Computation of Neural Network using C# with Respect to Bioinformatics

Kumar Sarvesh[*], Singh R.P.[**], Mishra A.[***], Kumar Hemant[****]

[*]Sarvesh Kumar, Research Scholar, Dept. of Statistics & Computer Applications, T.M.B. University, Bhagalpur (India)
[**]Dr. Rajeshwar Prasad Singh, Bhagalpur college of engineering, Sabour, Bhagalpur, India
[***] DR. Akshoy Kumar Mishra, Post graduate of Statistics and computer applications , T.M.B. University, Bhagalpur, India
[****] Hemant Kumar, Directorate of Economics & Statistics (Planning Cadre), GNCT Delhi, India

Abstract- Neural network is the emerging field in the era of globalization which is fully based on the concept of soft-computing technique and bioinformatics. In the competitive market of new development process, Bioinformatics play the vital role to give the process of integration aspect as multidisciplinary subject like- biological Science, medicine science, computer science, engineering, chemical science, physical science as well as mathematical science who gives the experiences of artificial activities of human behaviour in the form of software. Now a days neural Network and its multidimensional approach give the idea for solving bioinformatics problems to handle imprecision, uncertainty in large and complex search spaces. This paper gives the emphasis on multidimensional approaches of neural network with soft computing paradigm using C# in bioinformatics with integrative research methodology. The overall process of multidimensional approaches of bioinformatics neurons can also be understood with the help of flow chart and diagram is the major concerned.

Index Terms- Soft-computing technique of C#, Neural Network, bioinformatics, Bioinformatics tools, Genetic algorithms.

I. INTRODUCTION

The paper tried to explore the exact relationship among neural network, genetic algorithm, and bioinformatics with the help of C# computational approach [1][2]. We all know the running world is fully depends of computer technique which play vital role in living style as well as working life from here and there. Now we are generating the idea of modelling and computational programming technique having multidimensional prospects who can behave like the human activities by artificial component. Hence we are fully concerned on the soft-computing process by introducing the application and utilization of neural network, genetic algorithm & bioinformatics [3][4].

Artificial neural networks as a major soft-computing technology has been extensively studied & applied during the last three decades. The Neural Network, especially the multilayer perceptions network with a back propagation training algorithm, have gained recognition in research and applications with various scientific and engineering areas. Soft computing techniques demonstrates the high standards of technology, algorithms, and tools in bioinformatics for dedicated purposes such as reliable and parallel genome sequencing, fast sequence comparison, search in databases, automated gene identification, efficient modelling and storage of heterogeneous data, etc. On the other side the continuous development of high quality biotechnology, e.g. micro-array techniques and mass spectrometry, which provide complex patterns for the direct characterization of cell processes. With the advance of gene expression data in the bioinformatics field, the questions which frequently arise, for both computer and medical scientists, are which genes are significantly involved in discriminating cancer classes and which genes are significant with respect to specific cancer pathology. Numerous computational analysis models have been developed to identify informative genes from the microarray data; however, the integrity of the reported genes is still uncertain. The flow chart and diagrammatical approaches are also presented for easy understanding the concept of bioinformatics in neural.

The complete paper is divided into six section including introduction and conclusion. Section one contains the brief idea of computation of neural network with Bioinformatics & Genetic concept. Section two includes the concept of soft-computation technique and its component on bioinformatics & neural network in c#. Section three concentrated on bioinformatics with its objective, scope, application along with development of algorithms in updated version. Computation of neural network, its advantage, application and connection with bio information includes in section four. Relationship among Neural Network, Genetic Algorithm and Bioinformatics will be explain in section five with complete compilation among all of them. The modified version of General Mathematical Model of Neural Network for Bioinformatics impact of hidden layers as per input to be discussed with complete computation in C# .In each section research to be worked on diagrammatical presentation of bioinformatics in Neural, Genetics as well as computing is the major concerned. Total ten figure/diagram to be presented for better understanding in concise manner having multidimensional approaches of these thrice concept. Last section gives the future scope and conclusion of this paper.

II. SOFT COMPUTING TECHNIQUE AND ITS PARADIGM

Soft computing techniques are often used in conjunction with rule-based expert systems where the knowledge is usually in the form of if-then rules. Every computational process purposely includes imprecision into the calculation on one or more levels and allow this imprecision either to change (decrease) the granularity of the problem, or “to soften" the goal

of optimization at some stages which is define as belonging to the field of soft computing. It is clear that this is a quite general definition including all those techniques and approaches that are considered as the components of soft computing according to the practice. If we further go in this approach, it is possible to say that soft computing encompasses two main conceptual components namely approximate reasoning and functional approximation with optimization. From the above point we are considering the soft computation with using C# as is an evolving concept to explore the theory. Its evolution can be easily interpreted in the term of integration of new topics/techniques that are in accordance with these properties and the corresponding conceptual components. The term soft computing was coined to refer a family of computational techniques particularly adapted to cope with a class of problems for with other techniques are not quite well suited. Basically soft computing is not a homogeneous body of concepts and techniques it is a partnership of distinct methods that in one way or another conform to its guiding principle. The principal constituents of soft computing are neurocomputing, and probabilistic reasoning, with the latter subsuming genetic algorithms, belief networks, chaotic systems, and parts of learning theory As the presence of genetic algorithms has gained importance in the field of more recent enumerations which have been included evolutionary computation (EC, a family including genetic algorithms) as an independent constituent of soft computing(C#). Soft computing techniques are meant to operate in an environment that is subject to uncertainty and imprecision. The guiding principle of soft computing (C#) is exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness, low solution cost and better rapport with reality. Imprecision results from our limited capability to resolve detail and encompasses the notions of partial, vague, noisy and incomplete information about the real world. In other words it becomes not only difficult or even impossible but also inappropriate to apply hard computing techniques when dealing with situations in which the required information is not available. The behaviour of the considering the system is not completely known or the measures of the underlying variables are noisy. But we can go one step further and analyze those situations where imprecision is not a drawback of the information we are managing but an intrinsic characteristic of that information. In that sense it is quite important to distinguish in between measurements and perceptions, and to discriminate those situations where we work on the basis of imprecise measurements from those others where we compute with perceptions. Compute with perception is clearly an approach to approximate reasoning (again one of the constituents of soft computing(C#)).

In that sense of soft computing as a tool to cope with uncertainty and imprecision should clearly include computing with words and perceptions as one of its basic components. Computing with words is inspired by the remarkable human capability to perform a wide variety of physical and mental tasks without any measurements and any computations. As a methodology, computing with words provides a foundation for a computational theory of perceptions. One of the fundamental aims of science has been and continues to be that of progressing from perceptions to measurements of the potential achievements of soft computing will be to return to perceptions in order to take profit of its qualities as a carrier for intrinsically imprecise information. The Centre focuses on Soft Computing(C#) following a well known principle of Genetic Algorithms that could be described as maintain an appropriate balance in between Exploitation and Exploration, i.e. “try to improve those useful techniques and fundamentals SC already exploit at the same time search for completely new options (explore) that will eventually outperform those already known.

III. PROPERTIES OF SOFT COMPUTING AND ITS COMPONENT

The characterizations of Soft Computing gives the emphasis by the essential properties as a family of techniques, as a complement of hard computing, as a tool for coping with imprecision and uncertainty. However, none of them can be called the "correct" or "best" answer: depending on the context, the background of the person asking the question and other factors, Every computing process that purposely includes imprecision into the calculation on one or more levels and allows this imprecision either to change (decrease) the granularity of the problem, or to "soften" the goal of optimization at some stage, is defined as to belonging to the field of soft computing. It is clear that this is a quite general definition including all those techniques and approaches that, according to the practice, are considered as the components of soft computing. If we go further in this approach, it is possible to say that, in summary, soft computing encompasses two main conceptual components, namely approximate reasoning and function approximation and optimization. From that point, and considering that soft computing is an evolving concept, its evolution can be easily interpreted in terms of the integration of new topics/techniques that are in accordance with these properties and the corresponding conceptual components. Soft computing(C#) is the mixture of several pre-existing techniques. Basically, soft computing is not a homogeneous body of concepts and techniques. Rather, it is a partnership of distinct methods that in one way or another conform to its guiding principle. The major properties of soft computing neural are as follows.

