Deep Learning-I

(Introduction to Deep Learning, Neural Networks, Logistic Regression as Neural Network)

DR. JASMEET SINGH,
ASSISTANT PROFESSOR,
CSED, TIET

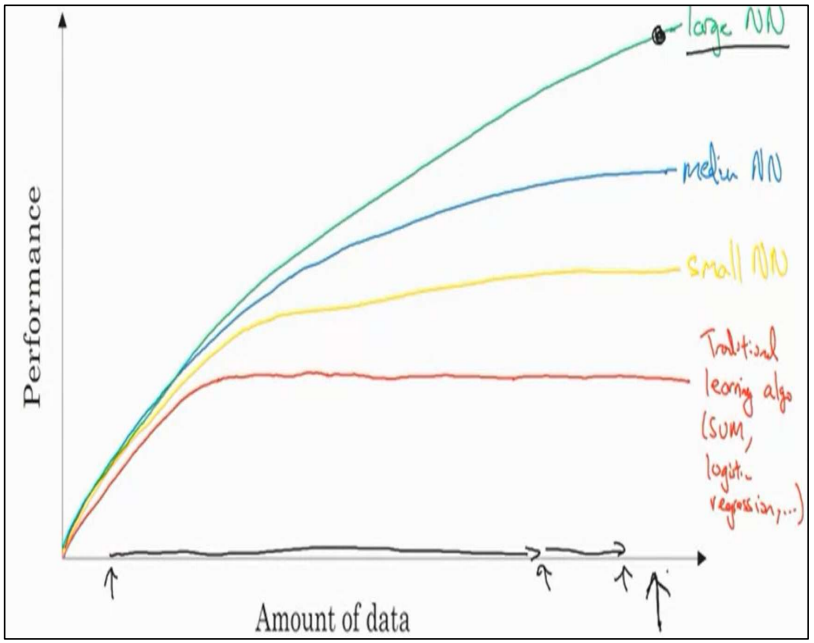
What is Deep Learning?

- Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning.
- Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolutional neural networks have been applied to various fields such as:
  - computer vision,
  - speech recognition,
  - natural language processing, machine translation,
  - bioinformatics, drug design, medical image analysis,
  - climate science, material inspection and board game programs,

  where they have produced results comparable to and in some cases surpassing human expert performance.

Why is Deep Learning taking off?

- The following graph shows improvement in the performance with the increase in size for different models.
- It is clear from machine learning models, the performance is not much affected by the size of data.
- But for neural networks models, there is large improvement with increase in size of data and neural network.
- Thus, it is clear from the graph, that for deep learning architectures to perform well, **data for training models**, **computational power for large sized deep learning models**, and **development of deep learning algorithms** is very important.
- Thus, the following factors are the reasons for rise of deep learning algorithms:
  - Data
  - Computational Power
  - Algorithmic Development.



What is a Neural Network?

- A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.
- In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.
- A “neuron” in a neural network is a mathematical function that collects and classifies information according to a specific architecture.
- Neural networks are broadly used, with applications for financial operations, enterprise planning, trading, business analytics, and product maintenance.

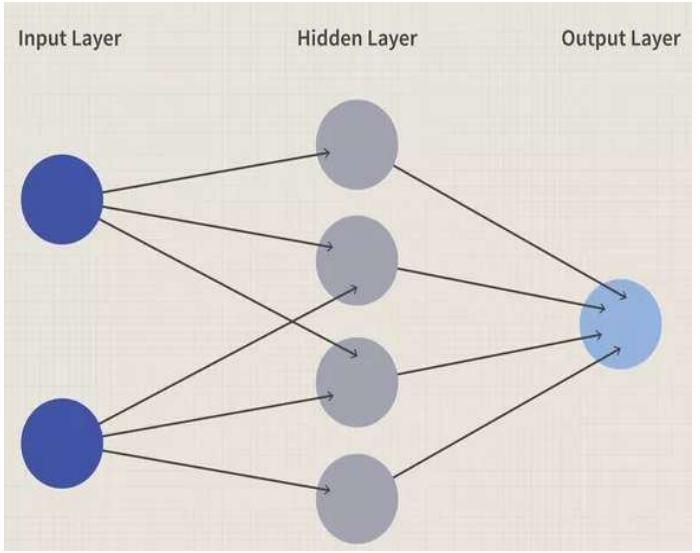
Supervised Learning with Neural Networks

- Neural Networks works well for supervised learning tasks both with structured data. Some of the applications of Neural Networks are listed below.

Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,...,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation
Image, Radar info	Position of other cars	Autonomous driving

Components of Neural Networks

- There are three main components: an input later, a processing (hidden) layer, and an output layer.
- The input layer collects input patterns.
- The output layer has classifications or output signals to which input patterns may map.
- Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal.
- It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs.
- While computing the total layers in neural network, we donot count the input layer.
- In a Neural Network, **the depth** is its number of layers including output layer but not input layer. **The width** is the maximum number of nodes in a layer.

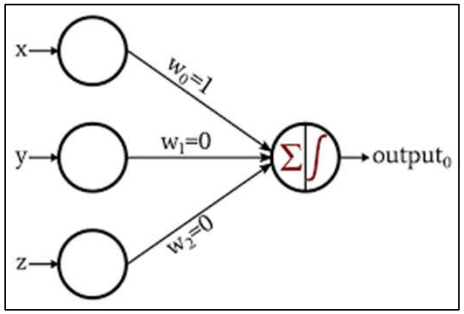


Types of Neural Networks

- Different types of neural networks are used for different data and applications. The different architectures of neural networks are specifically designed to work on those particular types of data or domain.

1. Perceptron

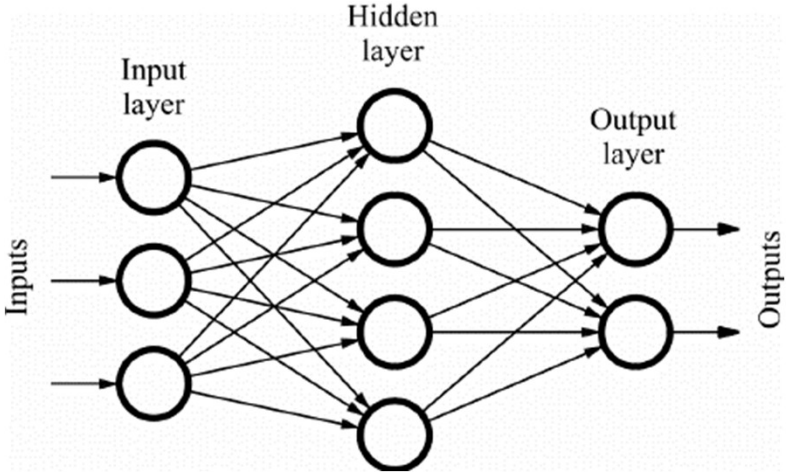
The Perceptron is the most basic and oldest form of neural networks. It consists of just 1 neuron which takes the input and applies activation function on it to produce a binary output. It doesn't contain any hidden layers and can only be used for binary classification tasks.



Types of Neural Networks (Contd….)

