NUS Fintech Society

Machine Learning Department
Training Wing

Session 8: Deep Learning (31/10/2020)

Content

1. Inspiration for Deep Learning

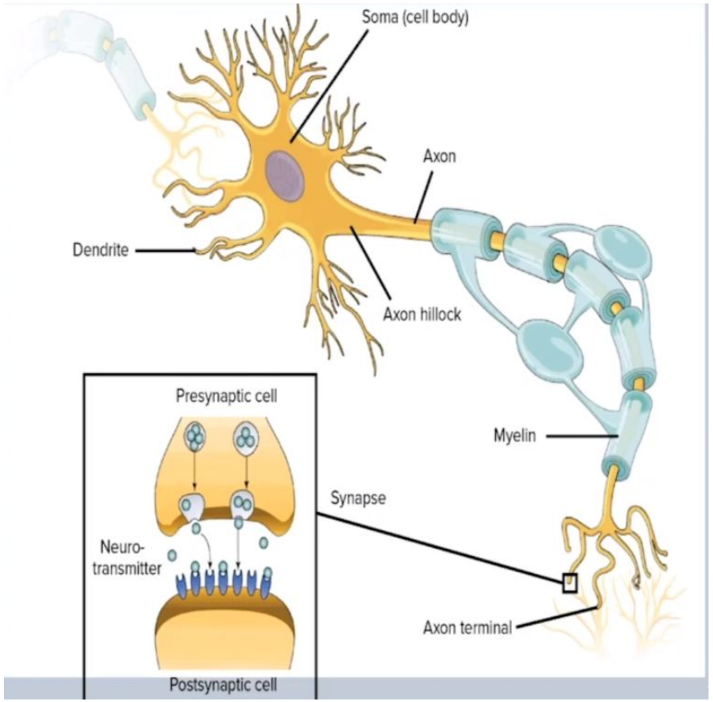
Biological Neuron

- There are approximately 100 billion neurons in an adult

- Each neuron is connected up to 10,000 other neurons through connections called synapses.

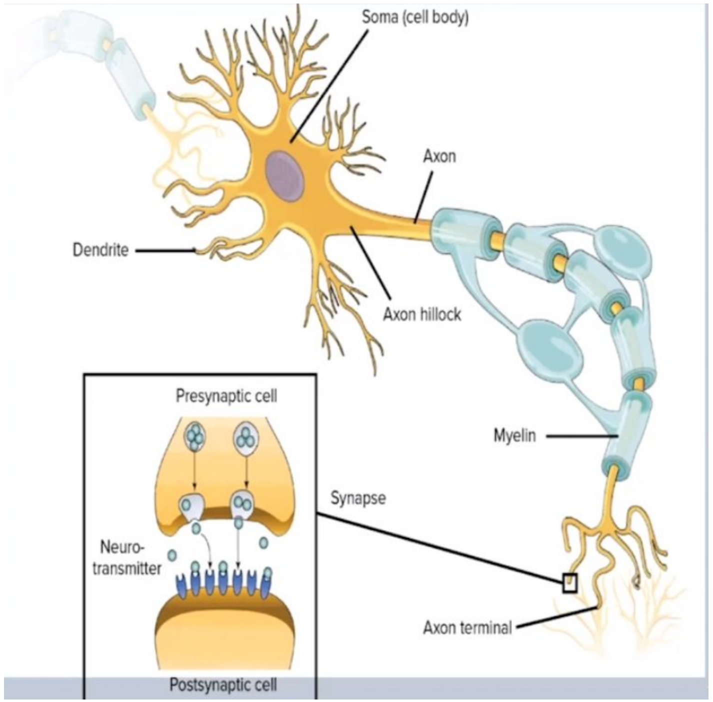
- These neurons vary in length from few centimeters to the ones beyond meter.

- A neuron fires electric signal when the membrane potential crosses threshold value (Action Potential).

