

Pattern Classification

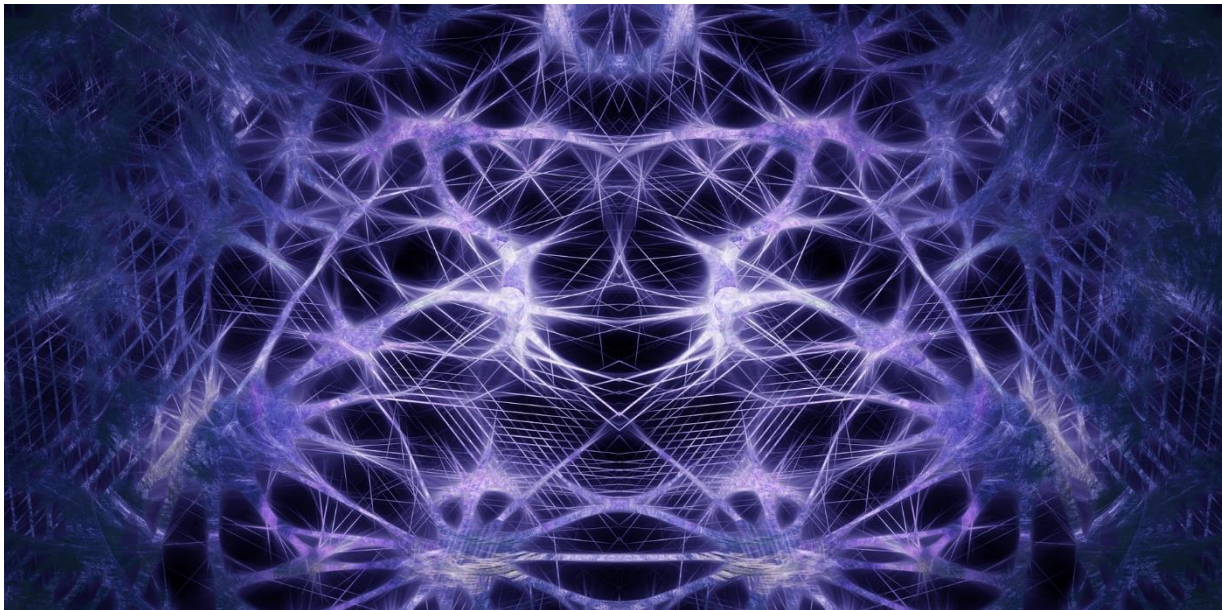
Neural Networks

AbdElMoniem Bayoumi, PhD

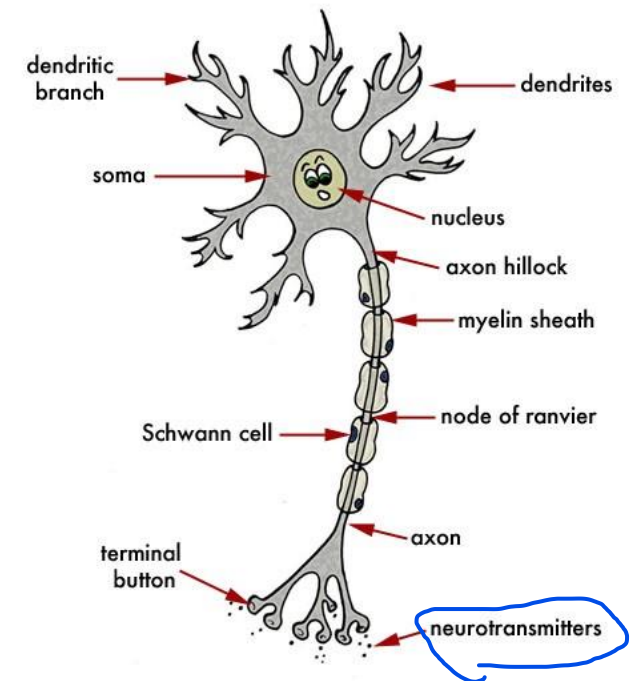
Fall 2021

Human Brain

- The brain has $10^{10} - 10^{11}$ cells that are highly connected
- Every cell (called neuron) is connected to about $10^3 - 10^4$ other cells in its neighborhood



