T.C.

ALTINBAS UNIVERSITY

Information Technologies

STUDENT PERFORMANCE PREDICTION USING DEEP BELIEF NETWORK

ABDULBASIT MAHMOOD OLEIWI AL-JUBOORI

Master Thesis

Supervisor: Asst. Prof. Dr. Sefer Kurnaz

Istanbul, 2019

STUDENT PERFORMANCE PREDICTION USING DEEP BELIEF NETWORK

by

Abdulbasit Mahmood Oleiwi AL-JUBOORI

Information Technologies

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Sefer KURNAZ

Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Prof. Dr. Osman Nuri UÇAN	School of Engineering and Natural Sciences, Altinbas University	_____
Prof. Dr. Oğuz BAYAT	School of Engineering and Natural Sciences, Altinbas University	_____
Prof. Dr. Hasan Hüseyin BALIK	Air Force Academy, National Defense University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. NAME

Head of Department

Approval Date of Graduate School of Science and Engineering: ___/____/____

_____

Asst. Prof. Dr. Sefer KURNAZ

Director

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Abdulbasit Mahmood Oleiwi AL-JUBOORI

DEDICATION

I give you this thesis in a form of appreciation and I may ask Allah to accept this work.

ACKNOWLEDGEMENTS

I wish to express my acknowledgements to my supervisor, Asst. Prof. Dr. Sefer Kurnaz who was abundantly helpful and offered invaluable support with his sincerity and belief in me.

ABSTRACT

STUDENT PERFORMANCE PREDICTION USING DEEP BELIEF NETWORK

AL-JUBOORI, ABDULBASIT MAHMOOD OLEIWI

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst Prof. Dr. Sefer kurnaz

Date: March 2019

Pages: 51

Education research involving techniques of data mining is growing rapidly. Data mining techniques are known as educational information mining to explore hidden knowledge and patterns of student performance in educational backgrounds. This work seeks to develop an academic prediction model for the student for the UCI (University of California Irvine) machine learning repository dataset selected in this work for prediction. Internal marks are a combination of attendance marks, average marks from two examinations and marks of assignment. The teachers can therefore classify students and start predicting their performance at an early stage. To improve performance over time, systemic approaches can be adopted. Better results can be expected at the final examinations due to early predictions and solutions.

Keywords: Artificial neural network, Backpropagation, Mean square error, Deep Belief Network, classification, student performance prediction.

ÖZET

DENİR İNANÇ AĞINI KULLANARAK ÖĞRENCİLERİN PERFORMANS TAHMİNİ

AL-JUBOORI, ABDULBASIT MAHMOOD OLEIWI

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Danışman: Asst Prof. Dr. Sefer kurnaz

Tarih: March 2019

Sayfalar: 51

Veri madenciliği tekniklerini içeren eğitim araştırması hızla artıyor. Veri madenciliği teknikleri, eğitim geçmişinde gizli bilgi ve öğrenci performans modellerini keşfetmek için eğitim bilgi madenciliği olarak bilinir. Bu çalışma, öngörü için bu çalışmada seçilen UCI (California Irvine Üniversitesi) makine öğrenme veri havuzu için öğrenciye yönelik bir akademik tahmin modeli geliştirmeyi amaçlamaktadır. İç işaretler, katılım işaretlerinin, iki sınavdan alınan ortalama işaretlerin ve ödev işaretlerinin birleşimidir. Öğretmenler bu nedenle öğrencileri sınıflandırabilir ve performanslarını erken bir aşamada tahmin etmeye başlayabilir. Zaman içindeki performansı arttırmak için sistemik yaklaşımlar benimsenebilir. Erken tahminler ve çözümler nedeniyle final sınavlarında daha iyi sonuçlar beklenebilir.

Anahtar kelimeleri: Yapay sinir ağı, Backpropagation, Ortalama kare hatası, Denir İnanç Ağı, sınıflandırma, öğrenci performans tahmini.

TABLE OF CONTENTS

Pages

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION

The neural system is arranged into hidden layers, input and output of the Artificial Neural Networks (ANNs). A series of synaptic weights joins the neurons together. ANN is a strong instrument in a multitude of issues for defining trends, forecasts and regressions. The ANN continually changes its synaptic values during the learning process until sufficient acquired knowledge (until a certain number of iterations have been achieved or the error value of the target has been achieved). After completion of the learning or training stage, the ability of the ANN to generalize the problem with samples other than those employed during the training stage must be assessed. Finally, during training and testing, it is expected that the ANN will be able to accurately classify the patterns of a particular problem. In recent years several classic ANN algorithms have been suggested and developed. However, many may still be attracted to unwanted alternatives; that is, the optimal or the best solution is far from them. Moreover, most algorithms cannot examine multimodal or non-constant surfaces.

Consequently, other types of technology are essential for training an ANN, like bio-inspired algorithms (BIAs). The artificial intelligence community considers BIAs as powerful instruments for optimization and can solve extremely complex problems with optimization. In big multimodal and continuous search areas, BIAs can search for a particular issue and discover the optimum solution value. BIAs rely on the conduct of nature as an intelligence swarm. The notion of [1] is defined as owned by unintelligent officials with restricted personal capabilities, but smart collective conduct.

Several studies have been carried out using evolutionary and bio motivated algorithms to form ANN [2]. The neural network training metaheuristic is based on studies in the areas of local communities, population systems and other methods such as cooperation models [3]. The writers demonstrate an outstanding job in which they review the changing ANN algorithms in literature [2]. However, the majority of research reports focus on developing synaptic weight, parameters [4] or changes to the number of cloaked layers of the neurons, but formerly the designer determines the number of cloaked layers. Furthermore, researchers do not need to develop transmission characteristics, an important element of an ANN that determines the output of every neuron. They suggested, for instance, a mix of the Ant Colony Optimization technique for weight adjustment

through the identification of ANN and PSO in [5]. Additional studies such as the [6] modify the Simulated Annealing (SA) PSO for the acquisition of a set of synaptic weight and threshold ANNs. In [7], In order to solve classification and forecast issues, the authors use Evolutionary Programming for architecture and weight. Another instance is Genetic Programming [8] where graphs representing a range of topologies have been produced. In [9] an ANN was designed with a Differential Evolution (DE) algorithm to fix an issue with weather projections. The writers use a synaptic weight algorithm in [10] to alter the relation of daily rainfall and Malay rivers. In [11] comparison with the fundamental PSO, the writers only adapt the synaptic weights of an ANN for classification solutions. The weights in [12] are developed using the differential change and fundamental PSO. The three principal elements of an ANN were simultaneously developed in other works such as [13]: architecture, transfers and synaptic weights. With the Evolution (DE) algorithm differential [14], the authors resolved the same problem, and proposed a new model for their authors with the PSO (NMPSO) algorithm. The author was also involved in the design of an ANN, using a two distinct fitness function artificial colony (ABC) algorithm.

1.1 PURPOSE AND THESIS IMPORTANCE

In this thesis, we're using deep belief network for the classification of Student dataset that is taken for UCI Machine learning for performance prediction. The ability that deep belief network has in the training according to the data and understanding its information might make this algorithm better than a lot of classification algorithms around us including artificial neural network. In this study, we calculated the classification accuracy, Mean Square Error (MSE), Specificity, and sensitivity.

2. LITERETURE REVIEW

The neural system, the neural networks (ANNs), is structured into concealed layers and input and output. A number of synaptic weights are attached to the cells. An ANN is a strong instrument for identifying different problems ' models, expectations, and regressions. The ANN converts its synaptic values continually into a sufficient sense during the training stage (up to some of the iterations or error value of the target is reached). After the training stage, the capacity of ANN to generalize the problem must be evaluated with samples different from those used during the training stage. The problem should be generalized.

In recent years, several classical ANN algorithms were suggested and created. Many, however, can still be stuck in unwanted options, so they are not as good as the finest. In most algorithms multimodal and not continuous surfaces can also be studied. Therefore, other methods are required for the practice of an ANN, such as bio-inspired algorithms (BIAs).

As BIAs are powerful optimizing instruments and can solve very complex optimizing issues, they have a positive support by artificial intelligence. BIAs can explore large multimodal and non - constant search areas for the best solution near the optimal value for a specific problem. BIAs are based on the behavior of nature described as the swarm. In [1] this concept is defined as owned by smart and limited personal agents with smart collective behavior.

A number of works are used to train ANN for another basic way of learning using evolutionary and bioinspired algorithms (2). Metaheuristic neural networking techniques are based on local research, demographic techniques and other models, for example cooperative modeling [3].

The writers demonstrate an extensive literary review of the growth of ANN's evolutionary algorithms [2]. However, the majority of the study papers focus solely on synaptic weight growth, parameters [4] and the growth of neuron count in concealed layers. The developer calculates the amount of layers concealed. The research also does not require the evolution of transmission features, a key component of an ANN that determines each neuron's output.

In [6], the authors suggested a method combining ANN and Particle Swarm Optimization (PSO) for a specific architecture (connections) to adjust the weight of the synaptics. Further tests like [6] for obtaining ANN and ANN weights have been conducted using simulated annealings for a

modified PSO blend. In order to solve classification and forecast problem [7] Authors use evolutionary programming to define architecture and the weights. Another instance is Genetic Programming [8] for graphs representing various topological topologies. In [9], an ANN was designed to fix a climate issue by using the Differential Evolution (DE) algorithm. In [10] the writers adapted the synaptic weights to model the daily connections between precipitation and rain in Malaysia by using the PSO algorithm. In [11], writers only adjust the synaptic weight of ANN for solving classification issues with the back-propagation process versus fundamental PSO. In [12], the weights set are changed through the Differential and Basic PSO evolutions.

The three main elements of the ANN have evolved simultaneously in other works such as [13]: architecture, transfer functions and synaptic weights. In [14], the authors solved the same problem using the Differential Evolution (DE) algorithm. They proposed a new PSO model. Another instance is [15] in which the writers are developing an ANN Design using two distinct fitness functions using an Artish Bee Colony (ABC).

Compared to the last three works, this research has important contributions. First and foremost, there are eight fitness functions to address three common problems arising from the ANN design: accuracy, overriding and reducing ANN. Fitness features take classification mistakes into consideration, mean square error, validation error, architecture declines and a combined ANN design problem. This research further examines the behaviour, using different parameter values, of the three bioinspired algorithms. The best parameter values are determined for these algorithms in order to obtain the highest outcomes during the experimentation stage. In addition, a set of statistically valid tests is created with the optimum configuration for each classification problem selected. In addition, the results obtained by the proposed methodology are presented and discussed regarding the connection number, the number of the neuron and the transfer functions selected in the ANN. Another contribution to this work concerns a fresh metric that enables the outcomes produced by an ANN to be compared effectively with the methodology suggested. This measurement requires account of the identification rate achieved during training and tests, where test precision is more accurately weighed compared with training precision. Finally, these findings are comparable to those obtained with two classical learning algorithms by the three bioinspired algorithms. The choice of three inspirational biological algorithms is based on a metaphor for a

fundamental PSO method because the algorithm of NMPSO is comparatively fresh (2009). It is important to compare its performance to other algorithms that are motivated by the same.

