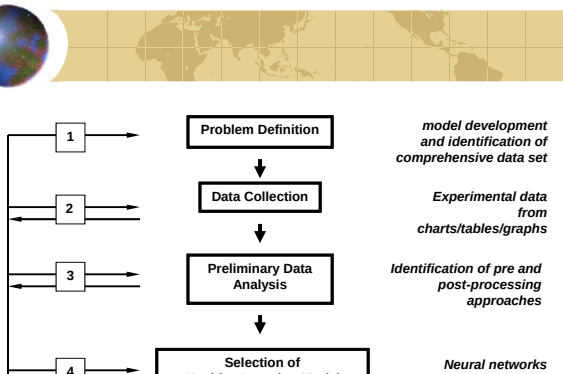
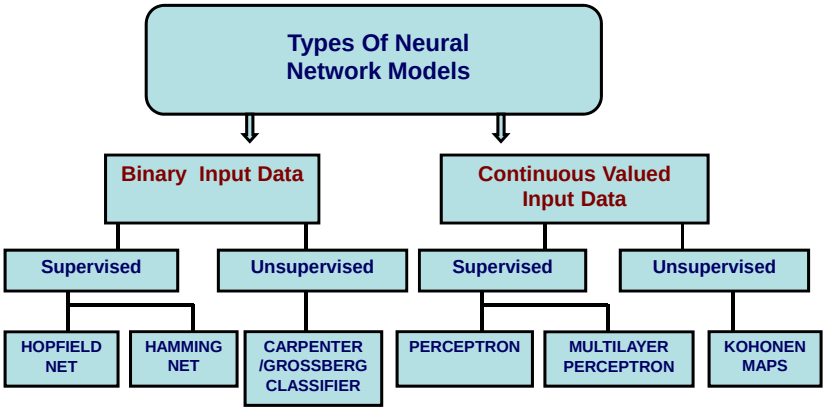
Neural Networks : Introduction

Lecture 3



Scheme of Modelling

Existing Neural Models

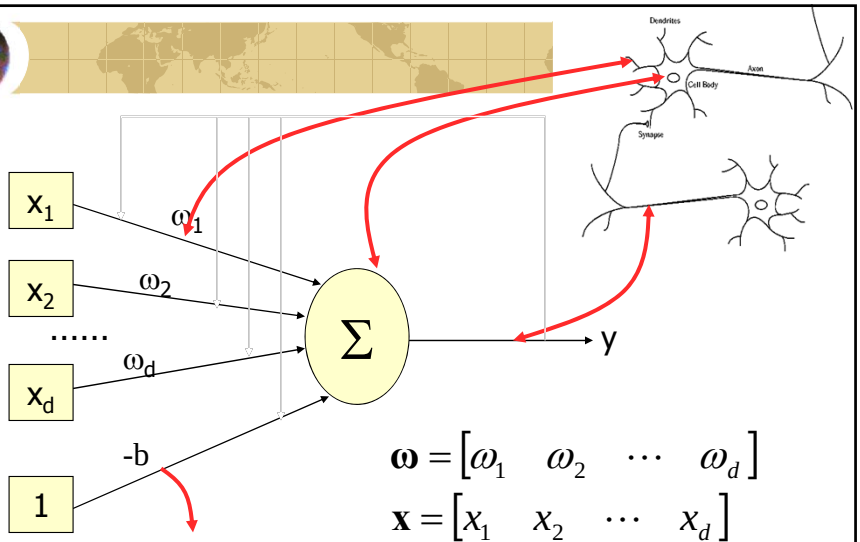


3

Supervised Learning in Neural Networks Model- Perceptron

Supervised Learning in Neural Networks

- The learning principle is to provide the input values and the desired output values for each of the training examples.
- The neural network changes its connection weights during training.
- Calculate the error:
  - training error - how well a NN has learned the data
  - test error - how well a trained NN generalizes over new input data.



$$\boldsymbol{\omega} = [\omega_1 \quad \omega_2 \quad \cdots \quad \omega_d]$$

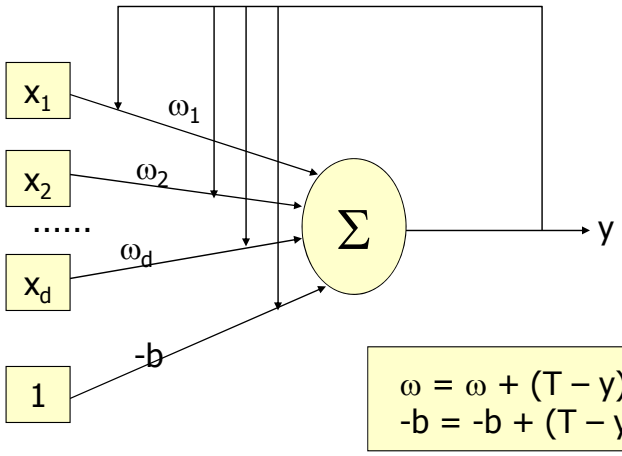
$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_d]$$

$$y = \begin{cases} 1 & \text{if } \omega^T x - b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Perceptron

- A program that learn "concepts" based on examples and correct answers
- It can only respond with "true" or "false"
- Single layer neural network
- By training, the weight and bias of the network will be changed to be able to classify the training set with 100% accuracy

Perceptron Learning Rule



$\omega = \omega + (T - y)\, x$
$-b = -b + (T - y)$

T is the expected output
y is the real output

Training Steps

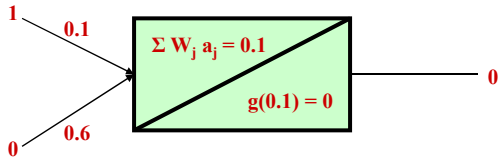
- Step1: Samples are presented to the network
- Step2: If the output is correct, no change is made; Otherwise, the weight and biases will be updated based on perceptron learning rule
- Step3: An entire pass through all the training set is called an "epoch". If no change has been made for the epoch, stop. Otherwise, go back Step1