2. Feed Forward Network

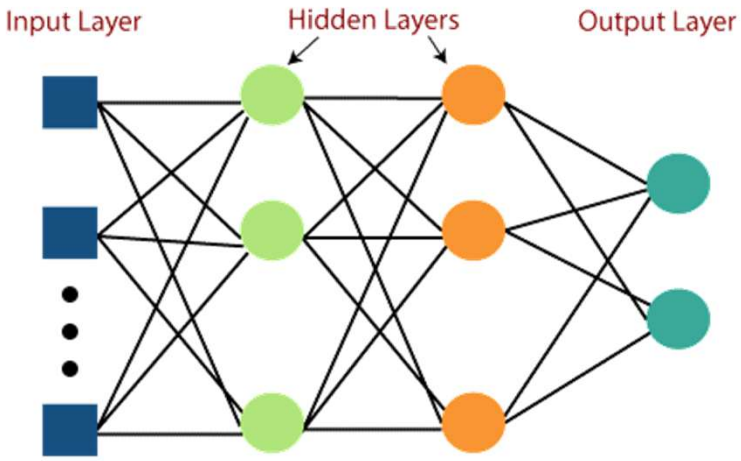
- The Feed Forward (FF) networks consist of multiple neurons and hidden layers which are connected to each other.
- These are called "feed-forward" because the data flow in the forward direction only, and there is no backward direction.
- Hidden layers might not be necessarily present in the network depending upon the application.



Types of Neural Networks (Contd….)

3. Multi-Layer Perceptron

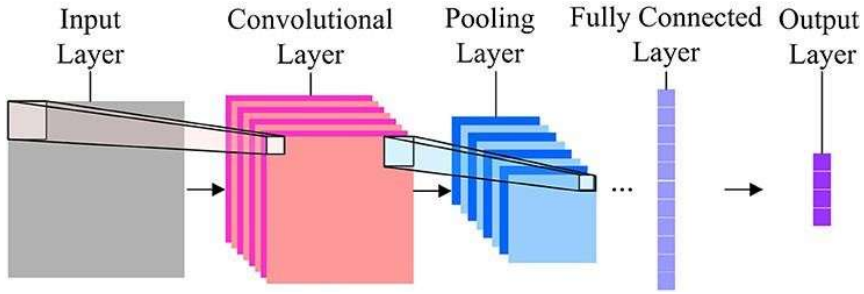
- Multi-layer Perceptrons are the neural networks which incorporate one or more hidden layers and activation functions. The multi-layer perceptron with one/two hidden layers is called **shallow neural network** and with more layers is called **deep neural network**.
- The learning takes place in a Supervised manner where the weights are updated by optimizers.



Types of Neural Networks (Contd….)

4. Convolutional Neural Networks

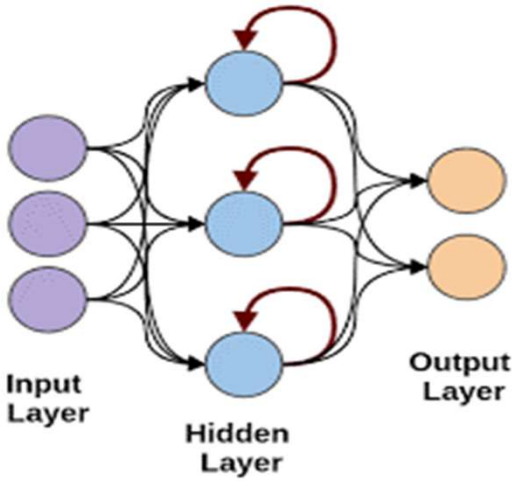
- When it comes to image classification, the most used neural networks are Convolution Neural Networks (CNN).
- CNN contain multiple convolution layers which are responsible for the extraction of important features from the image.
- The earlier layers are responsible for low-level details and the later layers are responsible for more high-level features.



Types of Neural Networks (Contd….)

