Metaheuristic Algorithms and Neural Networks in Hydrology

Metaheuristic Algorithms and Neural Networks in Hydrology

Edited by

Kuok King Kuok and Md Rezaur Rahman

Cambridge Scholars Publishing

Metaheuristic Algorithms and Neural Networks in Hydrology

Edited by Kuok King Kuok and Md Rezaur Rahman

This book first published 2024

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-1-0364-0804-6
ISBN (Ebook): 978-1-0364-0805-3

TABLE OF CONTENTS

CHAPTER 1

NEURAL NETWORK – A BLACK BOX MODEL

KUOK KING KUOK[1], CHIU PO CHAN[2],
MD REZAUR RAHMAN[3],
KHAIRUL ANWAR MOHAMAD SAID[3],
CHIN MEI YUN[1]

[1]Faculty of Engineering, Computing and Science, Swinburne University of Technology Sarawak Campus, Jalan Simpang Tiga, 93400, Kuching, Sarawak, Malaysia.
[2]Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Jalan Datuk Mohammad Musa, 94300, Kota Samarahan, Sarawak, Malaysia
[3]Faculty of Engineering, Universiti Malaysia Sarawak, Jalan Datuk Mohammad Musa, 94300, Kota Samarahan, Sarawak, Malaysia

Abstract

Artificial Neural Network (ANN) is a computational model based on the structure and operation of biological neural networks. It is a black box model due to its complexities and difficulties in understanding how to make decisions and predictions with complicated internal structures and huge parameters involved. The basic unit of ANN is the artificial neurons. A group of neurons forms a layer. There are three layers in ANN, namely, the input, hidden, and output layers. Forward and backward propagation are two common learning processes adopted for adjusting weights and biases in ANN. Various activation functions are used, such as Hard limit, Tan-Sigmoid, Linear, Log-Sigmoid, Rectified Linear Unit (ReLU), Hyperbolic Tangent (tanh), and Softmax, enabling ANN to simulate complicated relationships and perform nonlinear transformations. Three learning paradigms of ANN include

supervised, unsupervised, and reinforcement learning. A variety of metaheuristic algorithms have been used to train ANN, including Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Ant Colony Optimization (ACO), Tabu Search (TS), and Harmony Search (HS). To date, ANN has been successfully adopted in streamflow prediction, rainfall-runoff modeling, groundwater modeling, water quality modeling, and water demand forecasting.

Keywords: black box model, learning processes, activation functions, learning paradigms, metaheuristic algorithms

1. Introduction

Artificial Neural Network (ANN) is a computational model based on the structure and operation of biological neural networks, like the human brain. The fundamentals of deep learning are also based on ANN, a subfield of machine learning (Taye, 2023).

An ANN comprises interconnected artificial neurons, also known as nodes or units. These artificial neurons are grouped, forming layers. ANN consists of three layers: the input layer, hidden layer, and output layer (Di Franco & Santurro, 2021). Each neuron receives input signals, executes a computation, and outputs a signal. The output of one layer is transmitted as the input for the following layer, allowing data to flow throughout the network (Kuok, 2010).

The connections between neurons are represented by weights and biases to control the strength of the signals transmitted from one neuron to another. These weights and biases are adjusted during training to enhance the network's efficiency at a particular task. The backpropagation technique often accomplishes this by updating the weights and biases propagated backward through the layers to minimize the difference between the network's output and the desired output (Basheer & Hajmeer, 2000).

ANN has the ability to learn and generalize from examples. This enables ANN to solve various tasks effectively, including classification, regression, and pattern recognition. The ability of ANN to discover subtle patterns and relationships within the data automatically makes them excellent at addressing complex problems involving massive data volumes (Goel et al., 2022).

In recent years, simple ANN has been expanded to deep neural networks. A deep neural network consists of multiple hidden layers that enable them to learn hierarchical data representations. The multiple hidden layers enable

deep learning models to handle more complex and sophisticated tasks and process enormous volumes of data (Montavon et al., 2018).

2. ANN - Black Box Model

To some extent, ANN can be considered a black box model since internal workings or mechanisms are difficult for humans to interpret or comprehend. It is difficult to understand how ANN justifies a decision and prediction due to their complicated internal structures and large number of parameters. Below are the reasons ANN is categorized as a black box model:

a) Hidden Layers and Neuron Interactions: ANN can have multiple hidden layers and many neurons. Understanding and explaining how the inputs are handled and altered within the network is challenging, especially the computations of weights and biases within these hidden layers (Zhang et al., 2018).
b) Nonlinear Transformations: ANN uses nonlinear activation functions to capture the nonlinearity and complicated data relationship. Identifying and tracking the effects of specific features on the network's output is difficult due to these nonlinear transformations (Buhrmester et al., 2021).
c) High-Dimensional Parameter Space: ANN's training process often involves learning many weights and biases parameters. It is difficult to comprehend how each weight and bias affects the final prediction result (Buhrmester et al., 2021).
d) Lack of Transparency in Training: The training process of ANN involves adjusting the weights and biases based on given objective function(s). However, the network's specific decision-making rules or patterns may need to be clarified or understandable (Zhang et al., 2018).

Although ANN can be categorized as a black box model in terms of interpretability, its effectiveness in learning and making precise predictions has made it a superior tool in various domains, including image and speech recognition, natural language processing, classification, and predictive modeling.

3. Relationships Between ANN and Biological Neural System

3.1 The Biological Neuron

ANN is inspired by the structure and function of biological neurons. A biological neuron, also called a nerve cell, is a fundamental element of the human brain, which provides the ability to remember, think, and apply previous experiences to every action. These neurons can connect with up to 200000 other neurons wired up in a 3-dimensional pattern. The brain's power comes from the number of these essential components and their multiple connections (Kuok et al., 2010).

