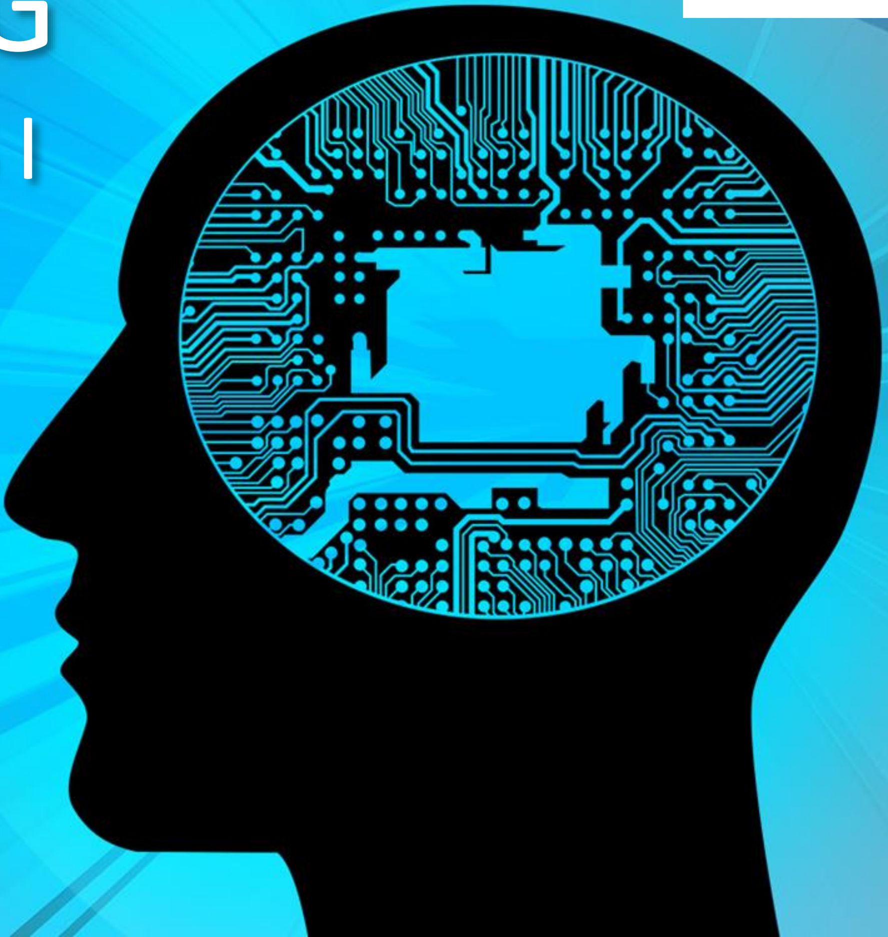
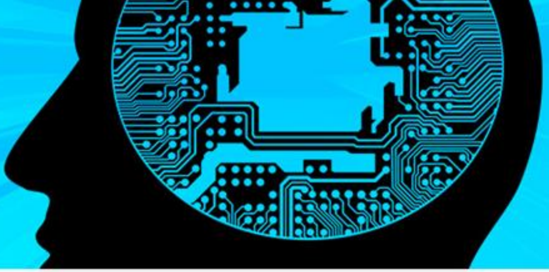
DEEP LEARNING & NEURAL NETWORKS I



Thakshila Dasun
BSc. Hons in Mechanical Engineering (Mechatronics)
CIMA, UK
Academy of Innovative Education

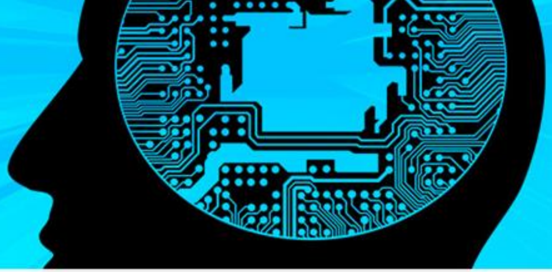


DEEP LEARNING

“In a neural network we don't tell the computer how to solve our problem. Instead, it learns from observational data, figuring out its own solution to the problem at hand.”

-In 2006 was the discovery of techniques for learning in so-called deep neural networks. These techniques are now known as deep learning. They've been developed further, and today deep neural networks and deep learning achieve outstanding performance on many important problems in computer vision, speech recognition, and natural language processing-

They use Deep Learning



airbnb Coca-Cola DeepMind GE Healthcare Google intel NERSC

AIRBUS DEFENCE & SPACE AMD arm BAKER HUGHES a GE company BITMAIN Bloomberg caicloud 才云 CAMBIA HEALTH SOLUTIONS

Castbox CEVA cheetahmobile CI&T Dropbox ebay 流利说 IBM Power Systems

JD.COM kakao kika Lenovo LinkedIn Matroid MEDIATEK 美团 meituan.com

mi Mobvoi Movidius an Intel company musical.ly 網易 NETEASE NVIDIA PayPal Qualcomm

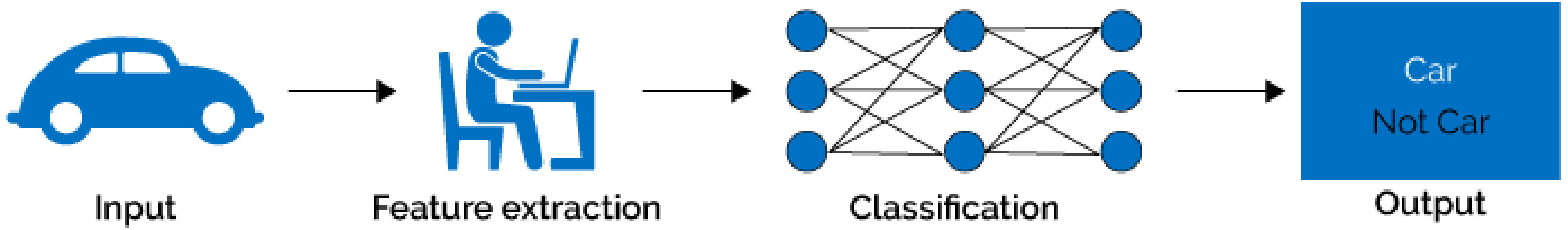
quantiphi SAP SINOVATION VENTURES Sogou 搜狗 腾讯社交广告 Tencent Social Ads Uber 云知声 Unisound

