Unit 2
The Neural Network

Prepared By: Arjun Singh Saud, Asst. Prof. CDCSIT

Building Intelligent Machines

- To build intelligent machines we have to incorporate AI capabilities in machines.
- Nowadays, artificial intelligence, machine learning, and deep learning are mostly talked topics concerned with making intelligent machine.
- AI is the field of computer science that deals with art and science of making intelligent mahines.
- When we talk about traditional AI techniques of making intelligent machines, it simply refers to programmed intelligence that imitate intelligent human behavior.

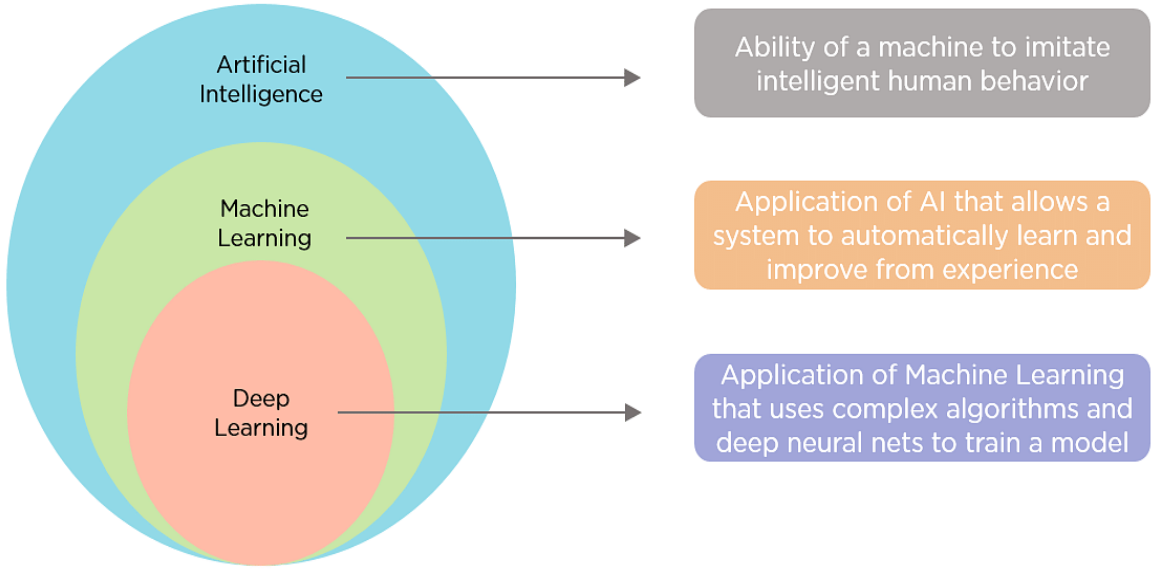
Building Intelligent Machines

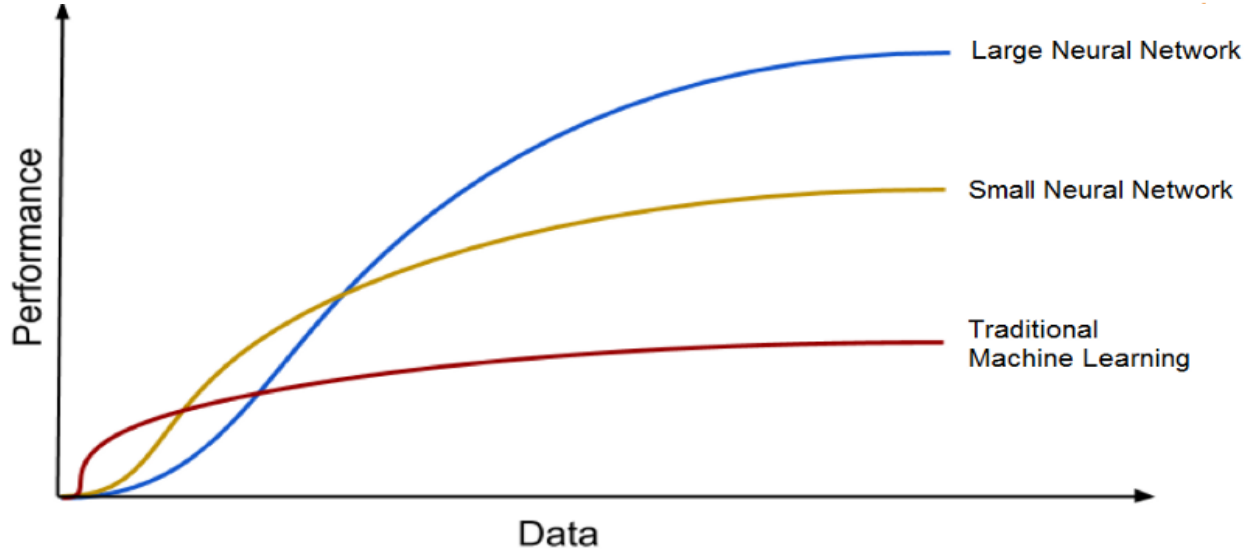
- Traditional AI technique simply shows intelligence on the basis of rule base.
- Machine learning is subset of AI that uses a series of algorithms that analyze data, learn from it and make informed decisions based on those learned insights.
- Performance of ML algorithms improve as they are exposed to more data over time.
- Deep learning is subset of machine learning that use multilayer neural network to learn from vast amount of data.

Building Intelligent Machines

- Deep learning is the recent an closest approach of building intelligent machines.
- Deep learning models tend to increase their accuracy with the increasing amount of training data, whereas traditional machine learning models such as SVM and Naïve Bayes classifier stop improving after a saturation point.
