Chapter – 1

**Neural Networks for Machine Learning**

**Geoffrey Hinton**

with **Nitish Srivastava** & **Kevin Swersky**

**Introduction to**

**Neural Networks**

Notes: Based on ChatGPT prompts & Web search

Why do we need Machine Learning

What are Neural Networks

Some simple Models Of Neurons

A simple Example of learning

Three Types of learning

**1.1 Why do we need Machine Learning**

Machine learning plays a crucial role in today's technology-driven world for several reasons:

1. Handling Big Data: With the explosive growth of data in various domains, traditional methods of data analysis and rule-based programming become inefficient. Machine learning algorithms excel at processing large datasets and extracting valuable insights from them.

* It is very hard to write programs that solve problems like *recognizing a three-dimensional object* from a *novel viewpoint* in *new lighting* *conditions* in a cluttered scene.
* We don’t know what program to write because we don’t know how its done in our brain.
* Even if we had a good idea about how to do it, the program might be horrendously complicated.

1. Complex Problem Solving: Machine learning enables us to *tackle complex problems* that are *difficult to solve* using *traditional* *algorithms*. Tasks like *image recognition*, *natural language processing*, and *speech recognition* require sophisticated models that can learn patterns and representations from data.

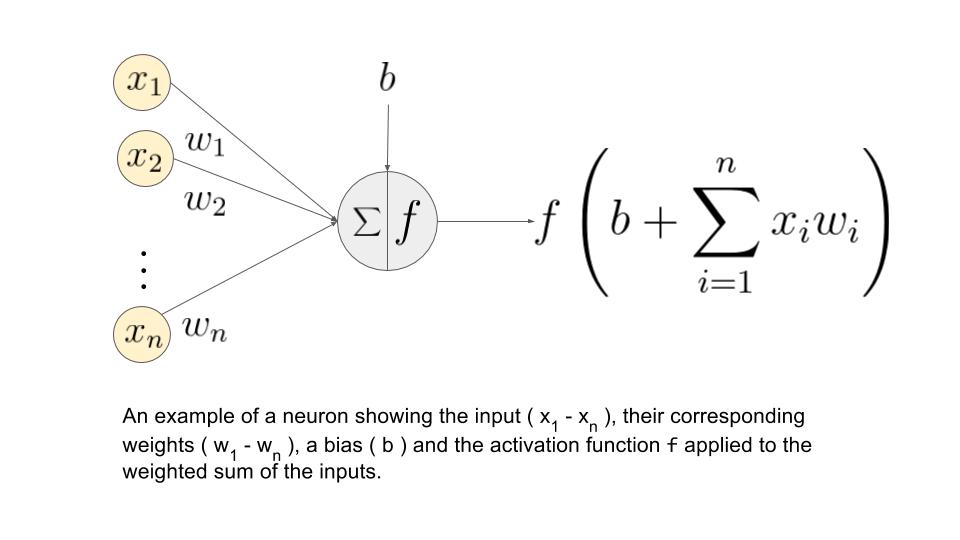
* Some problems involve complex relationships and interactions that cannot be easily captured by traditional mathematical models. Machine learning excels at capturing these intricate patterns.

1. Automation and Efficiency: Machine learning enables automation of tasks that would otherwise be time-consuming and labor-intensive. This efficiency boost can lead to significant cost savings and increased productivity across various industries.
2. Personalization: Machine learning allows businesses to deliver personalized experiences to users. For example, Recommender Systems use machine learning algorithms to suggest products or content tailored to individual preferences.
3. Predictive Analytics: ML models can analyze historical data to make predictions about future events, such as sales forecasting, stock market trends, and disease outbreaks.
4. Continuous Improvement: Machine learning models can continuously learn and improve their performance over time as they are exposed to more data. This adaptability is essential in dynamic environments where conditions change.
5. Handling Unstructured Data: Machine learning can process unstructured data, such as text, images, and audio, which traditional algorithms find challenging to handle. This ability is vital in dealing with the vast amount of unstructured data generated daily.
6. Decision Support: Machine learning can assist decision-making processes by providing data-driven insights and recommendations, helping humans make more informed choices.
7. Innovation and Research: Machine learning fosters innovation in various fields, including medicine, finance, climate science, and more, by providing powerful tools to analyze and understand complex systems.
8. Fraud detection: It is hard to write a program to compute the probability that a credit card transaction is fraudulent.

* There may not be any rules that are both simple and reliable. We need to combine a very large number of weak rules.
* Fraud is a moving target. The program needs to keep changing because the frauds keep changing their techniques.

**1.2 What are Neural Networks**

Neural networks are a class of artificial intelligence (AI) models inspired by the structure and functioning of the human brain. They have gained immense popularity in recent years due to their ability to solve complex problems and achieve state-of-the-art performance in various tasks such as image recognition, natural language processing, speech recognition, and more.

