neuron itself so that the cost functions is minimized and to exhibits some desired or interesting behavior.

Cost function:

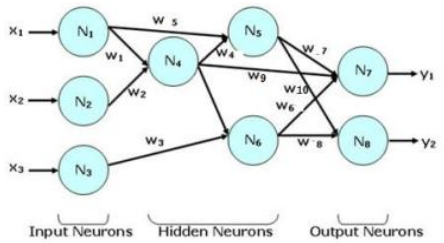
$\hat{C} = (\sum (x_i - x_i')^2) / N$
Where x_i is desired output and x_i's are the output of the neural network.
The supervised algorithms require labelled number of hidden neurons in neural network number of hidden neurons in neural network training data and external agent for supervision. In other words, they require more a prior knowledge about the training set.

The most important, and attractive, feature of ANNs is their capability of learning (generalizing) from example (Extracting knowledge from data) and adapt dynamically to the changing system. ANNs can do this without any pre-specified rules that define intelligence or represent an expert's knowledge. This feature makes ANNs a very popular choice for gene expression analysis and sequencing.

ANNs can select its own relevant input variable from the available data, and the model is developed subsequently which can in turn greatly increase the expert's knowledge and understanding of the problem.

In this paper, we discuss some important ANN like Feedforward Neural Networks and Backpropagation Neural Networks.

Feedforward Neural Network – The feedforward neural network consist of a series of layers. In this network the information moves in only one direction — forwards: From the input layer data goes through the hidden layer (if any) and to the output layer. There are no cycles or loops in the network. Any finite input-output mapping is possible with one hidden layer and enough neurons in it.



The network consists of **neurons** distributed in three different layers, each of which computes a function (called an **activation function**) of the inputs carried on the in-edges (edges pointing into the neuron) and sends the output on its out-edges (edges pointing out of the neuron). The inputs and outputs are weighed by **weights** W_i and shifted by **bias** factor specific to each neuron. It has been shown that for certain neural network topologies, any continuous function can be accurately approximated by some set of weights and biases. Therefore, we would like to have an algorithm which when given a function *f*, learns a set of weights and biases which accurately approximate the function. For **feed-forward neural networks** (artificial neural networks where the topology graph does not contain any directed cycles), the back-propagation algorithm described later does exactly that.

Output of a single neuron can be calculated as follows

Output = A ($\sum W_n * I_n$ + bias) where A is the activation function of the neuron, W_n is the weight of the n^{th} in-edge, In is the input carried across the n^{th} in-edge, and bias is the bias of the neuron.

Weight of a neuron can be calculated from the training data a pairs of sets of inputs and outputs (X_i,Y_i) where X_i is the input to all input neurons and Y_i is the output of the neural network after running the inputs in X_i.

The entire training dataset is D=Union $((X_i, Y_i))$

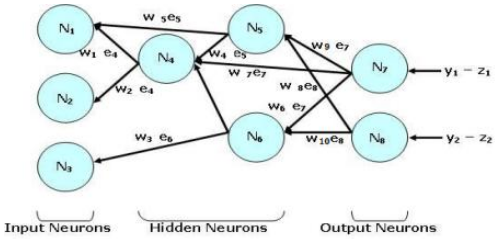
We need to find the weights of edges and biases of neurons in the network which minimize the sum of squares error (SSE) of the training data. The **SSE** of the network with respect to the training data D is computed using the formula,

$$SSE(D) = Sum((Y_i - Z_i)^2)$$

Feed-forward networks cannot perform temporal computation. More complex networks with internal feedback paths are required for temporal behavior.

Back propagation neural networks

The weights and biases are initially assigned to a random continuous (real) value between -1 and 1. The algorithm trains the neural network incrementally, meaning that each instance is considered independently, and after considering a training instance, the weights and biases are updated before considering another one. The update is done in a way that performs a gradient descent minimization of the error with respect to the weights and biases.



The learning algorithm can be split into three phases for an instance (X_i, Y_i) for i=1,2,3,….

Calculate Output - Calculate Z_i, the output of the network for the training instance's input set X_i.

Calculate Blame - Y_i is the desired value for Z_i, so when they are different, there is some error. From this error, we wish to compute a blame value for each neuron. Blames will later be used to decide by how much to adjust weights and biases. The blames are just the *linear* error. Blame can be shared with the help of Delta Rule.

$$\Delta W_{ji} = \eta * \delta_j * x_i$$

$$\delta_j = (t_j - y_j) * f'(h_j)$$

Where η is the learning rate of the neural network, t_j and y_j are targeted and actual output of the j^{th} neuron, h_j is the weighted sum of the neuron's inputs and f' is the derivate of the activation function f.

Problem with Multilayer Network is that we don't know the targeted output value for the Hidden layer neurons.

This can be solved by a trick:

$$\delta_i = \sum (\delta_k * W_{ki}) * f'(h_i)$$

The first factor in parenthesis involving the sum over k is an approximation to $(t_i - a_i)$ for the hidden layers when we don't know t_i.