- To paraphrase Andrew Ng, the chief scientist of China's major search engine Baidu, co-founder of Coursera, and one of the leaders of the Google Brain Project, if a deep learning algorithm is a rocket engine, data is the fuel.

Building Intelligent Machines





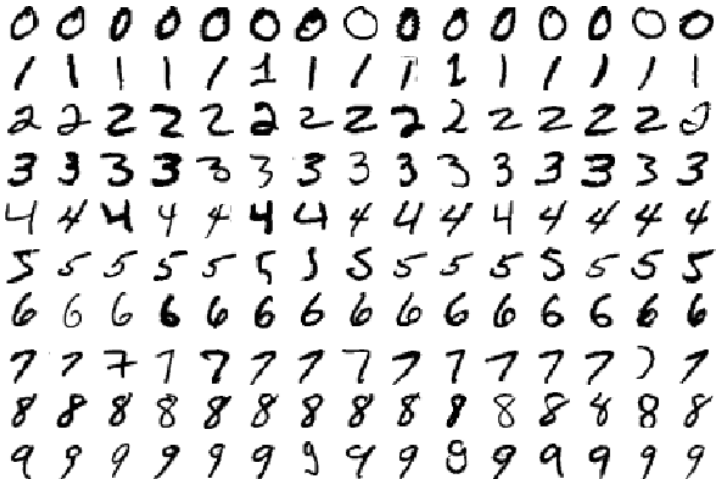
Building Intelligent Machines

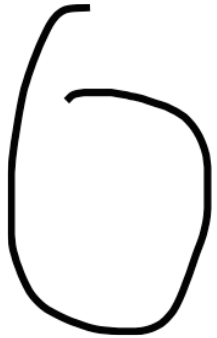
Machine learning	Deep learning
A subset of AI	A subset of machine learning
Can train on smaller data sets	Requires large amounts of data
Requires more human intervention to correct and learn	Learns on its own from environment and past mistakes
Shorter training and lower accuracy	Longer training and higher accuracy
Makes simple, linear correlations	Makes non-linear, complex correlations
Can train on a CPU (central processing unit)	Needs a specialized GPU (graphics processing unit) to train

Limits of Traditional Computer Programs

- Traditional computer programs are designed to be very good at two things:
  - performing arithmetic fast and
  - explicitly following a list of instructions
- However, if we want to do something interesting and smart tasks such as reading and recognizing handwritten digits, its is very difficult to write programs that performs above task accurately because topology of the digits written by individuals can be different.