5. **Recurrent Neural Networks**

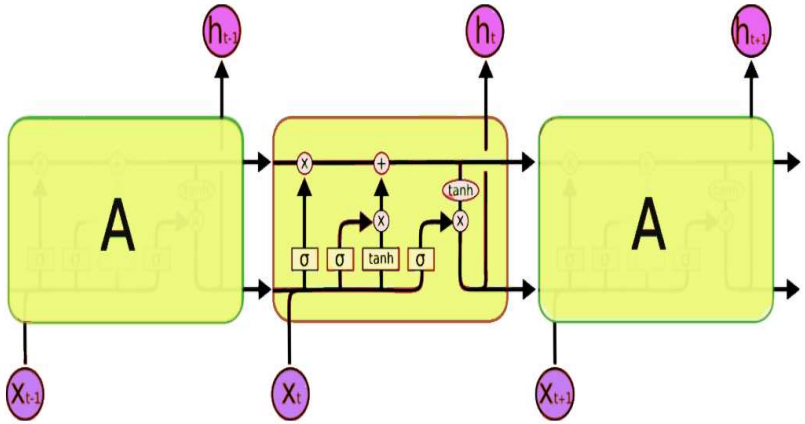
- Recurrent Neural Networks come into picture when there's a need for predictions using sequential data.
- Sequential data can be a sequence of audio, video words, etc.
- The RNN have a similar structure to that of a Feed-Forward Network, except that the layers also receive a time-delayed input of the previous instance prediction.
- This instance prediction is stored in the RNN cell which is a second input for every prediction.
- However, the main disadvantage of RNN is the Vanishing Gradient problem which makes it very difficult to remember earlier layers' weights.



Types of Neural Networks (Contd….)

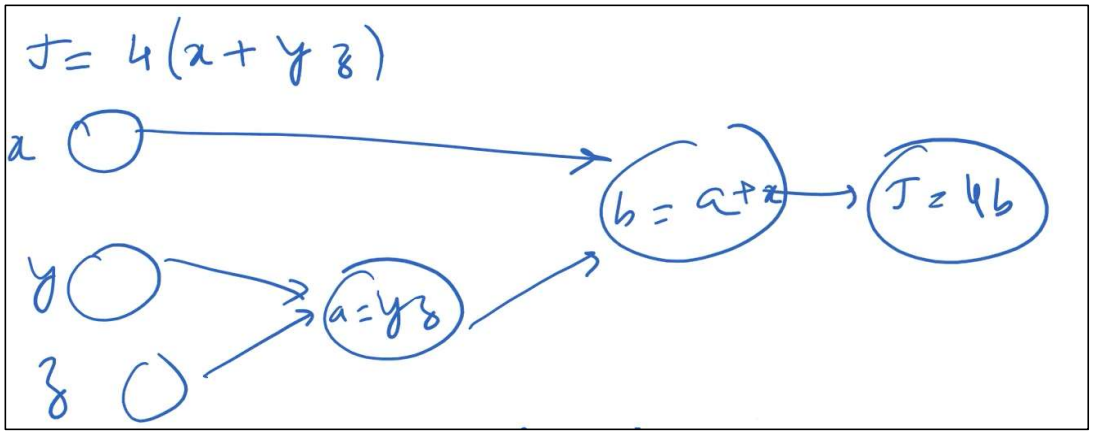
6. Long Short-Term Memory Networks

- LSTM neural networks overcome the issue of Vanishing Gradient in RNNs by adding a special memory cell that can store information for long periods of time.
- LSTM uses gates to define which output should be used or forgotten.
- It uses 3 gates: Input gate, Output gate and a Forget gate. The Input gate controls what all data should be kept in memory. The Output gate controls the data given to the next layer and the forget gate controls when to dump/forget the data not required.



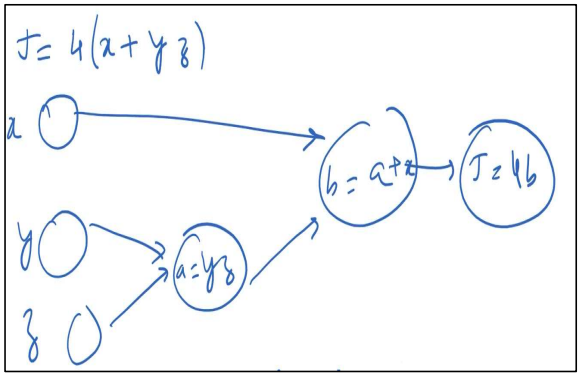
Computational Graphs

- A computational graph is a way to represent a math function in the language of graph theory.
- In a computational graph nodes are either input values or functions for combining values. Edges receive their weights as the data flows through the graph.
- For example, consider the relatively simple expression: J(x, y, z) = 4(x + yz).This is how we would represent that function as computational graph:



Computational Graphs (Contd….)