In principle, a number of input patterns, a set of desired patterns and the function to be determined as an ANN can be defined to minimize and define the maximum number of neurons. There are three regions (architecture, synaptic weight, transmission characteristics) within the search space.

Automatic handwriting identification in latest centuries has been one of the most active areas of studies. Due to the cursive nature and the vast array of styles, vocabulary … etc., the task of recognition is a challenge [45, 46]. Although character recognition rates are high, it is not easy to recognize offline text. The words are separated in their character parts by the analysis of character shapes in a large number of text-recognition pieces. For the purpose of performing text recognition, stochastic approaches like Hidden Markov Models [54, 57, 59] were used. HMM's avoid cursive word segmentation by joint segmentation and recognition into character/subwords [59]. Recent sequence classifications of Recurrent Neural Network (RNN) have been shown to perform better [47]. The BLSTM RNN architecture allows access both in the input direction to the long-range context.

A classifier combination approach can further improve recognition performance. Classification combination hinges on assuming that different classifiers are capable of compensating for each other by combining their own strengths and weaknesses. In order to record word assumptions [42], In a word recognition module, a character recognition system has been integrated. Multi-layered perceptrons (MLP) have been commonly used in order to classify characters as part of isolated or permanent, man-made word recognizers [53]. Recent studies in the field of the Deep Belief Network [50] education strategy have resulted in an improvement in the performance of machine learning and in the identification of patterns. In the deep networks, non-linear detector hierarchies are developed which better capture complicated data patterns. The DBNs offer superior performance than MLPs because of its deep learning mechanism. DBNs are used to upgrade existing handwriting systems in this work as a verification module.

This study provides a detailed analysis of how an ANN can automatically use bio-inspired algorithms, especially by implementing Basic Particle Swarm Optimization (PSO), SGPSO and the New PSO Model (NMPSO). The suggested approach evolves the architecture, synaptic

weights and type of transfer features simultaneously with the design of the ANNs which are most precise in a specific issue. In addition, a comparison is provided between the results of the particle swarm algorithm and the classical learning method (back-propagation and Levenberg-Marquardt). Furthermore, a fresh way of selecting maximum amount of neurons (MNN) is provided in this study. It is screened to resolve some actual and synthetic pattern recognition issues with the precision of the suggested methodology.

3. METHODOLOGY

3.1 DATA RESOURCE

This information focuses on secondary school student accomplishment in two Portuguese colleges. The data features include grades of college, population, social and education features) and are acquired through school reports and surveys. For two separate topics, math (matt) and Portuguese (por), two performance data sets are given. Both data sets were modeled on five grading and regression rates. Note: Objective G3 attribute is strongly associated with G2 and G1 characteristics. The first and second periods are the G3, the last year and G1 and G2. Last year's G3 grade. Without G2 and G1, G3 is difficult to foresee, but such a forecast is much more helpful.

The Attributes of Student Performance Database are described below:

- Student's school (binary: 'MS' - Mousinho da Silveira or 'GP' - Gabriel Pereira)
- The sex of student's ('M' - male or 'F' - female)
- The age of student's (from 15 to 22)
- The Home address type of the Student ('U' - urban or 'R' - rural)
- Family size ('LE3' - less or equal to 3 or 'GT3' - greater than 3)
- Cohabitation status of student’s Parents ('T' - living together or 'A' - apart)
- The mother’s cohabitation status degree (0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
- The father’s cohabitation status degree (0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
- The job that the mother has ('teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
- The job that the father has ('teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

- School choosing reason (close to 'home', school 'reputation', 'course' preference or 'other')
- The main guardian for the student ('mother', 'father' or 'other')
- School travel time (1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
- Study time (Weekly) (1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- Past class total failures (n if 1<=n<3, else 4)
- Extra educational support (yes or no)
- Family educational support (yes or no)
- The course subject based on Extra paid classes (Math or Portuguese) (yes or no)
- Extra-activities (yes or no)
- Attended nursery school (yes or no)
- Higher education willing (yes or no)
- Internet accessibility (yes or no)
- In a romantic relationship (yes or no)
- Family relationships Quality (from 1 - very bad to 5 - excellent)
- Is there a free time after school? (from 1 - very low to 5 - very high)
- Hanging out with friends (from 1 - very low to 5 - very high)
- Workday alcohol (from 1 - very low to 5 - very high)
- Weekend alcohol (from 1 - very low to 5 - very high)
- Health status (from 1 - very bad to 5 - very good)
- Total school absences (from 0 to 93)

3.2 DATA PREPROCESSING

The dataset had some missing values which was fix using the tool “IBM SPSS statics 24” and using the algorithm Standardization and Normalization.

Normalization

Data preprocessing plays a vital role in the general process of information discovery prior to information mining itself. The normalization of information is one of the first steps. In coping with parameters from various units and scales, the step is very crucial. For instance, some methods of information mining use the distance of Euclidean. For a reasonable comparison between them, therefore, all parameters should have the same scale.

In the following formula, all numeric variables are scaled in the range [0, 1]. One possible formula is given below:

$$X_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3.1)$$

3.3 DEEP BELIEF NETWORK

A network for deep growth may be defined as a stack of limited Boltzmann machines, where preceding and following layers of each RBM layer are connected. Each layer of the nodes does not communicate side-by-side. This set of RBMs could end in a softmax classification layer or simply help unlabeled information to be grouped into an unattended learning scenario. The entry layer (or the visible layer) for the following nodes has a dual role with the exception of the initial and final layers, each layer in the deep credential network serving as the hidden layer for the following nodes. It's a network built of one layer. Video and motion capture data, and images are created and composed using networks of deep creativity. A permanent deep-belief network is just an extension that accepts a number of decimal places instead of binary data which this method was introduced by Geoff Hinton and his students in 2006 [51].

The DBNs consist of Restricted Boltzmann (RBM) layers in the pre-train phase and then a fine tuning feed network. As illustrated in the DBN network architecture [51]:

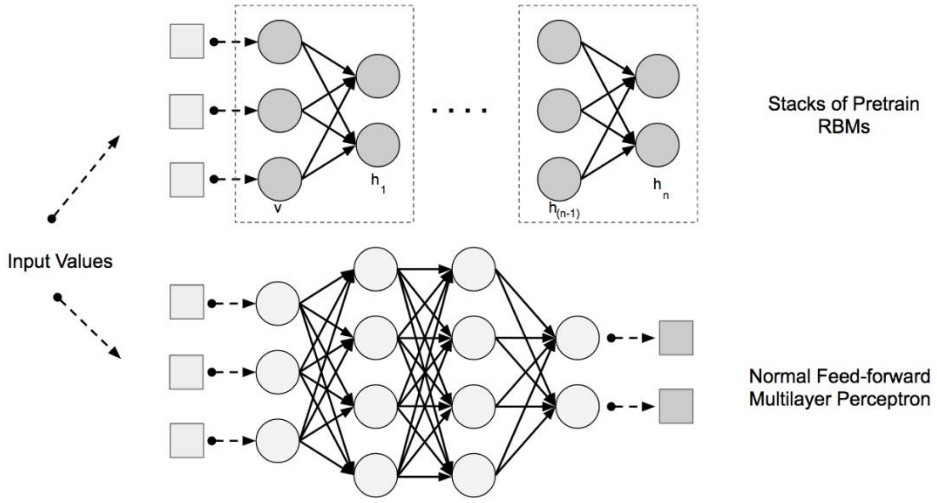

Figure 3.1: DBN's Network Architecture

In many machine learning projects with competitive outcomes, Deep Belief Networks (DBNs)[51] were effectively established. A limited Boltzmann machine (RBM) has been developed as a packaging form for the primary construction blocks. RBM is a random field of Markov-style with two layers of visible, stochastic units v hidden architecture h. An RBM training is used as a tool to model a lower level distribution of data in a CD algorithm [51]. Every layer of latent image is taught. The blended distribution of likelihood P(v, h) in visible units v and cache units h outcomes in model parameters for the joint distribution of probability units v and bias units. Based on the power function E(v, h) this distribution is calculated for binary RBMs:

$$E(v,h|\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij} v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j \tag{3.2}$$

Where parameters of the model= {w, b, a} have been identified. Unit I between Units I is visible and hidden. For visible unit I and hidden unit j, bi and aj are both predictive terms. V and H are unit numbers that are visible and hidden. P(v) is calculated as a marginal probability in [51]:

$$P(v|\theta) = \frac{\sum_h \exp\left(-E(v,h|\theta)\right)}{Z(\theta)} \tag{3.3}$$

Where the partition function is represented as Z(θ) [51] which can be calculated by:

$$Z(\theta) = \sum_v \sum_h \exp\left(-E(v, h|\theta)\right) \tag{3.4}$$

We use Eq.3 to calculate the probabilities for each data vector v for the hidden activation unit h. The hidden probabilities of activation are used for training for a new RBM. Once the workout of RBM has taken place, a number of cached layers of the neural network's hidden layers are initiated using the RBM count. Following training, the weight of the network can be discriminated by a randomly initialized softmax output layer.

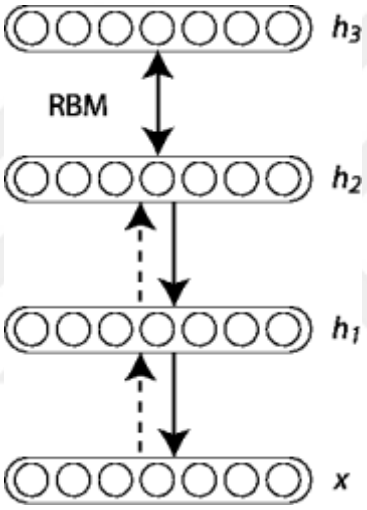

Figure 3.2: Hidden levels of RBMs

According to Figure 3.2, for DBNs with RBMs, as building blocks for each layer the principle of greedy layer-specific unmonitored training can be applied. The following is the process:

1. Form the first layer of raw input $x = h^{(0)}$ as its obvious layer as an RBM.

2. Use the first layer to display the input which will be used as the second layer data. There are two shared solutions. The mean activations $p\left(h^{(1)} = 1|h^{(0)}\right)$ or sample p can be chosen as this figure.

3. Train the second layer as RBM (visible RBM layer) and for instance train transformed information.

4. Iterates (2 and 3), each time spreading upward, samples or medium values, for the desired number of layers.

5. Fine tuning all of the parameters for the DBN proxy log probability or a supervised training criterion in this profound architecture (after the additions of additional training machines to convert the learner to a supervised forecast, e.g. a linear classifying).