These neurons control information transmission and processing through electrical and chemical signals. The four basic components of biological neurons are dendrites, soma, axons, and synapses (Molnar & Gair, 2015), as presented in Fig. 1. The process of these four basic components can be characterized as follows:

a) Dendrites: The finger-like cells located at the end of a neuron. Dendrites are short, branching fibers extending from the cell body of the nerve cell. This fiber increases the surface area available for receiving incoming information or input portions of a neuron.
b) Soma: The cell body of a neuron contains the nucleus and other structures that will combine the signals from the dendrites and pass them on.
c) Axons: The axons are located at the end of the soma and control the neuron's firing. The structure will fire a signal when the signal's total strength exceeds the axon hillock's threshold limit. These axons may fire 200 times per second.
d) Synapses: The gap in the terminal button is known as a synapse, which is responsible for sending the signal to other neurons by releasing neurotransmitters into the synaptic gap. This process of transfer takes between 0.1 and 0.2 milliseconds.

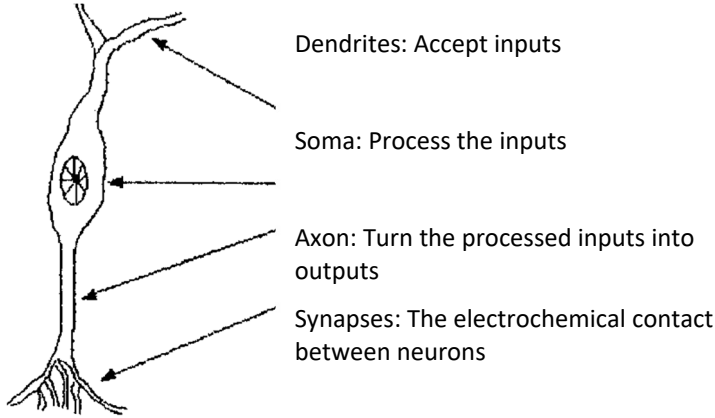

Fig. 1: Four Parts of a Typical Cell.

Biological neurons rely on the generation and propagation of electrical signals to function. A neuron generates an action potential, a rapid electrical impulse that travels down the axon as it receives enough input from dendrites. This action potential causes neurotransmitters to be released at the synapse, which sends the information to the next neuron in the neural circuit.

3.2 The Artificial Neuron

The basic unit of the artificial neurons simulates the four basic functions of biological neurons, as shown in Fig. 2. It is a computer simulation of a 'brain-like' system of interconnected processing units. However, artificial neurons are much simpler than biological neurons. The processing units are typically viewed as analogous to neurons and are presumed to operate in parallel (Kuok et al., 2021). The behavior of a single processing unit in an ANN can be characterized as follows (Minns & Hall, 1996):

a) The units compute the total signal sent to it by other processors in the network.
b) The unit applies an activation function to this total signal to adopt the particular level of internal activity.
c) The units send a signal to other processors in the network. This signal is a function of the unit's internal activity.
d) One processor sends the signal to another through a weighted connection, typically analogous to a synapse.

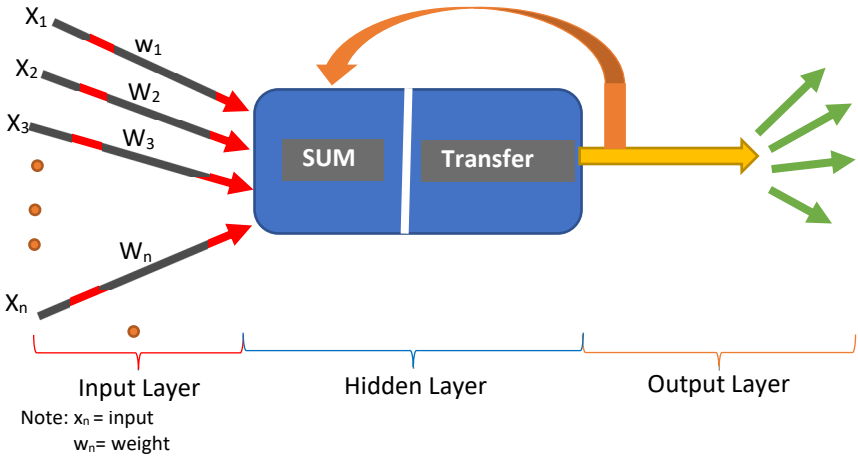

Fig. 2: The basics of an artificial neuron.

Various inputs to the network are represented by *x(n)*. Each input is multiplied by a connection weight represented by *w(n)*. In the simplest case, these products are summed and fed through a transfer function to generate a result and, subsequently, an output.

3.3 Comparison between ANN and Biological Neural System

ANN has exhibited brain-like characteristics such as learning from experiences, generalizing their knowledge, and performing extractions. In other words, ANN emulates the human brain neuron system. However, there are some differences between ANN and biological neural system. The main differences are summarized in Table 1. The analogies between biological and ANN are tabulated in Table 2.

Table 1: Comparison between ANN and biological neural system

Biological Neural System	**ANN**
Complex synapses	Simple synapses
Pulse transmission	Activity value
Learning as fast as one pass	Many passes required
10 billion neurons	Maximum in 1000s

Table 2: The analogy between biological and ANN

Biological Neural System	ANN
Soma	Neuron
Dendrites	Input
Axon	Output
Synapse	Weight

4. Model of A Neuron

The neuron is the basic element of ANN. It is an information processing unit fundamental to the operation of a neural network where the neuron links between units to form a neural network. The purpose of a neuron is to receive information from other neurons and then perform some relatively simple processing on this combined information and send the results to other neurons. Neurons are classified into three types: input neuron, output neuron, and hidden neuron. An input neuron has only one input, no weight adjustment, and the input is from an external source. An output neuron is one whose output is used externally as a network result. Meanwhile, a hidden neuron receives its inputs from other neurons and sends its output only to other neurons.

