Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #39

Artificial Intelligence for Data Science

Week-11:

MACHINE LEARNING (Part VII)

Perceptron and Neural Network

Course Instructor:

Dr. Monidipa Das

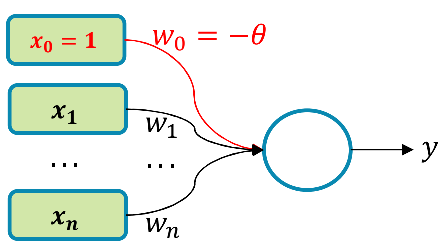
Assistant Professor

Department of Computational and Data Sciences

Indian Institute of Science Education and Research Kolkata, India 741246

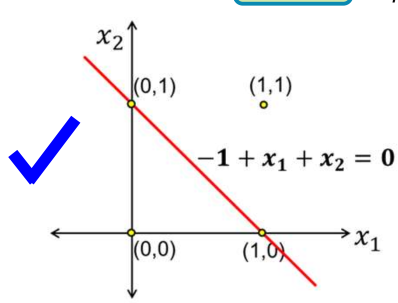
Perceptron: Introduction (revisit)

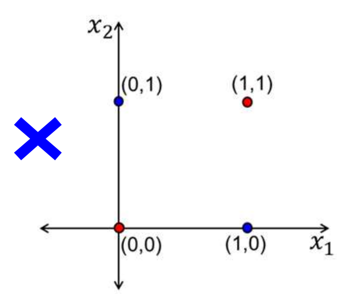
- More accepted convention:



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

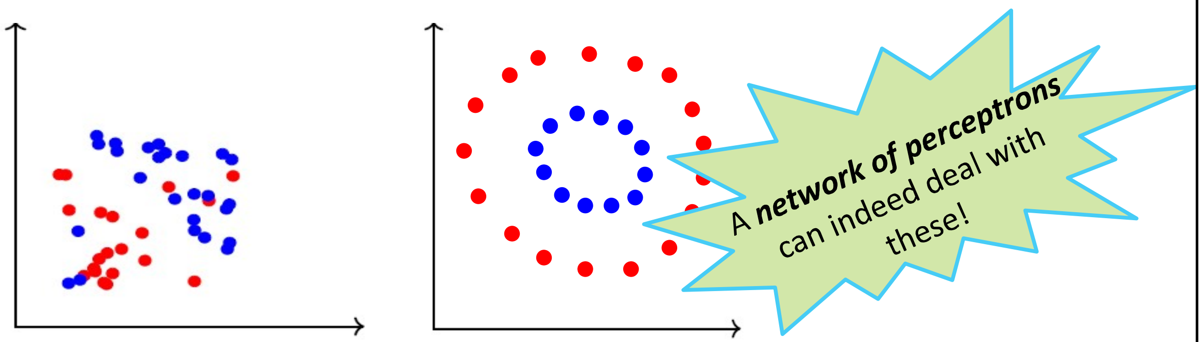
$x_0 = 1, w_0 = -\theta$





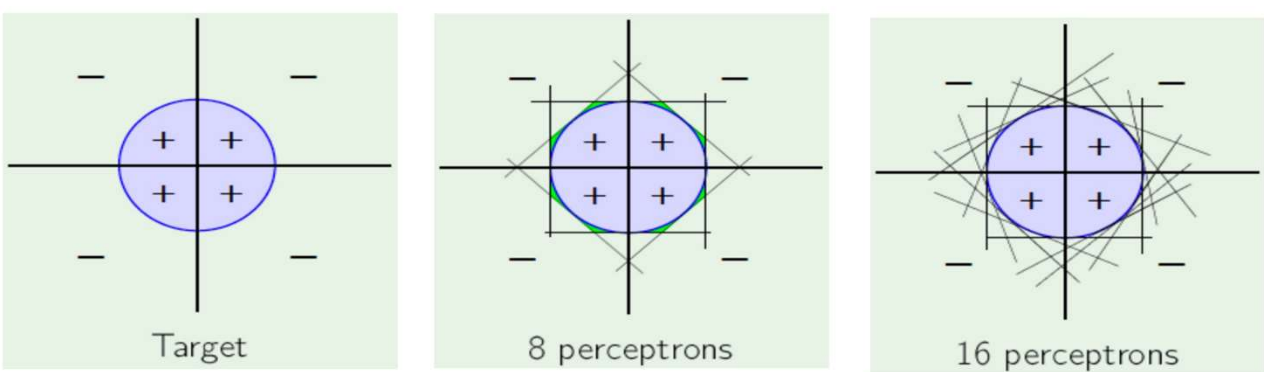
Points to Remember (revisit)

- Most real-world data is **not linearly separable** and will always contain some outliers



How do we implement functions that are not linearly separable ?

