Data Mining :: Unit-3

(Classification – Artificial Neural Network)

Er. Dinesh Baniya Kshatri
(Lecturer)

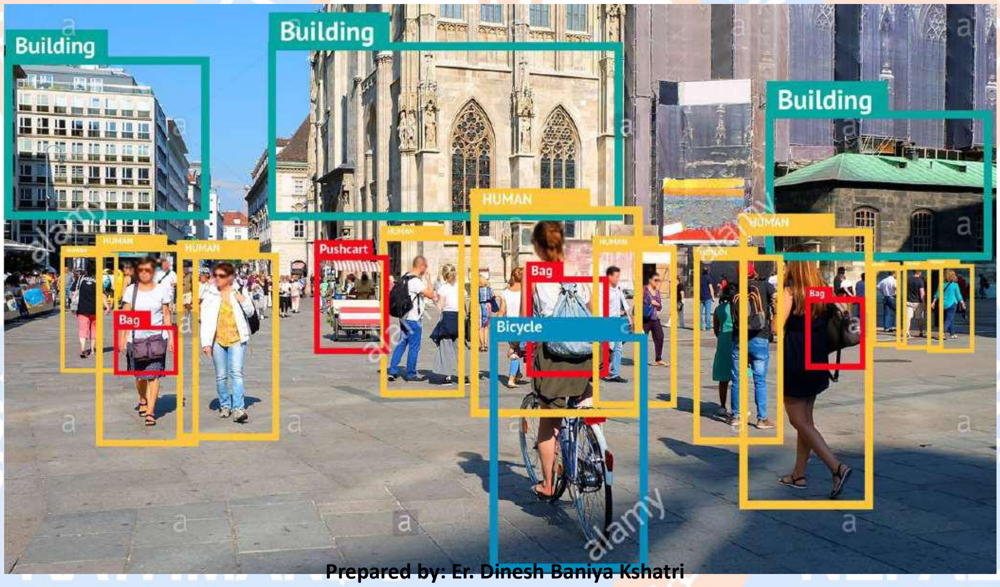
Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus

Breakthroughs with Neural Networks – [1]


Microsoft AI Beats Humans at Speech Recognition

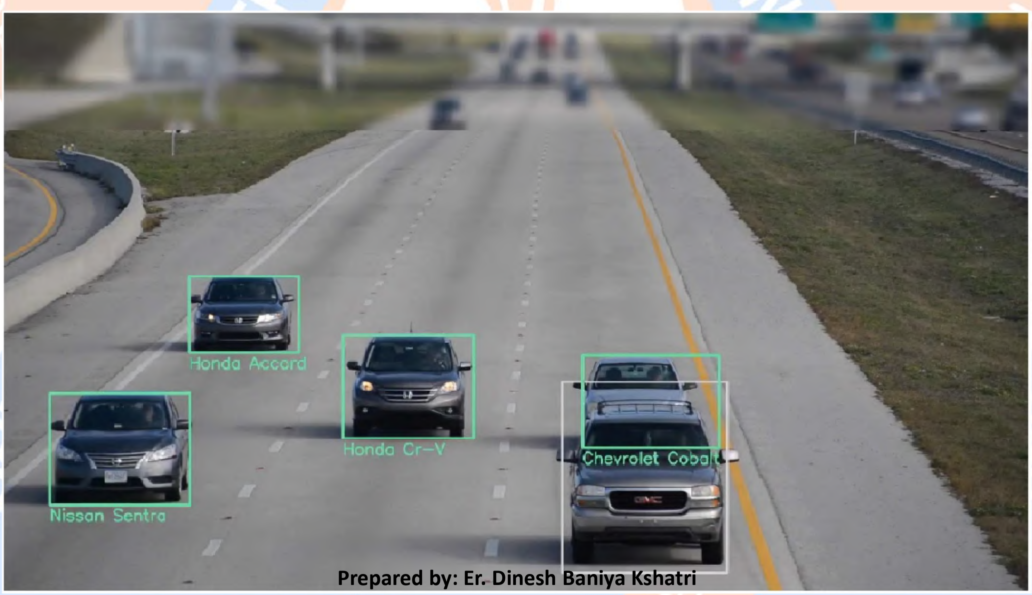
By Richard Adhikari
Oct 20, 2016 11:40 AM PT

Breakthroughs with Neural Networks – [2]



Prepared by: Er. Dinesh Baniya Kshatri 3

Breakthroughs with Neural Networks – [3]



Prepared by: Er. Dinesh Baniya Kshatri 4

Breakthroughs with Neural Networks – [4]

"man in black shirt is playing guitar"

"construction worker in orange safety vest is working on road"

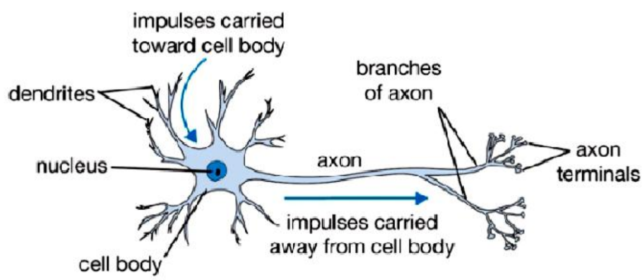
"black and white dog jumps over bar"

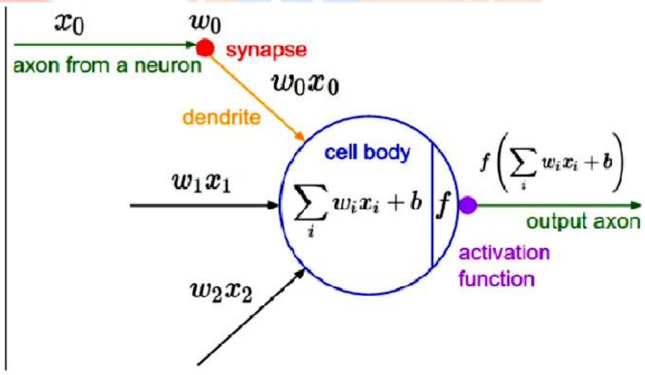
"man in blue wetsuit is surfing on wave"

Computers vs. Human Brain

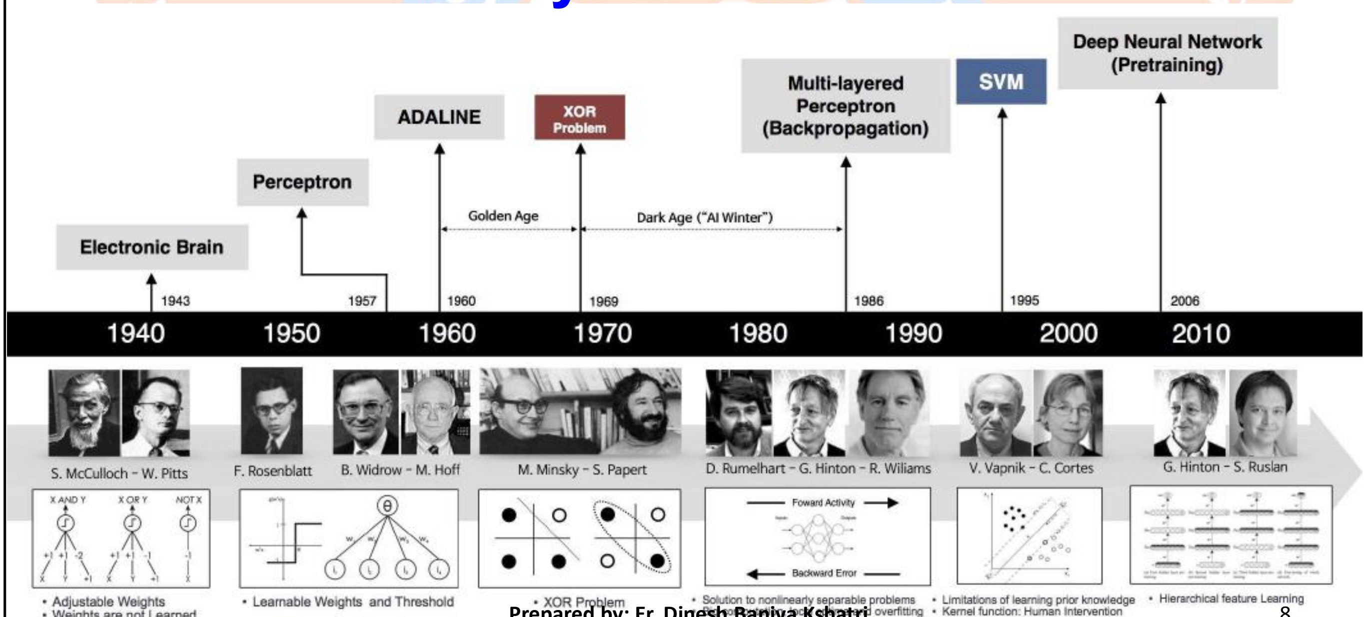
	Personal Computer	Human Brain
processing units	1 CPU, 2–10 cores 10^{10} transistors 1–2 graphics cards/GPUs, 10^3 cores/shaders 10^{10} transistors	10^{11} neurons
storage capacity	10^{10} bytes main memory (RAM) 10^{12} bytes external memory	10^{11} neurons 10^{14} synapses
processing speed	10^{-9} seconds 10^9 operations per second	$> 10^{-3}$ seconds < 1000 per second
bandwidth	10^{12} bits/second	10^{14} bits/second
neural updates	10^6 per second	10^{14} per second