We will focus on ending the pitch with a controlled descent in this tutorial. In order to sort Input X with the outputs of the previous hidden DBN layer, we use a logistic regression classifier, in particular. The completion of the negative log-like cost function is then performed with a controlled gradient descent. Since the controlled gradient for each layer is just zero in weight and occult layer bias (i.e. null for each RBM's visible bias), this process is equivalent to initializing the MLP parameters with the uncontrolled weight and hidden layer bias training strategy.

3.3.1 Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) are neural networks that belong to so called Energy Based Models. This type of neural networks may be not that familiar to the reader of this article as e.g. feedforward or convolution neural networks. However, in the context of a Netflix Prize, such neural networks have become very popular in latest years, where RBMs have attained cutting edge technology and won the most of the contest.

3.3.1.1 Architecture

In my view RBMs have one of all neural networks ' simplest architectures. As Figure 3.3 illustrates. A RBM comprises of an input / visible layer (v1 .... v6), a concealed layer (h1,h2) and the Bias and Bias vectors corresponding to each layer of Bias. The absence of an output layer is apparent. But as it can be seen later an output layer won’t be needed since the predictions are made differently as in regular feedforward neural networks.

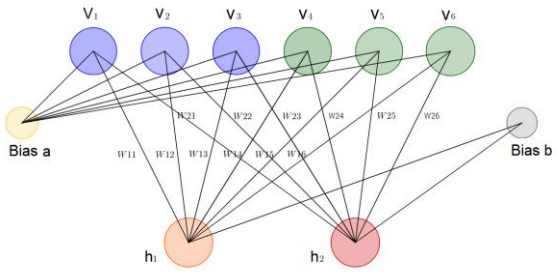

Figure 3.3: RBM architecture

3.3.1.2 Collaborative filtering with restricted boltzmann machines

Suppose some individuals have been requested to rate a collection of 1-5 star films. Each film could be described in classical factor analysis in terms of a number of latent variables. For example, movies like Harry Potter and Fast and the Furious might have strong associations with a latent factor of fantasy and action. On the other hand, users who like Toy Story and Wall-E might have strong associations with latent Pixar factor. These underlying factors are used to evaluate and detect which is done by RBMs. After some epochs of the training phase the neural network has seen all ratings in the training date set of each user multiply times. At this time the model should have learned the underlying hidden factors based on users preferences and corresponding collaborative movie tastes of all users.

The assessment of concealed variables is done binary. The user just tells if they like (rating 1) or not a particular film, and not gives the models continuing user ratings (e.g. from 1–5 stars). The rating values for the input/visible layer are the binary ratings. Due to the inputs of the RMB, latent variables in information are detected that can explain the decisions of the film. One of the latent variables is the concealed neuron. Given a large dataset consisting out of thousands of movies it is quite certain that a user watched and rated only a small amount of those. It is necessary to give yet unrated movies also a value, e.g. -1.0 so that the network can identify the unrated movies during training time and ignore the weights associated with them.

Take Lord of Rings and Harry Potter for example, but don't like Matrix, Fight Club or Titanic. Let's take the example below. There is still no noticeable Hobbit, so the score is 1. Based on the

inputs, three variables that are disguised in drama, fantasy and scientific fiction are recognizable in Boltzmann's machine.

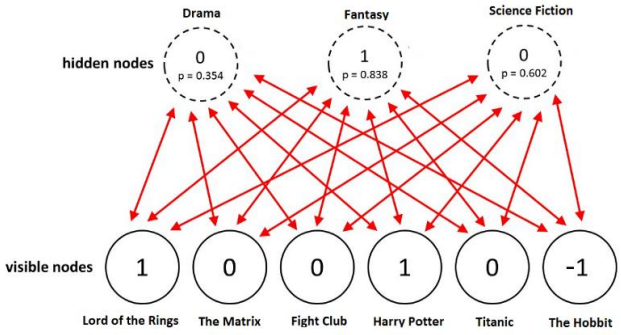

Figure 3.4: Identification of latent factors.

Because of the films, the RMB gives each hidden neuron a probability p(h) Samples taken from the Bernoulli distribution are used to determine the final binary values of the neurons. In this example only the hidden neuron that represents the genre Fantasy becomes activate. Given the movie ratings the Restricted Boltzmann Machine recognized correctly that the user likes Fantasy the most.

3.3.1.3 Training

The Boltzmann restricted machine's training varies from that of standard neural networks through stochastic gradient downward path. The deviation of the training procedure for a RBM won't be covered here. Instead I will give a short overview of the two main training steps and refer the reader of this article to check out the original paper on Restricted Boltzmann Machines.

1. **Gibbs Sampling:**
   Gibbs Sampling is the first component of the training. In order to predict concealed values (h), p(h) is used by an input matrix. The concealed values are known to use p(v)

for forecasting fresh input values v. After K iterations, we get another input vector vK, recreated from the initial input v0.

2. **Contrastive Divergence:**

During the contrasting difference, an update of the weight matrix takes place. For concealed values h0 and hk (Eq.4), vectors v0 and vk are used to calculate activation probabilities. The distinction between external goods with v0 and vk input vectors leads to the update matrix:

$$\Delta W = v_0 \otimes p(h_0|v_0) - v_k \otimes p(h_k|v_k) \tag{3.4}$$

The new weights can be determined by gradient ascent with the Update matrix provided by:

$$W_{new} = W_{old} + \Delta W \tag{3.5}$$

3.3.2 Deep Belief Network Training

The first step is to train a layer of properties which can obtain the input signals from the pixels directly. The next step is to consider the layer values as pixels in a second, concealed layer and learn about the previous characteristics. Each time a different layer of features or properties is added to the creed network, the lower limit of the log probability will be improved.

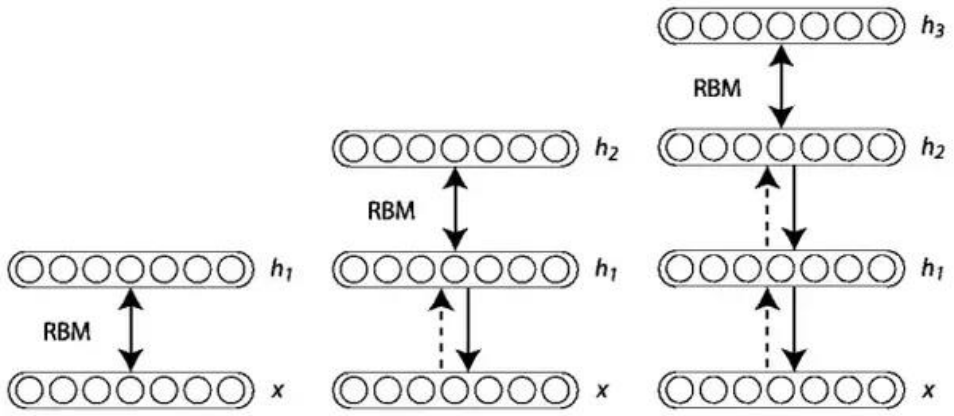

Figure 3.5: Deep Belief Network Training Example

Fine Tuning using Backpropagation

Backward propagation works better with gullible layer-wise training. We do not begin reverse propagation until sensitive detectors that are useful for discriminatory work have been identified. No new features are to be found for fine tuning purposes. DBM's objective is to improve the model's accuracy by identifying ideal layer-to-layer weight values. Once we have identified sensitive detectors, only a local search is necessary for reverse diffusion. Unlabeled data helps to find good characteristics. We also may have features that are not very helpful for the task of discrimination, but that is not a problem. From the raw input we still have useful functions. Generally speaking, input vectors contain more information than labels. The label is used only for fine tuning. Precious information. Labeling dataset supports associating dataset patterns and features. For the fine tuning with backward propagation, a small labeled dataset is used. In section 3.5, the background will be briefly explained.

3.4 ARTIFICIAL NEURAL NETWORKS

The neuron structure in the human brain inspires the Artificial Neural Network. The brain learns from experiences that go beyond the range of computers and thus adapts. This modeling also allows solutions to be less technologically developed in order to reduce human involvement.

The implementation of neural networks in computing gives us a key computational advance. Computers, like complex math and face recognition, do well. However, simple patterns are hard for computers. Computers cannot analyze, generalize and transform past patterns into future actions. The advanced neural network study allows people to understand the mechanism of thoughts, for example.

The study focusses on the' brain storage data as patterns method [17]. For individual faces to analyze and recognize, some of these patterns are very difficult. This process saves information as patterns, analyzes patterns and fixes a new computer field for problems. Training and creation of

these networks is involved in the neural network to solve specific problems. We can perform different techniques such as learning, behaving, forgetting and reacting, etc. through our neural network.

"All aspects of this processor are known: The human brain is still mysterious to operate accurately. In particular, the most important element in the human brain is a certain type of cell which does not appear to regenerate as opposed to the whole body. Since the only portion of the body not slowly replaced by this type of cell, we are supposed to be able to remember, think and put past experiences into practice in all our actions. These cells are all known as neurons". The neural network as the human brain's neural network offers power through its complex components, its control mechanisms and its subsystems. Genetics programming and learning are also involved. In electro-chemical ways, neurons transmit the information between them. Depending on the classification process used, these neurons are classified into different categories. Current systems remain incompatible with the human brain. "The most basic elements of this complex and powerful organization can only replace these artificial neural networks. Neural computing has never succeeded the developer who is trying to solve problems with the human brain. Neural computing was never substituted to human brain by the development company which attempts to solve problems. It's a new way of solving problems and machinery.

Let's see the overview of human neurons. The basic element of the neural network is neuron. The biological neurons are input and subsequently subject to various non - light operations and then produce final output. Four nerve cells are typical: dendrites, somas, axons and synapses. It accepts inputs as the task of dendrites. The input is processed by Soma. Axon-transforms the processed inputs and synapses— contacts neurons. The biological neurons in structure are not too straightforward but complex.

Biology improves the understanding of neurons. By understanding the biological brain, network designers can enhance their systems further. Neural artificial networks (ANN) are computer tools which have been widely accepted in many different disciplines to model complicated problems in the real world. ANNs could be defined as structures consisting of strongly linked adaptive elements called neurons that provide massive computations for the representation of knowledge.

"The principal features of the ANN are the ability to handle inaccurate and inadequate information, the ability to manage inaccurate information, robustness, failure and failure tolerances, fault and failure tolerance."

Artificial models possess following characteristics:

1. Nonlinearity makes it more suitable for data
2. Precise prediction for uncertain data and measurement errors due to noise insensitivity.
3. High parallelism means tolerance of hardware failure and quick processing.
4. Learn and adapt to the changing environment, allowing the system to update its internal architecture.
5. The model can be applied to unlearned data by generalization.