- The Neural Networks (NNs) exhibit mapping capabilities, that is, they can map input patterns to their associated output patterns.
- The Neural Networks (NNs) possess the capability to generalize. Thus, they can predict new outcomes from past trends.
- The Neural Networks (NNs) are robust systems and are fault tolerant. They can, therefore, recall full patterns from incomplete, partial or noisy patterns.
- The Neural Networks (NNs) can process information in parallel, at high speed, and in a distributed manner.
- **The Neural Networks (NNs) learn by examples**. Thus, Neural Network architectures can be 'trained' with known examples of a problem before they are tested for their 'inference' capability on unknown instances of the problem. They can, therefore, identify new objects previously untrained.

2.2 **SOFT COMPUTING vs HARD COMPUTING**

The term soft computing distinguishes those previously enumerated techniques from hard computing (conventional approaches) considered as less flexible and more computationally demanding. The key aspect for moving from hard to soft computing is the observation that the computational "To clarify that it would be easier to talk in terms of optimization, and the situation is that, for many different applications a sub-optimal solution is enough, and having that in mind when designing the optimization process will provide the difference in between obtaining a solution that satisfies our needs or getting lost when searching for the optimal solution".

Soft computing	**Hard computing**
Soft computing differs from conventional (hard) computing which tolerant of imprecision, uncertainty, partial truth, and approximation.	Hard computing is a conventional computing which requires a precisely stated analytical model and often a lot of computation time.
In effect, the role model for soft computing is the human mind.	Premises and guiding principles of Hard Computing are Precision, Certainty, and rigor.
The principal constituents, i.e., tools, techniques, of Soft Computing (SC) are Fuzzy Logic (FL), Neural Networks (NN), Support Vector Machines (SVM), Evolutionary Computation (EC), and Machine Learning (ML) and Probabilistic Reasoning (PR)	Many contemporary problems do not lend themselves to precise solutions such as Recognition problems (handwriting, speech, objects and images. Mobile robot coordination, forecasting, combinatorial problems etc.

Actually, the distinguishing feature of "soft computing" is straightforward however "Hard computing "uses an explicit model of the process under consideration while "Soft computing" does not do this. Instead, as an indispensable preliminary step, it infers an implicit model from the problem specification and the available data. Working out the analogy we can see that building the explicit model is a first step in the process of finding the optimal solution, while, in absence of such an explicit model, the use of an implicit model usually drives us to a (sub-optimal) solution satisfying our needs. Considering again the "view of soft computing" as blending approximate reasoning and function approximation/optimization, this view concentrates on the second part, the one usually assigned to neural networks and evolutionary computation. But if we consider optimization with a broader view, new components should be integrated in "soft computing(C#)". The term soft computing distinguishes those previously enumerated techniques from hard computing (conventional approaches) considered as less flexible and more computationally demanding. The key aspect for moving from hard to soft computing is the observation that the computational effort required by conventional approaches which makes in many cases the problem almost infeasible, is a cost paid to gain a precision that in many applications is not really needed or, at least, can be relaxed without a significant effect on the solution. In other words, we can assume that this imprecision is a small price to pay for obtaining more economical, less complex and more feasible solutions.

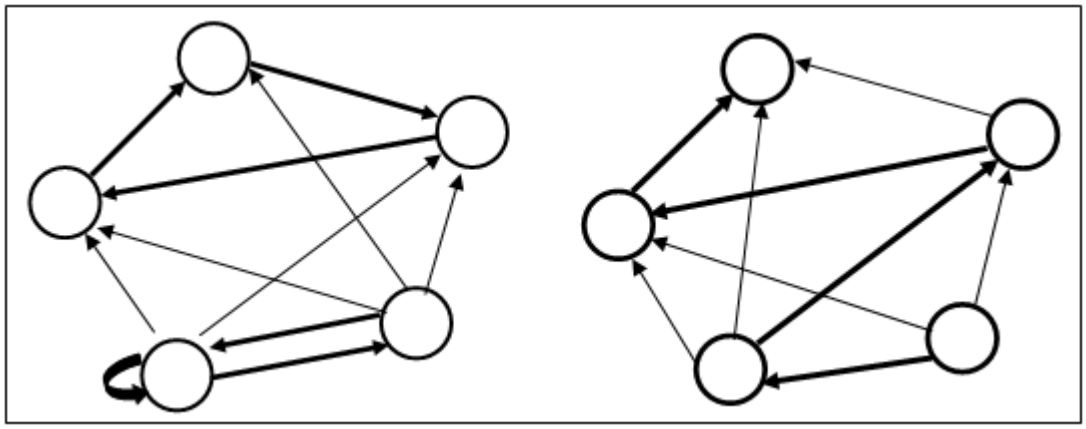
2.3 SOFT COMPUTING TECHNIQUES IN BIOINFORMATICS

Traditionally, a human being builds such an expert system by collecting knowledge from specific experts. The experts can always explain what factors they use to assess a situation however, it is often difficult for the experts to say what rules they use (for example, for disease analysis and control). This problem can be resolved by soft computing mechanisms. Soft computing(C#) mechanism can extract the description of the hidden situation in terms of those factors and then fire rules that match the expert's behaviour. In molecular biology research, new data and concepts are generated every day, and those new data and concepts update or replace the old ones. Soft computing can be easily adapted to a changing environment. This benefits system designers, as they do not need to redesign systems whenever the environment changes. Missing and noisy data is one characteristic of biological data. The conventional computer techniques fail to handle this. Soft computing based techniques are able to deal with missing and noisy data. With advances in biotechnology, huge volumes of biological data are generated. In addition, it is possible that important hidden relationships and correlations exist in the data. Soft computing methods are designed to handle very large data sets, and can be used to extract such relationships.

2.4. TYPE OF NEURAL NETWORK

There are several types of neural network topology exit on the basis of its properties, structure, design and architecture. The most famous are like:

Architectural Dynamics: The architectural dynamics specifies the network topology and its possible change. The architecture update usually applies within the framework of an adaptive mode in such a way that the network is supplied with additional neurons and connections when it is needed. However, in most cases the architectural dynamics assumes a fixed neural network topology, which is not changed anymore. Two types of architectures are distinguished: cyclic (recurrent) and acyclic (feed-forward) network. In the cyclic topology, there exists a group of neurons in the network, which are connected into a ring cycle. This means that in this group of neurons the output of the first neuron represents the input of the second neuron whose output is again the input for the third neuron, etc. as far as the output of the last neuron in this group is the input of the first neuron. The simplest cycle is a feedback of the neuron whose output serves simultaneously as its input. The maximum number of cycles is contained in the complete topology in which the output of each neuron represents the input for all neurons. An example of a general cyclic neural network is depicted in Fig. where all the cycles are indicated. On the contrary, the feed-forward neural networks do not contain any cycle and all paths lead in one direction. An example of an acyclic neural network is in Figure 1(a)&(b) where the longest path is marked.



(a) **(b)**

Figure1: Example of architectural dynamics: a. cyclic architecture, b. Acyclic architecture.