Biological Neuron & Mathematical Model





Brief History of Neural Networks



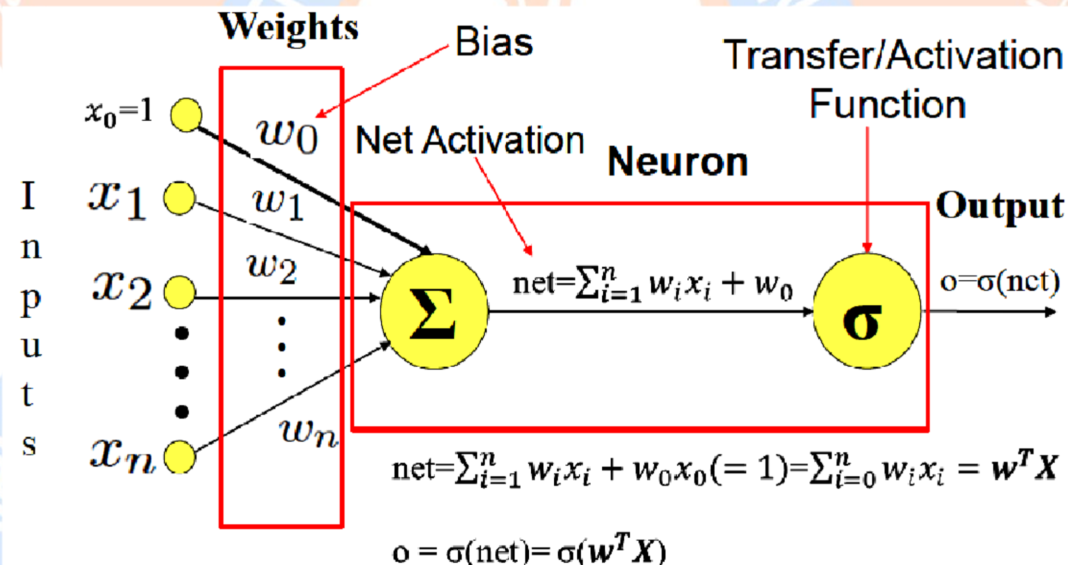
Artificial Neural Network

- Neural Network is a set of connected INPUT/ OUTPUT UNITS, where each connection has a WEIGHT associated with it
- Neural Network learning is also called CONNECTIONIST **learning** due to the connections between units

Artificial Neurons

- **McCulloch & Pitts (1943) are generally recognized as the designers of the first artificial neural network**
- **The Rosenblatt perceptron (1957) is a single layer neural network with a non-linear activation function, the step function**
  - Highly simplified computational model of a neuron

McCulloch-Pitts Neuron – [1]
(Rosenblatt Perceptron has Step Activation)

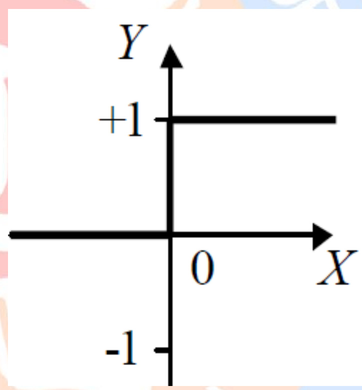


Prepared by: Er. Dinesh Baniya Kshatri 11

McCulloch-Pitts Neuron – [2]
(Rosenblatt Perceptron has Step Activation)

- **Activation function = Step function**

$$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{ if } X < 0 \end{cases}$$



Prepared by: Er. Dinesh Baniya Kshatri 12

Dimensions of Artificial Neural Networks

- **Architecture:**
  - How are the neurons connected ?
- **The Neuron:**
  - How information is processed in each unit?
- **Learning Algorithms:**
  - How a neural network modifies its weights in order to solve a particular learning task using a set of training examples?

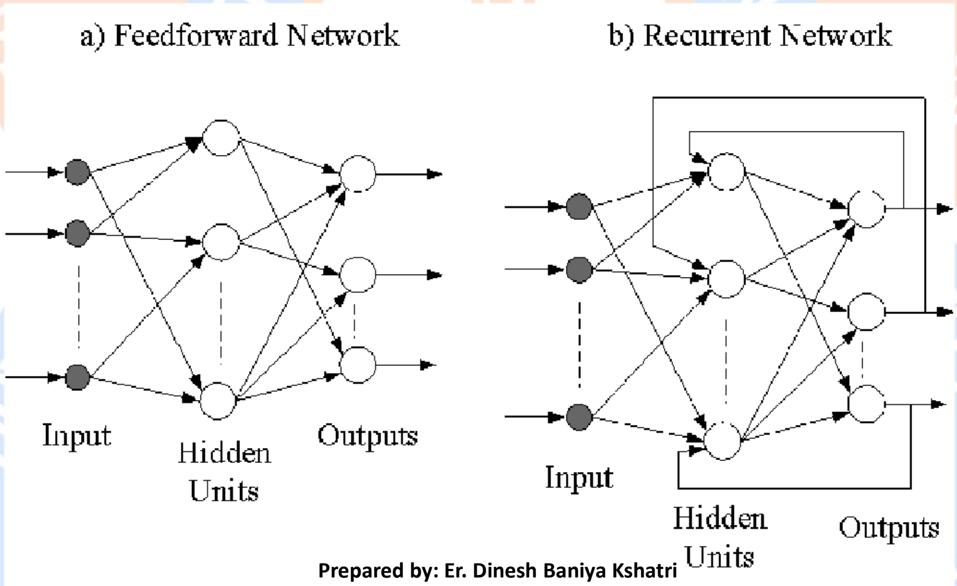
Neural Network Structures – [1]

Feed-forward networks:
- single-layer perceptrons
- multi-layer perceptrons

Feed-forward networks implement functions, have no internal state

Recurrent networks:
- recurrent neural nets have directed cycles with delays
  - $\Rightarrow$ have internal state (like flip-flops), can oscillate etc.

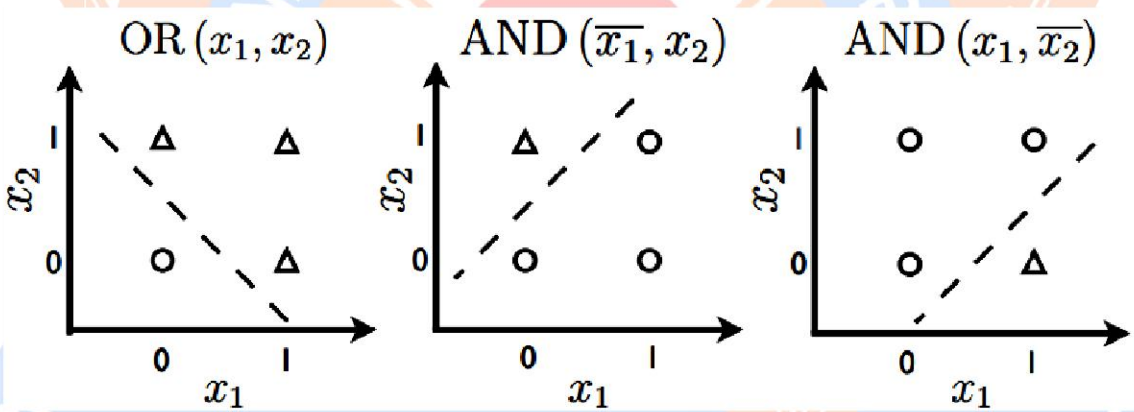
Neural Network Structures – [2]



Prepared by: Er. Dinesh Baniya Kshatri 15

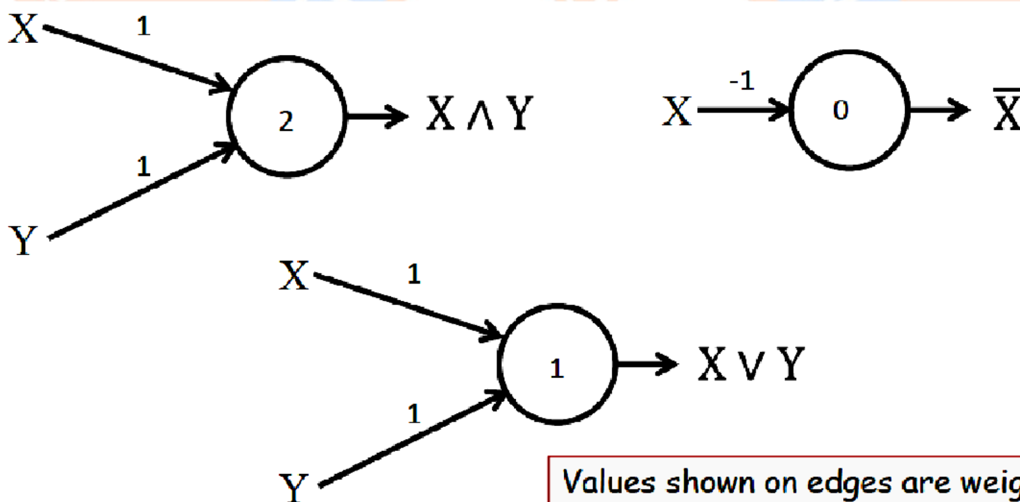
Capacity of a Single Neuron – [1]