360 TOTAL SECURITY 3DR

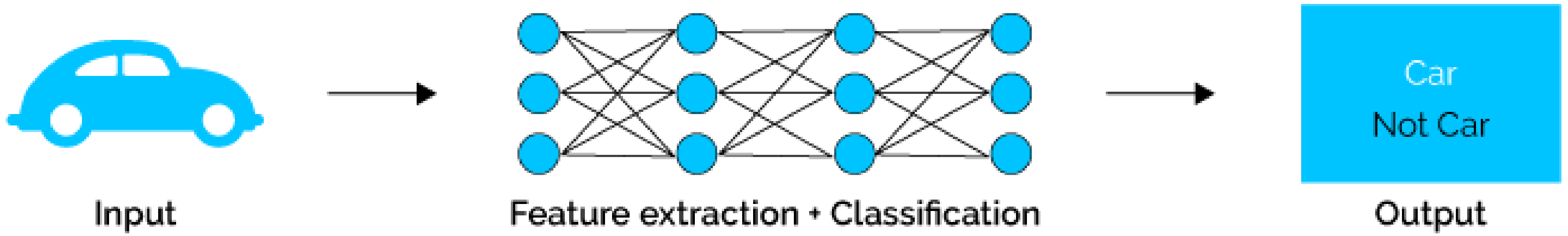
Applications

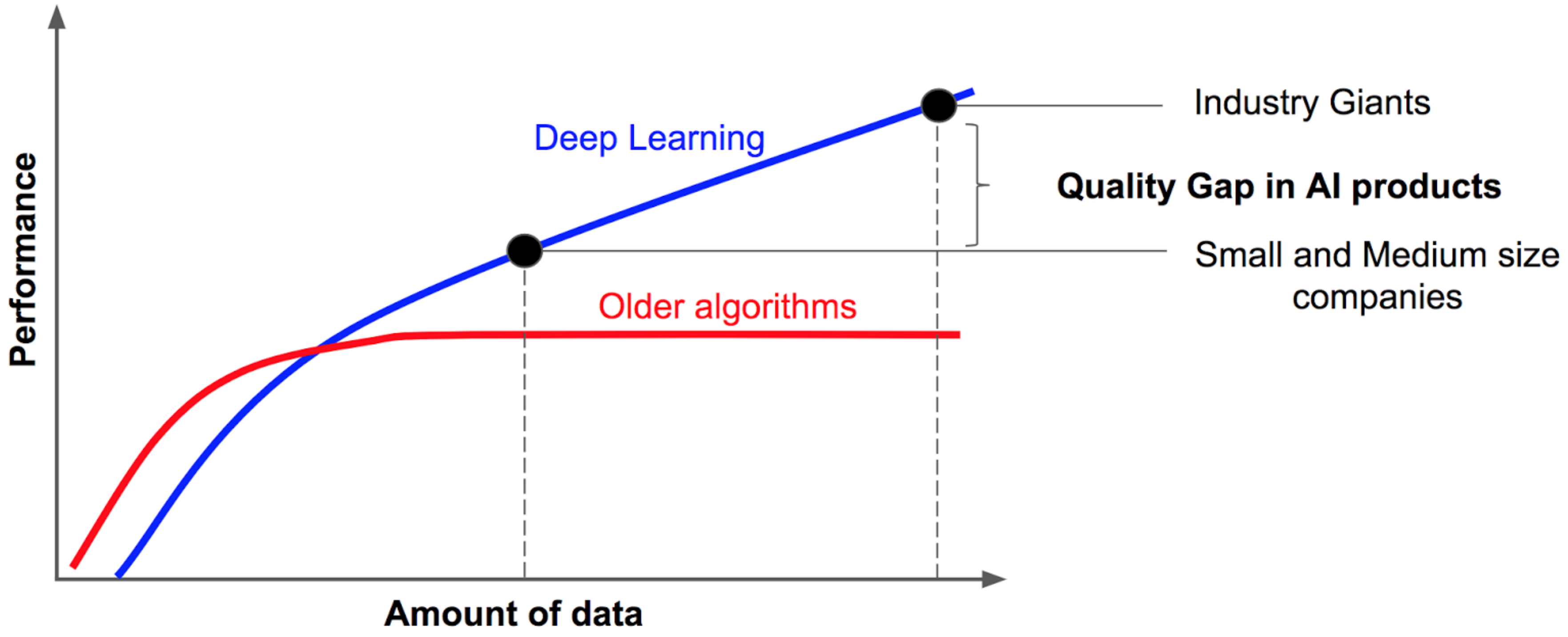
- Face Recognition
- Image Classification
- Speech Recognition
- Text-to-speech Generation
- Handwriting Transcription
- Machine Translation
- Medical Diagnosis
- Self Driving Cars
- Digital Assistants
- Ads, search, social recommendations
- Games with Deep Reinforcement Learning

Machine Learning



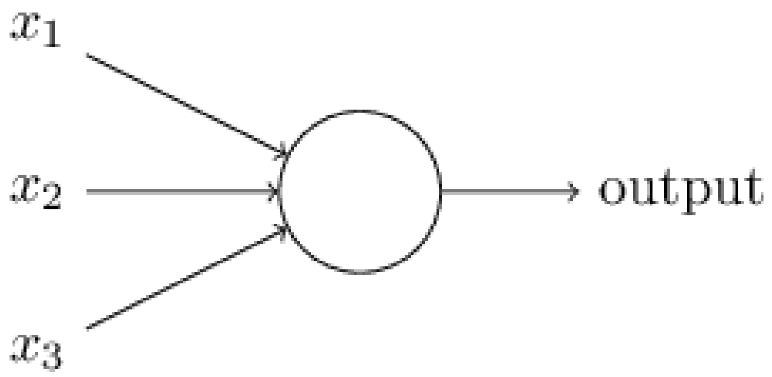
Deep Learning




Industry Giants
Deep Learning
Quality Gap in AI products
Small and Medium size companies
Older algorithms
Performance
Amount of data

History In Brief (1)

- The idea of neural networks began unsurprisingly as a model of how neurons in the brain function.
  - 1943: Portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitt
  - 1950-1960: Perceptrons were developed by the scientist Frank Rosenblatt,



