IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

An Analysis of BPA and ANN Applications used in Enhancement of Software Reliability

[1]Ms. Mahima, Research Scholar, Department of Computer Science & Engineering, Baba Mashnath, University, Rohtak.

[2]Dr, Mukesh Singla, Professor and Supervisor, Department of Computer Science & Engineering, Baba Mashnath,University, Rohtak.

Abstract

Neural networks are a computational metaphor inspired studies of the brain and nervous system in biological organisms. They are highly idealized mathematical models of how understand the essence of these simple nervous systems. Reliability in the general engineering sense, is the probability. It gives component or system in a define environment will operate correctly for a specified period of time. An important issue in developing such software systems is to produce high quality software system that satisfies user requirements. Software reliability models specify some reasonable form for this distribution, and are fitted to data from a software project. Once a model demonstrates a good fit to the available data, it can be used to determine the current reliability of the software, and predict the reliability of the software at future times.The description and demonstration of the several types of neural networks are given, applications of neural networks like ANNs in medicine are described, and the detailed historical backgrounds are provided. The relationship between the artificial and the actual thing is also thoroughly investigated and explained. Finally, the involved mathematical models are shown and explained.

Introduction

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the noble structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data

classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is applicable in ANNs as well.

History

First Attempt: The research article of McCulloch and Pitts which was published in 1943 was taken as the beginning of Neurocomputing. It demonstrated that even simple types of neural networks can perform any arithmetic or logical function. It was widely read and had massive influence. Other researchers, Norbert Wiener and von Neumann, wrote a book and research paper in which they suggested that the research into the design of brain-like or brain-inspired computers might be interesting. In 1949 Hebb wrote a book entitled 'The Organization of Behaviour' which introduced the idea that classical psychological conditioning is ubiquitous in animals as it is a property of individual neurons. Not that the idea was exclusive, but Hebb took it further than anyone before him had by proposing a specific learning law for the synapses of neurons. Hebb then used this learning law to build a qualitative explanation of some experimental results from psychology. Even though there were many other people examining the issues regarding the neurocomputing in the 1940s and early 1950s, their work was more like setting the stage for later developments than actually causing those developments. The construction of first neurocomputer (the Snark) by Marvin Minsky in 1951 happened in this era. The Snark did operate successfully from a technical stand point but it never actually carried out any particularly interesting information processing functions.

Promising & Emerging Technology: The 1950s and 1960s: The First Golden Age of Neural Networks Mark I perceptron, the first successful neuro-computer was developed during 1957 and 1958 by Frank Rosenblatt, Charles Wightman, and others. That is why Rosenblatt is known as the founder of Neurocomputing. He was primarily interested in pattern recognition. Besides inventing the perceptron, Rosenblatt also wrote an early book on Neurocomputing, 'Principles of Neurodynamics'. Slightly later than Rosenblatt, but similar to him, was Bernard Widrow. Widrow, working with his graduate students (most notably Marcian E. "Ted" Hoff, who later went on to invent the microprocessor) developed a different type of neural network processing element called ADALINE, which was equipped with a powerful new learning law which, unlike the perceptron leaning law, is still in widespread use. Widrow and his students applied the ADALINE successfully to a large number of toy problems, and produced several films of their successes. Besides Rosenblatt and Widrow, there were a number of other people during the late 1950s and early 1960s who had substantial success in the development of neural network architectures and implementation concepts. Notwithstanding the considerable success of these early Neurocomputing researchers, the field suffered from two glaringly obvious problems. First, the majority of researchers approached the subject from a qualitative and experimental point of view. This experimental emphasis resulted in a significant lack of rigor and a looseness of thought that bothered many established scientists and engineers who established the field. Second, an unfortunate large fraction of neural networks researchers were carried away by their enthusiasm in their statements and their writings. For example, there were widely publicized predictions that artificial brains were just a few years away from development, and other incredible statements. Besides the hype and general lack of rigor, by the mid 1960s

researchers had run out of good ideas. The last chapter of this era was a campaign led by Marvin Minsky and Seymour Papert to discredit neural network research and divert neural network research funding to the field of "Artificial Intelligence". The campaign was waged by the means of personal persuasion by Minsky and Papert and their allies, as well as by limited circulation of unpublished technical manuscript (which was further published in 1969 by Minsky and Papert as the book Perceptrons).

Period of Frustration & Disrepute: Despite of Minsky and Papert's exposition of the limitations of perceptrons, research on neural network continued. A great deal of neural network research went on under the headings of adaptive signal processing, pattern recognition, and biological modeling. In fact, many of the current leaders in the field began to publish their work during 1970s. Examples include Amari, Fukushima, Grossberg and Klopf and Gose. These people, along with few others who came in over the next 13 years, put the field of neural network on a firm floor and paved the way for the renovation of the field.

