Artificial Neural Networks (ANN)

Machine Learning (CS 306)

Instructor: Dr. Moumita Roy

Teaching Assistants: Indrajit Kalita, Veronica Naosekpam

Email-ids:

moumita@iiitg.ac.in

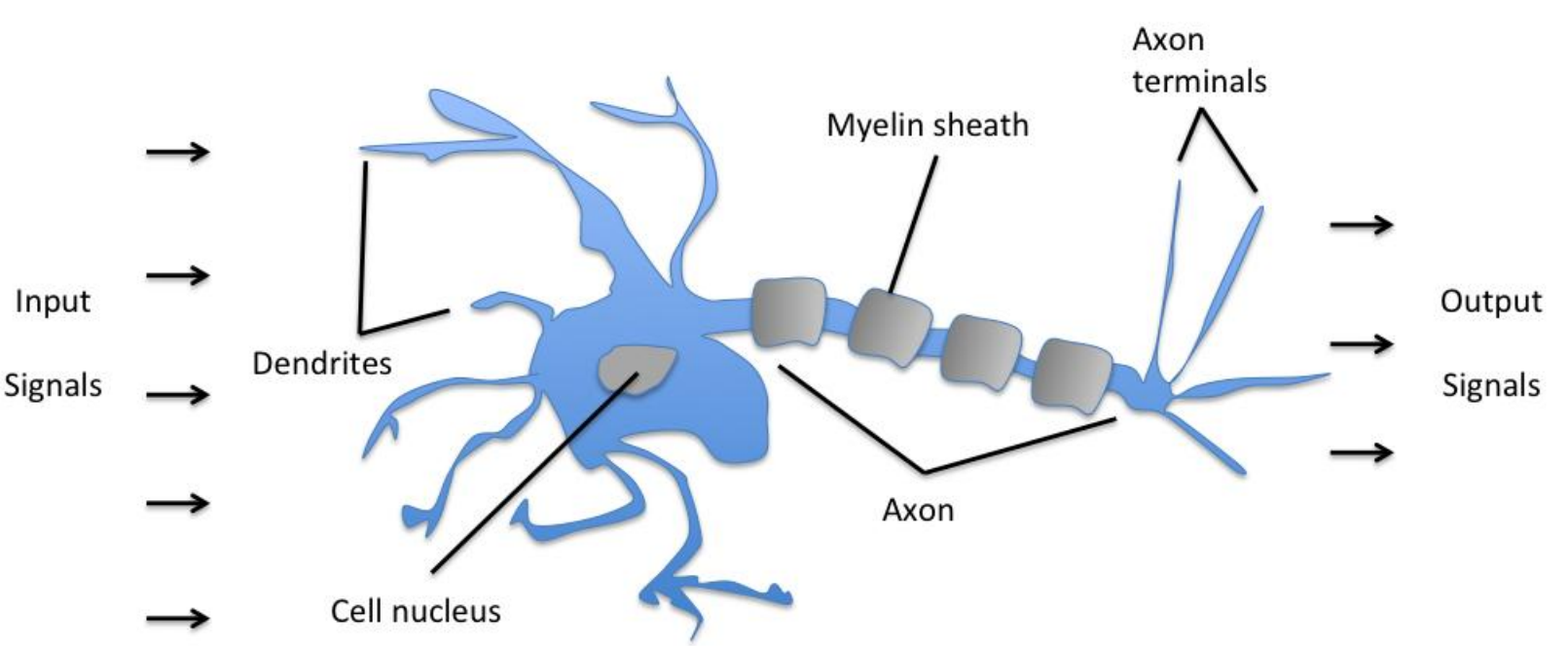
veronica.naosekpam@iiitg.ac.in

indrajit.kalita@iiitg.ac.in

Mobile No: +91-8420489325 (only for emergency quires)

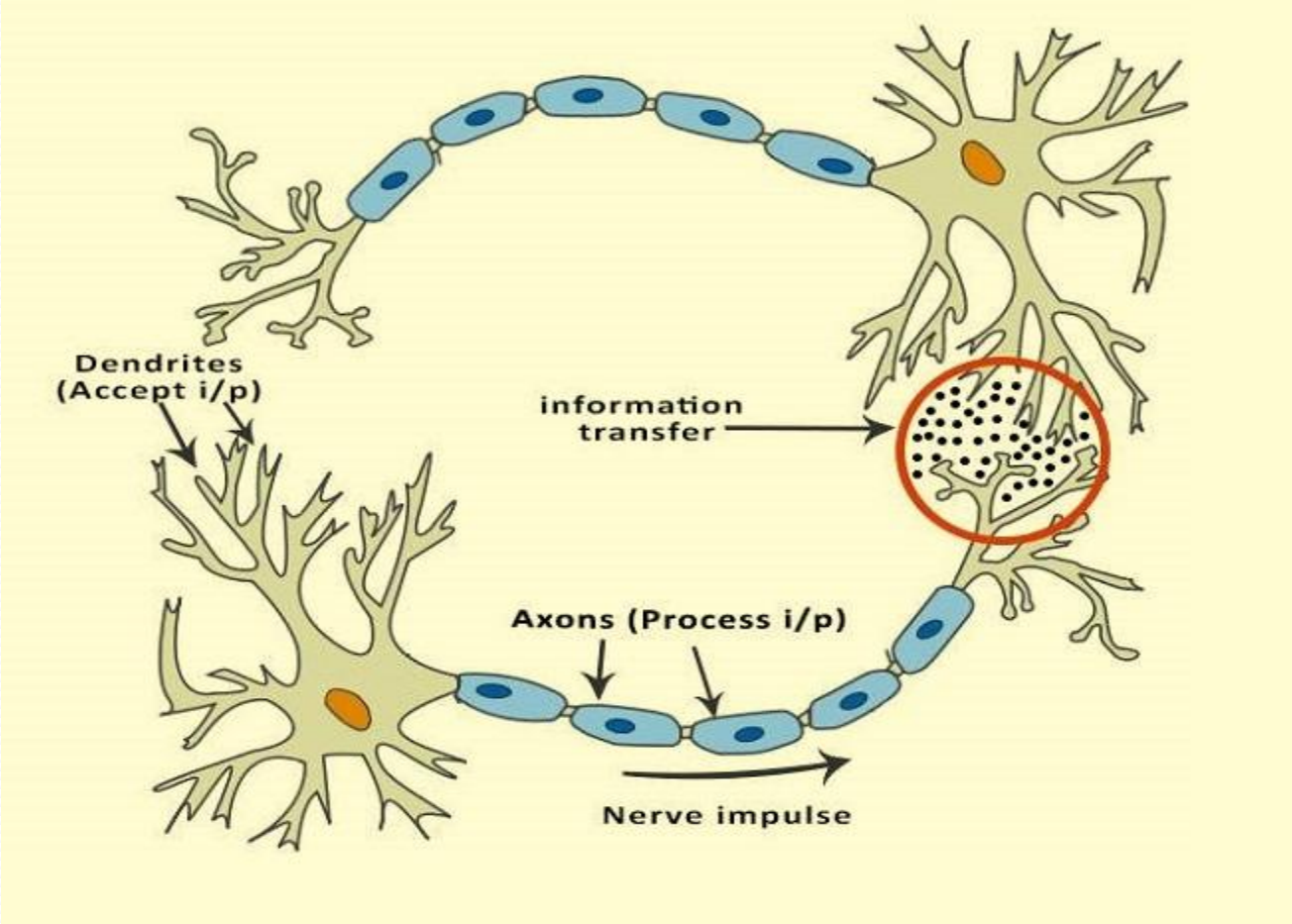
Reference: https://www.cse.iitm.ac.in/~miteshk/CS6910/Slides/Lecture2.pdf