Limits of Traditional Computer Programs





Limits of Traditional Computer Programs

- Additionally, handwritten digits may not follow topology of digits perfectly.
- Other problems that fall into above category are speech recognition, language translation, object recognition etc.
- These type of problems are very hard to solve by writing traditional computer programs. We need to devise ML or DL programs to solve such problems.

Mechanics of Machine Learning

- Let's define our model to be a function $h(x, \theta)$. The input x is an example expressed in vector form.
- The input θ is a vector of the parameters that our model uses. Machine learning program tries to perfect the values of these parameters as it is exposed to more and more examples.
- Let's say we wanted to determine how to predict exam performance based on the number of hours of sleep we get (x_1) and the number of hours we study the previous day (x_2).

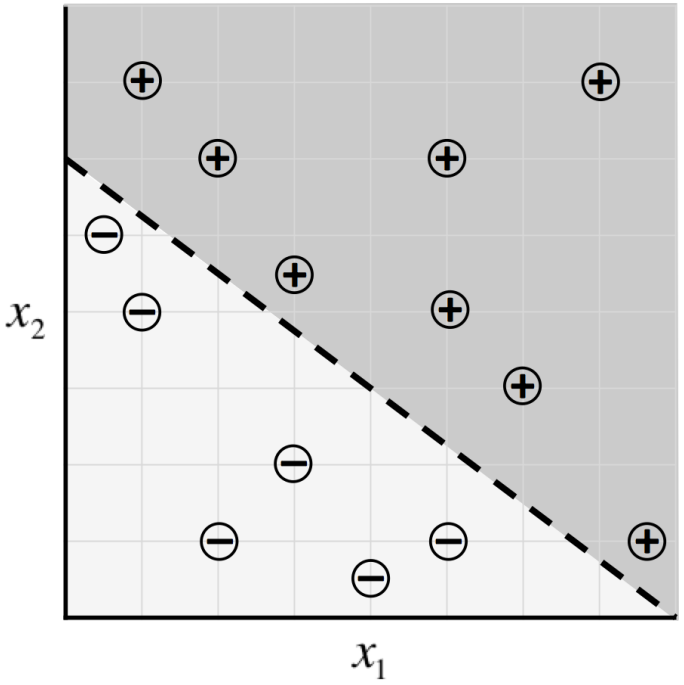
Mechanics of Machine Learning

- Our goal, then, might be to learn a model $h(x, \theta)$ with parameter vector $[\theta_0 \ \theta_1 \ \theta_2]$ such that our model makes the right predictions (-1 if performance is below average, and 1 otherwise) given an input example $\boldsymbol{x}$.

$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 < 0 \\ 1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 \geq 0 \end{cases}$$

Mechanics of Machine Learning

- Let's assume our data is as shown in the Figure below. Suppose that parameters are [-24,3,4] then output will be computed as below