Combining Many Linear Classifiers

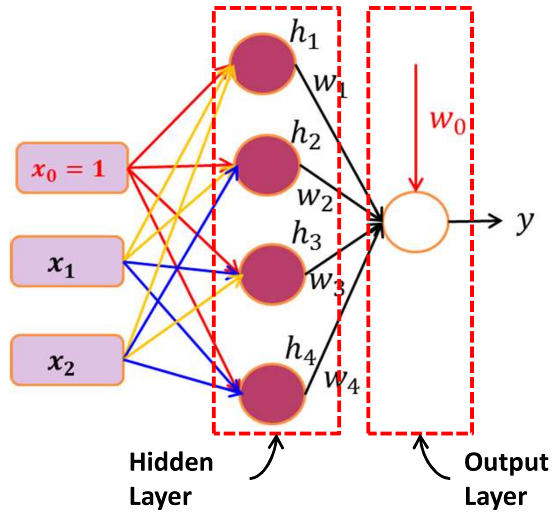


Perceptron Network

5

- Any boolean function of n inputs can be represented by a network of perceptrons containing ***1 hidden layer with 2^n perceptrons*** and ***one output layer containing 1 perceptron***

- Perceptron networks of these forms are called Multilayer Perceptrons (MLP)



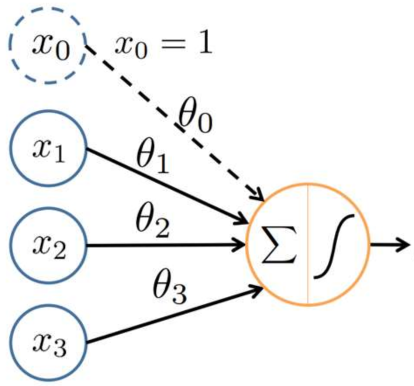
Perceptron: Other Issues

6

- A hard threshold decides the output (logical 0 or 1)
- Optimization becomes difficult with many perceptrons
- We would like to change the input a little and see how the output changes (iterative methods)
- **Desirable Property:**
  - Instead of a hard threshold, a smooth function that is efficient to differentiate
  - So that we can change the inputs a little, observe the corresponding small change in the output, hence compute gradient, etc.
- ***A perceptron with a smooth non-linear function*** is equivalent to a ***neuron*** *in NN*

Neuron Model: Logistic Unit

"bias unit"



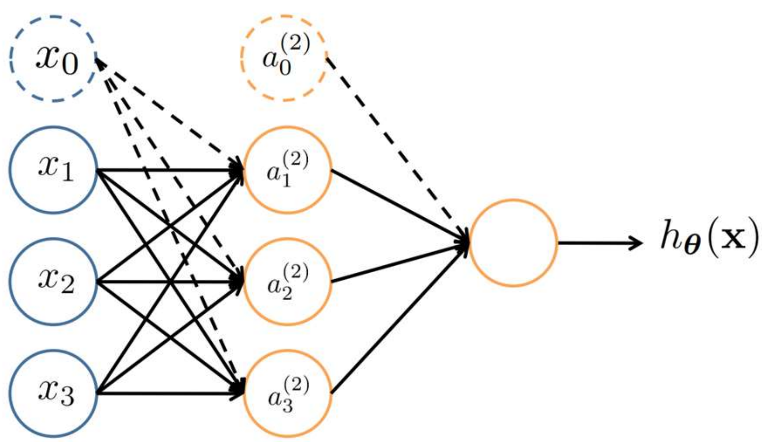
$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}\right) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \dfrac{1}{1 + e^{-z}}$

Neural Network

bias units



Layer 1
(Input Layer)

Layer 2
(Hidden Layer)

Layer 3
(Output Layer)

Feedforward Neural Network Structures

9

- Model is associated with a directed acyclic graph describing how functions composed
  - E.g., functions $f^{(1)}, f^{(2)}, f^{(3)}$ connected in a chain to form

$f(x) = f^{(3)}[f^{(2)}[f^{(1)}(x)]]$

    - $f^{(1)}$ is called the first layer of the network
    - $f^{(2)}$ is called the second layer, etc

- These chain structures are the most commonly used structures of neural networks