Calculate Blame for Output Neurons - The blame of each output neuron is calculated as $Y_i - Z_i$. The blame should be proportional to partial derivatives for the gradient to minimize the sum *squared* error of the whole network.

Calculate Blame for Input and Hidden Neurons - The blame for an input or hidden neuron is calculated in terms of the blames of its out-neurons (the ones that it is pointing to). Because of this dependency, the blames need to be calculated in a topologically sorted order of the reverse of the network topology graph. This is similar to the order in which the outputs of neurons are calculated except that the topology graph must first be reversed before topologically sorting. The formula for the blame of an input or hidden neuron is $\sum W_k * E_k$ where E_k is the

blame of an out-neuron and W_k is the weight of the edge that connects to the out-neuron. This step of propagating the error backwards is why the learning algorithm is called back-propagation.

Adjust Weights and Biases – This last step performs the actual gradient descent by adjusting the weights with the formula: $W_{ij} = W_{ij} + r * E_j * A_j'(I_j) * O_i$ where r is the learning rate, E_j is the blame of neuron j, Aj' is the derivative of neuron j's activation function, I_j is the input fed to neuron j during calculation of the output in the first step, and O_i is the output of neuron i during the first step. The biases are similarly adjusted using the formula: $bias\_i = bias\_i + r*E_i$.

Learning completes with the all training instances are run. To achieve better accuracy, we may wish to run the training data multiple times through the learning algorithm without over fitting.

Neural networks need a lot of parameters like NN architecture, number of hidden neurons in hidden layer, activation function, inputs, and updating of weights etc and many of these parameters can significantly affect the accuracy, stability and runtime of neural networks. The backpropagation algorithm, one can only expect the algorithm to have good performance with a finite training set.

A fundamental question that is often raised in the applications of neural networks is *"how large does the net**work** have to be to perform a desired task?".* Answers to this question are directly related to the capability of neural networks, and should be given independently of the learning algorithms employed.

Number of Hidden Neurons

It is crucial to pick a proper neural network topology from infinite set of directed acyclic graphs before training with some restriction is that the number of input nodes must match the number of inputs to the function and the number of output nodes must match the number of outputs of the function. But we are free to choose any topology for the hidden neurons as long as there is a path from each input neuron to some output neuron and vice versa.

The fixing of the hidden neuron is an important for the designing of Neural Network model. The random selection of hidden neuron may cause over fitting and under fitting problem in the network. Several researchers tried and proposed many approach for fixing hidden neuron. The survey has been made to find the number of hidden neurons in neural network is and described in a chronological manner.

Year	Method/Concept	Reference
1991	exactly solves the multilayer neural network training problem, for any arbitrary training set with bounds on the size of a multilayer neural network to exactly implement an arbitrary training set;[3]	Michael A. Sartori and Panos J. Antsaklis
	Bounds on the Number of Hidden Neurons in Multilayer Perceptrons	Shih-Chi Huang and Yih-Fang Huang
1993	Proposed two parallel hyper plane methods for finding the number of hidden neurons. The 2^n / 3 hidden neurons are sufficient for this design of the network.[4]	Arai
1995	Investigated the estimation theory to find the number of hidden units in the higher order feedforward neural network. This theory is applied to the time series prediction. The determination of an optimal number of hidden neurons is obtained when the sufficient number of hidden neurons is assumed. According to the estimation theory, the sufficient number of hidden units in the second-order neural network and the first-order neural networks are 4 and 7, respectively. The simulation results show that the second-order neural network is better than the first-order in training convergence. According to that, the network with few nodes in the hidden layer will not be powerful for most applications. The drawback is long training and testing time.[5]	Li et al.
1996	presented another method to find an optimal number of hidden units. The drawback is that there is no guarantee that the network with a given number of hidden units will find the correct weights. According to the statistical behaviours of the output of the hidden units, if a network has large number of hidden nodes, a linear	Hagiwara