The main purpose of ANN-oriented Computing is to produce mathematical algorithms that permit the development of human knowledge and information processing in artificial neural networks. Patterns based on ANN may provide virtually accurate solutions to specifically formulated problems and processes that only experimental and field observations understand. “In a number of applications from modeling, classification, design recognition and multi - variable data analysis, microbiological ANNs have been applied. The digital image processing interests are derived from two main applications: improvement for human - the reading of photo information; and processing of image information for storing, transferring and representing autonome sensor machine; and defining a two - dynamic function as the image, f (x, y).

“Digital image processing is the processing of digital images via digital computers. A numerous element with a specific position and value comprises a digital picture. They are called picture elements, elements, pixels and skins. An extensive and diversified application for digital image processing”.

3.4.1 General Properties of Artificial Neural Networks

1. Nonlinear I/O mapping

The analysis of high-dimensional data has an increasing importance due to the growth of sensor resolution and computer memory capacity. Typical examples are images, speech signals and

multi-sensor data. In the analysis of these signals they are represented in their entirety or part by part in a sample space. As an example, a single data point may represent a 16x16 (sub) image. A collection of these images constitutes a cloud of points in a 256-dimensional space. In most multi-sensor data sets there is a large dependency between the sensors or between the sensor elements. This is certainly true for nearby image pixels. But also more fundamentally, it is not to be expected that any physical experiment contains hundreds of degrees of freedom that are of significant importance. Consequently, multi-dimensional data sets may be represented by lower dimensional descriptions. There are various reasons why such representations may be of interest. They may reveal the structure of the data or the problem, they may be used for relating individual data points to each other (e.g. finding the most similar one in a database) or they may be used for retrieving missing data values.

2. Generalization ability

The capacity to generalize is one of the main advantages of the neural networks. Therefore, a trained network can classify data from the same class as never before seen. Developers normally only have little share of all possible neural network generation patterns in real life applications. To achieve the best generalization, the information set should be split into three parts. The training is provided for training in neural networking. During practice, this error is minimized.

The validation set is used to determine neural network performance in untrained models. A test set to finally monitor a neural net's overall performance. In the minimum validation error, learning should be stopped. The net generalizes best at this stage. If learning is not stopped, overtraining occur, and net performance decreases over all data, even if the error is still smaller in the training data. The network must finally be checked for the third dataset, the test set, after the completion of the learning phase. Each n training cycle, ANNS performs one validation cycle.

3. Fault-tolerance (graceful degradation)

Fault tolerance is the feature that permits the system, when certain components of it fail (or one or more inside failures), to continue to operate properly. In comparison to an unnatural system, the decrease in operating quality is proportionate to the severity, unless the fault is

totally unsuccessful. Fault tolerance is particularly sought in high-availability or life-critical systems. If parts of a system break up, characteristics are called graceful degradation.

A defect-tolerant design allows a system to operate at a lower level than to fail completely if some of the system fails. If the system does not work. The most common word for computer systems is used if the output is partial or if the response time is increased in higher or less complete operation. In other words, the entire system is not stopped due to hardware or software problems. A motor car is designed to continue to drive in the presence of damage caused by fatigue, corrosion, manufacturing defects or impacts, when a tire has been punctured or when structural integrity is maintained. Another example is an automotive vehicle.

Failure tolerance can be attained through anticipation of unique circumstances and the construction of a scheme to cope with them within the scope of the single scheme and, in particular, by attempting to stable themselves in order to bring the system together in a faultless state. However, some breeding is better used when the impacts of an unsatisfactory system are catastrophic or if the price is big enough. In each case, the system must be able to reverse it in safe mode, if a system failure is so catastrophic. The recovery is like a reversal, but if people are present, it can be a human action.

4. Biological analogy

Analogy is a cognitive process where information or significance is translated to a different objective or linguistic expression from one particular topic–analogy or source. A less narrowly defined inference or argument from one specific to another is in comparison to the analogy of deduction, induction and abduction where at least one premise or conclusion is general. The word analogy could also refer to the relation of the source and the goal itself, often but not necessarily similar, as in the concept of biology. The atom model of Rutherford analogized the atom with the solar system. Analogy plays a major role in problem solving, including decision-making, discussion, perception, generality, remembrance, creativeness, invention, prevision, emotions, explanation, conceptualization and communication. The analogy was argued as "the cognitive nucleus." In particular, the analogical language includes, although not metonymous, exemplification, comparatives, metaphors, like, allegories and parable. It is not methonymous,

but it does not contain the identification of places, objects or persons. As if it were, phrases like, etc. and so on, and the same term also depends on the analogous understanding of the message by the receiver. Analogy is important not only in ordinary language and common sense in the fields of science, philosophy, law and humanity. Analogy is closely linked to concepts of association, comparative approach, correspondence, mathematical and morphological equality, homomorphism and iconicity. The concept of conceptual metaphor in the cognitive linguistics can be the same as the concept of analogy. Analogy also provides a basis for all comparative arguments and experiments whose findings are conveyed to objects not examined.

3.4.2 Structure of ANN

The neural network is made up of 3 groups, layers, and phases. The entry, coverage and output phases.

- The raw data that the network receives are the activity or input unit.
- The cached phase is based on the input data and the connection weights between the input and the cached devices.
- The output phase depends on the cloak activity and weights between the cloak and output units.

On the basis of the layer activity, different network types exist. A simple network type is called where the hidden units can build their own input representation. Whenever each hidden unit is active, the masses of the hidden and input units can be adjusted to allow for a hide unit to choose what it is. Other architectures such as single and multilayer are also available. In a single layer each layer is connected to the other. Overall, the network of the single layer includes only entries and outputs. The inputs are supplied by a number of weights to outputs. In several layers all units have inputs, hidden and outputs in different layers.

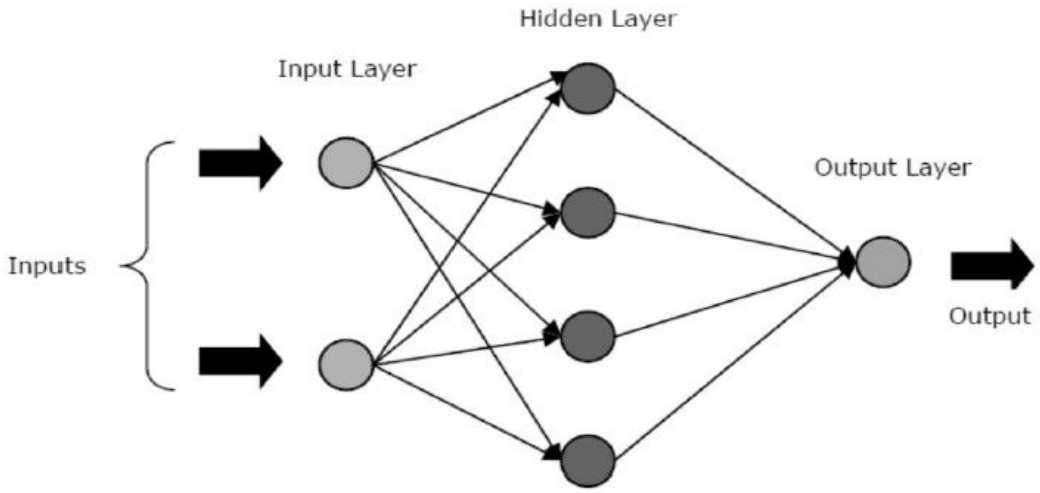

Figure 3.6: General neural network architecture [2]

3.4.3 Elements of Artificial Neural Networks

3.4.3.1 Inputs

The data, signal, feature, or outside environment measurements are received from this layer. These inputs are generally normalized within the limits of the activation functions. The values of samples or patterns. This standardization gives greater numerical accuracy for the network's mathematical operations.

3.4.3.2 Weights

The links between neurons are assigned to the number "weights" or "parameters" in artificial neural networks. These weights change as new data are fed into the neural net. So "learns the neural net.

The weights in an artificial neural network are an estimation of the combined multiple processes in biological neurons. Myelination is important, but not important. Myelination. Weights can be positive or negative in artificial neural networks. Weight: weight is comparable to an increased combination of dendrites between neurons, numbers of synapses between dendrites, density of postsynaptic terminal neurotransmitter receptors, as well as increased formation and fusion of neurotransmitters of the vesicles and of the pre-synaptic terminals.

Positive weights: The positive weights of the synapses releasing excitation neurotransmitters (i.e. glutamate) are analogous to the pre - synaptic terminals. The receiving cell will increase its likelihood of activation.

Negative weights: Negative weights are similar to those of the neurotransmitter synapse (i.e. GABA). The receiving cell is less likely to fire a potential action.

Myelination: Myelination increases the distance between the action potential and the axon. If the axon is not myelinated, the membrane voltage potential decreases far closer to the cell body. A garden hose is analogous. If the axon does not myelinate, the garden pants are leaking, and a lower water pressure (which cares for waves of water pressure, the potential action) leads to the end of the pants.

3.4.3.3 Additive function

An additive function is an arithmetic f(n) function in a positive integer that the product function is the sum of its functions when a and b are compressed. F(a) + f(b) holds a completely additive function even when they are not co-prime to all positive integral parts a or b if f(b) holds f(a)+ f(b). In this sense also, analogy with fully multiplicative functions is used with complete additive. F(1)=0 if f is a full additive feature. Each fully additive function is an additive, but not the other way around.

3.4.3.4 Activation function

The threshold or transfer feature is also known as activation functions. The activating functions have been used to transform neuron activation levels into output signals. Numerous activation functions are available in the neural network. Different function types are identity function, step function, linear portion and sigmoid function.

Identity activation function:

The activation function of identity is also referred to as "liner activation." The Network Activation function can be shown easily to fit a line regression model of the form if the ID is used in the network $Y_i = B_0 + B_1 + \cdots + B_k B_k$ where $x_1, x_2, \ldots\ldots\ldots, x_k$ are the k network inputs, Y_i is the i-

th network output $B_1, B_2, \ldots\ldots, B_k$ are the coefficients in the regression equation. Consequently, a neural network with identity activation used in all its sensors is uncommon to be found.

Sigmoid activation function:

Nonlinearity in the model is used in the artificial neural network sigmoid functions. The result of a linear combination of its input signals is calculated by a network neuroelement using a sigmoid function. The sigmoid function makes an interface between the product and itself easier and more popular in the Neural Network.

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{3.6}$$

Sigmoid function results are generally used in learning algorithms. The Sigmoid graph is shaped as' S'. This function is defined as an expanding function which is commonly used for neural artificial network development. Sigmoid is a function that strictly increases, and shows a balance between linear and nonlinear functions.