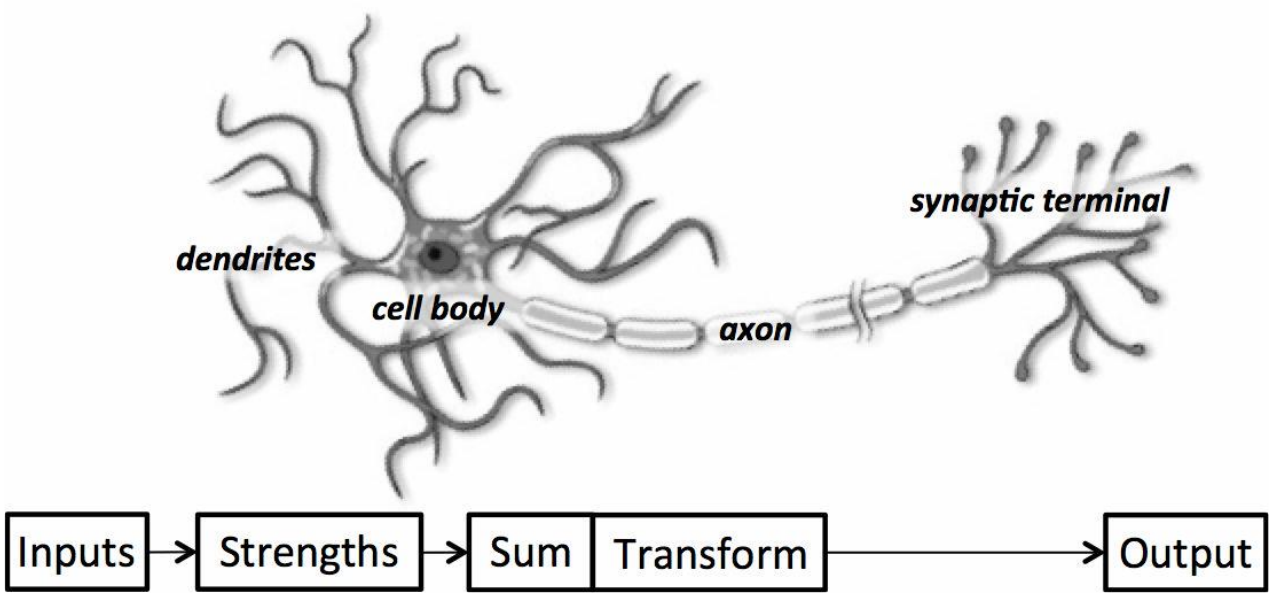


$$h(\mathbf{x},\theta) = \begin{cases} -1 & \text{if } 3x_1 + 4x_2 - 24 < 0 \\ 1 & \text{if } 3x_1 + 4x_2 - 24 \geq 0 \end{cases}$$

The Neuron

- The foundational unit of the human brain is the neuron. A tiny piece of the brain, about the size of grain of rice, contains over 10,000 neurons, each of which forms an average of 6,000 connections with other neurons. This massive biological network enables us to experience the world around us.
- ANN is a machine learning models that solve problems in an analogous way.