	relationship is obtained in the hidden nodes.[6]	
1997	Developed a method to fix hidden neuron with negligible error based on Akaike's information criteria. The number of hidden neurons in three layer neural network is N -1 and four-layer neural network is N/2 + 3 where N is the input-target relation[7]	Tamura and Tateishi
1998	Proposed a statistical estimation of number of hidden neurons. The merits are speed learning. The number of hidden neurons mainly depends on the output error. The estimation theory is constructed by adding hidden neurons one by one. The number of hidden neurons is formulated as N_h = K Log || P_cZ||/Log S, where S is total number of candidates that are randomly searched for optimum hidden unit c is allowable error.[8]	Fujita
1998	Upper bounds on the number of hidden neurons in feed forward networks with arbitrary bounded nonlinear activation functions- This paper used the fact what Sartori and Antsaklis pointed out in 1991, the sigmoid function is not necessary to be used as non-linearity of the neurons and . Huang and Babri showed that standard single-hidden layer feed forward networks (SLFN) with not more than N hidden neurons and with any bounded non linear activation function which has a limit at one infinity can learn these N distinct samples with zero error.	Guang-Bin Huang, Haroon A. Babri
1999	Presented a method to determine the number of hidden units which are applied in the prediction of cancer cells. Normally, training starts with many hidden units and then prune the network once it has trained. However pruning does not always improve generalization. The initial weights for input to hidden layer and the number of hidden units are determined automatically. The demerit is no optimal solution.[9]	Keeni et al
2001	Presented a statistical approach to find the optimal number of hidden units in prediction applications. The minimal errors are obtained by the increase of number of hidden units. Md. Islam and Murase proposed a large number of hidden nodes in weight freezing of single hidden layer networks. The generalization ability of network may be degraded when the number of hidden nodes(N_h) is large because N^{th} hidden node may have some spurious connections.[10]	Onoda, Md. Islam and Murase
2003	Implemented a set covering algorithm (SCA) in three-layer neural network. The SCA is based on unit sphere covering (USC) of hamming space. This methodology is based on the number of inputs. Theoretically the number of hidden neurons is estimated by random search. The output error decreases with N_h being added. The N_h is significant in characterizing the performance of the network. The number of hidden neurons should not be too large for heuristic learning system. The N_h found in set covering algorithm is 3L/2 hidden neurons where L is the number of unit spheres contained in N dimensional hidden space[11]	Zhang et al
2003	developed the model for learning and storage capacity of two-hidden-layer feedforward network. In 2003, Huang [12] formulated the following: in single hidden layer and in two hidden layer.	Huang
2004	A three-layer neural network (NN) with novel adaptive architecture has been developed. The hidden layer of the network consists of slabs of single neuron models, where neurons within a slab—but not between slabs— have the same type of activation function. The network activation functions in all three layers have adaptable parameters.	Panayiota Poirazi, Costas Neocleous, Costantinos S. Pattichis
2006	developed a separate learning algorithm which includes a deterministic and heuristic approach. In this algorithm, hidden-to-output and input-to-hidden nodes are separately trained. It solved the local minima in two-layered feedforward network. The achievement is best convergence speed[14]	Choi et al.
2006	Estimating the Number of Hidden Neurons in a Feedforward Network Using the Singular Value Decomposition - to quantify the significance of increasing the number of neurons in the hidden layer of a feedforward neural network architecture using the singular value decomposition (SVD).[15]	Eu Jin Teoh, Cheng Xiang, Kay Chen Tan

2007	Optimizing number of hidden neurons in neural networks - A method is proposed using, quantitative standard based on the Signal to Noise Ratio Figure (SNRF) to detect over fitting automatically using the training error only, to optimize the number of hidden neurons in Neural Network to avoid over fitting problem.[16]	Yinyin Liu, Janusz A. Starzyk, Zhen Zhu
2008	presented the lower bound on the number of hidden neurons. The number of hidden neurons is $N_h=q^n$ where q is valued upper bound function. The calculated values represent that the lower bound is tighter than the ones that has existed. The lower and upper bound on the number of hidden neurons help to design constructive learning algorithms. The lower bound can accelerate the learning speed, and the upper bound gives the stopping condition of constructive learning algorithms. It can be applied to the design of constructive learning algorithm with training set N numbers.[17]	Jiang et al
2009	investigated the effect of learning stability and hidden neuron in neural network. The simulation results show that the hidden output connection weight becomes small as number of hidden neurons N_h becomes large. This is implemented in random number mapping problems. The formula for hidden nodes is $N_h=\sqrt{N_iN_0}$ where N_i is the input neuron and N_0 is the output neuron. Since neural network has several assumptions which are given before starting the discussion to prevent divergent. In unstable models, number of hidden neurons becomes too large or too small. A trade-off is formed that if the number of hidden neurons becomes too large, output of neurons becomes unstable, and if the number of hidden neurons becomes too small, the hidden neurons becomes unstable again.[18]	Shibata and Ikeda
2009	Bernoulli Neural Network with Weights Directly Determined and with the Number of Hidden- Layer Neurons Automatically Determined Based upon polynomial interpolation and approximation theory, a special type of feedforward neural-network is constructed in this paper with hidden-layer neurons activated by Bernoulli polynomials to overcome the drawback of the conventional Back propagation neural network algorithm such as slow convergence and local minima existence. A weight detection algorithm is also proposed which detects weights of the neural network without the lengthy back propagation training procedures. More over a Structure Automatic Determination algorithm is proposed which can obtain the optimal number of hidden layers neurons in order of achieving the highest learning accuracy.[19]	Yunong Zhang, Gongqin Ruan
2010	proposed a technique to find the number of hidden neurons in MLP network using coarse-to-fine search technique which is applied in skin detection. This technique includes binary search and sequential search. This implementation is trained by 30 networks and searched for lowest mean squared error. The sequential search is performed in order to find the best number of hidden neurons[20]	Doukim et al.
2010	proposed the learning algorithms for determination of number of hidden neurons.[21]	Wu and Hong
2011	Panchal proposed a methodology to analyze the behavior of MLP. The number of hidden layers is inversely proportional to the minimal error.[22]	Panchal et al.
2011	Improving the character reorganization efficiency of feed forward BP Neural Network- Presented a method for improvement of character recognition capability of feed-forward back-propagation neural network by using one , two and three layers. Also it has been shown that with more hidden layers and modified momentum term a soft real time system can be introduced where performance is more critical. [23]	Amit Choudhary , Rahul Rishi
2011	A Clustering Based Method to Stipulate the Number of Hidden Neurons of MLP Neural Networks: Applications in Pattern Recognition we propose a clustering based method to state the number of hidden neurons that can produce good classification error rate. Our cluster based method uses much less CPU time consuming than method[24]	m.r. Silvestre, s.m. Oikawa2, f.h.t. Vieira, l.l. Ling
2012	Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture --	Karsoliya Saurabh