Biological Neuron



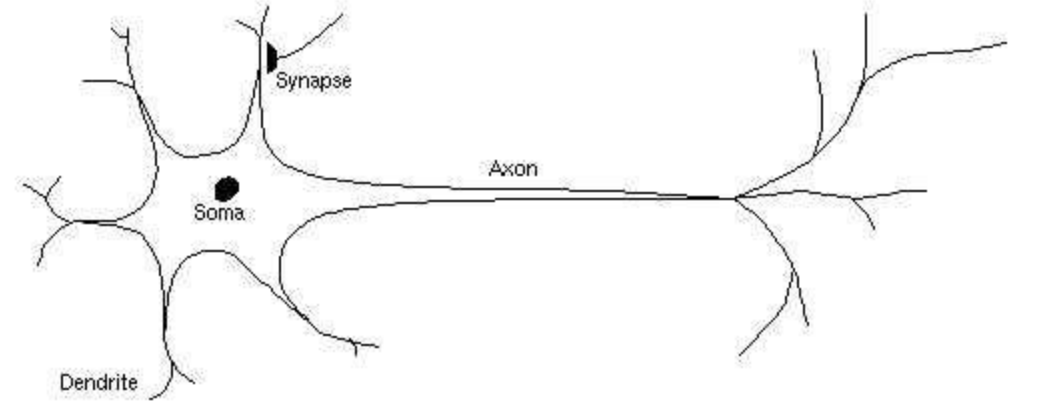
Schematic of a biological neuron.

Biological Neural network



Working principle of biological neuron

- **Dendrite**: receives signals from the axons of other neurons
- **Synapses**: placed between the dendrite and axons, signals are modulated in various amounts (synaptic weights)
- **Soma/Cell body**: Signals from the dendrites are joined and passed on (functional component of the neuron)
- **Axon**: carries nerve impulses away from the cell body. It connects one neuron with other neurons



Formal Definition

A artificial neural network is a massively parallel distributed processor, made up of a simple processing unit, which can store the knowledge and making it available for use.

It resembles the brain in two aspects:

1. Knowledge is acquired by the network from its environment through the learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Characteristic of ANN

- Massive parallel
- Learning ability
- Adaptability
- Distributed representation
- Fault tolerance

McCulloch-Pitts (MP) Neuron (1943)

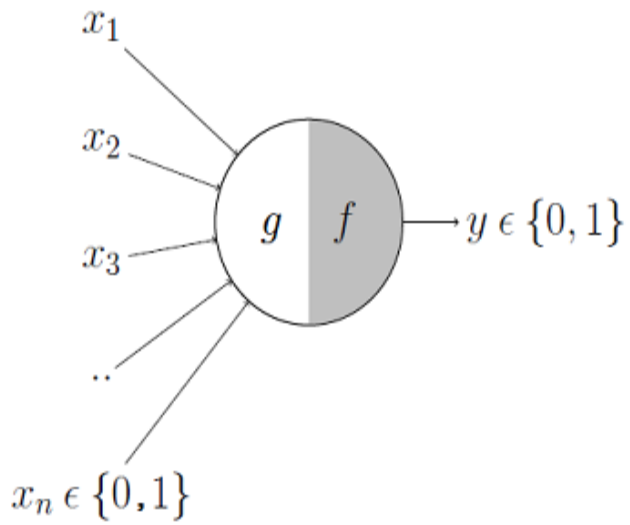
- McCulloch (neuroscientist) and Pitts (logician) proposed a highly simplified computational model of the neuron (1943)
- *g* aggregates the inputs and the function *f* takes a decision based on this aggregation (considering no inputs are inhibitory)

