AI 2: Artificial Neural Networks, ANN

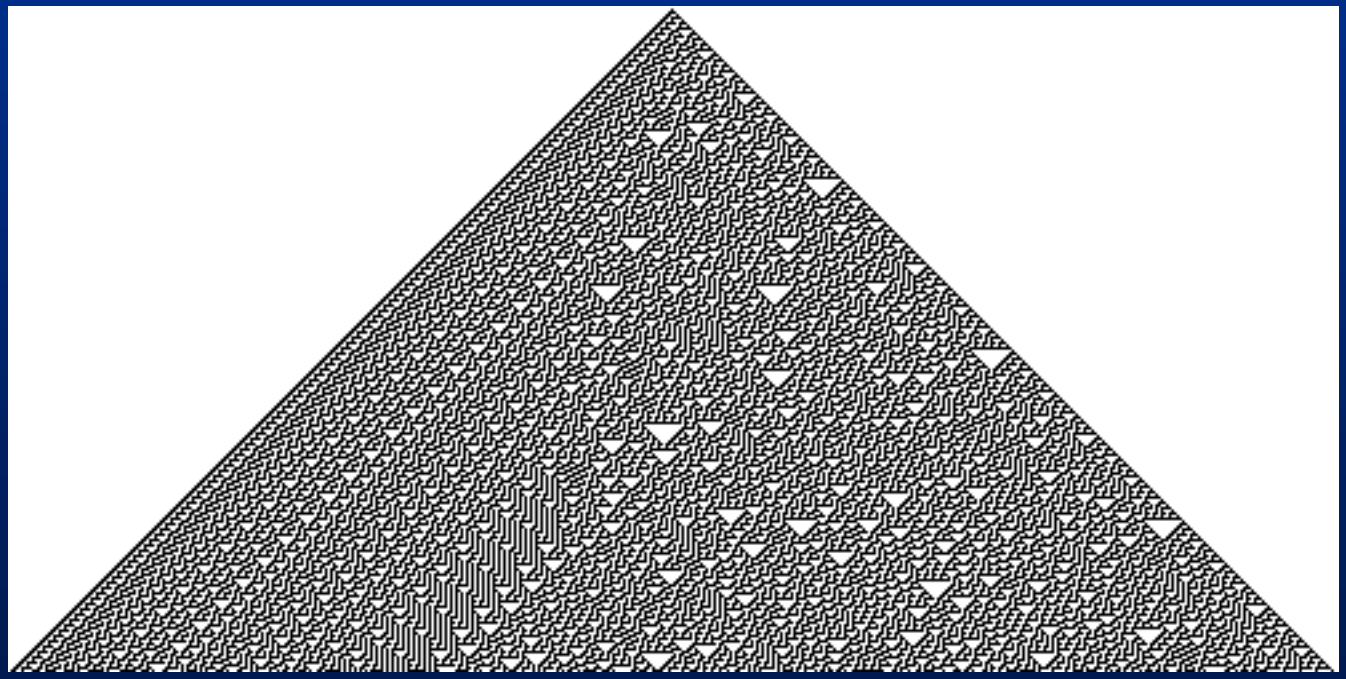
Anders Brodin

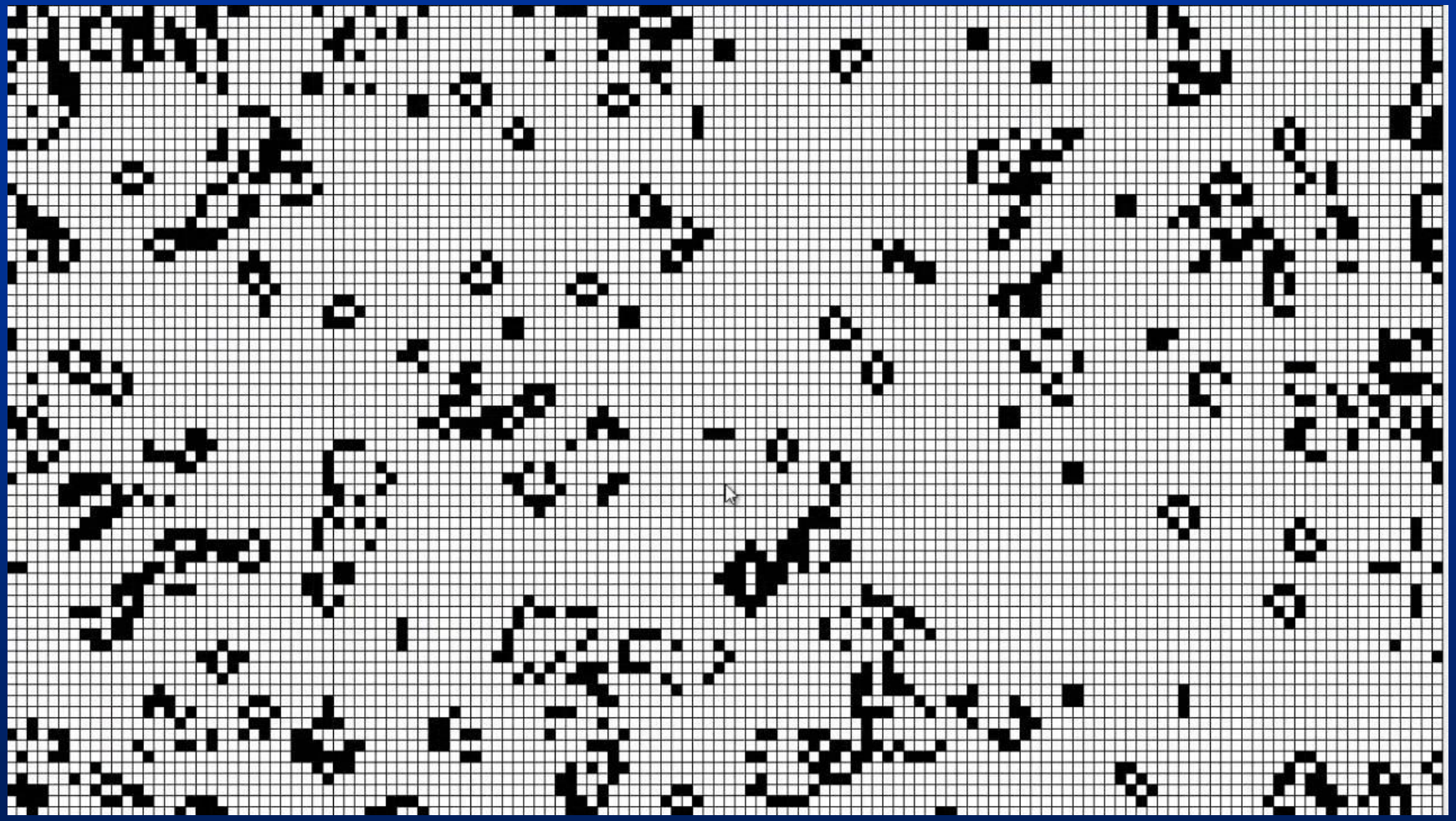
Evolutionary Ecology

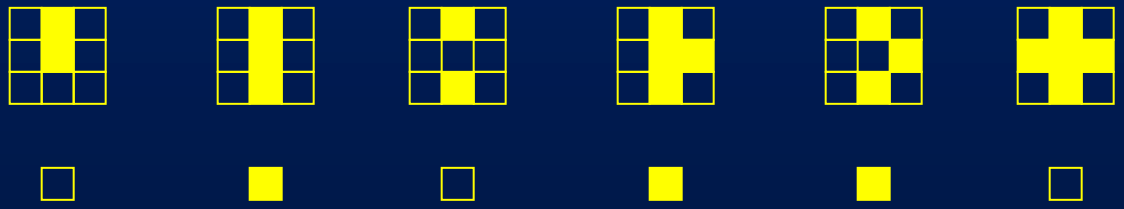
Lund University

Cellular automata - global effects from local rules:

- How does a one-dimensional automat work?
- How does a two-dimensional automat work?
- What is the ”game of life”
- What can cellular automata be used for?





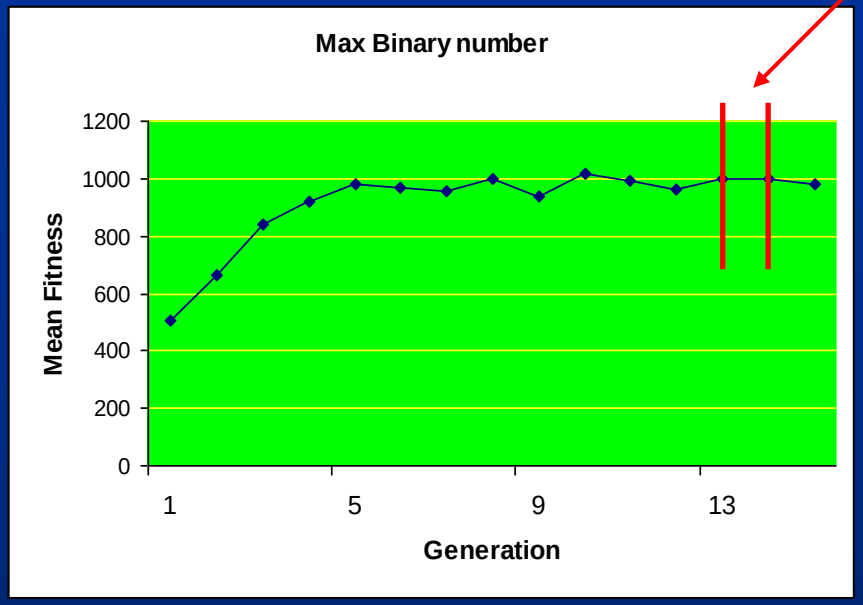


Genetic algorithms

1. Represent your problem as genes in a chromosome
2. Random generation of a population of chromosomes
3. Evaluate fitness of individual chromosomes
4. Sort after rank
5. Pairing, recombination
6. Insert offspring
7. Mutation

Cycle until terminantion

diff = $y_1 - y_2$




```
for (g = 1; g <= 15, g++)
   {cycle algorithm}

while (diff > 10)
   {double diff = y1 – y2;
    cycle algorithm
   }
```

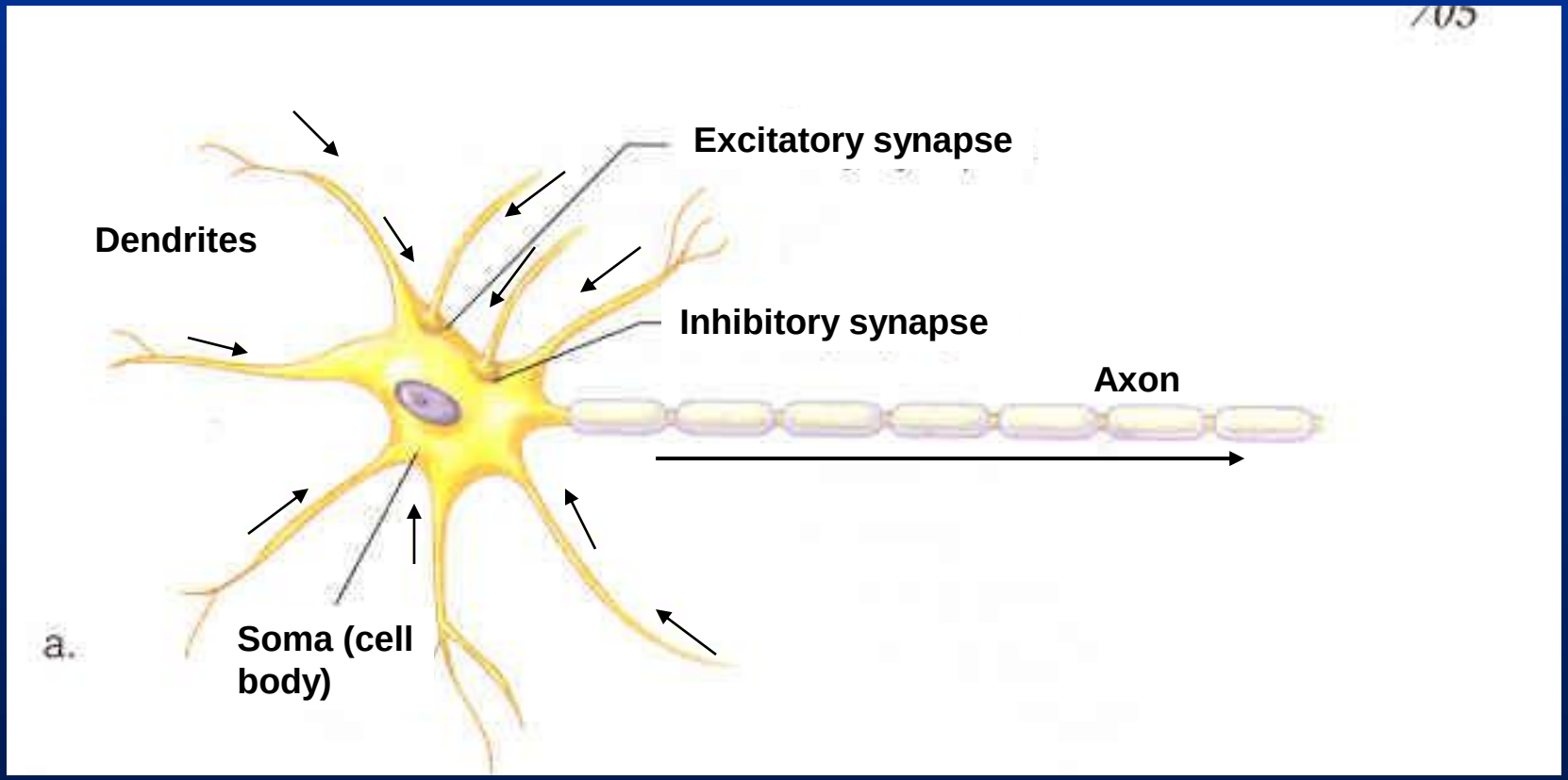
g = generation

Continuous vs discrete genetic algorithms:

Discrete (0 1 1 0 1)	Continuous (0.654, 3.564…)
binary	real numbers
goal function	direct use of numbers
simple crossing over	complex mating
traditional, original	newer

Artificial neural networks, ANN

Real neuron:

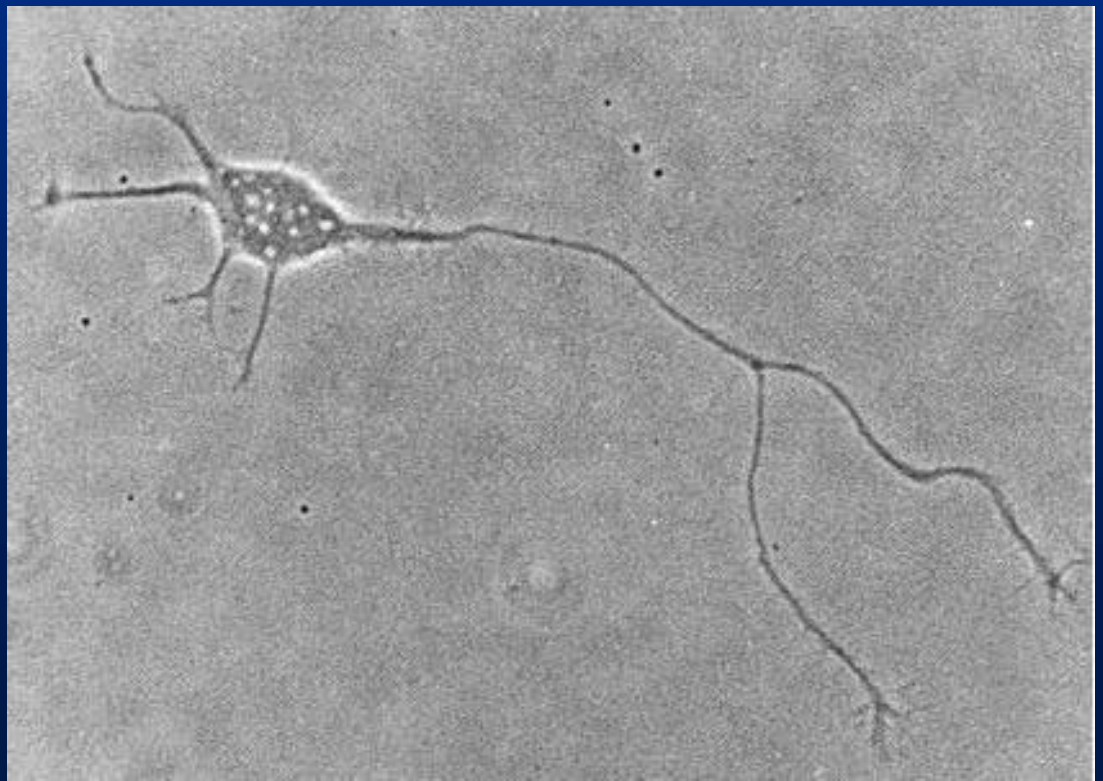


Neuron signals (firing):

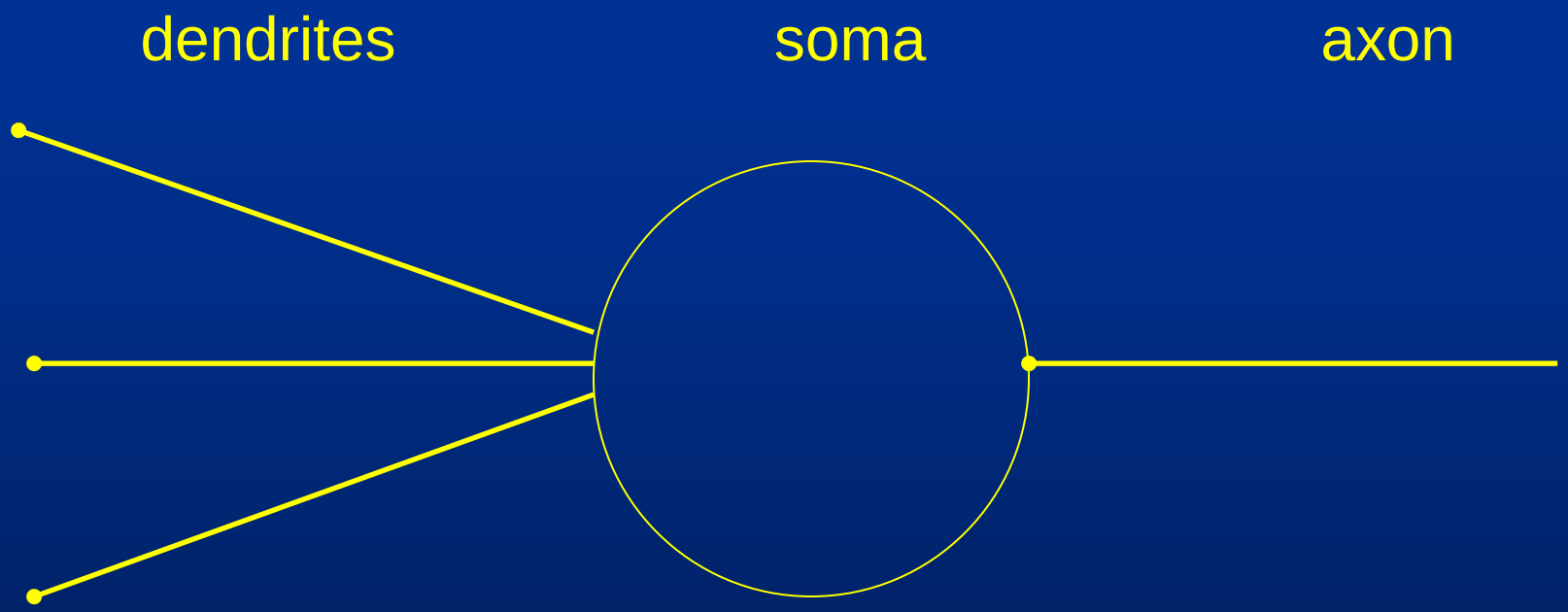
- a neuron accumulates signals from synapses on dendrites and soma

- fires at some threshold

- spike that propagates to other neurons in axon

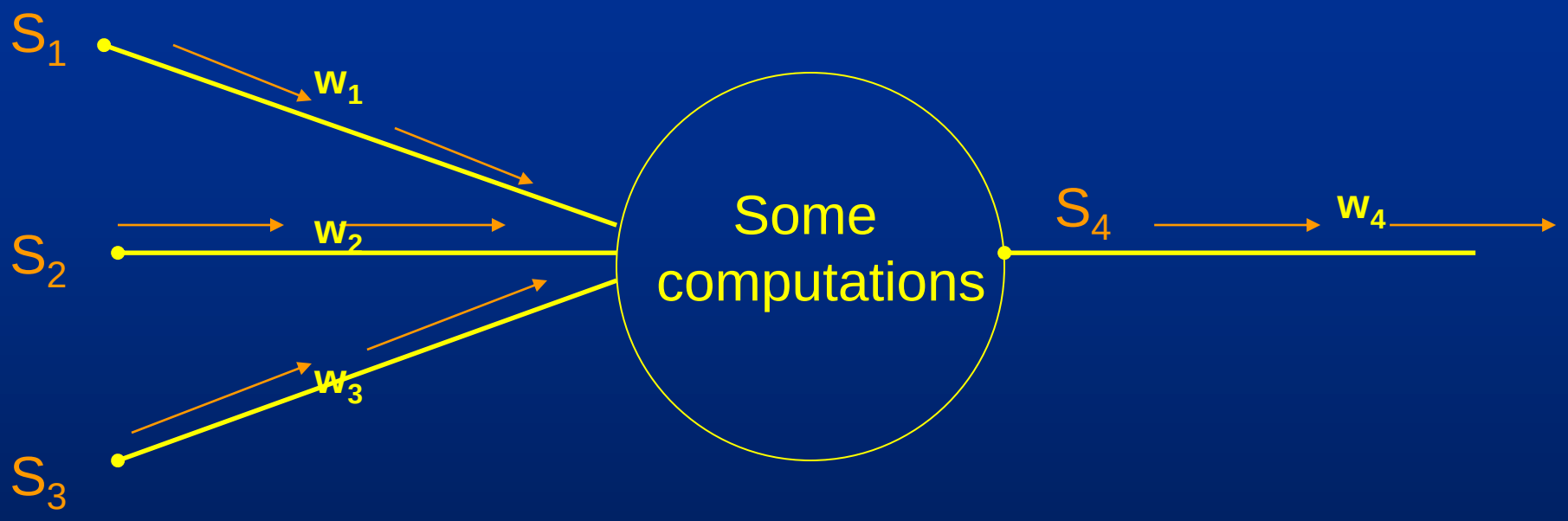


Artificial neuron:

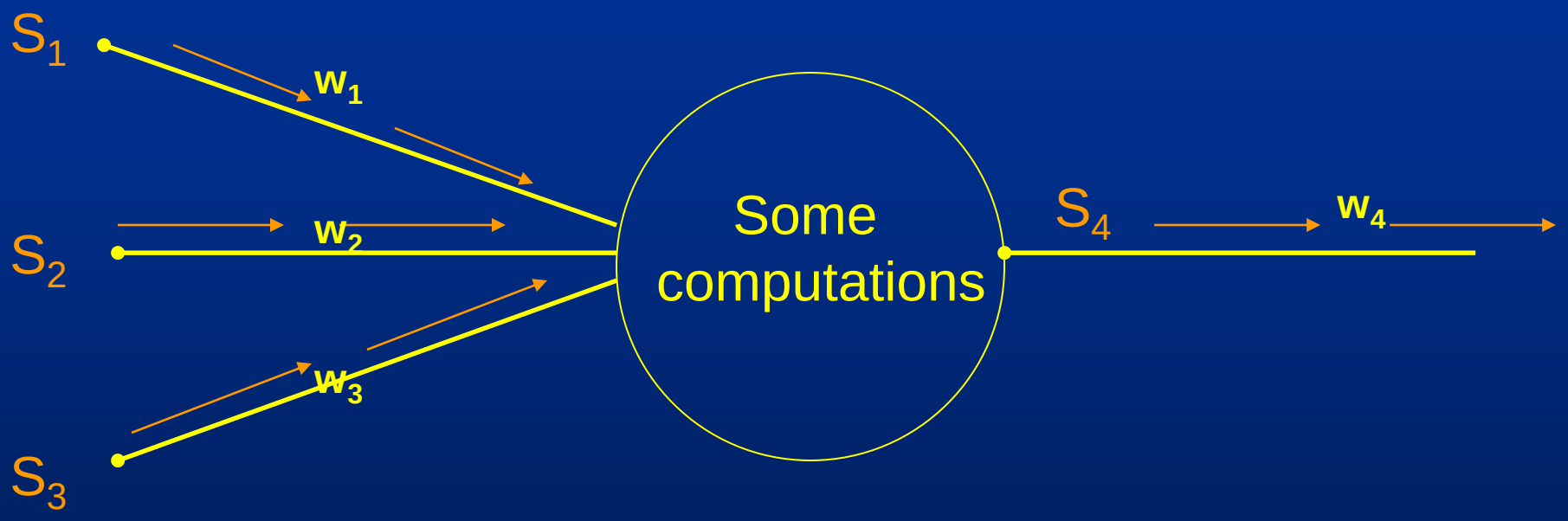


dendrites = axons!