	The process of deciding the number of hidden layers and number of neurons in each hidden layer is still confusing. In this paper, an survey is made in order to resolved the problem of number of neurons in each hidden layer and the number of hidden layers required.[25]	
2012	developed a method used in proper NN architectures. The advantages are the absence of trial-and-error method and preservation of the generalization ability. The three networks MLP, bridged MLP, and fully connected cascaded network are used. The implemented formula: as follows, $N_h=N+1$ for MLP Network, $N_h=2N+1$ for bridged MLP Network and $N_h=2^n-1$ for fully connected cascade NN. The experimental results show that the successive rate decreases with increasing parity number. The successive rate increases with number of neurons used. The result is obtained with 85% accuracy.[26]	Hunter et al

Some sources claim that one hidden layer of neurons between the input nodes and output nodes are enough to accurately approximate any continuous function.

This is an extremely important finding because the computational complexity of the learning algorithm increases quadratically with the number of neurons in the network and this experiment shows that only very few neurons are necessary for this dataset.

Comparison to Other Classifiers: Neural networks are even a good alternative to other classification algorithms such as decision trees, Bayesian network, Inductive logic programming and naïve Bayes. The chart below shows the classification accuracy of neural networks compared to that of decision trees and naïve Bayes. It seems that neural networks perform about the same as those two classifiers and often times produce a little better results than naïve Bayes. Neural networks Can model more arbitrary functions (nonlinear interactions, etc.) and therefore might be more accurate, provided there is enough training data. But it can be prone to over-fitting as well.

Layered, feed-forward, backpropagation neural networks: These are a class of ANNs whose neurons are organized in layers. The layers are normally fully connected, meaning that each element (neuron) of a layer is connected to each element of the next layer. However, self-organizing varieties also exist in which a network starts either with a minimal number of synaptic connections between the layers and adds to the number as training progresses (*constructive*), or starts as a fully connected network and prunes connections based on the data observed in training (*destructive*).
Backpropagation [27] is a learning algorithm that, in its original version, belongs to the gradient descent optimization methods [28]. The combination of backpropagation learning algorithm and the feed-forward, layered networks provide the most popular type of ANNS. These ANNS have been applied to virtually all pattern recognition problems, and are typically the first networks tried on a new problem. The reason for this is the simplicity of the algorithm, and the vast body of research that has studied these networks. As such, in sequencing, many researchers have also used this type of network as a first line of attack.
Examples can be mentioned in [29,30]. In [29] Wu has developed a system called gene classification artificial neural system (GenCANS), which is based on a three layered, feed-forward backpropagation network. GenCANS was designed to "classify new (unknown) sequences into predefined (known) classes. It involves two steps, sequence encoding and neural network classification, to map molecular sequences (input) into gene families (output)". In [30] the same type of network has been used to perform rapid searches for sequences of proteins.

Other examples can be mentioned in Snyder and Stormo's work in designing a system called *GeneParser*.[31] Here authors experimented with two variations of a single layer network (one fully connected, and one partially connected with an activation bios added to some inputs), as well as a partially connected two-layer network. The authors use dynamic programming as the learning algorithm in order to train the system for protein sequencing.
1 ANNS were the first group of machine learning algorithm to be used on a biological pattern recognition problem. [32]
2 Distinguish between sporadic colorectal adenomas and cancers (SAC), and inflammatory bowel disease (IBD)-associated dysplasias and cancers.