- **Can solve linearly separable problems**



Prepared by: Er. Dinesh Baniya Kshatri 16

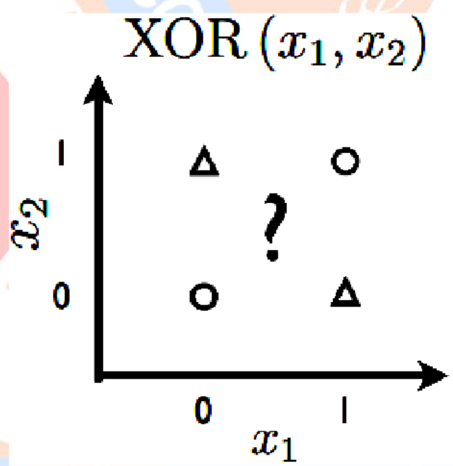
Capacity of a Single Neuron – [2]



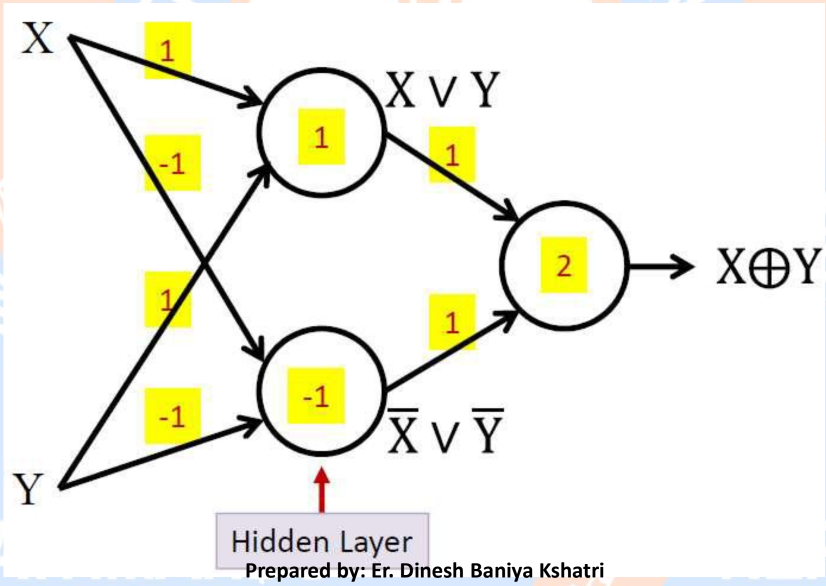
Values shown on edges are weights, numbers in the circles are thresholds

Capacity of a Single Neuron – [3]

- **Cannot solve non-linearly separable problems**
- **A single neuron is not enough**
- **Networked elements are required**



Solution to XOR Problem
(Multi-layer Perceptron)



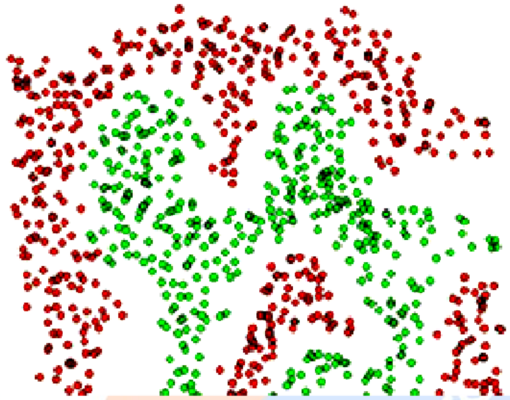
Activation Functions

- **Regulate the outputs of each layer of a neural network**
- **Determine whether a neuron is "fired" or not**
- **Add a level of complexity that neural networks without activation functions cannot achieve**

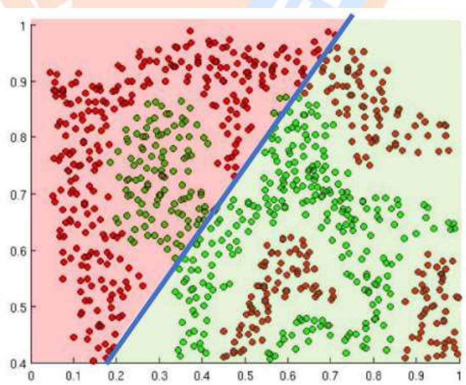
Need for Activation Functions – [1]

- The purpose of activation functions is to introduce non-linearity into the network

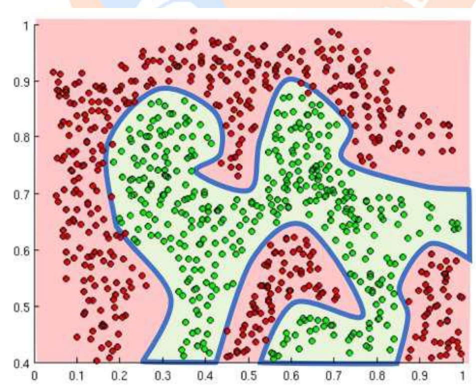
- In the figure, how is it possible to distinguish between green and red points?



Need for Activation Functions – [2]



Linear activation functions produce linear decisions



Non-linearity allows us to approximate arbitrarily complex functions

Common Activation Functions – [1]

(Equation Form: Sigmoid & its Derivative)

- **The Sigmoid function:**

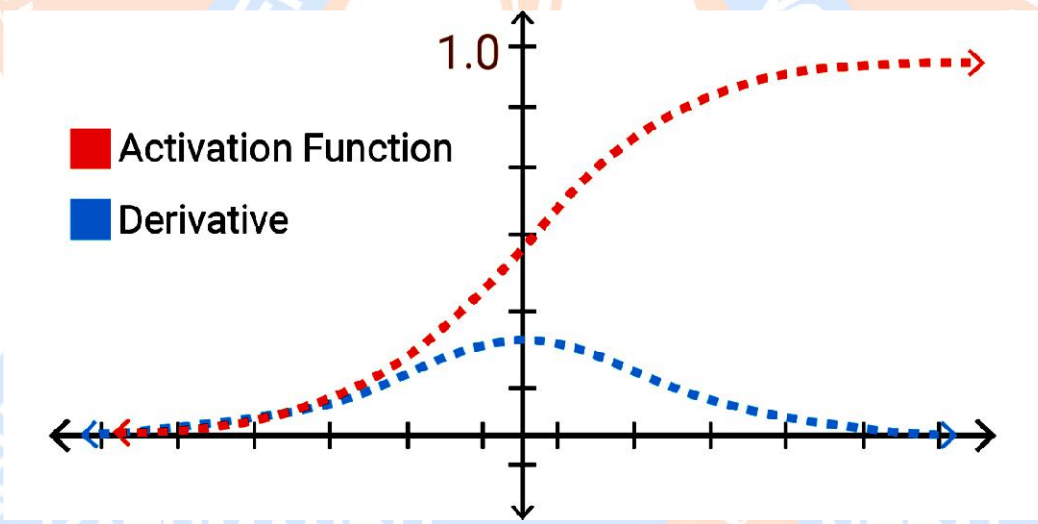
$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

- **Derivative of Sigmoid function:**

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

Common Activation Functions – [2]

(Plot of Sigmoid & its Derivative)



Common Activation Functions – [3]
(Properties of Sigmoid)

- **Advantages:**
  - Is nonlinear in nature
  - Has a smooth gradient
  - Output is bound in the range [0,1]
- **Disadvantages:**
  - Gives rise to a problem called “vanishing gradients”
    - Network refuses to learn further or is drastically slow
  - Its output is not zero centered
    - Makes optimization difficult

Common Activation Functions – [4]
(Equation: Hyperbolic Tangent & its Derivative)

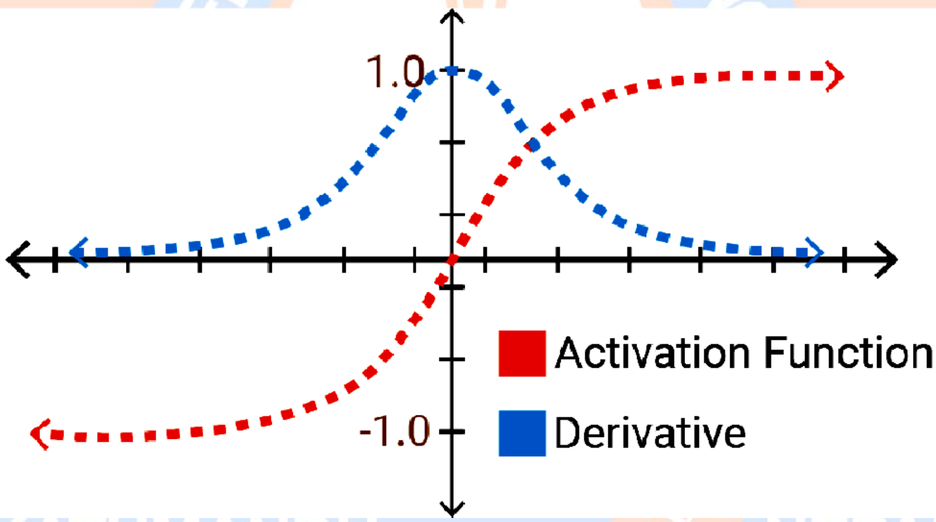
- **The Tanh function:**

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \left(= \frac{sinh(x)}{cosh(x)}\right) \quad \left(= \frac{2}{1+e^{-2x}} - 1\right)$$

- **Derivative of Tanh function:**

$$\frac{d}{dx} f(x) = \frac{d}{dx} tanh(x) = \frac{4}{(e^{-x} + e^{x})^2}$$

Common Activation Functions – [5]
(Plot of Tanh & its Derivative)



Common Activation Functions – [6]
(Properties of Tanh)

- **Advantages:**
  - It is non-linear in nature
  - Its output is zero-centered
  - The gradient is stronger for Tanh than Sigmoid
    - Derivatives are steeper
    - Able to differentiate between similar situations better
  - Output is bound in the range [-1, 1]
- **Disadvantages:**
  - Tanh suffers from "vanishing gradient" problem