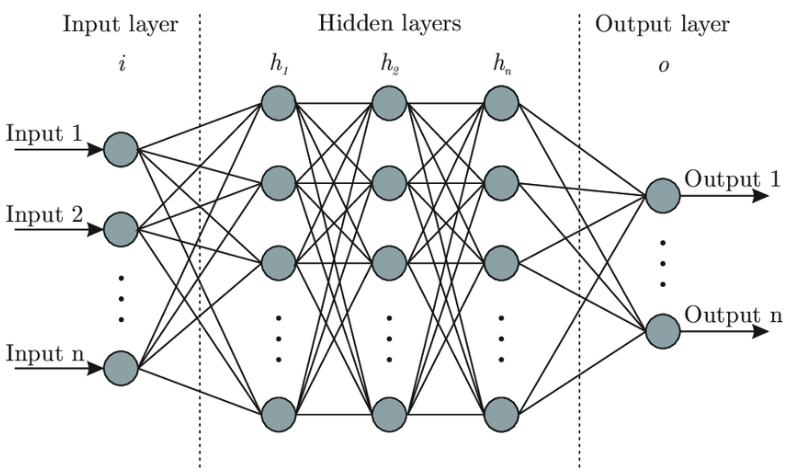


* Perceptron: The basic building block of a neural network is the ***artificial neuron***, also known as a ***perceptron***. Each neuron

1. Receives input from multiple sources,
2. Processes the information, and
3. Produces an output.

* Components of NN: The input is typically multiplied by ***weights***, and an ***activation function*** is applied to the sum of these weighted inputs to generate the output.
* A neural network consists of layers of interconnected neurons. The three main types of layers are:

1. Input layer: This layer receives the raw input data and passes it to the subsequent layers for processing.
2. Hidden layers: These layers come between the input and output layers. They are responsible for learning patterns and representations from the data.
3. Output layer: The final layer of the neural network produces the model's output based on the information learned from the input data.



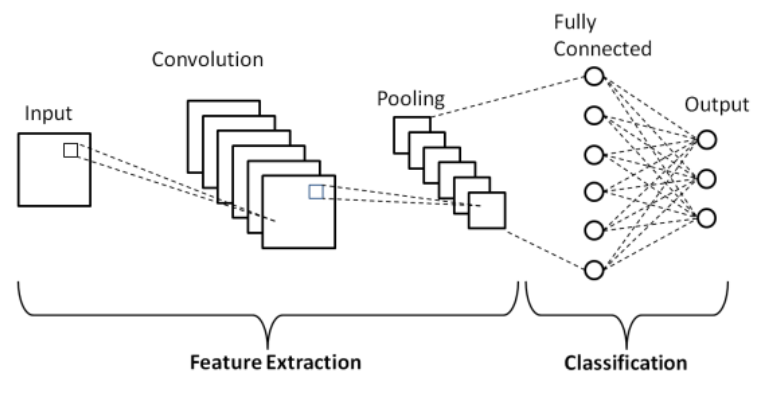
* Training a NN: The connections **(synapses)** between neurons in each layer have associated weights that are adjusted during training to *improve* the network's *performance*.
* Training a neural network involves
* Presenting the Model with Labeled Data (input and corresponding desired output) and
* Using Optimization Algorithms like Gradient Descent to **minimize** the difference between the predicted output and the true output.
* Popular types of neural networks:

1. **FEEDFORWARD Neural Networks (FNN):** The simplest form of neural networks where information flows in one direction, from input to output, without any feedback loops.

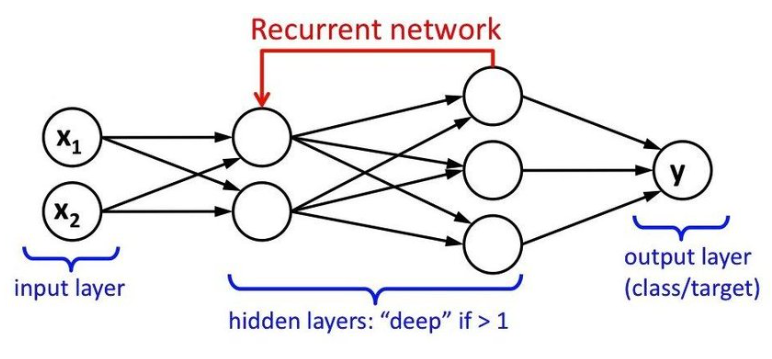
The simplest kind of feedforward neural network is a linear network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. The sum of the products of the weights and the inputs is calculated in each node.

|  |  |
| --- | --- |
| E:\1_Development_2.0\ML_phase_5_NN_note\mlp-diagram.jpg |  |

1. **CONVOLUTIONAL Neural Networks (CNN):** Designed specifically for *image recognition* tasks, CNNs use *convolutional* *layers* to automatically learn *hierarchical patterns* from the input data.