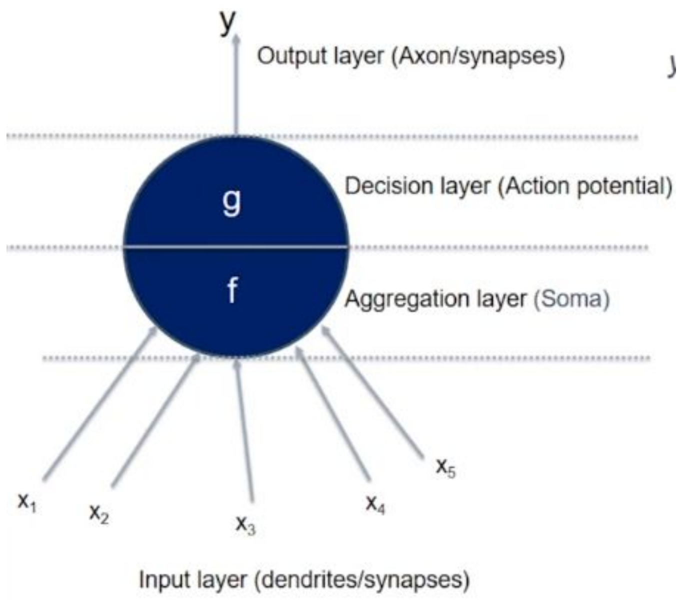


Biological Neuron

- These electric signal travels through the axon and reaches the synapses that are activated by neurotransmitters.
- Neurons that fire together wire together'
- If you don't use a connection, you lose it.



Artificial Neuron - McCulloch & Pitss Neuron (MP neurons)



y
Output layer (Axon/synapses)
g
Decision layer (Action potential)
f
Aggregation layer (Soma)
x_1
x_2
x_3
x_4
x_5
Input layer (dendrites/synapses)

$$y = 1 \text{ if } g(\textstyle\sum_{i=1}^{n} x_i) >= \text{threshold (fire)}$$

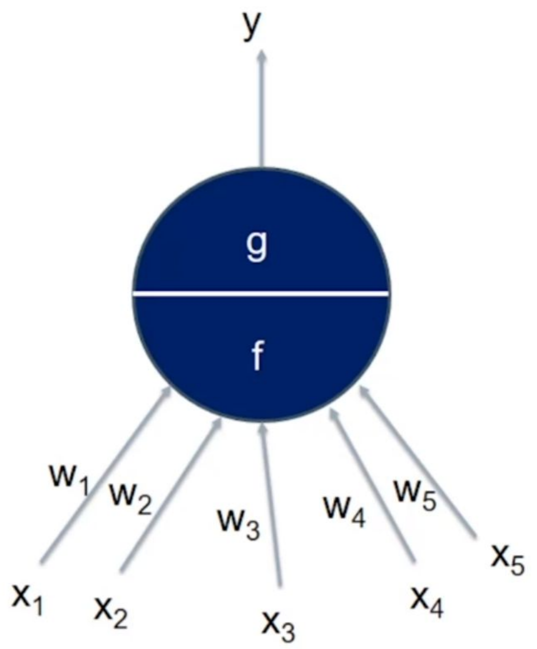
$$= 0 \text{ if } g(\textstyle\sum_{i=1}^{n} x_i) < \text{threshold (no fire)}$$

$x_1, x_2 \ldots x_n$ are Boolean values

MP neurons can be used to solve decision problems like AND, OR, NOR, NOT gates

All Boolean functions can not be represented by MP neurons e.g. XOR

Artificial Neuron- Perceptron



Key difference

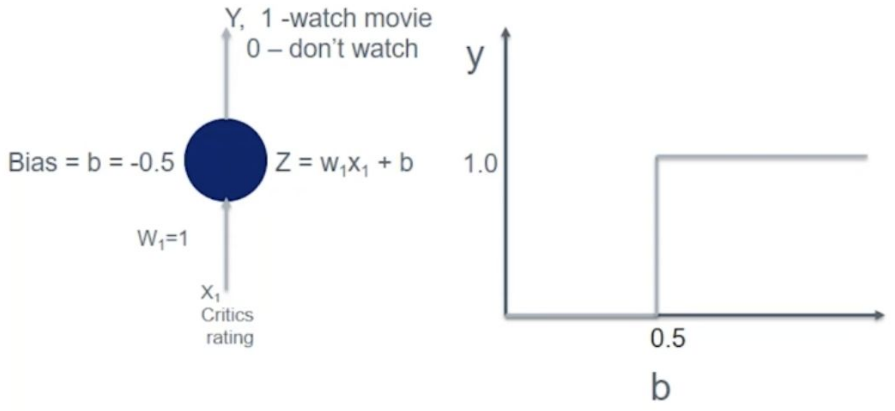
- A learning algorithm that can assign weights (W1, W2,....Wn)

- Inputs are no longer Boolean values but real numbers

e.g. predict the sale of a movie(y) based on various inputs parameters like genre (x_1), amount invested (x_2), rating of the movie (X_3) and so on.