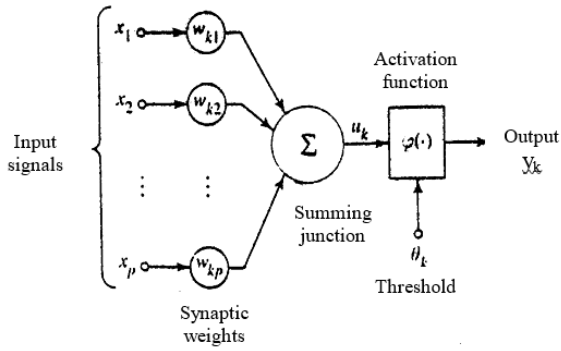

Fig. 3: Nonlinear model of a neuron

According to Haykin (1994), three basic elements of the neuron model are presented in Fig. 3, including:

a) A set of synapses or connecting links through weights. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. The first subscript refers to the neuron in question, and the second refers to the input end of the synapse to which the weight refers. The weight w_{kj} is positive if the associated synapse is excitatory and negative if the synapse is inhibitory.
b) An adder, $\sum$ was used to sum all the input signals, which were weighted by the respective synapses of the neuron. This operation described here constitutes a linear combiner.
c) An activation function for limiting the amplitude of the neuron's output. The activation function is also called the squashing function to limit the permissible amplitude range of the output signal to some finite value. Typically, the normalized amplitude range of the neuron's output is written as the closed unit interval [0,1] or alternatively [-1,1].

4.1 A neuron sample

A linear neuron with *R* inputs is illustrated in Fig. 4 below:

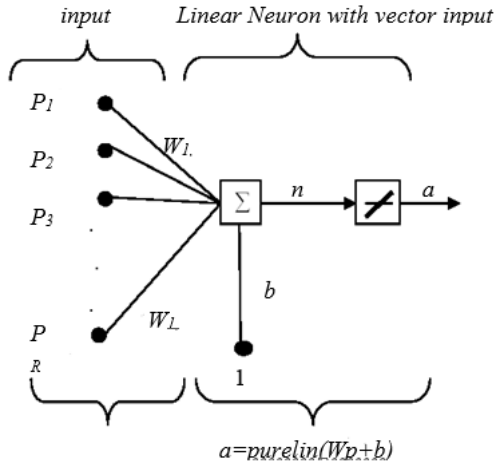

Fig. 4: Linear neuron model

Linear Transfer Function named *purelin* calculates the neuron's output by simply returning the value passed through, as presented in Eq. 1.

$$a = purelin(n) = purelin(Wp + b) = Wp + b \quad \text{Eqn. 1}$$

Single-layer linear network can perform linear function approximation. It can be designed directly or trained with the Widrow-Hoff learning rule to find a minimum error solution. In addition, linear network can be trained adaptively by allowing the network to track changes in its environment. The design of a single-layer linear network is entirely constrained by the problem to be solved. The number of network inputs and neurons in the layer is determined by the number of inputs and outputs required by the problem. Multiple layers in a linear network do not necessarily result in a more robust network. Therefore, a single layer is not necessarily a limitation. However, the linear network can solve only linear problems by making a linear approximation with the minimum sum-squared error. A linear network is suitable if the relationship between inputs and targets is linear or a linear approximation is desired.

Meanwhile, nonlinear relationships between inputs and targets cannot be exactly represented by a linear network (Demuth & Beale, 2004).

5. Learning Process

An ANN learns by training it on a dataset by adjusting its weights and biases, allowing it to make accurate predictions or carry out the tasks that have been programmed. The principal method for training ANN is backpropagation. It consists of two fundamental steps: forward propagation and backward propagation (Yin et al., 2021).

In forward propagation, the input data is fed into the network and propagates through the layers from the input layer to the output layer. Each neuron in the network receives inputs from the preceding layer. Then, it computes a weighted sum of those inputs, applies an activation function, and transmits the output to the following layer. The schematic diagram for forward propagation is presented in Fig. 5.

Following the forward propagation step, the network output is compared to the desired output. After comparison, an error value is calculated. Backward propagation attempts to propagate this error across the network by adjusting the weights and biases, as shown in Fig. 6, to minimize these error values.

The training procedure is repeated until the network achieves satisfactory performance or reaches a stable state. Training an ANN requires a suitably sizable and diverse dataset, an appropriate network architecture, appropriate activation functions, and adjustment of hyperparameters, including learning rate and regularisation methods to avoid overfitting.