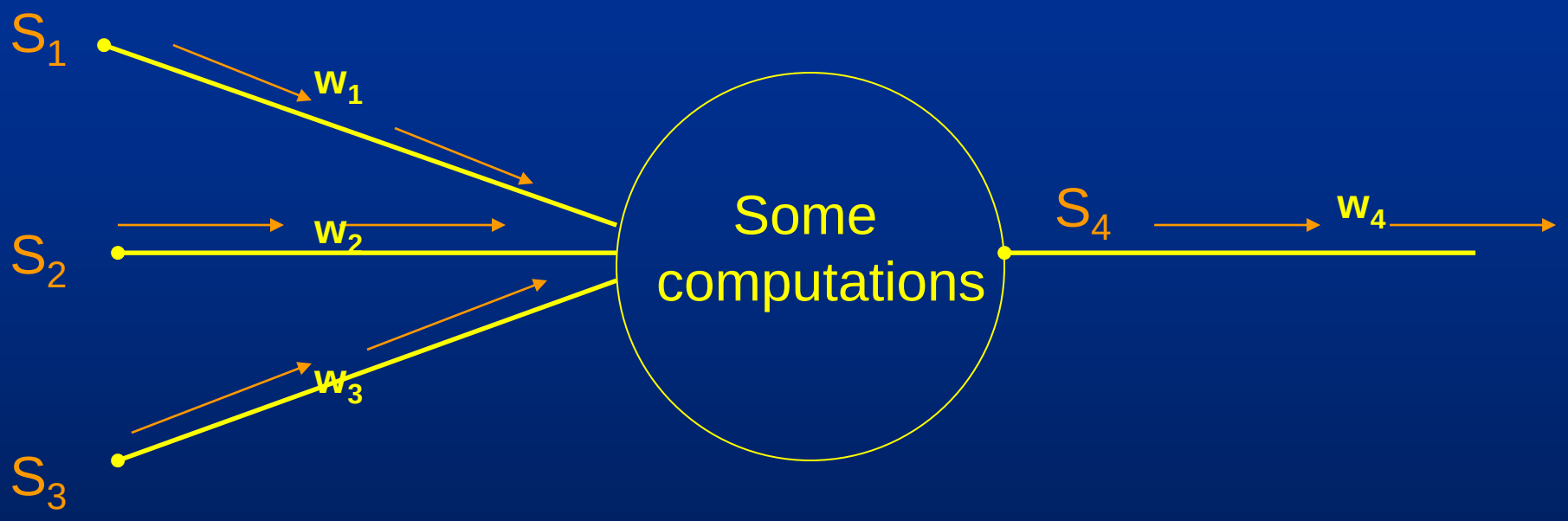
Dendrites and axons in real neurons, weights (w) in artificial ones:



Input: Three input signals, S_1,S_2,S_3



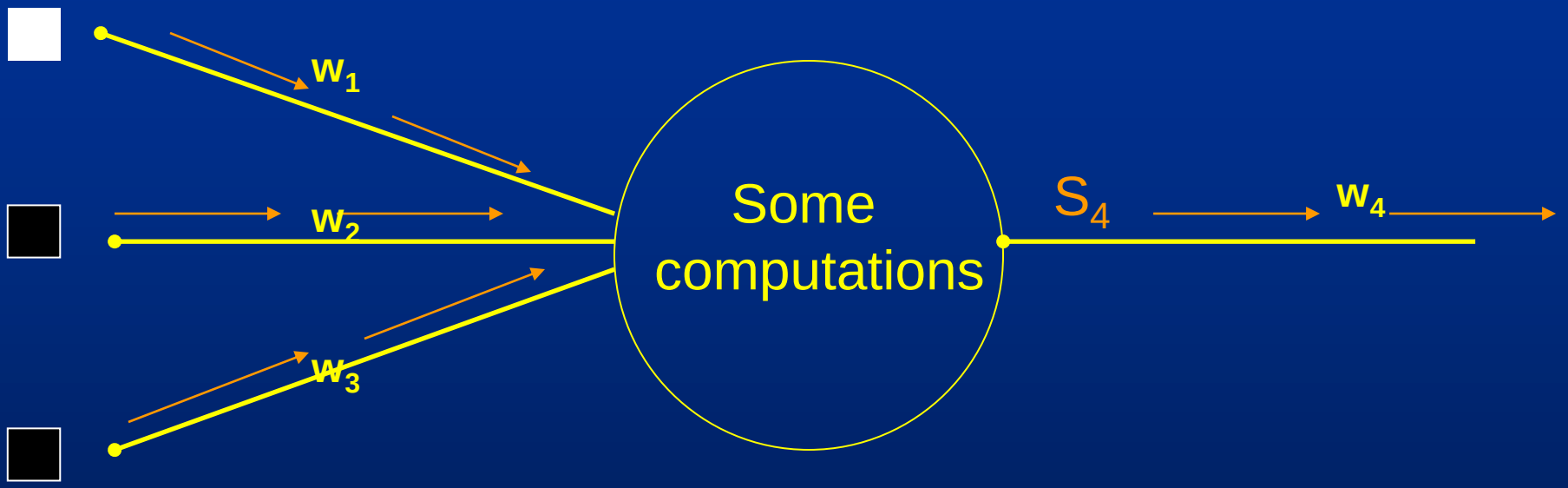
Input: Three input signals, S_1,S_2,S_3



Output: one signal, S_4

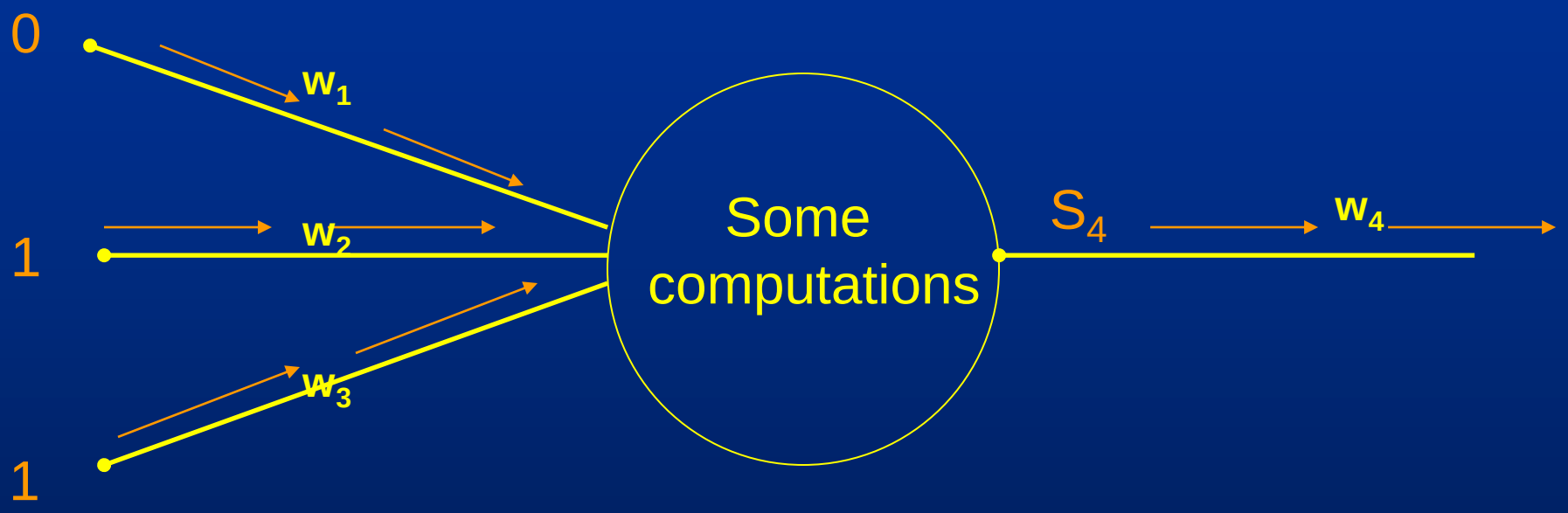
Input: Three input signals, S_1,S_2,S_3

Could represent white - black – black…

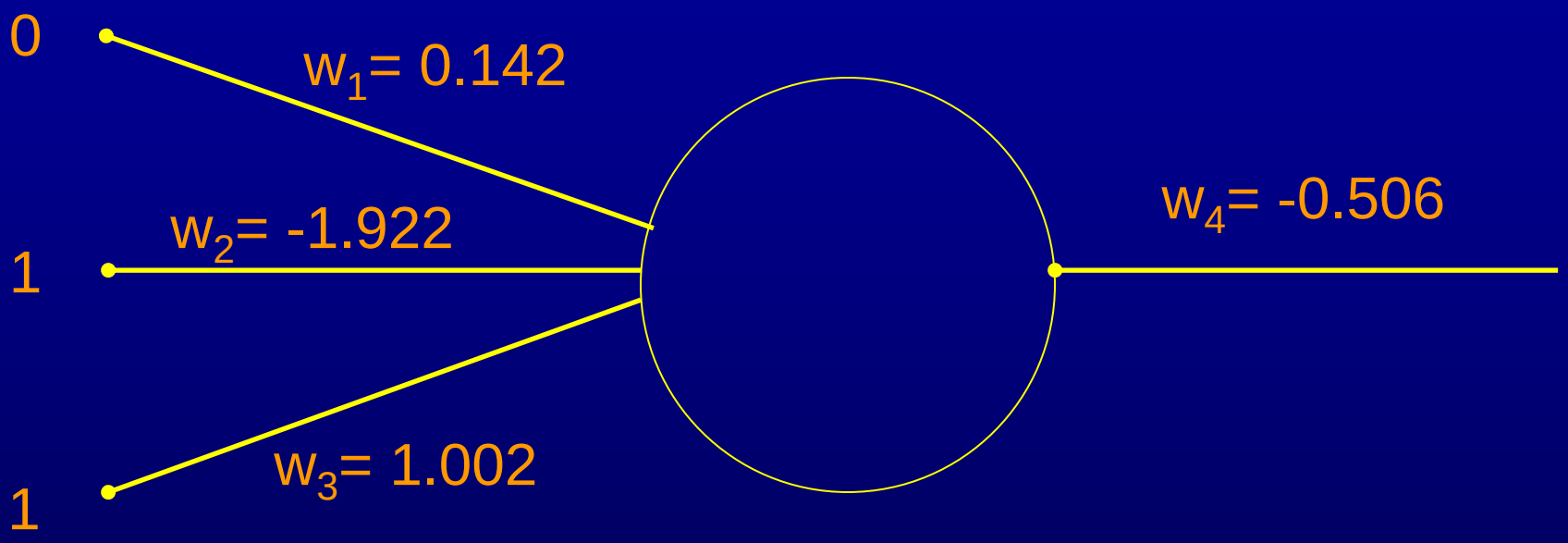


Input: Three input signals, S_1,S_2,S_3

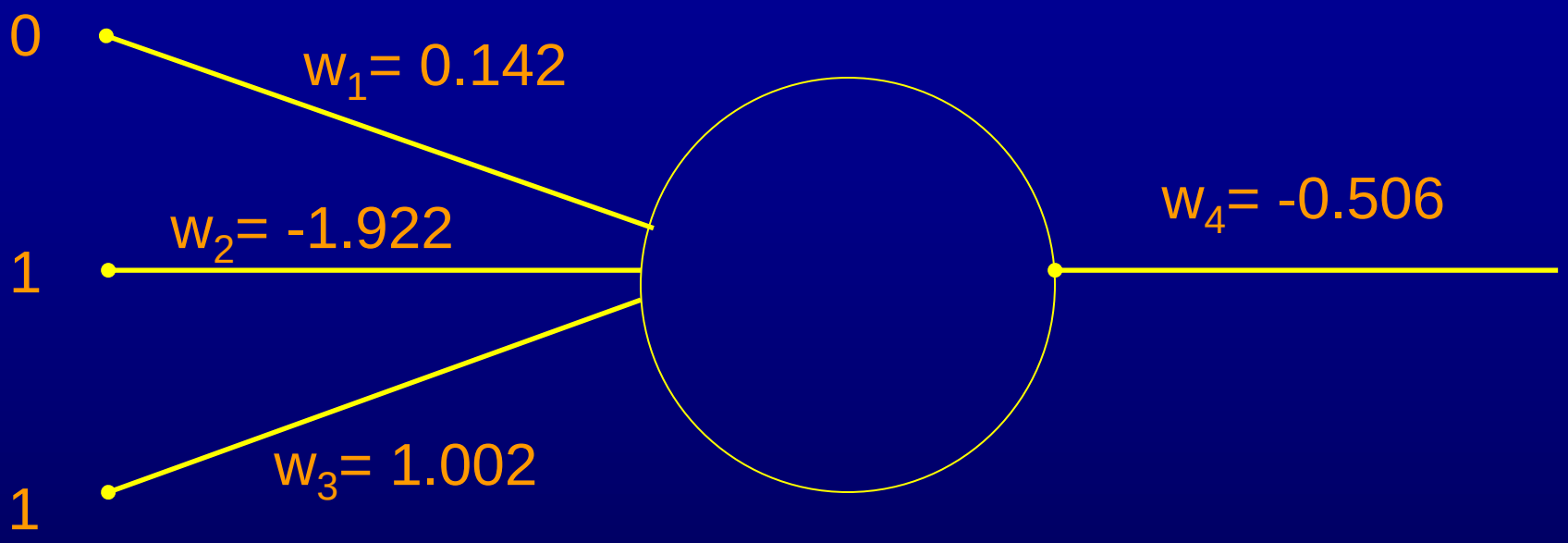
…or 0 1 1



Set weights to random decimal numbers

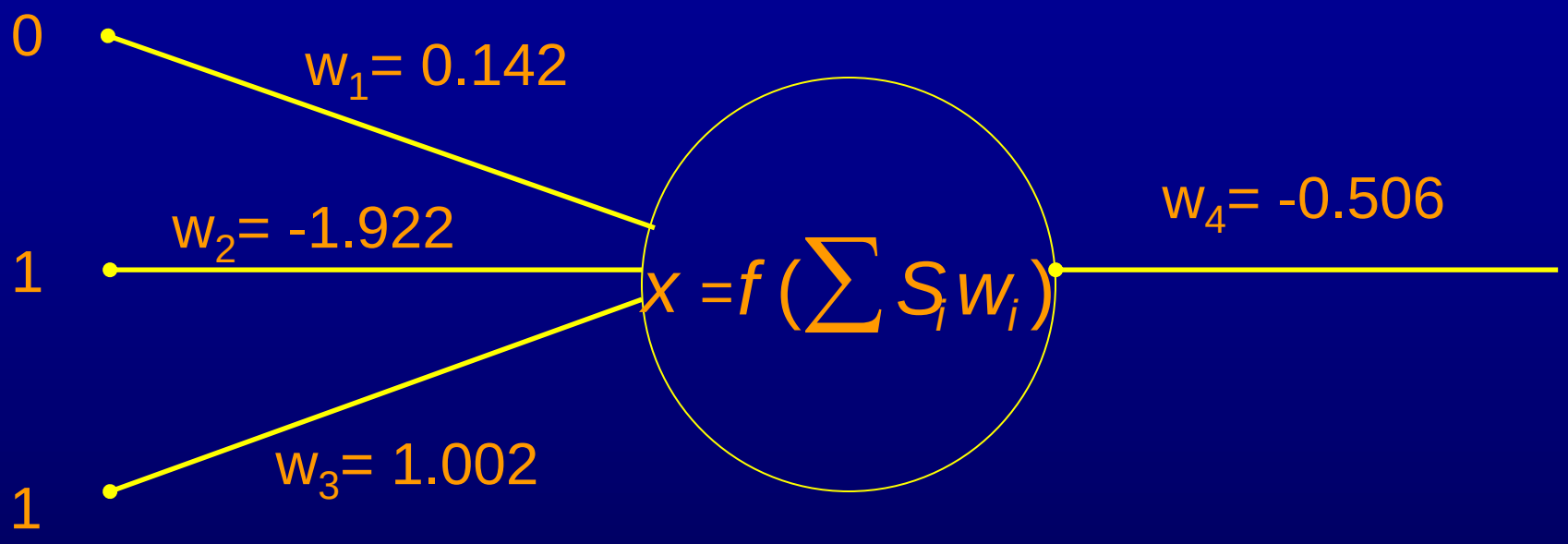


Multiply with input signals



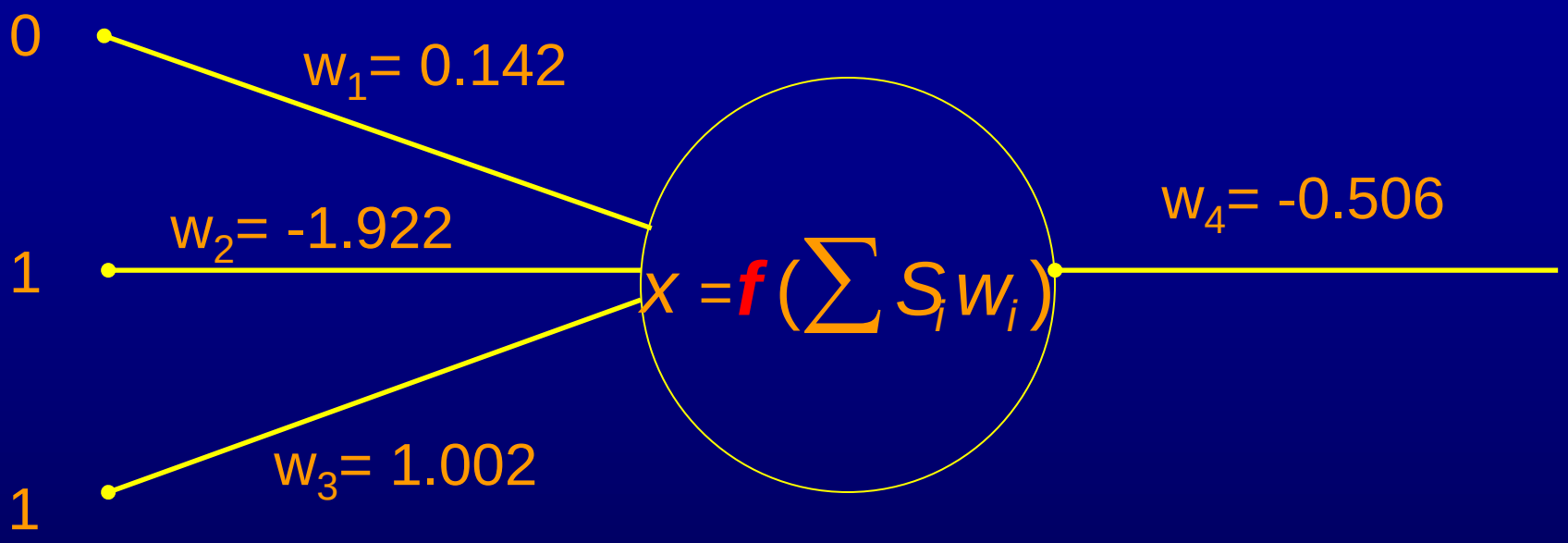
$$\sum S_i w_i = 0 * 0.142 + 1 * (-1.922) + 1 * 1.002$$

This happens in the ”soma” = node = neuron body



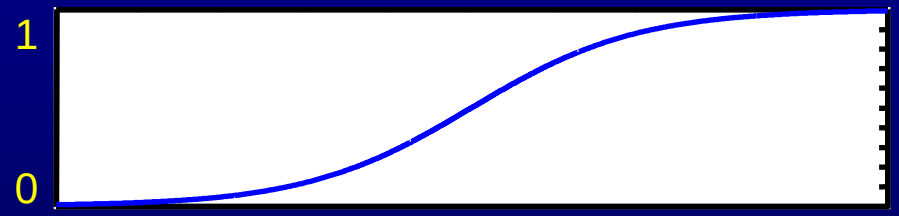
$$\sum S_i w_i = 0 * 0.142 + 1 * (-1.922) + 1 * 1.002$$

Activation function decides how signal propagates

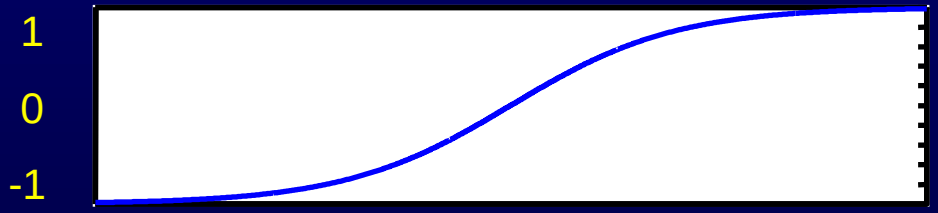


Activation (or transfer) functions

$$f(T) = \begin{cases} 0 \text{ if } T \leq 0 \\ 1 \text{ if } T > 0 \end{cases}$$



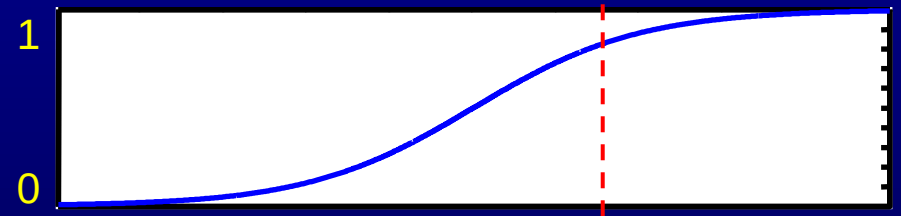
$$f(T) = \frac{1}{1 + e^{-T}}$$



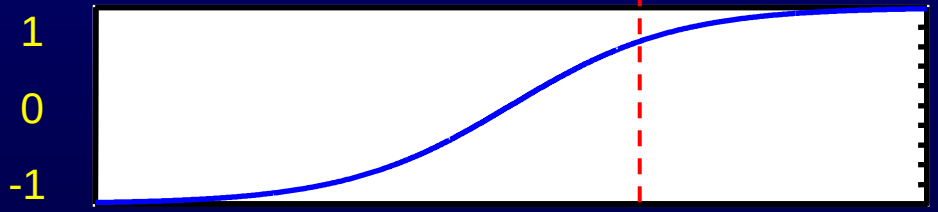
$$f(T) = \tanh(x)$$

Activation (transfer) functions

$$f(T) = \begin{cases} 0 \text{ if } T \leq 0 \\ 1 \text{ if } T > 0 \end{cases}$$



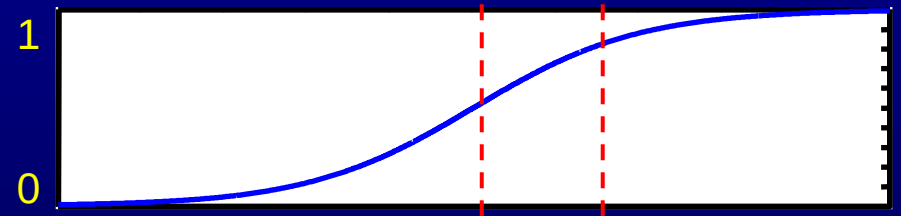
$$f(T) = \frac{1}{1 + e^{-T}}$$



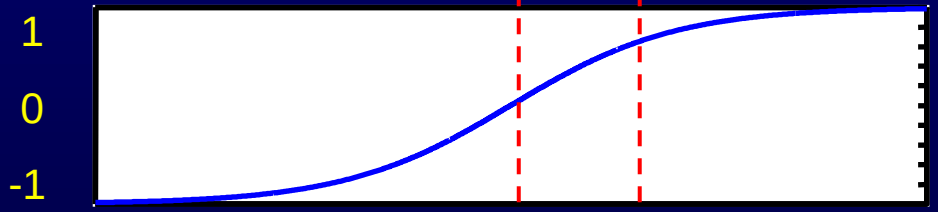
$$f(T) = \tanh(x)$$

Activation (transfer) functions

$$f(T) = \begin{cases} 0 \text{ if } T \leq 0 \\ 1 \text{ if } T > 0 \end{cases}$$