Dr. Monidipa Das, Department of CDS, IISER Kolkata

What a Hidden Unit Does

10

- Accepts a vector of inputs $\boldsymbol{x}$ and computes an affine transformation

$z = W^T\boldsymbol{x} + \boldsymbol{b}$

- Computes an element-wise non-linear function $g(z)$
- Most hidden units are distinguished from each other by the choice of activation function $g(z)$
- Design of hidden units is an active research area

Dr. Monidipa Das, Department of CDS, IISER Kolkata

Logistic Sigmoid

11

- Earlier, most neural networks used logistic sigmoid activation

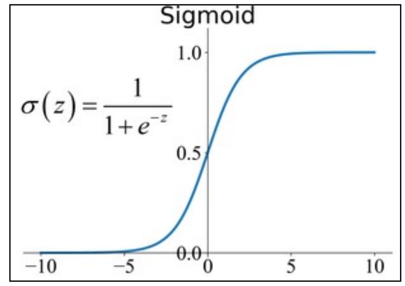
  $g(z)=\sigma(z)$

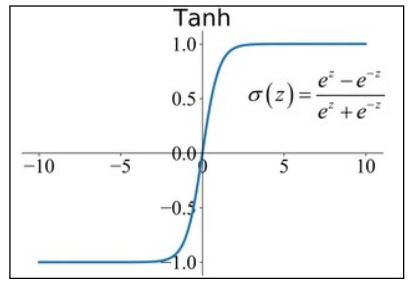
- Or the hyperbolic tangent

  $g(z)=\tanh(z)$

- These activation functions are closely related because

  $\tanh(z)=2\sigma(2z)-1$

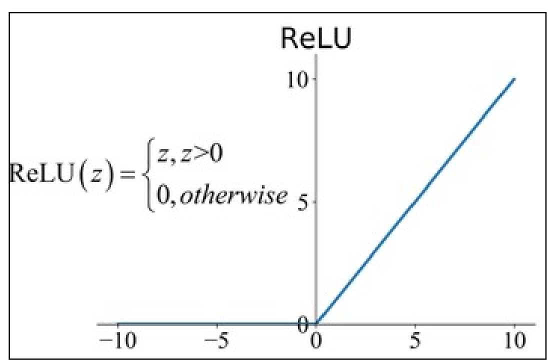




Rectified Linear Unit (ReLU)

12

- Rectified linear units use the activation function $g(z)=\max(0,z)$
  - They are easy to optimize due to their similarity to linear units
    - Only difference with linear units that they output 0 across half their domain
    - Derivative is 1 everywhere that the unit is active



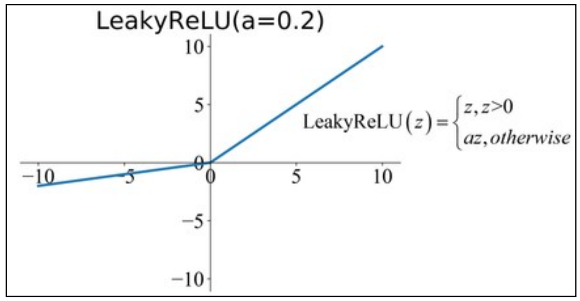
Generalizations of ReLU

13

- Three methods based on using a non-zero slope α_i when $z_i<0$:

$$h_i=g(z,\alpha)_i=\max(0,z_i)+\alpha_i \min(0,z_i)$$

1. Leaky ReLU:
   - fixes α_i to a small value like 0.01
2. Parametric ReLU or PReLU:
   - treats α_i as a parameter



Is Differentiability necessary?

14

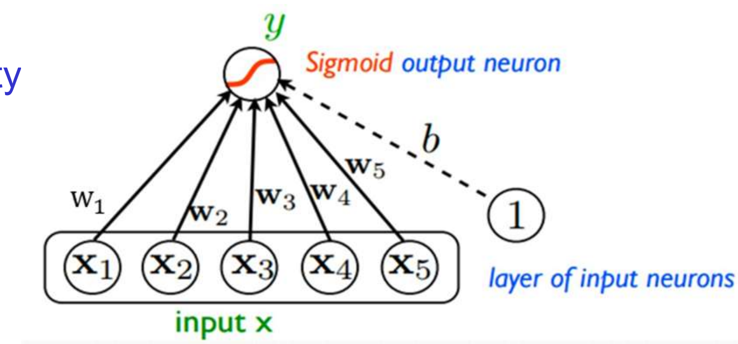
- Some hidden units are not differentiable at all input points
  - Rectified Linear Unit Function $g(z)=\max\{0,z\}$ is not differentiable at $z=0$
- Does this invalidate use in gradient-based learning?
- In practice gradient descent still performs well enough for these models to be used