$$g(x_1, x_2, x_3, \cdots, x_n) = g(x) = \sum_{i=1}^{n} x_i$$

$$y = f(g(x)) = 1 \quad \text{if } g(x) \geq \theta$$

$$= 0 \quad \text{if } g(x) < \theta$$

θ = Thresholding parameter

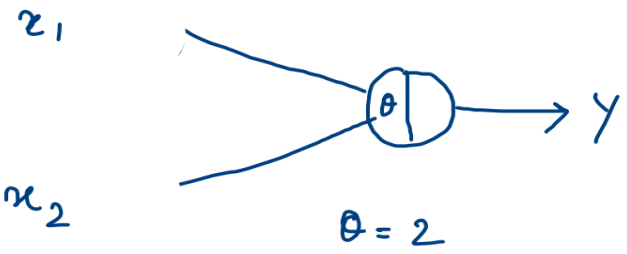


Boolean function representation

AND Function

Truth Table

	x_1	x_2	y
1.	1	0	0
2.	0	1	0
3.	1	1	1
4.	0	0	0



1. $g(x) = 1 + 0 = 1$

   $y = 0 \mid g(x) < \theta$

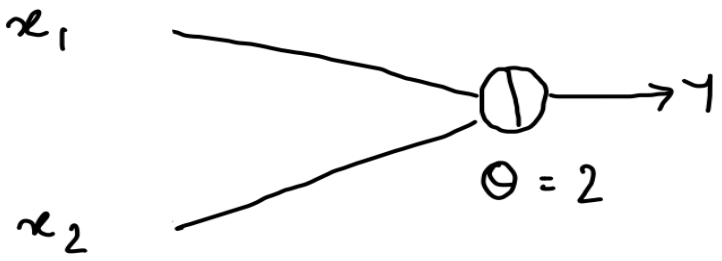
2. $g(x) = 0 + 1 = 1$

   $y = 0 \mid g(x) < \theta$

3. $g(x) = 0 + 0 = 0 \mid y = 0, g(x) < \theta$

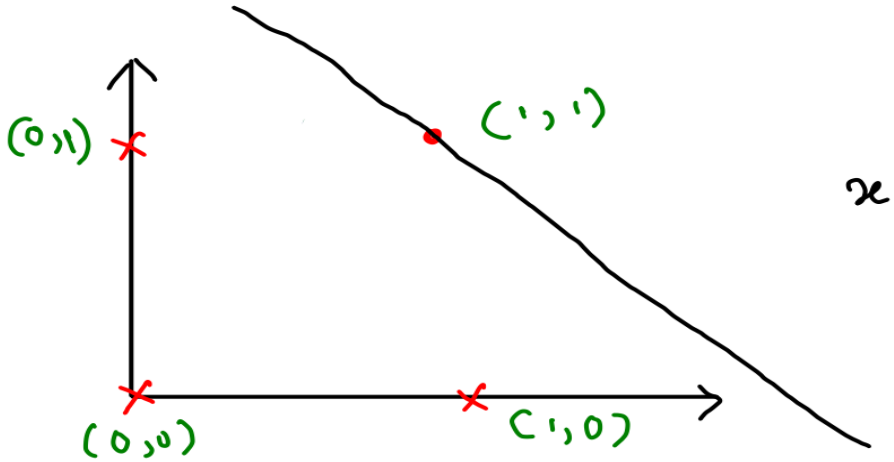
4. $g(x) = 1 + 1 = 2 \mid y = 1, g(x) \geq \theta$

Geometrical interpretation of the model



AND function

$x_1 + x_2 \geq 2 \rightarrow$ then $y = 1$
else $y = 0$



$x_1 + x_2 = \Theta$

$x_1 + x_2 = 2$

Check for other Boolean functions

- OR
- NAND
- NOR

→MP model can be used for linearly separable Boolean function

→What about real inputs?

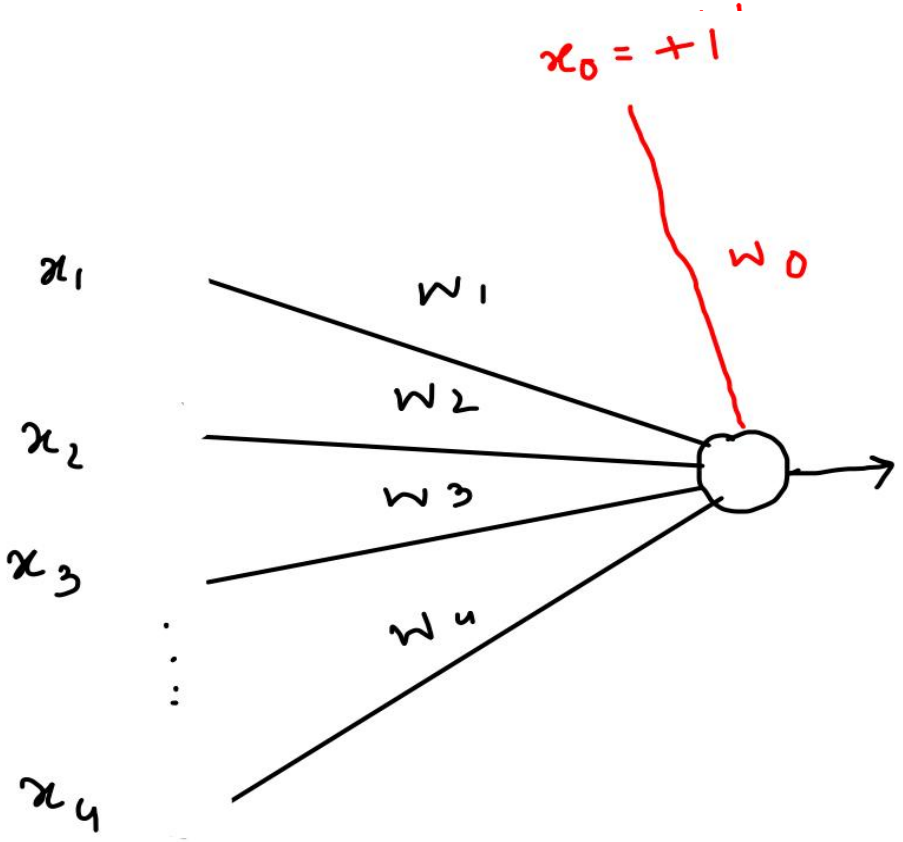
→ Always hand-coded the threshold?

→want to assign more weight (importance) to some inputs?

→functions which are not linearly separable?