- One important of computational graphs is that, if we move froward in the graph (from left-to-right), it computes the value of mathematical expression. This is called **Forward Propagation**.
- But if we move in backward direction (from right to left), then we can compute the gradients of the mathematical expression w.r.t the input variables using chain rule. **This is called Backward propagation**.
- For example, consider the computational graph J(x, y, z) =4 (x + yz), in the figure, for x=2, y=3, z=4, we can compute value of J through forward propagation as follows:



a= 3*4=12

b= 12+2=14

J=4*14=56

Computational Graphs (Contd….)

- Similarly, the gradients of J can be computed in backward direction (from right to left).

- For example, $\frac{\partial J}{\partial b} = \frac{\partial (4b)}{\partial b} = 4$

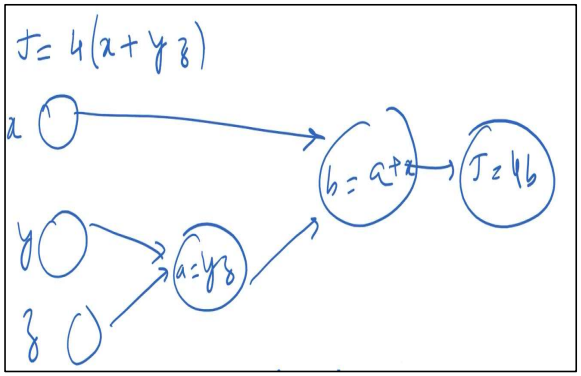
- It can be back propagated to compute $\frac{\partial J}{\partial a}$ and $\frac{\partial J}{\partial x}$ as follows:

$$\frac{\partial J}{\partial a} = \frac{\partial J}{\partial b}.\frac{\partial b}{\partial a} = 4X1 = 4 \; and \; \frac{\partial J}{\partial x} = \frac{\partial J}{\partial b}.\frac{\partial b}{\partial x} = 4X1 = 4$$

- Similarly, $\frac{\partial J}{\partial a}$ can be back propagated to compute $\frac{\partial J}{\partial y}$ and $\frac{\partial J}{\partial z}$ as follow:

$$\frac{\partial J}{\partial y} = \frac{\partial J}{\partial a}\frac{\partial a}{\partial y} = 4Xz = 4X4 = 16$$

$$\frac{\partial J}{\partial z} = \frac{\partial J}{\partial a}.\frac{\partial a}{\partial z} = 4Xy = 4X3 = 12$$



Logistic Regression as Neural Network

- Recall, Logistic Regression classifier, where the predicted value is given by:

$$\hat{y} = f(x) = \frac{1}{1 + e^{-z}}$$

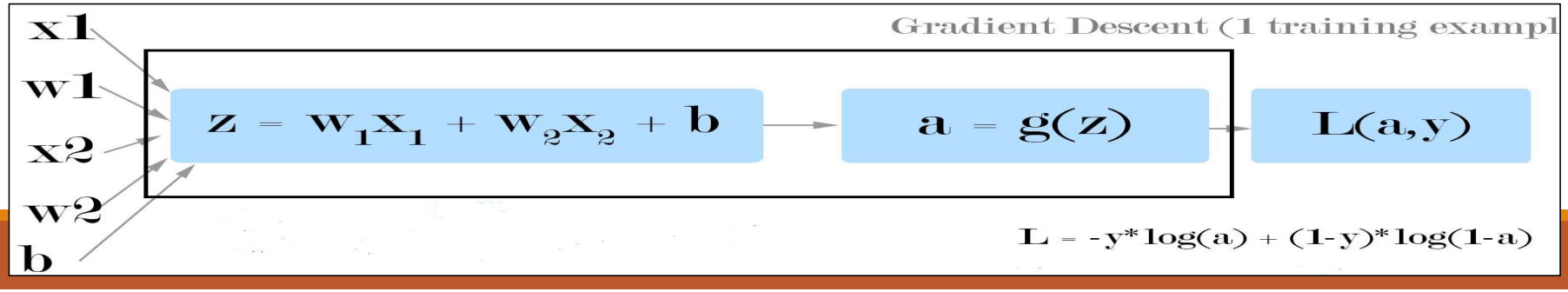
$$\text{and } z = b + w_1 x_1 + w_2 x_2 + \cdots \ldots \ldots \ldots \ldots + w_k x_k = W.X + b$$