Feed-forward Networks :The neurons in the feed-forward networks can always be disjointly split into layers which are ordered (e.g. arranged one over another) so that the connections among neurons lead only from lower layers to upper ones and generally, they may skip one or more layers. Especially, in a multilayered neural network, the zero (lower), input layer consists of input neurons while the last (upper), output layer is composed of output neurons. The remaining, hidden (intermediate) layers contain hidden neurons. The layers are counted starting from zero that corresponds to the input layer, which is then not included in the number of network layers (e.g. a two-layered neural network consists of input, one hidden, and output layer). In the topology of a multilayered network, each neuron in one layer is connected to all neurons in the next layer (possibly missing connections between two consecutive layers might be implicitly interpreted as connections with zero weights). Therefore, the multilayered architecture can be specified only by the numbers of neurons in particular layers, typically hyphened in the order from input to output layer. In addition, any path in such a network leads from the input layer to the output one while containing exactly one neuron from each layer. An example of a three-layered neural network with an indicated path is in Figure 2 which, besides the input and output layers, is composed of two hidden layers

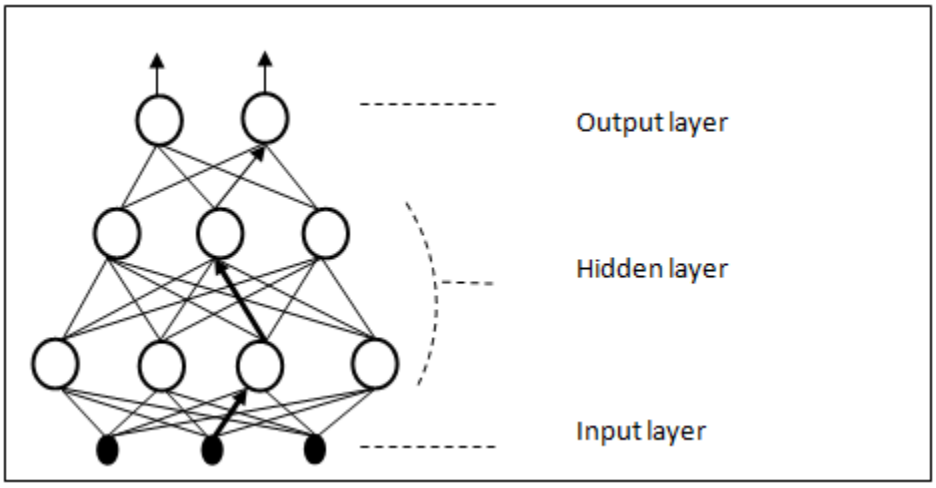


Figure2: Example of architecture of multilayered (three-layered) neural network.

The computational dynamics also determines the function of particular neurons whose formal form (mathematical formula) is usually the same for all (non-input) neurons in the network (homogeneous neural network) which has been inspired by a biological neuron operation. However, in neural network models, various neuron functions are in general use, which may not correspond to any neurophysiological pattern, but they are designed only by mathematical invention or even motivated by other theories (e.g. physics). On the other hand, sometimes the excitation level corresponds formally to the distance between the input and respective weight vector, etc. In addition, the transfer function is often approximated by a continuous (or differentiable) activation function or replaced by a completely different

function. The classification of ANN on the basis of computational dynamics is presented in Figure3

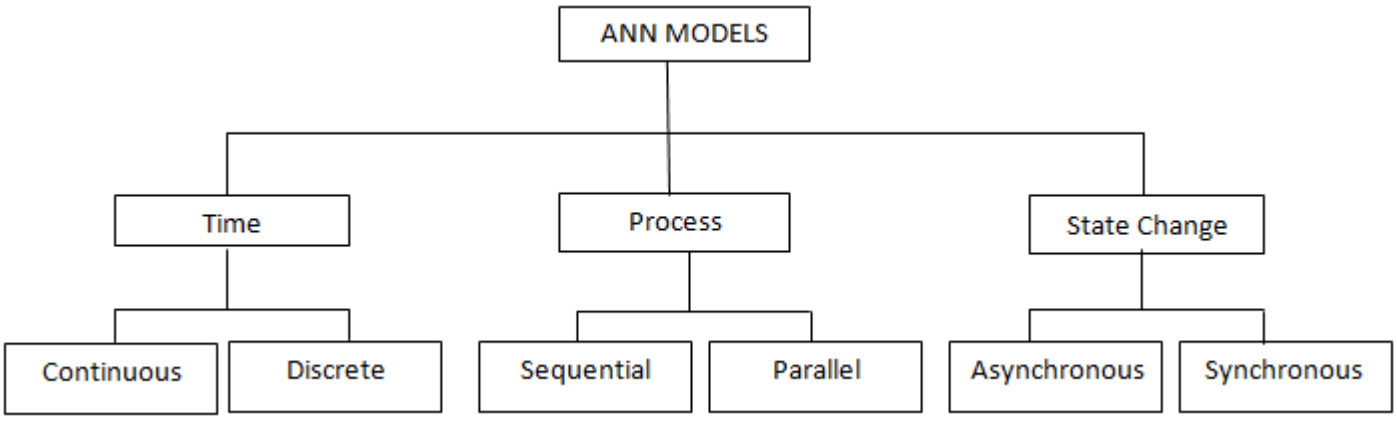


Figure3: Classification of neural network models according to computational dynamics.

IV. BIOINFORMATICS

We all know this is the field of statistics & computer science to explore Bioinformatics is a promising and innovative research field in 21st century. Despite of a high number of techniques specifically dedicated to bioinformatics problems as well as many successful applications, we are in the beginning of a process to massively integrate the aspects and experiences in the different core subjects such as biology, medicine, computer science, engineering, chemistry, physics, and mathematics. Recently the use of soft computing tools for solving bioinformatics problems have been gaining the attention of researchers because of their ability to handle imprecision, uncertainty in large and complex search spaces.

Advancement in soft computing techniques demonstrates the high standards of technology, algorithms, and tools in bioinformatics for dedicated purposes such as reliable and parallel genome sequencing, fast sequence comparison, search in databases, automated gene identification, efficient modelling and storage of heterogeneous data, etc. The basic problems in bioinformatics like protein structure prediction, multiple alignment, phylogenetic inference etc. are mostly NP-hard in nature. For all these problems, soft computing offers on promising approach to achieve efficient and reliable heuristic solution. On the other side the continuous development of high quality biotechnology, e.g. micro-array techniques and mass spectrometry, which provide complex patterns for the direct characterization of cell processes, offers further promising opportunities for advanced research in bioinformatics. So bioinformatics must cross the border towards a massive integration of the aspects and experience in the different core subjects like computer science and statistics etc. for an integrated understanding of relevant processes in systems biology. This puts new challenges not only on appropriate data storage, visualization, and retrieval of heterogeneous information, but also on soft computing methods and tools used in this context, which must adequately process and integrate heterogeneous information into a global picture. The molecular operation of Bioinformatics for achievement of soft computational approach in Neural Network to be understood with the following chart in Figure 4 who explains the summary of complete process of bioinformatics in neural.

