Neural Networks

Lecture 16

Supervised Learning and Backpropagation
Neural Networks (4)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

Web: www.mijanrahman.com | Email: mijanjkkniu@gmail.com; mijan@jkkniu.edu.bd

Supervised Learning and Backpropagation Neural Network

- **This chapter covers the following topics:**
  - Supervised Learning Network
  - Perceptron
  - Adaptive Linear Neuron (Adaline)
  - Multiple Adaptive Linear Neuron (Madaline)
  - Generalized Delta Learning Rule
  - Backpropagation Neural Networks
  - Backpropagation Algorithm and Network
    - Why We Need Backpropagation?
    - Types of Backpropagation Networks
  - Types of Activation Functions

Why We Need Backpropagation?

- **Most prominent advantages of Backpropagation are:**
  - Backpropagation is a standard method that generally works well.
  - It is fast, simple and easy to implement/program.
  - It has no parameters to tune apart from the numbers of input.
  - It is a flexible method as it does not require prior knowledge about the network. Thus, it does not need any special mention of the features of the function to be learned.

Activation Functions

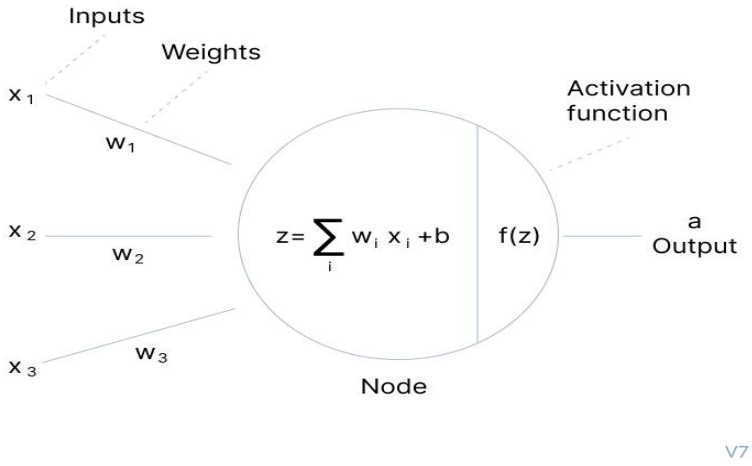
- **What is a Neural Network Activation Function?**

- **Activation Function** helps the neural network to use important information while suppressing irrelevant data points.
- **An Activation Function** decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.
- The role of the **Activation Function** is to derive output from a set of input values fed to a node (or a layer).

Activation Functions

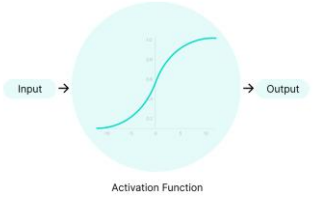
- In machine learning/deep learning, the role of the Activation Function, also referred to as a *Transfer Function* in Artificial Neural Network.

- The primary role of the Activation Function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output.