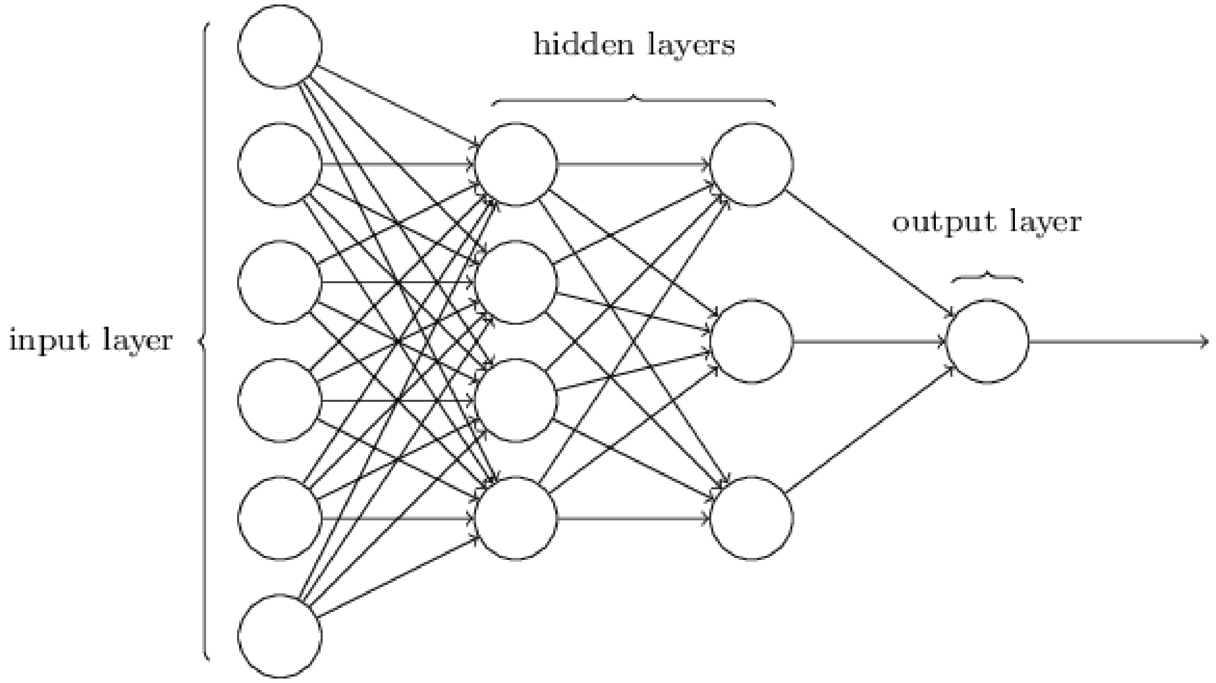
History In Brief (2)

- 1974-86: Backpropagation Algorithm, Recurrent NL
- 1989-98: Convolutional Neural Networks, Bi Directional RNN, Long Short Term Memory (LSTM), MNIST Data Set
- 2006: “Deep Learning” Concept
- 2009: ImageNet
- 2012: AlexNet, Dropout
- 2014: DeepFace
- 2016: AlphaGo
- 2017: AlphaGo Zero
- 2018: BERT

History In Brief (3)

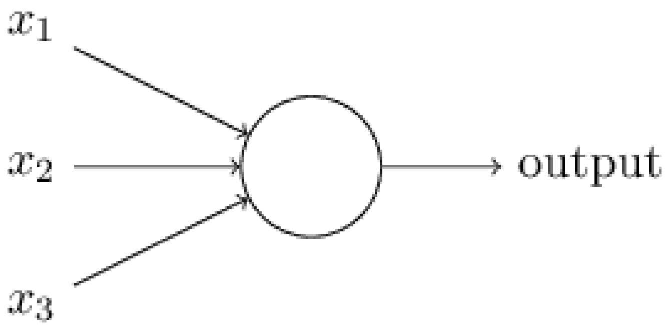
- Mark 1 Perceptron: 1960
- Torch: 2002
- CUDA: 2007
- Theano: 2008
- Caffe: 2015
- Tensorflow 0.1: 2015
- PyTorch 0.1: 2017
- TensorFlow 1.0: 2017
- Tensorflow 2.0: 2019

Neural Network Architecture (1)



Perceptron

- Building Model of Classic Artificial Neural Networks
- Inputs, x1,x2,x3
- weights, w1,w2,…, real numbers expressing the importance of the respective inputs to the output
- The neuron's output, 0 or 1, is determined by whether the weighted sum **∑jwjxj** is less than or greater than some **threshold value**

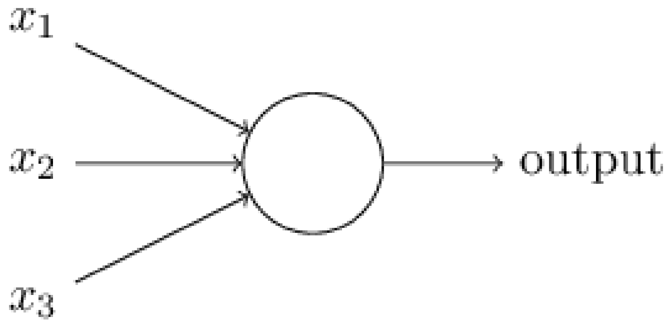


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

"Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts"

Sigmoid Neurons

- Sigmoid neuron has inputs, x1,x2,…. But instead of being just 0 or 1, these inputs can also take on any values between 0 and 1

