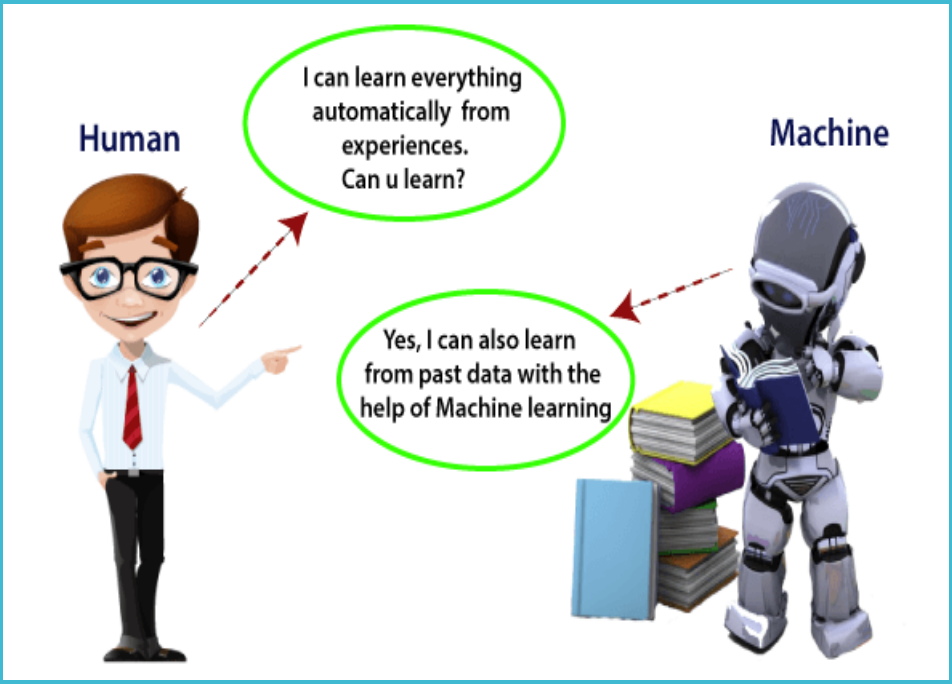
Machine Learning and Neural Network



Marwadi University

Department of Information and Communication Technology

Unit 7:
Deep Learning

Artificial Intelligence (01CT0703)

Prof. Nishith Kotak

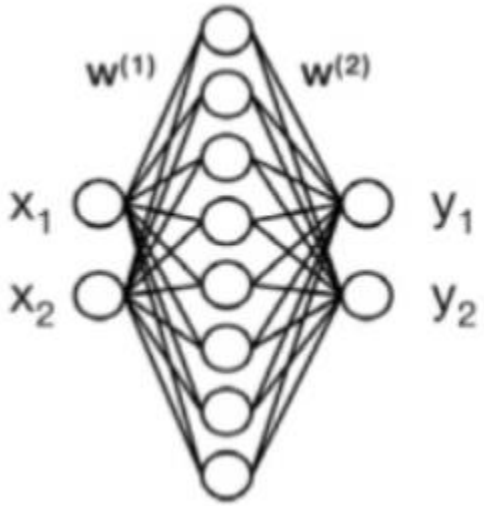
Where ML is Used?

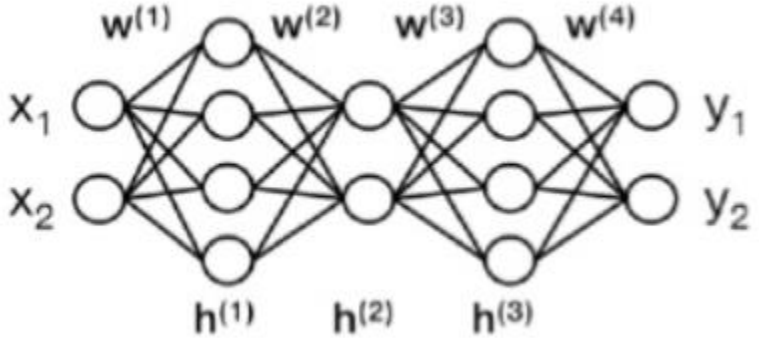
- ML is used when:
  - ✓ Human expertise does not exist
  - ✓ Humans can't explain their expertise
  - ✓ Models must be customized
  - ✓ Models are based on huge amount of data

What is NN?

- Requirement of NN
- Limitation of basic ML methods
- Advantages of NN

NN Vs DNN





- Generally more than 7 Layers (Not set as a rule)

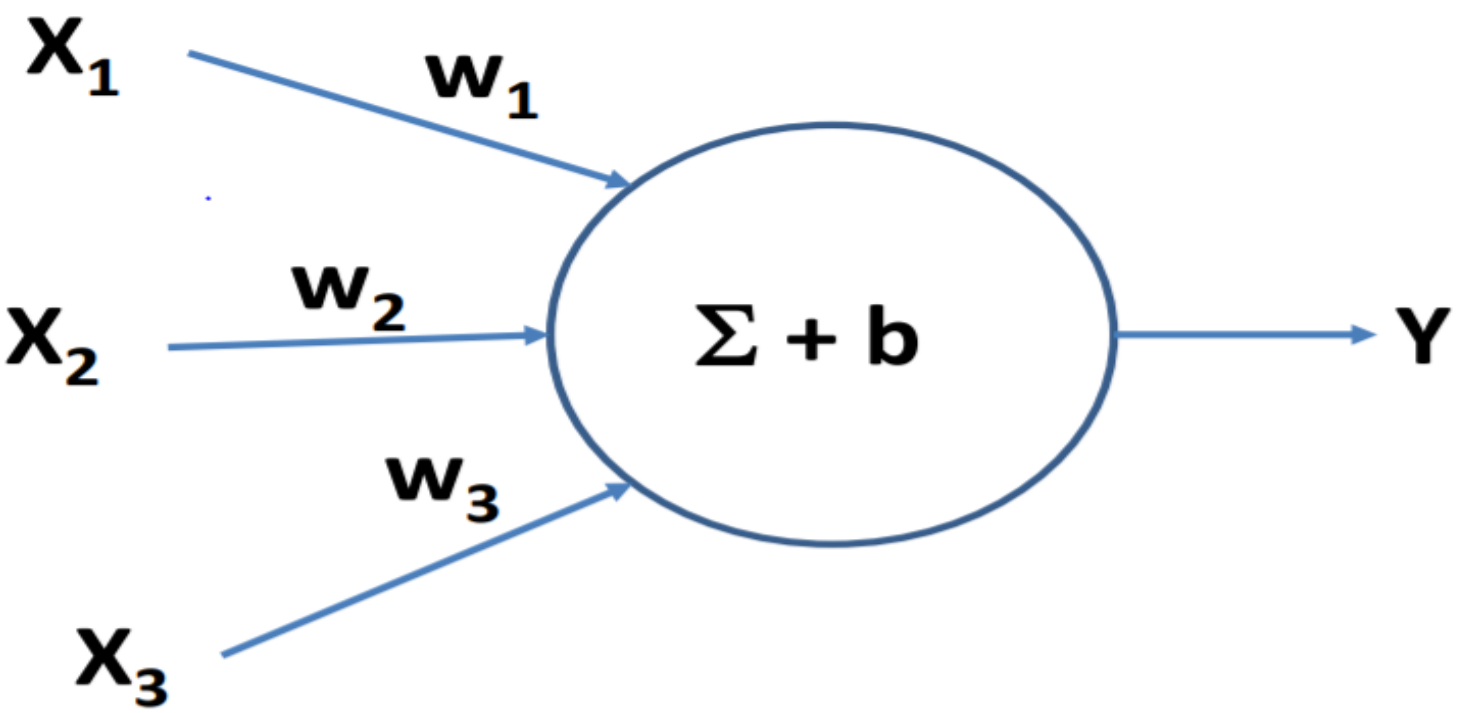
Why Deep Learning is essential?

- Deep learning models have revolutionized many areas of machine intelligence, e.g.,
  - Natural Language Processing (NLP)
  - Handwriting recognition
  - Image recognition
  - Speech recognition, etc. …
- Deep learning models provide a ***generalized approach*** to complex models
- Deep learning models ***learn features***
- They are ***less prone to errors*** compared to the traditional learning based systems

How Deep Learning Evolved?

- Neural network models have been around for a long time
- Very little research work done till 1970s
- Training deep NNs with backpropagation algorithm (Rummelhart et al., 1986) sparked new interest
- Rania et al. (2009), demonstrated that GPUs could be effectively applied to training deep NNs
- The advent of GPUs has removed the limiting computing capacity of the CPUs
- Deep learning solutions are now routinely deployed on a large scale on platforms like Azure, etc.