Artificial Neuron - Perceptron

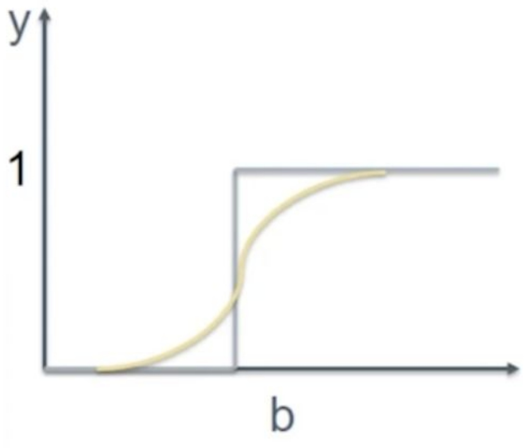
Y = f(x), where x and y are both real numbers (unlike Boolean values)



- Perceptron has a problem that it has jump (sharp decision boundary)
- For example a value of 0.49 results into y=0 while a 0.51 results into y=1
- Such fluctuations are characteristic of perceptron neurons

Sigmoid Neuron

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



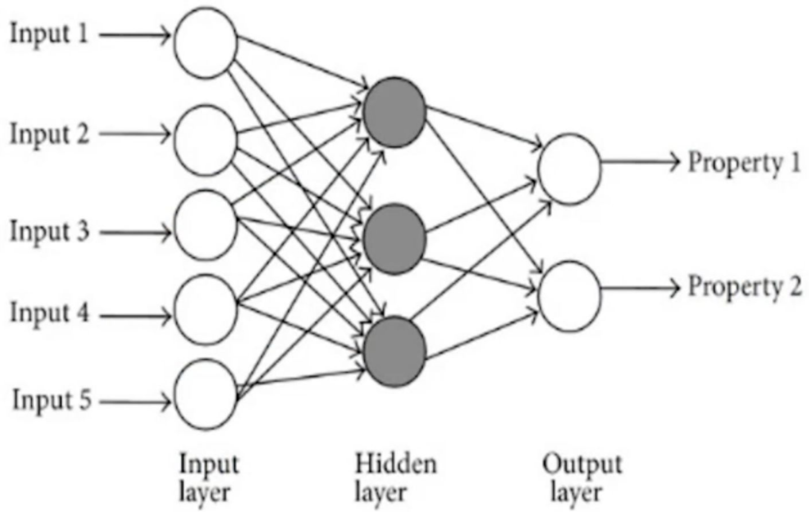
A sigmoid neuron

- A sigmoid neuron smoothens the output function
- The function becomes differentiable

Logistic function $Y' = \frac{1}{1+e^{-(\sum_{i=0}^{n} w_i * xi)}}$

$= \frac{1}{1+e^{-WTX}}$ (matrix form)

Artificial Neuron - Multi Layered Perceptron



Universal Approximation Theorem

- A multi-layered network of neurons with a single hidden layer can approximate any continuous function (whether linearly separable or not) to any desired level of approximation

Business application of Deep Learning

AUTOMATION

- Computer vision (Face recognition, object detection)
- Image Captioning
- Language modelling
- Machine translation ,
- Conversational modelling (Speech generation and recognition
- Fraud detection
- Document classifications

CREATIVITY

- New Image generation
- Music creation
- Automated news articles
- Deep art

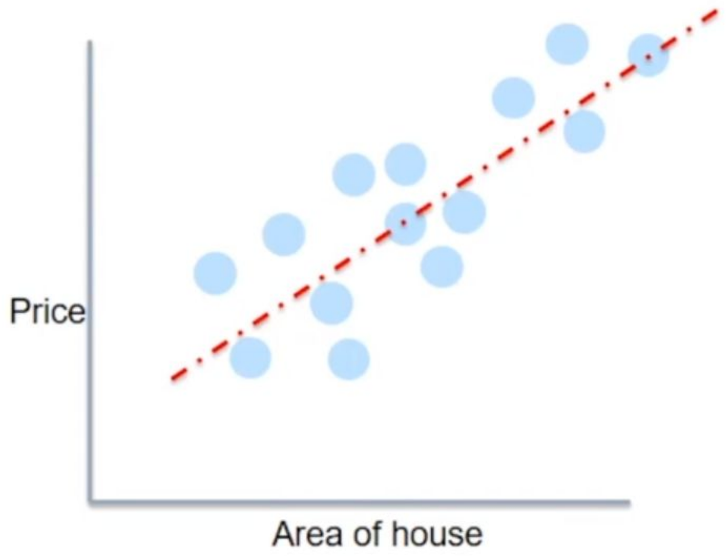
Quiz 1 (Inspiration for Deep Learning)

Go to https://b.socrative.com/login/student/

Room Name: NUSFINTECH

2. Key concepts of Deep Learning

From Machine Learning to Deep Learning



A linear equation can be solved without the need for hidden neurons. But if the decision boundaries are not linear (e.g circle then deep learning will be necessary)

Problem statement

- Given several pair of area and house prices predict what would be the expected price for the house of a given area.
- Presuming that weight is a linear function of height
- Model Pi = W * Ai + b
- W is called the slope weight and b is called bias.