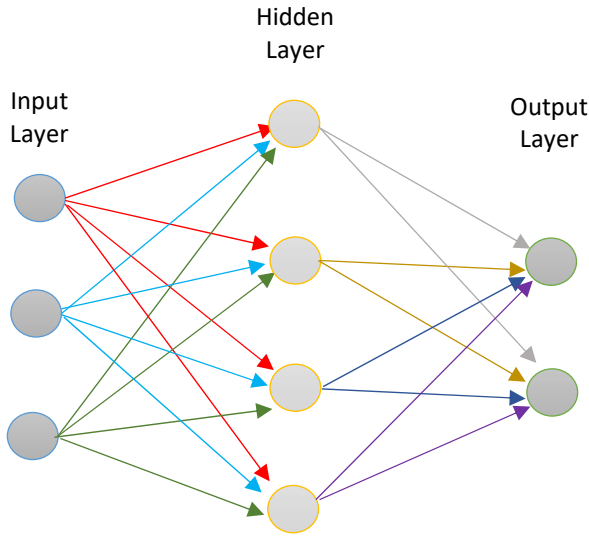

Fig. 5: Schematic diagram of forward propagation

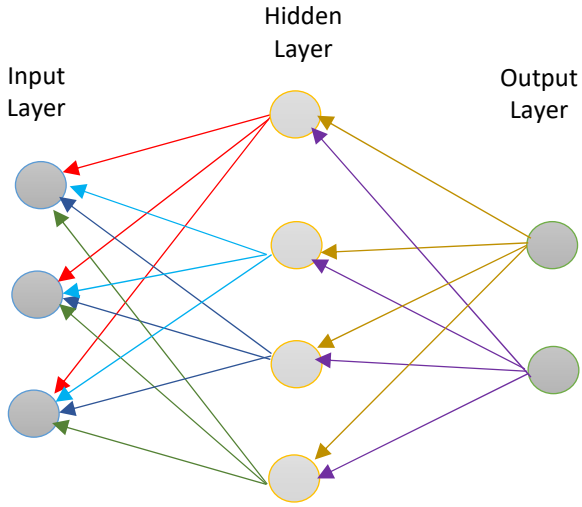

Fig. 6: Schematic diagram of backward propagation

Once trained, the ANN can predict new data by performing forward propagation with the learned weights and biases, accurately mapping input data to the intended output based on the learned patterns and relationships.

6. Activation Function

Activation function in ANN is a mathematical function applied to the weighted sum of inputs at each artificial neuron. The activation function allows nonlinearity to the neuron's output, enabling ANN to simulate complicated relationships and perform nonlinear transformations. The activation function decides whether a neuron will send an output signal based on its input.

The activation function selection is determined by the task, network architecture, and desired network features (Manessi & Rozza, 2018). The four most commonly used functions are the Hard Limit, Tan-Sigmoid, Linear, and Log-Sigmoid activation functions.

a) Hard Limit activation function

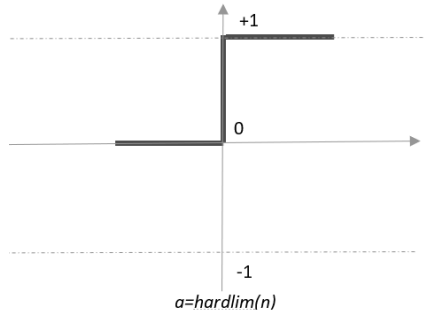

Fig. 7: Hard limit activation function

If its net input hits a threshold, the hard limit activation function forces a neuron to send an output of 1. In contrast, if the neuron doesn't reach that threshold, it sends an output of 0. This enables a neuron to make decisions or categorize information with the output of 1 for yes and 0 for no (Beale et al., 2010). Fig. 7 presents the schematic diagram of the hard limit activation function. The perceptron learning rule frequently trains neurons with the hard limit activation function.

b) Tan-Sigmoid (Tansig) activation function
Neurons with Tan-Sigmoid activation function named tansig, shown in Fig. 8.

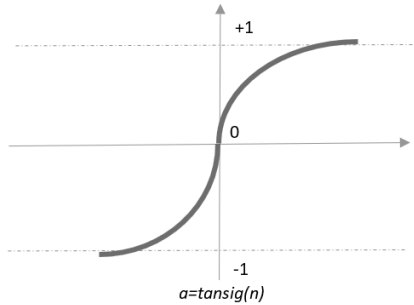

Fig. 8: Tansig activation function

Tansig is a neural transfer or threshold function. Tansig activation function calculates a layer's output from its net input. The output of any neuron is the result of thresholding of its activation, which in turn is the weighted sum of the neuron's inputs. Thresholding is done to scale down the activation and map it into a meaningful output for the problem. The sigmoid function transforms the input, which can have any value between plus and minus infinity, into a reasonable value between 0 and 1 (Demuth & Beale, 2004).

c) Linear activation function

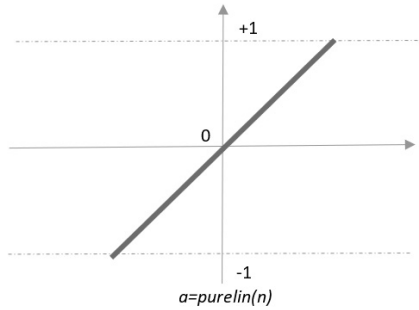

Fig. 9: Linear activation function

The linear activation function, as presented in Fig. 9, allows their outputs to take on any value. A linear network is designed when presented with a set of given input vectors, producing outputs corresponding to the target vectors (Beale et al., 2010).

d) Log-Sigmoid activation function
The log-sigmoid activation function, also known as the logistic sigmoid, as presented in Fig. 10, maps the weighted sum of inputs to a value between 0 and 1. As the input is close to negative infinity, the output becomes closer to 0. In contrast, as the input approaches positive infinity, the output gets closer to 1 (Beale et al., 2010). The log-sigmoid activation function is suitable for producing probabilities of binary classifications.

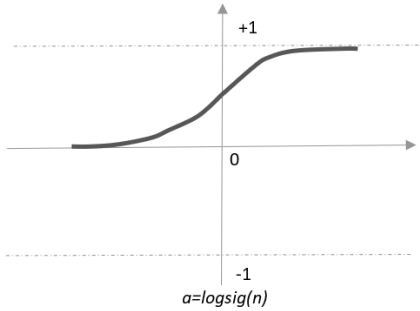

Fig. 10: Log-Sigmoid activation function

Some other popular activation functions are:

e) Rectified Linear Unit (ReLU) activation function
The Rectified Linear Unit (ReLU) activation function is widely used in deep learning. ReLU directly returns the input if it is positive and returns zero otherwise. ReLU activation is computationally efficient and helps to avoid the vanishing gradient problem for deep learning networks. ReLU is effective in many applications and encourages sparsity (Yu et al., 2020; Sporea et al., 2021; Surekcigil Pesch et al., 2022). The schematic diagram of the ReLU activation function is presented in Fig. 11.