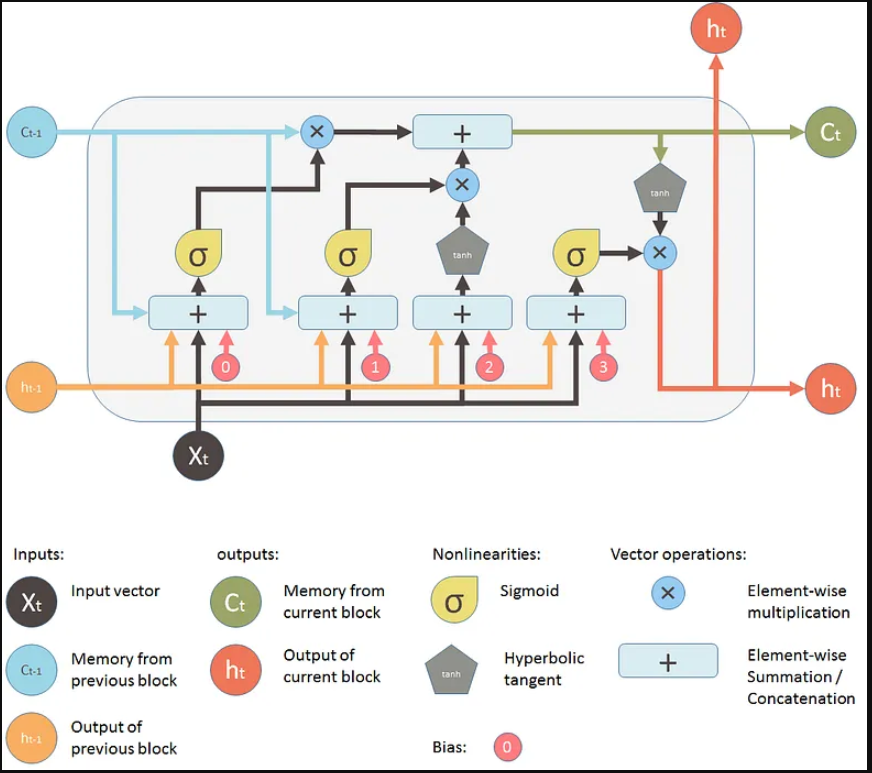


1. **RECURRENT Neural Networks (RNN):** Suited for Sequential Data like Time Series or Natural Language Processing, RNNs have connections that form feedback loops, allowing them to *retain information over time*.

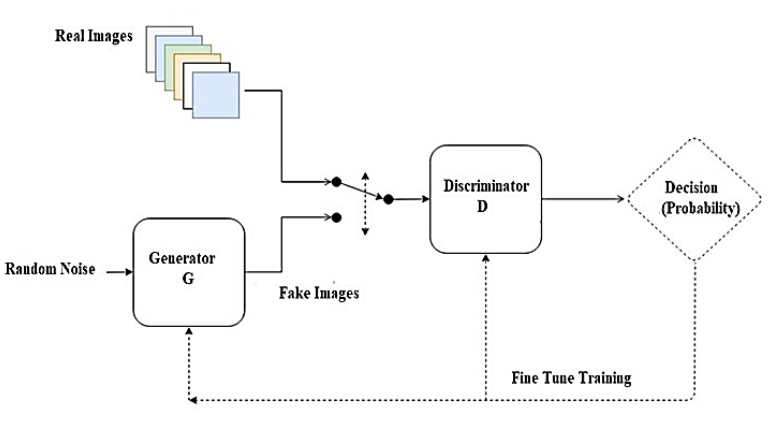


1. **Long Short-Term Memory (LSTM) networks:** A *special* type of RNN designed to overcome the *vanishing gradient* problem and better handle *long-term dependencies* in *sequential data*.