Common Activation Functions – [7]
(Equation Form: Rectified Linear Unit – ReLU)

- **The ReLU function:**

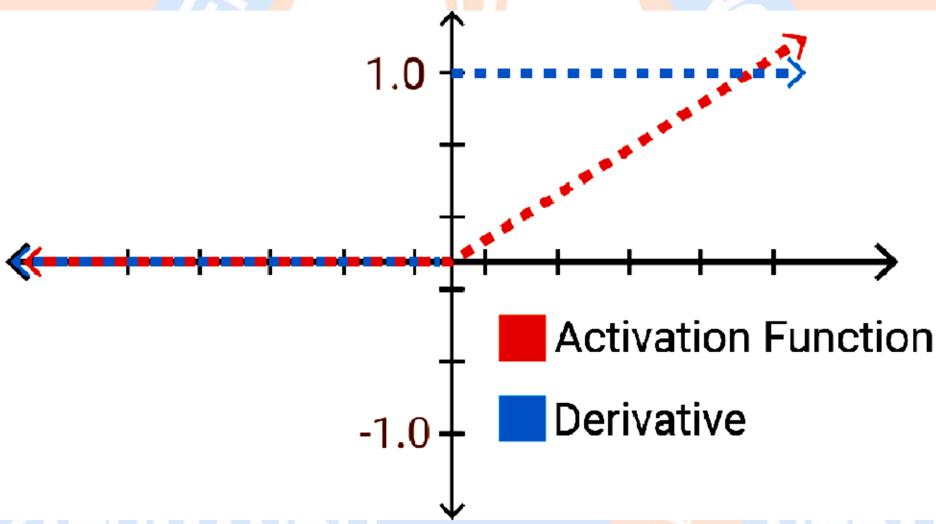
$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \quad (= max(x, 0))$$

- **Derivative of ReLU function:**

$$\frac{d}{dx} f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

Prepared by: Er. Dinesh Baniya Kshatri 29

Common Activation Functions – [8]
(Plot of ReLU & its Derivative)



Prepared by: Er. Dinesh Baniya Kshatri 30

Common Activation Functions – [9]
(Properties of ReLU)

- **Advantages:**
  - It avoids "vanishing gradient" problem in the region (x > 0)
  - Converges much faster than Sigmoid & Tanh
  - Is less computationally expensive than Tanh and Sigmoid
- **Disadvantages:**
  - For activations in the region (x < 0), gradient will be 0
    - Weights will not get adjusted during learning phase
    - This is called "dying" ReLU problem
  - In the range (x > 0), range of ReLU is [0, inf)
    - It can cause the activation to be massive
    - Output is not zero centered

Activation Function
(Equation Form: Leaky ReLU)

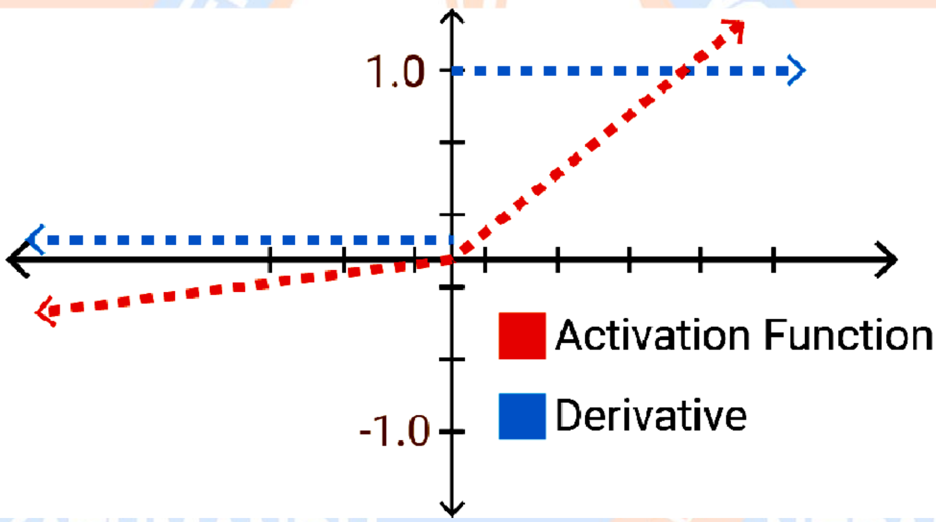
- **The Leaky ReLU function:**

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0.01x, & \text{if } x < 0 \end{cases}$$

- **Derivative of Leaky ReLU function:**

$$\frac{d}{dx} f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0.01, & \text{if } x < 0 \end{cases}$$

Activation Function
(Plot of Leaky ReLU)



Activation Functions
(Properties of Leaky ReLU)

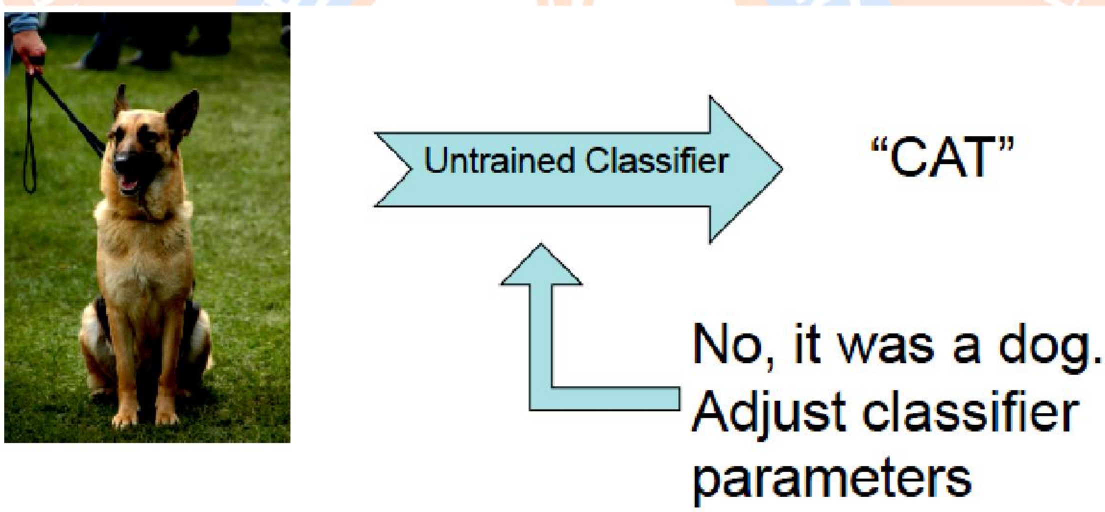
- **Advantages:**
  - Possess all benefits of ReLU
  - Some extra features are:
    - Allows a small, non-zero, constant gradient (normally 0.01) in the region (x < 0)
    - An attempt to fix the “dying” ReLU problem
- **Disadvantages:**
  - Can cause the activation to be massive in the region (x > 0)
  - Output is not zero centered

Neural Network Training

- **Neural Network learns** by adjusting the weights so as to be able to **correctly classify** the training data and hence, after testing phase, to classify unknown data

- **Neural Network** needs long time for training

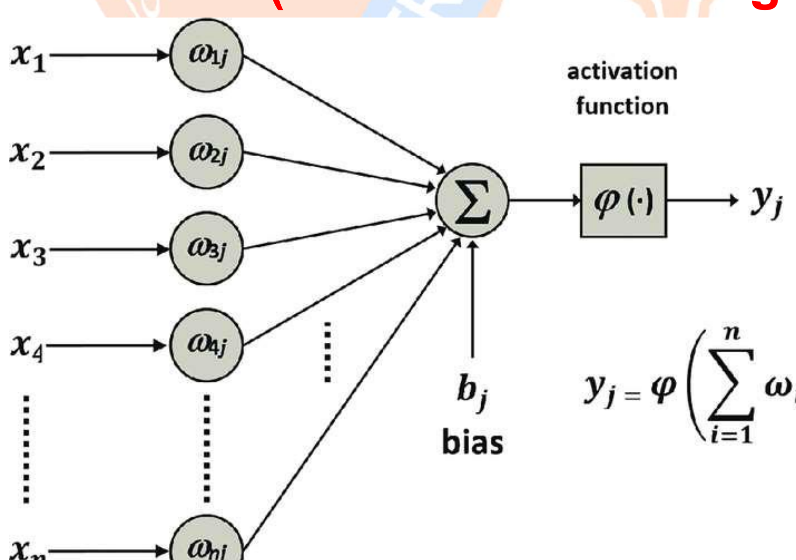
Training Neural Network
(Back-Propagation: Visual Concept)



Brief Overview of Back-Propagation

- **Back-propagation is a neural network training algorithm**
- **Neural network learns by iteratively processing a set of training data**
  - Compares the network's prediction with the actual known target
    - The target values maybe the known class label of the training tuple, or a continuous value for prediction
  - Weights associated with connections are updated till prediction matches actual target

Back-Propagation
(Mechanism of Weight Updates)



Learning rate Input

$$\Delta w_{ij} = \eta \cdot (t_j - y_j) \cdot x_i$$

Desired output Actual output Error

$$y_j = \varphi\left(\sum_{i=1}^{n} \omega_{ij}\, x_i + b_j\right)$$

Back-Propagation Algorithm – [1]

- **Step – 1:**
  - Initialize the weights and biases
    - The weights in the network are initialized to small random numbers
  - Each unit has a BIAS associated with it
    - The biases are also initialized to small random numbers
    - Allows to control the behavior of a neural network layer