Perceptron (Artificial Neuron)

- Frank Rosenblatt, an American psychologist, proposed the classical perceptron model (1958)
- Introduce numerical weights for inputs and a mechanism for learning these weights (same as synaptic weights in biological neuron)
- Inputs are no longer limited to Boolean values
- Refined and carefully analyzed by Minsky and Papert (1969) - their model is referred to as the perceptron model



$y = 1$ if $w_i x_i \geq \theta$

$= 0$ if $w_i x_i < \theta$

Rewrite:

$y = 1$ if $w_i x_i - \theta \geq 0$

$= 0$ if $w_i x_i - \theta < \theta$

Rewrite:

$x_0 = 1$

$w_0 = -\theta$

$$\begin{cases} y = 1 \text{ if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ = 0 \text{ if } \sum_{i=0}^{n} w_i x_i < 0 \end{cases}$$

Observation

- A single perceptron can only be used to implement linearly separable functions (start with binary classification)
- However, the weights (including threshold) can be learned and the inputs can be real valued
- Revisit the Boolean function with perceptron and then start learning of weights

AND function

$y = 1 \; ; \; w^T x \geq 0$

$= 0 \; ; \; w^T x < 0$

x_0	x_1	x_2	y	
1	0	0	0	$\rightarrow w_0 < 0$ — (1) ✓
1	0	1	0	$\rightarrow w_0 + w_2 < 0$ — (2) $-2+1<0$ ✓
1	1	0	0	$\rightarrow w_0 + w_1 < 0$ — (3) $-2+1<0$ ✓
1	1	1	1	$\rightarrow w_0 + w_1 + w_2 \geq 0$ — (4) $-2+1+1 \geq 0$ ✗

Can we have any solution to these inequalities?

$w_0 = -2, \; w_1 = 1, \; w_2 = 1$

Final one, $w^T x = 0$

or $w_0 x_0 + w_1 x_1 + w_2 x_2 = 0$

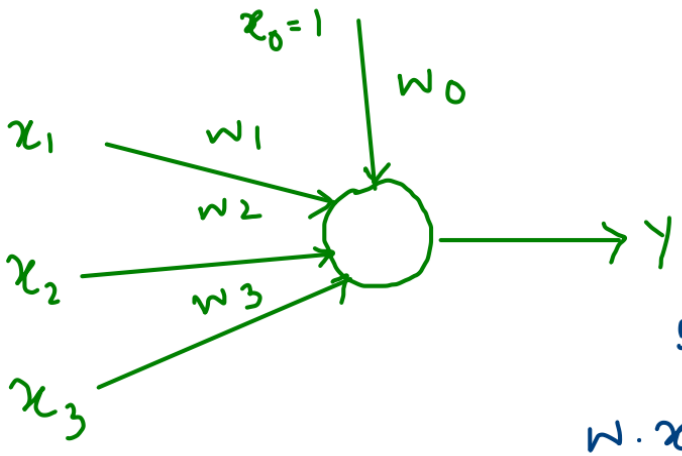
or $x_1 + x_1 = 2$

Same as MP neuron?

Perceptron Learning Algorithm

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while $!convergence$ **do**
    Pick random $\mathbf{x} \in P \cup N$;
    if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
        $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
    end
    if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
        $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
    end
end
//the algorithm converges when all the inputs are classified correctly



$w = [w_0 \; w_1 \; w_2 \; w_3]$

$x = [x_1 \; x_2 \; x_3 \; x_4]$

$w \rightarrow$ randomly initialized

pick any pattern x

$$w \cdot x = w^T x = \sum_{i=0}^{n} w_i x_i$$

we want;

$y = 1$; $w^T x \geq 0$

$y = 0$; $w^T x < 0$

interested to find line

$$w^T x = 0$$

train epoch by epoch until all the training samples are correctly classified

There is a proof of convergence of perceptron learning algorithm.

AND function

	x_0	x_1	x_2	y
TS1	1	0	0	0
TS2	1	0	1	0
TS3	1	1	0	0
TS4	1	1	1	1

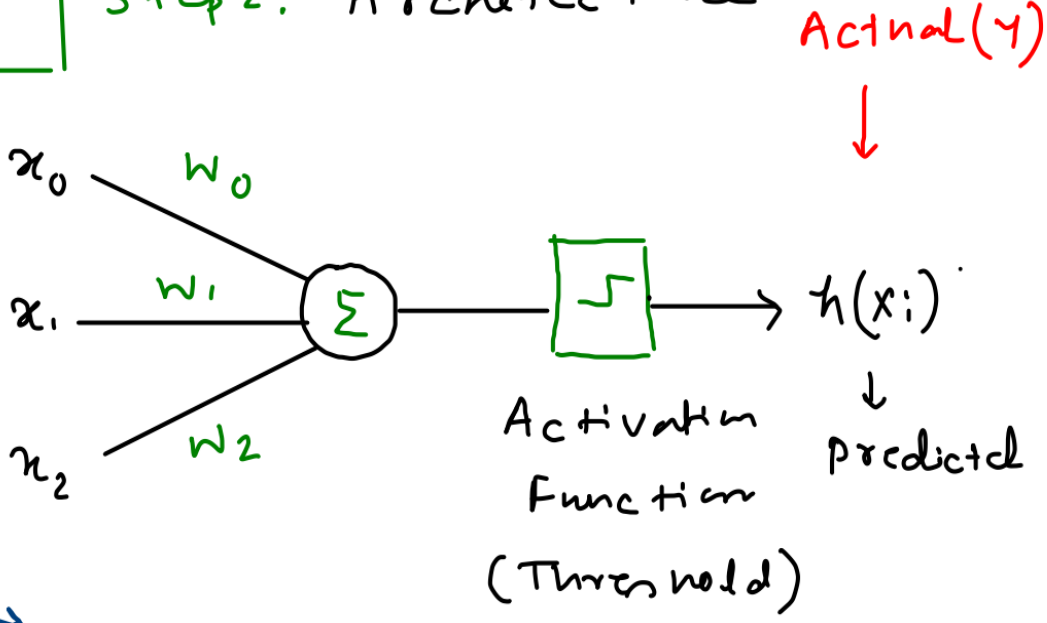
Dataset

Step 1. Randomly initialized weights (Small value)

$W = [W_0 \quad W_1 \quad W_2]$

$= [0.1 \quad 0.1 \quad 0.1]$

Step 2: Architecture



(SLP)

Single perceptron / Single layer Perceptron

Step3: Learning (perceptron learning algorithm) $\alpha = 1$

Epoch 1

Training sample 1:

$Y < 0$ (N)

$$h(TS1) = g(w_0 x_0 + w_1 x_1 + w_2 x_2) > 0$$

$$= g(0.1 \times 1 + 0.1 \times 0 + 0.1 \times 0)$$

$$= g(0.1) = 1$$

$$w_0 = w_0 - x_0 = w_0 + \alpha(y - h(x_i))x_0 = 0.1 - 1$$

$$= w_0 + 1(0 - 1)x_0 = -0.9$$

$$w_1 = w_1 - x_1 = 0.1 - 0 = 0.1$$

$$w_2 = w_2 - x_2 = 0.1 - 0 = 0.1$$

Training sample 2 (∈ N⁻) $y = 0$

$$h(TS2) = g(w_0 x_0 + w_1 x_1 + w_2 x_2) < 0$$

$$= g(-0.9 \times 1 + .1 \times 0 + .1 \times 1)$$

$$= g(-0.9 + .1 + .1) = g(-0.7)$$

No updation

Training sample 3

$h(TS3)$ .....