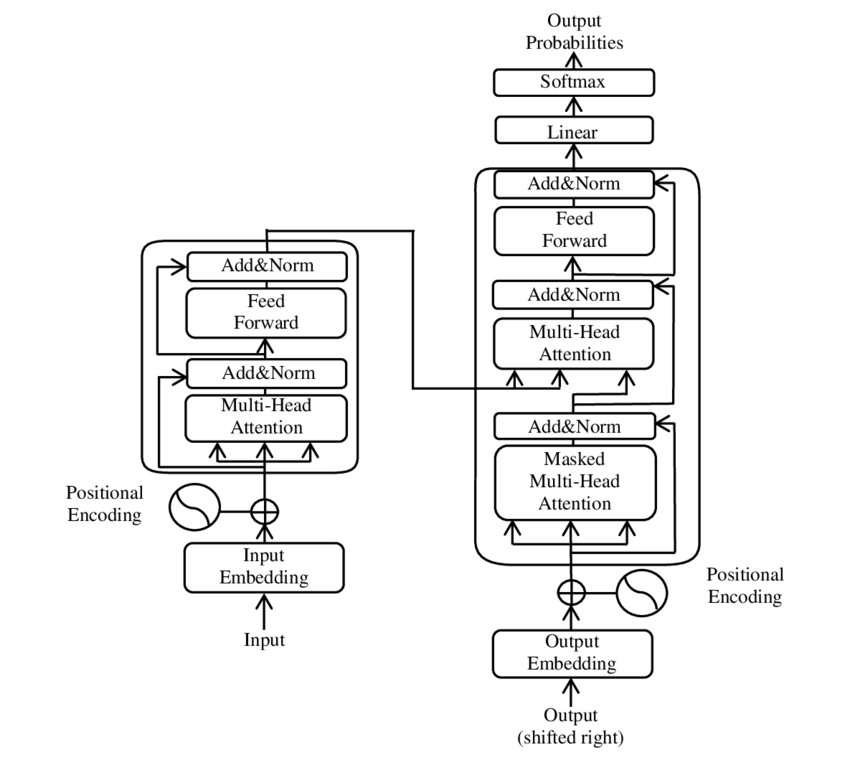
Following is a single cell of an LSTM

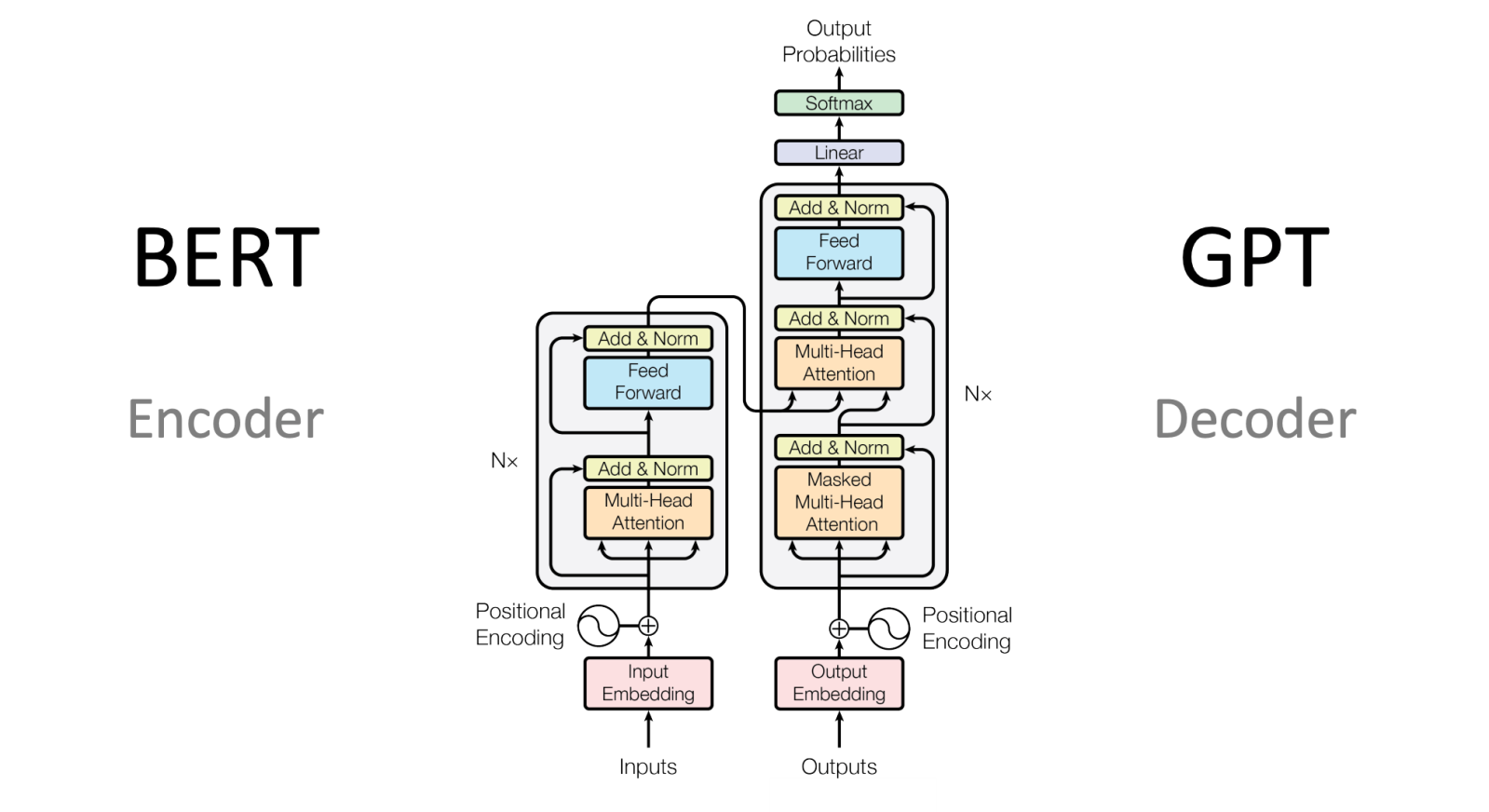


1. **GENERATIVE Adversarial Networks (GAN):** Consisting of two networks (a generator and a discriminator), GANs are used for generating new data samples, such as *realistic* *images* or *audio*.

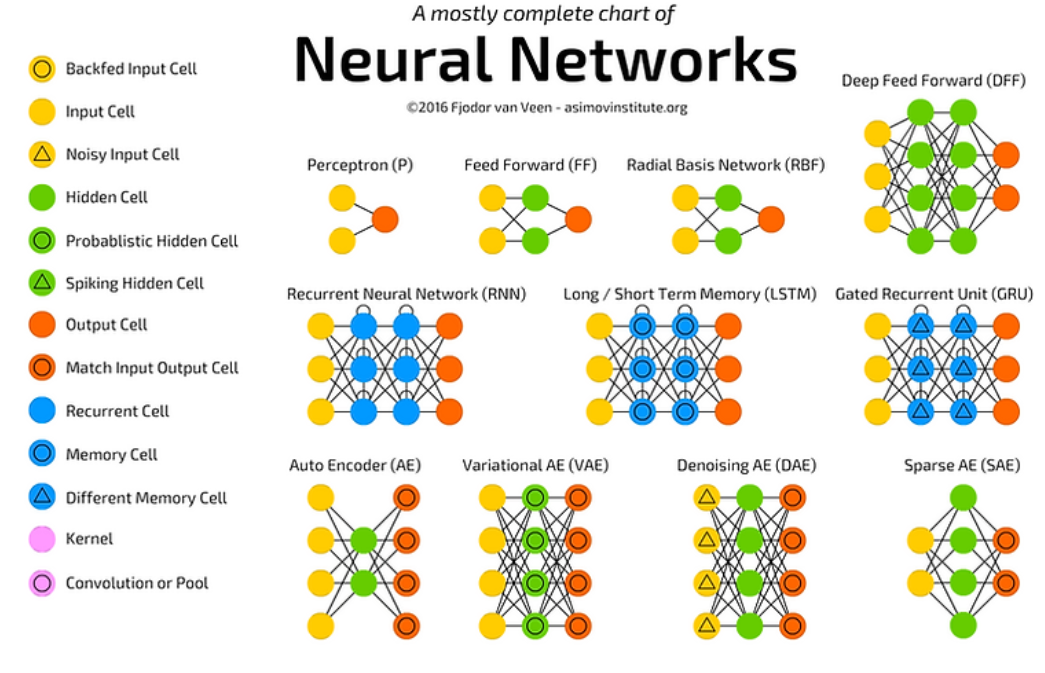


1. **TRANSFORMER:** A model architecture that employs Self-Attention Mechanisms, primarily used in natural language processing tasks and responsible for the state-of-the-art performance in tasks like machine translation (e.g., Google's BERT and OpenAI's GPT series).