Source: Luis Bermudez – Machine Vision

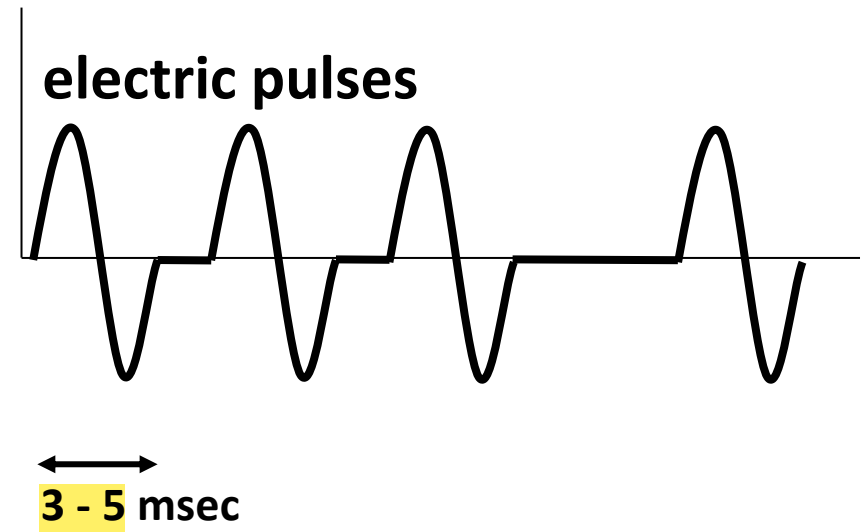
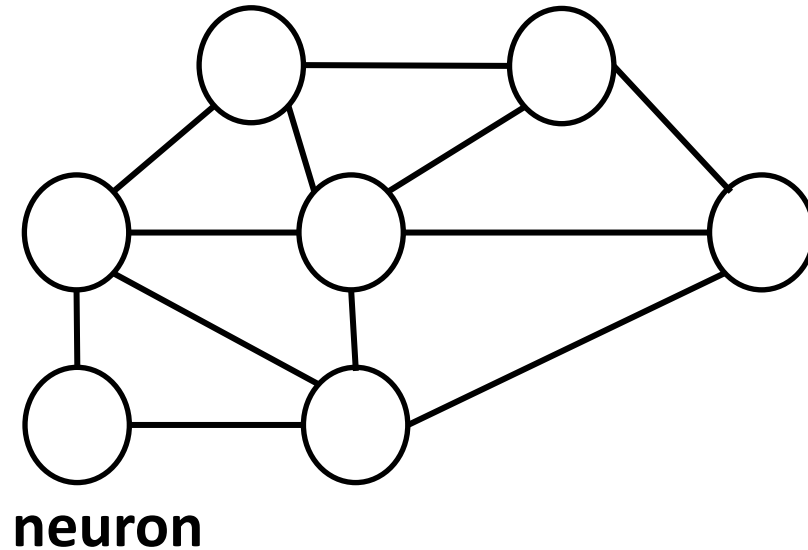


Source: Shubham Panchal - Predict

Human Brain

- The cells exchange information in the form of electric signals (pulses)
- That is how the brain processes information and performs intelligent tasks
- Success of the brain is a result of the parallelism of this huge network of cells