One - polar – is the sigmoid function.

Step function:

This is a unipolar threshold, known as.

$$\varphi(v) = \begin{cases} 1 \; if \; v \geq 0 \\ 0 \; if \; v > 0 \end{cases} \tag{3.7}$$

The neuron K output with a threshold is

$$y(k) = \begin{cases} 1 \; if \; v_k \geq 0 \\ 0 \; if \; v_k > 0 \end{cases} \tag{3.8}$$

$v_k \; is \; the \; induced \; local \; field \; of \; the \; neuron$

$$v_k = \sum_{j=1}^{m} W_{kj} X_j + b_k \tag{3.9}$$

When the neuron output is 1 when the local neuron field induced is not - negative, the neuron output is 0.

Piece Wise Linear Function

It can be defined as a unipolar function

$$\varphi(v) = \begin{Bmatrix} 1, & v \geq +1/2 \\ v, +\frac{1}{2} > & v > -1/2 \\ 0, & v \leq -1/2 \end{Bmatrix} \quad (3.10)$$

When it is expected that the amplification factor is within the linear area

1. The specific circumstances of linear functions are
   - If the linear operating area is maintained without saturation, a linear combiner is produced.
   - If the linear region's amplification factor is infinitely large, it reduces to a threshold feature.

Learning Rules in neural network

In the neural network there are many distinct studies, generally separated into 3 classifications.

- Supervised Learning
- Unsupervised Learning

- *Supervised Learning*

Training sets are available for supervised learning. This type of rule includes a set of examples with proper network behavior. The inputs are given as a training in controlled learning and the expected results are achieved. Parameters in this type of study are adjusted step by step by error signal; parameters are adjusted step by step by step by error signal.

A number of examples (trainings set) together with correct network conduct are provided for the learning rule.

$$\{x1, d1\}, \{x2, d2\}, \dots\dots\dots, \{xn, dn\} \quad (3.11)$$

The network input is xn in this case and dn is the required destination input. The output is generated by input. In order to make network outputs more exact, the Study rule is employed to change network biases and weights.

We commit ourselves with supervised learning to give the system the desired answer (d) when the entry is implemented. The distance between the real response and the desired response is used to correct the network parameter externally. For example, the error can be used to change weighing in the study of input patterns or circumstances where the answer to the error is recognized. For the learning mode, the training set, several input and output patterns are needed.

- *Unsupervised learning*

In unexpected learning, self-organized learning is also known. Objective output is not available in uncontrolled learning. In that case, only network input changes weights and biases. For pattern reorganization, unattended study grouping is used. The answer required is not known in unattended learning, therefore explicit error information cannot be utilized for improving network behavior. Information of this type is not available to correct the wrong answers so that learning has to be done based on observations of marginalized or unknown responses to the data.

The algorithms in unchecked learning use redundant row data, which have no etiquette for class membership or associations. In order to identify its parameters in this way, the network needs to detect any existing patterns, properties, regulations, etc. Unattended study means learning without the teacher because it is not necessary for the teacher to participate, but the teacher must set objectives. Feedback on neural networks is important as well. Feedback is called progressive learning, which for uncontrolled learning is very important.

3.4.3.5 Outputs

This layer contains also neurons that generate and display the final neural network outputs at the previous layers. Thus the artificial neural networks' principal architectures can be divided, taking into consideration the connection and the structure of the neuronal disposition.

3.4.4 Classification of Artificial Neural Networks

3.4.4.1 Artificial neural networks according to constructions

a. Single layer feed forward network

A neural network with a source node, but not one single feed or an acyclic network that projects a neural output level. The individual layer refers to the output layer in one network layer calculation node Figure 3.2.

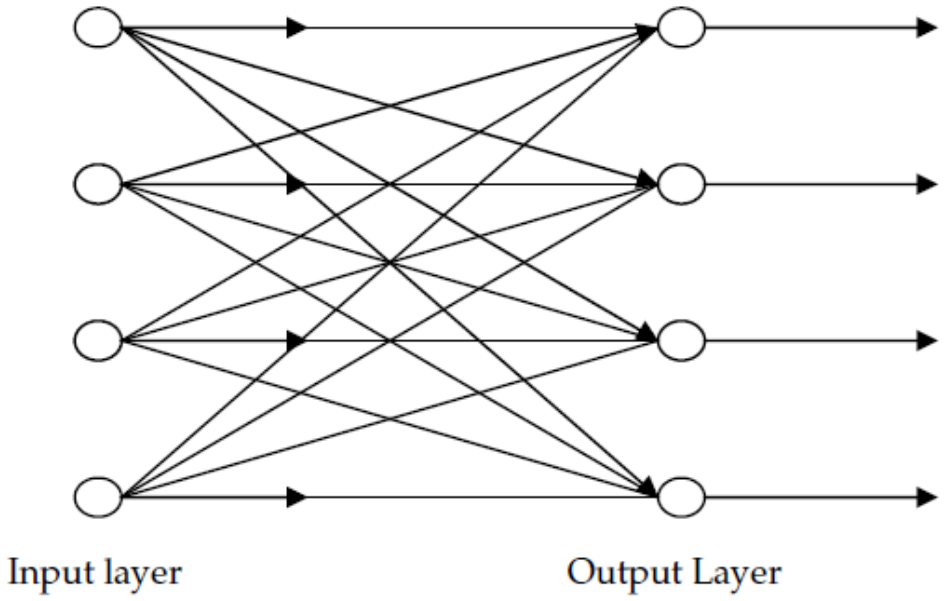

Figure 3.7: Feed Forward Single Layer Neural Network [2]

b. Multilayer feed forward network

The system consists of at least one cache layer known as clad neurons or clad unit hubs. The ability of clad neurons is to work between external information and system output and to get separate insights on a higher demand. The source hubs flag for the neurons in the second layer is given in the system input layer. The second layer of output signals, etc. are the third layer of inputs. The general reaction of the system to the actuation design provided by sources in the main layer of the info is neuron output motions in the system yield level.

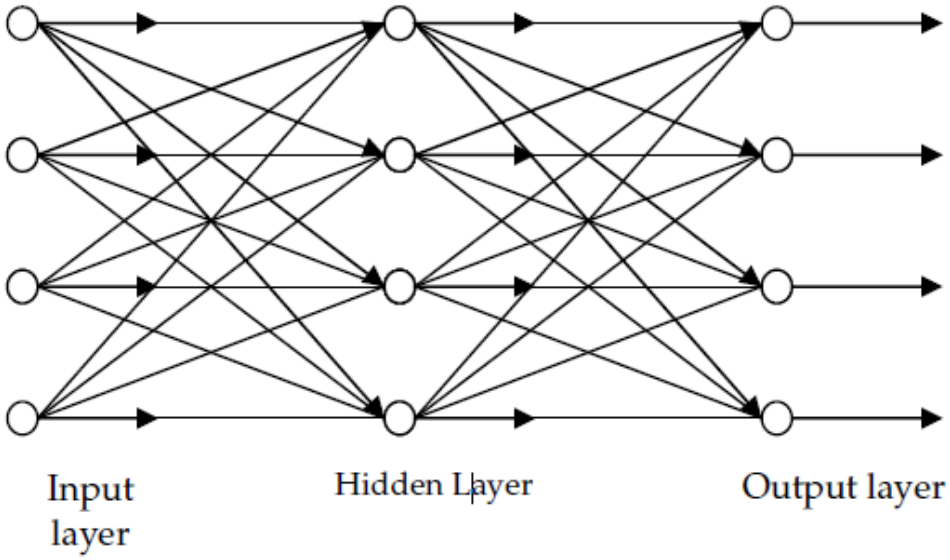

Figure 3.8: Multilayer Feed Forward Network [2]

Short feeding network characterization:

1. In general, activation is provided via ' hidden layers ' from input to output but many other architectures exist.
2. Static input - output mappings are implemented by mathematics.
3. Most popular algorithm for backpropagation supervised training:
4. Has been useful as an approximation of nonlinear functions and as a classification model in many practical applications.

c. Recurrent network

The repetitive system in the figure is known as a forward neural system of something like the input circle, which includes at least one shrouded layer. 2.3.2. The first time. Critics could be your own critique, i.e. if your own information returns to neuronal yield. The use of unit deferring components, which leads to a unique conduct, has sometimes been criticised, since the neural system has non-direct units.

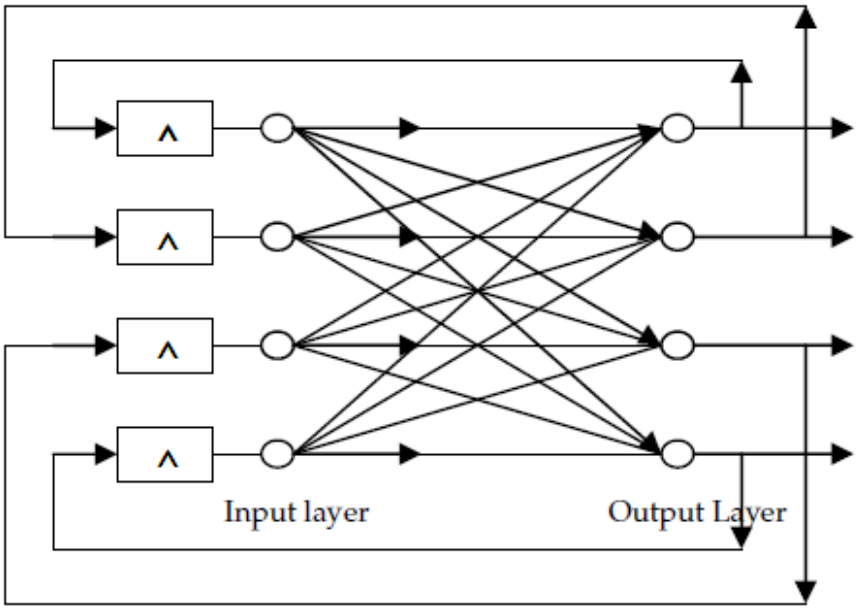

Figure 3.9: Recurrent Network [2]

Other network types are available; Delta - bar - delta, hop field, quantifying vectors, counter propagation, probabilist, hamming, memory Boltzmann, spatium - temporal patent, adaptive resonance, auto - order, recirculation etc.

The cyclic path of synaptic connections is a recurring neural network. Basic characteristics:

1. Recurring are all biological neural networks

2. They implement dynamic systems mathematically

3. Various types of algorithms, no clear winner, are known

4. In general, theoretical and practical problems have prevented practical applications up to now.

3.4.4.2 Artificial neural networks according to learning algorithms

If a network for a specific application is structured, then it is ready for training. Initial weights are randomly selected at the start and beginning of the training. There are two approaches; unattended and controlled.