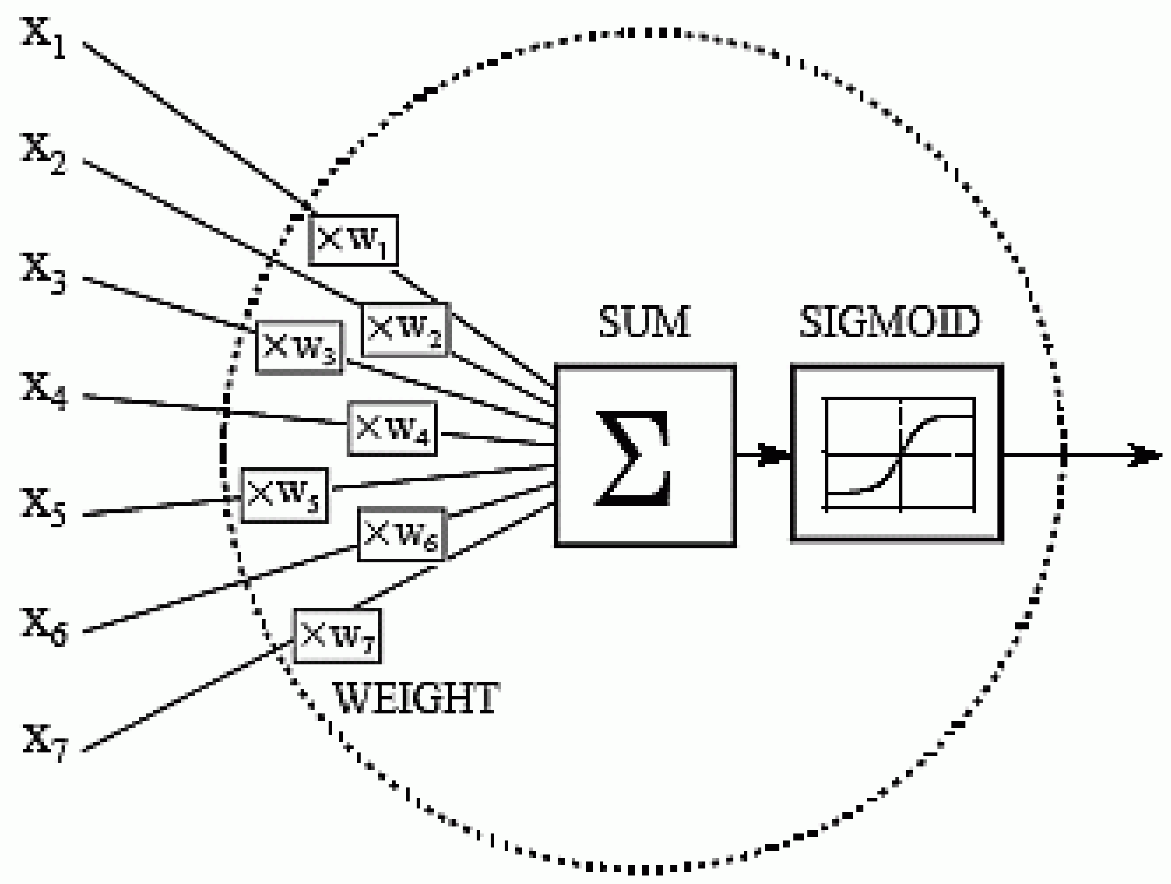


$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}.$$

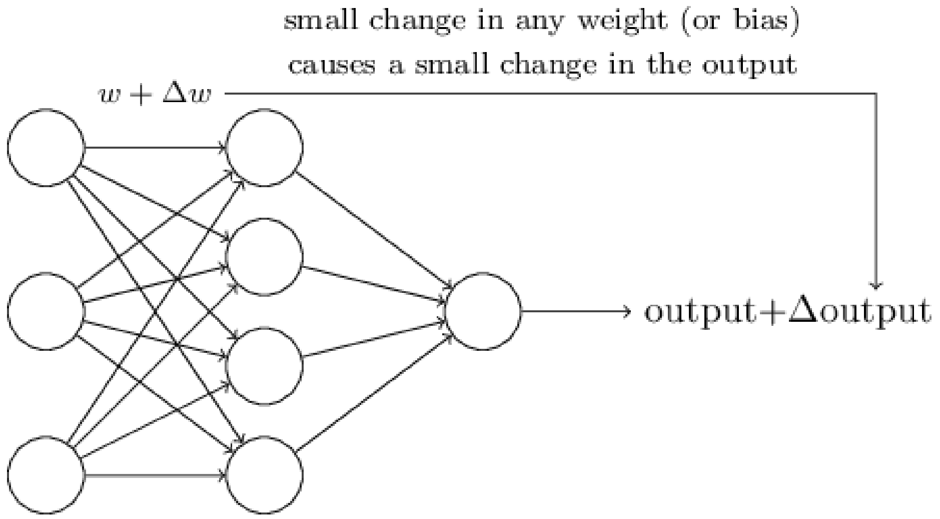
“One big difference between perceptrons and sigmoid neurons is that sigmoid neurons don't just output 0 or 1. They can have as output any real number between 0 and 1, so values such as 0.173… and 0.689… are legitimate outputs”

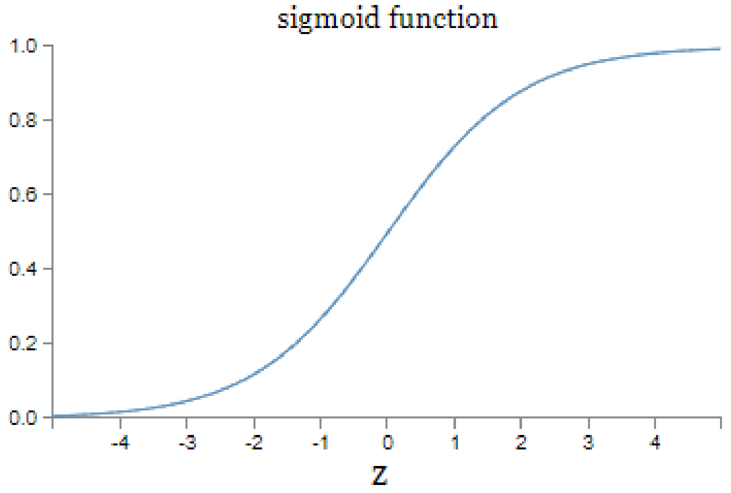
Sigmoid Neurons



Why Sigmoid Neurons (1)

- In a neuron small change in weight should cause only a small corresponding change in the output from the network


small change in any weight (or bias)
causes a small change in the output
$w + \Delta w$
output+Δoutput



Why Sigmoid Neurons (2)

- The smoothness of σ means that small changes Δwj in the weights and Δb in the bias will produce a small change Δoutput in the output from the neuron. In fact, calculus tells us that Δoutput is well approximated by

$$\Delta \text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b,$$

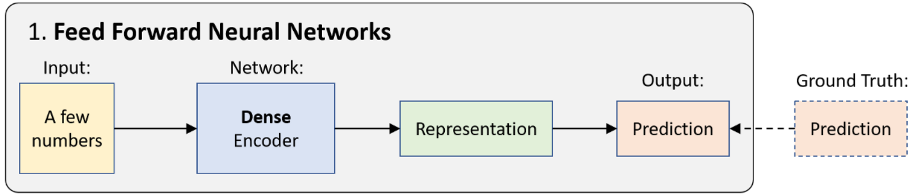
Neural Network Architecture (2)