Linear Neural Network

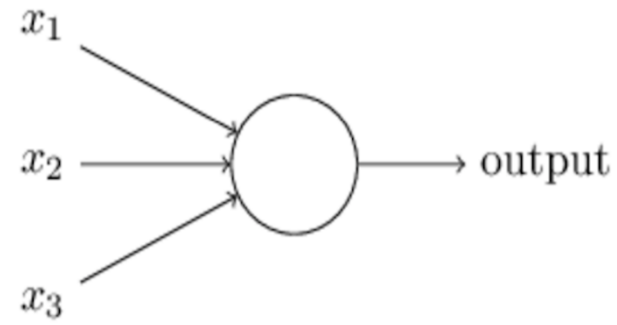


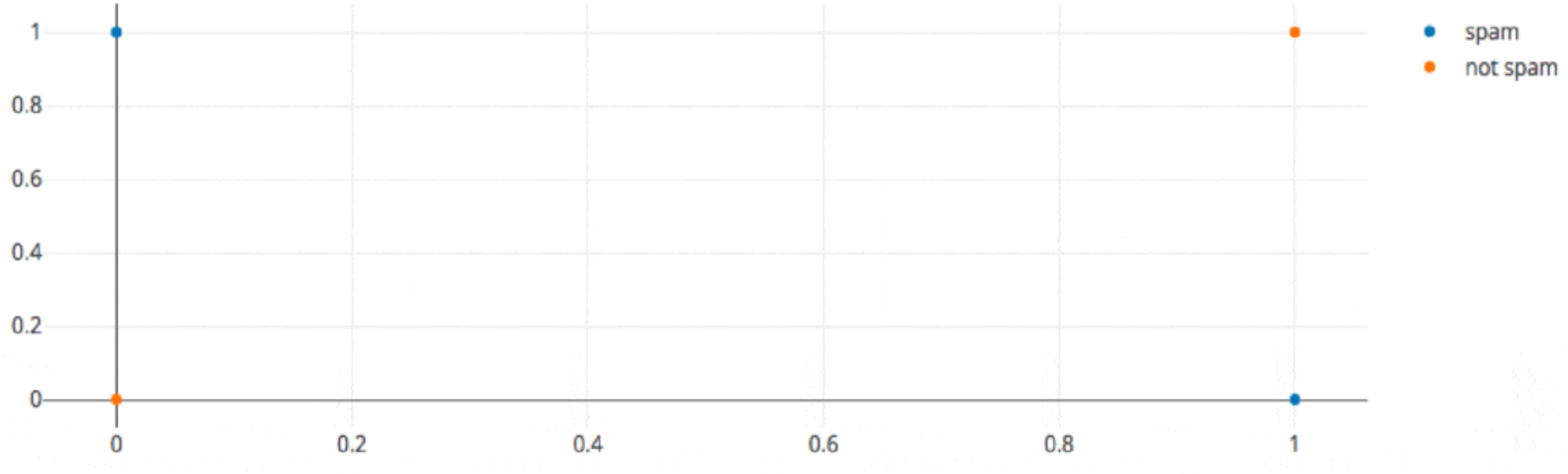
$$y = f(x) = \sum_i w_i \cdot x_i + b$$

https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Linear Neural Network

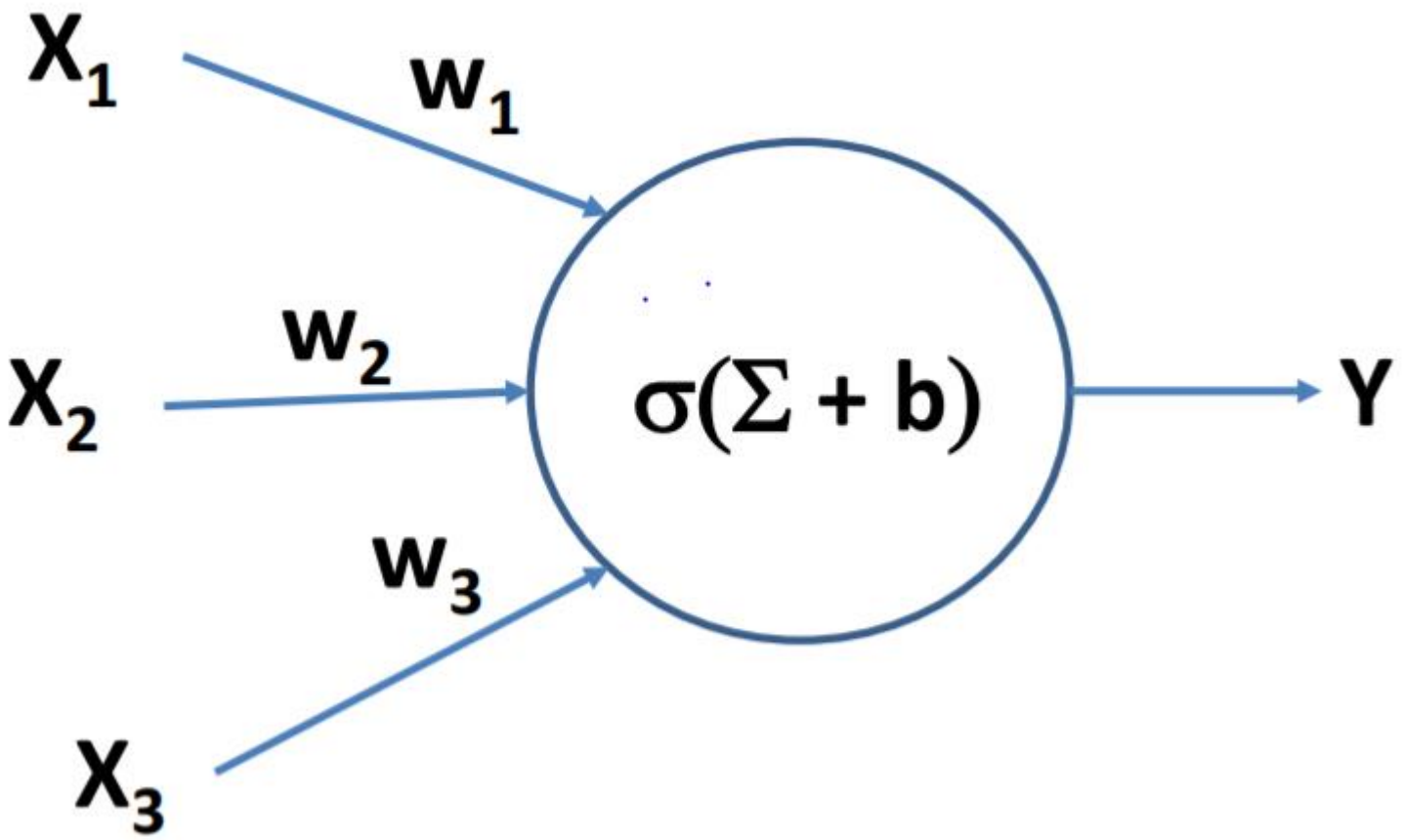
Single layer Perceptron (linear model)





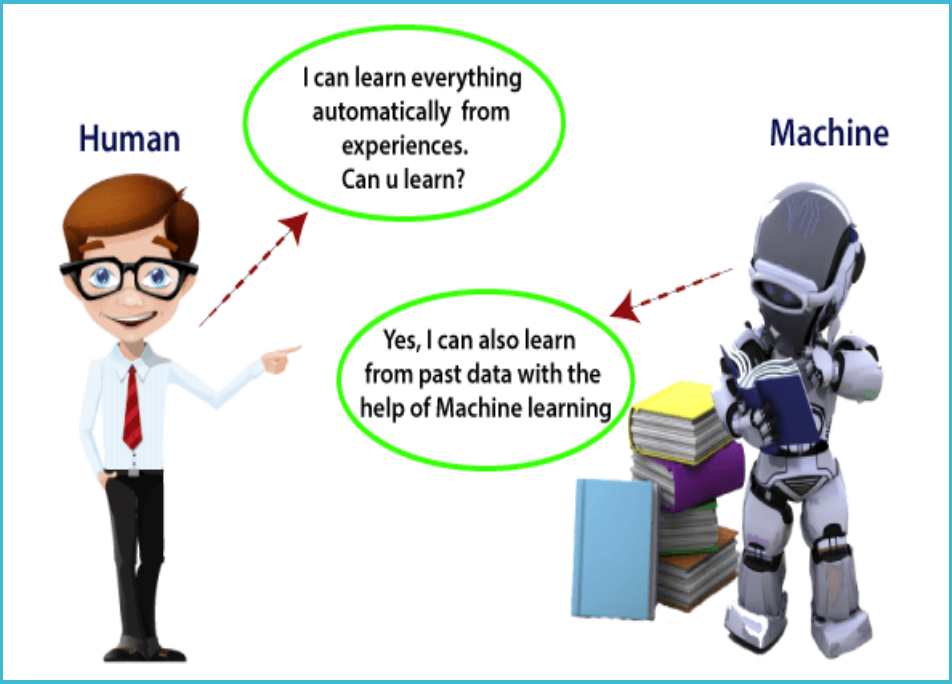
https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Linear NN with Non Linear Activation



https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Activation Functions



Marwadi University

Department of Information and Communication Technology

Unit 7:
Deep Learning

Artificial Intelligence (01CT0703)

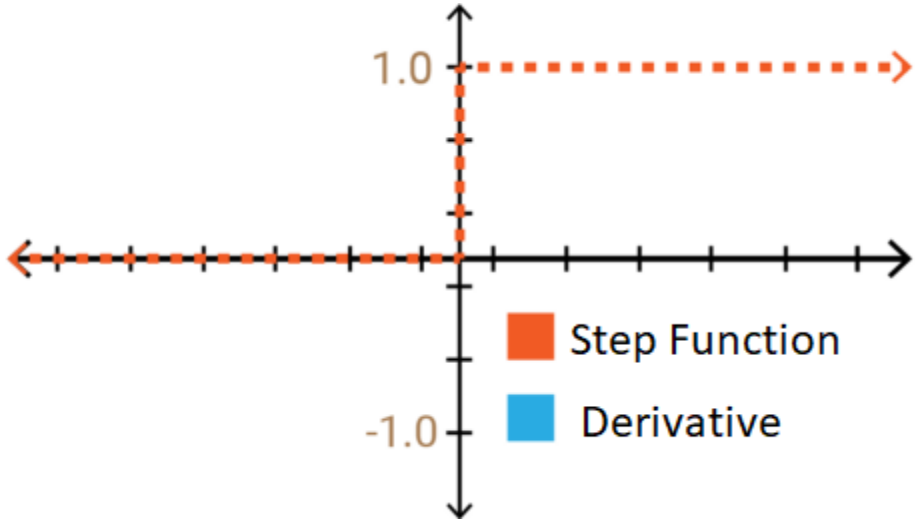
Prof. Nishith Kotak

Desired Features of Activation Function

- Vanishing Gradient Problem
- Zero-Centred
- Computational Expense
- Differentiable

https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

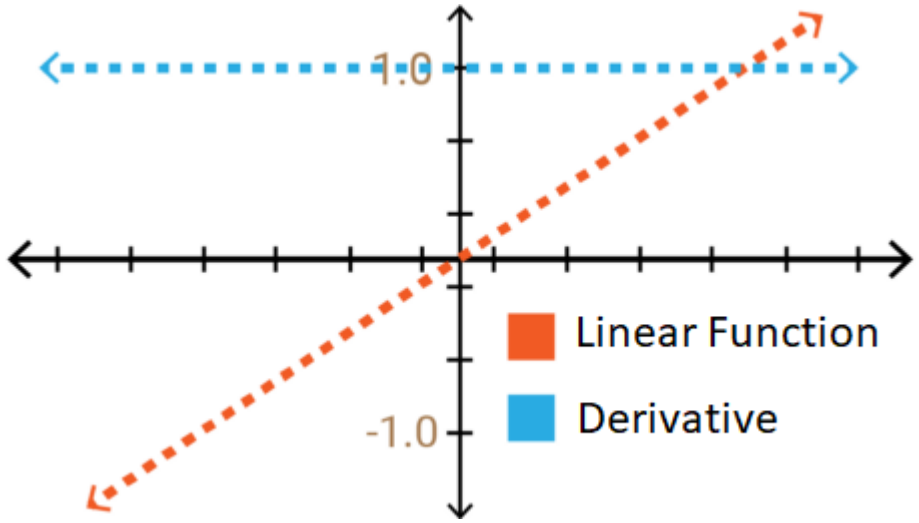
Activation Function: Step Function



- It is a function that takes a binary value and is used as a binary classifier. Therefore, it is generally preferred in the output layers.
- Differentiation leads to zero, which doesn't affect the change in the loss function and doesn't help getting optimized value

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

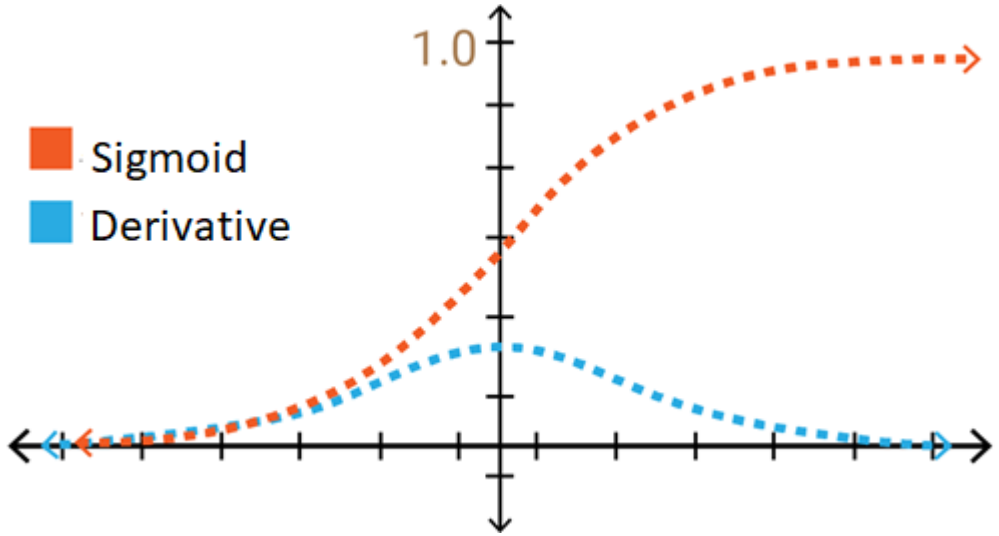
Activation Function: Linear Function



- It generates a series of activation values and these are not binary values, as in the step function. It certainly allows to connect several neurons (nerve cells) together.
- There is no limit to the values since it goes from (-infinite,+infinite)
- During backpropagation, for learning process, the derivative is constant, hence doesn't change the rate of reaching the optimized value.
- Using linear function in all the layers gives the same result at output layer as applying the linear function on input. Since Linear combination of linear function is also linear function.