A. Supervised training

Inputs and outputs are given for monitored training. The outcome is likened to the outcomes. The data is processed by the network. The system spreads errors to adjust network weights. This process takes place time and again when weight is constantly changed. The same data set is processed many times during networking training, as the weight of connections is always improved. The training dataset is called the ' training set'.

Sometimes a network can never learn. This can take place as there are no specific information in the input data to produce the required output. Networks also do not converge when there are not enough data to enable complete learning. In order to retain part of the data as a test, there should ideally be sufficient data. Multiple nodes in many layered networks can store data. To monitor the network, the supervised training must retain data for the system to test after training of a system, to determine whether the system simply saves its data.

If the problem cannot simply be solved by the network, the designer must examine inputs and outputs, layer numbers, numeric layer components, layer connections, capacity transfers and training, and the initial weights. The training rules govern another part of the creativity of the designer. Many laws (algorithms) are used in the training session to make the required weight adjustment feedback. The most common method is back-propagation. It is a conscious analysis and not only a technique to prevent overtraining of the network. General statistical developments in the data are the initial configuration of the artificial neural network. It then' learns' about other aspects of the data which are generally falsified.

The weights can be frozen upon request, if finally the system is properly trained and no further training is necessary. This network can then be converted on certain systems into hardware. During production, other systems do not lock in, but continue learning.

B. Unsupervised or adaptive training

Uncontrolled (learning) is the other kind of training. The network has inputs but not desired outputs in this kind of way. The system must then determine its own functions in order to group the input data. Often this is termed autonomy or adaptation. These networks do not adjust their weight using external influences. Their performance is instead monitored internally. The networks seek the

regularity or trends of the input signal and adjust it to the network functions. Although the network still needs to be able to provide information on how to organize itself without knowing whether or not this is right. These data are incorporated into the network's topology and study rules. The cooperation between the processing element clusters may be emphasized by an unchecked learning algorithm. The clusters would work together under such a scheme. If any external input activated a node within the cluster, it could increase the whole activity of the cluster. Furthermore, it can inhibit the whole cluster if external input to the cluster nodes has been decreased. The competition between elements of processing might also provide a basis for learning. Competitive cluster training can improve individual groups' response to special incentives. In that sense, these groups would be linked and a special appropriate response would be provided. The weight of the winning processing element is normally only updated when the learning contest is effective. Unattended learning is not well understood at the present time and a lot of research is in progress.

C. Learning laws (algorithms)

There are many common uses for learning laws. The majority are a variation of ' Hebb's rule ' which is the best - known and oldest.

Hebb's Rule: In the Compatibility Organization, Donald Hebb introduced this. The fundamental rule of thumb is that the weight should be strengthened if the neuron comes from another nerve and both neurons are very active (the same sign).

Hopfield Law: Increase connectivity weight by the study rate and when both active and inactive the desired outcomes and inputs.

The Delta Rule: The concept is to constantly change the force of the input links to reduce the difference between the required output and the element's real output (delta).

The Gradient Descent Rule: This is similar to the Delta rule because the transfer function derivative is still used to change the delta bug prior to the application of connection weights. However, a proportional additional constant associated with the learning rate accompanies the final weight - based change factor.

Kohonen's Law: This allows processing parts to acquire or update their weight. The strongest item is declared the winner and its competitors can be inhibited and its neighbors excited. Only the winner can adjust his weight and only the winner plus neighbors can adjust it.

3.4.5 Artificial Neural Network's Output Calculations

Sensitivity, septicity and accuracy are preferred statistics for determining the performance of a classifier. Susceptibility is the estimation rate for patients with epileptic diseases, specialty is the estimation speed for healthy people and accuracy is true. Equality. These statistical numbers are calculated using (36), (37) and (38).

$$Sensitivity = \frac{TP}{TP+FN} \tag{3.12}$$

$$Specificity = \frac{TN}{TN+FP} \tag{3.13}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{3.14}$$

In the above equations, the number of TP - diagnosed epileptic patients, the total number of normal epileptic patients, for whose epileptic disease was diagnosed, and the total number of normal epileptic patients, for which FN was diagnosed.

3.5 BACKPROPAGATION

Backpropagation is a way to calculate the gradient needed for network weight calculation in artificial neural networks. Backpropagation means "retroactive error propagation" as an error is calculated at the output and reversed across all layers of the network. It is often used for the formation of deep neural networks.

Discharge is a common Delta rule application on multilayer feed systems that allows the measurement of iterating gradients by the chain rule of every layer. The Gauss algorithm of Newton is closely related and forms part of ongoing neural research.

The reverse propagation is a special case for an automatic differentiation technique. In the course of learning, back propagation is usually used for adjusting the weight of neurons by means of the downgrading algorithm for calculating the loss function degree.

Take the following network in one case, e.g.: (1), 1,0 and x1 and x2 inputs are 1 and 1. The result is parabolic if the y output is tracked for E error on the axis. The lowest y-output is that which reduces e-error. In a single workout case, the minimum affects the horizontally axis, which means the error is zero and the network can generate an output that matches the desired output exactly. In order to optimize searches of fewer errors, this reduces the problem of mapping inputs to outputs.

$$y = x_1 w_1 + x_2 w_2 \tag{3.15}$$

where x1 and x2 weights are linked to the input unit connection with the output unit. The error depends, therefore, on the weight of the neuron input that must be changed in this entire learning network. The result is a parabola with a separate horizontal axis with a vertical axis defect in every weight. For an elliptical k+1 neuron with a k weight, the same tract would be required.

Any complex system can be abstracted merely or at least divided into its abstract basic parts. Multiple single layers accumulate creating complexity. This article explains how neural networks function with the simplest abstraction. We attempt to decrease NN's machine learning mechanism to its basic abstract parts. We attempt to make use of the few mathematical equations and code feasible and concentrate solely on abstract ideas as opposed to other articles explaining neural networks. A monitored neural network can be described as black boxes in the largest and easiest abstract representation using 2 techniques:

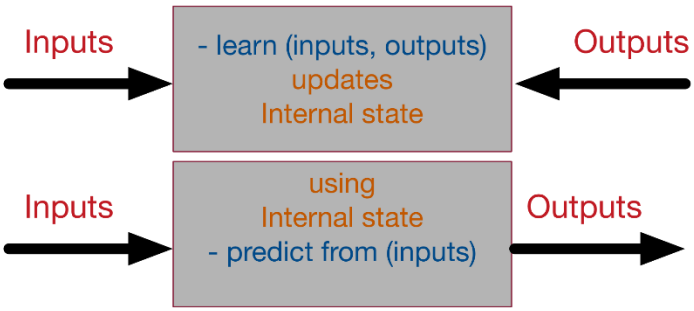

Figure 3.10: Backpropagation Training Method [24]

Learning as a problem of optimization

It first of all aids us in getting a mathematical derivation of the backpack propagation algorithm to understand the link between the actual neuron output and the exact output of a certain training example. There are two inputs, one output device and no hidden device in the neural network. In contrast to most neural networks, each neuron uses a linear output (note 1), which is the weighted amount. At first, before training, weights are altered. In this instance, the neuron will learn from examples consisting of the tuples x1, x2, t where x1 and x2 are the network inputs and t is the correct output (network output should be produced when these inputs are trained). The first network calculates a y output which can differ from t (random weights) when x1 and x2 is specified. Squared error measurement is a common method for measuring differences between the expected t and the current y output:

$$E = (t - y)^2 \tag{3.16}$$

where E is an error or discrepancy.

Consider the network for one training case, for instance: (1), 1,0 and therefore x1 and x2 inputs are 1 and 1. The result is a parabolic when the y output on the vertical axis of the horizontal axis is tracked against the E error. The y output that minimizes the E error is the minimal parabola. The minimum in one workout case affects the horizontal axis, which means that the error is zero, and a y output that matches the desired t output can be generated by the network. In order to optimize the search for the least fault, the inputs to outputs are reduced.

The neuron output depends, however, on the sum of all the inputs weighted:

$$y = x_1 w_1 + x_2 w_2 \tag{3.17}$$

where the weights of x1 and x2 are connected to the output unit by the input unit connection. The error depends therefore on the incoming weights of the neuron, which must be changed over the network ultimately in order to allow learning. The result is a parable bowl with a separate horizontal axis and a vertical axis fault with each weight. For the same tract, an elliptical paraboloid k+1 dimensions would be required for a neuron with k weights.

3.6 TRAINING OF ARTIFICIAL NEURAL NETWORKS

Artificial neural biopsy systems are neural networks or network systems that make up the animal brain. These systems learn tasks through the following examples, normally without task-specific programming (a step towards performance improvements). The analysis of flames that manually mark "cat" or "no-cat," for example fur, tails and fur, do not know cats-like face, can help you to identify pictures that contain cats and use the results to identify them in other images. Rather, from the learning material they process, they create their own set of appropriate features. An ANN is based on the artificial neuronal collection of a unit or node (similar to the biology of the brain neurons of the animal). Any link between artificial cells (such as a synapse) can signal it. The signaling artificial neuron can continue and signal the artificial neurons linked.

In the common ANN implementation, the signal is an actual value in the relation between artificial neurons, and the output is calculated by a nonlinear input sum function from each artificial neuron. Artificial neurons and connections usually weigh in accordance with learning. The weight enhances or reduces the power of the connection signal. An artificial neuron threshold can only be present when the aggregate signal crosses this threshold. Artificial neurons are usually arranged in layers. Inputs can be converted into different types from different layers. Signals pass from the first level to the last layer (output), maybe several times after the layers are transmitted.

The ANN approach was originally designed to solve problems such as the human brain. Over time, the focus was on matching certain mental skills that contribute to biodiversity. ANNs are used to carry out various duties, like computer vision, voice acknowledgement, machine translation, social network filtering, video games, playboard, and medical diagnosis.

3.6.1 Training Artificial Neural Networks with Backpropagation.

The method of gradient decline contains derivatives for network weight of the squared error function. This is usually done in exchange. The square error function is as follows:

$$E = \frac{1}{2}(t - y)^2 \tag{3.18}$$

where

E is the error,

t is the target output for the training, and

y is the output of neurons.

The 1/2 factor in the exponent is distinguished. Then the term is increased by an arbitrary learning rate. Whether a continuous coefficient is implemented now is not important.

For each neuron *j*, its output o_j is defined as

$$o_j = \varphi\left(net_j\right) = \varphi\left(\sum_{k-1}^{n} w_{kj} o_k\right) \tag{3.19}$$

The neuron netj input is the weighted amount of past neuron outputs. The ok of the input layer is just a network xk input. The wkj variable indicates weight between the neuronal and j.