Self-organizing neural networks: These networks are a very large class of neural networks whose structure (number of neurons, number of synaptic connections, number of modules, or number of layers) changes during learning based on the observed data. There are two classes of this type of networks: destructive and constructive. Destructive networks are initially a fully connected topology and the learning algorithm prunes synapses (sometime entire neurons, modules, or layers) based on the observed data. The final remaining network after learning is complete, usually is a sparsely connected network. Constructive algorithms start with a minimally connected network, and gradually add synapses (neurons, modules, or layers) as training progresses, in order to accommodate for the complexity of the task at hand.
Self-organizing networks divide into some major and very well known groups. Bellow we summarize the application some of these major groups in sequencing and expression level data analys is **Self-Organizing Map**: A self-organizing map (SOM) [33] is a type of neural network approach first proposed by Kohonen [34]. SOMs have been used as a divisive clustering approach in many areas including genomics. Several groups have used SOMs to discover patterns in gene expression data. Among these Toronen et al. [35], Golub et al. [36], and Tamayo et al. [37] can be mentioned. Tamayo and colleagues [37] use self-organizing maps to explore patterns of gene expression generated using Affymetrix arrays, and provide the GENECLUSTER implementation of SOMs. Tamayo and his colleagues explain the implementation of SOM for expression level data as follows: **"**An SOM has a set of nodes with a simple topology and a distance function on the nodes. Nodes are iteratively mapped into k-dimensional "gene expression space" (in which the i^{th} coordinate represents the expression level in the i^{th} sample)" [38]. A SOM assigns genes to a series of partitions on the basis of the similarity of their expression vectors to reference vectors that are defined for each partition. The summary of the basic SOM algorithm is perhaps best described in Quackenbush's review [39]: "First, random vectors are constructed and assigned to each partition. Second, a gene is picked at random and, using a selected distance metric, the reference vector that is closest to the gene is identified. Third, the reference vector is then adjusted so that it is more similar to the vector of the assigned gene. The reference vectors that are nearly on the two-dimensional gird are also adjusted so that they are more similar to the vector of the assigned gene. Fourth, steps 2 and 3 are iterated several thousand times, decreasing the amount by which the reference vectors are adjusted and increasing the stringency used to define closeness in each step. As the process continues, the reference vectors converge to fixed values. Last, the genes are mapped to the relevant partitions depending on the reference vector to which they are most similar"**.**
SOMs, like the *gene shaving* approach[40], have the distinct advantage that they allow a priori knowledge to be included in the clustering process. Tamayo et. al. explain this and other advantages of the SOM approach as follows: "The SOM has a number of features that make them particularly well suited to
clustering and analyzing gene expression patterns. They are ideally suited to exploratory data analysis, allowing one to impose partial structure on the clusters (in contrast to the rigid structure of hierarchical clustering, the strong prior hypotheses used in Bayesian clustering, and the nonstructural of k-means clustering) facilitating easy visualization and interpretation. SOMs have good computational properties and are easy to implement, reasonably fast, and are scalable to large data sets".
The most prominent disadvantage of the SOM approach is that it is difficult to know when to stop the algorithm. If the map is allowed to grow indefinitely, the size of SOM is gradually increased to a point when clearly different sets of expression patterns are identified. Therefore, as with k-means clustering, the user has to rely on some other source of information, such as PCA, to determine the number of clusters that best represents the available data. For this reason, Sasik and his colleagues believe that "SOM, as implemented by Tamayo et al is essentially a restricted version of k-means: Here, the *k* clusters are linked by some arbitrary user-imposed topological constraints (e.g. a 3 x 2 grid), and as such suffers from all of the problems mentioned above for k-means (and more), except that the constraints expedites the optimization process". [37]
There are many varieties to SOM, among which the self-organizing feature maps (SOFM) should be mentioned. The *growing cell structure* (GCS) [41] is another derivative of SOFM. It is a self organizing and incremental (constructive) neural learning approach. The unsupervised variety of GCS was used by Azuaje [42] for discovery of gene expression patterns in B-cell lymphoma. Using cDNA microarray expression data, GCS was able to "identify normal and diffuse large B-cell lymphoma (DLBCL) patients. Furthermore, it distinguishes patients with molecularly distinct types of DLBCL without previous knowledge of those subclasses".

Self-organizing trees*:* Self-organizing trees are normally constructive neural network methods that develop into a tree (usually binary tree) topology during learning. Among examples of these networks applied to sequencing the work of Dopazo et al. [43], Wang et al. [44,45], and Herrero et al. [46] can be mentioned.
Dopazo and Carazo introduce the self-organizing tree algorithm (SOTA). [43] SOTA is a hierarchical neural network that grows into a binary tree topology. For this reason SOTA can be considered a hierarchical clustering algorithm [41]. SOTA is based on Kohonen's SOM discussed above and Fritzke's growing cell [34]. SOTA was originally designed to analyze pre-aligned sequences of genes (polygenetic reconstruction).

Wang and colleagues extended that work in [43,44] in order to classify protein sequences: "(SOTA) is now adapted to be able to analyze patterns associated to the frequency of residues along a sequence, such as protein dipeptide composition and other n-gram compositions". In [46], Herrero and colleagues demonstrate the application of SOTA to Microarray expression data. The show that SOTA's performance is superior to that of classical hierarchical clustering techniques. Among the advantages of SOTA as compared to hierarchical cluster algorithms, one can mention its time complexity, and its top-to-bottom hierarchical approach. SOTA's runtimes are approximately linear with the number of items to be classified, making it suitable for large datasets. Also, because SOTA forms higher clusters in the hierarchy before forming the lower clusters, it can be stopped at any level of hierarchy and still produce meaningful intermediate results.

There are many other types of self-organizing trees that for space considerations cannot be mentioned here.

SOTA was only chosen here to give a brief introduction to these types of networks, and not because we believe that these are the most common types of self-organizing trees.