Training sample 4

...

Epoch 2 :

So on

Continue the process epoch by epoch till convergence (no misclassification)

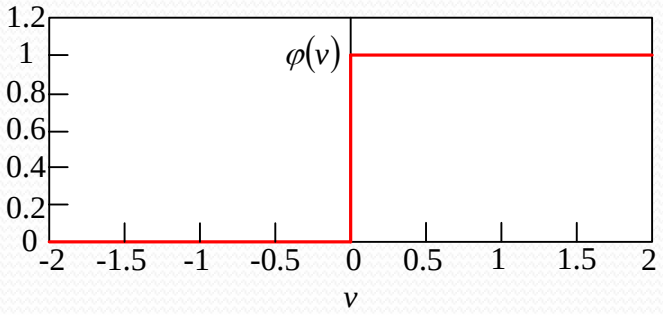
Important Aspects

- Architecture (Feed-forward/signal can move in forward direction)
- Activation function (Threshold)
- Learning process (Perceptron learning algorithm)

Types of Activation function

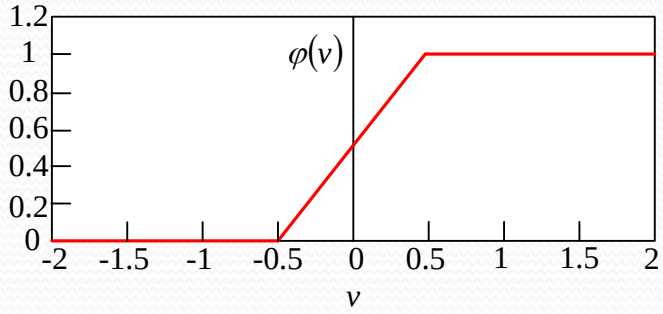
- Threshold function

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases}$$



- Piecewise-Linear Function

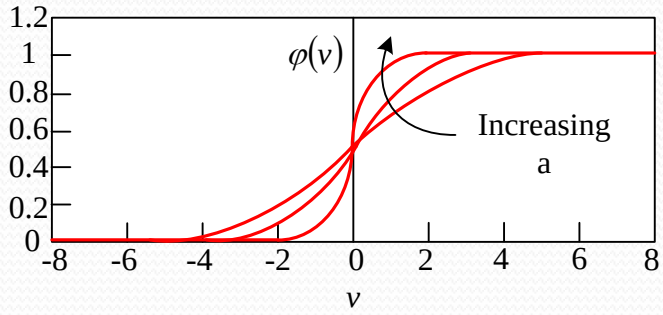
$$\varphi(v) = \begin{cases} 1 & v \geq 1/2 \\ v & -1/2 < v < 1/2 \\ 0 & v \leq -1/2 \end{cases}$$



- Sigmoid Function/logistic function

$$\varphi(v) = \frac{1}{1+\exp(-av)}$$

a is the slope parameter



Sigmoid (logistic) neuron (binary classification/linearly separable)

- A perceptron neuron will fire if the weighted sum of its input is greater than a threshold ($-w_0$)
- The threshold activation function is very harsh (behave like a step function)
- We want some smoother decision function (gradually moves from 0 to 1)
- Introduce sigmoid neuron using one form of the sigmoid function (i.e., logistic function) as activation function/ output function
- Now there is no sharp transition around the threshold.

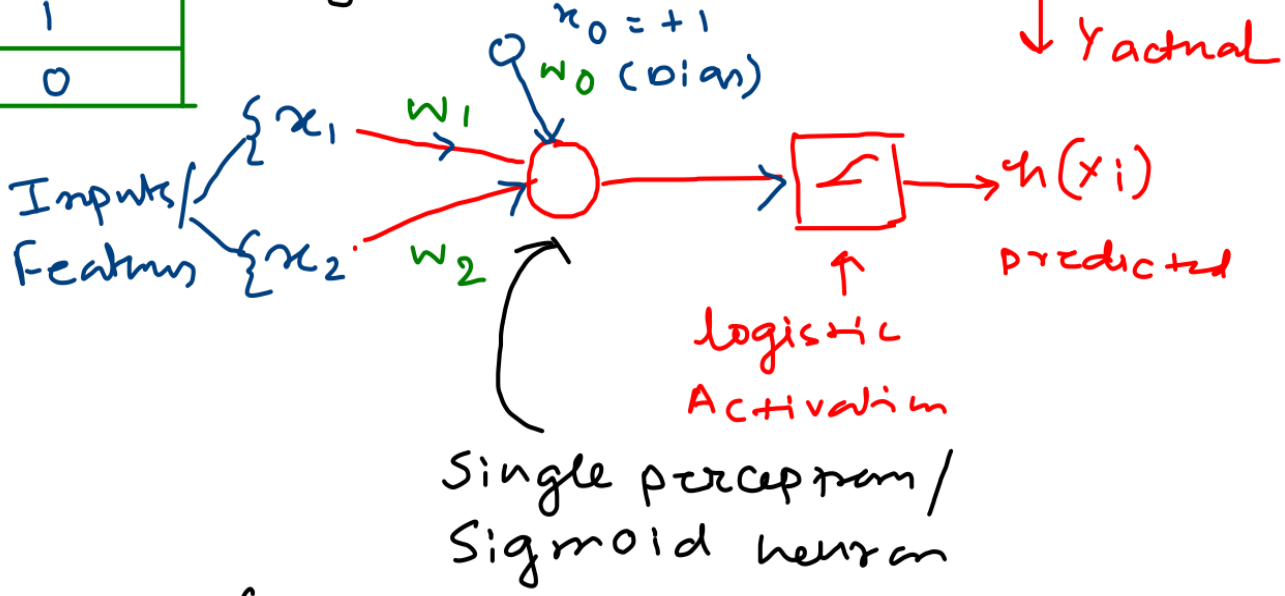
Dataset

x_0	x_1	x_2	Y
1	0.2	3	1
1	0.5	1	0
1	1.6	2	1
1	2.3	4	0

→ Training Samples

Step1: Normalization and split dataset (train, test, validation set)

Step2: Architecture (SLP/ Single layer perceptron) with Sigmoid neuron



Step3: Train using GD learning algorithm

Step4: After convergence, test for unknown patterns

Single perceptron to Singlelayer Perceptron (SLP)

- A singlelayer perceptron (**SLP**) is a feed-forward neural network
- Generally, it **(single perceptron)** has following properties:

  1. one perceptron neuron (threshold activation function)
  2. classifier model for binary classification of linearly separable patterns
- However, some modifications have been incorporated.