Prepared by: Er. Dinesh Baniya Kshatri 39

Back-Propagation Algorithm – [2]

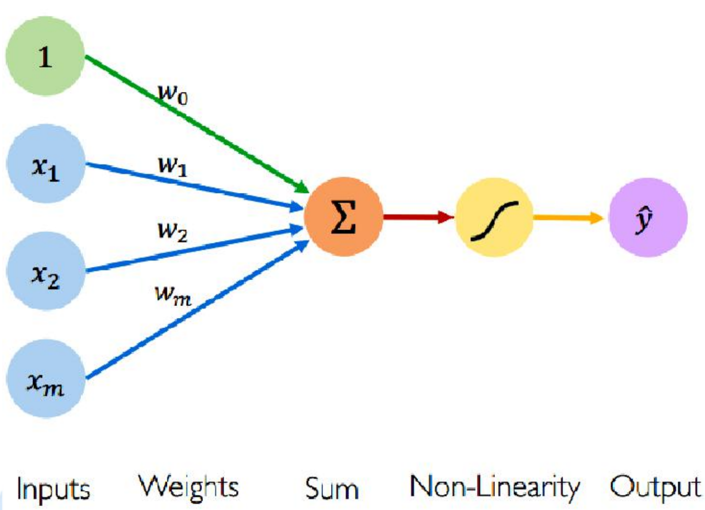
- **Step – 2:**
  - Feed the training samples to the neural network
- **Step – 3:**
  - Propagate the inputs forward by applying the activation function
- **Step – 4:**
  - Back-propagate the error

Prepared by: Er. Dinesh Baniya Kshatri 40

Back-Propagation Algorithm – [3]

- **Step – 5:**
  - Update weights and biases to reflect the propagated errors
- **Step – 6:**
  - Repeat and apply terminating conditions

Example: Forward Propagation – [1]



Output

Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i\right)$$

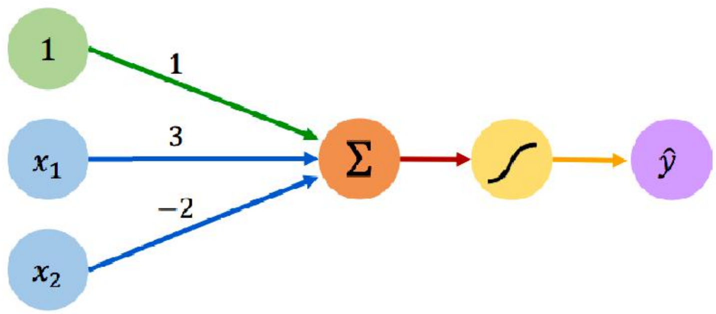
Non-linear activation function

Bias

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W}\right)$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

Example: Forward Propagation – [2]



We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g\left(w_0 + \boldsymbol{X}^T \boldsymbol{W} \right)$$
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$
$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

Example Problem – [1]

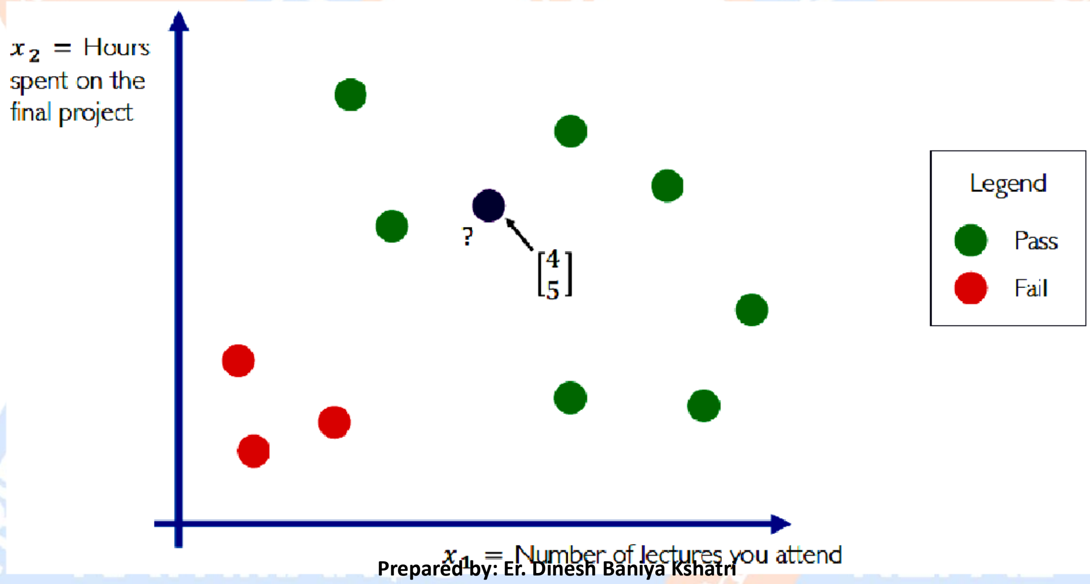
Will I pass this class?

Let's start with a simple two feature model

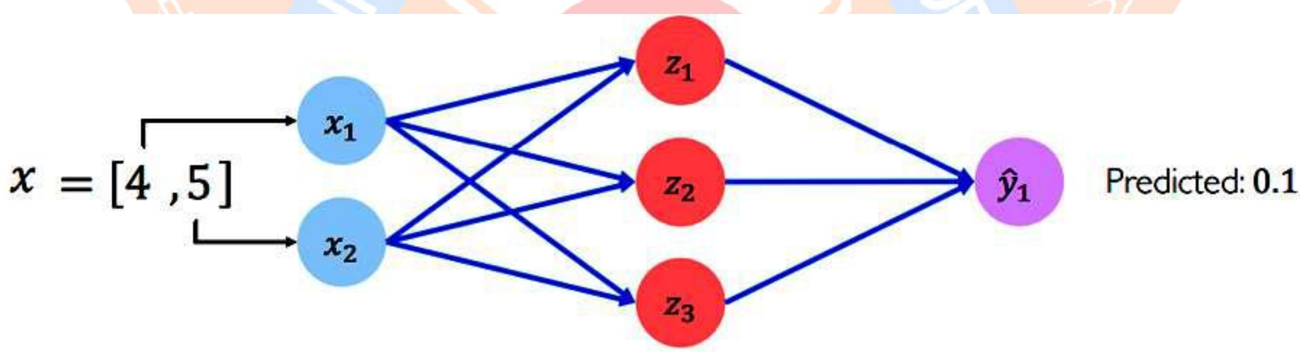
x_1 = Number of lectures you attend

x_2 = Hours spent on the final project

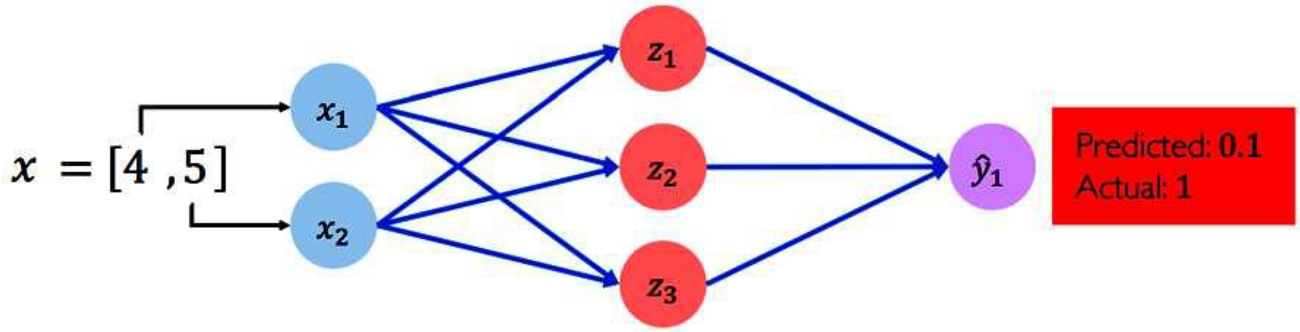
Example Problem – [2]



Example Problem – [3]

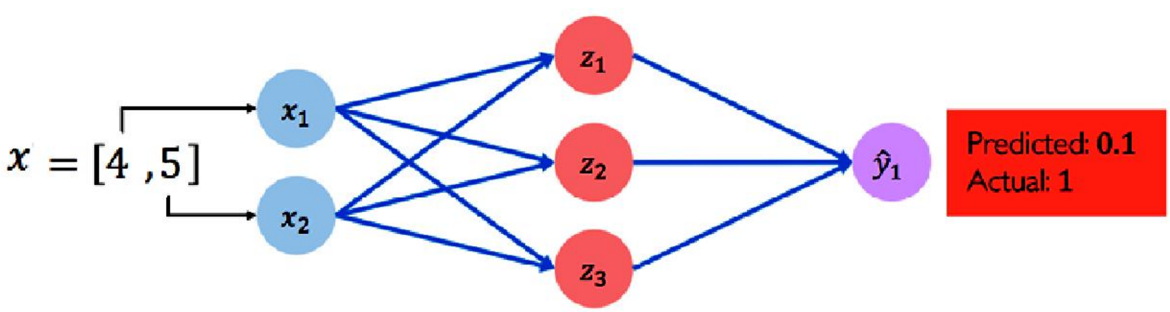


Example Problem – [4]



Example Problem – [5]