β is non - linear and distinctive with the activation function. The logistic function is a commonly used activation feature:

$$\varphi(z) = \frac{1}{1+e^{-z}} \tag{3.20}$$

which has a convenient derivative of:

$$\frac{d\varphi}{da} = \varphi(a)(1 - \varphi(a)) \tag{3.21}$$

Identifying the error derivative

Twice by the chain rule is calculated the partial derivation of an error in relation to a wij:

$$\frac{\vartheta E}{\vartheta w_{ij}} = \frac{\vartheta E}{\vartheta o_j}\frac{\vartheta o_j}{\vartheta net_j}\frac{\vartheta net_j}{\vartheta w_{ij}} \tag{3.22}$$

In the last right side factor of the foregoing w_{ix} depends only one term in the sum net_h

$$\frac{\vartheta E}{\vartheta net_j} = \frac{\vartheta}{\vartheta w_{ij}}\varphi\left(\sum_{k-1}^{n} w_{kj}o_k\right) = \frac{\vartheta}{\vartheta w_{ij}} w_{ij}o_i = o_i \tag{3.23}$$

When the first layer of the neuron after the input layer is, o_i is only X_I.

With respect to the input, only the active function parameters are a derivative of neuron j output (if logistics function is used):

$$\frac{\vartheta o_j}{\vartheta net_j} = \frac{\vartheta}{\vartheta net_j}\varphi\left(\vartheta net_j\right) = \varphi\left(\vartheta net_j\right)(1 - \varphi\left(\vartheta net_j\right)) \tag{3.24}$$

That is why a differentiated activation function is required for back propagation. (The activation function of Rectified Linear Unit (ReLU), which cannot be differentiated by 0, is very popular in the recent past, for instance in AlexNet)

The first factor to evaluate is simple, because then oj = y and if the neuron is in the output layer.

$$\frac{\vartheta E}{\vartheta o_j} = \frac{\vartheta E}{\vartheta y} = \frac{\vartheta}{\vartheta y}\frac{1}{2}(t - y)^2 = y - t \tag{3.25}$$

But if j is within an arbitrary inner network layer, it is less obvious to determine derivative E with respect to the oj.

$$\frac{\vartheta E(o_j)}{\vartheta o_j} = \frac{\vartheta E(net_1, net_2 \dots net_u)}{\vartheta o_j} \tag{3.26}$$

Taking into account E as a function where all neurons are inputs (L=1, 2 ... u)

$$\frac{\vartheta E}{\vartheta o_j} = \sum_{\partial \in L}\frac{\vartheta E}{\vartheta net_j}\frac{\vartheta net_j}{\vartheta o_j} = \sum_{\partial \in L}\frac{\vartheta E}{\vartheta o_j}\frac{\vartheta o_j}{\vartheta net_j} w_{j\partial} \tag{3.27}$$

Thus, if all derivatives are known with respect to oj — that which is closer to the neuron output — the derivative with regards to oj can be calculated. Combining everything:

$$\frac{\vartheta E}{\vartheta w_{ij}} = \partial_j o_i \tag{3.28}$$

In 1960 Henry J. Kelley and 1961 Arthur E. Bryson derived the basics of continuous back-propagation from control theory. They have used dynamic programming principles. A simple derivation based only on the chain rule was published in 1962 by Stuart Dreyfus. This is a multi - phase method used to optimize dynamic systems by Bryson and Ho in 1969.

In the beginning of the 1960's, background propagation was derived by a number of researchers and launched by Seppo Linnainmaa as early as 1970, for instance Arthur E. Bryson and Yu-Chi Ho, among the researchers of the 1960s. After a careful analysis in his PhD dissertation in 1974, the first person to propose use for neural networks in the United States was Paul Werbos. The prize for work of David E. Rumelhart was awarded in 1986 to Geoffen E. Hinton, Ronald J. Williams & James McClelland.

The general automated differentiation (AD) method of discernible connected networks for nestled differentiating functions was issued by Linnainmaa in 1970. This is the backbone that even with sparsely networked networks is efficient. In 1973 Dreyfus used back-propagation to adjust parameters of the controller in accordance with error gradients. Werbos noted in 1974 the potential for this principle to be applied on neural artificial networks, and used the Linnainmaa AD method in 1982 in its present application on neural networks.

In 1986, Rumelhart, Hinton and Williams showed that in hidden layers in neural networks the process could generate useful internal images of the inbound data. Wan was the first winner in 1993 of an international competition for pattern recognition. In the 2000s it was disadvantageous, but it came back in 2010 with cheap, high-performance computer systems based on GPUs. In particular, research was conducted into linguistic structures, where connectivity models could explain a number of first language and second language learning phenomes using this algorithm.

4. IMPLEMENTATION AND RESULTS

This chapter describes the extensive simulation results for the methods studied in this project which these methods are Deep Belief Network (DBN) and Artificial Neural Network (ANN) on student performance dataset.

4.1 COMPARATIVE RESULTS

We used the 80 % training and 20 % testing scenario with varying number neurons of the hidden layers is 23 and 5 respectively for each layer which the number of layer that been used in this study were 2 hidden layers.

4.2 STUDENT PERFORMANCE DATASET RESULTS

This experiment proves the performance of the DBN against ANN-BP algorithms, the construction of ANN-BP and DBN algorithms consists mainly of 30 inputs, two hidden layers of 23 and 5 neurons respectively, and output layer is consisted of one output, where in this study sigmoid function have been used.

Figure 4 shows that the classification accuracy of DBN are much better than ANN-BP, has obtained that accuracy with only 500 iterations while ANN-BP needed 500 iterations too. Furthermore, comparing the MSE of both algorithms shows that the MSE of DBN is much less than ANN-BP. It also converges faster than the ANN-BP.

This dataset is originally taken from UCI machine learning repository which contains about 395 instances with 30 features and 3 results attributes (G1, G2, and G3) which in this proposed study G3 was used as a target output where the if the instance’s target is less than 10 the output is 0 else the output is 1.

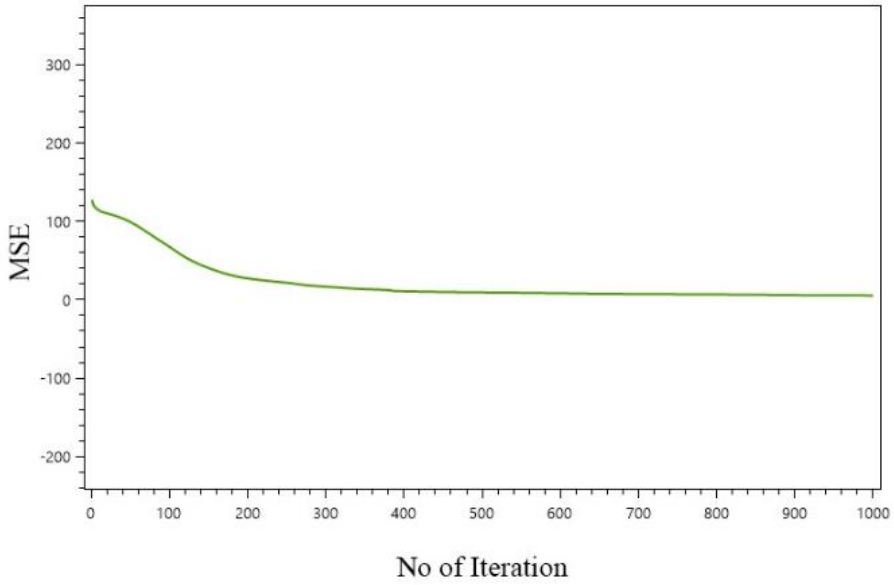

Figure 4.1: The results of the implementation using DBN

Table 4.1: Performance Comparison between DBN and ANN

Method	Deep Belief Network	Artificial Neural Network
MSE	0.005526575057776	0.01957278250572732
TRAINING ACCURACY	99.4473424942224	0.98042721749427268
TESTING ACCURACY	100	95.75
SPECIFICTY	100	93.3333333
SENSETIVITY	100	100

Table 4.2: TP, FP, TN and FN between DBN and ANN

Method	Deep Belief Network	Artificial Neural Network
TP	53	49
TN	27	27
FP	0	0
FN	0	4

Table 4.1 contains the results for the methods that are used in this study according to Figure 4.1 which shows the training error and accuracy, testing error and accuracy (using MSE as a fitness function) and both the specificity and sensitivity for training and testing.

5. CONCLUSION

We introduced a checked ranking approach to student performance prediction using the Deep Belief Networks (DBNs). In comparison to the Artificial Neural Network, DBNs are effective as discriminatory. In this study, in order to forecast students' performance, DBN classification method is used on student dataset. The current study demonstrates that the students' academic performance does not always depend on their own efforts. Other factors have a major influence on student performance as demonstrated by our research. This proposal will enhance insights into current approaches. In terms of classification performance DBN has higher quality in classifying student according to their features compared to ANN.

5.1 SUGGESTIONS

In this research work, we perform the ANN and DBN on student performance dataset using the research datasets. For future suggestions we want to perform below points:

- The current evaluation is based on single class problems for classification, however under the real time scenario this will not be the case always. So we suggest evaluating the performance of proposed model using multi-class datasets.

Second point is, the consideration of more real time will be the interesting future direction for this research work.

REFERENCES

[1] G. Beni and J. Wang, “Swarm intelligence in cellular robotic systems,” in Robots and Biological Systems: Towards a New Bionics? vol. 102 of NATO ASI Series, pp. 703–712, Springer, Berlin, Germany, 1993.

[2] X. Yao, “Evolving artificial neural networks,” Proceedings of the IEEE, vol. 87, no. 9, pp. 1423–1447, 1999.

[3] E. Alba and R. Martí, Metaheuristic Procedures for Training Neural Networks, Operations Research/Computer Science Interfaces Series, Springer, New York, NY, USA, 2006.

[4] J. Yu, L. Xi, and S. Wang, “An improved particle swarm optimization for evolving feedforward artificial neural networks,” Neural Processing Letters, vol. 26, no. 3, pp. 217–231, 2007.

[5] M. Conforth and Y. Meng, “Toward evolving neural networks using bio-inspired algorithms,” in IC-AI, H. R. Arabnia and Y. Mun, Eds., pp. 413–419, CSREA Press, 2008.

[6] Y. Da and G. Xiurun, “An improved PSO-based ANN with simulated annealing technique,” Neurocomputing, vol. 63, pp. 527–533, 2005.

[7] X. Yao and Y. Liu, “A new evolutionary system for evolving artificial neural networks,” IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 694–713, 1997.

[8] D. Rivero and D. Periscal, “Evolving graphs for ann development and simplification,” in Encyclopedia of Artificial Intelligence, J. R. Rabuñal, J. Dorado, and A. Pazos, Eds., pp. 618–624, IGI Global, 2009.

[9] H. M. Abdul-Kader, “Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting34,” International Journal of Computer Science and Network Security, vol. 9, no. 3, pp. 92–99, 2009.