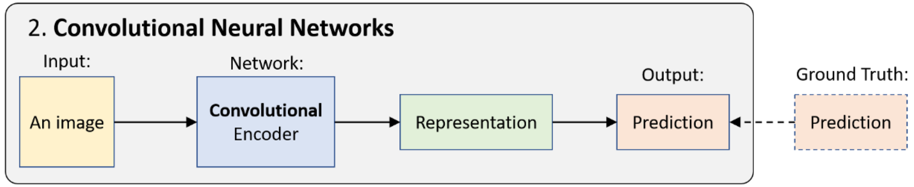
“At a high-level, neural networks are either encoders, decoders, or a combination of both”

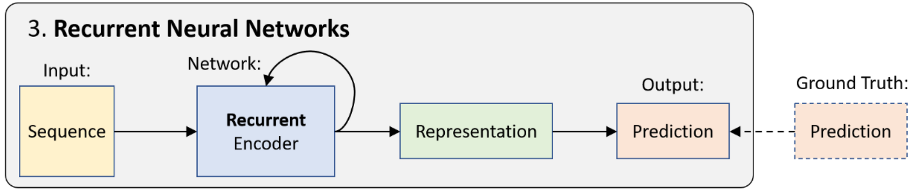
- **Encoders** find patterns in raw data to form compact, useful representations.
- **Decoders** generate high-resolution data from those representations. The generated data is either new examples or descriptive knowledge.

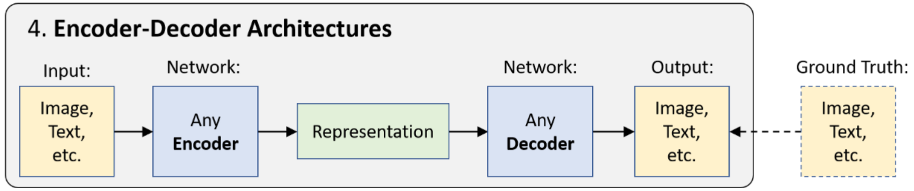
Neural Network Architecture (3)

Supervised Learning

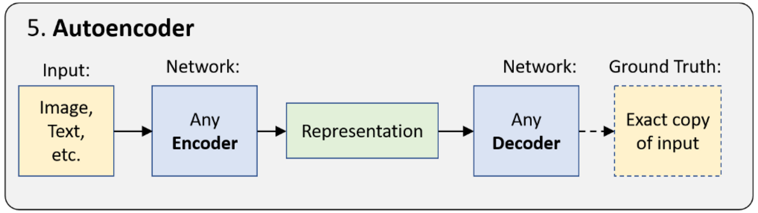


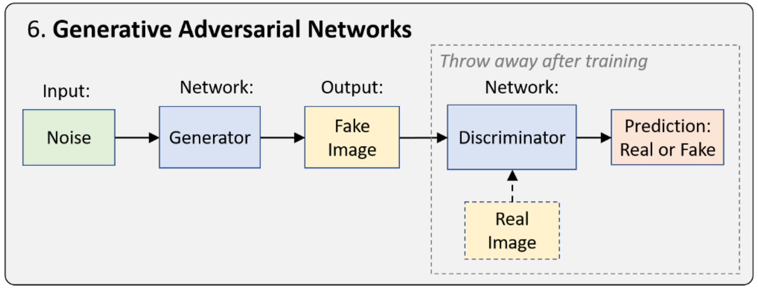




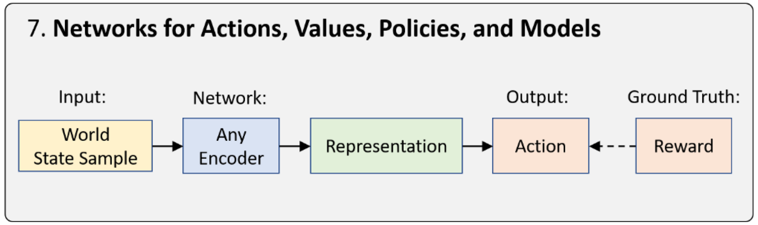


Unsupervised Learning



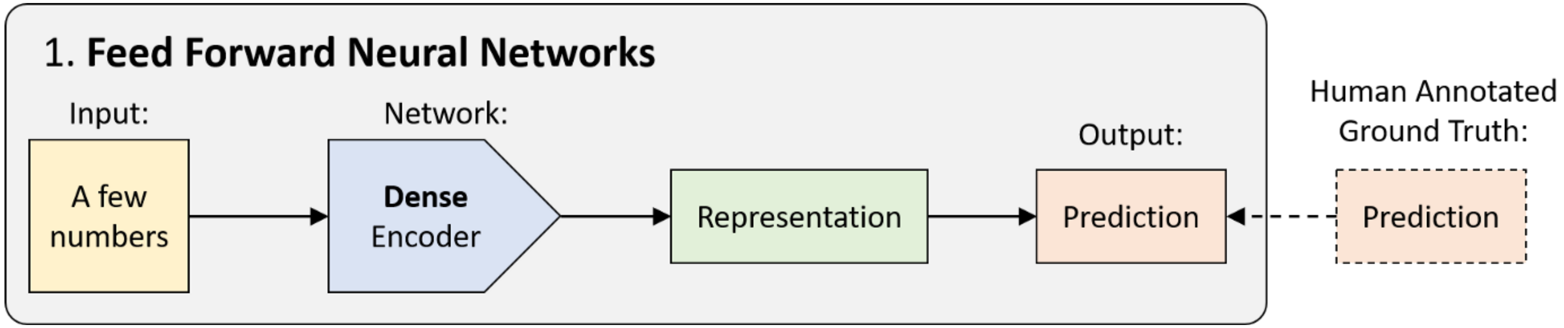


Reinforcement Learning



Feed Forward Neural Networks (FFNNs)

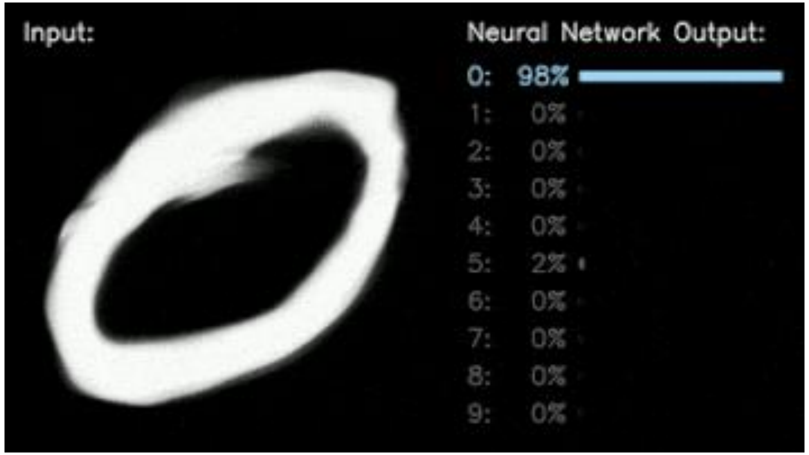
- Data passes from input to output in a single pass without any “state memory” of what came before.
- “FFNN” refers to its simplest variant: a densely-connected multilayer perceptron (MLP).
- Dense encoders are used to map an already compact set of numbers on the input to a prediction: either a classification (discrete) or a regression (continuous).