Human Brain



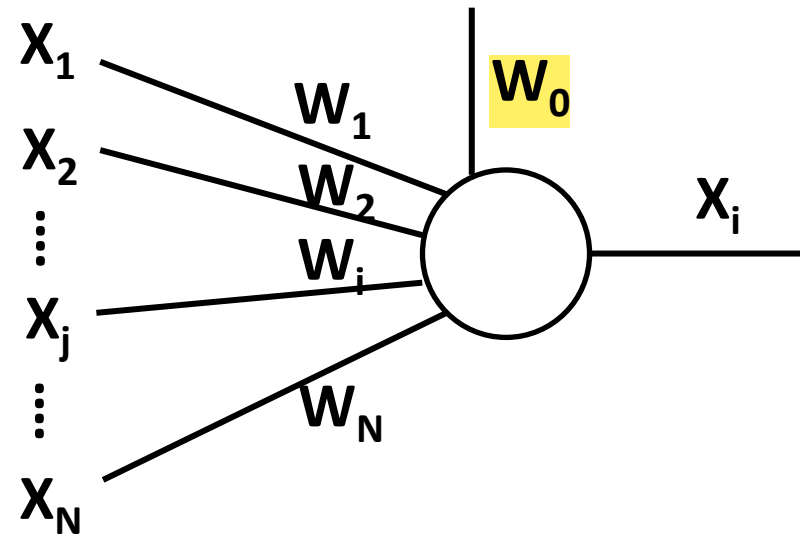
- Time constant $\approx 3 - 5$ msec
- Traditional computer's time constant $\sim 10^{-9}$ or nano sec

Human Brain

- The “time constant” of the brain is significantly larger than that of the traditional computers
- Brain is superior because of the massive parallelism

Neural Networks

- Mimic how the brain works

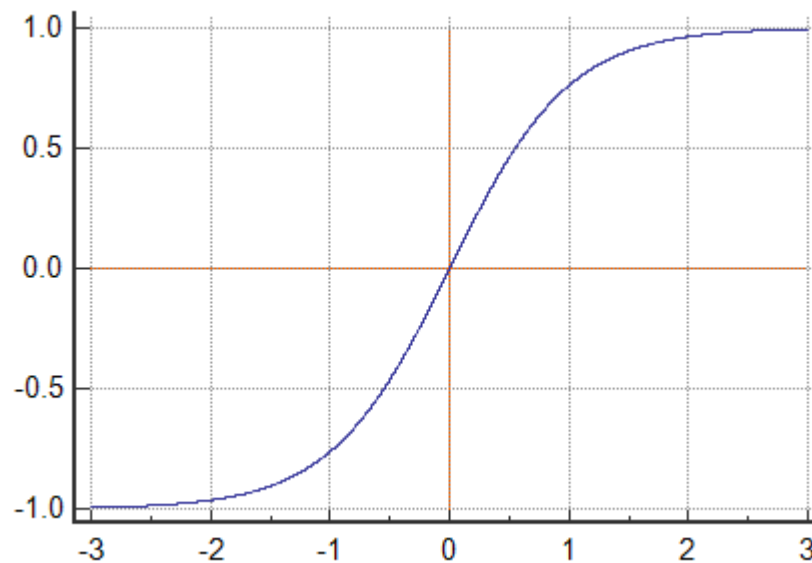


$$X_i = f \left(\sum_{j=1}^N W_j X_j + W_0 \right)$$

- $W_j \equiv$ weight \rightarrow gives the degree of how input cell j affects the receiving cell I
- $W_0 \equiv$ some constant called bias or threshold
- $f(x) \equiv$ activation function

Neural Networks

$$X_i = f\left(\sum_{j=1}^N W_j X_j + W_0\right)$$



tanh

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

Neural Networks

- Every neuron produces output X_j
- If neuron j is connected to neuron i then X_j will have an affect on X_i
- Each neuron can have a different effect, hence, we multiply by weight w_j fkda el khebra bta3t el model de bt3dl el weights bta3t el khlaya bta3t el neurons.
- Weights are the parameters that encode the **information in the brain** el weights fe mokh el ensan hya el bt3br 3n shakhseto

Neural Networks

- Weights act like the “storage” in computers
- When a human encounters a new experience the weights of his brain gets adjusted
- This adjustment results in his learning of the new experience
- Memories are encoded in the weights

Neural Networks

- We try in the field of neural networks (NNs) to exploit the **learning feature observed** in the brain
- The main concept of NNs is that we have a **model given in terms of parameters**, i.e. **weights**.
- The weights determine the functionality of the model

