XOR PROBLEM AND A FEEDFORWARD NEURAL NETWORK FOR ITS SOLVING.
HOPFIELD NEURAL NETWORK
Limited Functionality of the Threshold Neuron

• **Threshold (linearly separable) functions** can be learned by a single threshold neuron

• **Non-threshold (nonlinearly separable) functions** cannot be learned by a single neuron. For learning of these functions a neural network created from threshold neurons is required (Minsky-Papert, 1969)

• The number of all Boolean functions of $n$ variables is equal to $2^{2^n}$, but the number of the threshold ones is substantially smaller. Really, for $n=2$ fourteen from sixteen functions (excepting XOR and *not* XOR) are threshold, for $n=3$ there are 104 threshold functions from 256, but for $n>3$ the following correspondence is true ($T$ is a number of threshold functions of $n$ variables):

$$\frac{T}{2^{2^n}} \underset{n>3}{\rightarrow} 0$$

• For example, for $n=4$ there are only about 2000 threshold functions from 65536
Threshold Functions and Pattern Recognition (Classification)

• If two classes of objects cannot be linearly separated, this means that there is no way to solve a corresponding pattern recognition problem using the threshold neuron.
When we need a neural network?

• The functionality of a single neuron is limited. For example, the threshold neuron cannot learn non-linearly separable functions.

• To learn those input/output mappings that cannot be learned by a single neuron, a neural network should be used.
The simplest Feedforward Neural Network from Threshold Neurons

Input Layer. No processing here, just distribution of inputs among the hidden layer neurons.

No processing here, just distribution of inputs among the hidden layer neurons.
The simplest Feedforward Neural Network from Threshold Neurons

- This neural network is called a feedforward network because it does not contain feedback connections. It forwards outputs of all neurons just to a neuron (neurons) from a following layer.
Solving the XOR problem

• XOR and NXOR are the only non-linearly separable Boolean functions of two variables. They cannot be learned using a single threshold neuron.

• However, we can represent XOR as a superposition of linearly-separable Boolean functions
Solving the XOR problem

- A **Disjunctive Normal Form (DNF)** (also referred to as a **Sum of Products Form**) is a representation of a logical formula through a disjunction of elementary conjunctions.

- An **elementary conjunction** is a conjunction of Boolean variables or their negations (no other logical operations are involved in the elementary conjunction). For example,

\[ x_1 x_2 \bar{x}_3; \quad x_1 x_2 x_3; \quad \bar{x}_1 x_2 \bar{x}_3; \quad x_1 \bar{x}_2 x_3 \]
Solving the XOR problem

• A Full Disjunctive Normal Form (FDNF) is such a DNF where each of its variables appears exactly once in every elementary conjunction either without negation or with negation.

• Any FDNF of $n$ variables contains the same amount of elementary conjunctions that the number of 1s among the values of the corresponding function.
Solving the XOR problem

- Since both conjunction and disjunction are threshold functions, and any non-threshold function can be presented as their superposition in FDNF, then it is possible to create a feedforward neural network with one hidden layer containing the same amount of neurons as the number of elementary conjunctions in FDNF and one output layer containing a single neuron.

- The hidden layer neurons implement elementary conjunctions of inputs, and a single output neuron implements disjunction of these conjunctions.

- This makes it possible to learn any non-linearly separable Boolean function using this neural network.
A Feedforward Neural Network

Hidden layer  Output layer
An algorithm for obtaining an FDNF of a Boolean function of $n$ variables:

• Create a table containing $2^n$ rows and $n+1$ columns. Put in the first $n$ columns values of all the variables and in the last column corresponding values of the function.

• Create elementary conjunctions corresponding to those values of a function, which equal 1 and containing $n$ variables each. If the corresponding Boolean variable equals 1, take it without negation, if it equals 0, take it with negation (with respect to the Boolean alphabet \{0,1\}).