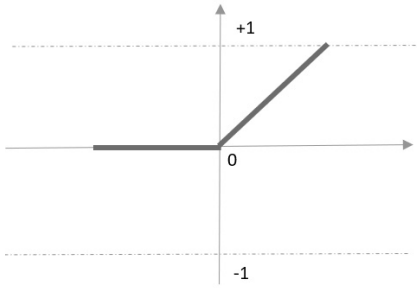

Fig. 11: Rectified Linear Unit (ReLU) activation function

f) Hyperbolic Tangent (tanh) activation function
The hyperbolic tangent activation function, also known as the tanh, maps the weighted sum of inputs to a value between -1 and 1. As the tanh function is symmetric around the origin, it works well in models where the output is required to be centered around zero (Wang et al., 2022). The schematic diagram of the tanh activation function is presented in Fig. 12.

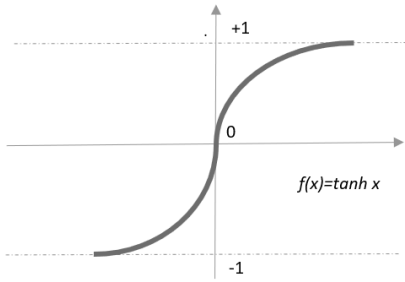

Fig. 12: Hyperbolic Tangent (tanh) activation function

g) Softmax activation function
The softmax activation function is typically applied to solve classification problems with multiclass output layers. Softmax activation function maps the weighted sum of inputs to a probability distribution over multiple classes and ensures that the output sum is 1. Fig. 13 presents the schematic diagram of the softmax activation function (Sharma et al., 2017; Kouretas & Paliouras, 2019). The softmax transfer function is helpful for deriving class probabilities in multiclass classification tasks.

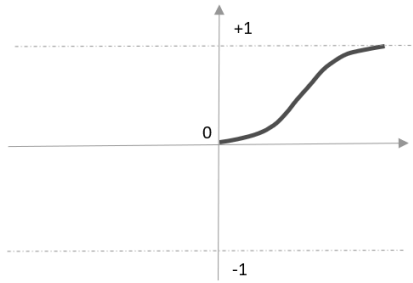

Fig. 13: Softmax activation function

Apart from these few activation functions, other activation functions, such as leaky ReLU, parametric ReLU, and exponential linear unit (ELU), have been developed to solve various problems and achieve promising performance in different scenarios. The selection of the activation function depends on the task requirements and the architecture of the network.

7. Learning Paradigms

Learning paradigms are the most salient feature of neural networks. The learning paradigm largely depends on the neural network structure and the data characteristics. Three basic classes of learning paradigms are supervised learning, reinforcement learning, and unsupervised (self-organized) learning.

7.1 Supervised Learning

The fundamental learning paradigm in ANN is supervised learning. The training data consists of input-output pairs. Each input is associated with a corresponding desired output or target value. Supervised learning aims to develop a function or mapping to correctly forecast the result for brand-new inputs (Cunningham et al., 2008).

Supervised learning networks have been the mainstream of neural model development. Supervised learning is performed under the supervision of an external 'teacher' named as the target. This target is knowing the environment represented by a set of input-output examples. By virtue of built–in knowledge, the target is able to provide the neural network with a desired response for that training vector. The desired response represents the optimum action to be performed by the neural network (Uddin et al., 2019).

The network parameters are adjusted under the combined influence of the training vector and the error signal (the difference between the actual response of the network and the desired response). This adjustment is carried out iteratively to make the ANN emulates the target eventually. Knowledge of the environment available to the target is transferred to the ANN as fully as possible (Kuok et al., 2019). When this condition is reached, the target is dispensed, letting the ANN deals entirely with the environment. The block diagram of supervised learning is shown in Fig. 14.

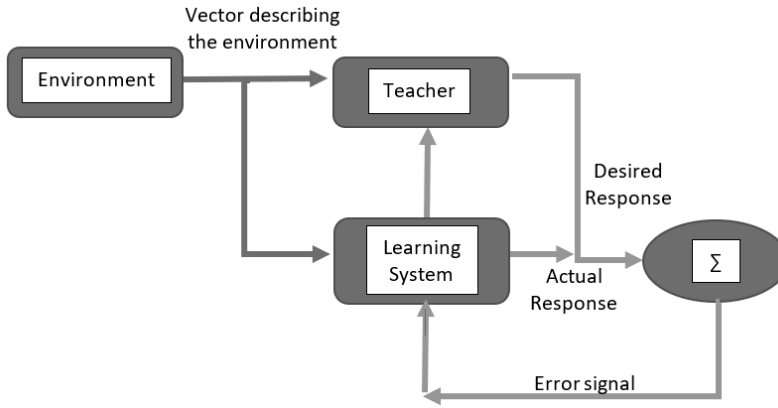

Fig. 14: Block diagram of supervised learning

Supervised learning can be performed in an offline or online manner. In the offline case, a separate computational facility designs the supervised learning system. Once the desired performance is accomplished, the design is frozen, where the ANN will operate statically. On the other hand, the learning procedure for online learning is implemented solely within the system itself and not by acquiring a separate computational facility (Jiang et al., 2020; Chiu et al., 2021). In other words, learning is accomplished in real time because the neural network is dynamic. Naturally, the requirement of online learning places a more severe requirement on a supervised learning procedure than offline learning.

The main disadvantage of supervised learning is that without a target, ANN is unable to learn new strategies for particular situations not covered by the set of examples used to train the network. The use of reinforcement learning may overcome this limitation.