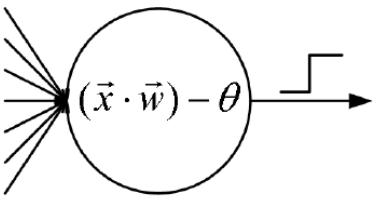
Neural networks have revolutionized the field of AI and have found applications in various industries, ranging from computer vision, speech recognition, and language translation to finance, healthcare, and many others. They continue to be an active area of research, with ongoing efforts to improve their efficiency, scalability, and generalization capabilities.



**1.3 Some simple Models Of Neurons**

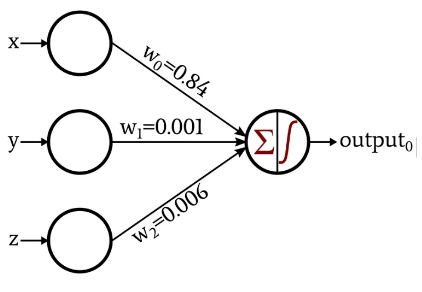
1. Binary Threshold Neuron (McCulloch-Pitts Neuron): Takes *multiple binary inputs*, computes the *weighted sum*, and outputs ***1*** if the sum is ***above*** a certain ***threshold***; otherwise, outputs ***0***.

* Binary threshold neurons have Piecewise Constant Activation Functions such that the derivative of this activation function and thus the weight change is always zero



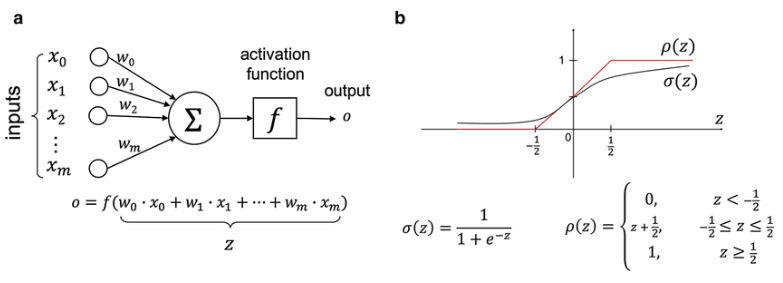
1. Perceptron: An extension of the binary threshold neuron with an ***adjustable bias term***, which allows for *supervised* *learning*. It uses a *step function* as its *activation function*.

* In the context of Neural Networks, a PERCEPTRON is an artificial neuron using the Heaviside Step Function as the *activation function*.
* Perceptron is a Linear Classifier (binary). Also, it is used in supervised learning. It helps to classify the given input data
* It is the simplest possible Neural Network



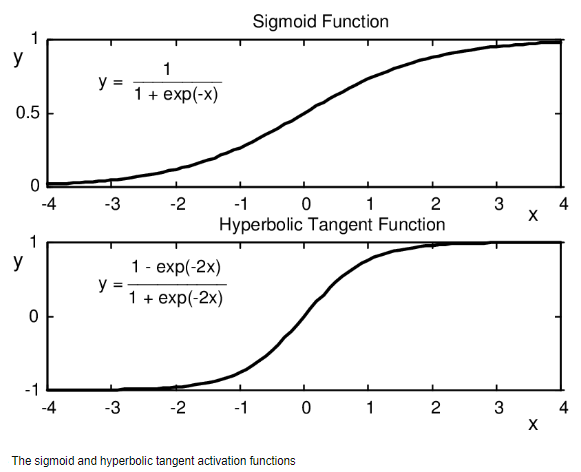
1. Sigmoid Neuron: Takes multiple inputs, computes the weighted sum, and passes the sum through a sigmoid activation function, producing an output in the range **(0, 1)**.

* It is suitable for **problems** involving **probabilities** or **non-linear** relationships.



* The Building Block of the Deep Neural Networks is called the sigmoid neuron.
* Sigmoid neurons are similar to *perceptrons*, but they are slightly modified such that the *output* from the sigmoid neuron is much *smoother* than the *step functional output from perceptron*.

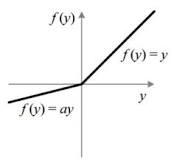
1. Hyperbolic Tangent (Tanh) Neuron: Similar to the sigmoid neuron, but uses the Hyperbolic Tangent Function as the activation function, producing an output in the **range (-1, 1)**.