***ART* and its derivatives:** *Adaptive Resonance Theory* was introduced by Stephen Grossberg [47] in

1976. Networks designed based on ART are unsupervised and self-organizing, and only learn in the socalled "resonant" state. ART can form (stable) clusters of arbitrary sequences of input patterns by learning (entering resonant states) and self-organizing. Since the inception, many derivatives of ART have emerged. Among these *ART-1* (the binary version of ART; forms clusters of binary input data) [48], *ART-2*(analog version of ART) [49], *ART-2A* (fast version of ART-2) [50], *ART-3* (includes "chemical transmitters" to control the search process in a hierarchical ART structure) [50], *ARTMAP* (supervised version of ART) [51] can be mentioned. Many hybrid varieties such as *Fuzzy–ART* [52], *Fuzzy-ARTMAP* (supervised Fuzzy-ART) [52] and *simplified Fuzzy-ARTMAP* (SFAM) [51] have also been developed.

The ART family of networks has a broad application in virtually all areas of pattern recognition. As such, they have also been applied in biological sequencing problems. In general, in problem settings where the number of clusters is not previously known, researchers tend to use unsupervised ART, where when number of clusters is known a priori, usually the supervised version, ARTMAP, is used.

Among the unsupervised implementations, the work of Tomida et al. [53] should be mentioned. Here the authors used Fuzzy ART for expression level data analysis. This study is significant in that it compares results of Fuzzy ART with those of hierarchical clustering, k-mean clustering, and SOMs in analyzing and sequencing expression level data on the genomic scale. In this study the authors consider a two dimensional problem space in which the first dimension is the expression level and the second dimension is time, in order to consider the variation of expression levels in time. In this study, the authors report the lowest gap index3 among the four algorithms considered for Fuzzy ART. In terms of robustness of clustering results, the authors report a robustness of 82.2% for Fuzzy ART versus 71.1%, 44.4% and 46.7%, for hierarchical clustering, *k*-means clustering, and SOMs respectively. As a result the authorsconclude that Fuzzy ART also performs best in noisy data.

Among supervised implementations, Azuaje's use of SFAM in cDNA data analysis with the purpose of discovery of gene function in cancer can be mentioned. [54] In this study the authors use a data set generated for a previous study [18] using SOFM. Here the authors were able to also distinguish between classes of cancer using SFAM instead of SOFM (reported earlier in this paper).

Other neural network techniques

***Molecular computers* and *molecular neural networks*:** The idea of molecular and cellular computing dates back to the late 1950's when Richard Feynman delivered his famous paper describing "sub-microscopic" computers. He suggested that living cells and molecules could be thought of as potential computational components. Along the same lines, a revolutionary form of computing was developed by Leonard

Aldeman in 1994, when he announced that he had been able to solve a small instance of a computationally intractable problem using a small vial of DNA. [55] Aldeman had accomplished this by representing information as sequences of bases in DNA molecules. He then showed how to exploit the self assembly property of DNA and use DNA-manipulation techniques to implement a simple but massively parallel random search.

As a result, it has been suggested byMills [56] (among others) that a "DNA computer using cDNA as input might be ideal for clinical cell discrimination". In particular, Mills recommends the neural networks version of Aldeman's DNA computing scheme developed also by Mills and his colleagues [56] to the classification and sequencing tasks involved in expression level profiling. Mills reports in [56] that his "preliminary experimental results suggest that expression profiling should be feasible using a DNA neural network that acts directly on cDNA".

The DNA neural networks are the most recent and innovative ANN solution suggested for expression data profiling and sequencing. Its massively parallel nature offers exceedingly fast computation. The method presents high potential in various aspects of the solution that it offers and should offer breakthrough advancement to the field.

Bayesian Neural Networks: There are a number of recent networks that have been suggested as solutions for sequencing and expression data analysis. For instance, Bayesian neural networks (BNNs), are another technique that has been recently used for gene expression analysis. Liang et al. [57] have used the BNNs with structural learning for exploring microarray data in gene expressions. The BNNs are an important addition to the host of ANN solutions that have been offered to the problem at hand, as they represent a
large group of hybrid ANNs that combine classical ANNs with statistical classification and prediction theories.

CONCLUSION

There are a number of ANN solutions that have been offered to the problem of biological sequencing and expression data analysis. While many researchers have applied "classical" ANN solutions to the problem and are reporting favorable results as compared to non-adaptive counter parts, others have suggested or developed pioneering and revolutionary ANNs. Many of these techniques offer feasible solutions to the
above mentioned problems, while some also offer significant speedup in achieving the solutions. We believe that the performance of a technique should ultimately be measured in the biological soundness and relevance of its results. As such, we believe that techniques that allow the incorporation of a priori biological knowledge show greater potential.

3 In this study, the authors use gap index to "evaluate the similarity of profiles as area between each profile and average profile during temporal phase". This review represents only a small part of the research being conducted in the area, and only is meant as a complementary/continuation of the survey that others have conducted in this area. It should in no way to be taken as a complete survey of all algorithms. If not for any other reason, for the reason of limited space, some significant developments in the area had to be left out. Furthermore, new techniques and algorithm are being proposed for microarray data analysis on a daily basis, making survey articles such as this highly time dependent.