7.2 Unsupervised Learning

Unsupervised learning is a learning paradigm that does not have explicit target outputs. Unsupervised learning uses artificial intelligence (AI) algorithms to find patterns in data sets that are neither labeled nor categorized. Unsupervised learning aims to find hidden links, structures, or patterns in data without prior knowledge of the target outputs. The model learns from the data's underlying structure and distribution (Ghahramani, 2004).

Unsupervised learning models are superior for finding patterns, groupings, and distinctions in unstructured data. It works exceptionally well for image recognition, consumer segmentation, and exploratory data analysis. Unsupervised learning algorithms can categorize, label, and group the data points found in data sets without needing external assistance. It is able to find patterns on its own in data sets within a system. Unsupervised learning organizes unsorted data based on similarities and differences with the help of AI systems, despite the absence of categories.

Unsupervised learning begins with an unlabeled dataset containing only input data fed through training algorithms by machine learning engineers or data scientists (Zaadnoordijk et al., 2022). The data consists of a set of examples or observations without any associated target values or classifications.

The goal of unsupervised learning is for the algorithms to identify patterns in training data sets and categorize input objects based on the patterns identified by the system. The algorithms evaluate the underlying structure of the data sets by extracting useful information or attributes. These algorithms are anticipated to develop particular outputs by identifying the relationships between the input object of each sample (Iqbal et al., 2022).

For example, when unsupervised learning algorithms were provided with fruit image data sets, they could categorize fruits into groups according to their physical characteristics, such as colour, scales or shapes. As the algorithms learn to recognize distinctions by uncovering and identifying patterns within each category, unsupervised algorithms further categorize the data into increasingly specific or ever-finer subgroups, as presented in Fig. 15. Pattern recognition occurs automatically in unsupervised learning without receiving any instruction that teaches the system to distinguish specific categories.

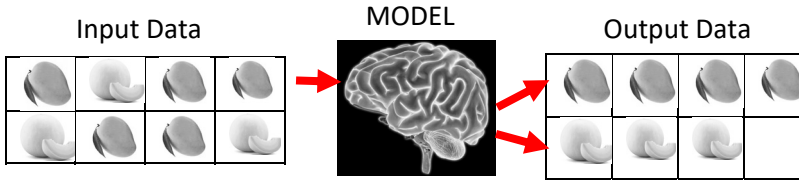

Fig. 15: Block diagram of unsupervised learning

Unsupervised learning often focuses on clustering similar objects or data points while placing dissimilar objects in other clusters. Clustering algorithms for unsupervised learning can be categorized into:

a) Exclusive clustering that specifies a data point can exist only in one cluster.
b) Overlapping clustering enables data points to belong to multiple clusters at different membership levels.
c) Hierarchical clustering classifies the data into agglomerative or divisive. Agglomerative clustering initially set the data points as separate groupings and merged at a later stage. In contrast, divisive clustering divides a single data cluster based on data points.
d) Probabilistic clustering groups the data points based on their potential belonging to a specific distribution.

7.3 Reinforcement Learning

Reinforcement learning is utilizing an appropriate action to maximize reward in a specific situation. Reinforcement learning is used by many software and machines to figure out the best action or way to take within a particular situation. There is no target output for reinforcement learning during the training phase, but the reinforcement agent determines how to complete the task. Without a training dataset, reinforcement learning must gain knowledge from its experience (Wiering & Van Otterlo, 2012). Fig. 16 presents the block diagram of reinforcement learning

Reinforcement learning is the science of decision-making based on the trial and error method, intending to learn the optimal behavior in an environment to maximize reward. It is a self-teaching system that learns by trial and error to achieve the best results. Reinforcement learning employs algorithms that learn from outcomes and determine the following action. After each action, the algorithm receives feedback that helps to evaluate whether the option it made was accurate, neutral, or erroneous (Sutton &

Barto, 2018). It is an effective strategy for automated systems that make numerous small decisions without human intervention.

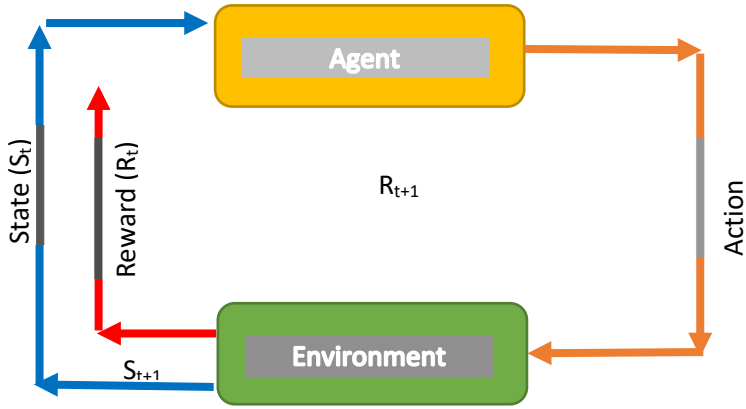

Fig. 16: Block diagram of reinforcement learning

The key points of reinforcement learning are:

a) Input: Input is the initial state from which the model will begin.
b) Output: Output is the solution to a given problem.
c) Training: Training is based on the input. The model training is based on the input, and the user will determine whether to reward or penalize the model based on its output.
d) The model continues to learn.
e) The optimal solution is decided based on the greatest reward.