  1. Sigmoid neuron
  2. Can we use same architecture for multi-class classification problem ? (same one-vs.-all approach)
  3. Can we use similar architecture for regression problem?
  4. **SLP: Single layer of perceptrons/artificial neurons (to handle multi-class classification in one architecture)**

Single Perceptron for multi-class classification

x_0	x_1	x_2	y
1	.6	1	1
1	.2	3	2
1	.4	5	2
1	6	.2	3

Original dataset

class 1

x_0	x_1	x_2	y
1	.6	1	1
1	.2	3	0
1	.4	5	0
1	6	.2	0

Data set for SLP1

class 2

x_0	x_1	x_2	y
1	.6	1	0
1	.2	3	1
1	.4	5	1
1	6	.2	0

Dataset for SLP2

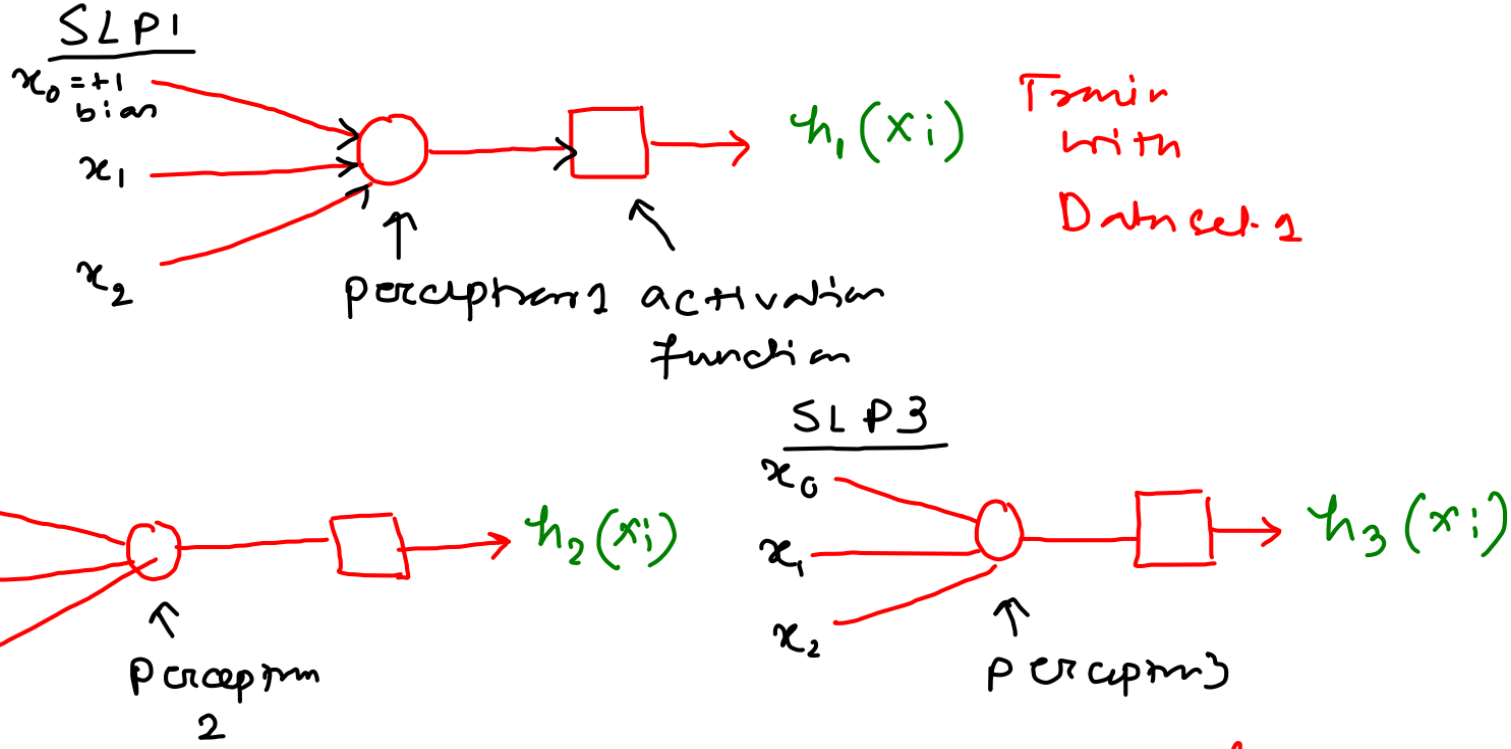
class 3

x_0	x_1	x_2	y
1	.6	1	0
1	.2	3	0
1	.4	5	0
1	6	.2	1

Data Set for SLP3

Step1 Training phase

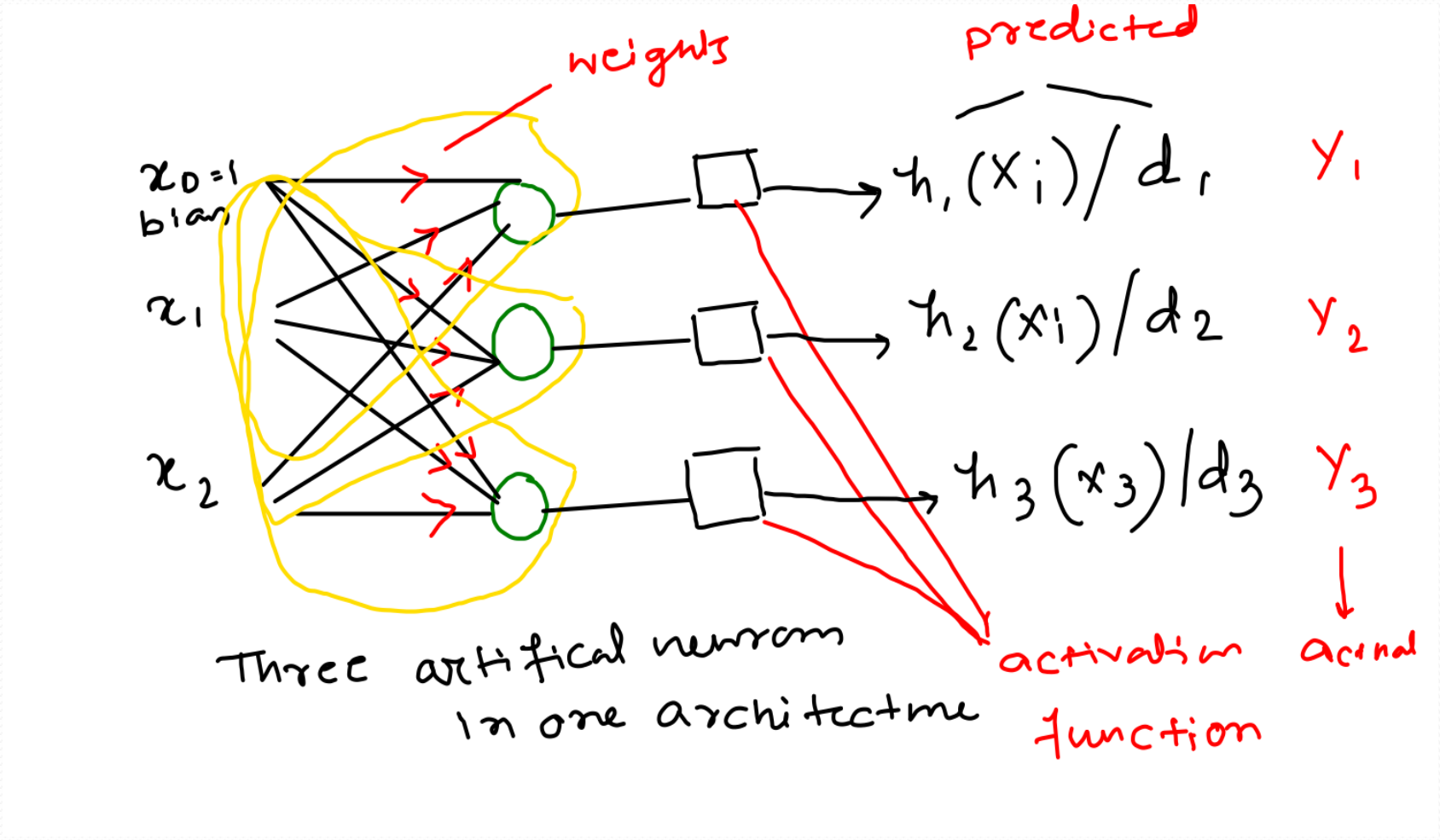
[train 3 SLPs for 3 class problem for seperate datasets]



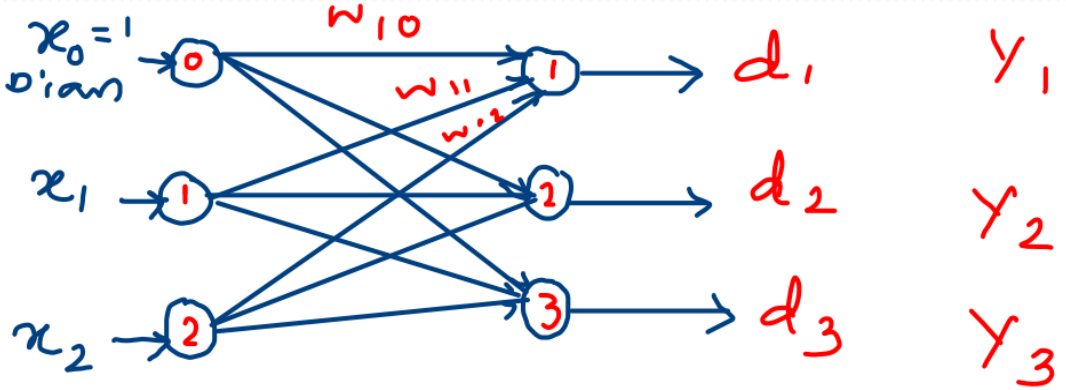
Step2 Choose k with $\max_k h_k(x_i)$

Singlelayer Perceptron (SLP)

Multi-class classification



Architecture for SLP



Inputs / Input neuron

Input layer

Output neurons

Output layer

$$W = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \end{bmatrix}$$

→ weights between output neuron i with input neurons / inputs

Singlelayer percepron (SLP)

- It is a feed-forward neural network.
- It has a single layer of perceptrons or artificial neurons.
- It can handle multiclass problem in one architecture.
- Generally, from architecture point of view, it has two layers:

1. Input layer: number of inputs = number of features+1
2. Output layer: number of output neurons/perceptron/sigmoid neuron=number of classes

- Each input is connected to all the output neurons.

One hot encoding

First step, pre-process the dataset (label) using one hot encoding to prepare it for the multi-class classification

$Y = 0 / 1$ — binary classification

$Y \longrightarrow Y_1 \quad Y_2 \quad Y_3$

One hot encoding

x_0	x_1	x_2	class label	y_1	y_2	y_3
1	0.3	1	1	1	0	0
1	0.4	6	2	0	1	0
1	0.2	1.2	1	1	0	0
1	6.2	7	2	0	1	0
1	6.5	6	3	?	?	?

↓

Each output neuron represents one class

Learning process of SLP

Target:

- Each output neuron is trained to activate for exactly one class.
- The correct output for SLP : one output neuron is activated (what indicates the classification decision), others are not activated.
- Perceptron learning algorithm can be used for threshold activation function.
- Gradient descent algorithm can be used for sigmoid activation function.