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

Activation Function: Sigmoid Function

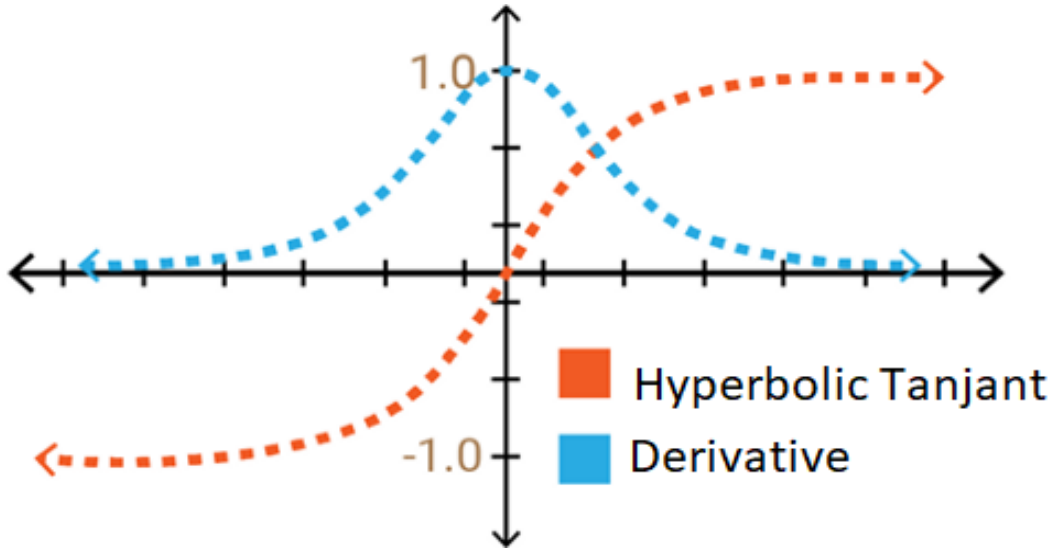


$$\phi(z) = \frac{1}{1 + e^{-z}}$$

- If we examine the graph x is between -2 and +2, y values change quickly. Small changes in x will be large in y. This means that it can be used as a good classifier.
- Another advantage of this function is that it produces a value in the range of (0,1) when encountered with (- infinite, + infinite) as in the linear function.
- But suffers from vanishing gradient and learning is minimal.
- So it reaches the optimized value very slowly.
- Not zero centered.

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

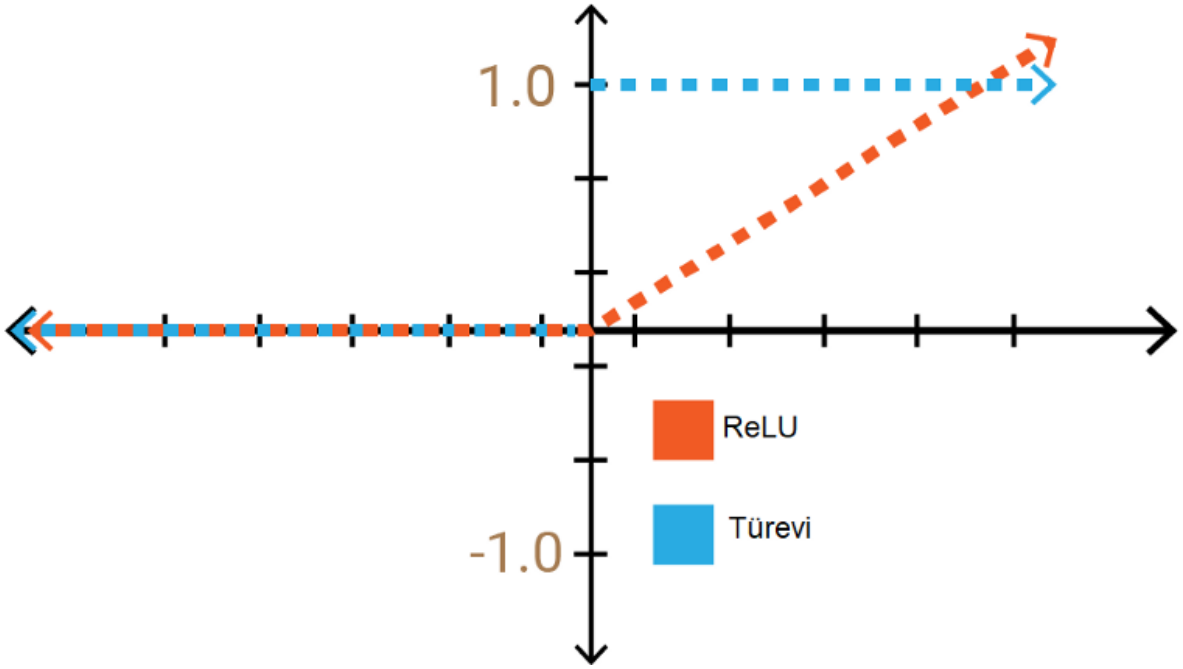
Activation Function: TanH Function



- Expanded version of Sigmoid.
- Zero-Centered
- Faces the phenomenon of vanishing gradient.

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

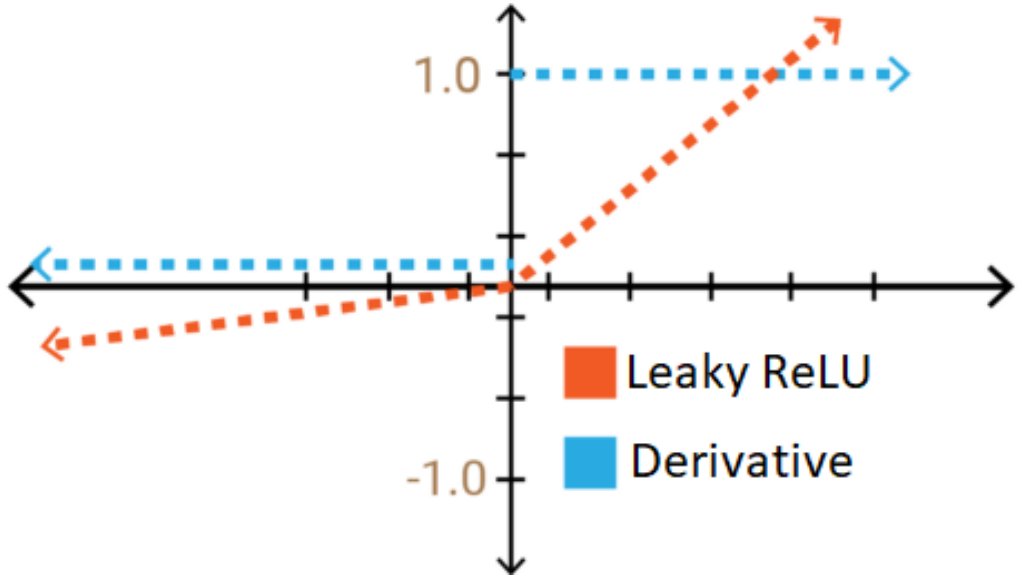
Activation Function: ReLU Function



- Same functionality as linear function
- Fast gradient descent
- Zero is also included
- If the value encountered at a particular instance is negative, then the value becomes Zero and hence the neuron gets dead

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

Activation Function: Leaky ReLU Function



- Same functionality as linear function and advance version of ReLU
- Fast gradient descent
- Zero is also included
- Ranges from +infinite value to – infinite value
- If the value encountered at a particular instance is negative, then the value doesn't become Zero and hence the neuron is not dead
- Once encountered the negative region, the values remains negative only everytime i.e. no limit to negative value.

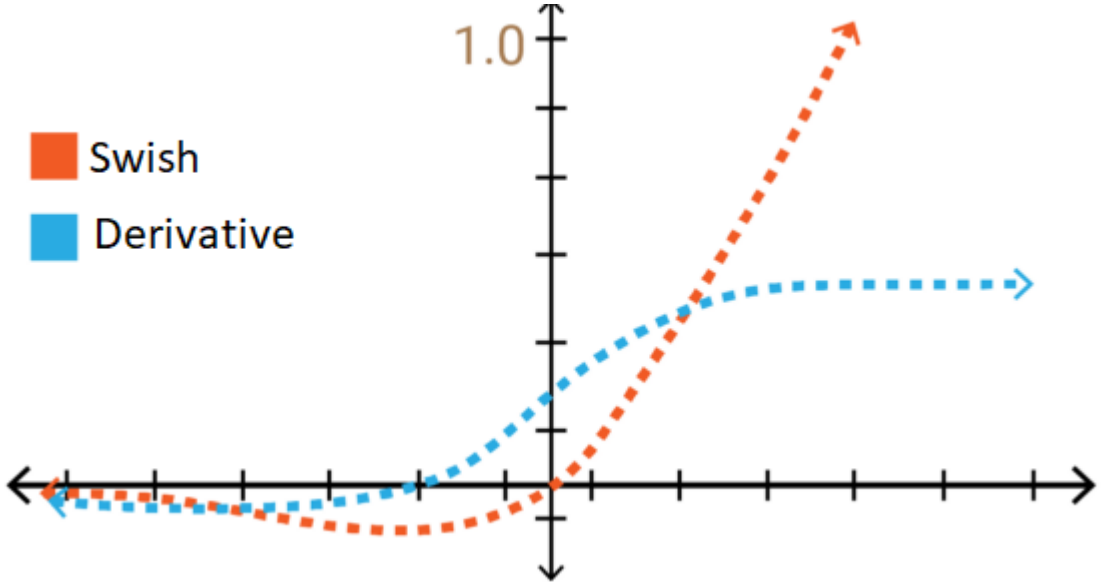
https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

Activation Function: Softmax Function

- Multi-layer classifier
- It is preferred in the output layer of deep learning models, especially when it is necessary to classify more than two classes.
- It allows determining the probability that the input belongs to a particular class by producing values in the range 0-1. So it performs a probabilistic interpretation.