The Neuron

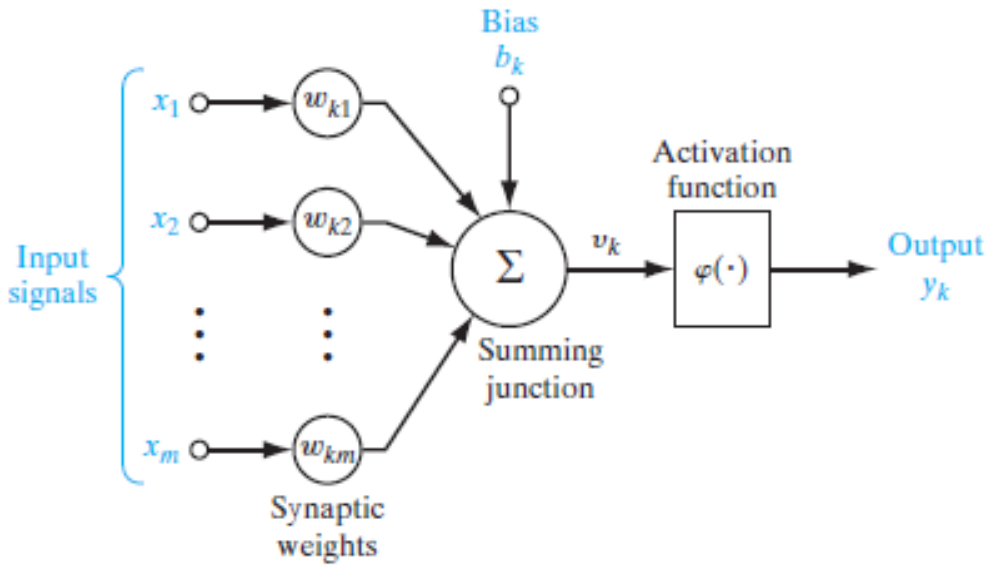

synaptic terminal
dendrites
cell body
axon
Inputs
Strengths
Sum
Transform
Output

The Neuron

- The neuron receives its inputs along antennae-like structures called *dendrites*.
- Each of these incoming connections is dynamically strengthened or weakened based on how often it is used.
- Strength of each connection determines the contribution of the input to the neuron's output.
- After being weighted by the strength of their respective connections, the inputs are summed together in the *cell body*.
- This sum is then transformed into a new signal that's propagated along the cell's *axon* and sent off to other neurons.

The Neuron

- We can translate this functional understanding of the neurons in our brain into an artificial model that we can represent on our computer as shown in the figure below.
- This a model is first described by Warren S. McCulloch and Walter H. Pits.



The Neuron

- Just as in biological neurons, our artificial neuron takes in some number of inputs, $x_1, x_2, \ldots, x_n$, each of which is multiplied by a specific weight, $w_{k1}, w_{k2}, \ldots, w_{kn}$.
- These weighted inputs are, as before, summed together to produce net input for the neuron.

$$u_k = \sum_{j=1}^{n} x_j * w_{kj}$$

The Neuron

- In many cases, the net input also includes a *bias*, which is a constant.

$$v_k = u_k + b_k$$

- The net input is then passed through a function *f* to produce the output *y*. This function is called activation function. This output can be transmitted to other neurons.

$$y_k = \varphi(u_k + b_k) = \varphi(v_k)$$

Expressing Linear Perceptron as Neuron

- In "The Mechanics of Machine Learning", we talked about using machine learning models to capture the relationship between success on exams and time spent studying and sleeping.

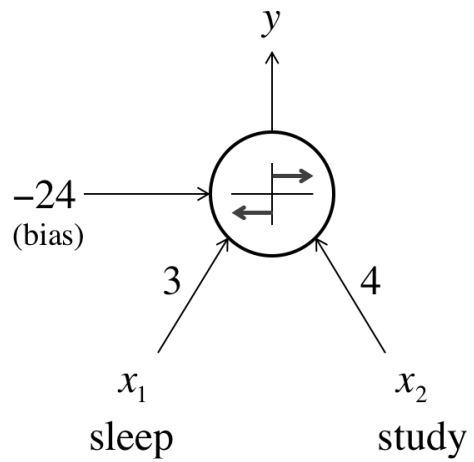
$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } 3x_1 + 4x_2 - 24 < 0 \\ 1 & \text{if } 3x_1 + 4x_2 - 24 \geq 0 \end{cases}$$

- Here we will construct a linear perceptron classifier to tackle this problem. The perceptron should divide the Cartesian coordinate plane into two halves:

Expressing Linear Perceptron as Neuron

- Consider the neuron depicted in Figure below. The neuron has two inputs, a bias, and uses the given activation function.