Re-Emergence: It was by the early 1980s that many Neuro-computing researchers became strong enough to begin submitting proposals to explore the development of neuro-computers and of neural network applications. In the years 1983–1986 John Hopfield, an established physicist of worldwide reputation had become interested in neural networks a few years earlier. Hopfield wrote two highly readable papers on neural networks in 1982 and 1984 and these, together with his many lectures all over the world, persuaded hundreds of highly qualified scientists, mathematicians, and technologists to join the emerging field of neural networks. In 1986, with the publication of the "PDP books" (Parallel Distributed Processing, Volumes I and II, edited by Rumelhart and McClelland), the field exploded. In 1987, the first open conference on neural networks in modern times, the IEEE International Conference on Neural Networks was held in San Diego, and the International Neural Network Society (INNS) was formed. In 1988 the INNS journal Neural Networks was founded, followed by Neural Computation in 1989 and the IEEE Transactions on Neural Networks in 1990.

Role of Neural Network in Software Reliability:

The reliability model till now been designed are based on the study of failure associated with the code and the environment where it is been implemented [20] .All the software reliability models are designed on the basis of execution time and calendar time. Execution time of any program is the time that is actually required or spent by the processor in executing the instruction of that program [21].Calendar time is referred as the elapsed time from start to end of program execution on a running computer.Neural network is a collection of fast processing and computing nodes called artificial neurons. These neurons are designed on the basis of study of the behavior of biological neuron. These neurons are connected in a specific manner that is layer like structure known as neural network architecture. Neural network is an output based computing technique. It is a technology generally been used for optimizing problem. ANN (Artificial Neural Network) software reliability models have recently aroused more research interest. Traditionally, both kinds of models only consider single fault detection

process (FDP) and data for analysis are only from FDP. However, while data from both FDP&FCP (fault correction process) are available, NHPP and ANN models can be extended into paired NHPP models and combined ANN models, providing more accurate predictions [3,6]. Generally speaking, data-driven approach is much less restrictive in assumptions compared to analytical approach.

Architecture Of Neural Network:

The artificial neural network is designed on the basis of the study of biological neural network. A human brain is responsible for all the action and reaction taken by human body. These actions are controlled by brain and are carried to different parts of the body by a fast processing node known as neurons. Similarly in case of artificial neurons also we have a fast processing node responsible for performing the computation and are known as Neurons [19]. A neural network is a collection one to multiple neurons which are arranged in a specific manner to perform the computation. The basic structure of a neuron is represented in the given below figure:

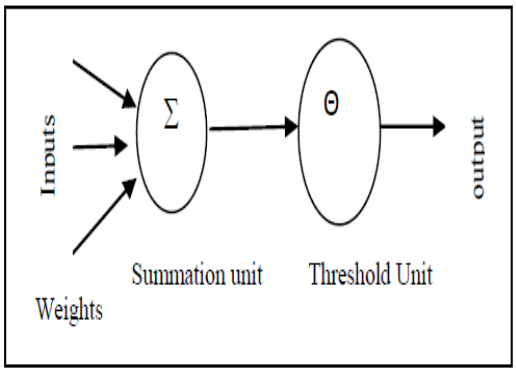

Figure1: Structure of a Neuron

The model is designed for a set of input values and corresponding desired output. Suppose consider that „I" is the set of input values such that I={i1,i2…….in}and we associate a variable parameter with it which controls the behavior of neurons. Let us consider weights associated with the input as the variable parameter such that W={w1,w2…….wn}.All the input along with the weight is passed through the summation unit where compute the net input which given by

$$Net_{input} = \sum_{i=1}^{n} IiWi \text{ where n = number of input to the network} \quad .....(1.11)$$

$$= i1*w1+i2*w2+i3*w3+.....+i_iwi$$

These neurons are arranged in different layers to represent a specific structure known as neural network architecture. For designing a reliability model we will require a multilayer of neurons. The neurons are arranged in three different layers known as input layer, hidden layer and output layer. The multi layer network is shown in below Figure2:

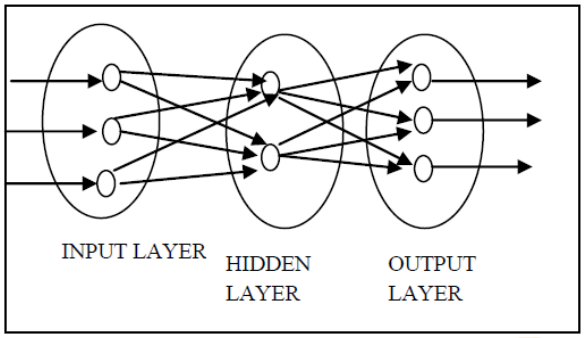

Figure2: Multi Layer Neural Networks

Back Propagation Algorithm

It is the training or learning algorithm. Example play a huge role here. If you submit to the algorithm the example of what you want the network to do, it changes the network's weights so that it can produce required output for a particular input after completing the training.