Output Units

15

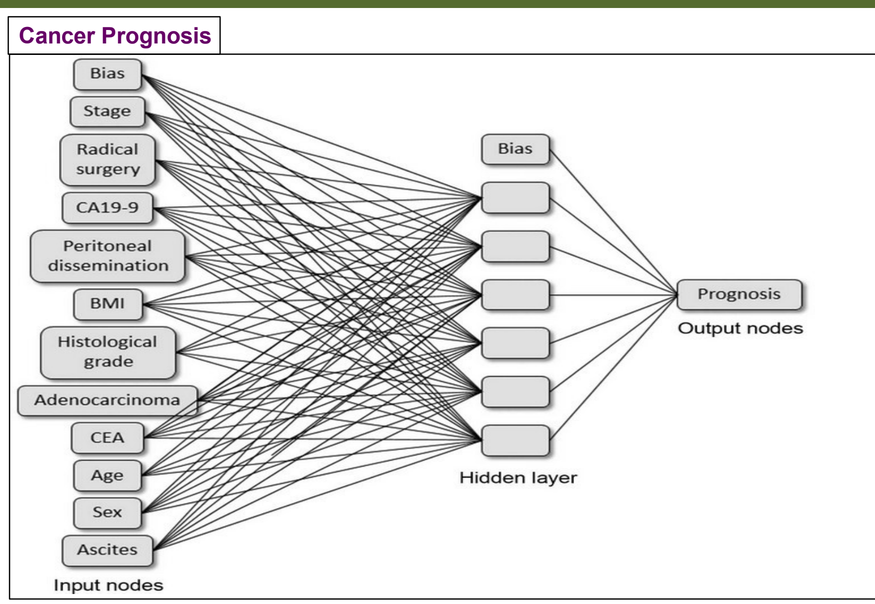
Types:

1. Linear units: no nonlinearity
2. Sigmoid units
3. Softmax units



Specific Application Architectures

16



Architecture for multi-class classification

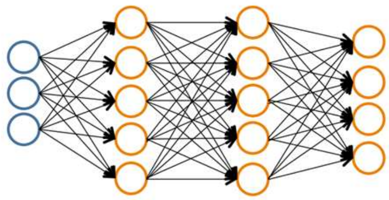
17


Pedestrian


Car


Motorcycle


Truck



$$h_\Theta(\mathbf{x}) \in \mathbb{R}^K$$

We want:

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$ when pedestrian

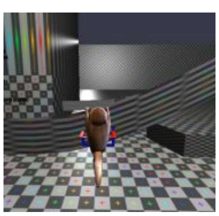
$h_\Theta(\mathbf{x}) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$ when car

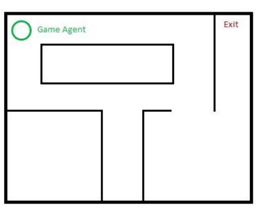
$h_\Theta(\mathbf{x}) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$ when motorcycle

$h_\Theta(\mathbf{x}) \approx \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$ when truck

An Architecture for Game Design

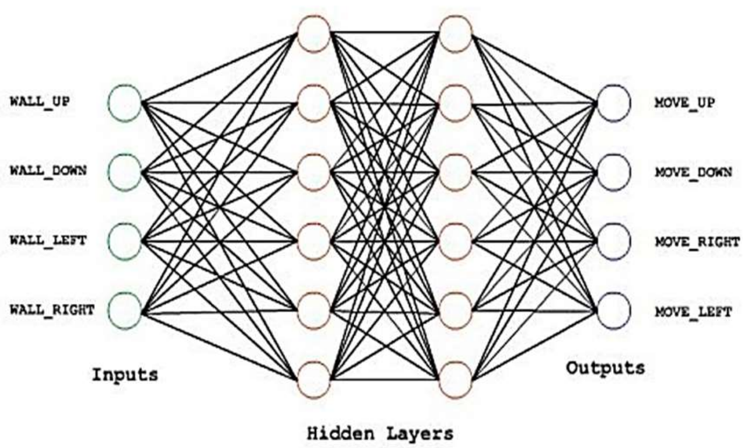
18





Maze Game Recording Output File

Inputs: Wall Directions				Outputs: Game Agent Direction			
UP	DOWN	LEFT	RIGHT	UP	DOWN	LEFT	RIGHT
1	0	0	0	0	1	0	0
1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1
0	0	1	0	0	0	0	1