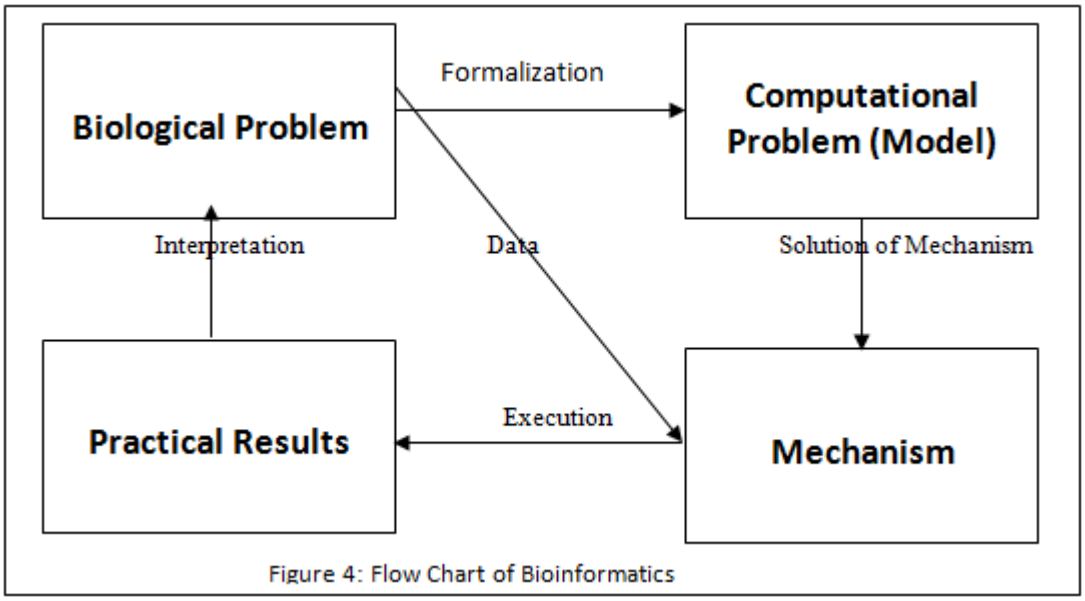


Figure 4: Flow Chart of Bioinformatics

3.1 **OBJECTIVE OF BIOINFORMATICS**

The basic objective of Bioinformatics is varies on the basis of area where we move to use. The global objectives of bioinformatics are as follows:

- To organize data in a way that allows researchers to access existing information and to submit new entries when they are produced.
- To develop tools and resources that aid in the analysis and management of data.
- To use this data to analyze and interpret the results in a biologically meaningful manner.
- To help researchers in the pharmaceutical industry in understanding the protein structures
- To make the drug design easy and integrate the soft comp using appropriate to bioinformatics

3.2 SCOPE OF BIOINFORMATICS

In this paper bioinformatics to be used for mathematical model is neural network for soft computing approach with the help of c#. Scope of bioinformatics involves the study of genes, proteins, nucleic acid structure prediction, and molecular design with docking. A broad classification of the various bioinformatics tasks is given as follows. The same will be also view in flow chart.

- Alignment and comparison of DNA, RNA, and protein sequences.
- Gene mapping on chromosomes gene finds and promoter identification from DNA sequences.
- Interpretation of gene expression and micro-array data.
- Gene regulatory network identification & analysis.
- Construction of phylogenetic trees for studying evolutionary relationship.
- DNA & RNA structure prediction.
- Protein structure prediction and classification with Molecular design and molecular docking.

The complete scope at present and coming time of bioinformatics(shown in shadow part in Figure5) can be used as presented concrete diagram in multidimensional approaches.

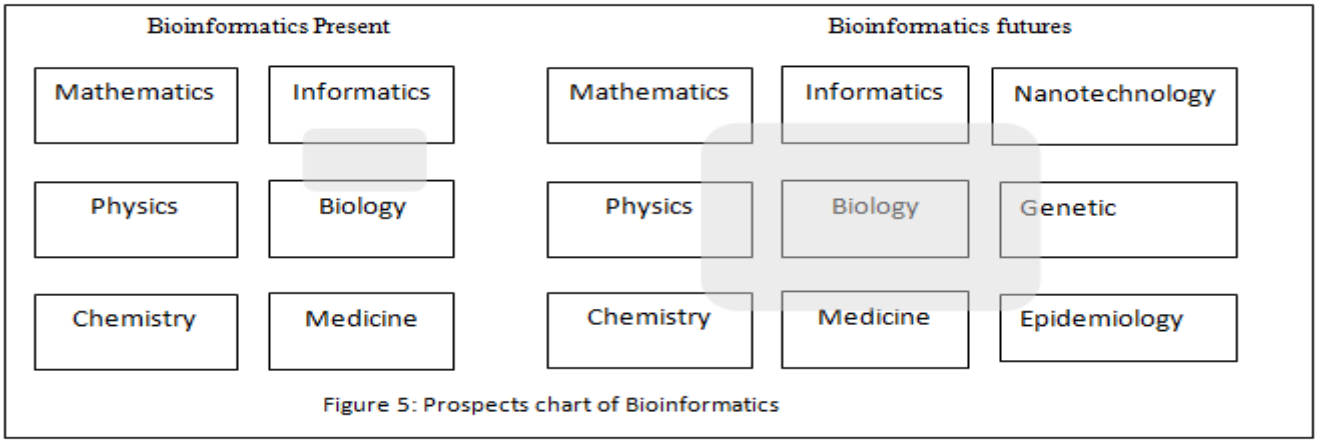


Figure 5: Prospects chart of Bioinformatics

So, Bioinformatics has found its applications in many areas. It helps in providing practical tools to explore proteins and DNA in number of other ways. Bio-computing is useful in recognition techniques to detect similarity between sequences and hence to interrelate structures and functions. Another important application of bioinformatics is the direct prediction of protein 3-Dimensional structure from the linear amino acid sequence. It also simplifies the problem of understanding complex genomes by analyzing simple organisms and then applying the same principles to more complicated ones. This would result in identifying potential drug targets by checking homologies of essential microbial proteins. Bioinformatics is useful in designing drugs.

3.3 ALGORITHMIC DEVELOPMENT OF BIOINFORMATICS

Basically algorithms give the basic idea to develop the computational programming & judge the situation of the Bioinformatics technique. Attempted to be explore "How bioinformatics can be used in modern era for development of new concept with the help of soft-computing C#" for well behaving human activities i.e. Neural network and genetic concept. Usually a biological problem can be transformed into a computational problem in a number of ways that feature different levels of accuracy and complexity. Highly accurate models often result in intractable computational problems while less accurate models may produce meaningless results. Goal of Bioinformatics Algorithm "to maintain an acceptable level of accuracy keeping the computational problem effectively solvable"

The following are some of the most important algorithmic trends in bioinformatics related to Biological Science:-

- Finding similarities among strings (such as proteins of different organisms).
- Detecting certain patterns within strings (such as genes, α-helices).
- Finding similarities among parts of spatial structures (such as **motifs**).
- Constructing trees (called polygenetic trees expressing the evolution of organisms whose
- DNA or proteins are currently known.
- Classifying new data according to previously clustered sets of annotated data.
- Reasoning about microarray data and the corresponding behaviour of pathways.

The first three trends can be viewed as instances of pattern matching. However, pattern matching in biology differs from its counterpart in computer science. DNA strings contain millions of symbols, and small local differences may be tolerated. The pattern itself may not be exactly known, because it may involve inserted, deleted, or replacement symbols. Regular expressions are useful for specifying a multitude of patterns and are ubiquitous in bioinformatics. However, what biologists really need is to be able to infer these regular expressions from typical sequences and establish the likelihood of the patterns being detected in new sequences.