Select a line in such a way that the average distance of each point is minimum

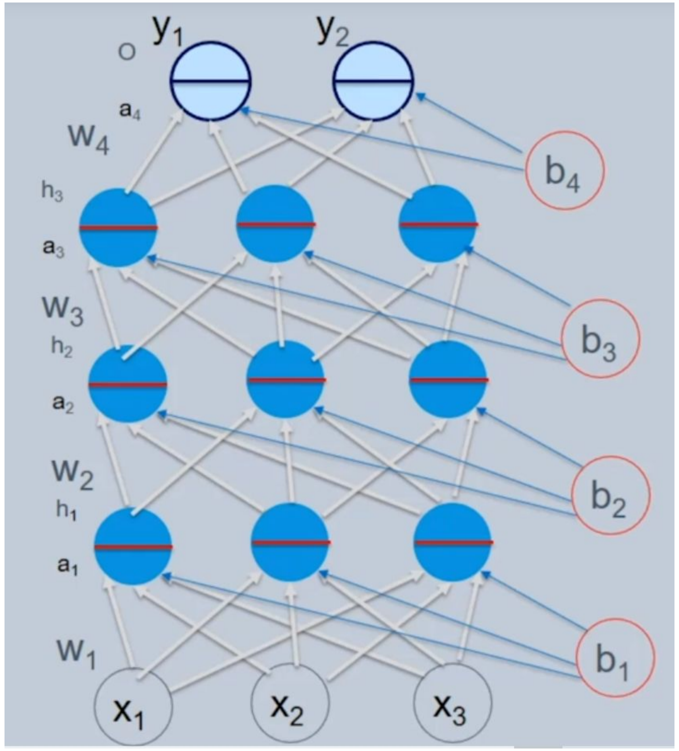
Objective of the training is to Minimize the loss function

Mean square error = $\sum_{i=1}^{n} (Pi-Pi')2$
By selecting the right value of W (slope) and b (bias)

Constructing deep neural network

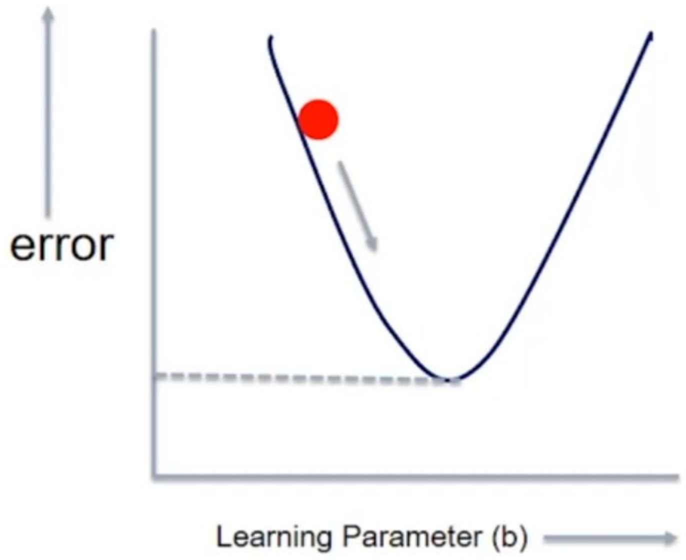
- x_i -> inputs
- Y_i -> output
- W_i -> weights
- b_i -> biases
- a_i -> aggregation layer
- h_i -> activation function
- O -> final output activation function

- **Data** : $\{x_i, y_i\}$
- **Model** $y' = f(x_i) = O(W_4 g(W_3 g(W_2 g(W_1+b_1)+b_2)+b_3)+b_4)$
- **Parameters to learn** — W, b
- **Algorithm-** Gradient descent