Details About Training

- Learning from examples
  - Examples consist of input and correct output
- Learn if network's output doesn't match correct output
  - Adjust weights to reduce difference
  - Only change weights a small amount (η)
- Basic perceptron learning
  - $W_{i,j} = W_{i,j} + \eta(t-o)a_j$
  - If output is too high (t-o) is negative so $W_{i,j}$ will be reduced
  - If output is too low (t-o) is positive so $W_{i,j}$ will be increased
  - If a_j is negative the opposite happens

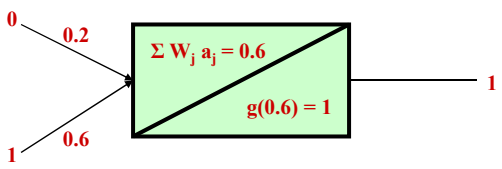
Perceptron Example

- Single perceptron to represent OR
  - Two inputs
  - One output (**1 if either inputs is 1**)
  - Step function (**if weighted sum > 0.5 output a 1**)
- Initial state (below) gives error on (1,0) input
  - Training occurs

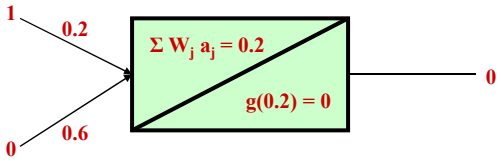


Perceptron Example

- Wj = Wj + η(t-o)aj
- W1 = 0.1 + 0.1(1-0)1 = 0.2
- W2 = 0.6 + 0.1(1-0)0 = 0.6
- After this step, try (0,1)→1 example
  - No error, so no training

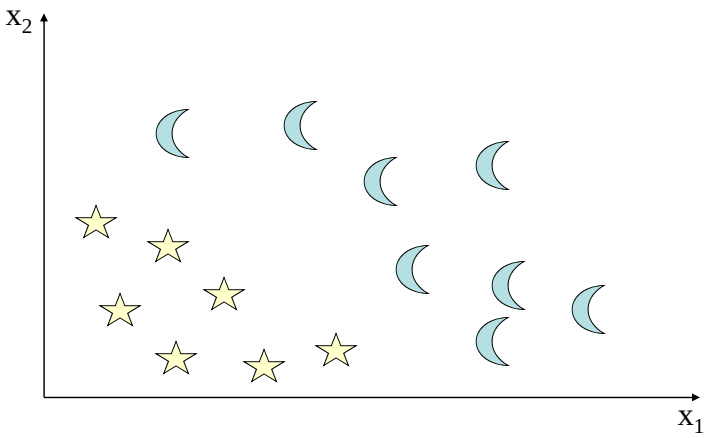


Perceptron Example



- Try (1,0)→1 example
  - Still an error, so training occurs
- $W_1 = 0.2 + 0.1(1-0)1 = 0.3$
- $W_2 = 0.6 + 0.1(1-0)0 = 0.6$

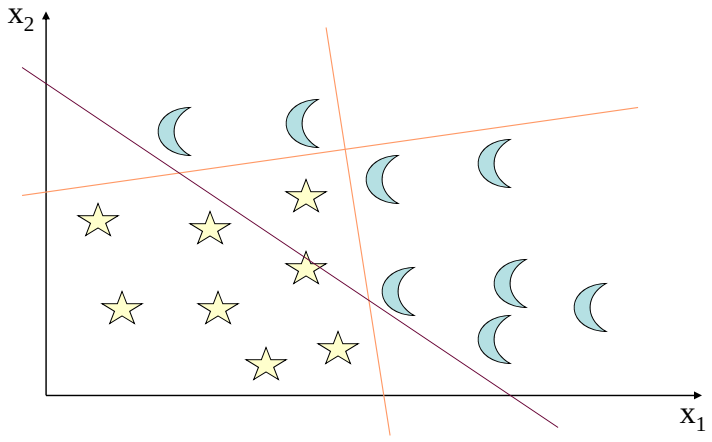
Classification Example



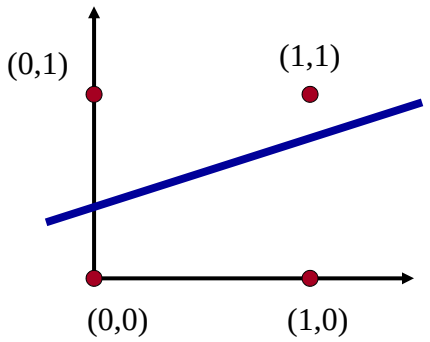
Limitations

- The output only has two values (1 or 0)
- Can only classify samples which are linearly separable (straight line or straight plane)
- Can't train a network functions like XOR

Linearly Separable -- Not



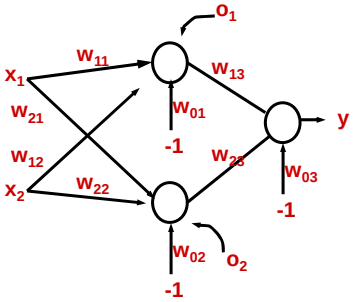
XOR Problem



Four Input vectors for the XOR problem. The line $y = w_1x_1 + w_2y_2$ divides the plane into two regions but cannot successfully isolate the set of points (0,0) and (1,1) from the points (0,1) and (1,0)

Solving the XOR Problem

Network Topology:
2 hidden nodes
1 output



Desired behavior:

x1	x2	o1	o2	y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

Weights:
w11= w12=1
w21=w22 = 1
w01=3/2; w02=1/2; w03=1/2
w13=-1; w23=1