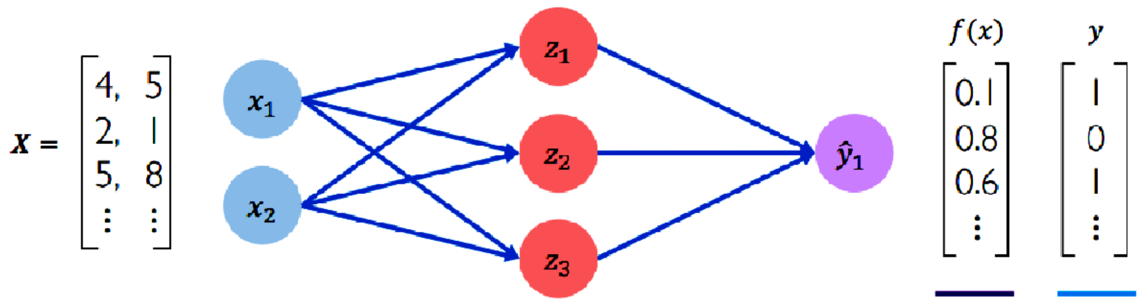
*The **loss** of our network measures the cost incurred from incorrect predictions*



$$\mathcal{L}\big(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\big)$$

Example Problem – [6]

The **empirical loss** measures the total loss over our entire dataset



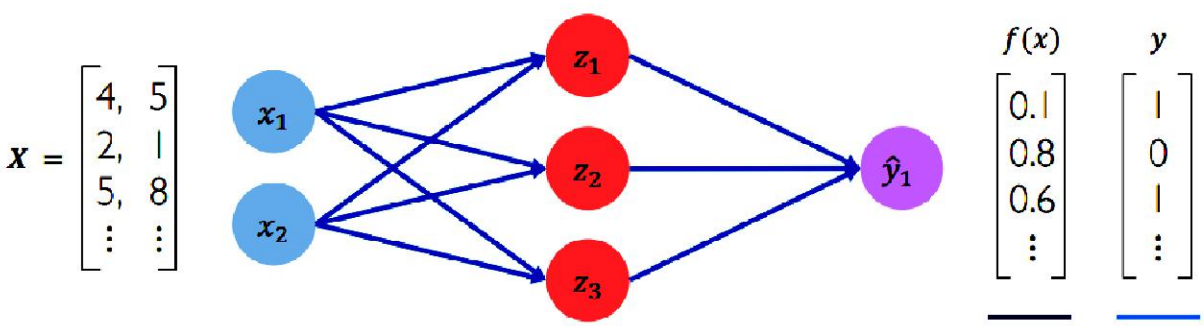
Also known as:

- Objective function
- Cost function
- Empirical Risk

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\big)$$

Binary Cross Entropy Loss

Cross entropy loss can be used with models that output a probability between 0 and 1

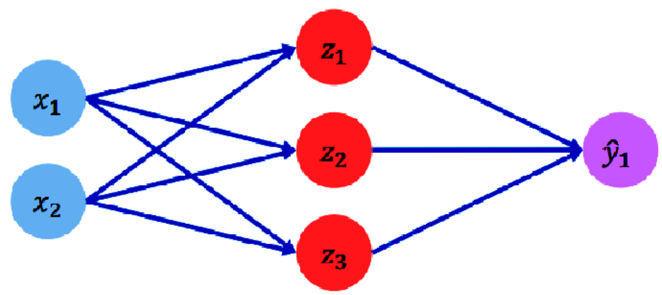


$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + \big(1-\underbrace{y^{(i)}}_{\text{Actual}}\big)\log\Big(1-\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*

$$\boldsymbol{X} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



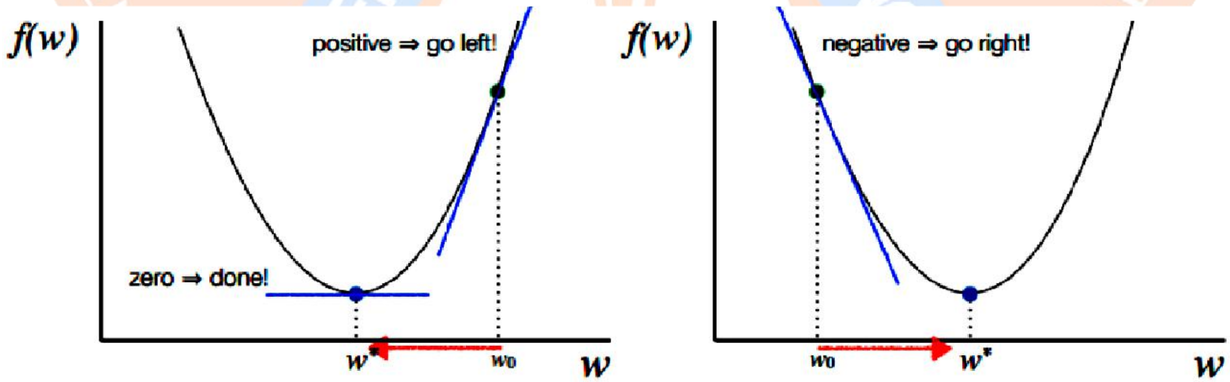
$$\underbrace{\begin{array}{cc} f(x) & y \\ \begin{bmatrix} 30 \\ 80 \\ 85 \\ \vdots \end{bmatrix} & \begin{bmatrix} 90 \\ 20 \\ 95 \\ \vdots \end{bmatrix} \end{array}}_{\text{Final Grades (percentage)}}$$

$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n}\Big(\underbrace{y^{(i)}}_{\text{Actual}} - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)^2$$

Loss Optimization – [1]
(Concept of Gradient Descent)

- **Want to find the network weights that achieve the lowest loss**
- **Calculate the derivative of the cost function with respect to each weight**
- **Move in the direction towards the minimum of the cost function**

Loss Optimization – [2]
(Concept of Gradient Descent)



Step Size

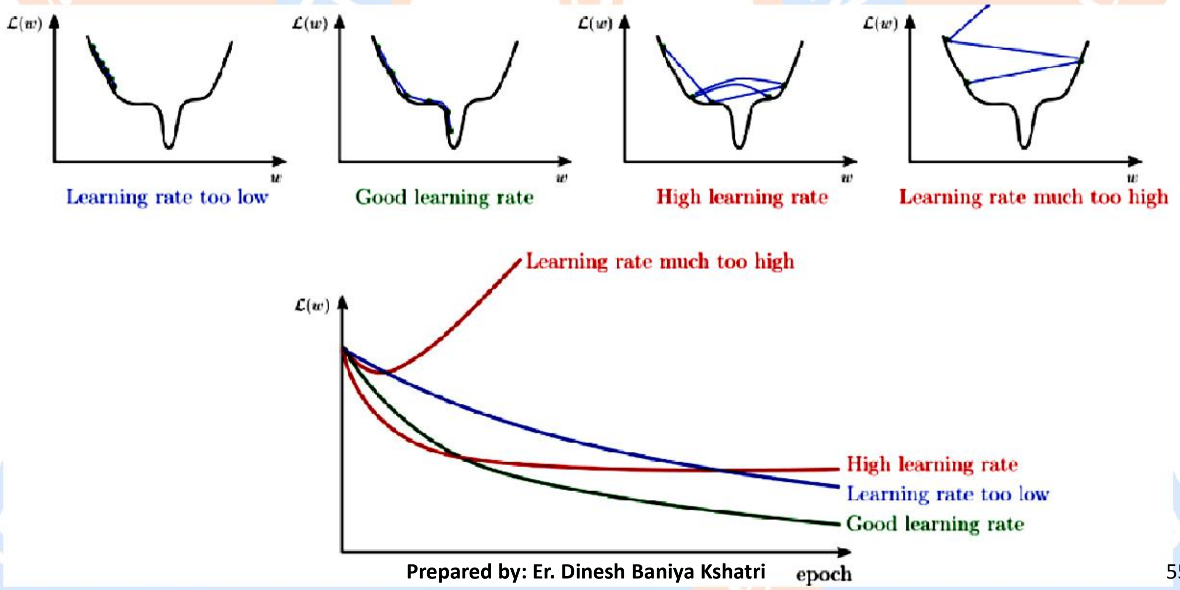
Update Rule: $w_{i+1} = w_i - \alpha_i \frac{df}{dw}(w_i)$

Negative Slope

Loss Optimization – [3]
(Concept of Gradient Descent)

- **If the current point of the slope (gradient) is positive, then choose the direction to the left**
- **If the current slope (gradient) is negative, then choose the direction to the right**
- **The negative slope gives the direction of descent**

Loss Optimization – [4]
(Effect of Learning Rate)



Side Note: Background Material – [1]
(Epoch vs. Batch vs. Iteration)

- **Epoch:**
  - When an entire dataset is passed forward and backward through a neural network only once
- **Batch:**
  - One epoch is too big to feed to a computer at once
  - Dataset is divided into smaller sets or parts
- **Iteration:**
  - Number of batches needed to complete one epoch

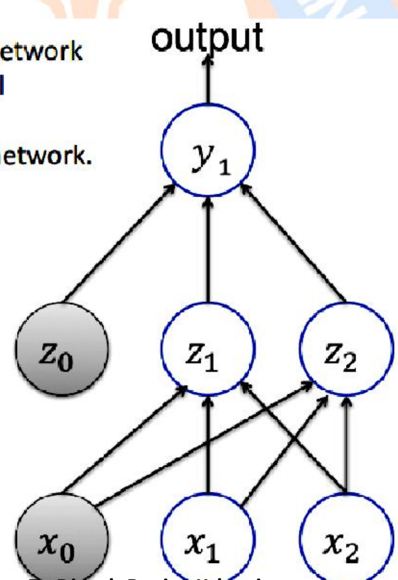
Side Note: Background Material – [2]
(Epoch vs. Batch vs. Iteration)

Let's say we have 2000 training examples that we are going to use .