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

Activation Function: Swish Function



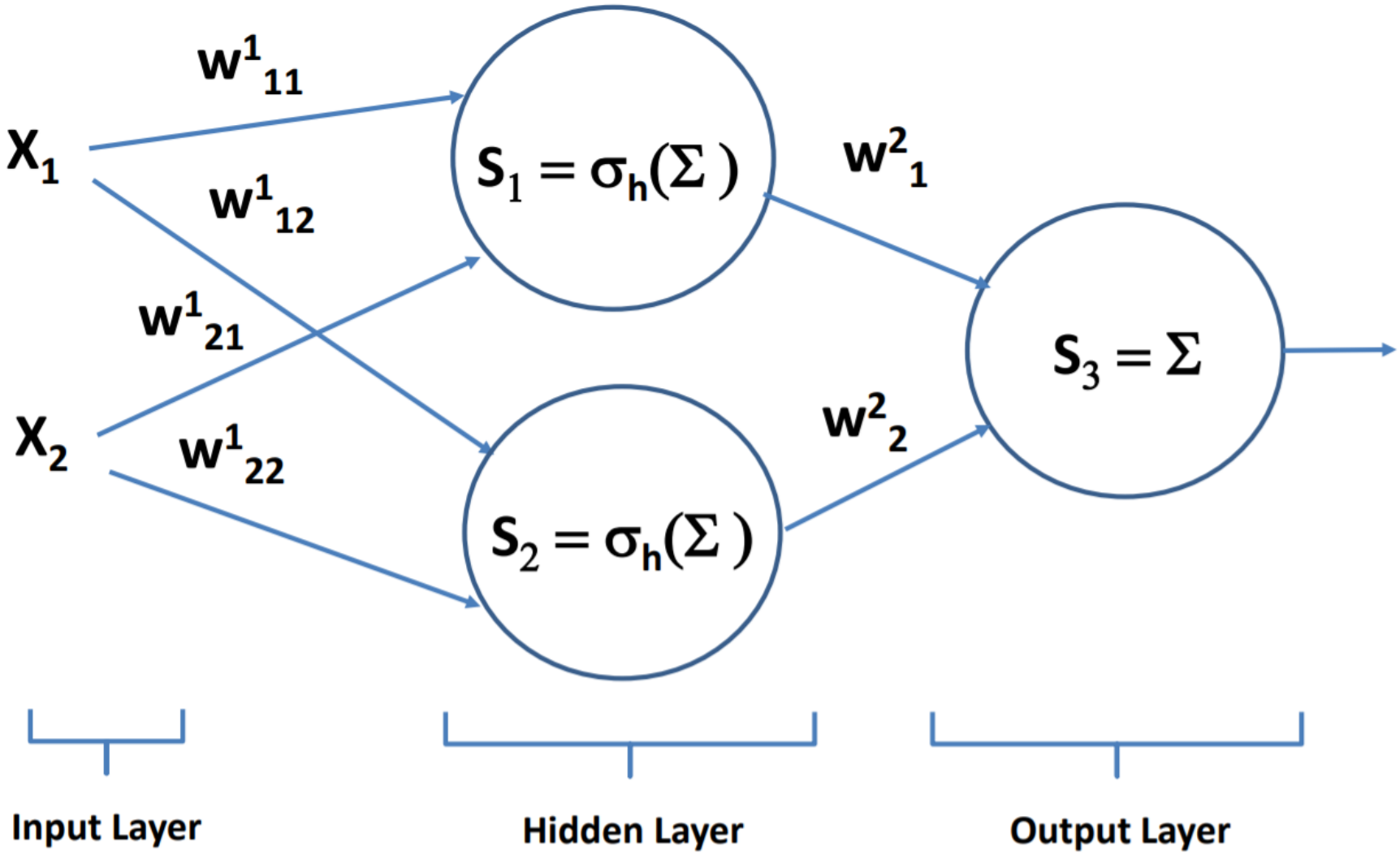
- F(x)=2*x*sigmoid(BETA*x)
- If BETA=0, it is linear function
- If BETA is very large (like infinity), it becomes two valued function i.e. (0 for x<0 and 1 for x>0) which is ReLU.
- So BETA=1 is chosen to get the functionality as given.
- It also don't have the vanishing gradient.

https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a

We Need Better Deep Representation

- An architecture with hidden layers for greater model capacity was required to understand the complex non linear functions and system behaviour
- The Neural network with Input Layer, (Multiple) Hidden Layers and Output Layer was designed
- Apply nonlinear activations in hidden units
- Can fully connect between layers
- Learn weights for complex function approximation

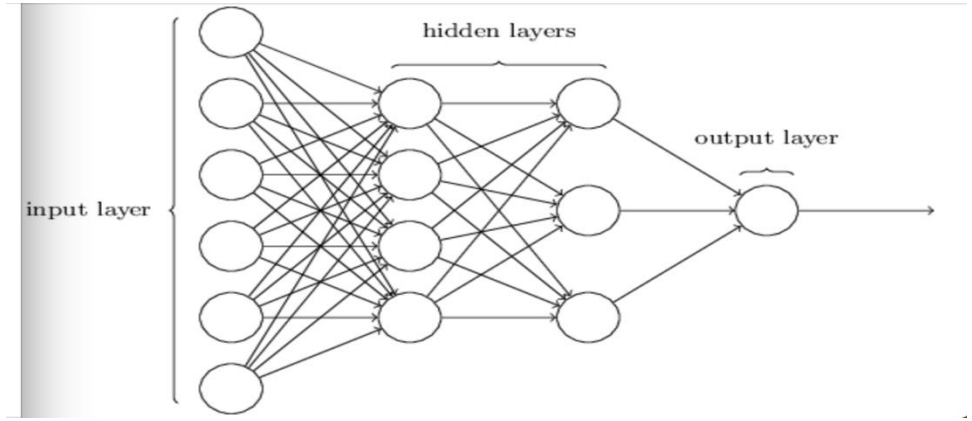
We Need Better Deep Representation

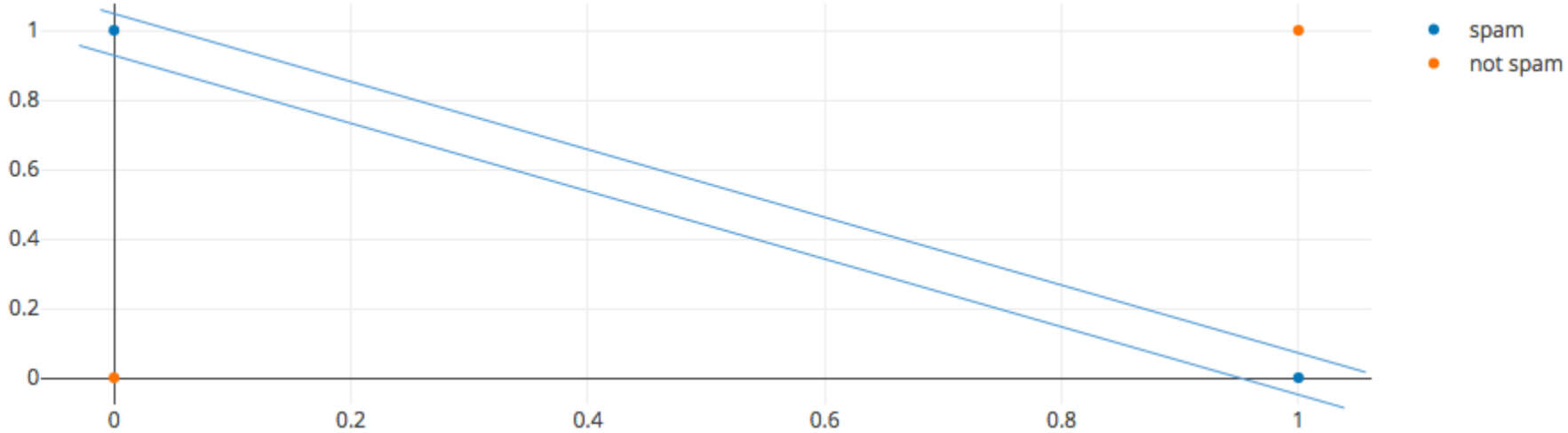


https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Neural Network

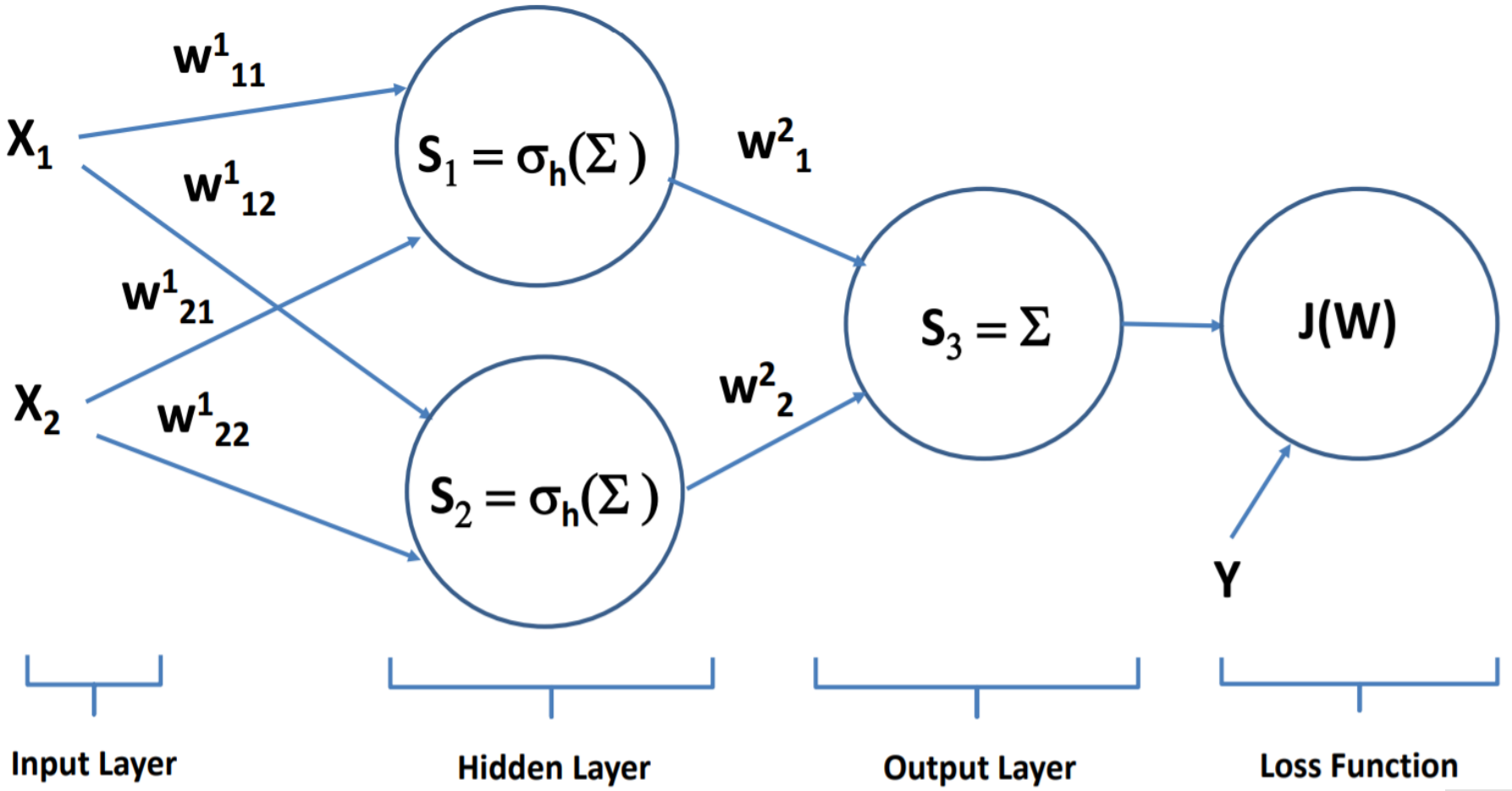
Multi layer Perceptron (non linear model)





https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Back Propagation Algorithm