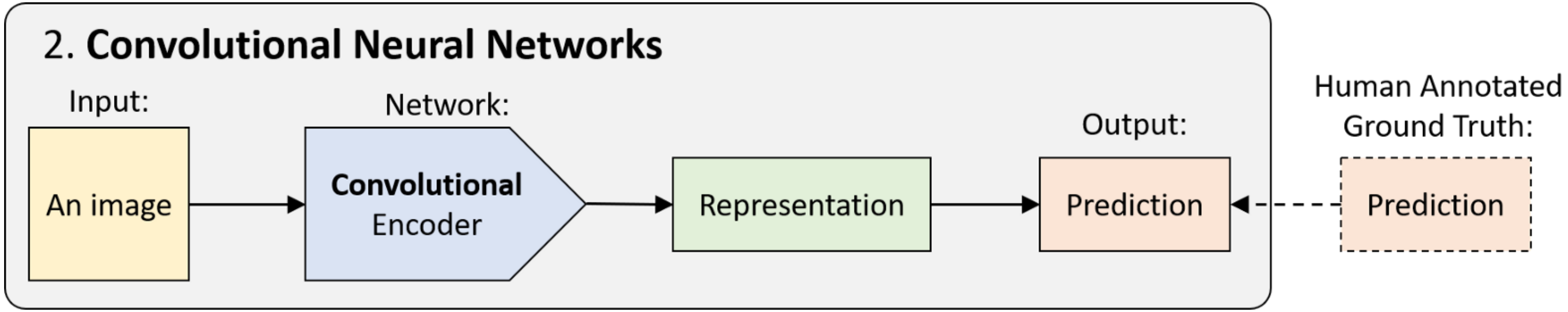


Convolutional Neural Networks (CNNs) (1)

- CNNs (aka ConvNets) are feed forward neural networks that use a spatial-invariance trick to efficiently learn local patterns, most commonly, in images.
- Spatial-invariance means that a cat ear in the top left of the image has the same features as a cat ear in bottom right of the image.
- CNNs share weights across space to make the detection of cat ears and other patterns more efficient.
- Instead of using only densely-connected layers, they use convolutional layers (convolutional encoder).
- These networks are used for image classification, object detection, video action recognition, and any data that has some spatial invariance in its structure (e.g., speech audio).

Convolutional Neural Networks (CNNs) (2)




2. **Convolutional Neural Networks**
Input:
An image
Network:
Convolutional Encoder
Representation
Output:
Prediction
Human Annotated Ground Truth:
Prediction

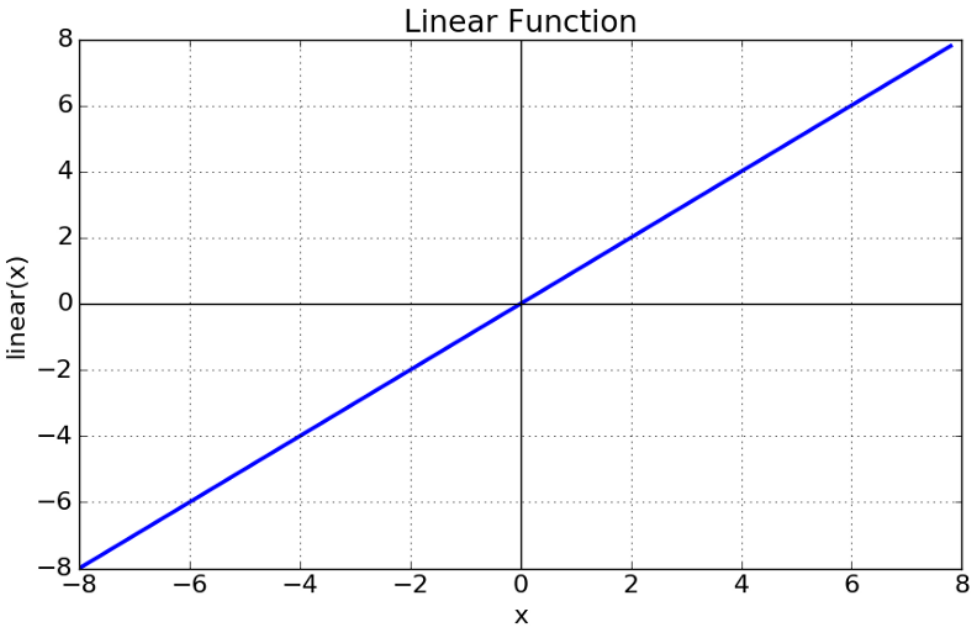
Types of activation functions

- The Activation Functions can be basically divided into 2 types
  - Linear or Identity Activation Function
  - Non-linear Activation Functions

Linear or Identity Activation Function

- The activation is proportional to the input. . This can be applied to various neurons and multiple neurons can be activated at the same time

- **Equation :** f(x) = x
- **Range :** (-infinity to infinity)



Non-linear Activation Functions

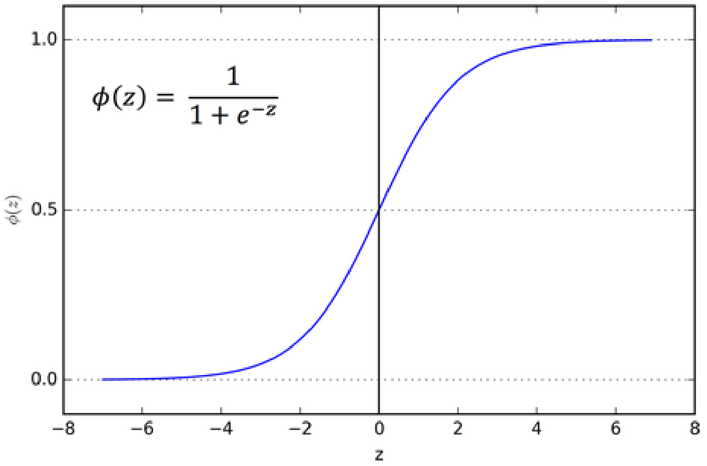
- The Nonlinear Activation Functions are the most used activation functions. It makes it easy for the model to generalize or adapt with variety of data and to differentiate between the output.