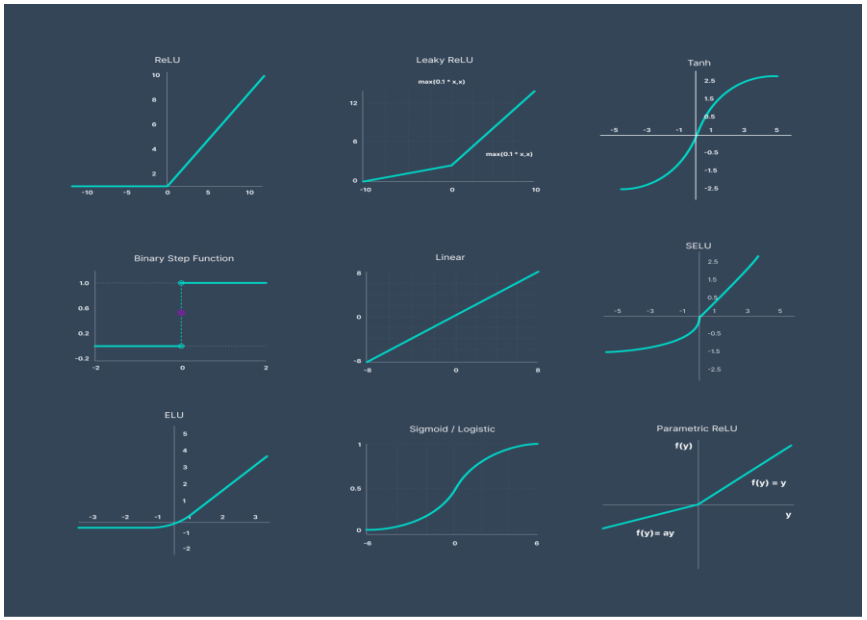
Activation Functions

- **Why do Neural Networks Need an Activation Function?**
- The purpose of an activation function is to add non-linearity to the neural network.
- Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it.
- Let's suppose we have a neural network working *without* the activation functions.
- In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. It's because it doesn't matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself.



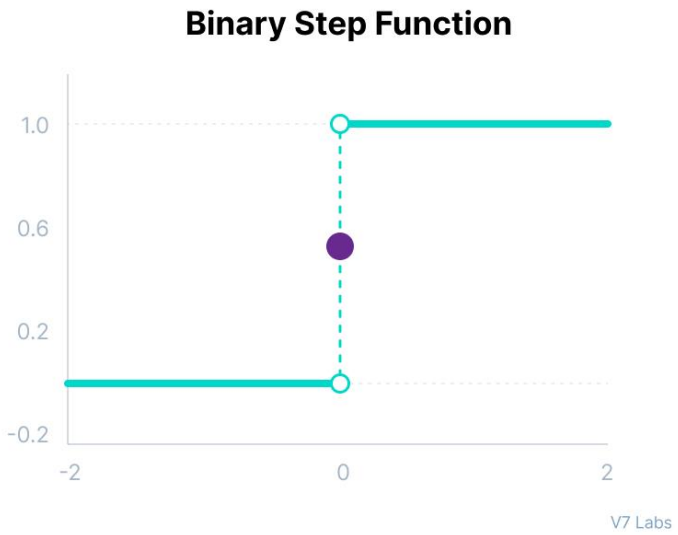
Activation Functions

- **Three Types of Neural Networks Activation Functions**
  - Binary Step Function
  - Linear Activation Function
  - Non-Linear Activation Functions



Binary Step Activation Function

- Binary step function depends on a threshold value that decides whether a neuron should be activated or not.
- The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.



Binary Step Activation Function

- Mathematically, **Binary Step Function** can be represented as:

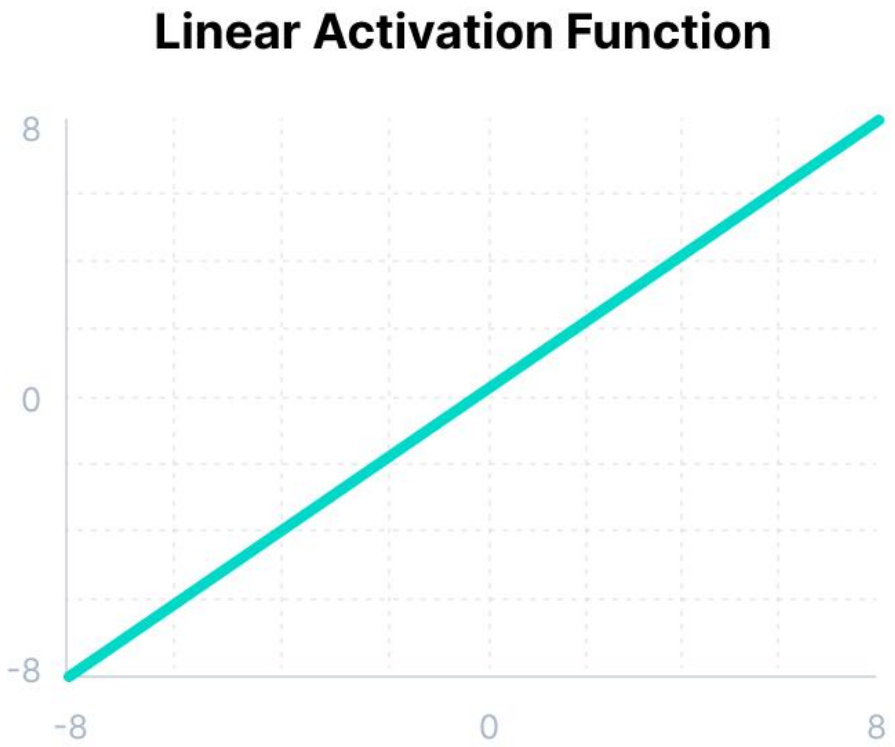
$$\textit{Binary step}$$

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

- Here are some of the limitations of binary step function:
  - It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
  - The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

Linear Activation Function

- The **linear activation function** is also known as *Identity Function* where the activation is proportional to the input.



Linear Activation Function

- Mathematically, Linear Activation Function can be represented as:

$$Linear$$

$$f(x) = x$$

- However, a linear activation function has two major problems :
  - It's not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
  - All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

Non-Linear Activation Functions

- The linear activation function, described earlier, is simply a linear regression model. Because of its limited power, this does not allow the model to create complex mappings between the network's inputs and outputs.
- Non-linear activation functions solve the following limitations of linear activation functions:
  - They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.
  - They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

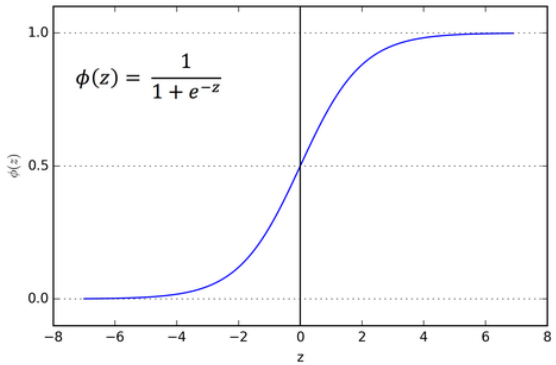
Non-Linear Activation Functions

- **10 (Ten) Non-Linear Neural Networks Activation Functions**
  1. Sigmoid / Logistic Activation Function
  2. Tanh Function (Hyperbolic Tangent)
  3. ReLU Function
  4. Leaky ReLU Function
  5. Parametric ReLU Function
  6. Exponential Linear Units (ELUs) Function
  7. Softmax Function
  8. Swish
  9. Gaussian Error Linear Unit (GELU)
  10. Scaled Exponential Linear Unit (SELU)

Non-Linear Activation Functions

Non-Linear Activation Functions

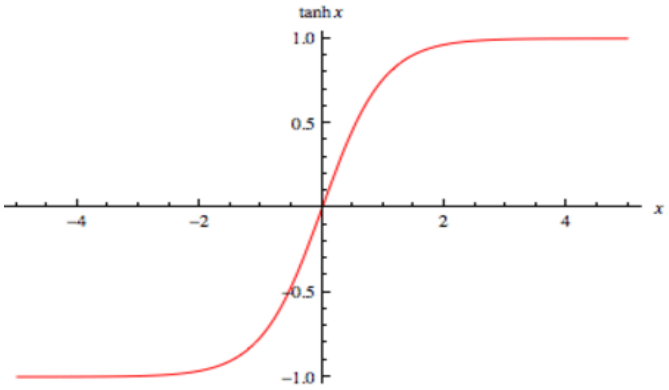
- **Sigmoid or Logistic Activation Function**
- The Sigmoid Function curve looks like a S-shape.



- The main reason why we use sigmoid function is because it exists between **(0 to 1).** Therefore, it is especially used for models where we have to **predict the probability** as an output.Since probability of anything exists only between the range of **0 and 1,** sigmoid is the right choice.
- The function is **differentiable**. That means, we can find the slope of the sigmoid curve at any two points.
- The logistic sigmoid function can cause a neural network to get stuck at the training time. The **softmax function** is a more generalized logistic activation function which is used for multiclass classification.