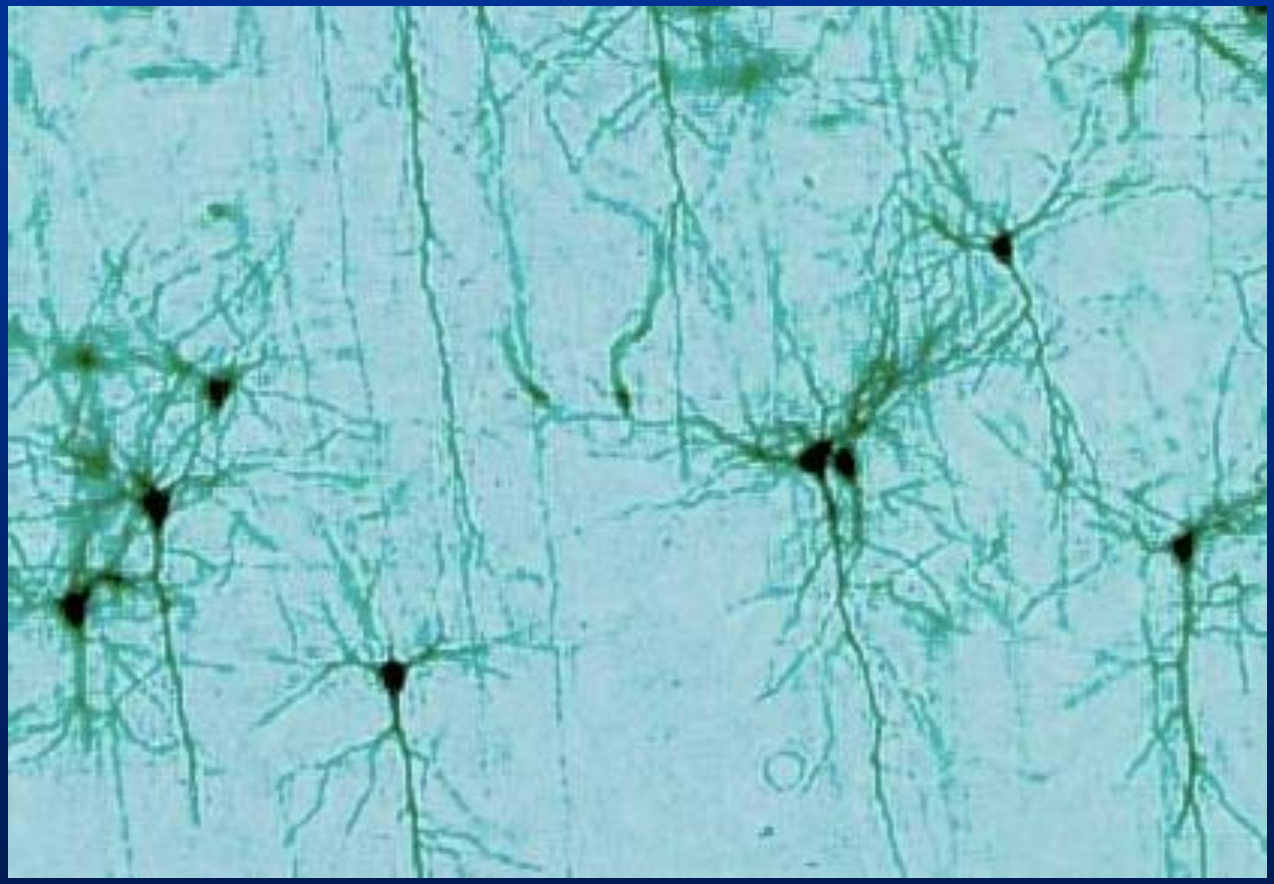


$$f(T) = \frac{1}{1 + e^{-T}}$$



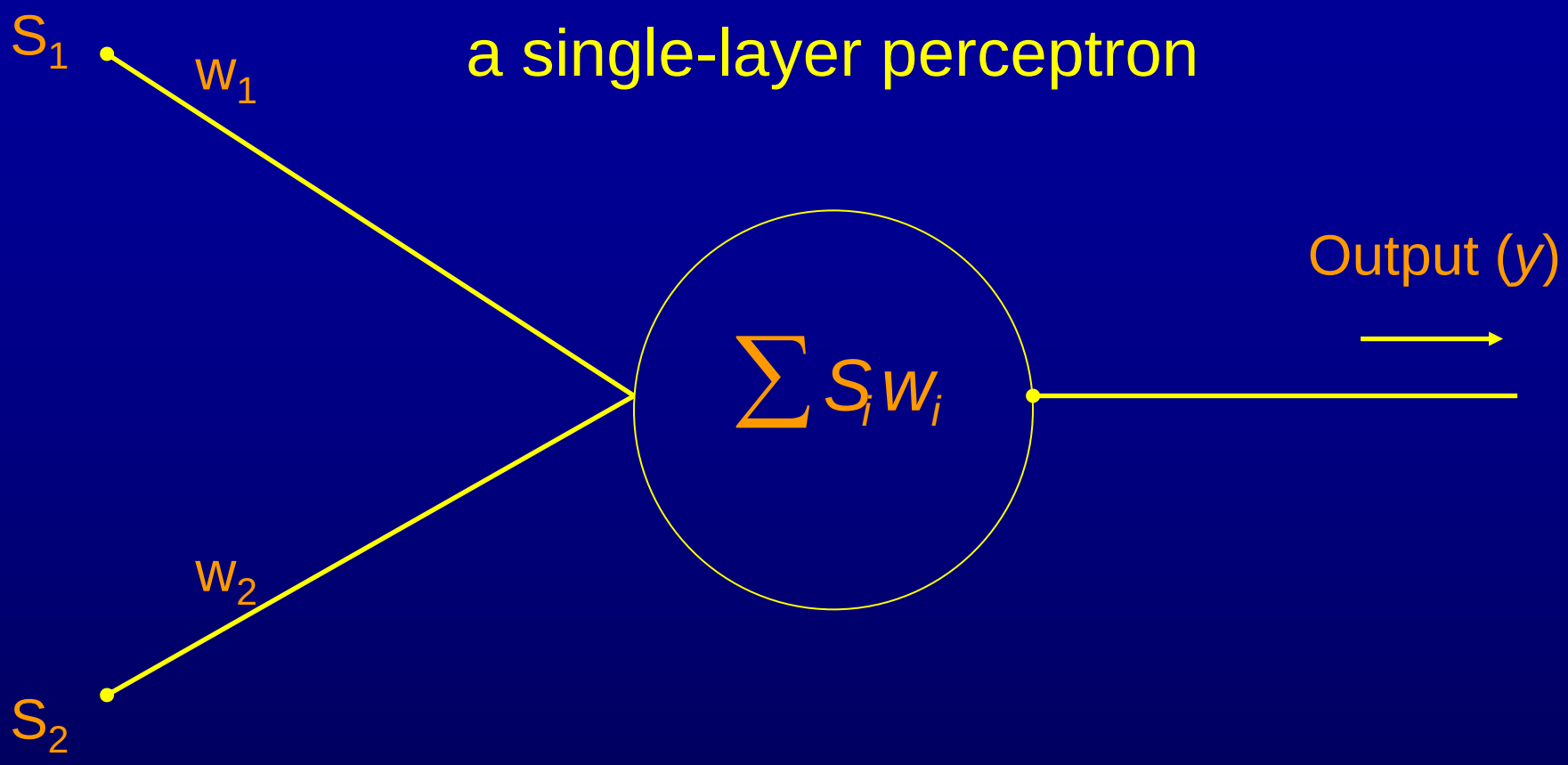
$$f(T) = \tanh(x)$$

Real network:

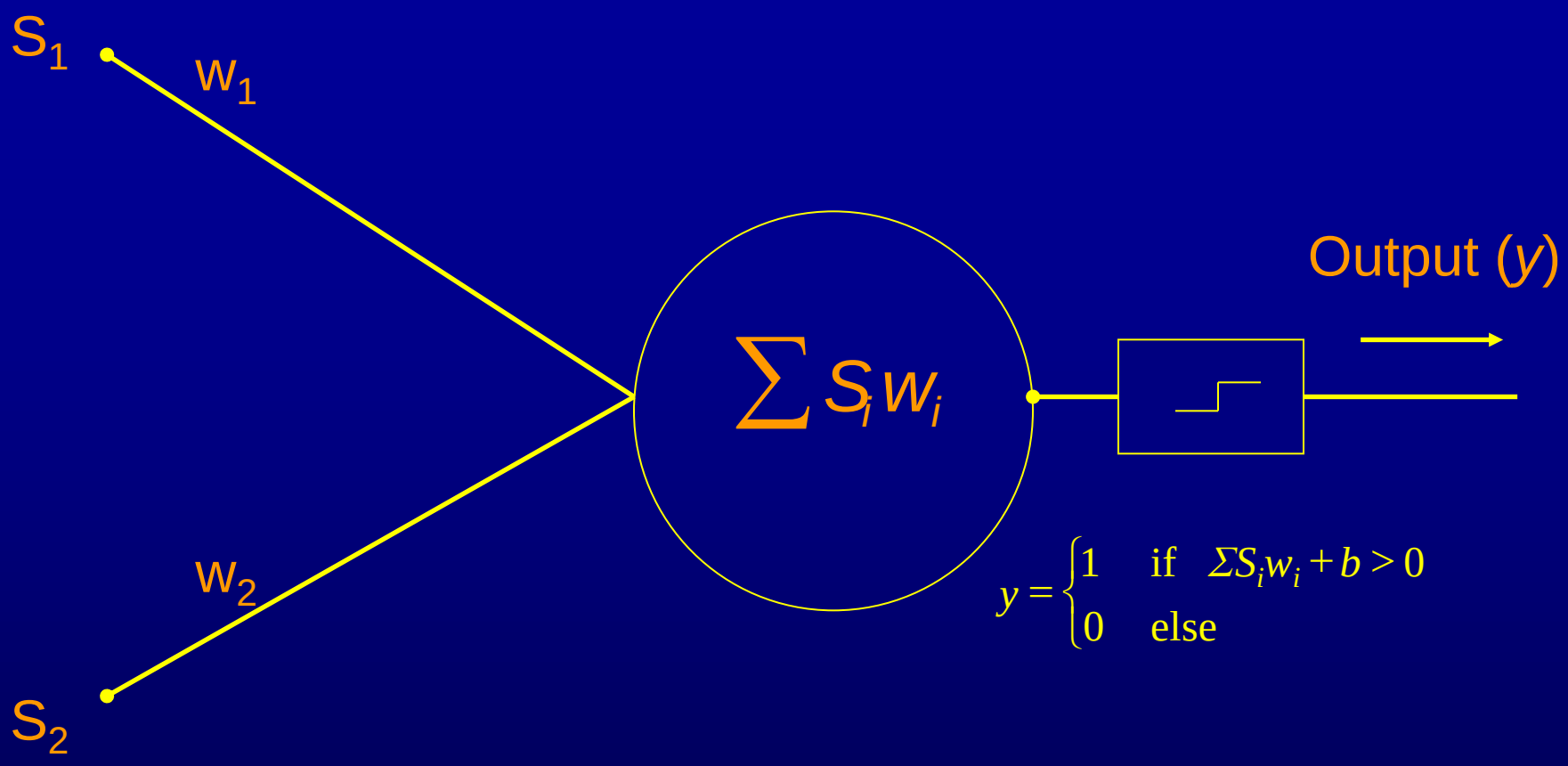


The simplest network:

a single-layer perceptron

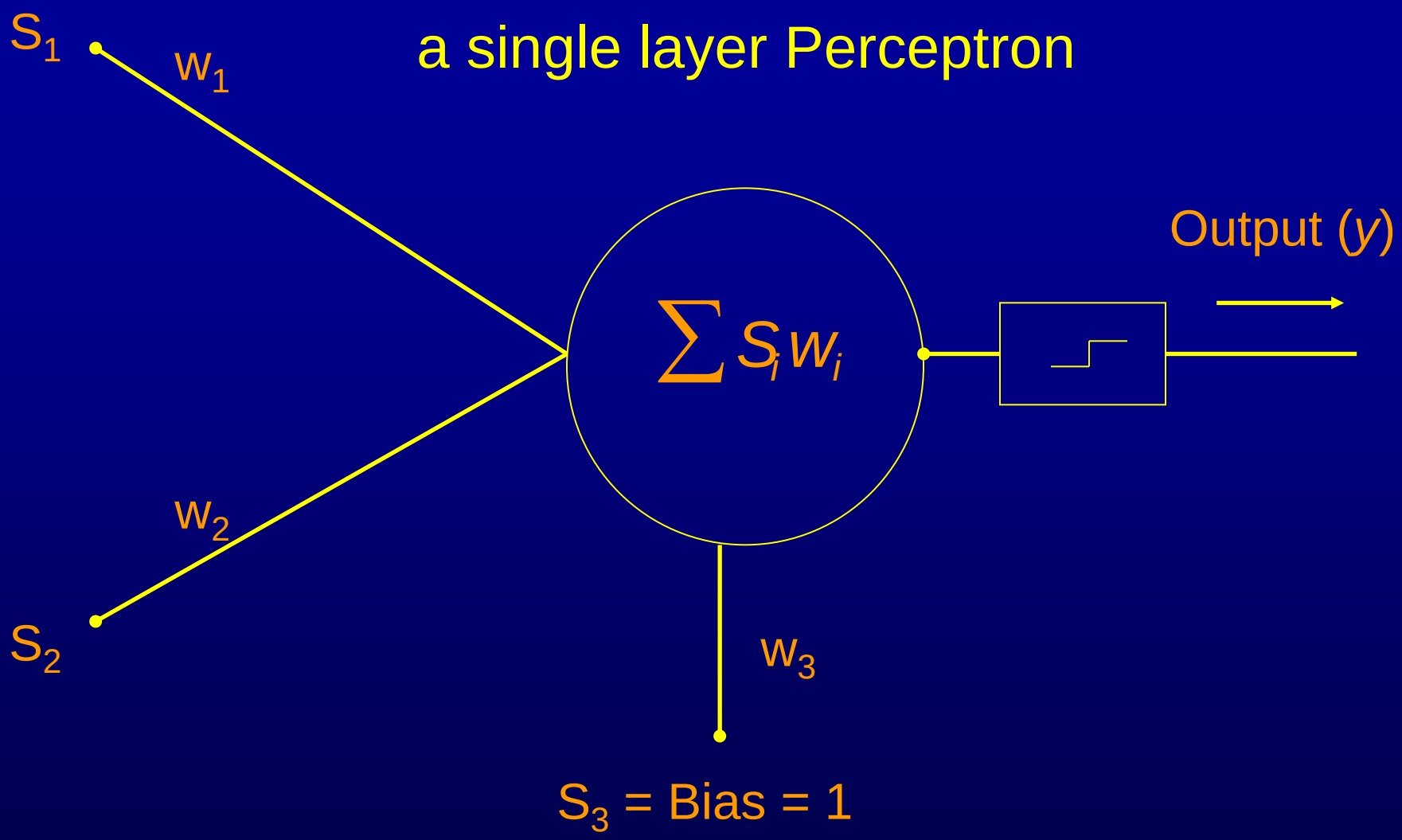


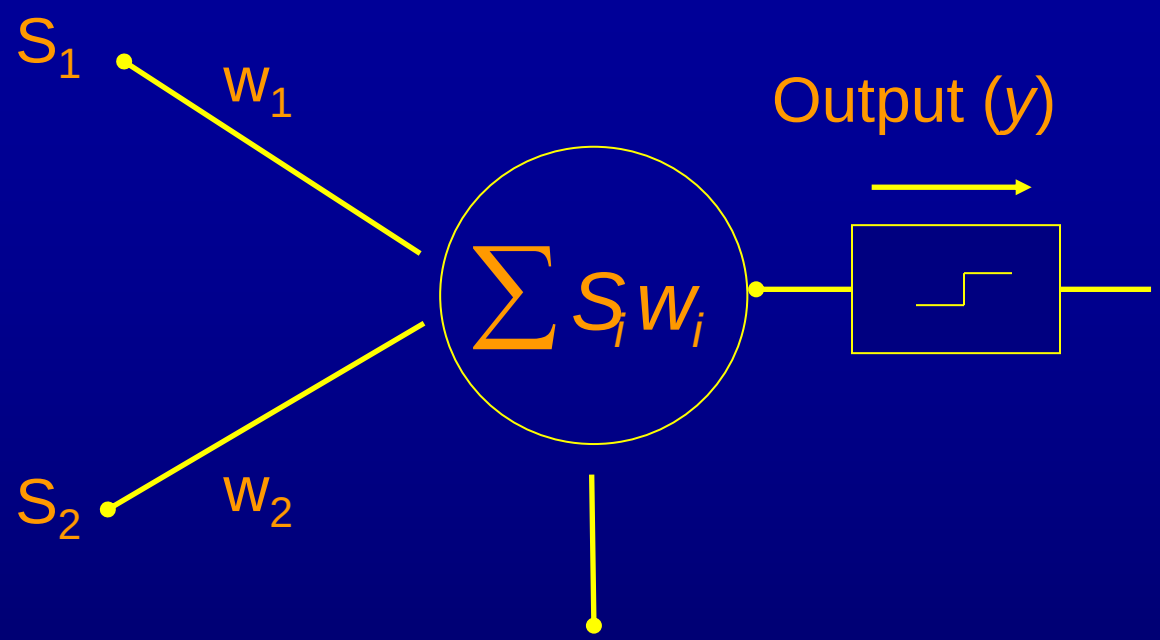
Activation function



The simplest network:

a single layer Perceptron





s_1	s_2	out
0	0	?
1	0	?
0	1	?
1	1	?

Training it on logical operators:

AND:

s_1	s_2	out
0	0	0
1	0	0
0	1	0
1	1	1

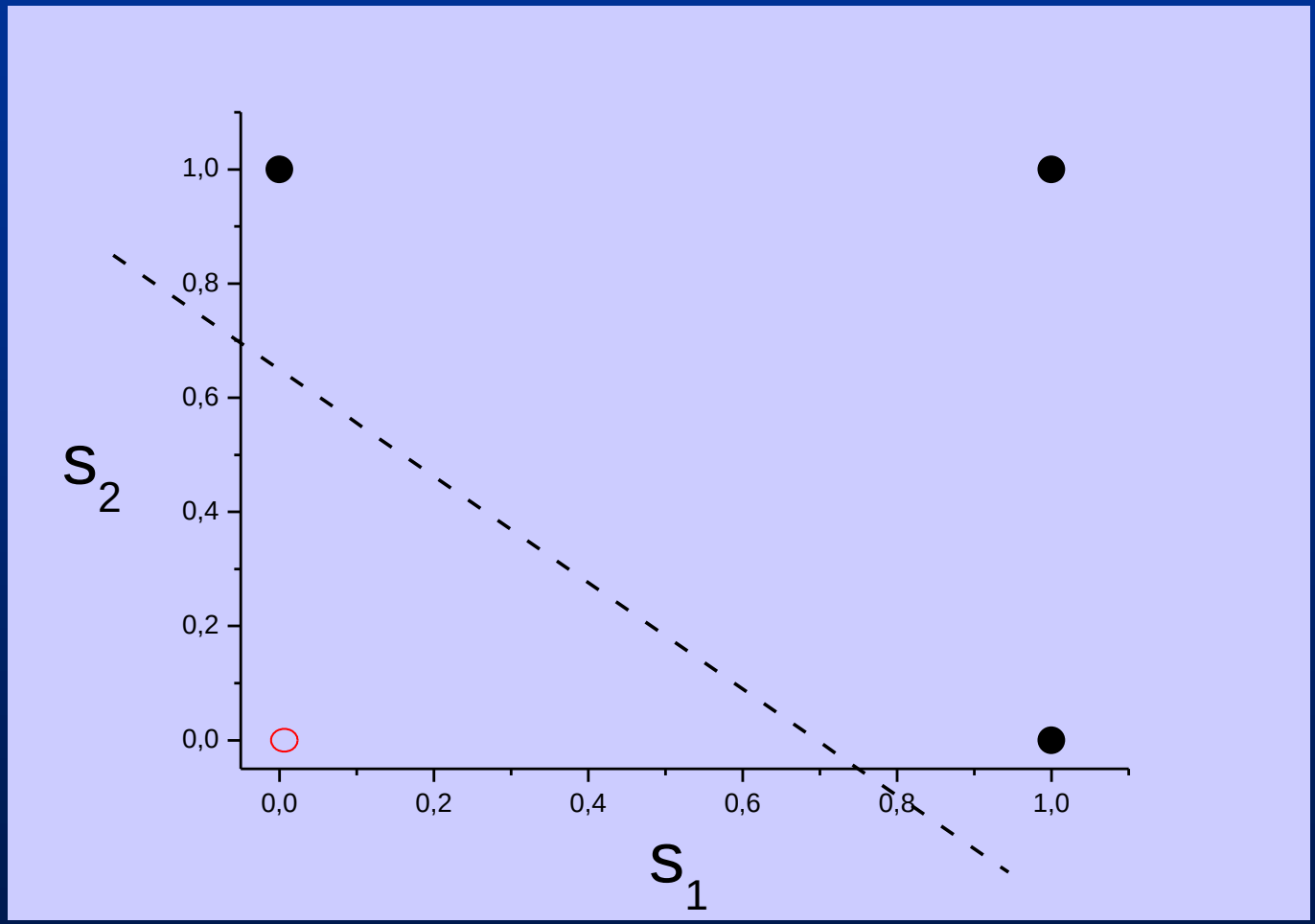
OR:

s_1	s_2	out
0	0	0
1	0	1
0	1	1
1	1	1

XOR:

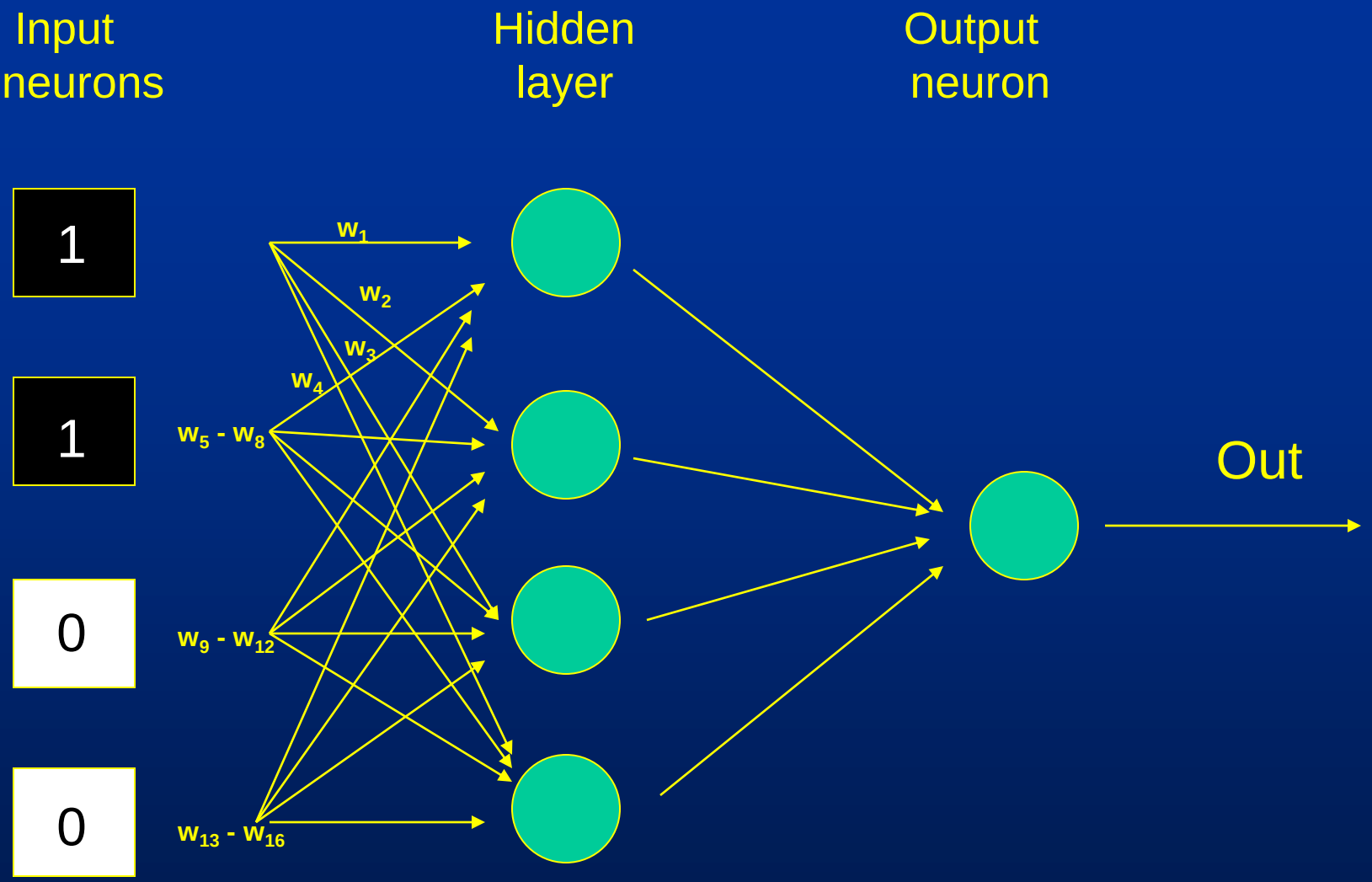
s_1	s_2	out
0	0	0
1	0	1
0	1	1
1	1	0

Graphical representation OR



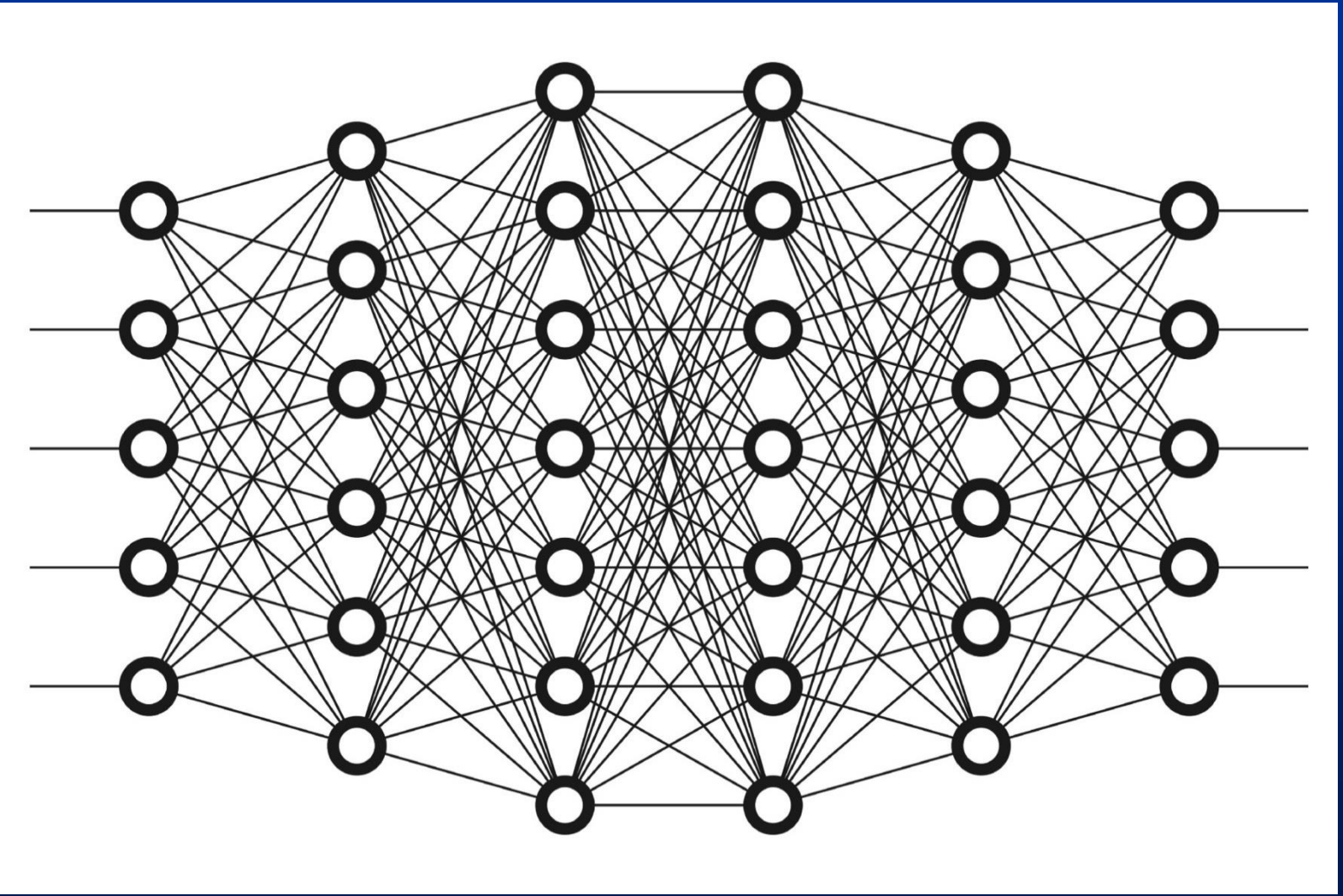
Question: Show the graphical representation of how a network should separate AND and XOR.