1. **Sigmoid or Logistic Activation Function**
2. **Tanh or hyperbolic tangent Activation Function:**
3. **ReLU (Rectified Linear Unit) Activation Function**
4. **Leaky ReLU**
5. **Softmax**

Sigmoid or Logistic Activation Function

- **Equation :** f(x) = 1 / 1 + exp(-x)
- **Range :** (0 to 1)

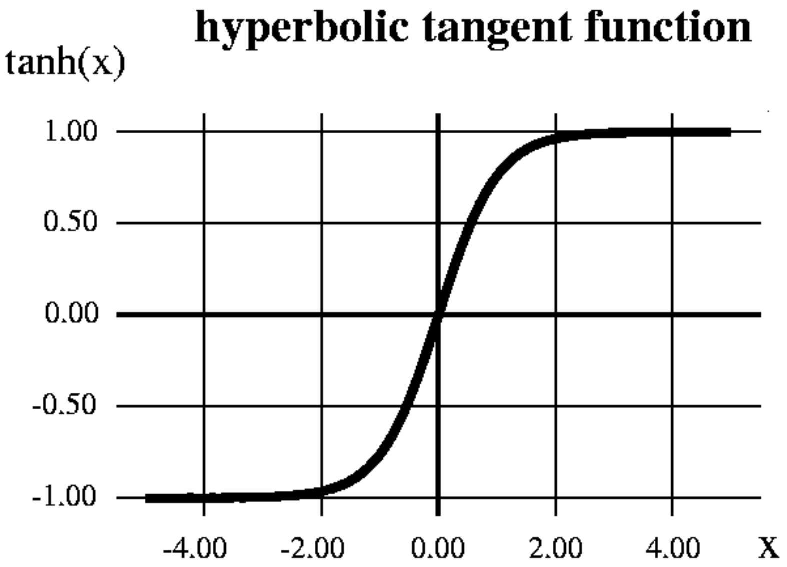
- It gives rise to a problem of "**vanishing gradients**", since the Y values tend to respond very less to changes in X
- It saturate and kill gradients.



Tanh or hyperbolic tangent Activation Function

- **Equation : f(x) = 1 — exp(-2x) / 1 + exp(-2x) or 2 *sigmoid(2x)-1**
- **Range :** (-1 to 1)

- It also suffers vanishing gradient problem
- It saturate and kill gradients.

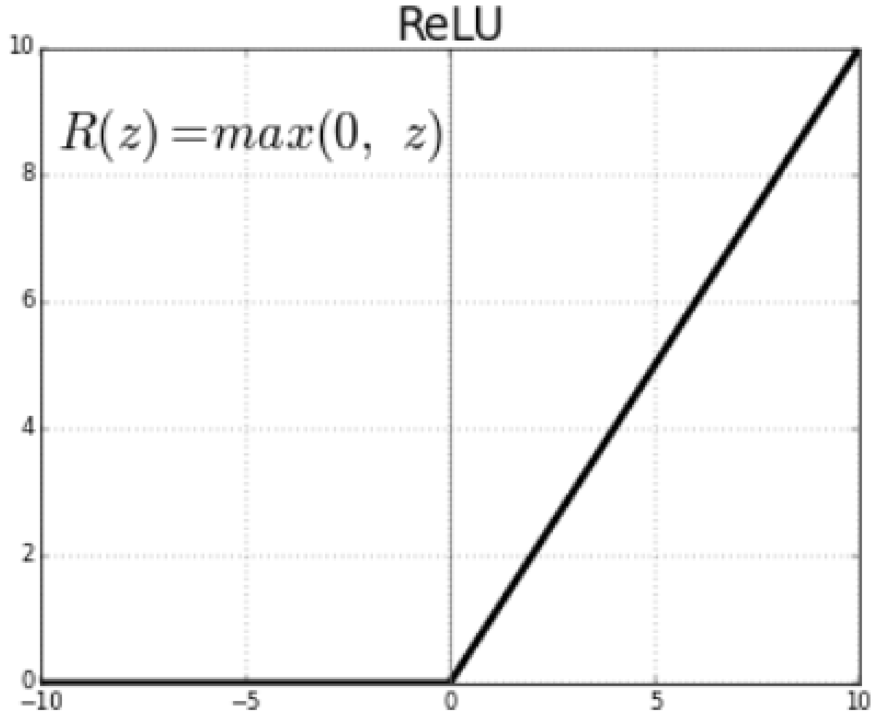


ReLU (Rectified Linear Unit)

- The ReLU is the most used activation function in the world right now

- **Equation : f(x) = max(0,x)**
- **Range :** (0 to infinity)

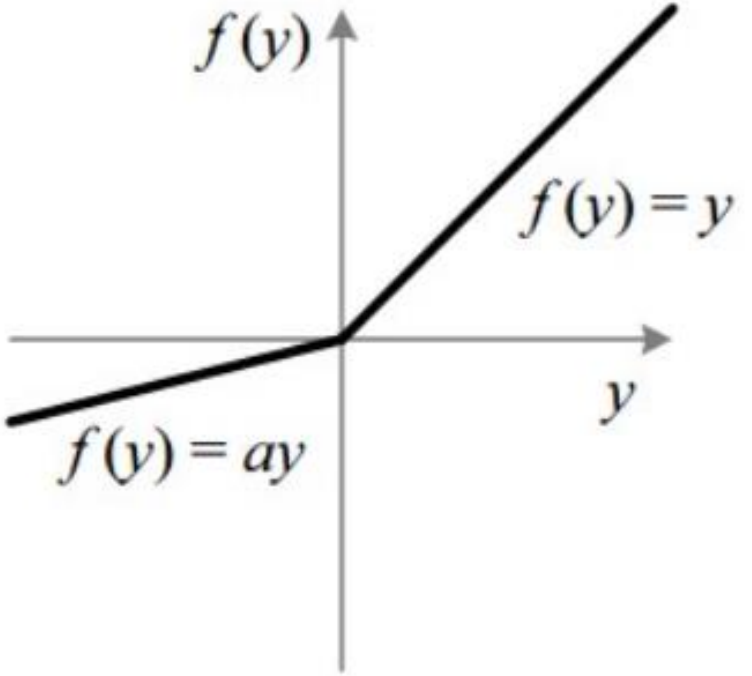
- The outputs are not zero centered similar to the sigmoid activation function
- When the gradient hits zero for the negative values, it does not converge towards the minima which will result in a dead neuron while back propagation