The discussion suggests that both optimization and probabilistic approaches are necessary for developing biology-oriented pattern-matching algorithms. Which are like "Dynamic programming technique ,Patter matter approach, Multiple alignments, Approximate methods, Support vector machines(SVM).In the intermediate time of twentieth century a dynamic programming technique was devised to match two strings, taking into account the costs of insertions, deletions, and substitutions called global pair-wise alignment. This technique was subsequently extended to consider local alignments and today, both methods are often used in bioinformatics. However, dynamic programming is time consuming (it involves quadratic complexity) and therefore cannot be applied in a practical way to strings with hundreds of thousands of symbols. A remarkable bioinformatics development at the end of the twentieth century a pattern-matching approach called BLAST, or the Basic Local Alignment Search Tool, that mimics the behaviour of the dynamic programming approach and efficiently yields good results using heuristic based approach. It is fair to say that BLAST is the most frequently used tool for searching sequences in genomic databases. Another widely used and effective technique is multiple alignments, which helps align several sequences of symbols, so identical symbols are properly lined up vertically, with gaps allowed within symbols. The sequences may represent variants of the same proteins in various species. But the goal is to find conserved parts of the proteins that are unchanged during evolution. Finding conserved parts of proteins also provides hints about a protein's possible function. Methods for multiple alignments are based on dynamic programming techniques developed for pair wise alignment. After aligning multiple genomic or protein sequences, biologists usually depict trees representing the degree of similarity among the sequences being studied. Depicting evolutionary trees is in itself a domain within bioinformatics called polygenetic trees. The problem of matching spatial structures can be viewed as a combination of computational geometry and computer graphics. Approximate methods are often required to find the longest linkage that is common in two 3D structures. Bioinformatics involves the pervasive use of searches in genomic databases that often yield very large sets of long sequences. Such searches are often performed automatically by scripting to download massive amounts of genomic data from a number of Web sites. An approach commonly used in bioinformatics is: Given a human-annotated list of strings with boundaries specifying meaningful substrings—the learning set— now establish the corresponding likely boundaries for a new string (examples in bioinformatics involve finding genes and identifying the components of proteins). Solutions to these problems are being explored through approaches from machine learning, neural networks, genetic algorithms, and clustering. In the same time clustering technique called support vector machines (SVM) has had considerable success in biology. Classification and machine learning have been studied extensively in artificial intelligence to sort out new data based on a human-annotated set of examples. Two other algorithmic trends relevant to this discussion are related to micro-arrays and biologists' interest in computational linguistics. Recall that the main goal of analyzing micro-array data is to establish relationships among gene behaviour, possible protein interactions, and the effects of a cell's environment. From a computer science perspective, that goal is amounted to the generation of parts of a program (flowchart) from data.

Information about the relationships among genes is often buried in countless articles describing the results of biological experiments. In the case of protein interaction, pharmaceutical companies have teams whose task is to search the available literature and "manually" detect phrases of interest. Efforts have been made to computerize these searches. Their implementation requires expertise in biology, computational linguistics and heuristic based methodologies. Based on the above discussion, it is mandatory to have a machine learning/soft computing based approach for various tasks in bioinformatics.

The overview can be seen as diagrammatical presentation (Figure 6) of time bounded manner of development of bioinformatics which are as :

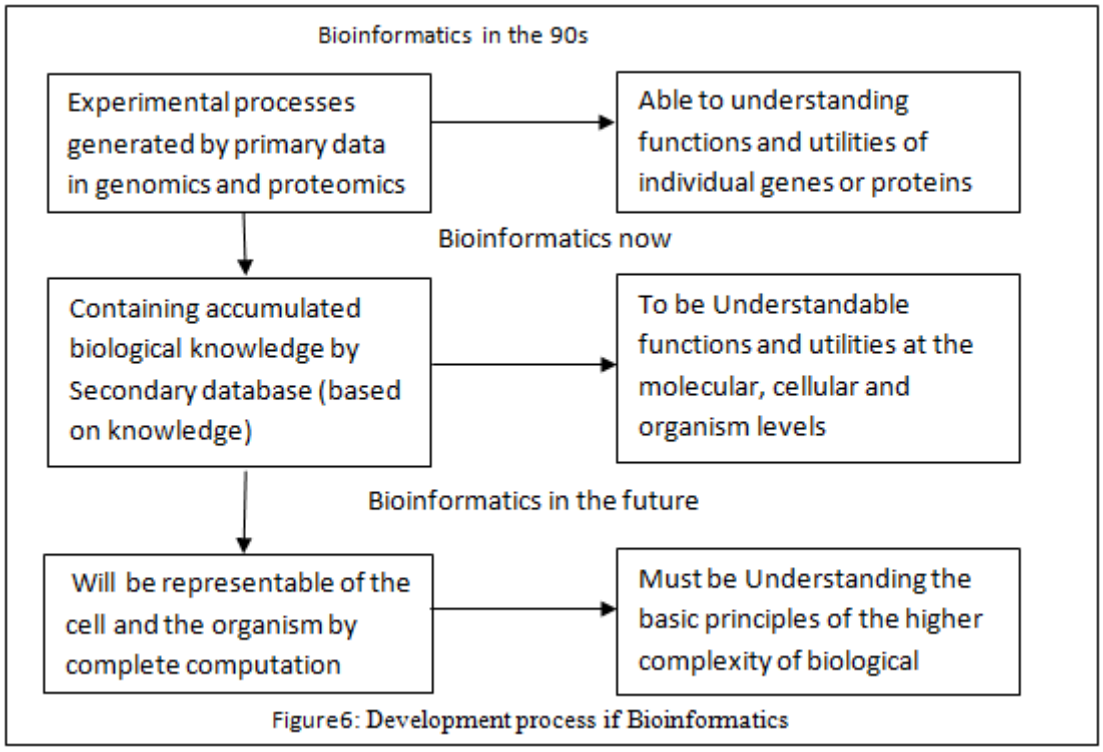


Figure6: Development process if Bioinformatics

3.4 BOTTLENECKS OF BIOINFORMATICS

The basic problems in bioinformatics like protein structure prediction, multiple alignment, polygenetic inference etc. are mostly NP-hard in nature. For all these problems, soft computing offers on promising approach to achieve efficient and reliable heuristic solution. The basic limitation of bioinformatics in neural network can be understood with the help of following diagrammatic presentation in figure 7.

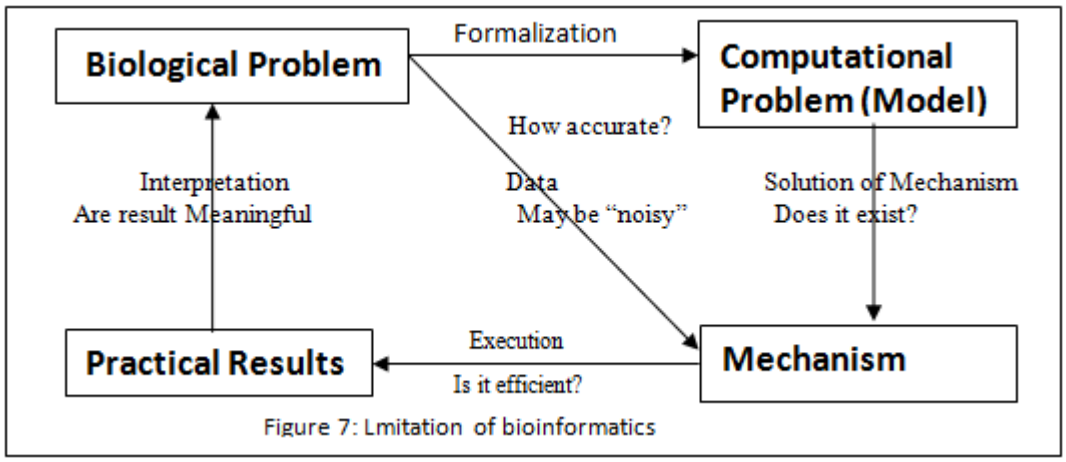


Figure 7: Lmitation of bioinformatics

V. COMPUTATION AND TYPE OF NEURAL NETWORK

The provision of embedding neural networks into software applications can enable variety of Artificial Intelligence systems for individual users as well as organizations. Previously, software implementation of neural networks remained limited to only simulations or application specific solutions & tightly coupled solutions end up in monolithic systems as well as non reusable programming efforts. We adapt component based software engineering approach to effortlessly integrate neural network models into AI systems in an application independent way. This paper presents componentization of three famous neural network models (1) Multi Layer Perceptions (2) Learning Vector Quantization and (3) Adaptive Resonance Theory family of networks. Main premise of this paper states that the integration and deployment of neural networks into production environments in an application independent way can be greatly facilitated by employing component based software engineering approach. The CompoNet (Componentized-neural-Networks) facilitates development of neural network based software systems by designing and implementing neural networks as software components. These components can be reused effortlessly across different application and, thus, shipping of trained models from simulation to production environment is possible with minimal programming effort. Essentially CompoNet functions as binary unit of composition into larger systems. It can encapsulate arbitrary size and structure of underlying neural network models and exposes programmatic interfaces in order to i) embed a neural network into external software application and ii) persist neural network on permanent storage and restore it later. For the sake of realization of the concept and in order to provide variety of choices to developer community, we componentized three famous neural network models.