8. Metaheuristic Algorithms

Metaheuristic algorithms are a class of optimization algorithms designed to search and discover approximate solutions to complicated optimization problems. These algorithms are applicable to solve various problem domains, including engineering, logistics, finance, and machine learning, where finding the global optimum is challenging or computationally costly. Some examples of metaheuristic algorithms are genetic algorithm (GA), particle swarm optimization (PSO), simulated annealing (SA), ant colony optimization (ACO), tabu search, harmony search (HS), and others. Each algorithm has distinct features and strategies for exploring and utilizing the search space.

a) Genetic Algorithms (GA):
Natural selection and evolution are the sources of inspiration for the GA. It operates on a population of candidate solutions. Each solution represents a set of parameters. The algorithm uses selection, crossover, and mutation operations to generate new candidate solutions. It continuously improves the solutions over generations (Kuok et al., 2011).
b) Particle Swarm Optimization (PSO):
The collective behavior of fish schools or bird flocks inspires PSO. Each fish or bird was assumed to be a particle. As the fish schools or bird flocks travel through the search space, each particle in the population represents a potential solution. The particles eventually converge on the best solution by shifting their positions in accordance with both their individual best-known position and the total population's best-known position (Kuok & Chan, 2012).
c) Simulated Annealing (SA):
The metallurgical annealing procedure serves as the basis of SA. This algorithm is a stochastic optimization technique that begins with a preliminary solution and iteratively investigates the search space by accepting moves that enhance the solution. Over time, the probability of accepting less desirable answers lowers, allowing the algorithm to avoid local optima and converge to the overall best solution (Chibante, 2010; Delahaye et al., 2019).
d) Ant Colony Optimization (ACO):
The fundamental ACO is based on the foraging habits of ants. It entails replicating the behavior of ants that leave pheromone trails to communicate and discover the fastest route between their nest and food sources. The algorithm utilizes pheromone trails to search for an optimal solution, updating the strength of the trails based on the effectiveness of the solutions discovered (Dorigo et al., 2006; Dorigo, 2007).
e) Tabu Search:
Tabu Search is a metaheuristic algorithm based on local searches and retains a short-term memory referred to as the "tabu list" to avoid revisiting recently explored solutions. It investigates the current neighborhood solution by employing neighborhood movements while avoiding moves that violate specific tabu conditions (Glover & Laguna, 1997; Venkateswarlu, 2021). Tabu Search is effective at eluding local optima and finding better solutions.

f) Harmony Search (HS):
The basis for HS is musical improvisation. By merging the components of existing solutions, HS simulates the process of musical improvisation to produce new solutions (Gao et al., 2015; Kim, 2016). The algorithm remembers the most successful solutions found and continuously improves the solution in accordance with harmony values.

Apart from these few metaheuristic algorithms, a series of metaheuristic algorithms have been utilized and will be discussed in the following chapters.

9. Application of ANN in Hydrological Modelling

Artificial neural network (ANN) has several applications in hydrological modeling. Applying ANN has improved the understanding and prediction of different hydrological processes. Below are a few particular applications of ANN in hydrological modeling:

a) Streamflow Prediction: ANN can be used to model and predict streamflow, which is extremely important for managing water resources, forecasting floods, and monitoring droughts. In order to estimate streamflow accurately, ANN can learn intricate correlations and create relationships among meteorological factors, including rainfall, temperature, humidity, and streamflow. Halff et al. (1993) designed a three-layer feedforward ANN using the observed rainfall hyetographs as inputs and hydrographs recorded by the US Geological Survey at Bellevue, Washington, as the model's output. Harun et al., (1999) applied ANN in daily rainfall-runoff modeling to estimate inflows into the Pedu and Muda reservoirs in Kedah, Malaysia, using three-layer feedforward neural networks with a backpropagation learning algorithm. Elshorbagy et al., (2000) have used ANN to predict the daily runoff of the Red River in southern Manitoba, Canada. Gautam et al., (2000) have simulated runoff for the Tono catchment in Japan using a three-layer feedforward network backpropagation algorithm. Predicting daily watershed runoff as a function of rainfall, snow water equivalent, and temperature was investigated by Tokar and Markus (2000). Dibike and Solomatine (2001) investigated the use of ANN for daily river flow prediction in the Apure River basin in the southwest part of Venezuela. Garcia-Bartual (2002) applied ANN models for short-term river flow forecasting under heavy rain storms for the upper Serpis basin, with the outlet in Beniarrés Reservoir, Spain. Dolling and Varas (2002)

utilized ANN to predict monthly streamflow in mountain watersheds in Chile. Nor et al., (2007) applied a radial basis function (RBF) network to simulate streamflow hydrographs for Sungai Bekok Catchment, Johor, Malaysia, and Sungai Ketil Catchment, Kedah, Malaysia. Noori and Kalin (2016) successfully coupled SWAT and ANN models to enhance daily streamflow prediction for 29 nearby watersheds around Atlanta, in the southeastern United States. Dalkiliç and Hashimi (2020) carried out a study to predict daily streamflow for Büyük Menderes River, Turkey, using ANN, wavelet neural networks (WNNs), and adaptive neuro-fuzzy inference system (ANFIS) models. Teng and Kuok (2021) developed an ANN model to forecast the precipitation and streamflow in Sarawak River, Malaysia. Gunathilake et al. (2021) developed hydrological models and ANN to simulate streamflow in a tropical catchment of Sri Lanka. Hassan and Hassan (2021) improved ANN-based streamflow forecasting models through data preprocessing for thirteen stations located in the Upper Indus Basin, Pakistan.

