Supplementary Note

on

Neural Network

`A Basic overview to the Neural Networks`

For Summer Internship Report 2024

Submitted to:
Dr. Kuntal Roy
Department of Electrical Engineering & Data Science
IISER Bhopal

Submitted by:
Sudipta Majumder – 23PH40055
2nd year MSc Physics
Department of Physics
Indian Institute of Technology, Kharagpur

Neutral Networks

1. Introduction

Neural networks, a subset of machine learning, have revolutionized various fields such as computer vision, natural language processing, and robotics. Inspired by human brain's structure, neural networks consist of interconnected nodes, or neurons, which work collectively to process data and recognize patterns.

1.1. Historical Background

The concept of neural networks dates back to the 1940s and 1950s with the work of Warren McCulloch and Walter Pitts, who introduced a simplified model of the neuron. Their model demonstrated that a network of such neurons could, in theory, perform any logical or arithmetic operation.

1.2. Evolution of Neural Networks:

Since, the 1940s, there have been a number of noteworthy advancements in the field of neural networks:

• *1940s-1950s: Early Concepts*
Neural networks began with the introduction of the first mathematical model of artificial neurons by McCulloch and Pitts. But computational constraints made progress difficult.

• *1960s-1970s: Perceptron*
This era is known as the work of Rosenblatt on perceptron. Perceptrons are single-layer networks whose applicability was limited to issues that could be solved linearly separately.
In 1958, Frank Rosenblatt first developed the Perceptron, an algorithm capable of binary classifications. As a first try, despite its initial success, the Perceptron faced limitations as its inability to solve non-linearly separable problems like the XOR problem. Which led to a period of reduced interest in neural networks, often it is referred to as the 'AI Winter'

• *1980s: Backpropagations and Connectionism*
In 1986, Rumelhart, Hinton, and Williams' proposed backpropagation method, which helps on Multi-layer network training. With its emphasis on learning through interconnected nodes, connectionism gained appeal.

• *1990s: Boom and Winter*
With applications in image identification, finance, and other fields, neural networks saw a boom. Neural network research did, however, experience a 'winter' due to exorbitant computational costs and inflated expectations during this period.

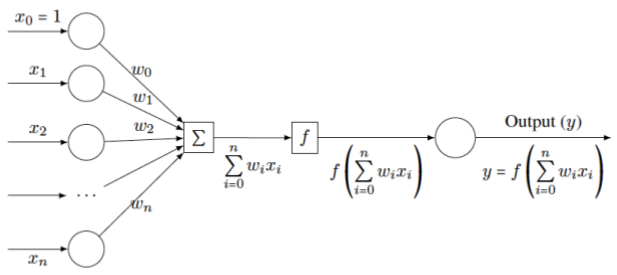
• *2000s: Resurgence and Deep Learning*
Larger datasets, innovative structures, and enhanced processing capability spurred a comeback. Deep learning has shown amazing effectiveness in a number of disciplines by utilizing numerous layers.

• *2010s-Present: Deep learning Dominance*
Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), two deep learning architectures, dominated machine learning. Their power was demonstrated by innovations in gaming, picture recognition, and natural language processing.

2. Fundamental Concepts

2.1. Artificial Neurons

An artificial neuron is a mathematical function conceived as a model of biological neurons. Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs (representing excitatory postsynaptic potentials and inhibitory postsynaptic potentials at neural dendrites) and sums them to produce an output. Each input is separately weighted, and the sum is passed through a function known as an activation function or transfer function.



$x_1, x_2, \dots, x_n$: input signals
$w_1, w_2, \dots, w_n$: wrights associated with input signals
x_0 : input signal taking the constant value 1
w_0 : weight associated with x_0 (called bias)
$\sum$: indicates summation of input signals
f : function which produces the output
y : output signal

The function f can be expressed in the following form:

$$y = f\left(\sum_{i=0}^{n} w_i x_i\right)$$

Note:
The small circles in the schematic representation of the artificial neuron shown in Figure 9.3 are called the nodes of the neuron. The circles on the left side which receives the values of $x_0, x_1, x_2, \dots, x_n$ are called the input nodes and the circle on the right side which outputs the value of y is called output node.