$$f(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$



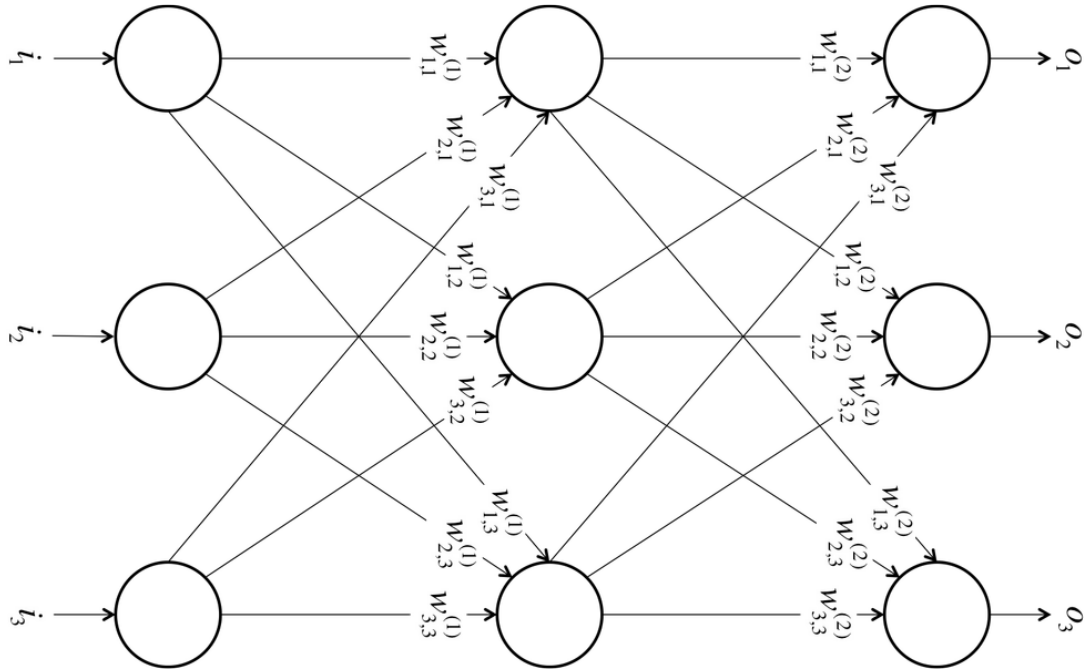
- It's very easy to show that our linear perceptron and the neuron model are perfectly equivalent.

Feed-Forward Neural Networks

- A Feed Forward Neural Network (FFNN) is an artificial neural network in which the connections between nodes does not form a cycle.
- Simplest FFNN may contain input layer and output layer only, Where input layer sends input signal to the network and output layer neurons performs necessary computations and produces output.
- However, to solve complex problem we may need to construct multi-layer FFNN. These networks contains one or more hidden layers of neurons.

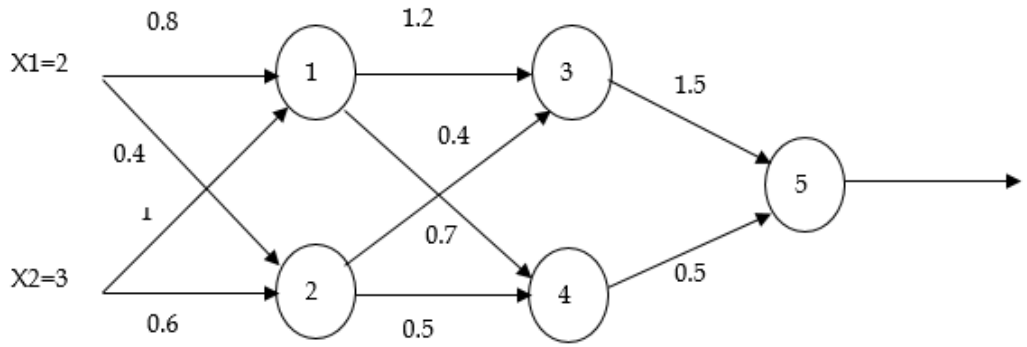
Feed-Forward Neural Networks

- Hidden layer neurons are just responsible for performing computations, intermediate results produced by hidden layer neurons is sent to neurons of next layer as input.