[10] K. K. Kuok, S. Harun, and S. M. Shamsuddin, “Particle swarm optimization feedforward neural network for modeling runoff,” International Journal of Environmental Science and Technology, vol. 7, no. 1, pp. 67–78, 2010.

[11] B. A. Garro, H. Sossa, and R. A. Vázquez, “Back-propagation vs particle swarm optimization algorithm: which algorithm is better to adjust the synaptic weights of a feed-forward ANN?” International Journal of Artificial Intelligence, vol. 7, no. 11, pp. 208–218, 2011.

[12] B. Garro, H. Sossa, and R. Vazquez, “Evolving neural networks: a comparison between differential evolution and particle swarm optimization,” in Advances in Swarm Intelligence, Y. Tan, Y. Shi, Y. Chai, and G. Wang, Eds., vol. 6728 of Lecture Notes in Computer Science, pp. 447–454, Springer, Berlin, Germany, 2011.

[13] B. A. Garro, H. Sossa, and R. A. Vazquez, “Design of artificial neural networks using a modified particle swarm optimization algorithm,” in Proceedings of the International Joint Conference on Neural Networks (IJCNN '09), pp. 938–945, IEEE, Atlanta, Ga, USA, June 2009.

[14] B. Garro, H. Sossa, and R. Vazquez, “Design of artificial neural networks using differential evolution algorithm,” in Neural Information Processing. Models and Applications, K. Wong, B. Mendis, and A. Bouzerdoum, Eds., vol. 6444 of Lecture Notes in Computer Science, pp. 201–208, Springer, Berlin, Germany, 2010.

[15] B. A. Garro, H. Sossa, and R. A. Vazquez, “Artificial neural network synthesis by means of artificial bee colony (abc) algorithm,” in Proceedings of the IEEE Congress on Evolutionary Computation (CEC '11), pp. 331–338, IEEE, New Orleans, La, USA, June 2011.

[16] R. C. Eberhart, Y. Shi, and J. Kennedy, Swarm Intelligence, The Morgan Kaufmann Series in Evolutionary Computation, Morgan Kaufmann, Boston, Mass, USA, 1st edition, 2001.

[17] M. Chen, “Second generation particle swarm optimization,” in Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08), pp. 90–96, Hong Kong, June 2008.

[18] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in Proceedings of the Congress on Evolutionary Computation (CEC '99), vol. 3, IEEE, Washington, DC, USA, July 1999.

[19] M. Løvberg, T. K. Rasmussen, and T. Krink, “Hybrid particle swarm optimizer with breeding and subpopulations,” in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01), pp. 469–476, Morgan Kaufmann, San Francisco, Calif, USA, July 2001.

[20] N. Higashi and H. Iba, “Particle swarm optimization with Gaussian mutation,” in Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03), pp. 72–79, IEEE, April 2003.

[21] S. Mohais, R. Mohais, C. Ward, and C. Posthoff, “Earthquake classifying neural networks trained with random dynamic neighborhood PSOs,” in Proceedings of the 9th Annual Genetic and Evolutionary Computation Conference (GECCO '07), pp. 110–117, ACM, New York, NY, USA, July 2007

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, D. E. Rumelhart, J. L. McClelland, and PDP Research Group, Eds., pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.

[23] J. A. Anderson, An Introduction to Neural Networks, The MIT Press, 1995.

[24] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” Proceedings of the IEEE, vol. 78, no. 10, pp. 1550–1560, 1990.

[25] D. N. A. Asuncion, UCI machine learning repository, 2007.

[26] R. A. V. E. De Los Monteros and J. H. Sossa Azuela, “A new associative model with dynamical synapses,” Neural Processing Letters, vol. 28, no. 3, pp. 189–207, 2008.

[27] B. A. Garro and R. A. Vázquez, “Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms,” *Computational Intelligence and Neuroscience*, 2015.

[28] V. G. Gudise and G. K. Venayagamoorthy, “Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks,” in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS’03 (Cat. No.03EX706)*, 2003, pp. 110–117.

[29] “Pima Indians Diabetes Database.” [Online]. Available: https://www.kaggle.com/uciml/pima-indians-diabetes-database. [Accessed: 20-Dec-2018].

[30] BAGLEY, J. D. The Behavior of Adaptive Systems Which Employ Genetic and Correlative Algorithms. PhD thesis, University of Michigan, Ann Arbor, 1967.

[31] BAUER, P., BODENHOFER, U., AND KLEMENT, E. P. A fuzzy algorithm for pixel classification based on the discrepancy norm. In Proc. 5th IEEE Int. Conf. on Fuzzy Systems (New Orleans, September 1996), vol. III, pp. 2007–2012.

[32] BODENHOFER, U. Tuning of fuzzy systems using genetic algorithms. Master’s thesis, Johannes Kepler Universit¨at Linz, March 1996.

[33] BODENHOFER, U., AND BAUER, P. A formal model of interpretability of linguistic variables. In Interpretability Issues in Fuzzy Modeling, J. Casillas, O. Cord´on, F. Herrera, and L. Magdalena, Eds., vol. 128 of Studies in Fuzziness and Soft Computing. Springer, Berlin, 2003, pp. 524–545.

[34] BODENHOFER, U., AND HERRERA, F. Ten lectures on genetic fuzzy systems. In Preprints of the International Summer School: Advanced Control—Fuzzy, Neural, Genetic, R. Mesiar, Ed. Slovak Technical University, Bratislava, 1997, pp. 1–69.

[35] BODENHOFER, U., AND KLEMENT, E. P. Genetic optimization of fuzzy classification systems — a case study. In Computational Intelligence in Theory and Practice, B. Reusch and K.-H. Temme, Eds., Advances in Soft Computing. Physica-Verlag, Heidelberg, 2001, pp. 183–200.

[36] BONARINI, A. ELF: Learning incomplete fuzzy rule sets for an autonomous robot. In Proc. EUFIT’93 (1993), vol. I, pp. 69–75. 121 122 BIBLIOGRAPHY

[37] BONARINI, A. Evolutionary learning of fuzzy rules: Competition and cooperation. In Fuzzy Modeling: Paradigms and Practice, W. Pedrycz, Ed. Kluwer Academic Publishers, Dordrecht, 1996, pp. 265–283.

[38] BONARINI, A. Anytime learning and adaptation of hierarchical fuzzy logic behaviors. Adaptive Behavior 5 (1997), 281–315.

[39] BULIRSCH, R., AND STOER, J. Introduction to Numerical Analysis. Springer, Berlin, 1980.

[40] E. Augustin, M. Carre, E. Grosicki, J.-M. Brodin, E. Geoffrois, and F. Preteux. Rimes evaluation campaign for handwritten mail processing. In IWFHR, pages 231–235, 2006.

[41] A. B. Bernard, F. Menasri, R. H. Mohamad, C. Mokbel, C. Kermorvant, and L. Sulem. Dynamic and contextual information in HMM modeling for handwritten word recognition. IEEE Trans. PAMI, 33:2066–2080, 2011.

[42] A. Britto, R. Sabourin, F. Bortolozzi, and C. Suen. A two-stage HMM-based system for recognizing handwritten numeral strings. In ICDAR, pages 396–400, 2001.

[43] Y. Chherawala, P. P. Roy, and M. Cheriet. Feature design for offline Arabic handwriting recognition: handcrafted vs automated?, 678-682. In ICDAR, 2013.

[44] P. Doetsch, M. Hamdani, H. Ney, A. Gimenez, J. Andres-Ferrer, and A. Juan. Comparison of Bernoulli and Gaussian HMMs using a vertical repositioning technique for off-line handwriting recognition. In ICFHR, page 37, 2012.

[45] A. El-Yacoubi, R. Sabourin, C. Y. Suen, and M. Gilloux. An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. IEEE T-PAMI, 21:752–760, 1999.

[46] H. Fujisawa. Forty years of research in character and document recognition an industrial perspective. Pattern Recognition, 41:2435–2446, 2008.

[47] A. Graves. RNNLIB: A recurrent neural net-work library for sequence learning problems. http://sourceforge.net/projects/rnnl/.

[48] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. IEEE T-PAMI, 31:855–868, 2009.

[49] E. Grosicki and H. E. Abed. ICDAR 2009 hand-writing recognition competition. In ICDAR, page 13981402, 2009.

[50] S. Haykin. Neural Networks: A Comprehensive Foundation. Prentice Hall, 1998.

[51] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. Neural Comput., 18:1527–1554, 2006.

[52] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. IEEE Trans. PAMI, 20(3):226–239, 1998.

[53] A. L. Koerich, R. Sabourin, and C. Y. Suen. Recognition and verification of unconstrained handwritten words. IEEE Trans. PAMI, 27:1509– 1522, 2005.

[54] U. V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition sys-tem. IJPRAI, 15:65–90, 2001.

[55] F. Menasri, J. Louradour, A. L. Bianne-Bernard, and C. Kermorvant. The A2iA French handwriting recognition system at the RIMES-ICDAR2011 competition. In DRR XIX, 2012.

[56] R. Mohamad, L. Likforman-Sulem, and C. Mokbel. Combining slanted-frame classifiers for im-proved HMM-based Arabic handwriting recognition. IEEE Trans. PAMI, 31:1165–1177, 2009.

[57] M. P. Schambach and S. F. Rashid. Stabilize sequence learning with recurrent neural networks by forced alignment. In ICDAR, pages 1270–1274, 2013.

[58] A. Vinciarelli and S. Bengio. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. IEEE Trans. PAMI, 26:709–720, 2004.

[59] S. Young. The HTK Book, Version 3.4. Cambridge Univ. Eng. Dept., 2006.

[60] N. R. Stepanek and B. Dorn, “Automated Analysis of Lecture Video Engagement Using Student Posts,” in *Artificial Intelligence in Education*, 2017, pp. 565–569.

[61] R. L. U. Cazarez and C. L. Martin, “Neural Networks for Predicting Student Performance in Online Education,” *IEEE Latin America Transactions*, vol. 16, no. 7, pp. 2053–2060, Jul. 2018.

[62] G. Zhao *et al.*, “Deep convolutional neural network for drowsy student state detection,” *Concurrency and Computation: Practice and Experience*, vol. 30, no. 23, p. e4457, 2018.

[63] C. Lee *et al.*, “PSO-Based Fuzzy Markup Language for Student Learning Performance Evaluation and Educational Application,” *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2618–2633, Oct. 2018.

[64] Y. Ma, C. Cui, X. Nie, G. Yang, K. Shaheed, and Y. Yin, “Pre-course student performance prediction with multi-instance multi-label learning,” *Sci. China Inf. Sci.*, vol. 62, no. 2, p. 29101, Feb. 2019.

[65] P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.