• Take disjunction of all $2^n$ elementary conjunctions. This will be FDNF
Solving the XOR problem

- FDNF for the XOR function

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$\text{XOR}(x_1, x_2)$</th>
<th>$\text{XOR}(x_1, x_2)$ in {1,-1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$\text{XOR}(x_1, x_2) = x_1 \oplus_{\text{mod} 2} x_2 = \overline{x}_1 x_2 \lor x_1 \overline{x}_2$
Solving the XOR problem

\[ x_1 \oplus x_2 = x_1 \bar{x}_2 \lor \bar{x}_1 x_2 = f_1(x_1, x_2) \lor f_2(x_1, x_2) \]

\[ f_1(x_1, x_2) \]

\[ f_2(x_1, x_2) \]
Solving the XOR problem

<table>
<thead>
<tr>
<th>#</th>
<th>Inputs</th>
<th>Neuron 1</th>
<th>Neuron 2</th>
<th>Neuron 3</th>
<th>XOR= $x_1 \oplus x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$Z$</td>
<td>$Z$</td>
<td>$Z$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sgn($z$)</td>
<td>sgn($z$)</td>
<td>sgn($z$)</td>
</tr>
<tr>
<td>1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2)</td>
<td>1</td>
<td>-1</td>
<td>-5</td>
<td>-1</td>
<td>7</td>
</tr>
<tr>
<td>3)</td>
<td>-1</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4)</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
HOPFIE LD NEURAL NETWORK
Hopfield Neural Network

• In 1982, John Hopfield proposed a fully connected recurrent neural network with feedback links with binary inputs and binary outputs, built from the threshold neurons.

• The *Hopfield Neural Network* is a multiple-loop feedback neural network, which can be used first of all as an associative memory.

• All the neurons in this network are connected to all other neurons except to themselves that is there are no self-feedbacks in the network.
Hopfield Neural Network

The Hopfield neural network with 8 neurons

The Hopfield neural network with 4 neurons
Hopfield Neural Network

• A main idea behind the Hopfield net is to use it as an **associative memory** (content-addressable memory)

• An associative memory may learn patterns (for, example, to store $n \times m$ images in the associative memory, we should take the $n \times m$ Hopfield network whose each neuron learns the intensity values in the corresponding pixels; in this case, there is a one-to-one correspondence between a set of pixels and a set of neurons)
Hopfield Neural Network

• The Hebbian learning rule should be efficiently used to learn patterns that one need to store.

• After the learning process is completed, the associative memory may retrieve those patterns, which were learned, even from their fragments or from distorted (noisy or corrupted) patterns.
Hopfield Neural Network

• The weight $w_{ij}$ corresponds to the synaptic connection of the $i$th neuron and the $j$th neuron. It is important that in the Hopfield network, for the $i$th and $j$th neurons $w_{ij} = w_{ji}$. Since there is no self-connection, $w_{ii} = 0$. The network works cyclically updating the states of the neurons.
Hopfield Neural Network

• The retrieval process is iterative and recurrent
• D. Hopfield showed in that this retrieval process always converges. A set of states of all the neurons on the $t^{th}$ cycle is called a state of the network.

• The state (output) of the $j$th neuron at cycle $t + 1$ is

$$ s_j(t+1) = \varphi \left( w^j_0 + \sum_{i \neq j} w_{ij} s_i(t) \right) $$

• The network state on $t^{th}$ cycle is the network input for the $t+1^{st}$ cycle.
Hopfield Neural Network

- The network is characterized by its energy corresponding to the current state:

\[ E_t = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i(t) s_j(t) + \sum_i w_0^i s_i(t) \]

- Updating its states during the retrieval process, the network converges to the local minimum of the energy function, which is a stable state of the network.

- Once the network reaches its stable state, the retrieval process should be stopped.

- In practical implementation, the retrieval process should continue either until a means square error (MSE) or root mean square error (RMSE) between the states on cycle \( t \) and \( t+1 \) drop below some pre-determined minimum.
Hopfield Neural Network

• It is important to mention that the Hopfield neural network not only is the first comprehensively developed neural network with a learning algorithm, which does not depend on the network size and an input/output mapping

• It also stimulated active research in areas of neural networks and dynamical systems in general

• D. Hopfield generalized later all principles that he developed for a network with binary inputs/outputs for a network with real-valued inputs/outputs having a continuous monotonic increasing and bounded activation function (a typical example is a sigmoid activation function)