Feed-Forward Neural Networks

- **<u>Example:</u>** *Consider following Neural Network and compute its output using activation function F(x)=2x-1. Weights of synaptic links are provided above each link.*



Feed-Forward Neural Networks

For Node 1

*u1=2*0.8+3*1=4.6* *y1=f(u1)=2*4.6-1=8.2*

For Node 2

*u2=2*0.4+3*0.6=2.6* *y2=f(u2)=4.2*

For Node 3

*u3=8.2*1.2+4.2*0.4=11.51* *y3=f(u3)=22.04*

For Node 4

*u4=8.2*0.7+4.2*0.5=7.84* *y4=f(u4)=14.68*

For Node 5

*u5=22.04*1.5+14.68*0.5=40.4* *y5=f(u5)=79.8*

Thus,

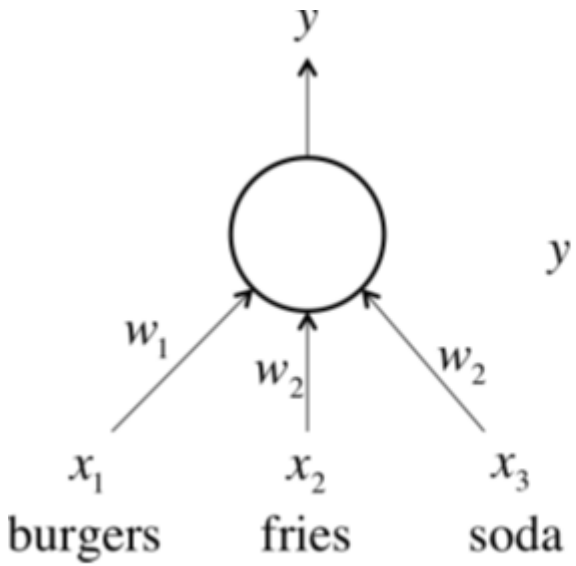
Final output of the neural network (y)=79.8

Linear Neurons and Their Limitations

- Neurons that use linear activation function are called linear neurons.
- Neural network in which every neuron use linear activation function is called linear neural network.
- Such neurons or neural networks can only solve problems where linear relationship exists between input and output.
- Activation function of the form $f(x)=ax+b$ is called linear activation function.

Linear Neurons and Their Limitations

- For example, consider a neuron that attempts to estimate a cost of a meal in a fast-food restaurant would use a linear neuron where a = 1 and b = 0.



$$y = w_1x_1 + w_2x_2 + w_3x_3$$

Linear Neurons and Their Limitations

- Linear neurons or networks are not suitable for solving complex problems.
- Neural networks that only contains linear neurons can be represented as single layer neural network.
- Thus, expressive power of linear neural networks is very limited.

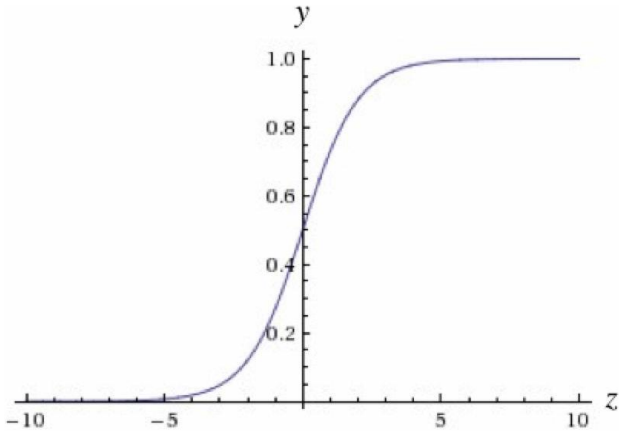
Sigmoid, Tanh, and ReLU Neurons

- There are three major types of neurons that are used in practice that introduce nonlinearities in their computations. The first of these is the *sigmoid neuron*, which uses the activation function:

$$f(x) = \frac{1}{1+e^{-ax}}$$

- Sigmoid function squashes the output between 0 and 1. It is the class of functions whose graph is S-shaped curve.
- By varying the parameter *a (slope)*, we obtain sigmoid functions of different slopes.