1. **The Multi Layer Perceptions** (MLP) is presented in (Figure 2,8,9) general model of neural network analysis. This network has an input layer (on the left) with three neurons, one hidden layer (in the middle) with three neurons and an output layer (on the right) with three neurons.
2. **Learning Vector Quantization (LVQ)** contains an input layer a kohonen layer(which leads and perform the classification) and an output layer. The input layer contains one node for each input feature. The kohonen layer contains equal number of nodes for each class, in the output layer each output node represent a particular class.
3. **Adaptive Resonance Theory (ART)** networks represent a family of neural networks which is based on resonance refers to resonant state of neural network in which a category prototype vector matches close enough to the current input vector.

These entire three models is on the basis of structural, development and working process of neural network having three basic part i.e. input, output and hidden layers is presented in general neural network(Figure 2,8,9) which is presented below.

4.1 ARTIFICIAL NEURAL NETWORK

The terminology of artificial neural networks has developed from a biological model of the brain. A neural network consists of a set of connected cells: the neurons. The neurons receive impulses from either input cells or other neurons and perform some kind of transformation off the input and transmit the outcome to other neurons or to output cells. The neural networks are built form layers of neurons connected so that one layer receives input from the preceding layer of neurons and pass the output on to the subsequent layer.

A neuron is a real function of the input vector $(y_1,...,y_k)$. The output is obtained as obtained as $\int(x_j) = f\left(a_j + \sum_{i=1}^{k} w_{ij} y_j\right)$, where f is a function, typically the sigmoid (logistic or tangent hyperbolic) function. A graphical presentation of neuron is given in figure below. Mathematically a Multi-Layer Perceptions network is a function consisting of compositions of weighted sums of the functions corresponding to the neurons.

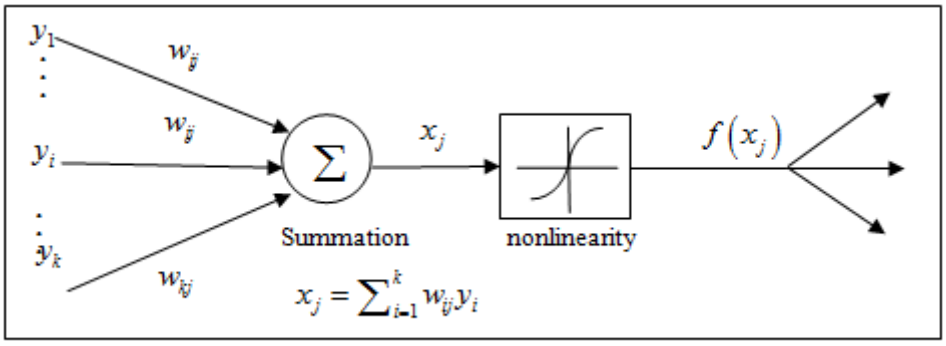


Figure 8: Flow chart of General Mathematical Function of ANN

That means Artificial Neural Networks are biologically inspired clusters of highly connected parallel processing nodes. Their remarkable ability to tolerate noise and adapt to unseen situations, wide scale application of neural networks has been carried out in many areas such as manufacturing , intelligent control systems power engineering , pattern recognition ,speech recognition and Ubiquitous Computing Systems to name a few. Mostly, the practical implementation considerations for neural

network based AI systems are studied from hardware point of view. However, several software applications demand incorporation of neural networks as well. This requirement holds for building hybrid intelligent systems based on combination of neural networks and knowledge-based system. Moreover, AI researchers often need to verify and test their application of neural networks on novel problems in real life but software environment. Advanced software environments for neural network development and training are limited only to simulation of neural networks in restricted simulation environment e.g. C #, **MATLAB Neural Network Toolbox etc.** Therefore, In order to train a model and verify the results, AI researchers have to bring training and test data from real life environment into a simulation environment. After the neural network model is optimized and trained, the integration of this model into real-life environment remains a secondary task and left to domain specific software programs which results in non-reusable effort. The basic architecture of ANN is presented below when a system with "n" inputs and "m" outputs. Each ANN Target- output corresponds to the one of m classes, and it is activated by the proper values of the corresponding inputs that are associated to its class.

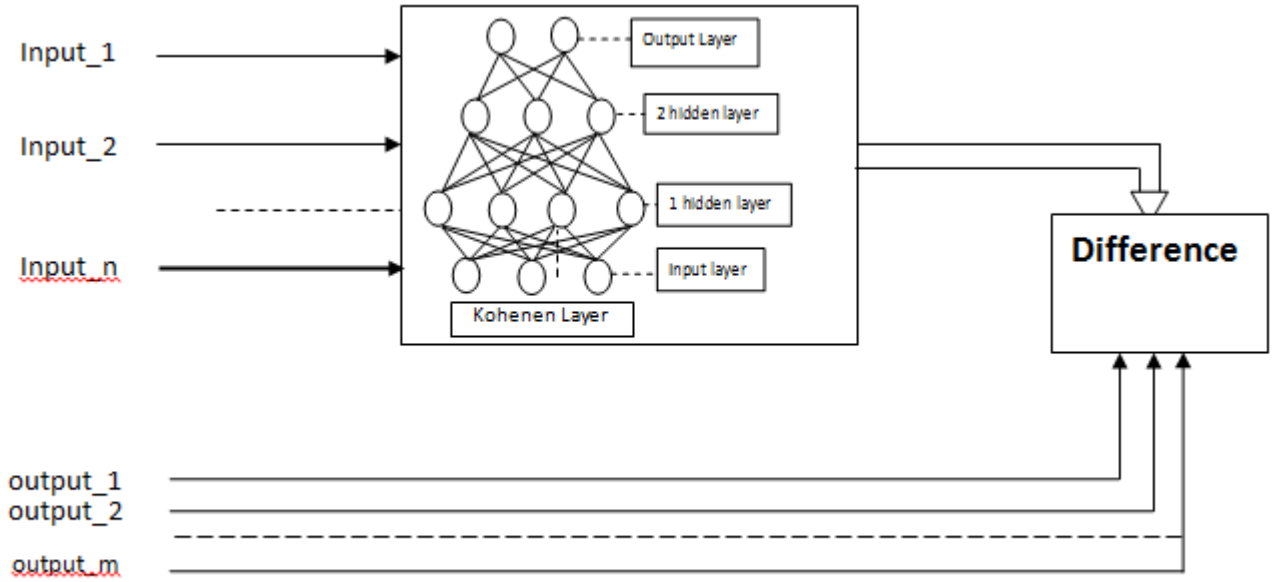


Figure 9: General Model of Neural Network

4.2 ADVANTAGE OF ARTIFICIAL NEURAL NETWORK

An artificial neural network (ANN) is the overcome of SVM (Support Vector Machines) which programmed the computational model having aims to replicate the neural structure and functioning of the human brain. It is made up of an interconnected structure of artificially produced neurons that function as pathways for data transfer. Artificial neural networks are flexible and adaptive, learning and adjusting with each different internal or external simultaneously. Artificial neural networks are used in sequence and pattern recognition systems, data processing, robotics and modelling. There are different types of neural networks, including feed-forward neural network, radial basis function (RBF).