- The parameters, W and b are optimized using gradient descent optimization which optimizes the cross entropy cost function given by:

$$J = -\frac{1}{n}\sum_{i=1}^{n} y_i log(f(x_i)) + (1 - y_i)\log(1 - f(x_i))$$

Logistic Regression as Neural Network (Contd….)

- The logistic regression classifier can be implemented as a single layer single neuron neural network classifier (as shown in fig. below , for one training example).
- For the sake of simplicity, two features x_1 and x_2 are considered. The two nodes marked in rectangle is a single neuron, which first computes z as weighted sum of inputs and then computes sigmoid of z to get predicted value (a)
- In order to fit a hypothesis function, we need to learn w_1, w_2, and b. These are learnt through gradient descent optimization.
- So, in each iteration, predicted value id computed through **forward propagation** and gradients of loss function w.r.t weights are computed through **backward propagation**.



Logistic Regression as Neural Network (Contd….)

- **Forward propagation (for a single training example)**
  - Calculate the weighted sum $z=w_1x_1+w_2x_2+b$.
  - Calculate the activation $a = \sigma(z) = \frac{1}{1+e^{-z}}$
  - Compute the loss $L(a, y)=-[y \log(a)+(1-y) \log(1-a)]$.

Logistic Regression as Neural Network (Contd....)

- **Backpropagation (for a single training example)**
  - Compute partial derivatives of L w.r.t w_1, w_2, and b , which are computed as follows:

$$\frac{\partial L}{\partial a} = \frac{\partial(-\text{y log(a)}+(1-\text{y}) \text{ log}(1-\text{a}))}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a} = \frac{\boldsymbol{a-y}}{\boldsymbol{a(1-a)}}$$

Now, $\frac{\partial L}{\partial a}$ is used to compute $\frac{\partial L}{\partial z}$ as follows:

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}.\frac{\partial a}{\partial z} = \frac{a-y}{a(1-a)}.\frac{e^{-z}}{(1+e^{-z})^2} = \frac{a-y}{a(1-a)}.\frac{1}{1+e^{-z}}.\left(1-\frac{1}{1+e^{-z}}\right) = \frac{a-y}{a(1-a)}.a(1-a) = \boldsymbol{a-y}$$

Now, $\frac{\partial L}{\partial z}$ is used to computed $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial w_2}$, $\frac{\partial L}{\partial b}$ as follows:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z}.\frac{\partial z}{\partial w_1} = (\boldsymbol{a-y})\boldsymbol{x_1}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial z}.\frac{\partial z}{\partial w_2} = (\boldsymbol{a-y})\boldsymbol{x_2}$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}.\frac{\partial z}{\partial b} = (\boldsymbol{a-y})$$

Logistic Regression as Neural Network (Contd….)

- For n training examples and k features, with the feature matrix X_{kXn} and the output matrix Y_{1Xn}, the gradients of the cost function J w.r.t weight matrix W_{kX1} and b_{1X1} is computed as :

$$\frac{\partial J}{\partial W} = \frac{1}{n}(A - Y).X^T$$

$$\frac{\partial J}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}(A - Y)$$

- where A_{1Xn} is the activation matrix (that contains predicted values of n examples).