REFERENCES

[1] McCulloch, W.S. and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics. 5: 115-133.

[2] MH Beale, 1992, Neural Network Toolbox - MathWorks.

[3] M. A. Sartori and P. J. Antsaklis, "A simple method to derive bounds on the size and to train multilayer neural networks," IEEE Transactions on Neural Networks, vol. 2, no. 4, pp. 467–471, 1991.

[4] M. Arai, "Bounds on the number of hidden units in binary-valued three-layer neural networks," Neural Networks, vol. 6, no. 6, pp. 855–860, 1993.

[5] J.Y. Li, T. W. S. Chow, and Y. L. Yu, "Estimation theory and optimization algorithm for the number of hidden units in the higher-order feedforward neural network," in Proceedings of the IEEE International Conference on Neural Networks, vol. 3, pp. 1229–1233, December 1995.

[6] M. Hagiwara, "A simple and effective method for removal of hidden units and weights," Neurocomputing, vol. 6, no. 2, pp. 207–218, 1994.

[7] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," IEEE Transactions on Neural Networks, vol. 8, no. 2, pp. 251–255, 1997.

[8] O. Fujita, "Statistical estimation of the number of hidden units for feedforward neural networks," Neural Networks, vol. 11, no. 5, pp. 851–859, 1998.

[9] K.Keeni, K. Nakayama, and H. Shimodaira, "Estimation of initial weights and hidden units for fast learning of multi-layer neural networks for pattern classification," in Proceedings of the International Joint Conference on Neural Networks (IJCNN '99), vol. 3, pp. 1652–1656, IEEE, July 1999.

[10] T. Onoda, "Neural network information criterion for the optimal number of hidden units," in Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 1, pp. 275–280, December 1995.

[11] M. M. Islam and K. Murase, "A new algorithm to design compact two-hidden-layer artificial neural networks," Neural Networks, vol. 14, no. 9, pp. 1265–1278, 2001

[12] Z. Zhang, X. Ma, and Y. Yang, "Bounds on the number of hidden neurons in three-layer binary neural networks," Neural Networks, vol. 16, no. 7, pp. 995–1002, 2003.

[13] G. B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," IEEE Transactions on Neural Networks, vol. 14, no. 2, pp. 274–281, 2003. View at Publisher ·

[14] B. Choi, J. H. Lee, and D. H. Kim, "Solving local minima problem with large number of hidden nodes on two-layered feed-forward artificial neural networks," Neurocomputing, vol. 71, no. 16–18, pp. 3640–3643, 2008

[15] Teoh EJ, Tan KC, Xiang C., "Estimating the number of hidden neurons in a feedforward network using the singular value decomposition" IEEE Trans Neural Netw. 2006 Nov;17(6):1623-9.

[16] Yinyin Liu, Janusz A. Starzyk, Zhen Zhu. "optimizing number of hidden neurons in neural networks" 2007

[17]. N. Jiang, Z. Zhang, X. Ma, and J. Wang, "The lower bound on the number of hidden neurons in multi-valued multi-threshold neural networks," in Proceedings of the 2nd International Symposium on Intelligent Information Technology Application (IITA '08), pp. 103–107, December 2008

[17] K. Jinchuan and L. Xinzhe, "Empirical analysis of optimal hidden neurons in neural network modeling for stock prediction," in Proceedings of the Pacific-Asia Workshop on Computational Intelligence and Industrial Application, vol. 2, pp. 828–832, December 2008

[18]. K. Shibata and Y. Ikeda, "Effect of number of hidden neurons on learning in large-scale layered neural networks," in Proceedings of the ICROS-SICE International Joint Conference 2009 (ICCAS-SICE '09), pp. 5008–5013, August 2009.

[19] Y Zhang. "Bernoulli Neural Network with Weights Directly Determined and with the Number of Hidden- Layer Neurons Automatically Determined" ISNN '09 Proceedings of the 6th International Symposium on Neural Networks on Advances in Neural Networks Pages 36-45, 2009.

[20] C. A. Doukim, J. A. Dargham, and A. Chekima, "Finding the number of hidden neurons for an MLP neural network using coarse to fine search technique," in Proceedings of the 10th International Conference on Information Sciences, Signal Processing and Their Applications (ISSPA '10), pp. 606–609, May 2010.

[21]. Y. K. Wu and J. S. Hong, "A literature review of wind forecasting technology in the world," in Proceedings of the IEEE Lausanne Power Tech, pp. 504–509, July 2007.

[22]. G. Panchal, A. Ganatra, Y. P. Kosta, and D. Panchal, "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers," International Journal of Computer Theory and Engineeringvol, vol. 3, no. 2, pp. 332–337, 2011.

[23] Amit Choudhary, Rahul Rishi "Improving the character recognition efficiency of feed forward BP neural network"

[24] MR Silvestre "A Clustering Based Method to Stipulate the Number of Hidden Neurons of mlp Neural Networks: Applications in Pattern Recognition" TEMA Tend. Mat. Apl. Comput., 9, No. 2 (2011)

[25] *Saurabh Karsoliya* ," Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture "International Journal of Engineering Trends and Technology , 2012,

[26] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures: a comparative study," IEEE Transactions on Industrial Informatics, vol. 8, no. 2, pp. 228–240, 2012

[27]. M. H. Hassoun. *Fundamentals of Artificial Neural Networks*. MIT Press, March 1995.