Gradient Descent

- Intuition: Rolling the ball down the valley for it to find the minimum height

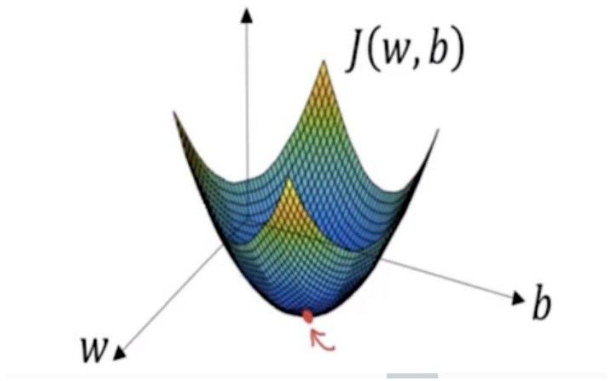


- To minimize the error move in the direction opposite to the gradient

for $y = J(w,b)$

$$w_{t+1} = w_t - \mu * \frac{\partial(J(w,b))}{\partial w}$$

$$b_{t+1} = b_t - \mu * \frac{\partial(J(w,b))}{\partial b}$$



Different Types of Gradient Descent

1. Momentum based Gradient descent— If the gradient is high move slow, if the gradient is low move fast. This will ensure faster convergence. It may require some U turn (overshoot the minimum)

2. Nesterov Gradient descent — Use the gradient at the look ahead point. This can lead to faster convergence by having smaller oscillating U turns

3. RMSProp — For sparse data use the number of updates as one of the parameter

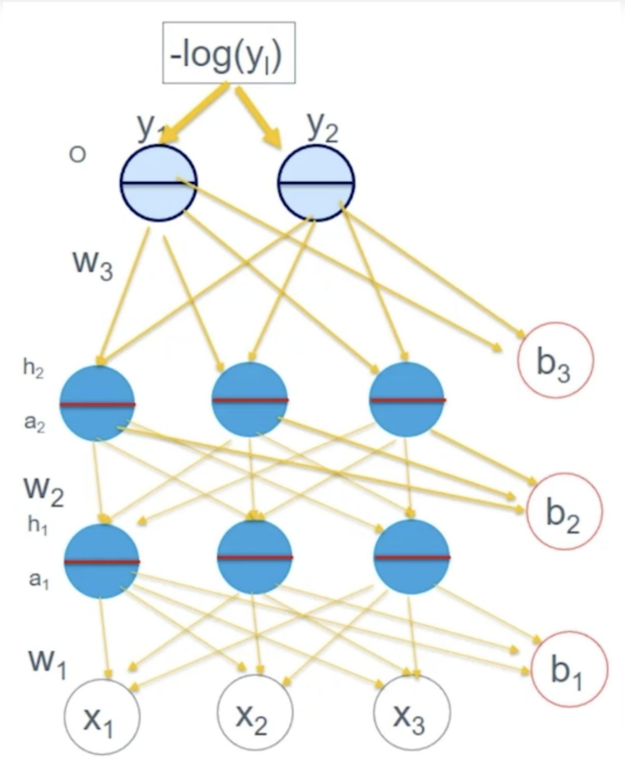
4. Adam —Adaptive momentum (updating the bias), this is most commonly used.

Backpropagationn

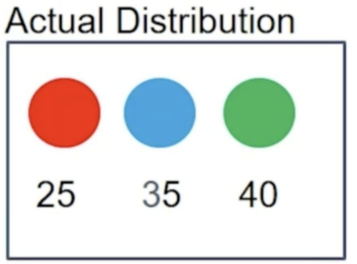
- Arrive at the error (loss) at the output layer
- Find out how the connected neurons are contributing to the error. Partial derivatives of loss with hidden layers

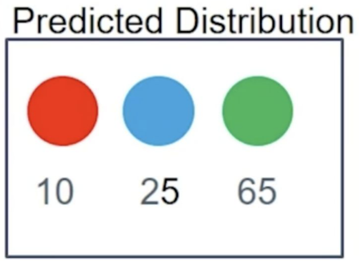
A chain rule can be applied from output function to hidden layers

- Find the Gradient w.r.t hidden layers
- Find the Gradient w.r.t weights and bias
- Keep traversing the path till the input layer



Classification Problem





In case of classification problem (e.g. red, blue green balls in a bag)
- The actual output is true probability (pi) distribution (0.25, 0.35, 0.40)
- Estimated probability (qi) distribution (0.10, 0.25, 0.65)

We can use mean squared error as the difference between the true and estimated probabilities.

However there is better method to solve the probability distribution problem

In case of classification problems there is one more uniqueness that all the probabilities should sum to 1. Hence sigmoid function may not be useful because the probabilities will not sum to 1. Softmax function makes sure that the sum is 1.