Non-Linear Activation Functions

- **Tanh or hyperbolic tangent Activation Function**
- **tanh** is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).
- **Hyperbolic Tangent Function: $\tanh(x) = (e^x – e^{-x}) / (e^x + e^{-x})$**
- The function produces outputs in scale of [-1, +1]. Moreover, it is continuous function. In other words, function produces output for every x value.

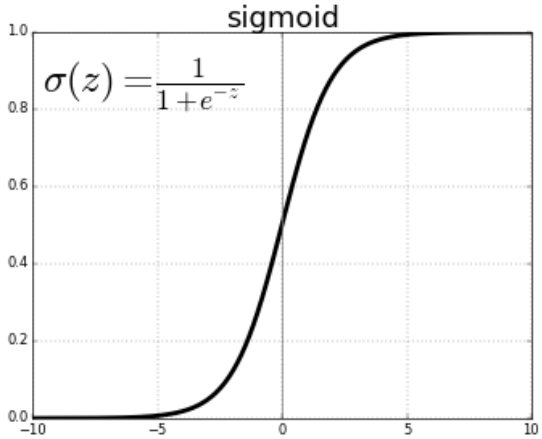


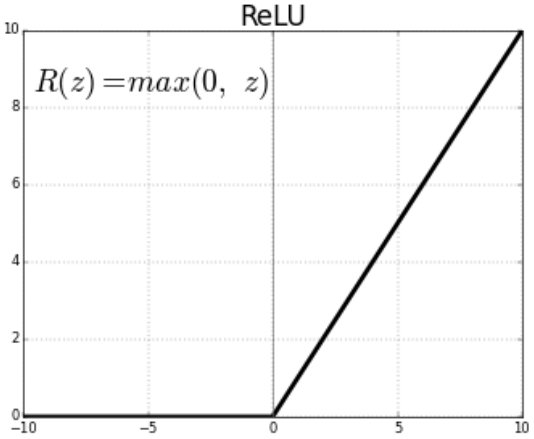
- The tanh function is mainly used classification between two classes. *Both tanh and logistic sigmoid activation functions are used in feed-forward nets.*

Non-Linear Activation Functions

- **ReLU (Rectified Linear Unit) Activation Function**
- The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.
- The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

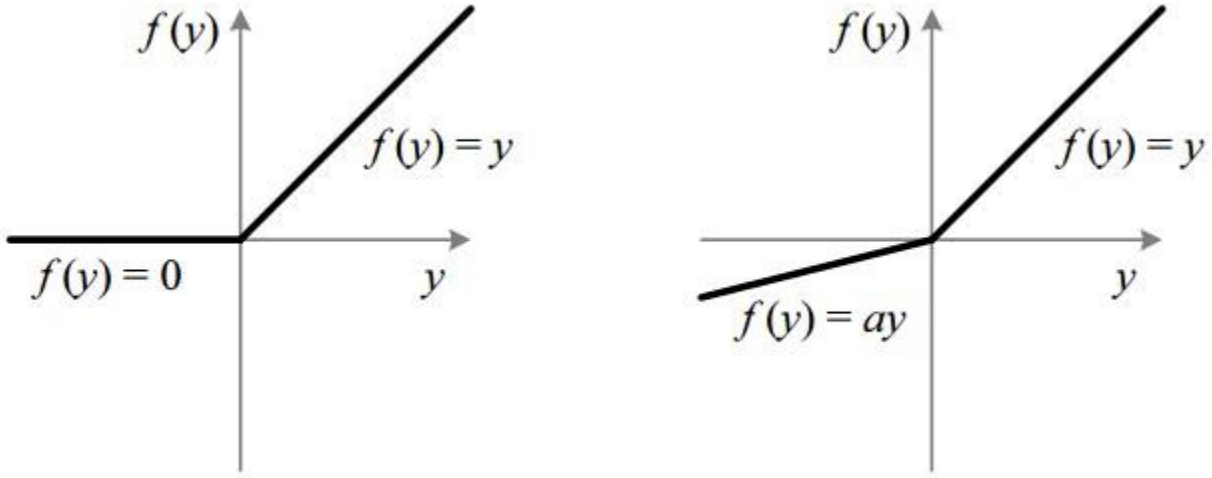
- **Range:** [0 to infinity)