Flexibility: Artificial neural networks have the ability to generalize and learn. They acquire knowledge from their surroundings by adapting to internal and external parameters. The network learns from examples and adapts to situations based on its findings. It generalizes knowledge to produce adequate responses to unknown situations. Artificial neural networks solve complex problems that are difficult to manage by approximation.

Non-Linearity: A computational neuron can produce a linear or a non-linear answer. A non-linear artificial network is made by the interconnection of non-linear neurons. Non-linear systems have inputs that are not proportional to the outputs. This function allows the network to efficiently acquire knowledge through learning. This is a distinct advantage over a traditionally linear network that is inadequate when it comes to modeling non-linear data.

Greater Fault Tolerance: An artificial neuron network is capable of greater fault tolerance than a traditional network. The network is able to regenerate a fault in any of its components without the loss of stored data. It uses instances and examples from the past to reassemble the functioning of a damaged node or other network constituent.

Adaptive Learning: An artificial neuron network is based around the concept of abstract learning. Three learning paradigms function to equip the network for adaptive learning. These are reinforcement learning, unsupervised learning and supervised learning. Neuron networks can be trained via specialized algorithms including non-parametric methods, expectation maximization, simulated annealing and evolutionary methods. The neurons of an artificial neuron network are flexible enough to be attuned to various input signal patters and acclimatize to a diverse array of unknown situations. They are constantly accepting and replacing previously learned information, keeping their repository of problem solving techniques updated. Partial destruction of a network leads to the corresponding degradation of performance. However, some

network capabilities may be retained even with major network damage.

Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.

Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.

Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

Massive parallelism: Massively parallel systems is used in Artificial neural nets (ANNs) with large numbers of interconnected simple processors

4.3 APPLICATION OF ARTIFICIAL NEURAL NETWORK

Neural networks have been successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology, physics and biology. The excitement stems from the fact that these networks are attempts to model the capabilities of the human brain. From a statistical perspective neural networks are interesting because of their potential use in prediction and classification problems. Artificial neural networks (ANNS) are non-linear data driven self adaptive approach as opposed to the traditional model based methods. They are powerful tools for modelling, especially when the underlying data is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. After training, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. Thus they are ideally suited for the modelling of agricultural data which are known to be complex and often non-linear. A very important feature of these networks is their adaptive nature, where "learning by example" replaces "programming" in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available. These networks are "neural" in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. In fact majority of the network are more closely related to traditional mathematical and/or statistical models such as non-parametric pattern classifiers, clustering algorithms, nonlinear filters, and statistical regression models than they are to neurobiology models. Neural Networks (NNs) have been used for a wide variety of applications where statistical methods are traditionally employed. They have been used in classification problems, such as identifying underwater sonar currents, recognizing speech, and predicting the secondary structure of globular proteins. In time-series applications, NNs have been used in predicating stock market performance. As statisticians or users of statistics, these problems are normally solved through classical statistical methods, such as discriminant analysis, logistic regression, logistic regression, Bayes analysis, multiple regression, and ARIMA time-series models. It is, therefore, time to recognize neural networks as a powerful tool for data analysis.

Here we have using c# for soft component in model of Neural Network. The application of artificial neural network which is commonly used in real world is as follows:-

- Predict the translation initiation sites in DNA sequences and Study human TAP transporter
- Explain the theory of neural networks using applications in biology
- Predict immunologically interesting peptides by combining an evolutionary algorithm
- Carry out pattern classification and signal processing successfully in bioinformatics; in fact, a large number of applications of neural network can be found in this area.
- Perform protein sequence classification; neural networks are applied to protein sequence classification by extracting features from protein data
- Protein secondary structure prediction and Analyze the gene expression patterns as an alternative to hierarchical clusters
- Gene expression can even be analyzed using a single layer neural network
- Protein fold recognition using ANN and SVM

VI. RELATIONSHIPS AMONG NEURAL NETWORK, GENETIC ALGORITHM AND BIOINFORMATICS

The combined effort of Genetic programming-optimized NN (GPNN) in an attempt to improve upon the trial-and-error process of choosing an optimal architecture for a pure feed-forward Bioinformatics Programming NN (BPNN). The GPNN optimizes the inputs from a larger pool of variables, the weights, and the connectivity of the network including the number of hidden layers and the number of nodes in the hidden layer. The genetic algorithm is a population of strings (called chromosomes or the genotype of the genome), which encode individuals, creatures, or phenotypes solution to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached. Genetic algorithms a biologically inspired technology, are randomized search and optimization techniques guided by the principles of evolution and natural genetics. They are efficient, adaptive, and robust search processes, producing near optimal solutions, and have a large degree of implicit parallelism. Therefore, the application of GAs for solving certain problems of bioinformatics, which need optimization of computation requirements, and robust, fast and close approximate solutions, appears to be appropriate and natural. Moreover, the errors

generated in experiments with bioinformatics data can be handled with the robust characteristics of GAs. To some extent, such errors may be regarded as contributing to genetic diversity, a desirable property. The problem of integrating GAs and bioinformatics constitutes a new research area. GAs is executed iteratively on a set of coded solutions, called population, with three basic operators: selection/reproduction, crossover, and mutation. They use only the payoff (objective function) information and probabilistic transition rules for moving to the next iteration. Of all the evolutionarily inspired approaches, Gas seem particularly suited to implementation using DNA, protein, and other bioinformatics tasks. This is because GAs is generally based on manipulating populations of bit-strings using both crossover and point-wise mutation.

5.1 SIGNIFICANCE AND APPLICATION OF GENETIC ALGORITHM IN BIOINFORMATICS

Genetic algorithms find application in bioinformatics, polygenetic, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields. The most suitable applications of GAs in bioinformatics are:

- Alignment and comparison of DNA, RNA, and protein sequences.
- Gene mappings in chromosomes.
- Gene finding and promoter identification from DNA sequences.
- Interpretation of gene expression and micro array data.
- Gene regulatory network identification.
- Construction of polygenetic tree for studying evolutionary relationship
- DNA structure prediction and RNA structure prediction.
- Protein structure prediction and clustering.
- Molecular design and molecular docking.

A typical genetic algorithm requires: A genetic representation of the solution domain, and a fitness function to evaluate the solution domain. A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used. Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions (usually randomly) and then to improve it through repetitive application of the mutation, crossover, inversion and selection

5.2 ADVANTAGE OF GENETIC ALGORITHM IN BIOINFORMATICS

Several tasks in bioinformatics involve optimization of different criteria (such as energy, alignment score, and overlap strength), thereby making the application of Gas more natural and appropriate.

- Problems of bioinformatics seldom need the exact optimum solution; rather, they require robust, fast, and close approximate solutions, which GAs are known to provide efficiently.
- GAs can process, in parallel, populations billions times larger than is usual for conventional computation. The usual expectation is that larger populations can sustain larger ranges of genetic variation, and thus can generate high-fitness individuals in fewer generations.
- Laboratory operations on DNA inherently involve errors. These are more tolerable in executing evolutionary algorithms than in executing deterministic algorithms. (To some extent, errors may be regarded as contributing to genetic diversity—a desirable property.)