* The **tanh** function has been mostly used in Recurrent Neural Networks (RNN) for Natural Language Processing (NLP) and Speech Recognition tasks.
* However, the **tanh** function, too, has a limitation – just like the sigmoid function, it cannot solve the vanishing gradient problem
* Activation Function: In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

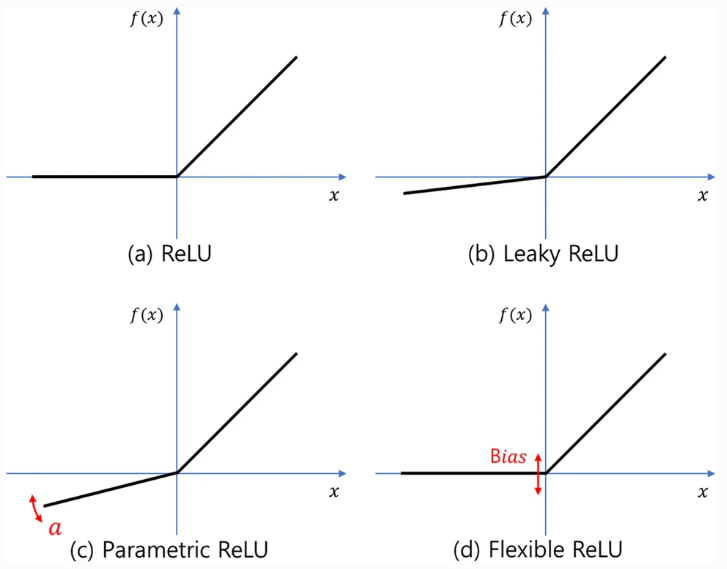
|  |  |
| --- | --- |
| 1. Rectified Linear Unit (ReLU): An activation function widely used in modern neural networks. It **outputs** the **input value** if it's **positive**; otherwise, it outputs **zero**.  * The **Rectified Linear Activation Function** or **ReLU** for short is a *piecewise linear function* that will **output** the **input** **directly** if it is **positive**, otherwise, it will output **zero**. * It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. |  |

1. Leaky ReLU: An extension of **ReLU** that allows a **small non-zero gradient** for **negative inputs**, addressing the "dying ReLU" problem where neurons can become inactive during training.



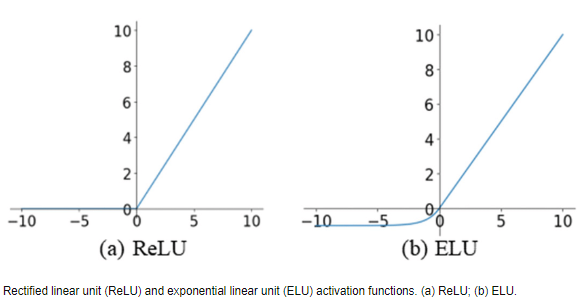
* Leaky Rectified Linear Unit, or Leaky ReLU, is a type of activation function based on a ReLU, but it has a ***small slope*** for ***negative values*** instead of a *flat slope*. The slope coefficient is determined before training, i.e. it is *not learnt during training*.

1. Parametric ReLU (PReLU): A generalization of ReLU where the slope for negative inputs is learned during training rather than being a fixed value.



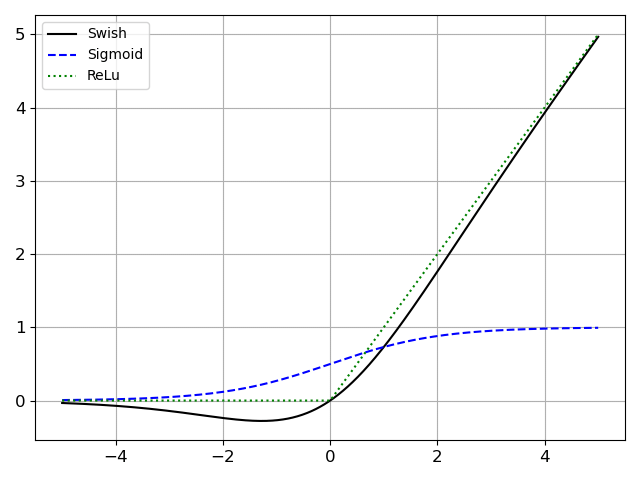
* A **Parametric Rectified Linear Unit**, or **PReLU**, is an activation function that *generalizes* the *traditional rectified unit* with a *slope* for *negative values*.

1. Exponential Linear Unit (ELU): An activation function that behaves as *ReLU* for positive inputs and *smoothly approaches* a *negative exponential* function for *negative inputs*.



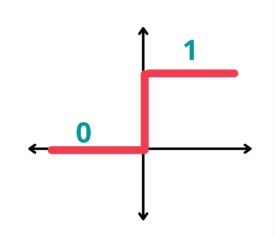
1. Swish: An activation function that multiplies the input by the sigmoid of the input. It combines elements of ReLU and sigmoid activation functions.

* Google Brain Team has proposed a new activation function, named Swish, which is simply **f(x) = x.sigmoid(x)**



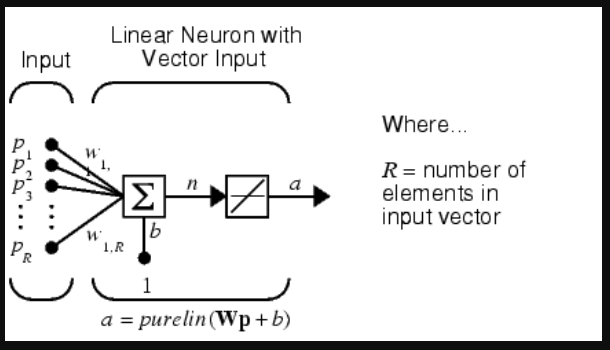
1. Binary Step Neuron: Outputs 1 for positive inputs and 0 for negative inputs. It is a simple binary classifier.