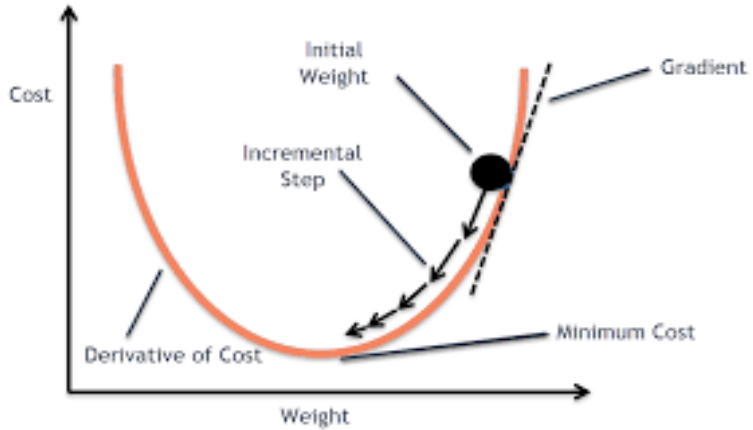


https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Gradient Descent

In backpropagation, to learn model weight tensor we must minimize the loss function using the gradient

$$W_{t+1} = W_t + \alpha \nabla_W J(W_t)$$



Where:

W_t = the tensor of weights or model parameters at step t

$J(W)$ = loss function given the weights

$\nabla_W J(W)$ = gradient of J with respect to the weights W

α = step size or learning rate

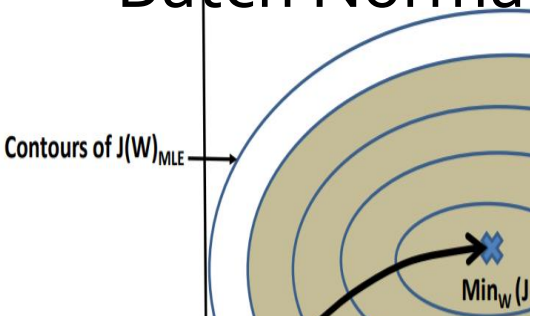
https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwi68cDi7cHmAhWZ6nMBHdwcCjEQjRx6BAgBEAQ&url=https%3A%2F%2Fgiphy.com%2Fgifs%2Fgradient-6QlTwkigqg4yk&psig=AOvVaw2fAKXSiPpaavP8DaNq5ZbP&ust=1576849890937900

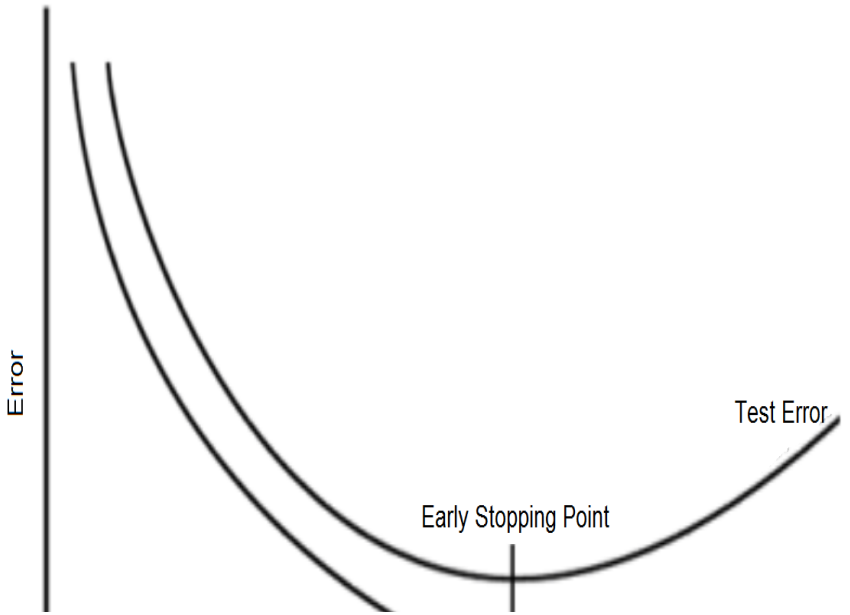
Regularization for Neural Network

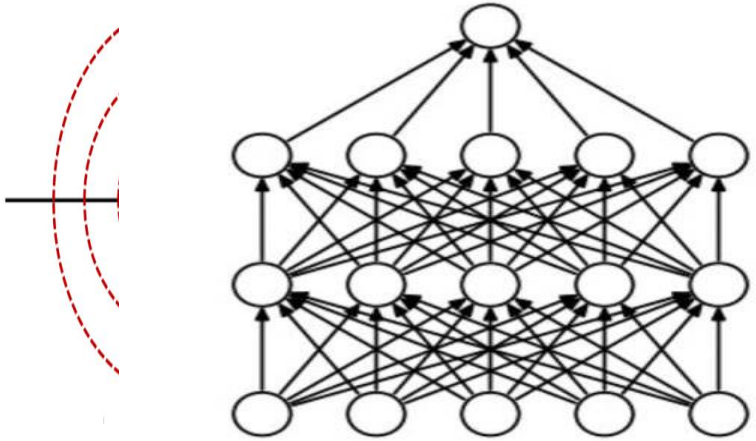
- Deep learning models have very large numbers of parameters which must be learned
- Large number of parameters lead to high chance of overfitting deep learning models
- Over-fit models do not generalize well and gives poor response to the input testing dataset

Regularization for Neural Network

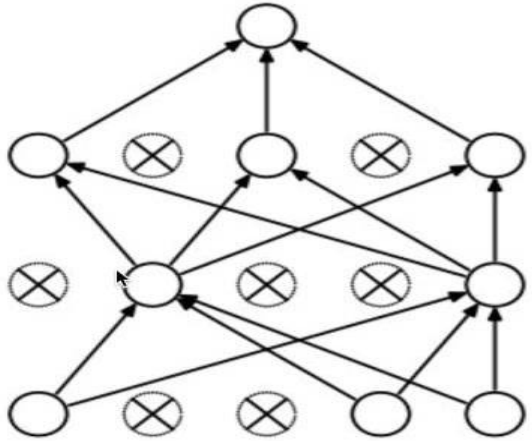
- Bias-Variance Trade-Off
- Early Stopping
- Dropout Regularization
- Batch Normalization







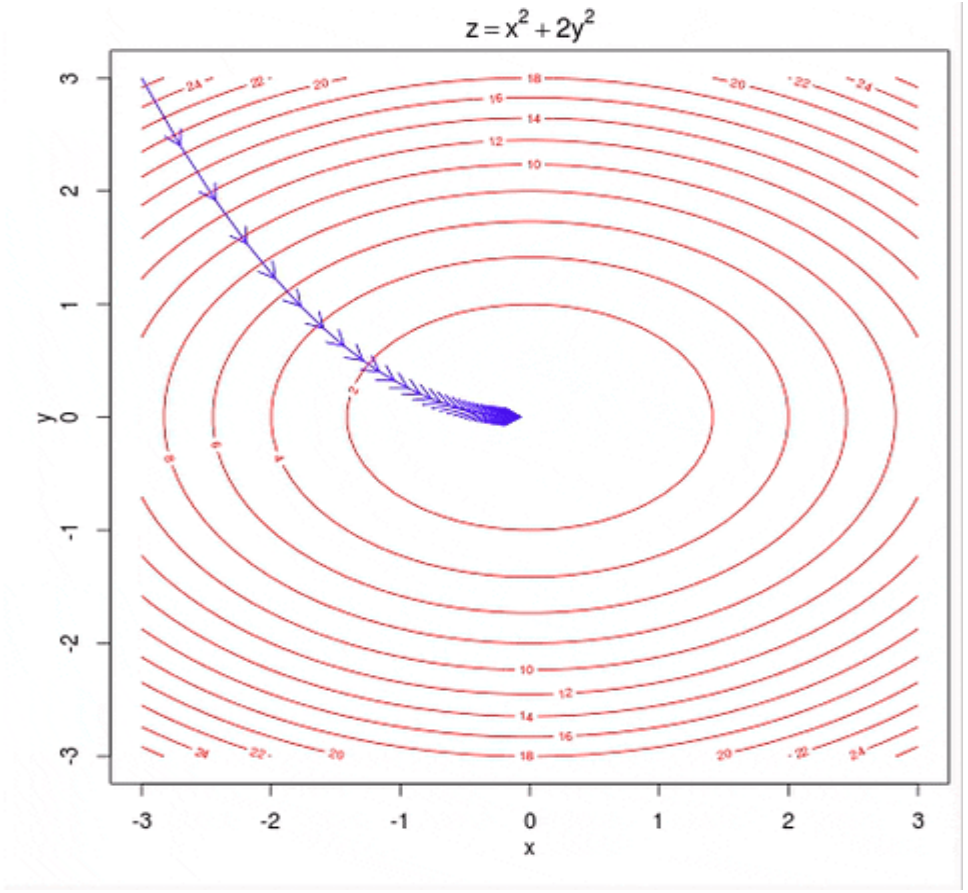
(a) Standard Neural Net



(b) After applying dropout.

Optimization for Neural Network

Neural networks learn weights using the backpropagation algorithm using Gradient Descent Method



https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwio8cDIychmAhWZ6nMBHdwcCjEQjRx6BAgBEAQ&url=https%3A%2F%2Fgiphy.com%2Fgifs%2Fgradient-6QlTwkigqg4yk&psig=AOvVaw2fAKXSiPpaavP8DaNq5ZbP&ust=1576849890937900

Optimization for Neural Network

- Ideally, the loss function is convex with respect to the weights
- Convex loss function has one unique minimum
- Convergence for convex loss function is guaranteed
- Real-world loss functions are typically not convex
- There can be multiple minima and maxima; a multi-modal loss function
- Finding the globally optimal solution is hard
- In practice, we are happy with a good local solution, if not, the globally optimal solution
- In practice, nature of gradient is not guaranteed
- Gradient can vanish or explode

Types of Optimization Techniques for Neural Network

- Gradient Descent

- Stochastic Gradient Descent

Stochastic Gradient Descent performs a parameter update for each training example unlike normal Gradient Descent which performs only one update. Thus it is much faster.