Neural Networks

- We apply some learning algorithm that adjusts the weights in small steps
- The goal is to lead the NN to learn to implement the given problem

Recall: Feature Vector

- Let $\underline{X}(\mathbf{m}) = \begin{bmatrix} X_1(\mathbf{m}) \\ X_2(\mathbf{m}) \\ \vdots \\ X_N(\mathbf{m}) \end{bmatrix}$ be the feature vector of the m^{th} training pattern
- N is the number of features, i.e., the dimension of $\underline{X}(\mathbf{m})$
- M is the number of the training patterns

Example

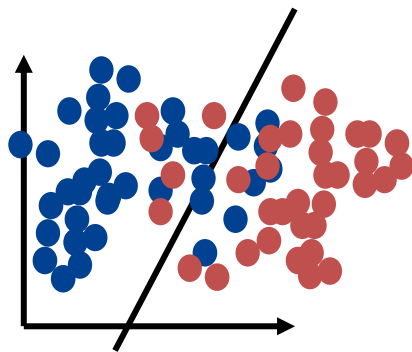
- Given a training set $\underline{X}(1), \underline{X}(2) \cdots \underline{X}(M)$, design a linear classifier
 - Determine the parameters $W_0, W_1 \cdots W_N$

$$y(m) = f(W_0 + \underline{w}^T \underline{X}(m)) = \begin{cases} 1, & \text{if } \underline{X}(m) \in C_1 \\ 0, & \text{if } \underline{X}(m) \in C_2 \end{cases}$$

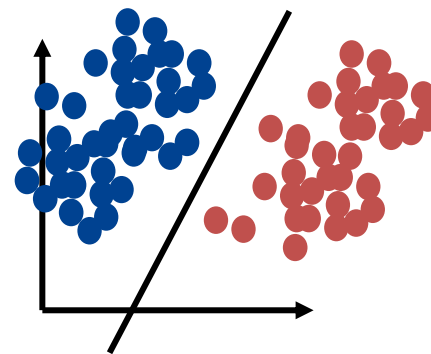
$$\underline{w} = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_N \end{bmatrix} \equiv \text{weight vector}$$

Recall: Types of Problems

- A problem is said to be ***linearly separable*** if there is a hyperplane that can separate the training data points of class C_1 from those of C_2
- Otherwise it is said to be ***not linearly separable***