Sigmoid, Tanh, and ReLU Neurons

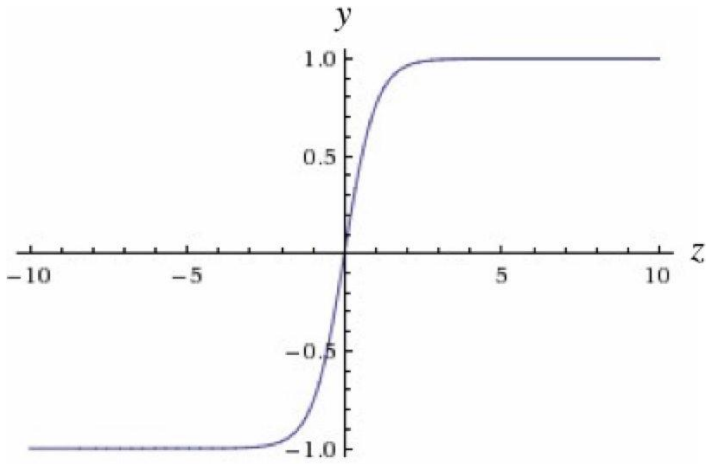


- It is the most common form of activation function used in the construction of neural networks. Logistic function is the sigmoid function *where a=1*.

Sigmoid, Tanh, and ReLU Neurons

- Tanh neurons use a similar kind of S-shaped nonlinearity, but instead of ranging from 0 to 1, the output of tanh neurons range from −1 to 1.
- Tanh is also a very popular and widely used activation function. It is special case of sigmoid function.

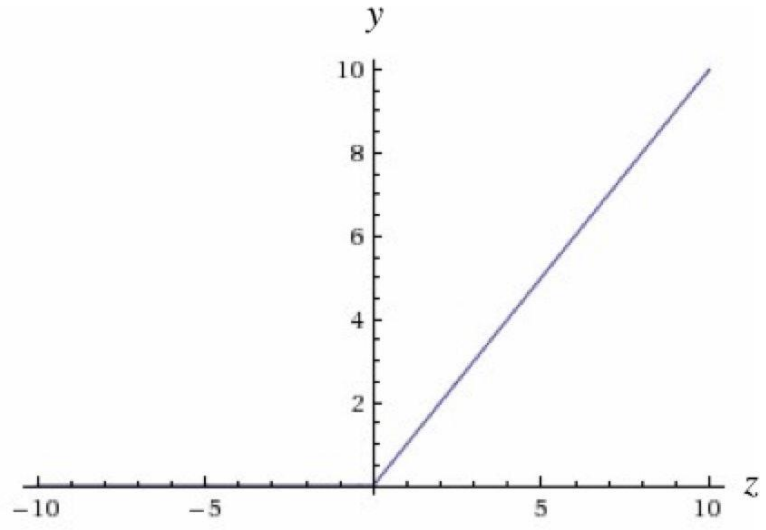
$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1$$



Sigmoid, Tanh, and ReLU Neurons

- The rectified linear activation function or ReLU is a non-linear function or piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It is the most commonly used activation function in neural networks.

$$f(x) = \max(0, x)$$



Sigmoid, Tanh, and ReLU Neurons

- It is simple yet it is more effective than it's predecessors like sigmoid or tanh.
- ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations.
- ReLU is much helpful to tackle with vanishing gradient problems associated with deep neural networks.

Softmax Output Layers

- Softmax is fundamentally a vector function. It takes a vector as input and produces a vector as output.
- The Softmax function also squashes the outputs of each unit to be between 0 and 1.
- But it also divides each output such that the total sum of the outputs is equal to 1.

$$S(x) = \frac{e^{x}}{\sum_{i=1}^{n} e^{x_i}} \quad for\ i = 1,2,3.....n$$

Softmax Output Layers

- Softmax activation function is normally used in output layer neurons of neural network when multiclass classification problem need to be solved.
- We can also use softmax activation function with output layer neurons of neural networks that aims to solve binary class classification problems.
- However, sigmoid of tanh activation function can also be used in such case.