b) Rainfall-Runoff Modeling: ANN simulates the transformation from rainwater into surface runoff. The transformation of rainfall into runoff is a complicated phenomenon. However, the ability of ANN to capture the nonlinear relationship between the input rainfall and the output runoff has made the accurate simulation of runoff volumes and hydrographs possible. Shamseldin (1997) used ANN for daily rainfall-runoff modeling and then compared the performance with a simple linear model (SLM), seasonally based linear perturbation model (LPM), and nearest neighbor linear perturbation model (NNLPM). Coulibaly et al., (2000) used an early stopped training approach (STA) to improve the neural network daily reservoir inflow forecasting for the Chute du-Diable watershed, Northern Quebec. Rainfall-runoff models using the ANN method in water resource projects were developed by Harun et al., (2002). Nourani et al., (2000) utilized a multivariate ANN-wavelet approach for rainfall–runoff modeling in Tabriz, Iran. Sarkar and Kuandar (2012) modeled the event-based rainfall-runoff using ANN for the Ajay River basin in India. Chakravarti et al., (2015) analyzed rainfall-runoff in India using ANN. Aoulmi et al., (2021) assessed ANN rainfall-runoff models under different input meteorological parameters for Seybouse basin, Northeast Algeria. Mohseni and Muskula (2023) modeled rainfall-runoff using ANN for the Purna sub-catchment of Upper Tapi Basin, India. Lai et al., (2023) developed the metaheuristic ANN to enhance the performance of the rainfall-runoff model.

c) Flood Forecasting: ANN can be used for flood forecasting by combining real-time and historical values for both meteorological and hydrological data. ANN is able to forecast and provide early warnings of flood disasters by learning the correlations among precipitation, river water levels, and other pertinent elements. This helps in the preparation and response of disaster management. Liong et al., (1996) developed a real-time hourly flow forecasting scheme and an early warning flow forecasting scheme for the Upper Bukit Timah catchment in Singapore using a feedforward multilayer perceptron network with a backpropagation training algorithm. Thirumalaiah and Deo (2000) have applied ANN for real-time forecasting of hourly flood runoff and daily river stage at Kunta and Koida, India. Imrie et al., (2000) developed an ANN model to predict extreme runoff values at downstream of Trent River, United Kingdom. ANN is also employed in river flood prediction of ungauged catchments in the United Kingdom by Wright and Dastorani (2001). Ayalew et al., (2007) utilized ANN for real-time flood forecasting at the Omo River in southern Ethiopia. Mukerji et al., (2009) simulated the flood using ANN, neuro-fuzzy, and neuro-GA models for the Ajay River Basin in Jharkhand, India. Elsafi (2014) adopted ANN for flood forecasting at Dongola Station in the River Nile, Sudan. Agarwal et al., (2021) studied flood forecasting and flood flow modeling in a river system using ANN for Tar Basin, North Carolina, USA. Mistry and Parekh (2022) forecasted the flood events using ANN at Deo River, Gujarat, India.

d) Groundwater Modelling: ANN was used to model and predict groundwater levels to ensure sustainable groundwater usage in managing water resources. To estimate groundwater levels accurately, the parameters that can be used to create the groundwater model include precipitation, evapotranspiration, soil characteristics, and pumping rates. Nourani et al., (2008) adopted an ANN-based model for spatiotemporal groundwater level forecasting in northwestern Iran. Mohammadi (2008) estimated the groundwater table using MODFLOW and ANN. Trichakis et al., (2011) studied groundwater level simulation using ANN for Edward's aquifer in Texas, USA. Khaki et al., (2016) used ANN to model groundwater levels in Langat Basin, Malaysia. Lee et al., (2019) adopted ANN for groundwater level forecasting and assessed the impacts of groundwater levels on Yangpyeong riverside area in South Korea. Moghaddam et al., (2019) developed comparative mathematic models for forecasting groundwater levels at Birjand Aquifer, South

Khorasan, Iran. Al-Waeli et al., (2022) utilized ANN to predict groundwater salinity in the West Najaf–Kerbala region, Iraq.

e) Water Quality Modeling: ANN has been utilized to model and forecast water quality data in rivers, lakes, and reservoirs. ANN is able to forecast temperature, dissolved oxygen, pH, and pollutant concentrations into consideration in water. This will help in understanding and managing water quality problems, such as eutrophication and contamination. Musavi-Jahromi and Olabi (2008) applied ANN in water quality modeling at Karoon River, Iran. Thair et al., (2014) successfully developed ANN to predict the water quality of the Euphrates River in Iraq. Sarkar and Pandey (2015) modeled the river water quality using the ANN technique at Mathura City, located on the bank of River Yamuna in Uttar Pradesh, India. Isiyaka et al., (2019) developed a water quality model using ANN and multivariate statistical techniques. Sulaiman et al., (2019) classified the water quality using ANN for three locations: Pontian River, Batu Pahat River, and Muar, Malaysia. Ubah et al., (2021) forecasted water quality parameters, including pH, Total Dissolved Solids (TDS), Electrical Conductivity (EC), and Sodium (Na) using ANN for irrigation purposes at Nnewi, Anambra State, Nigeria. Setshedi et al., (2021) used ANN to predict the physicochemical characteristics of water quality for three district municipalities in Eastern Cape Province, South Africa. Rustam et al., (2022) adopted ANN for water quality and consumption prediction using data from renowned sources such as Kaggle and GitHub.

f) Water Demand Forecasting: ANN is able to forecast water demand patterns and trends based on variables, including population growth, climatic conditions, economic indicators, and historical water consumption data. Accurate water demand estimation is essential for water supply planning and infrastructure design. Jain et al., (2000) forecasted short-term water demand using ANN for the Indian Institute of Technology (IIT) Kanpur campus, India. Al-Zahrani and Abo-Monasar (2015) predicted urban residential water demand based on ANN and time series models for Al-Khobar City in the Kingdom of Saudi Arabia. Gwaivangmin and Jiya (2017) predicted water demand using ANN for supervisory control at the Laminga Water Treatment Plant and its water distribution network in Jos, Nigeria. Lorente-Leyva et al., (2019) developed ANN for urban water demand forecasting for a principal city at zone north of Ecuador. Awad and Zaid-Alkelani (2019) predicted the water demand using ANN and a statistical model for Jenin City in the north