2.2. Activation Function

In an artificial neural network, the function which takes the incoming signals as input and produces the output signal is known as the activation function.
Some simple activation functions:

1. Threshold activation function
$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \le 0 \end{cases}$$
2. Unit step function
$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$$
3. Sigmoid activation function (logistic function)
$$f(x) = \frac{1}{1 + e^{-x}}$$
4. Linear activation function
$$f(x) = mx + c$$
5. Piecewise linear activation function
$$f(x) = \begin{cases} 0 & \text{if } x < x_{min} \\ mx + c & \text{if } x_{min} \le x \le x_{max} \\ 0 & \text{if } x > x_{max} \end{cases}$$
6. Gaussian activation function
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
7. Hyperbolic tangential activation function
$$f(x) = \frac{e^x - e^{-x}}{e^x - e^{-x}}$$

3. Perceptron:

The perceptron is a special type of artificial neuron in which the activation function has a special form.

Consider an artificial neuron having $x_1, x_2, \dots, x_n$ as the input signals and $w_1, w_2, \dots, w_n$ as the associated weights. Let w_0 be some constant. The neuron is called a perceptron if the output of the neuron is given by the following function:

$$o(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \cdots w_nx_n > 0 \\ -1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \cdots w_nx_n \le 0 \end{cases}$$

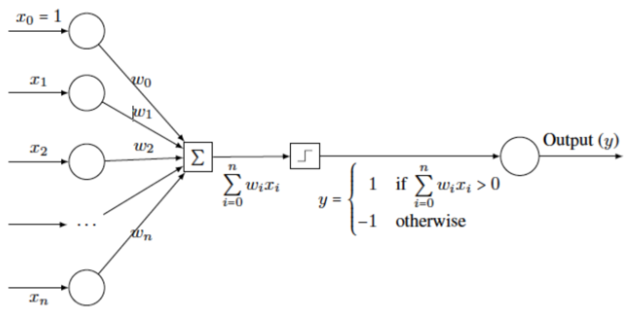

$x_0 = 1$
x_1
x_2
$\cdots$
x_n
w_0
w_1
w_2
w_n
Σ
$\sum_{i=0}^{n} w_ix_i$
$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_ix_i > 0 \\ -1 & \text{otherwise} \end{cases}$
Output (y)

Fig 3.1. Simple diagram of Single layer Perceptron

In 1958, Frank Rosenblatt first developed the Perceptron, an algorithm capable of binary classifications. As a first try, despite its initial success, the Perceptron faced limitations as its inability to solve non-linearly separable problems like the XOR problem. Which led to a period of reduced interest in neural networks, often it is referred to as the 'AI Winter'.

Algorithm

Step 01: Initialize the weights and threshold. Weights may be initialized to 0 or to a small random value.

Step 02: For each example j in the training set D, perform the following steps over the input x_j and desired output d_j:

a) Calculate the actual output:

$$y_j(t) = f\left[w_0(t)x_{j0} + w_1(t)x_{j1} + \cdots + w_n(t)x_{jn}\right]$$

b) Update the weights:

$$w_j(t+1) = w_j(t) + \left(d_j - y_j(t)\right)x_{ji}$$

For all features $0 \le i \le n$.

Step 03: Step 02 is repeated until the iteration error, defined as $\frac{1}{s}\sum_{j=1}^{s}\left|d_j - y_j(t)\right|$ is less than a user-specified error threshold γ, or a predetermined numbers of iterations have been completed, where s is again the size of the sample set.

4. Types of Neural Networks:

There are seven types of neural networks that can be used.

Feedforward Networks:

A feedforward neural network is a simple artificial neural network architecture in which data moves from input to output in a single direction. It has input, hidden and output layers; feedback loops are absent. Its straightforward architecture makes it appropriate for a number of applications, such as regression and pattern recognition.

Multilayer Perceptron(MLP):

MLP is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers, and an output layer. It uses nonlinear activation functions.

Convolutional Neural Networks (CNN):

A CNN is a specialized artificial neural network designed for image processing. It employs convolutional layers to automatically learn hierarchical features from input images, enabling effective image recognition and classification. CNNs have revolutionized computer vision and are pivotal in tasks like object detection and image analysis.

Recurrent Neural Network (RNN):

An artificial neural network type intended for sequential data processing is called a Recurrent Neural Network (RNN). It is appropriate for applications where contextual dependencies are critical, such as time use of feedback loops, which enable information to survive within the network.