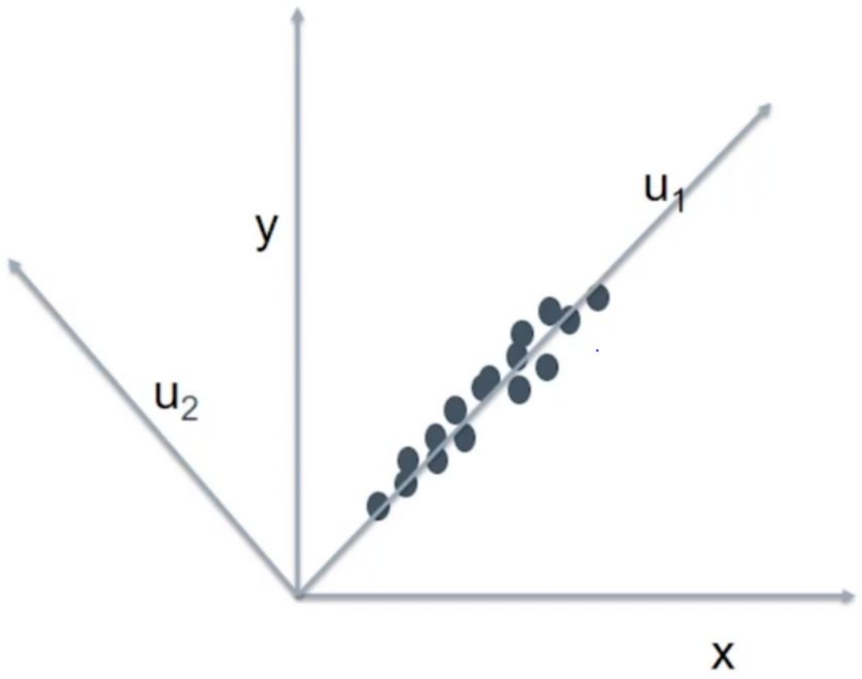
Quiz 2 (Key concepts of Deep Learning)

Go to https://b.socrative.com/login/student/

Room Name: NUSFINTECH

3. Artificial Neural Network(ANN)

Principal Component Analysis



- Suppose the data is represented as the points.

- There is high variance across x and y axis.

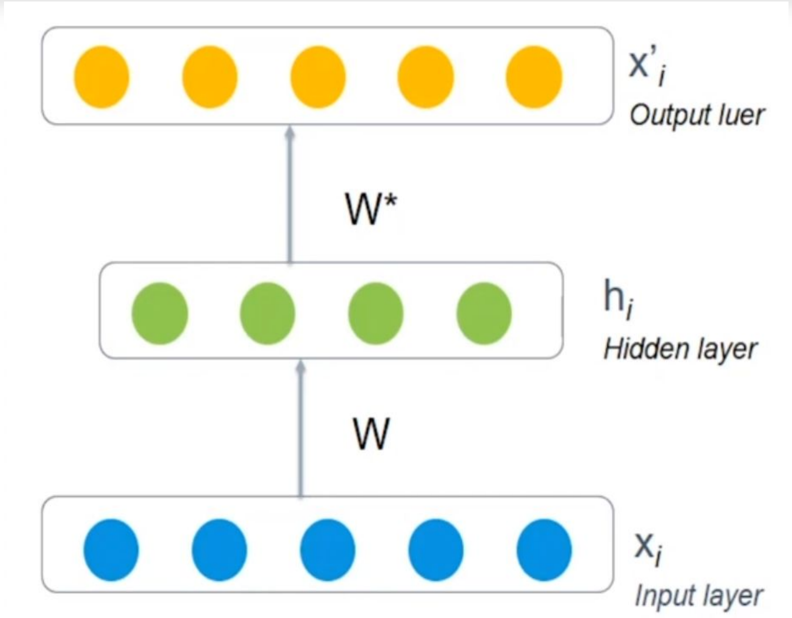
- If we can find a new orthogonal vector ul and u2 (linearly independent and orthogonal)

- Variability for u1 is high and hence dimension u2 can be dropped as the variability is not significant

- Principal component analysis is transforming the data to a new basis where the dimensions are non redundant (low covariance) & high variance.

-The objective is to identify top k-dimension.

Autoencoders



Autoencoders are special type of feed forward neural network that

Step 1 : Encodes the input into a hidden representation
$h = g(Wx_i + b)$
Step 2 : Decodes the input again from the hidden representation

$x' = f(W^*h + c)$
Step 3: Minimize the error between x and x'

Types of Autoencoder
1. Under-complete Autoencoder ,If dimension (h,) < dimension (x,)
2. Over-complete Autoencoder, If dimension (h,) > dimension (x)

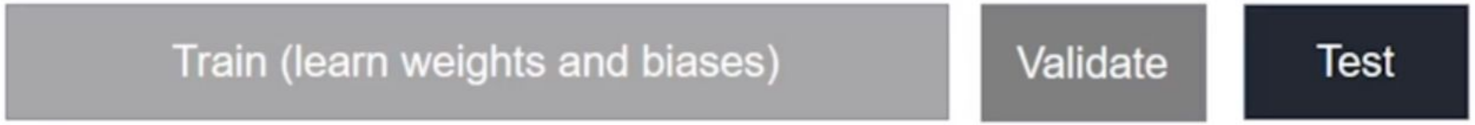
Data segregation - train, validate and test

To arrive at the most robust model the original data sets are divided into

• Train Data — Data used to train/fit the model. The model learns weights and biases using this dataset.