Non-Linear Activation Functions

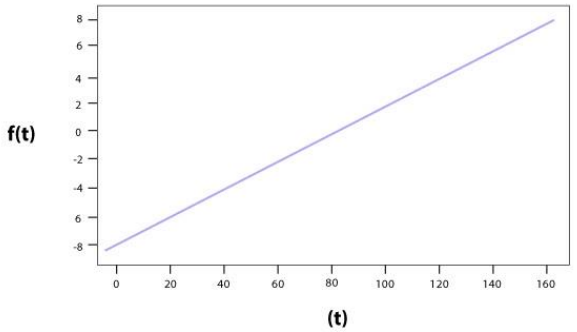
- **Leaky ReLU**
- It is an attempt to solve the dying ReLU problem.
- The **leak** helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so. When **a is not 0.01** then it is called **Randomized ReLU**.
- Therefore the **range** of the Leaky ReLU is (-infinity to infinity).



Non-Linear Activation Functions

- **Softmax Activation Functions**
- Softmax is used mainly at the last layer i.e output layer for decision making the same as sigmoid activation works. The softmax function outputs a vector of values that sum to 1.0 that can be interpreted as probabilities of class membership.
- It is related to the argmax function that outputs a 0 for all options and 1 for the chosen option. Softmax is a "*softer*" version of argmax that allows a probability-like output of a winner-take-all function.
- As such, the input to the function is a vector of real values and the output is a vector of the same length with values that sum to 1.0 like probabilities. The softmax function is calculated as follows:

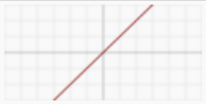
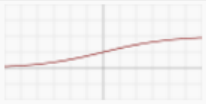
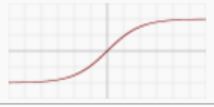
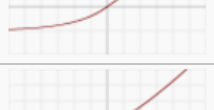
  f(x) = e^x / sum(e^x)



Softmax on Binary Classification

- For **Binary classification**, both **sigmoid**, as well as **softmax**, are equally approachable but in case of multi-class classification problem we generally use softmax and cross-entropy along with it.

Activation Functions and their Derivatives

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

How to choose the right Activation Function?