Back Propagation Network

19

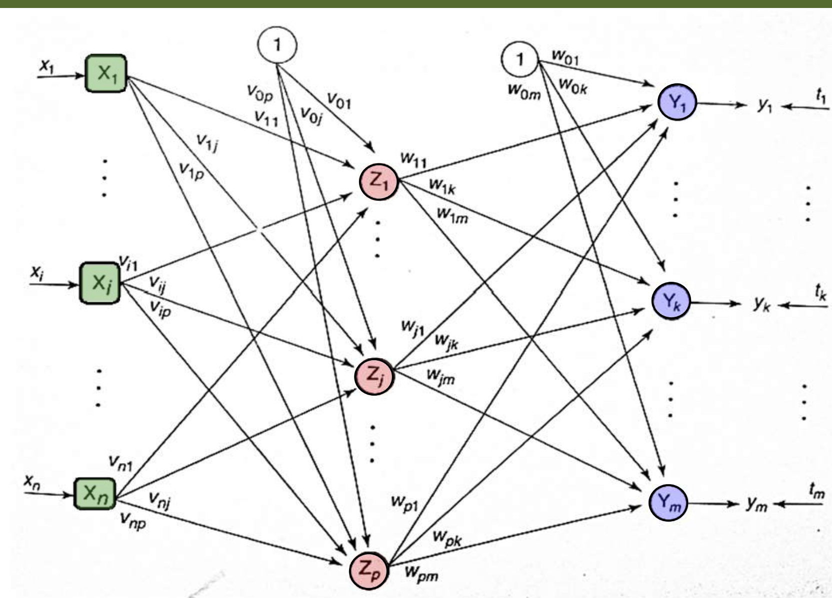
- Networks associated with ***back propagation algorithm***
- Applied to multilayer feed-forward networks consisting of processing elements with continuous differentiable activation functions
- Training is done in ***three stages***
  - Feed-forward of the input training pattern
  - Calculation and back-propagation of the error
  - Updating weights

Gradient descent

Back Propagation Network (BPN)

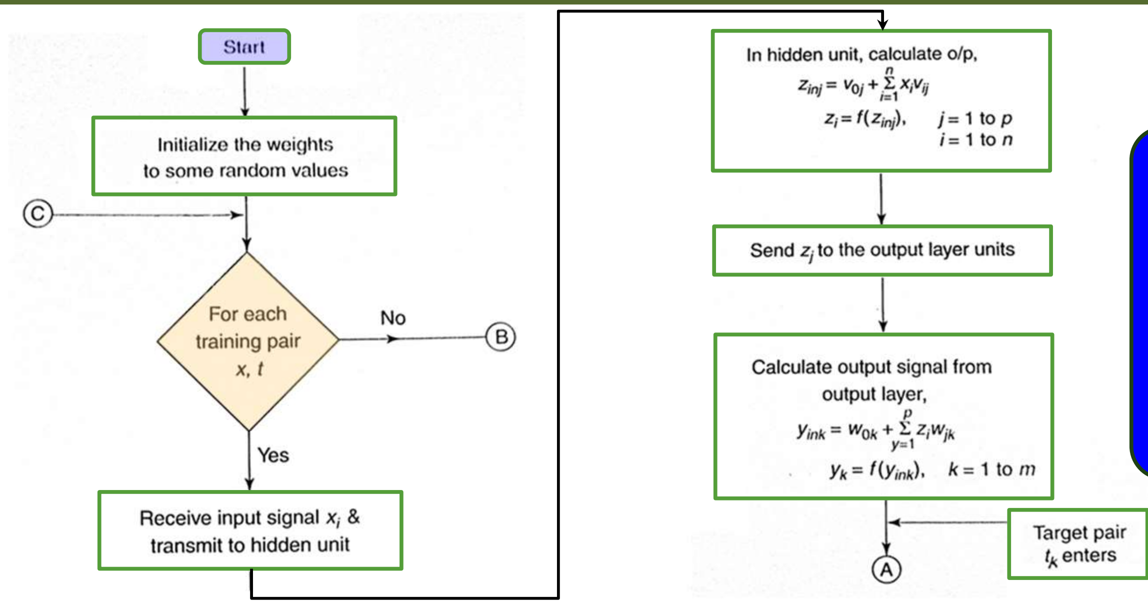
20

- Architecture



BPN Training

21



To be continued in the next lecture

22

Questions?