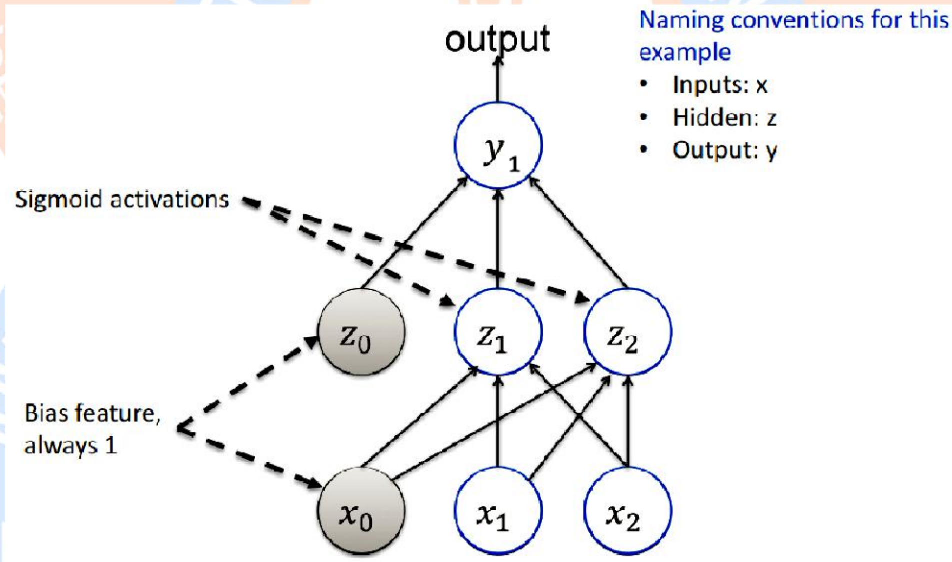
We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.

Example: Neural Network – [1]

We will use this example network as to introduce the general principle of how to make predictions with a neural network.

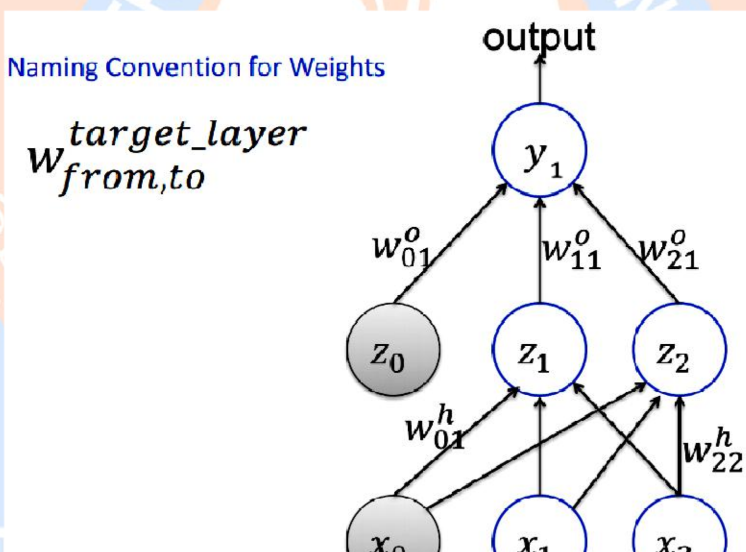


Example: Neural Network – [2]



Prepared by: Er. Dinesh Baniya Kshatri 59

Example: Neural Network – [3]

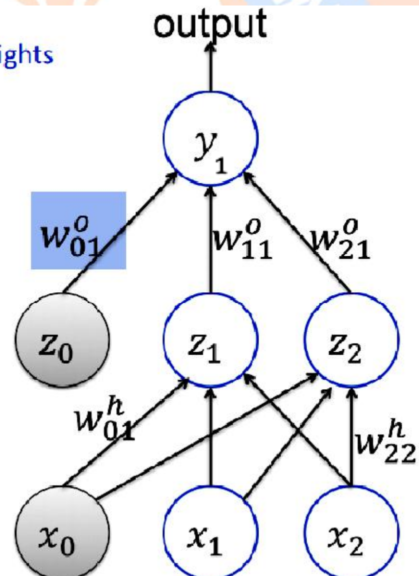


Prepared by: Er. Dinesh Baniya Kshatri 60

Example: Neural Network – [4]

Naming Convention for Weights

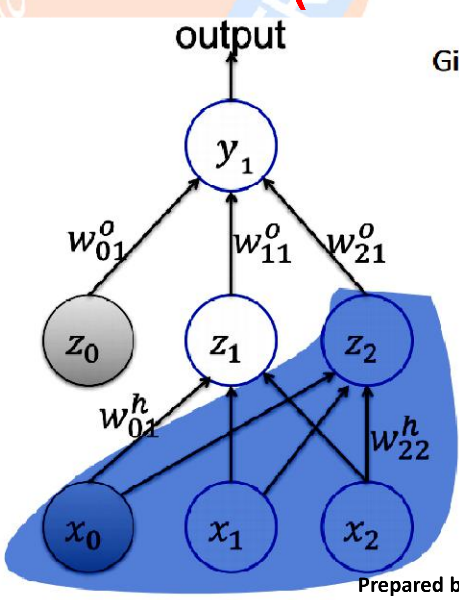
$$w_{from,to}^{target\_layer}$$



$$w_{01}^{o}$$

From neuron #0
to neuron #1 in
output layer

Example: Neural Network – [5]
(The Forward Pass)



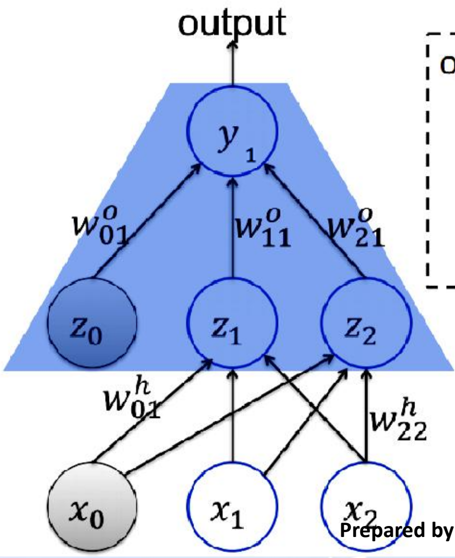
Given an input **x**, how is the output predicted

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

$$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$$

Example: Neural Network – [6]
(The Forward Pass)

Given an input **x**, how is the output predicted



output $y = w^o_{01} + w^o_{11} z_1 + w^o_{21} z_2$

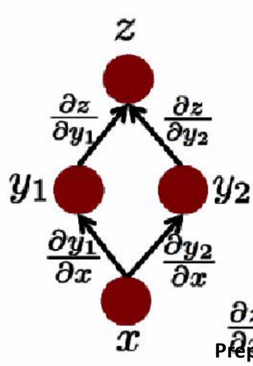
$$z_2 = \sigma(w^h_{02} + w^h_{12} x_1 + w^h_{22} x_2)$$

$$z_1 = \sigma(w^h_{01} + w^h_{11} x_1 + w^h_{21} x_2)$$

In general, before visiting (i.e. computing) the value of a node, visit all nodes that serve as inputs to it.

Side Note: Background Material – [1]
(Chain Rule for Derivatives)

- If z = a function of y_1 + a function of y_2, and the y_i's are functions of x
  - Then z is a function of x, as well
- Question: how to find $\frac{\partial z}{\partial x}$



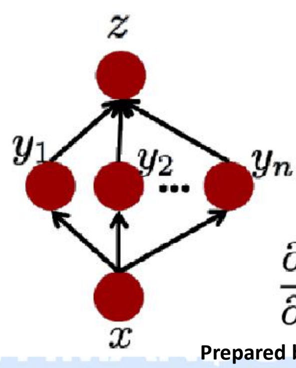
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x}$$

Side Note: Background Material – [2]

(Chain Rule for Derivatives)

– If z is a sum of functions of y_i's, and the y_i's are functions of x
  - Then z is a function of x, as well

– Question: how to find $\frac{\partial z}{\partial x}$

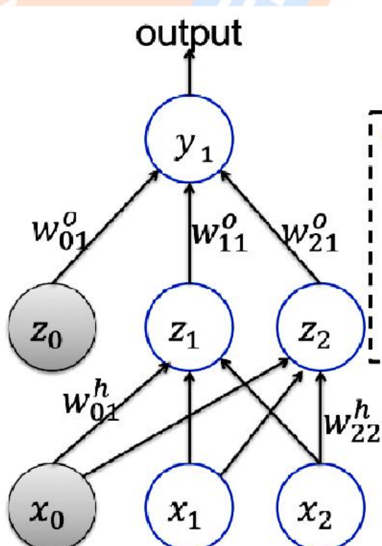


$$\frac{\partial z}{\partial x} = \sum_{i=1}^{n} \frac{\partial z}{\partial y_i} \frac{\partial y_i}{\partial x}$$

Prepared by: Er. Dinesh Baniya Kshatri 65

Example: Neural Network – [7]

(Back-Propagation)



$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w^o_{01} + w^o_{11} z_1 + w^o_{21} z_2$