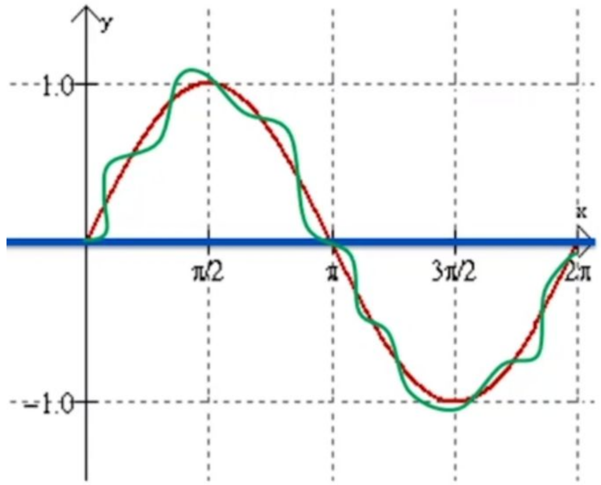
• Validate Data — Dataset used to tune the model. This is part of the sample data that provides unbiased evaluation of the model fit during tuning of the hyper-parameters. This data is not used to learn weights and biases.

• Test Data — Dataset used to evaluate the model when it is trained completely using above two datasets.



Overfitting is the tendency of the model driving training error to zero

Model complexity



Model Choices

1. Simple model (straight line)

y = mx +c

- Low variance between train and test data
- High bias (far away from true function)

2. Complex model (degree 25 polynomial)

$y = \sum_{k=0}^{25} m_k x^k + c$

- High variance between train and test data
- Low bias (over fitment to train data)

An optimum model is the one that does not drive the train error to zero but to a minimum constant value.

Regularization

Overfitting of data – the model makes the error as zero for training data but it still may have large error for test data. In case of over-complete auto-encoders, it can just copy the input to output creating a wrong model for other data. Overall idea of regularization is to restrict the model so that the weights and biases can't assume any arbitrary values.

Regularization is added to overcome the above problem

- Weight decaying (with weights so that it can't assume any arbitrary value)

  New Loss function = squared error + $\mu \|f(w)\|^2$

- Tieing the weights $W^* = W$
- De-noising - Corrupting the input data and consider the error function with original input

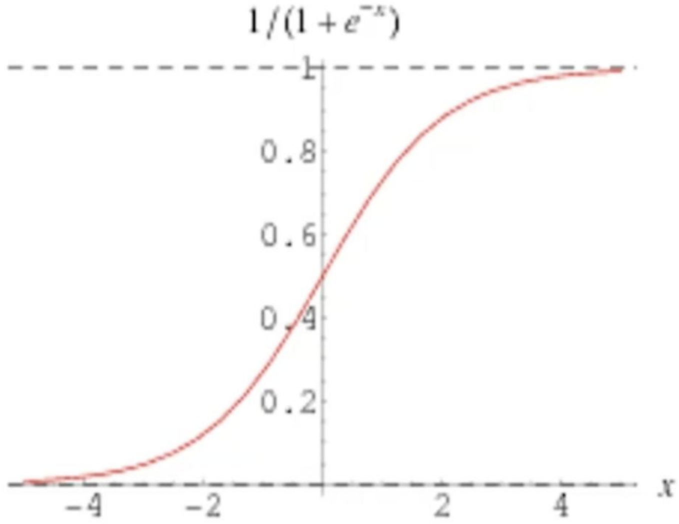
  - add a Gaussian noise to each inputs

  - randomly remove some inputs

- Sparse regularization – restrict the weights in such a way that average activation of each neuron is close to zero. A sparsity constraint is added to loss function for this purpose.