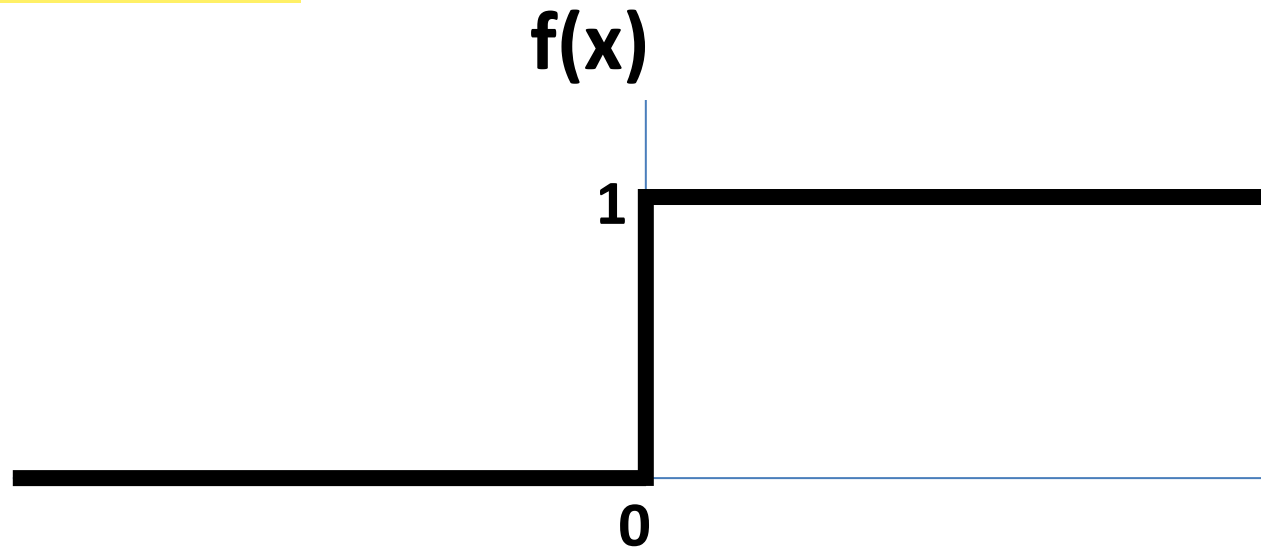
Not linearly separable



Linearly separable

Example – Contd.

- Define the activation function $f(x)$ as a unit step function



unit step fn.

or hard limiting fn.

or hard threshold fn.

Example – Contd.

neuron

- $y(m) = f(W_0 + \underline{w}^T \underline{X}(m)) = \begin{cases} 1, & \text{if } \underline{X}(m) \in C_1 \\ 0, & \text{if } \underline{X}(m) \in C_2 \end{cases}$

- The neuron can implement a linear classifier

Example – Contd.

- Define augmented vectors:

$$\underline{W} = \begin{bmatrix} W_0 \\ \vdots \\ W_N \end{bmatrix} = \begin{bmatrix} W_0 \\ \underline{w} \end{bmatrix} \quad \text{where } \underline{w} = \begin{bmatrix} W_1 \\ \vdots \\ W_N \end{bmatrix}$$

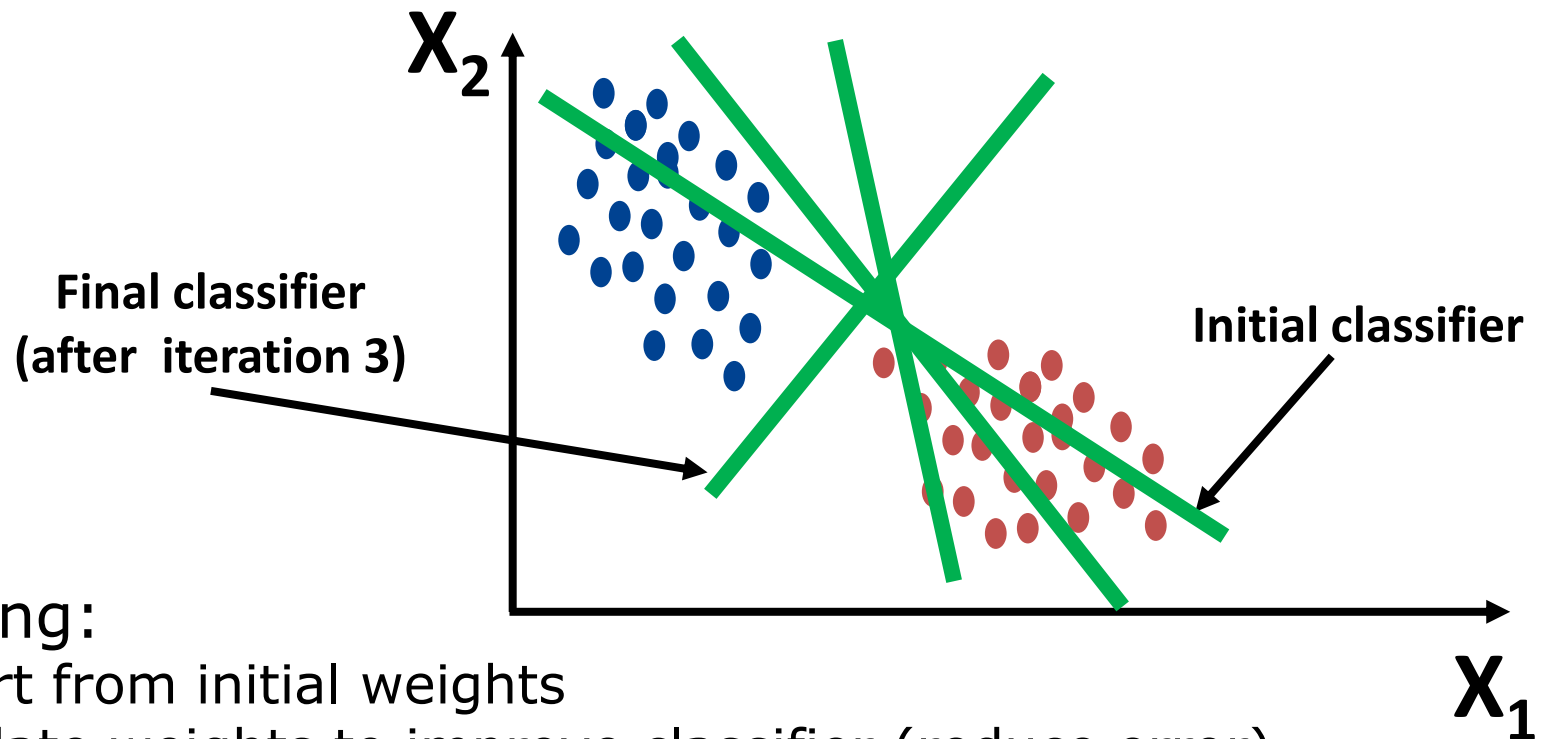
$$\underline{u}(m) = \begin{bmatrix} 1 \\ \underline{X}(m) \end{bmatrix} = \begin{bmatrix} 1 \\ X_1(m) \\ \vdots \\ X_N(m) \end{bmatrix}$$