* Binary step function is one of the simplest activation functions. The function produces binary output and thus the name binary step funtion

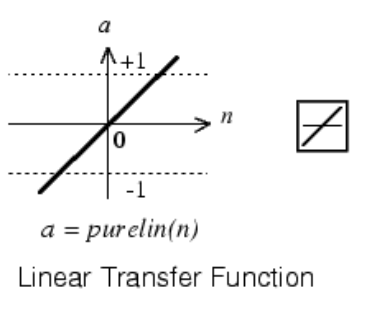


1. Linear Neuron: The output is a linear combination of the inputs, similar to the weighted sum of inputs in other models.

* A linear neuron is trained to respond to specific inputs with target outputs
* The inputs in a linear neuron model can be thought of as the action potentials from other neurons that are impinging upon (strike) the neuron's synapses.
* The ***weights*** can be thought of as the ***efficacies*** (ability or capability) of the synapses. The **larger weight**, the **more affects** on the neurons **output**.
* E.G. A linear neuron with **R** inputs is shown below.

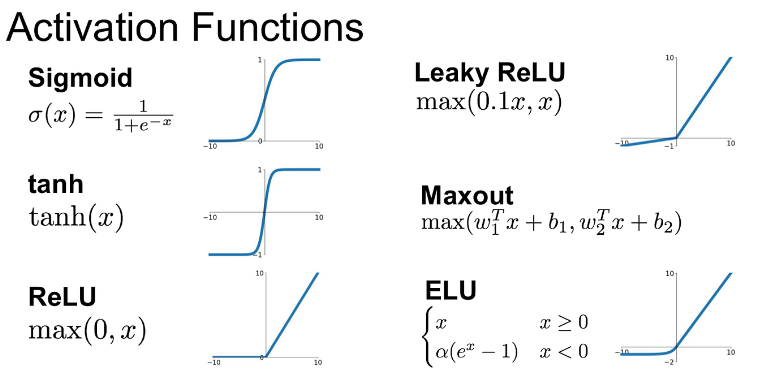


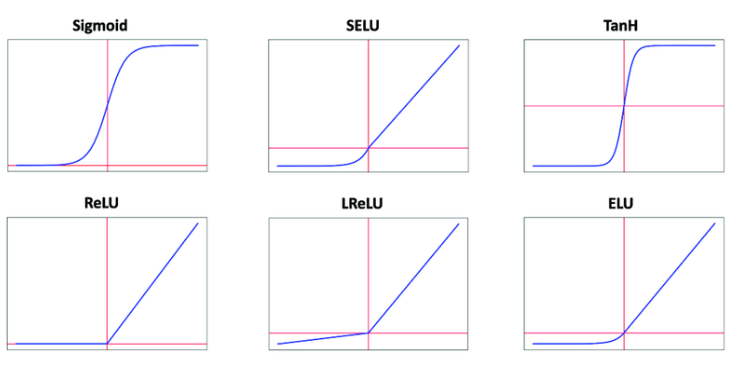
* This network has the same basic structure as the ***perceptron***. The only difference is that the linear neuron uses a linear transfer function, which we name ***purelin***.
* Linear Transfer Function: The linear transfer function calculates the neuron's output by simply returning the value passed to it.





* This neuron can be trained to learn an affine function of its inputs, or to find a linear approximation to a nonlinear function. A linear network cannot, of course, be made to perform a *nonlinear computation*.





* Above are some of the simple models of neurons, each with its activation function and behavior. While some of these models are historically significant, others find practical applications in modern deep learning architectures.
* Neural networks can use different combinations of these neurons to build complex models capable of handling diverse tasks and learning intricate patterns from data.

**1.4 A simple Example of learning**

Let's take a simple example of a perceptron learning to classify points on a 2D plane into two classes: blue and red. The perceptron will learn a decision boundary to separate the two classes using supervised learning.

|  |  |
| --- | --- |
| * Suppose we have these training data: | Blue points: (1, 2), (2, 3), (3, 3)  Red points: (1, 0), (2, 1), (3, 1) |

* Our goal is to train the perceptron to correctly **classify new points** into either the blue or red class based on their coordinates **(x, y)**.
* Here's how the training process might proceed:

1. Initialize the perceptron with random weights (w1 and w2) and a bias term (b).
2. Randomly choose one data point from the training set.
3. Compute the weighted sum of inputs and add the bias term:
4. Apply the step function as the activation function to get the perceptron's output:
5. Compare the perceptron's output with the actual class of the data point.
6. **Adjust** the **weights** and **bias** according to the **error** between the **predicted** **output** and the **true** **class**:

* If the *predicted output* is ***0*** but the *true class* is ***1*** (**false negative**), ***increase*** the ***weights*** and ***bias***.
* If the *predicted output* is ***1*** but the *true class* is ***0*** (**false positive**), ***decrease*** the ***weights*** and ***bias***.

1. Repeat steps 2 to 6 for a fixed number of iterations or until the perceptron reaches a *satisfactory performance level*.
2. **Test** the **trained perceptron** on **new data** **points** to check its **classification accuracy**.