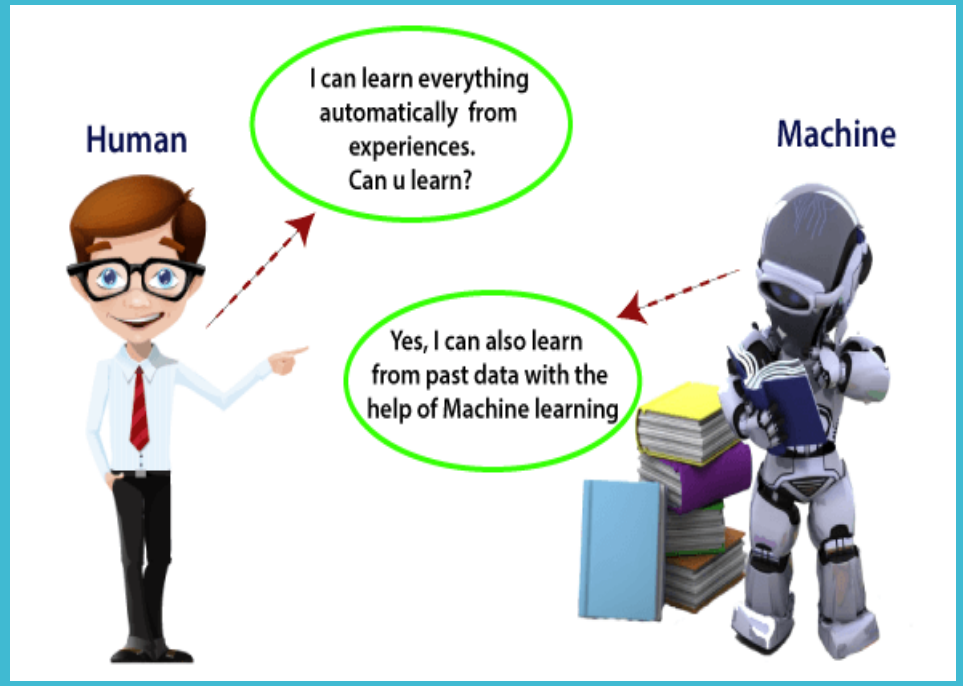
- Adagrad

Adagrad is more preferrable for a sparse data set as it makes big updates for infrequent parameters and small updates for frequent parameters. It uses a different learning Rate for every parameter θ at a time step based on the past gradients which were computed for that parameter. Thus we do not need to manually tune the learning rate.

- Adam

Adam stands for Adaptive Moment Estimation. It also calculates different learning rate. Adam works well in practice, is faster, and outperforms other techniques.

Understanding the significance of Loss Function Curves



Marwadi University

Department of Information and Communication Technology

Unit 7:
Deep Learning

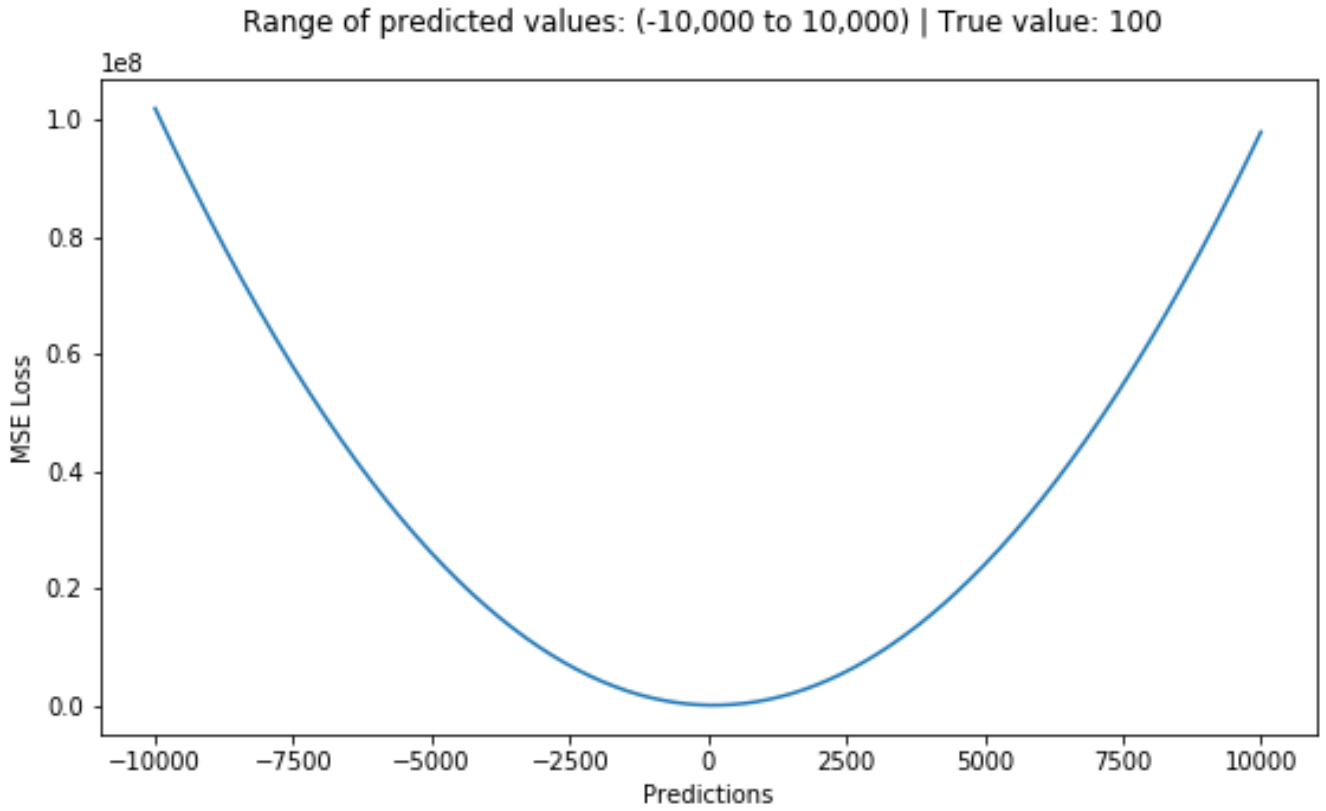
Artificial Intelligence (01CT0703)

Prof. Nishith Kotak

Regression Loss Functions

- **Mean Square Error, Quadratic loss, L2 Loss**

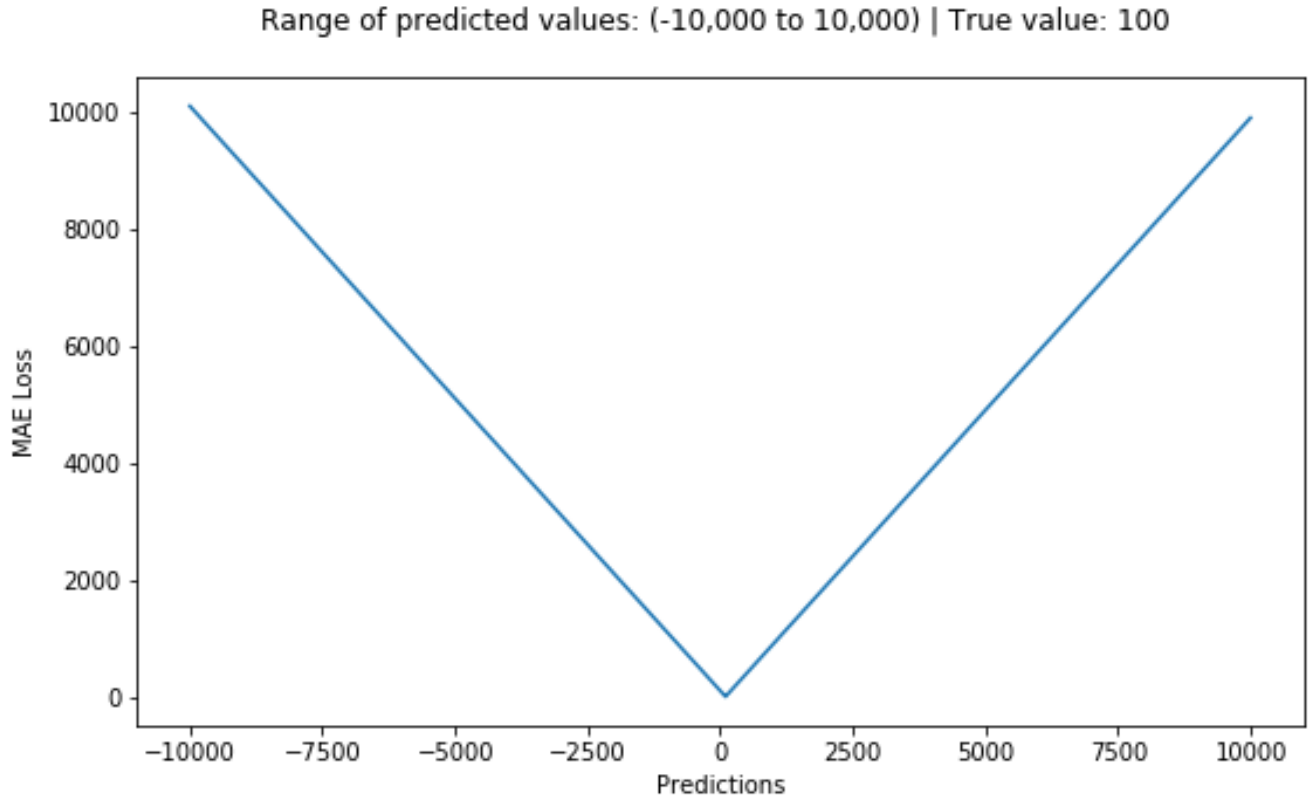
$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$