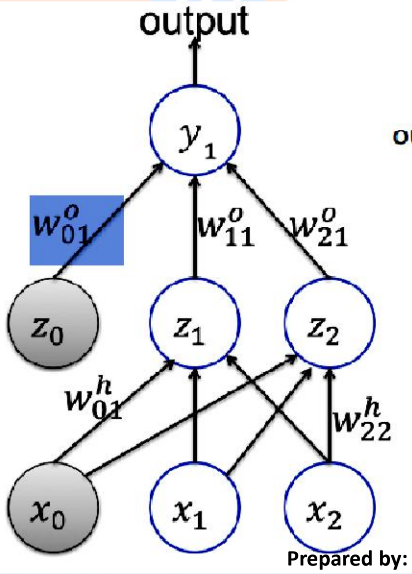
$$z_2 = \sigma(w^h_{02} + w^h_{12} x_1 + w^h_{22} x_2)$$

$$z_1 = \sigma(w^h_{01} + w^h_{11} x_1 + w^h_{21} x_2)$$

We want to compute $\frac{\partial L}{\partial w^o_{ij}}$ and $\frac{\partial L}{\partial w^h_{ij}}$

Prepared by: Er. Dinesh Baniya Kshatri 66

Example: Neural Network – [8]
(Back-Propagation)



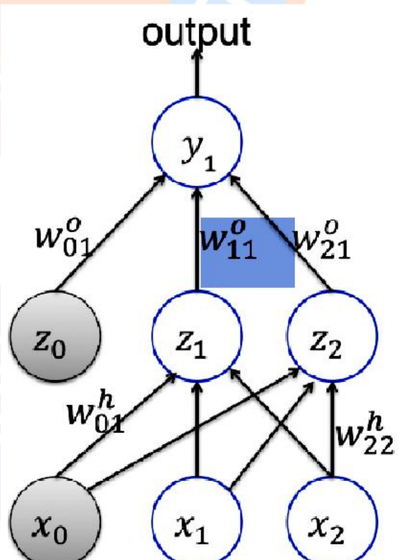
$$L = \frac{1}{2}(y - y^*)^2$$

output $\quad y = w^o_{01} + w^o_{11} z_1 + w^o_{21} z_2$

$$\frac{\partial L}{\partial w^o_{01}} = \boxed{\frac{\partial L}{\partial y}} \boxed{\frac{\partial y}{\partial w^o_{01}}}$$

$$\frac{\partial L}{\partial y} = y - y^* \qquad \frac{\partial y}{\partial w^o_{01}} = 1$$

Example: Neural Network – [9]
(Back-Propagation)



$$L = \frac{1}{2}(y - y^*)^2$$

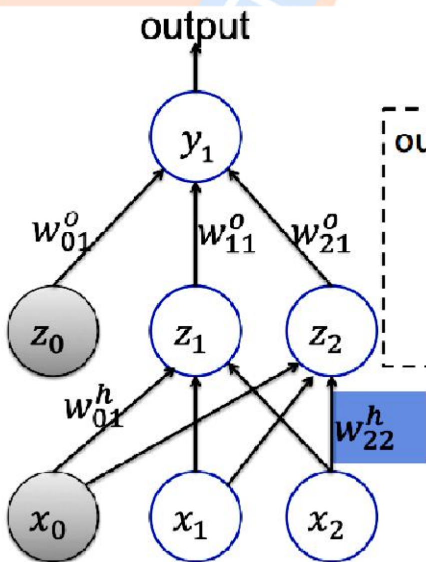
output $\quad y = w^o_{01} + w^o_{11} z_1 + w^o_{21} z_2$

$$\frac{\partial L}{\partial w^o_{11}} = \boxed{\frac{\partial L}{\partial y}} \boxed{\frac{\partial y}{\partial w^o_{11}}}$$

$$\frac{\partial L}{\partial y} = y - y^* \qquad \frac{\partial y}{\partial w^o_{11}} = z_1$$

Example: Neural Network – [10]

(Back-Propagation)



$$L = \frac{1}{2}(y - y^*)^2$$

output $y = w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2$

$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

$$z_1 = \sigma(w_{01}^h + w_{11}^h x_1 + w_{21}^h x_2)$$

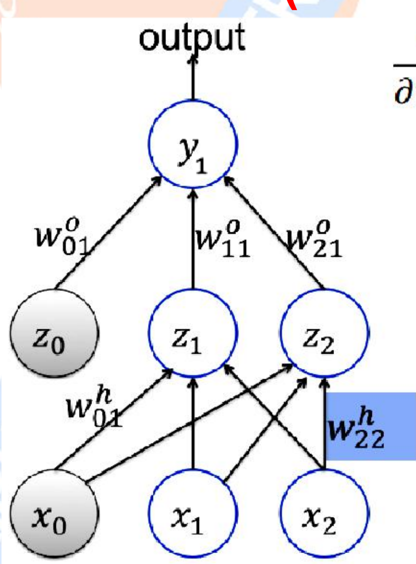
We want $\frac{\partial L}{\partial w_{22}^h}$

Prepared by: Er. Dinesh Baniya Kshatri

69

Example: Neural Network – [11]

(Back-Propagation)



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{22}^h}$$

$$= \frac{\partial L}{\partial y}\frac{\partial}{\partial w_{22}^h}(w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2)$$

$$= \frac{\partial L}{\partial y}(w_{11}^o \underbrace{\frac{\partial}{\partial w_{22}^h} z_1}_{0} + w_{21}^o \frac{\partial}{\partial w_{22}^h} z_2)$$

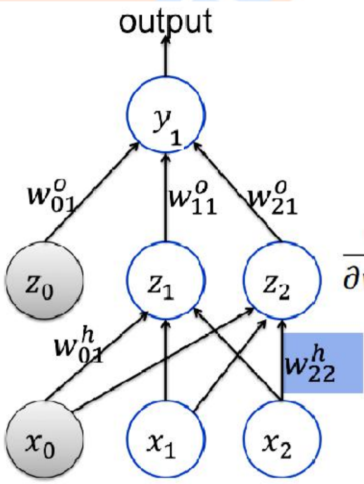
z_1 is not a function of w_{22}^h

$$= \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial w_{22}^h}$$

Prepared by: Er. Dinesh Baniya Kshatri

70

Example: Neural Network – [12]
(Back-Propagation)



$$z_2 = \sigma(w_{02}^h + w_{12}^h x_1 + w_{22}^h x_2)$$

Call this s

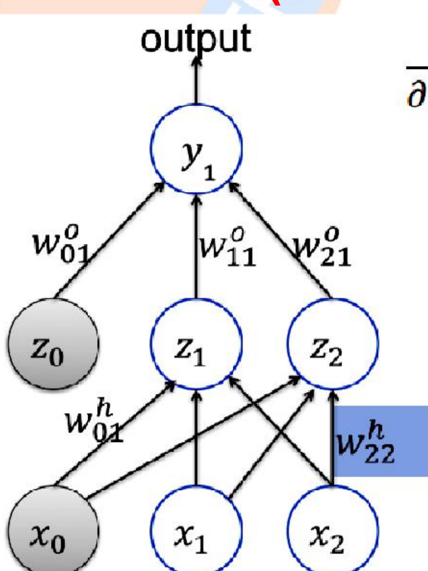
$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{22}^h}$$

$$= \frac{\partial L}{\partial y}\frac{\partial}{\partial w_{22}^h}(w_{01}^o + w_{11}^o z_1 + w_{21}^o z_2)$$

$$= \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial w_{22}^h}$$

$$= \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial s}\frac{\partial s}{\partial w_{22}^h}$$

Example: Neural Network – [13]
(Back-Propagation)



$$\frac{\partial L}{\partial w_{22}^h} = \frac{\partial L}{\partial y} w_{21}^o \frac{\partial z_2}{\partial s}\frac{\partial s}{\partial w_{22}^h}$$

Each of these partial derivatives is easy

$$\frac{\partial L}{\partial y} = y - y^*$$

$$\frac{\partial z_2}{\partial s} = z_2(1 - z_2)$$

Why? Because $z_2(s)$ is the logistic function we have already seen

$$\frac{\partial s}{\partial w_{22}^h} = x_2$$

Example: Neural Network – [14]
(Weight Update)

- **The weight is updated as follows:**

$$w_{22}^{h} = w_{22}^{h} - \eta \frac{\partial L}{\partial w_{22}^{h}}$$

- **In general:**
  - Want to update w_i
  - Update rule: $w_i := w_i - \eta \frac{\partial L}{\partial w_i}$

Prepared by: Er. Dinesh Baniya Kshatri 73

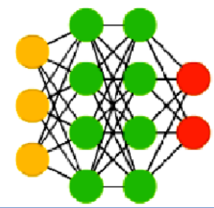
Few Words on DNN and CNN

- **Deep Neural Network (DNN)**
  – Neural network with more than one hidden layer
- **Convolutional Neural Network (CNN)**
  – Consists of convolution and pooling layers
  – Convolution layer convolves an input signal with a filter
  – Pooling down-samples an input representation reducing its dimensionality

Prepared by: Er. Dinesh Baniya Kshatri 74

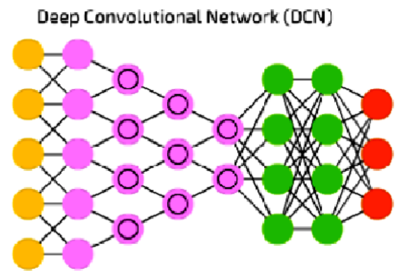
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations



Recurrent Networks

- Keep hidden state
- Have cycles in computational graph