Back Propagation networks are useful for simple Pattern Recognition and Mapping Tasks.

Bayesian Networks (BN)

These are also known as Belief Networks or Bayes Nets .They are the graphical structures used to represent the probabilistic relationship among a set of random variables. BNs reason about the uncertain domain.

Each node represents a random variable with specific propositions in these networks. For example, in a medical diagnosis domain, the node Cancer represents the proposition that a patient has cancer. The edges connecting the nodes represent probabilistic dependencies among those random variables. If out of two nodes, one is affecting the other then they must be directly connected in the directions of the effect. The strength of the relationship between variables is measured by the probability associated with each node.

The only constraint on the arcs in a BN is that you cannot return to a node simply by following directed arcs sufficing for the BNs to be called Directed Acyclic Graphs (DAGs).

BNs are capable of handling multivalued variables simultaneously. The BN variables are composed of two dimensions −

Range of prepositions

Probability assigned to each of the prepositions.

Consider a finite set X = {X1, X2, …,Xn} of discrete random variables, where each variable Xi may take values from a finite set, denoted by Val(Xi). If there is a directed link from variable Xi to variable, Xj, then variable Xi will be a parent of variable Xj showing direct dependencies between the variables.The BN is ideal for aggregating prior knowledge and observed data. It can be used to learn the causal relationships and understand various problem domains and to predict future events, even in the case of missing data.

Building a Bayesian Network

A Bayesian network can be built by a knowledge engineer. Following are the number of steps the knowledge engineer needs to take while building it.

Example problem – *Lung cancer*. A patient has been suffering from breathlessness. He visits the doctor, suspecting that he has lung cancer. The doctor knows that other than the lung cancer, there are various other possible diseases the patient might be suffering from, such as tuberculosis and bronchitis.

Collect Relevant Information of Problem-

- Is the patient a smoker? If yes, then high chances of cancer and bronchitis.
- Is the patient exposed to air pollution? If yes, what sort of air pollution?
- Take an X-Ray. Positive X-ray would indicate either TB or lung cancer.

Applications of Neural Networks

They can perform tasks that are easy for a human but difficult for a machine −

- Aerospace − Autopilot aircrafts, aircraft fault detection.
- Automotive − Automobile guidance systems.
- Military − Face recognition Weapon orientation and steering, target tracking, object discrimination, signal/image identification.
- Electronics − Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.
- Financial − Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis, currency value prediction, document reading, credit application evaluating.

- Industrial − Manufacturing process control, product design and analysis, quality inspection systems, welding quality analysis, paper quality prediction, chemical product design analysis, dynamic modelling of chemical process systems, machine maintenance analysis, project bidding, planning, and management.
- Medical − Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
- Speech − Speech recognition, speech classification, text to speech conversion.
- Telecommunications − Image and data compression, automated information services, real-time spoken language translation.
- Transportation − Truck Brake system diagnosis, vehicle scheduling, routing systems.
- Software − Pattern Recognition in facial recognition, optical character recognition, etc.
- Time Series Prediction − Predictions on stocks and natural calamities.
- Signal Processing − Process an audio signal and filter it appropriately in the hearing aids.
- Control − Make steering decisions of physical vehicles.
- Anomaly Detection – Since ANNs are expert at recognizing patterns, they can also be trained to obtain an output when something undesirable occurs that mismatches the pattern.

Advantages:

- A neural network can perform tasks which a linear program cannot perform.
- Due to their parallel nature, even when an element of the neural network fails, it can continue without any problem.
- A neural network need not be reprogrammed as it learns itself.
- It can be implemented in an easily without any problem.
- As the adaptive, intelligent systems, neural networks are robust and expert at solving complex problems. They are efficient in their programming and the scientists agree that the advantages of using ANNs outweigh the risks.
- It can be implemented in any type of application.

Disadvantages:

- The neural networks need training to operate.
- High processing time is required for large neural networks.
- The architecture of a neural network is different from the architecture of microprocessors. So, they have to be emulated.

Conclusions:

Today's industry systems more and more depend on software which is sometimes very complex. Software complexity increases also with the risk factor of the environment where the whole system is deployed. From these reasons the requirements for software reliability cannot be missed out when designing such system. Software reliability differs from hardware reliability because it reflects the design perfection, rather than manufacturing perfection. Similar to hardware the reliability of software should be evaluated and measured, even it is not so simple task as it is in hardware because software is hard to touch. Many organizations and individuals developed methods for software reliability evaluation. In the existing software reliability models the failure check is performed after the coding and the implementation phase. Sometimes due to faulty designs requires reimplementation of the project. It leads to wastage of the resources and the time.