Regression Loss Functions

- **Mean Absolute Error, L1 Loss**

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$



Regression Loss Functions

- **MAE VS MSE i.e. L1 VS L2 Loss**

MAE vs. RMSE for cases with slight variance in data

ID	Error	|Error|	Error²
1	0	0	0
2	1	1	1
3	-2	2	4
4	-0.5	0.5	0.25
5	1.5	1.5	2.25

MAE: 1 RMSE: 1.22

MAE vs. RMSE for cases with outliers in data

ID	Error	|Error|	Error²
1	0	0	0
2	1	1	1
3	1	1	1
4	-2	2	4
5	15	15	225

outlier

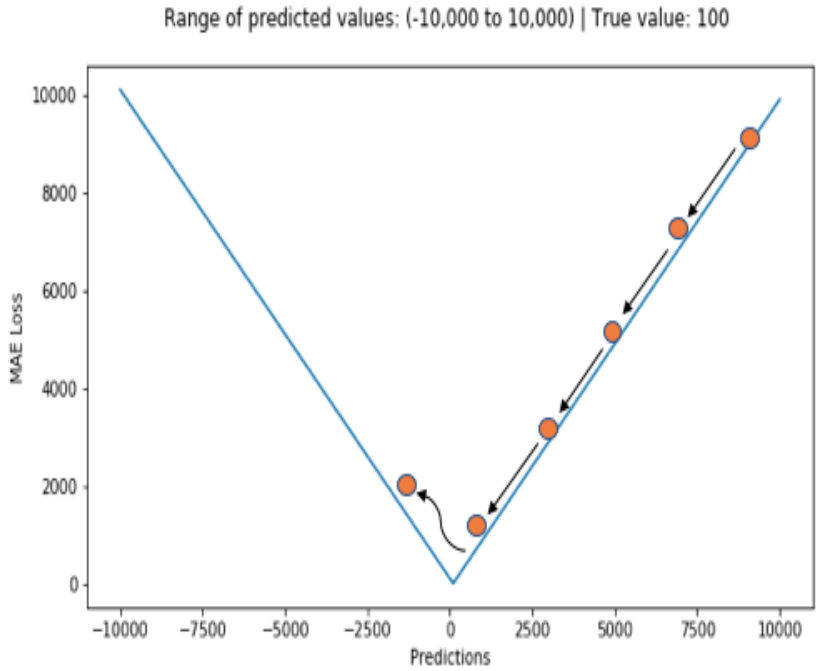
MAE: 3.8 RMSE: 6.79

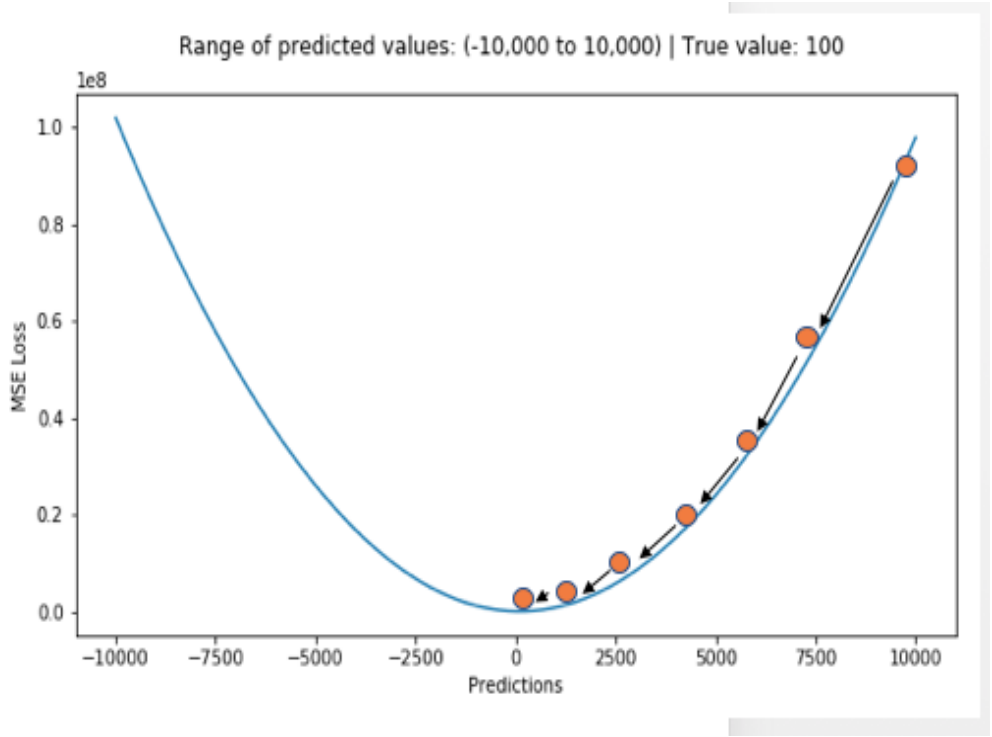
- L1 loss is more robust to outliers, but its derivatives are not continuous, making it inefficient to find the solution.
- L2 loss is sensitive to outliers, but gives a more stable and closed form solution (by setting its derivative to 0.)

Regression Loss Functions

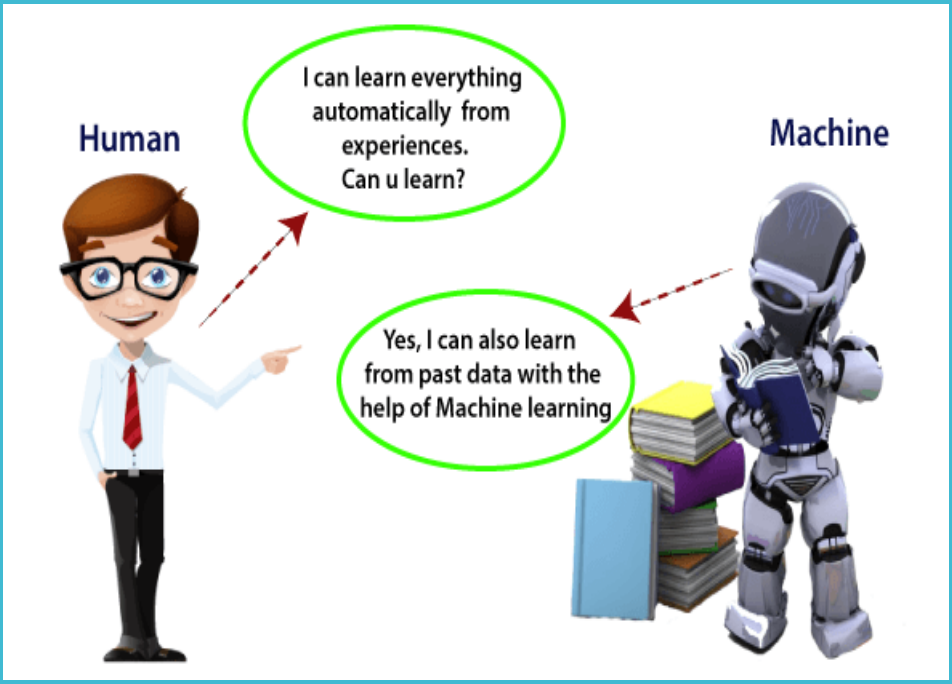
- **MAE VS MSE i.e. L1 VS L2 Loss**

- One big problem with using MAE for training of neural nets is its constantly large gradient, which can lead to missing minima at the end of training using gradient descent.
- For MSE, gradient decreases as the loss gets close to its minima, making it more precise





CNN and RNN models



Marwadi University

Department of Information and Communication Technology

Unit 7:
Deep Learning

Artificial Intelligence (01CT0703)

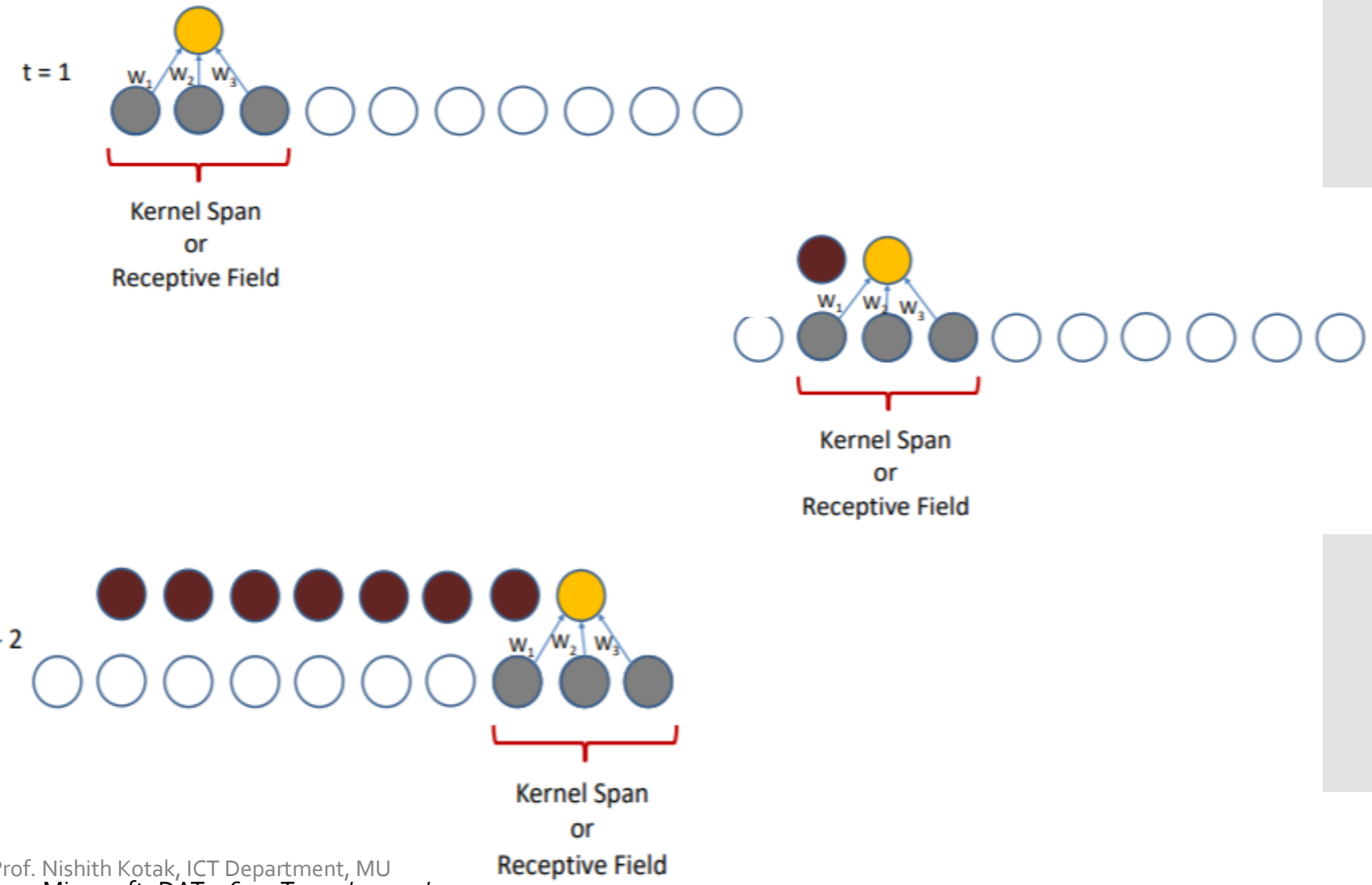
Prof. Nishith Kotak

Convolutional Neural Network

- CNNs are used to learn complex feature maps
- Reduce the dimensionality of input tensors
- Share weights and are relatively easy to train
- The weights of the convolutional kernel must be learned
- Each weight of a fully connected network must be learned independently
- CNNs are efficient to train
- Weights are learned using backpropagation and gradient descent methods