New Loss function = squared error + Sparsity constraint

Sigmoid Activation Function



Advantage of a sigmoid activation function

1. It is differentiable at each point hence the weights can be arrived at during back propagation

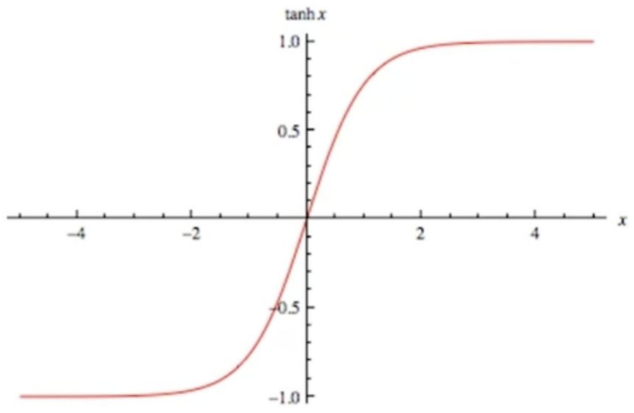
2. The output values are capped between 0 to 1

Disadvantage

1. It is not zero centered – this takes a longer time to converge.

2. For a very high value or low value of input the neuron saturates and hence zero gradient.

Tanh Activation Function



Advantage of a tanh activation function

1. It is differentiable at each point hence the weights can be arrived at during back propagation

2. The output values are capped between -1 to 1

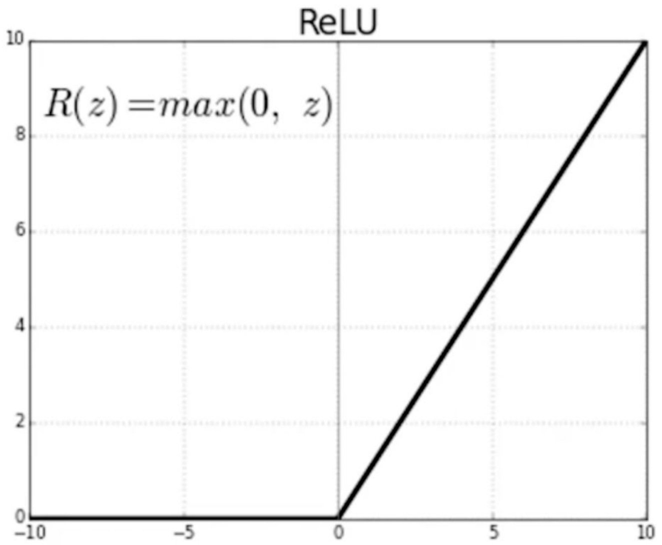
3. It is zero centered — this ensures that it takes less time to converge.

Disadvantage

1. For a very high value or low value of input the neuron saturates and hence zero gradient.

2. It is more expensive computationally

ReLU Activation Function



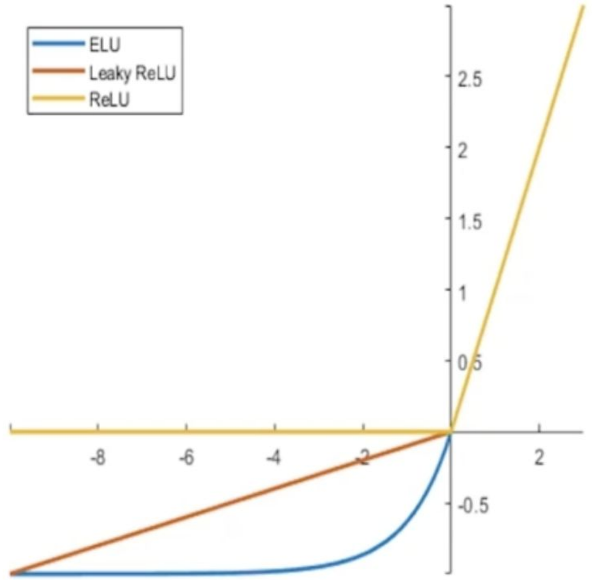
Advantage of a ReLU activation function

1. It is differentiable at positive side
2. Computationally efficient
3. Converges much faster than sigmoid and tanh
4. It does not saturate in the higher region, it saturates for low values

Disadvantage

1. Saturates at lower value. The neurons dies – dead neuron at lower values. To overcome this problem a Leaky neuron is proposed

Exponential or Leaky ReLU Activation Function



It is pretty much similar to ReLU with the advantage that the neuron does not die at the lower values and some gradient stays always

Quiz 3 (Artificial Neural Network)

Go to https://b.socrative.com/login/student/

Room Name: NUSFINTECH

Reminder of Project 2

https://www.kaggle.com/c/nus-fintech-news-headline-sentiment-analysis/overview

News Headline Sentiment Analysis
DDL: 11/14/2020 11:59 PM

THANKS!

And much thanks to these ppl :D

- Adapted from Udemy - Sunil Kumar Mishra