References:

[1] Garima Chawla et. al. , *A Fault Analysis based Model for Software Reliability Estimation*, International Journal of Recent Technology and Engineering (IJRTE),ISSN: 2277 -3878, Volume-2, Issue-3, July 2013.

[2] J. Laprie and K. Kanoun "Software Reliability and System Reliability", In M. R. Lyu, editor, Handbook of Software Reliability Engineering, pages 27–69. McGraw Hill, 1996.

[3] K. Goseva-Popstojanova, and K. Trivedi "Failure Correlation in Software Reliability Models", IEEE Transactions on Reliability, Vol. 49, No. 1, March 2000.

[4] S., Dick and A. Kandel "Computational Intelligence in Software Quality Assurance", World Scientific Publishing Co. 2005.

[5] Rita G. Al gargoor et. al. , *Software Reliability Prediction Using Artificial Techniques*, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2, July 2013,ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784,www.IJCSI.org.

[6] Al-Rahamneh Z., Reyalat M. Sheta A. F., Bani-Ahmad S., and Al-Oqeili S., *A New Software Reliability Growth Model: Genetic-Programming-Based Approach* , Journal of Software Engineering and Applications, 2011, pp. 476-481.

[7] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.

[8] Su, Yu Shen, et al. *An Artificial Neural-Network-Based Approach To Software Reliability Assessment*, TENCON 2005, IEEE Region 10. IEEE, 2005

[9] S. Yamada, and S. Osaki, *Reliability Growth Models for Hardware and Software Systems Based on Non-homogeneous Poisson Processes: a Survey*, Microelectronics and Reliability, 23, 1983, pp. 91-112.

[10] S. Yamada, and S. Osaki, *S-shaped Software Reliability Growth Model with Four Types of Software Error Data,* Int. J. Systems Science, 14, 1983, pp. 683-692

[11] Inoue S., and Yamada S ,*Two-Dimensional Software Reliability Measurement Technologies*, IEEE , 2009, pp. 223 – 227

[12] Quadri S. M., Ahmad N. and Farooq S. U. ,*Software Reliability Growth Modeling With Generalized Exponential Testing –Effort And Optimal Software Release Policy* , Global Journal of Computer Science and Technology , 2011, pp.27 – 42

[13] Prof. C. J. van Duijn et. al. ,*Statistical Procedures for Certification of Software Systems*, © Corro Ramos, Isaac 2009.

[14] Latha Shanmugam et. al., *A Comparison Of Parameter Best Estimation Method For Software Reliability Models*, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.5, September 2012.

[15] Mohd. Anjum et. al. , *Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value*, I.J. Information Technology and Computer Science, 2013, 02, 1-14.

[16] Mamta Arora et. al. , *Software Reliability Prediction Using Neural Network*, International Journal of Software and Web Sciences 5(2), June-August, 2013, pp.88-92.

[17] Musa JD " Validility of the Execution time theory of Software Reliability "IEEE trans on Reliability R-283) pp 181-191 Aug1979.

[18] Eckhardth D.E et al "An experimental Evaluation of Software redundancy as a strategy for improving reliability" IEEE trans on software engineering.

[19] By KK Agarwal and Yogesh Singh "Software Engineering"

[20] M.R. Lyu, "Handbook of Software Reliability Engineering", *IEEE Computer Society Press*, McGraw Hill, 1996.

[21] K. Mehrotra, C.K. Mohan and S. Ranka, "Elements of Artificial Neural Networks", *Penram International Publishing*, 1997.

[22] Tian, L. & Noore, A., "Evolutionary neural network modeling for software cumulative failure time prediction," *Reliability Engineering & System Safety*, vol. 87, no. 1, pp. 45-51, 2005.

[23] Guo, P. & Lyu, M.R., "A pseudo inverse learning algorithm for feed forward neural networks with stacked generalization applications to software reliability growth data," *Neurocomputing*, vol. 56, pp. 101-121, 2004.

[24] Ho, S.L., Xie, M. & Goh, T.N., "A study of the connectionist models for software reliability prediction," *Computers & Mathematics with Applications*, vol. 46, no. 7, pp. 1037-1045,2003.

[25] Cai, K.Y., Cai, L., Wang, W.D., Yu, Z.Y. & Zha ng, D, "On the neural network approach in software reliability modeling", *Journal of Systems and Software*, vol. 58, no. 1, pp. 47-62,2001.

[26] Karunanithi, N., Malaiya, Y. K., & Whitley, D. (1991, May). Prediction of software reliability using neural networks. In *Software Reliability Engineering, 1991. Proceedings., 1991 International Symposium on* (pp. 124-130). IEEE.