5.3 GENERAL NURAL MODEL FOR BIOINFORMATICS

An Artificial Neural Network (ANN) is an information processing model that is able to capture and represent complex input-output relationships. The motivation the development of the ANN technique came from a desire for an intelligent artificial system that could process information in the same way the human brain. These brains collect the information through connected neurons with the help of bioinformatics. The general concrete and basic model of bioinformatics can be presented here. This model may also be computed with the help of soft-computing programming of C#. Its novel structure is represented as multiple layers of simple processing elements, operating in parallel to solve specific problems. ANNs resemble human brain in two respects: learning process and storing experiential knowledge. An artificial neural network learns and classifies a problem through repeated adjustments of the connecting weights between the elements. In other words, an ANN learns from examples and generalizes the learning beyond the examples supplied. Artificial neural network applications have recently received considerable attention. The methodology of modelling, or estimation, is somewhat comparable to statistical modelling. Neural networks should not, however, be heralded as a substitute for statistical modelling, but rather as a complementary effort (without the restrictive assumption of a particular statistical model) or an alternative approach to fitting non-linear data. A typical neural network (shown in Figure) is composed of input units $X= X_1, X_2,..... X_n$ corresponding to independent variables, a hidden layer known as the first layer, and an output layer (second layer) whose output units $Y=Y_1,Y_2,....Y_m$

corresponds to dependent variables (expected number of accidents per time period)

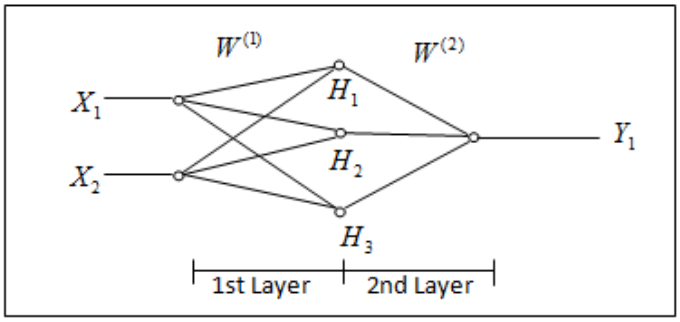


Figure 10: General ANN model with weight

In between are hidden units H1, H2 … corresponding to intermediate variables. These interact by means of weight matrices W (1) and W (2) with adjustable weights. The all the hidden units(H_K) corresponded with the weights and sharing the information with input(X_i)(i=1,2,….n) and output(Y_j)(j=1,2,….m). In this case no of hidden units depends on input as per informative flow of output. The values of the hidden units are obtained from the formulas:

$$H_j = f\left(\sum_k w_{jk}^{(1)} x_k\right) \quad \text{.....(A)}$$

$$Y_i = f\left(\sum_k w_{jk}^{(2)} H_j\right) \quad \text{.....(B)}$$

In One multiplies the first weight matrix by the input vector X = (X1, X2,...) and then applies an activation function f to each component of the result. Likewise the values of the output units are obtained by applying the second weight matrix to the vector H = (H_1, H_2,....,H_K) of hidden unit values, and then applying the activation function f to each component of the result. In this way one obtains an output vector Y= (Y1, Y2,...).The activation function f is typically of sigmoid form and may be a logistic function, hyperbolic tangent, etc.:

$$f(u) = \frac{1}{1+e^{-u}} \quad \text{.....(C)}$$

$$f(u) = \frac{e^{x} - e^{-u}}{e^{x} + e^{-u}}$$

Usually the activation function is taken to be the same for all components but it need not be. Values of W (1) and W (2) are assumed at the initial iteration. The accuracy of the estimated output is improved by an iterative learning process in which the outputs for various input vectors are compared with targets (observed frequency of accidents) and an average error term E is computed(see section 5.4):

$$E = \frac{\sum_{n=1}^{N} \left(Y^{(k)} - T^{(n)}\right)^2}{N} \quad \text{----------(D)}$$

Here

N = Number of highway sites or observations
Y (n) = Estimated number of accidents at site n for n = 1, 2... N
T (n) = Observed number of accidents at site n for n = 1, 2... N.

After one pass through all observations (the training set), a gradient descent method may be used to calculate improved values of the weights W(1) and W(2), values that make E smaller. After revaluation of the weights with the gradient descent method, successive passes can be made and the weights further adjusted until the error is reduced to a satisfactory level. The computation thus has two modes, the mapping mode, in which outputs are computed, and the learning mode, in which weights are adjusted to minimize E. Although the method may not necessarily converge to a global minimum, it generally gets quite close to one if an adequate number of hidden units are employed. The most delicate part of neural network modelling is generalization, the development of a model that is reliable in predicting future accidents. Over fitting (i.e., getting weights for which E is so small on the training set that even random variation is accounted for) can be minimized. The complete process can be summarised on the above mathematical model.

5.4 Soft Computing Approaches of Neural Network by C#

The above model can be computed with the help of programming in C# to easy understand of modified general neural network model for bioinformatics with hidden layers who can pass the information from input to output layers in easy manner. The above mathematical function and equation (C) &(D) to be computed by C#.

```
public Double observationInput(Int32 EstimatedNoAccidents, Int32 ObservedNoAccidents,
Int32 NohighwaySite)
{
            Double finalval = 0;
            Double finalvalallCal = 0;
            for (int i = 1; i == NohighwaySite; i++)
            {
                finalval += (EstimatedNoAccidents ^ i - ObservedNoAccidents ^ i) *
(EstimatedNoAccidents ^ i - ObservedNoAccidents ^ i);
            }

            return finalvalallCal = finalval / NohighwaySite;
}
public Double logisticMethod(double logisticInput)
{
            Double result = 1 / (1 + Math.Exp(logisticInput));
            return result;
}
public Double HyperbolicM(double hyperInput)
{
          Double result =  (Math.Exp(hyperInput) - Math.Exp(-hyperInput)) /
                           (Math.Exp(hyperInput) + Math.Exp(-hyperInput));
          return result;
}
```

VII. CONCLUSION AND FUTURE SCOPE

The demand of interdisciplinary subject has been increasing rapidly with its great significances and complete utilization in the sense of application point of view. The Bioinformatics is one of them, its involvement of other sciences like neural network, genetic for computation holds great promise; this century's major research and development efforts will likely be in the biological and health sciences. Computer science diversifies their offerings, which can gain through early entry into bioinformatics. Even using minimal resources and maximum utilization of bioinformatics in neural network with the computational process such efforts are wise and signified in recent time. Still unclear is whether bioinformatics will eventually become an integral part of computer science or will develop into an independent application. Regardless of the outcome, computer scientists are sure to benefit from being active and assertive partners with biologists. The tools like neural network and genetic algorithm gives the powerful emphasis on bioinformatics to computing the combined effect for development model and exploring it with development of soft impact by C#. The multidimensional mathematical model can be developed for together working process of neural, genetic with bioinformatics in futures. The computational programmings for same model to be also work out in coming time.

REFERENCES

[1] E Balagurusamy, "Programming in c#", 2008, Second edition

[2] Christian Nagel, Bill Evjen, Jay Glynn, Morgan Skinner, Karli Watson, "Professional C# 2008", 2008

[3] K. Chenand, L. Kurgan (2007), "PFRES: protein fold classification by using evolutionary information and predicted secondary structure", Bioinformatics, 23(21): 2843 – 2850

[4] A. Narayanan, E. Keedwell, and B. Olsson (2003), "Artificial Intelligence Techniques for Bioinformatics", Applied Bioinformatics, Vol.1, No. 4, pp. 191-222.

AUTHORS

First Author – Sarvesh Kumar, Research Scholar, Dept. of Statistics & Computer Applications, T.M.B. University, Bhagalpur (India), Email:-sarveshmcasoft@gmail.com
Second Author – Dr. Rajeshwar Prasad Singh, Bhagalpur college of engineering, Sabour, Bhagalpur, India, Email: - dr.rajeshwarprasad@gmail.com
Third Author – DR. Akshoy Kumar Mishra, Post graduate of Statistics and computer applications , T.M.B. University, Bhagalpur, India, Email: - akmishra.tmbu@gmail.com
Fourh Author – Hemant Kumar, Directorate of Economics & Statistics (Planning Cadre), GNCT Delhi, India, Email: - iitd.hement@gmail.com