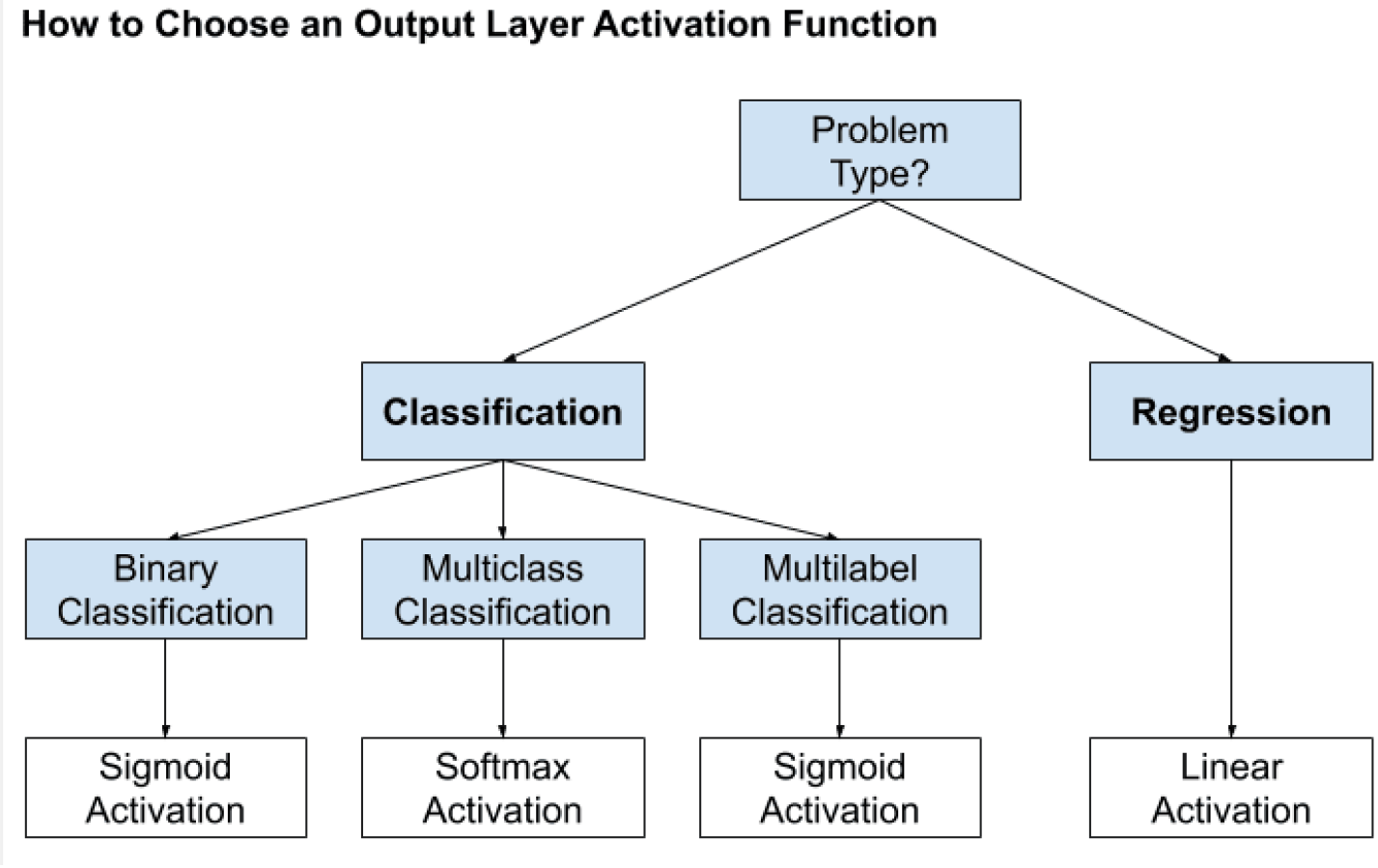
- You need to match your activation function for your output layer based on the type of prediction problem that you are solving—specifically, the type of predicted variable.
- Here's what you should keep in mind.
  - As a rule of thumb, you can begin with using the ReLU activation function and then move over to other activation functions if ReLU doesn't provide optimum results.
- And here are a few other guidelines to help you out.
  1. ReLU activation function should only be used in the hidden layers.
  2. Sigmoid/Logistic and Tanh functions should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients).
  3. Swish function is used in neural networks having a depth greater than 40 layers.

How to choose the right Activation Function?

- Finally, a few rules for choosing the activation function for your output layer based on the type of prediction problem that you are solving:
  - Regression - Linear Activation Function
  - Binary Classification - Sigmoid/Logistic Activation Function
  - Multiclass Classification - Softmax
  - Multilabel Classification - Sigmoid

- The activation function used in hidden layers is typically chosen based on the type of neural network architecture.
  - Convolutional Neural Network (CNN): ReLU activation function.
  - Recurrent Neural Network: Tanh and/or Sigmoid activation function.

How to choose the right Activation Function?



Types of Backpropagation Networks

- **Two Types of Backpropagation Networks are:**
  - Static Back-propagation
  - Recurrent Backpropagation

- **Static back-propagation:** It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.
- **Recurrent Backpropagation:** Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.

- The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation.

Key Points

- Backpropagation is especially useful for deep neural networks working on error-prone projects, such as image or speech recognition.
- Backpropagation takes advantage of the chain and power rules allows backpropagation to function with any number of outputs.
- The actual performance of backpropagation on a specific problem is dependent on the input data.
- Back propagation algorithm in data mining can be quite sensitive to noisy data.

SUPERVISED LEARNING AND BACKPROPAGATION NETWORK PARADIGMS

The End.