Long Short-Term Memory(LSTM):

LSTM is a type of RNN that is designed to overcome the vanishing gradient problem in training RNNs. It uses memory cells and gates to selectively read, write and erase information.

5. Working of a Neural Network

Neural networks are complex systems that mimic some features of the functioning of the human brain. It is composed of an input layer, one or more hidden layers, and an output layer made up of layers of artificial neurons that are coupled. The two stages of the basic process are called backpropagation and forward propagation.

5.1. Forward Propagation

Input layer:

Each feature in the input layer is represented by a node on the network, which receives input data.

Weights and Connections: The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.

Hidden layers:

Each layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.

Output:

The final result is produced by repeating the process until the output layer is reached.

5.2. Backpropagation

Backpropagation is an algorithm that backpropagates the errors from the output nodes to the input nodes. Therefore, it is simply referred to as the backward propagation of errors. It computes the gradient of the loss function with respect to the network weights via chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

Working:

It compares generated output of neural network model to the desired output and generates an error report if the result does not match the generated output vector. Then it adjusts the weights according to the bug report to get the desired output or close to it.

Algorithm:

Step 01: Inputs X, arrive through the preconnected path.

Step 02: The input is modelled using true weights W, which are chosen randomly.

Step 03: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

Step 04: Calculate the error in the outputs:

Backpropagation Error = Actual Output – Desired Output

Step 05: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 06: Repeat the process until the desired output is achieved.

Illustrative Example:

Let us assume that there are two observations:

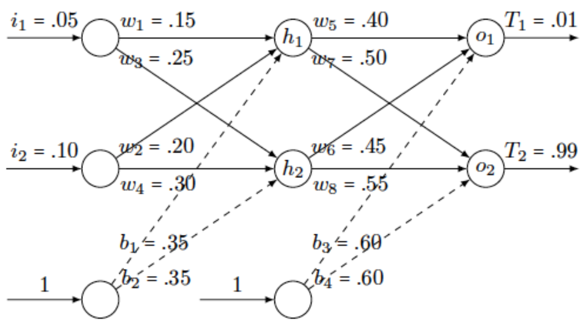
Sample	Input 1 i_1	Input 2 i_2	Output 1 T_1	Output 2 T_2
1	0.05	0.10	0.15	0.99
2	0.25	0.18	0.23	0.79

We are required to estimate the optimal values of the weights $w_1, w_2, \dots, w_8$, b_1, b_2, b_3, b_4. Here b_1 and b_2 are the biases. For simplicity, we have assigned the same biases to both nodes in the same layer.

Step 01: Initialise the connection weights to small random values.

$$w_1 = 0.15, w_2 = 0.20, w_3 = 0.25, w_4 = 0.30, w_5 = 0.40, w_6 = 0.45, w_7 = 0.50, w_8 = 0.55$$

$$b_1 = 0.35, b_2 = 0.35, b_3 = 0.60, b_4 = 0.60$$



Step 02: Present the first sample inputs and the corresponding output targets to the network.

Step 03: Pass the input values to the first layer (the layer with nodes h_1 and h_2). And calculate the outputs from h_1 and h_2.

We use the logistic activation function $f(x) = \frac{1}{1+e^{-x}}$

$\text{out}_{h_1} = f(w_1 \times i_1 + w_2 \times i_2 + b_1)$
$= f(0.15 \times 0.05 + 0.20 \times 0.10 + 0.35) = f(0.3775) = \frac{1}{1+e^{-0.3775}} = 0.59327$

$\text{out}_{h_2} = f(w_3 \times i_1 + w_4 \times i_2 + b_2)$
$= f(0.25 \times 0.05 + 0.30 \times 0.10 + 0.35) = f(0.3925) = \frac{1}{1+e^{-0.3925}} = 0.59689$

Step 05: We repeat this process for every layer. And get the outputs from the nodes in the output layers as follows:

$\text{out}_{o_1} = f\left(w_5 \times \text{out}_{h_1} + w_6 \times \text{out}_{h_2} + b_3\right)$
$= f(0.40 \times 0.59327 + 0.45 \times 0.59689 + 0.60)$
$= f(1.10591) = \frac{1}{1+e^{-1.10591}} = 0.75137$