[28] N LeRoux,"Using Gradient Descent for Optimization and Learning" 2009

[29] Wu, C. H. (1995) Chapter titled "Gene Classification Artificial Neural System" in *Methods In Enzymology: Computer Methods for Macromolecular Sequence Analysis*, Edited by Russell F. Doolittle, Academic Press, New York.

[30] Wu, Cathy, S. Zhao, H. L. Chen, C. J. Lo and J. McLarty. (1996). Motif identification neural design for rapid and sensitive protein family search. *CABIOS*, 12 (2), 109-118.

[31] Snyder, E. E., Stormo, G. D. (1995) Identification of Coding Regions in Genomic DNA. *J. Mol. Biol.* 248: 1-18.

[32] F Valafar. "Pattern Recognition Techniques in Microarray Data Analysis: A Survey", Special issue of Annals of New York Academy of Sciences, *Techniques in Bioinformatics and Medical Informatics*. (980) 41-64, December 2002.

[33] T. Kohonen, *Self-Organizing Maps*, Third Extended Edition. Springer, Berlin, Heidelberg, New York, 2001.

[34] Kohonen, T. "The Self-Organizing Map" (1991) *Proc. IEEE* 78, 1464–1480

[35] Toronen, P. *et al* Analysis of gene expression data using self organizing maps (1999) FEBS Letters, 451, 142-146

[36]. Golub, T. R. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring (1999) Science, 286, 531-537

[37] Tamayo, P. *et al*. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differtiation (1999) Proc. Natl. Acad. Sci. USA 96, 2907-2912

[38] Pablo Tamayo"Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation" Proc Natl Acad Sci U S A. Mar 16, 1999; 96(6): 2907–2912.

[39] Quackenbush, J. Computational Analysis of Microarray Data (2001) *Nature Genetics* 2, 418-427

[40] T Hastie" 'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns" Genome Biol. 2000; 1(2): research0003.1–research0003.21. Published online Aug 4, 2000.

[41] Fritzke B. 1994. Growing cell structures--A self-organizing network for unsupervised and supervised learning. *Neural Network 7*:1141-1160.

[42] Azuaje F, Unsupervised neural network for discovery of gene expression patterns in B-cell lymphoma. Online Journal of Bioinformatics 1:26-41, 2001

[43] Dopazo J. & Carazo J. M. (1997) Phylogenetic reconstruction using an unsupervised growing neural network that adopts the topology of a phylogenetic tree. *Journal of Molecular Evolution* 44, 226-233

[44] Wang HC, Dopazo J, Carazo JM (1998) Self-organizing tree growing network for classifying amino acids *Bioinformatics* 14(4): 376-377

[45] Wang HC, Dopazo J, de la Fraga LG, Zhu YP, Carazo JM (1998) Self-Organizing Tree-growing Network for the classification of Protein Sequences. *Protein Science* 7:2613-2622

[46] J. Herrero, A. Valencia, and J. Dopazo. *A hierarchical unsupervised growing neural network for clustering gene expression patterns*. Bioinformatics, 17:126--136, 2001

[47] Stephen Grossberg. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121-134, 1976. Reprinted in Anderson and Rosenfeld, 1988.

[48] Gail A. Carpenter and Stephen Grossberg. Art 2: Selforganisation of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919-4930, 1987

[49] Gail A. Carpenter, Stephen Grossberg, and D.B. Rosen. Art2-a: An adaptive resonance algorithm for rapid category learning and recognition. *Neural Networks*, 4:493-504, 1991.

[50] Gail A. Carpenter and Stephen Grossberg. Art3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, 3:129-152, 1990.

[51] T. Kasuba, "Simplified fuzzy ARTMAP," AI Expert, November (1993)

[52] Gail A. Carpenter, Stephen Grossberg, and D.B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759-771, 1991.

[53] S. Tomida, T. Hanai, H. Honda, and T. Kobayashi. Gene Expression Analysis Using Fuzzy ART *Genome Informatics* 12: 245–246 (2001) 245

[54] Azuaje, F. A computational neural approach to support the discovery of gene function and classes of cancer. IEEE Trans Biomed Eng 48:332-339, 2001.

[55] Adleman, L.M. (1994) Molecular computation of solutions to combinatorial problems. *Science* 266, November 11, 1021–1024.

[56] Mills, A.P. Jr. et al. (2000) DNA analog vector algebra and physical constraints on large-scale DNA-based neural network computation. In DNA Based Computers V, (Winfree, E. and Gifford, D.K., eds), pp. 65–73, American Mathematical Society

[57] Liang, Y. et al. Bayesian Neural Network for Microarray Data Proceedings of the IEEE International Joint Conference On Neural Network, Hawaii, (2002)