* The *learning process continues* until the perceptron correctly separates the blue and red points with a reasonably low classification error.
* Note that this example uses a simple *perceptron* with a *step function* as the *activation function*, which can only learn *linear decision* *boundaries*.
* For more complex data with non-linear decision boundaries, we may need to use more sophisticated models like neural networks with non-linear activation functions (e.g., **sigmoid**, **ReLU**) or even deeper architectures like ***Multilayer Perceptrons (MLPs)*** or ***Convolutional*** ***Neural Networks (CNNs)***.
* Keep in mind that this example is meant to illustrate the basic concept of learning in a simple setting. In practice, deep learning frameworks like **TensorFlow** or **PyTorch** are commonly used to **train** and build **more complex models** that can handle *real-world* problems with *higher-dimensional* data.

**1.5 Three Types of learning**

The three main types of learning in the context of machine learning are:

1. Supervised Learning: Supervised learning is a type of learning where

* The model is trained on a labeled dataset, meaning that each training example in the dataset has both ***input features*** and the corresponding ***correct output*** (label).
* The goal is:
* To learn the mapping between the input features and the target output based on the given examples.
* Once trained, the model can make predictions on new, unseen data.
* The primary task in supervised learning is to approximate the underlying mapping function that maps inputs to outputs.

Examples of supervised learning tasks include ***image classification*** (assigning labels to images), ***speech recognition*** (converting audio to text), and ***regression*** (*predicting* continuous values, such as *housing prices*).

1. Unsupervised Learning: Unsupervised learning is a type of learning where

* The model is trained on an unlabeled dataset, which means that the data points do not have any corresponding output labels.
* The goal is:
* To discover patterns, structures, or relationships within the data without any specific guidance or predefined categories.
* This type of learning is useful for clustering similar data points or reducing the dimensionality of the data.

Examples of unsupervised learning tasks include clustering (grouping similar data points together based on their features) and dimensionality reduction (reducing the number of input features while preserving essential information).

1. Reinforcement Learning: Reinforcement learning is a type of learning that involves an agent interacting with an environment to learn how to achieve a goal.

* The agent takes actions in the environment, and based on those actions, it receives feedback in the form of rewards or penalties.
* The goal is:
* For the agent to learn an optimal policy (a strategy) that maximizes the cumulative reward over time.
* The agent explores the environment, learns from the consequences of its actions, and adjusts its strategy accordingly.

Reinforcement learning is commonly used in applications like ***robotics***, ***game playing***, and ***autonomous vehicles***.

These three types of learning represent *different approaches* to *teaching machines* how to process and *interpret* data, and they form the *foundation* for various *machine learning algorithms* and techniques used to solve a wide range of real-world problems.

**Best Resources: a list of resource to learn "Neural Networks" from basics**

Here is a list of resources to learn about Neural Networks from the basics:

1. Online Courses:

* **Coursera:** "Neural Networks and Deep Learning" by Andrew Ng. This course provides a solid introduction to ***neural networks*** and ***deep learning concepts***.
* **Udacity:** "Deep Learning Nanodegree Program" covers fundamental neural network architectures and their applications.
* **edX:** "Deep Learning Fundamentals with Keras" introduces the basics of deep learning using Keras.

1. Books:

* "Neural Networks and Deep Learning: A Textbook" by Charu C. Aggarwal. This book provides a comprehensive introduction to neural networks and their applications.
* "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville. This book is a standard reference for deep learning concepts and architectures.

1. Tutorials and Blogs:

* TensorFlow Tutorials: TensorFlow's official website offers beginner-friendly tutorials on building neural networks using TensorFlow.
* PyTorch Tutorials: PyTorch's official website provides tutorials on creating neural networks using PyTorch.
* Medium: There are numerous blog posts and articles on Medium that explain neural network concepts with practical examples.

1. YouTube Channels:

* Sentdex: The "Neural Networks from Scratch" series provides hands-on implementations of neural networks using Python.
* 3Blue1Brown: The "Neural Networks" series offers a visual and intuitive explanation of neural network concepts.

1. GitHub Repositories:

* Neural Networks and Deep Learning: This repository contains Jupyter notebooks with examples and explanations of neural network concepts.
* TensorFlow Tutorials: TensorFlow's GitHub repository has various examples and tutorials for building neural networks.

1. Interactive Platforms:

* Kaggle: Kaggle offers various competitions and tutorials related to neural networks, allowing you to learn by participating in real-world projects.
* Google Colab: Google Colab provides free Jupyter notebooks with GPU support, making it convenient for experimenting with neural networks.
* Remember that learning about neural networks involves both theory and practice.
* It's essential to Implement Neural Networks Hands-On to deepen your understanding. Start with the basics and gradually work your way up to more advanced topics to build a strong foundation in neural networks.

<https://paperswithcode.com/method/prelu>

<https://machinelearningmastery.com/>

<https://deeplearninguniversity.com>

How to Train a Basic Perceptron Neural Network:

<https://www.allaboutcircuits.com/technical-articles/how-to-train-a-basic-perceptron-neural-network/>

Understanding Feedforward Neural Networks (note all the basics from here)

<https://learnopencv.com/understanding-feedforward-neural-networks/>

Understanding LSTM and its diagrams

<https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714>

DPReLU: Dynamic Parametric Rectified Linear Unit and Its Proper Weight Initialization Method

<https://link.springer.com/article/10.1007/s44196-023-00186-w>

Swish

<https://vevesta.substack.com/p/how-activation-function-swish-outperforms>

Power of a Single Neuron

<https://towardsdatascience.com/power-of-a-single-neuron-perceptron-c418ba445095>

Linear Neuron Model

Provide Figures and images.