Convolutional Neural Network

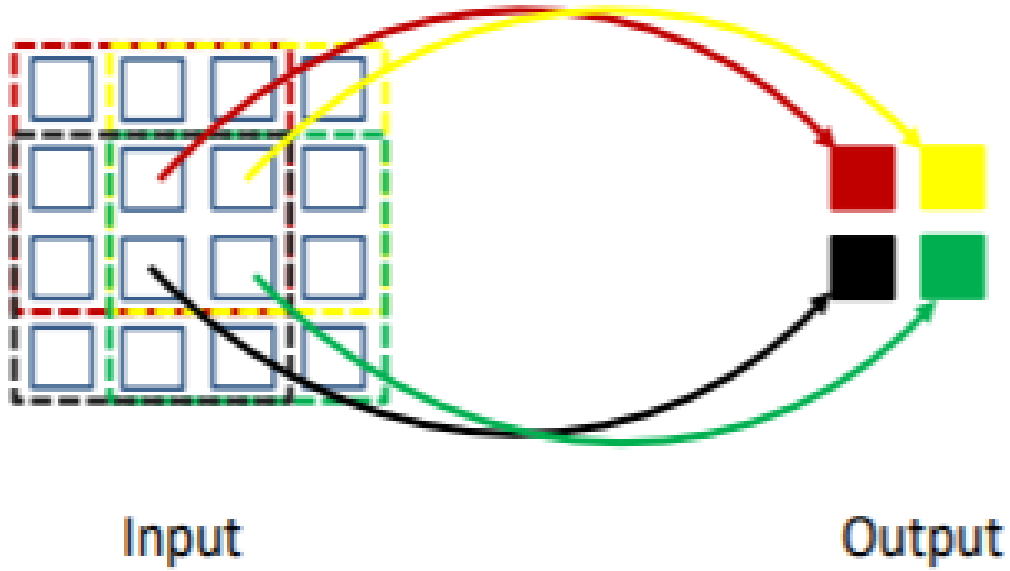
- 1-D Convolution



https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

Convolutional Neural Network

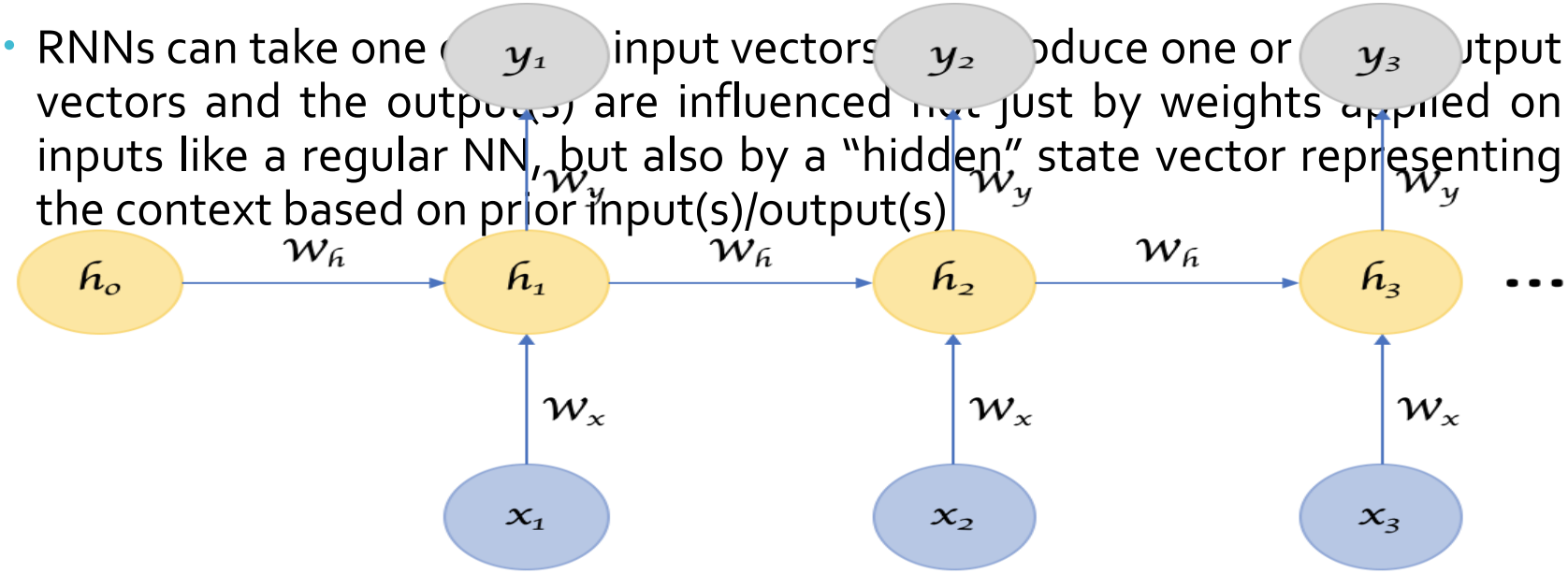
- 2-D Convolution
  - ✓ 4 x 4 input tensor
  - ✓ 3 x 3 convolution operator
  - ✓ 2 x 2 output tensor



https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+2T2019/course/

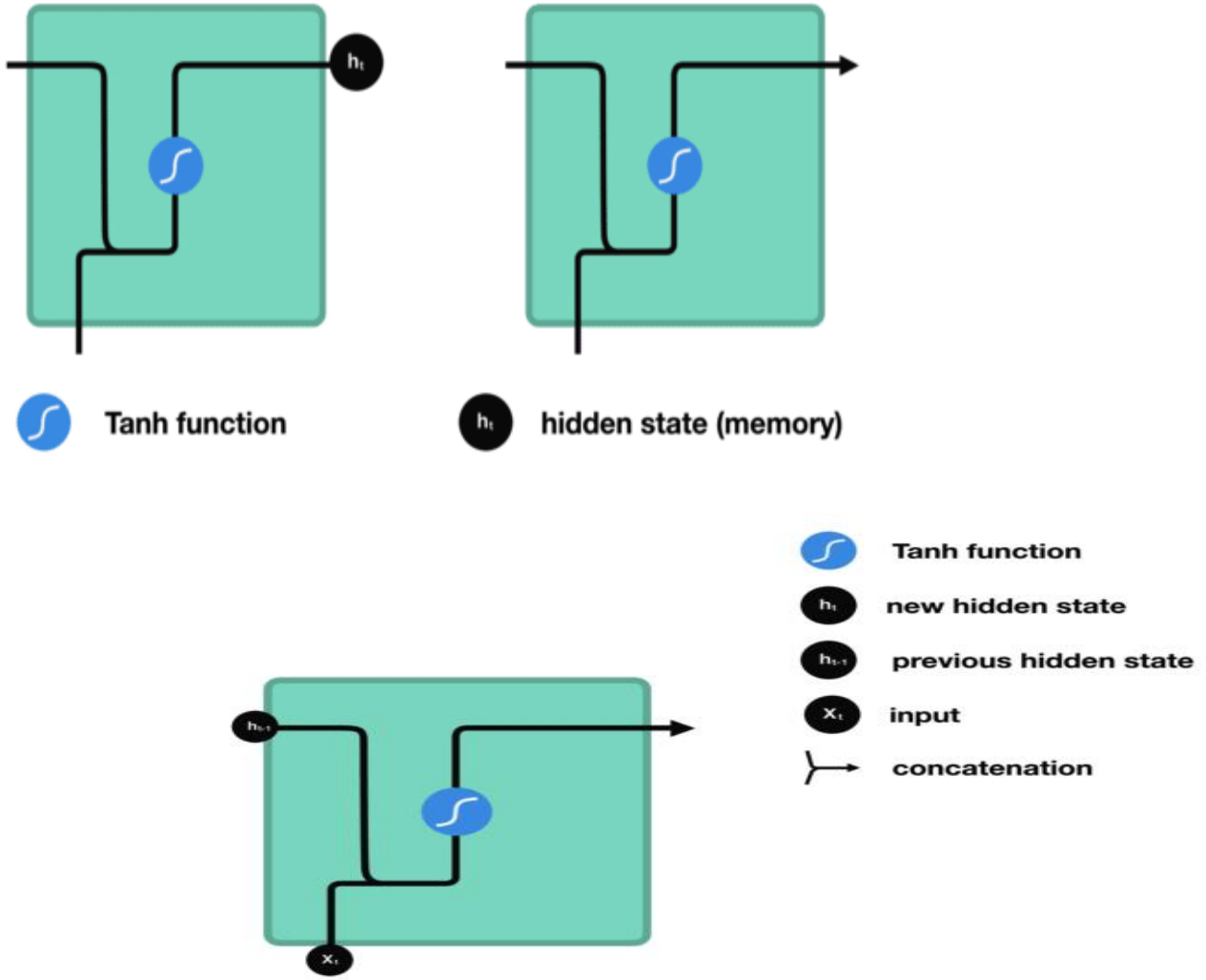
Recurrent Neural Network

- An RNN recursively applies a computation to every instance of an input sequence conditioned on the previous computed results.
- RNN remembers the past and it's decisions are influenced by what it has learnt from the past.
- The main strength of an RNN is the capacity to memorize the results of previous computations and use that information in the current computation.
- RNNs can take one (input vectors oduce one or utput vectors and the output(s) are influenced t just by weights applied on inputs like a regular NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s)



https://medium.com/dair-ai/deep-learning-for-nlp-an-overview-of-recent-trends-d0d8f40a776d

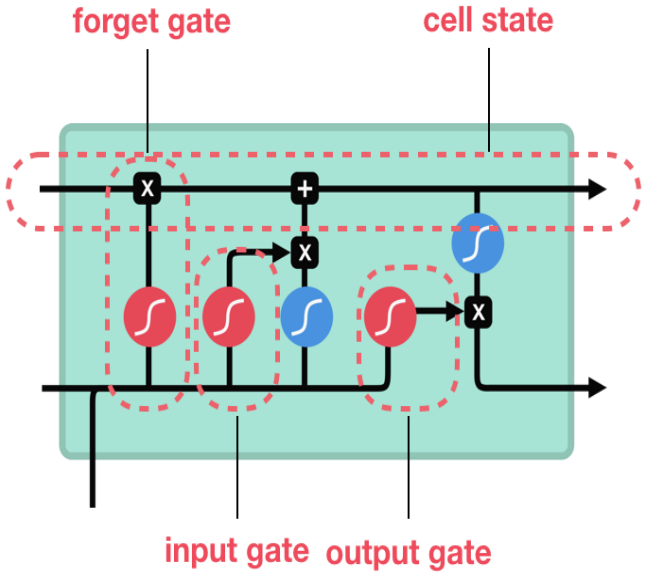
Recurrent Neural Network



Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Long Short Term Memory

- For a long sequence of data, RNN stops learning while backpropagation due to vanishing gradient phenomenon
- LSTM have gates that regulates the flow of information
- The gates are different neural networks that decide which information is allowed on the cell state.
- These gates can learn which data in a sequence is important to keep or throw away in the long chain of sequences to make predictions



Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Long Short Term Memory

- The Forget gate decides what is relevant to keep from prior steps.
- The input gate decides what information is relevant to add from the current step.
- The output gate determines what the next hidden state should be.

Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Long Short Term Memory

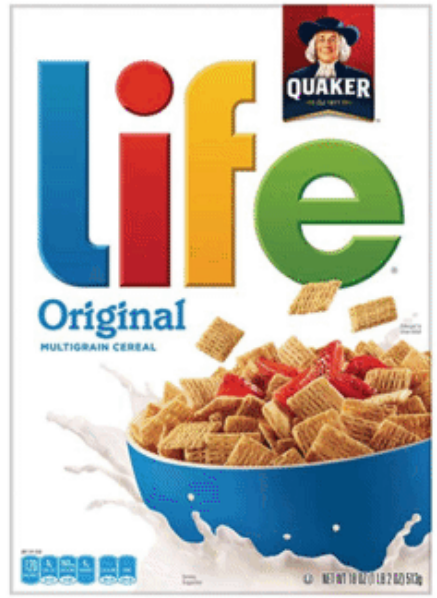
Customers Review 2,491

Thanos

September 2018

Verified Purchase

Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!



A Box of Cereal

$3.99

Source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21