$\text{out}_{o_2} = f\left(w_7 \times \text{out}_{h_1} + w_8 \times \text{out}_{h_2} + b_3\right)$
$= f(0.50 \times 0.59327 + 0.55 \times 0.59689 + 0.60)$
$= f(1.22492) = \frac{1}{1+e^{-1.22492}} = 0.77293$

The sum of the squares of the output errors is given by

$$E = \frac{1}{2}\left(T_1 - \text{out}_{o_1}\right)^2 + \frac{1}{2}\left(T_2 - \text{out}_{o_2}\right)^2$$
$$= \frac{1}{2}(0.01 - 0.75137)^2 + \frac{1}{2}(0.99 - 0.77293)^2 = 0.298371$$

Step 06: We begin backward phase. We adjust the weights and denote them as $w_1^+, w_2^+, \dots, w_8^+, b_1^+, b_2^+, \dots, b_8^+$. For computation we choose learning rate $\eta = 0.5$.

(a) Computation of adjusted weights leading to o_1 and o_2:

$\delta_{o_1} = \left(T_1 - \text{out}_{o_1}\right) \times \text{out}_{o_1} \times \left(1 - \text{out}_{o_1}\right)$
$= (0.01 - 0.75137) \times 0.75137 \times (1 - 0.75137)$
$= -0.13850$

$w_5^+ = w_5 + \eta \times \delta_{o_1} \times \text{out}_{h_1}$
$= 0.40 + 0.5 \times (-0.13850) \times 0.59327 = 0.35892$

$w_6^+ = w_6 + \eta \times \delta_{o_1} \times \text{out}_{h_2}$
$= 0.45 + 0.5 \times (-0.13850) \times 0.59689 = 0.40867$

$b_3^+ = b_3 + \eta \times \delta_{o_1} \times 1$
$= 0.60 + 0.5 \times (-0.13850) = 0.53075$

Similarly, $\delta_{o_2} = \left(T_2 - \text{out}_{o_2}\right) \times \text{out}_{o_2} \times \left(1 - \text{out}_{o_2}\right)$
$= (0.99 - 0.77293) \times 0.77293 \times (1 - 0.77293) = 0.03810$

$w_7^+ = w_7 + \eta \times \delta_{o_2} \times \text{out}_{h_1}$
$= 0.50 + 0.5 \times 0.03810 \times 0.59327 = 0.51130$

$w_8^+ = w_8 + \eta \times \delta_{o_2} \times \text{out}_{h_2}$
$= 0.55 + 0.5 \times 0.03810 \times 0.59689 = 0.56137$

$b_4^+ = b_4 + \eta \times \delta_{o_2} \times 1$
$= 0.60 + 0.5 \times 0.03810 = 0.61905$

(b) Computation of adjusted weights leading to h_1 and h_2

$\delta_{h_1} = \left(\delta_{o_1} \times w_5 + \delta_{o_2} \times w_7\right) \times \text{out}_{h_1} \times \left(1 - \text{out}_{h_1}\right)$
$= (-0.13850 \times 0.40 + 0.03810 \times 0.50) \times 0.59327 \times (1 - 0.59327)$
$= -0.00877$

$w_1^+ = w_1 + \eta \times \delta_{h_1} \times i_1$
$= 0.15 + 0.5 \times (-0.00877) \times 0.05 = 0.14978$

$w_2^+ = w_2 + \eta \times \delta_{h_1} \times i_2$
$= 0.20 + 0.5 \times (-0.00877) \times 0.10 = 0.19956$

$b_1^+ = b_1 + \eta \times \delta_{h_1} \times 1$
$= 0.35 + 0.5 \times (-0.00877) \times 1 = 0.34562$

$\delta_{h_2} = \left(\delta_{o_1} \times w_6 + \delta_{o_2} \times w_8\right) \times \text{out}_{h_2} \times \left(1 - \text{out}_{h_2}\right)$
$= (-0.13850 \times 0.45 + 0.03810 \times 0.55) \times 0.59689 \times (1 - 0.59689)$
$= -0.00995$

$w_3^+ = w_3 + \eta \times \delta_{h_2} \times i_1$
$= 0.25 + 0.5 \times (-0.00995) \times 0.05 = 0.24975$