So:

$$y(m) = f(\underline{W}^T \underline{u}(m))$$

Example – Contd.

- $y(m) = f(\underline{W}^T \underline{u}(m)) = \begin{cases} 1, & \text{if } \underline{u}(m) \in C_1 \\ 0, & \text{if } \underline{u}(m) \in C_2 \end{cases}$



- Learning:
 - Start from initial weights
 - Update weights to improve classifier (reduce error)
 - Repeat until arriving to the best possible classifier

el fekra bta3t el NN enk tbd2 b initial weight
w tefdl t3dl fel parameters l7d ma tewsl l a7sn classifier momken.

Linear Perception

- The linear neuron (linear perception):
The linear perceptron can classify correctly only linearly separable problems
- It can be used for non-linearly separable problems as long as it produces a low classification error rate

Linear Perceptron Algorithm

1. Initialize the weights and threshold (bias) randomly
2. Present the augmented input (or feature) vector of the m^{th} training $\underline{u}(m)$ and its corresponding desired output $d(m)$

$$d(m) = \begin{cases} 1, & \text{if } \underline{u}(m) \in C_1 \\ 0, & \text{if } \underline{u}(m) \in C_2 \end{cases}$$

3. Calculate the actual output for pattern m :

$$y(m) = f(\underline{W}^T \underline{u}(m))$$

4. Adapt the weights according to the following rule (called Widrow-Hoff rule):

$$\underline{W}(\text{new}) = \underline{W}(\text{old}) + \eta[d(m) - y(m)]\underline{u}(m)$$

where η is a constant called the learning rate

5. Go to step 2 until all patterns are classified correctly, i.e., $d(m)=y(m)$ for $m=1, \dots, M$

Note: the algorithm is sequential w.r.t. the patterns $\underline{u}(m)$

Understanding Widrow-Hoff Update

- If $d(m)=y(m)$ then no change is needed in the weights, i.e., $\underline{W}(new) = \underline{W}(old)$, because $d(m)-y(m)=0$
- If $d(m) \neq y(m)$ then weights get updated

$$\underline{W}(new) = \underline{W}(old) + \eta[d(m) - y(m)]\underline{u}(m)$$

Acknowledgment

- These slides have been created relying on lecture notes of Prof. Dr. Amir Atiya