Leaky ReLU

- To solve the ReLU problem we have leaky ReLU

- **Equation :** f(x) = ax for x<0 and x for x>0
- **Range :** (0.01 to infinity)



Softmax

- The softmax function is also a type of sigmoid function but it is very useful to handle classification problems having multiple classes.
- The softmax function is shown above, where z is a vector of the inputs to the output layer
- The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Loss Functions

Mean Squared Error:

Mean Squared Error (MSE), or quadratic, loss function is widely used in linear regressionas the performance measure, and the method of minimizing MSE is called Ordinary Least Squares (OSL),

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

Loss Functions

Cross Entropy:

Cross Entropy is commonly-used in **binary classification** (labels are assumed to take values 0 or 1) as a loss function (For multi-classification, use **Multi-class Cross Entropy**), which is computed by

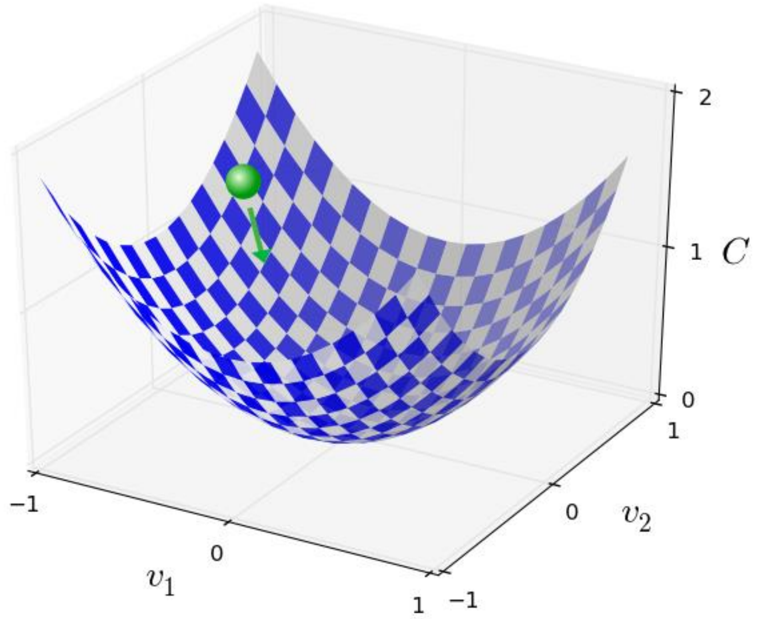
$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

Types of Optimizers

- Optimization algorithms helps us to minimize (or maximize) an Objective function (another name for Error function)
- E(x) is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model.
- For example we call the Weights(W) and the Bias(b)

Gradient Descent

- ***Gradient Descent*** is the most important technique and the foundation of how we train and o
- *It Find the Minima , control the variance and then update the Model's parameters and finally lead us to Convergence.*



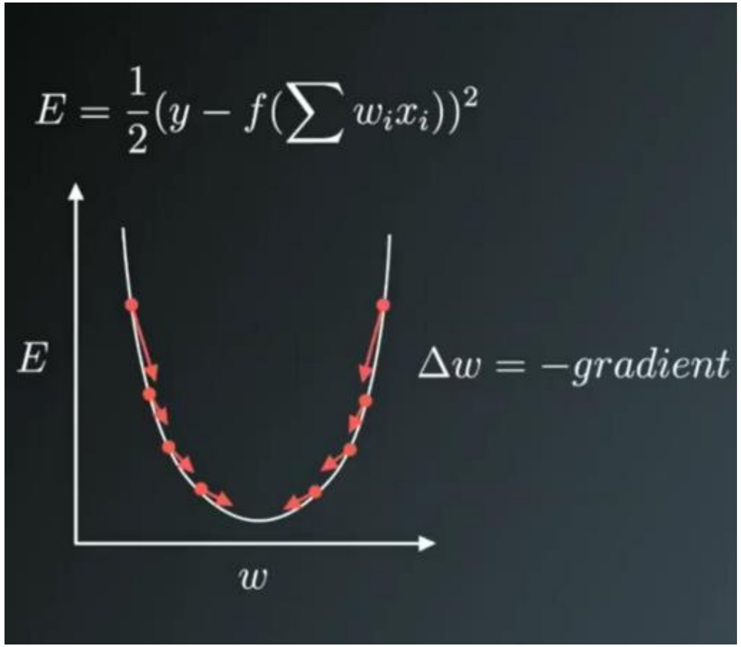
$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2.$$

Gradient Descent

- Gradient descent is majorly used to do **Weights updates** in a Neural Network Model , i.e update and tune the Model's parameters in a direction so that we can minimize the **Loss function**.
- Now we all know a Neural Network trains via a famous technique called **Backpropagation ,** in which we first propagate forward calculating the dot product of Inputs signals and their corresponding Weights and then apply a ***activation function*** to those sum of products

Gradient Descent

- Gradient descent is majorly used to do **Weights updates** in a Neural Network Model , i.e update and tune the Model's parameters in a direction so that we can minimize the **Loss function**.
- Now we all know a Neural Network trains via a famous technique called **Backpropagation ,** in which we first propagate forward calculating the dot product of Inputs signals and their corresponding Weights and then apply a ***activation function*** to those sum of products



Adam

- Adam stands for **Adaptive Moment Estimation.** Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter.

AdaDelta

- It is an extension of **AdaGrad** which tends to remove the *decaying learning Rate* problem of it. Instead of accumulating all previous squared gradients, ***Adadelta*** limits the window of accumulated past gradients to some fixed size **w**.

Adagrad

- It simply allows the learning Rate -**η** to **adapt** based on the parameters. So it makes big updates for infrequent parameters and small updates for frequent parameters. For this reason, it is well-suited for dealing with sparse data.

"TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications."

"Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research"*

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.