$w_4^+ = w_4 + \eta \times \delta_{h_2} \times i_2$
$= 0.30 + 0.5 \times (-0.00995) \times 0.10 = 0.29950$

$b_2^+ = b_1 + \eta \times \delta_{h_2} \times 1$
$= 0.35 + 0.5 \times (-0.00995) \times 1 = 0.34503$

Step 07: Now we set:

$w_1 = w_1^+, w_2 = w_2^+, w_3 = w_3^+, w_4 = w_4^+, w_5 = w_5^+, w_6 = w_6^+, w_7 = w_7^+, w_8 = w_8^+;$
$b_1 = b_1^+, b_2 = b_2^+, b_3 = b_3^+, b_4 = b_4^+$

We choose the next sample input and the corresponding output targets to the network and repeat steps 2 to 6 again.

Types of Backpropagation:
There are two types of backpropagation networks:
Static Backpropagation: Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
Recurrent Backpropagation: Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached.
Static backpropagation provides an instant mapping. While recurrent backpropagation does not provide an instant mapping.

6. Applications of Neural Networks

Computer Vision
Neural networks, particularly CNNs (Convolution Neural Networks), have transformed computer vision tasks such as image classification, object detection and segmentation. Applications range from medical imaging to autonomous driving, where accurate visual perception is crucial.
Natural Language Processing (NLP)
In NLP, neural networks has various applications, including machine translation, sentiment analysis, and text generation. Transformer-based model like BERT and GPHT have set new benchmarks in understanding and generating human language.
Healthcare
Neural networks are increasingly used in healthcare for tasks as disease diagnosis, drug discovery, and personalized treatment plans. They analyse vast amount of medical data to identify patterns and predict patient outcomes.

Finance
In finance, neural networks assist in algorithmic trading, fraud detection, and credit scoring. They process large volumes of financial data to uncover trends and make predictions.
Robotics
Neural networks enable robots to perceive their environment, make decisions, and learn from interactions. Applications include industrial automation, autonomous vehicles, and service robots.
Gaming and Entertainment
Neural networks contribute to game development by creating realistic environments, character behaviors, and adaptive gameplay. They are also used in music and art generation, producing creative works based on learned patterns.

7. Ethical Considerations and Challenges

Bias and Fairness
Neural networks can inadvertently learn and propagate biases present in the training data, leading to unfair outcomes. Ensuring fairness and transparency in AI systems is a crucial challenge that requires careful dataset curation and algorithmic adjustments.
Interpretability
The black-box nature of neural networks makes it difficult to interpret their decisions. Developing methods to explain and understand neural network predictions is essential for building trust and accountability in AI systems.
Data Privacy
To train neural network requires a huge dataset for better results. And the use of personal data in training raises privacy concerns.
Energy Consumption
Training large neural networks requires significant computational resources and energy. Research into more efficient algorithms and hardware is crucial to reduce the environmental impact of AI.

8. Future Directions

Neuromorphic Computing
Neuromorphic computing aims to mimic the brain's architecture and functioning in hardware, potentially leading to more efficient and powerful neural networks. This field holds promise for advancing AI capabilities while reducing energy consumption.
Quantum Neural Networks
Quantum computing offers the potential to solve complex problems that are intractable for classical computers. Quantum neural networks could leverage this power to enhance learning algorithms and tackle previously unsolvable tasks.
Integration with other AI Technologies
Combining neural networks with other AI Technologies, such as reinforcement learning and symbolic reasoning, can lead to more robust and versatile systems. This integration could drive advancements in areas like autonomous systems and human-AI collaboration.

9. Conclusion

Neural networks have come a long way from their early beginnings, evolving into powerful tools that drive innovation across various domains. Their ability to learn from data and make intelligent decisions has opened up new possibilities and transformed industries. However, challenges related to bias, interpretability, privacy, and energy consumption must be addressed to harness their full potential responsibly. The future of neural networks lies in continues research, interdisciplinary collaboration, and a commitment to ethical principles, ensuring that these remarkable technologies contribute positively to society.

10. References:

i. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386-408.
ii. Rumelhart, D.E. hinton, G.E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533-536
iii. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*, MIT Press.
iv. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning, Nature, 521(7753), 436-444.
v. Lecture Notes in Machine Learning, Dr. V N Krishnachandran , Vidya Centre for Artificial Intelligence Research, 111-132.