Incremental GD for SLP (with Sigmoid neuron)

Epoch 1 error = $E_{itr} = 0$

Training Sample 1

$j \rightarrow$ output neuron / class

$i \rightarrow$ input neuron / feature

$g(\cdot) \rightarrow$ activation function

all weights are initialized to 0.1

$$d_j = g\left(\sum w_{ji}\, x_i\right)$$

$$d_1 = g(0.1 * 1 + 0.1 * 0.3 + 0.1 * 1) = .6$$ Suppose

$$d_2 = g(0.1 * 1 + 0.1 * 0.3 + 0.1 \times 1) = .6$$

$$d_3 = g(0.1 * 1 + 0.1 * 0.3 + 0.1 \times 1) = .6$$

$$E_{itr} = E_{itr} + \frac{1}{2}\left[(d_1 - y_1)^2 + (d_2 - y_2)^2 + (d_3 - y_3)^2\right]$$

$$E_{itr} = E_{itr} + \frac{1}{2}\sum_{j=1}^{3}(d_j - y_j)^2$$

$$= 0 + \frac{1}{2}\left[(.6 - 1)^2 + (.6 - 0)^2 + (.6 - 0)^2\right]$$

Weight update

$\alpha = 0.5$

$$w_{ji} := w_{ji} + \alpha (y_j - d_j) x_i \boxed{d_j (1 - d_j)}$$

↳ if sigmoid activation

$$w_{10} = 0.1 + 0.5 (1 - .6) * 1 * 0.6 (1 - 0.6)$$

$$w_{11} = 0.1 + 0.5 (1 - .6) * 0.3 * 0.6 (1 - 0.6)$$

$w_{12} = ?$ $w_{20}, w_{21}, w_{22}, w_{30}, w_{31}, w_{32} = ?$

Training sample2 : Same procedure.

$E_{tr} = ?$

$w_{10} = ?$ for all weights ??

$w_{11} = ?$

$w_{12} = ?$

Repeat same process for all training samples

Repeat Epoch 1, Epoch 2, ..., Epoch 3

itr itr+1 itr+2 . . .

Epoch 1 Complete

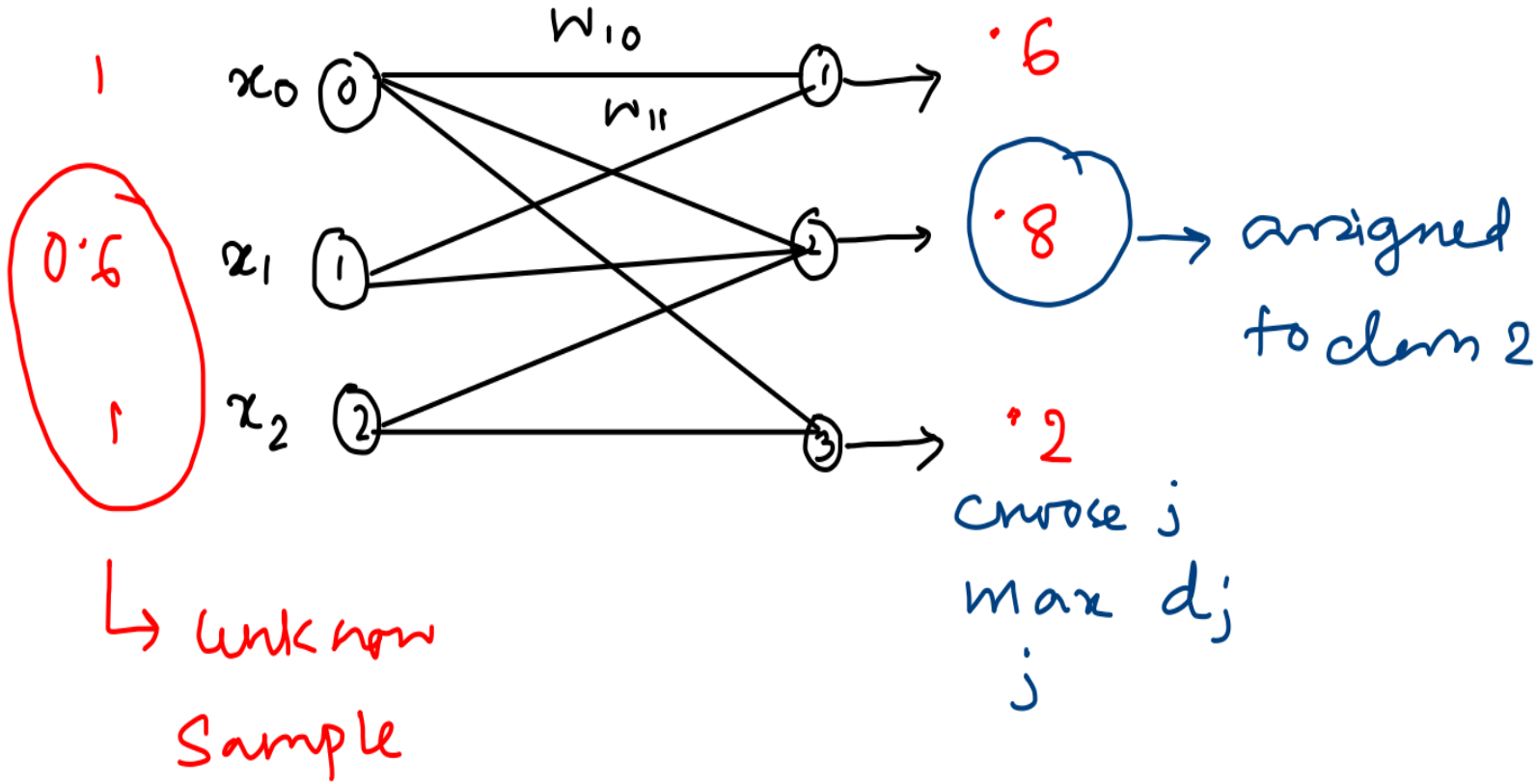
$E_{itr} = \frac{1}{m} E_{itr}$

Convergence:

$|E_{itr} - E_{itr-1}| <$ small value or

number of epochs > large value

Testing phase



Question

x_1	x_2	Class Label
1	2	1
0.2	3	2
4	1	2
6	1	3

Design SLP using the given training samples.

Note the following during demonstration:

1. Threshold activation function
2. Number of epochs:2
3. All weights are initialized as 0.1.

Predict the class label for test sample [3 0.1].