A larger network

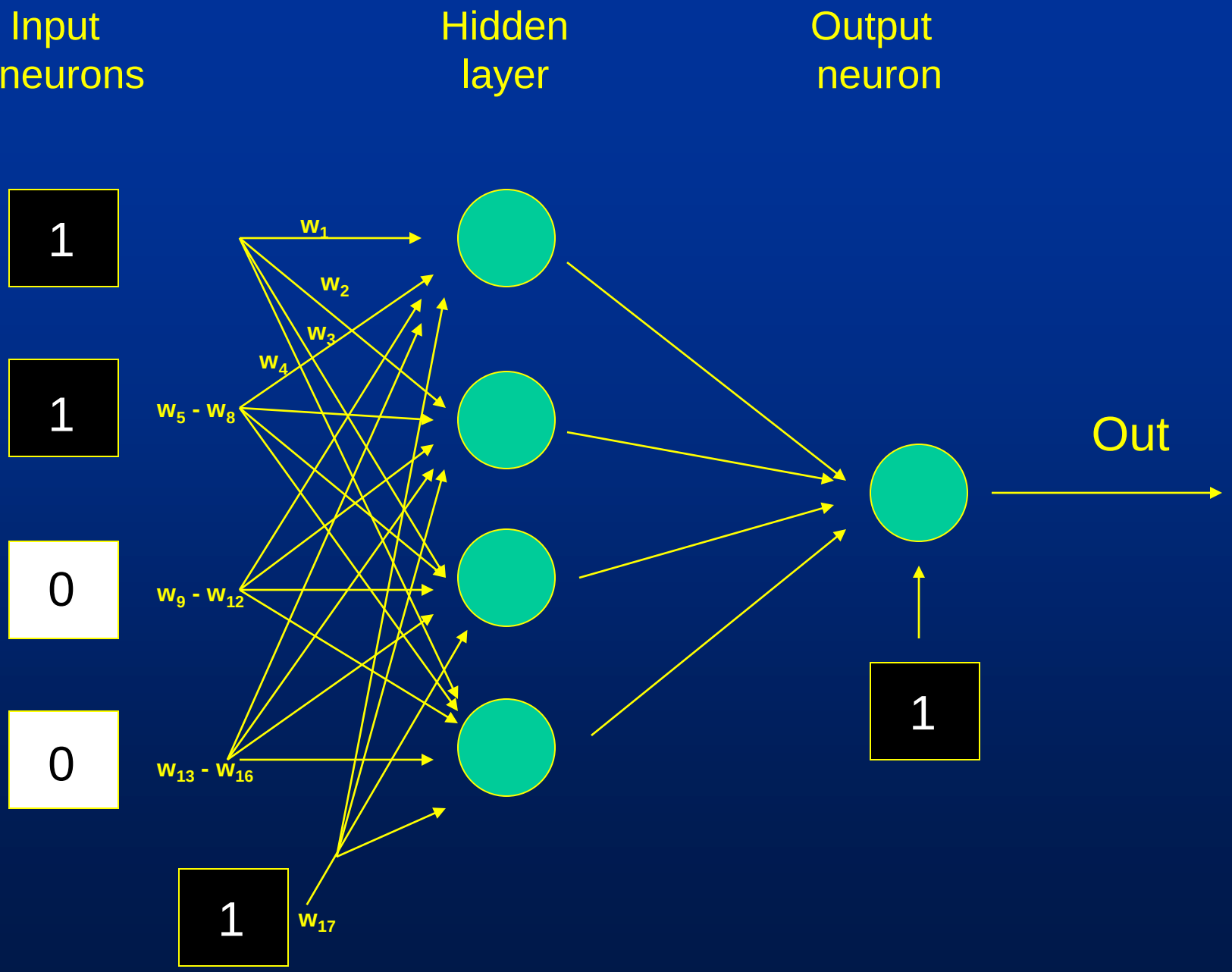


Deep learning:

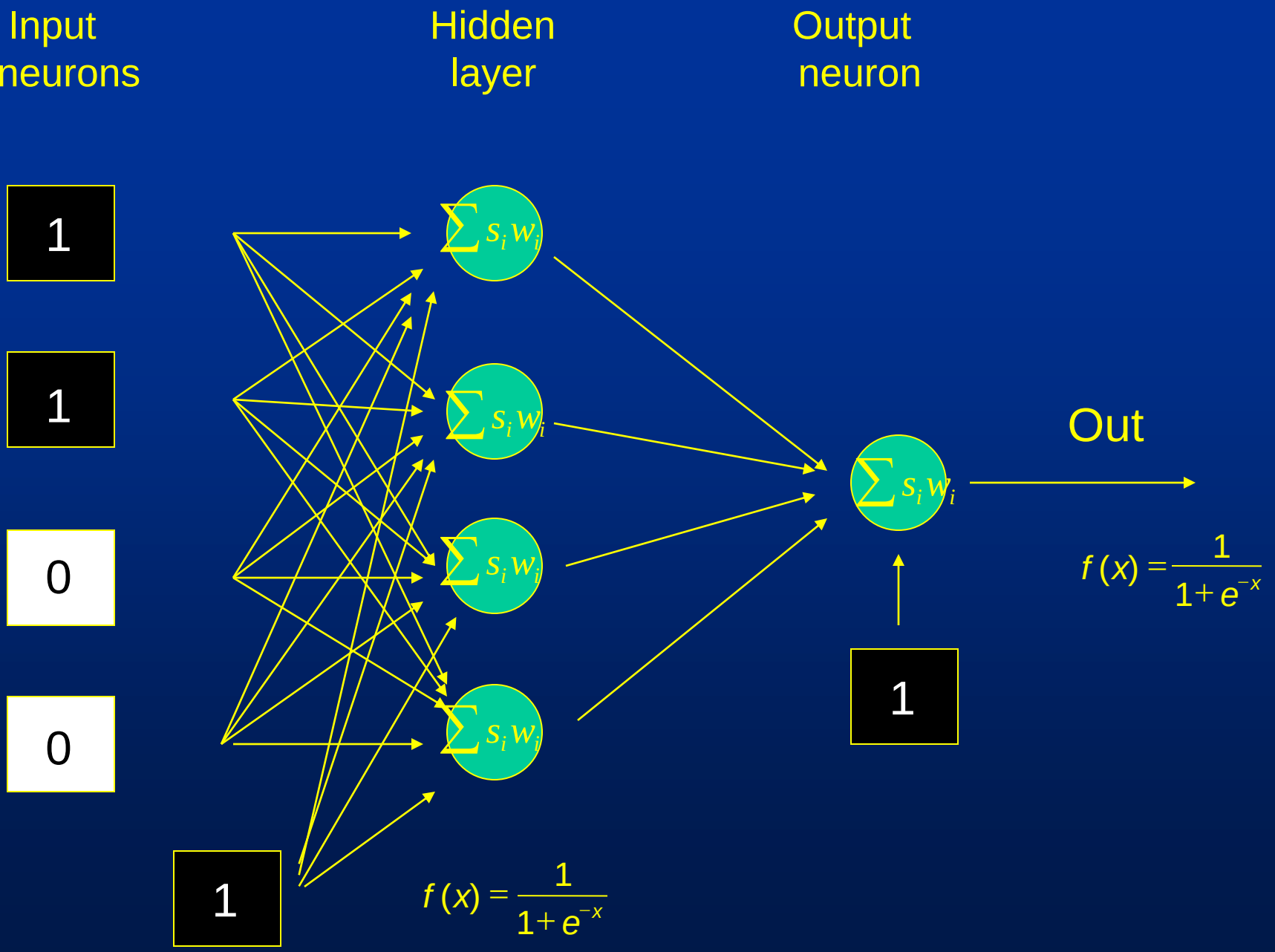
Layers: input hidden 1 hidden 2 hidden 3 hidden 4 output



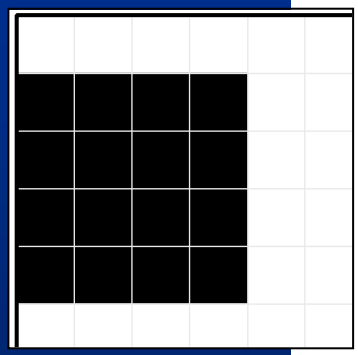
Adding biases:

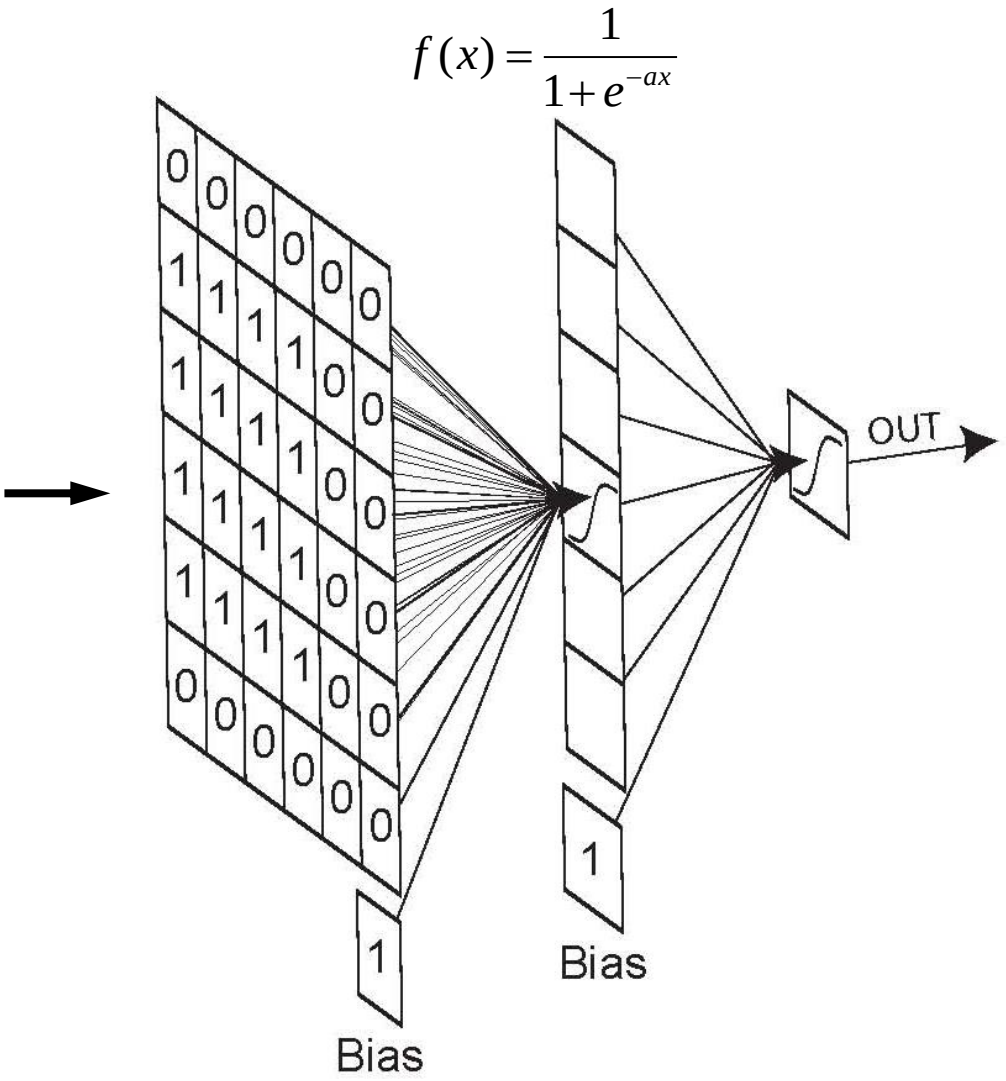


Adding activation functions:



Sending a black square to the network:



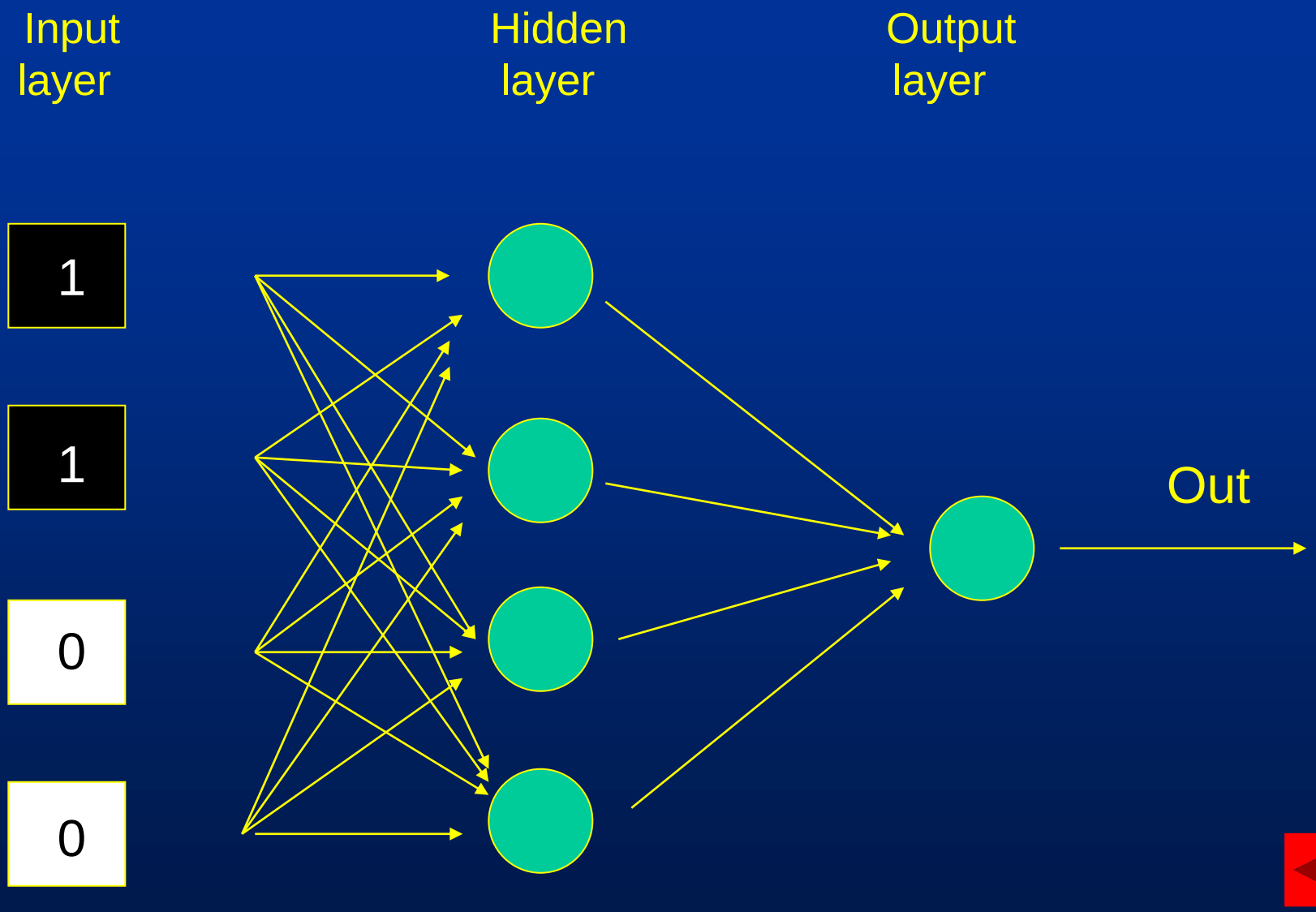


Question: Where in an ANN is the knowledge stored?

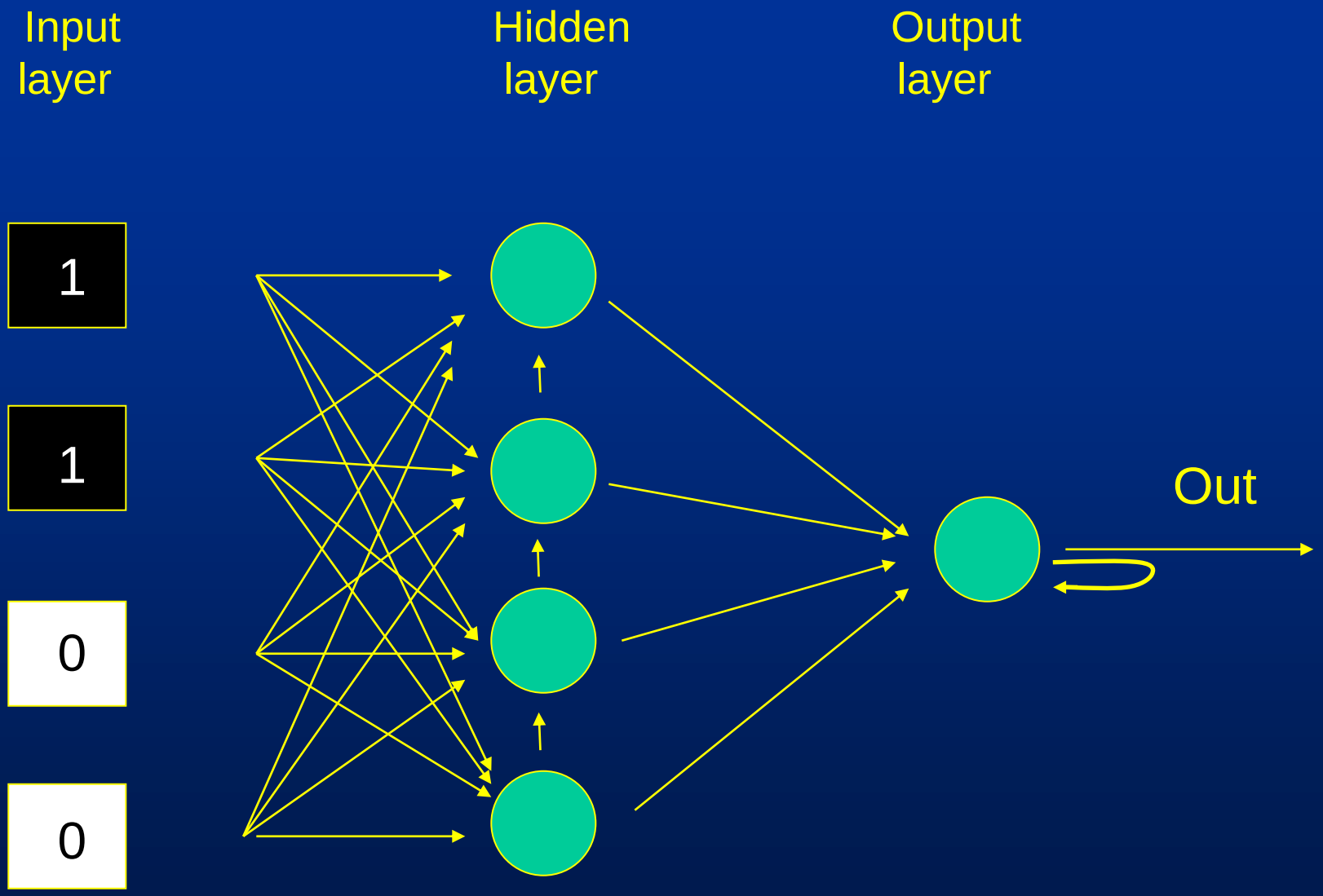
ANN Terminology:

- feedforward network
- counter propagation
- supervised or unsupervised training
- batch training, incremental training
- learning rules
- a biological application

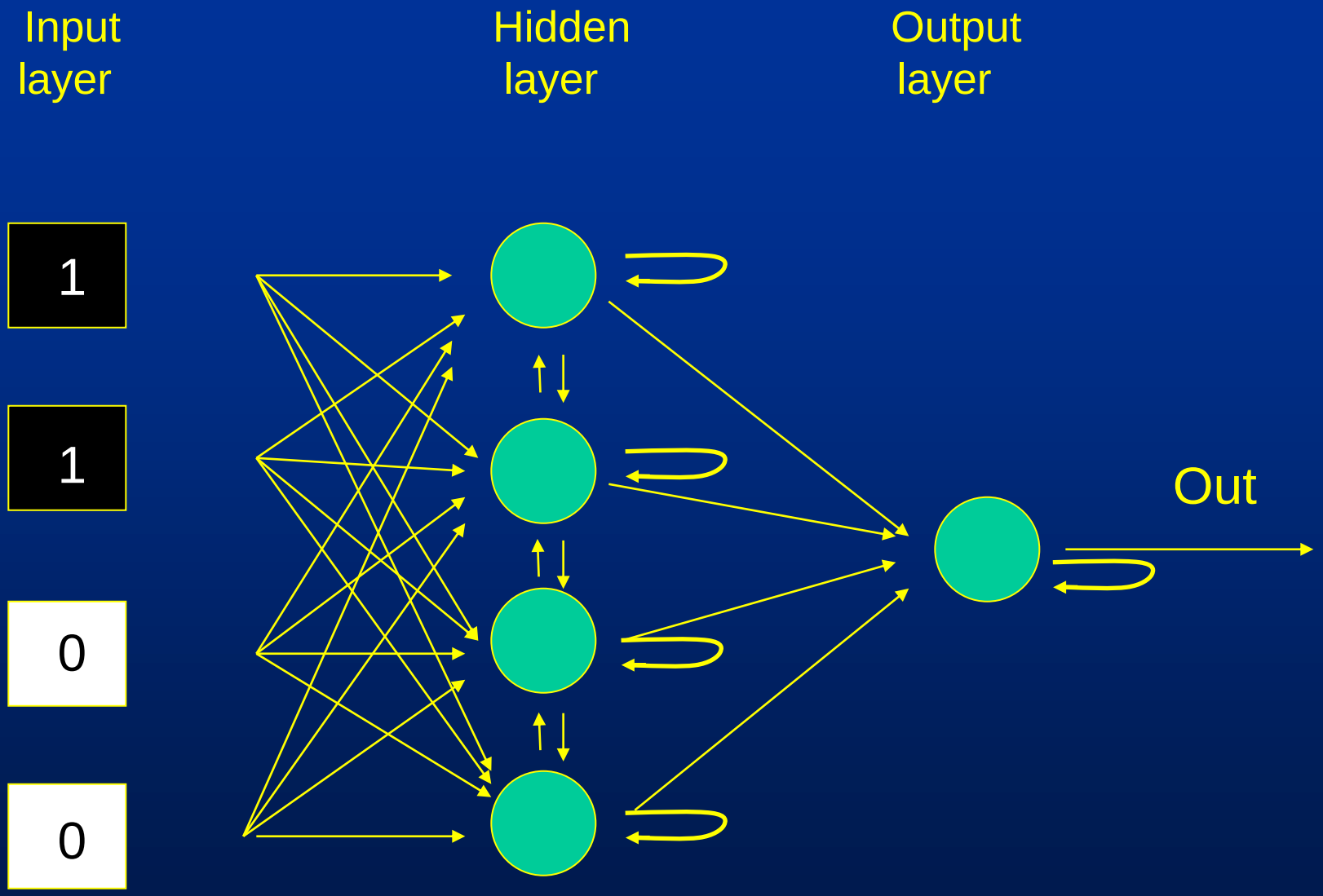
Feedforward network:



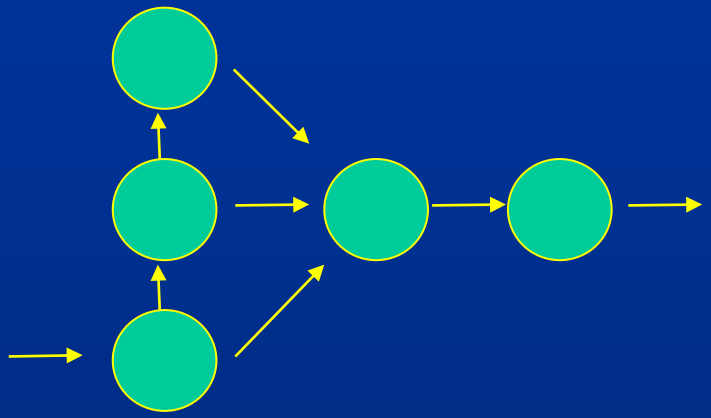
Counterpropagation:

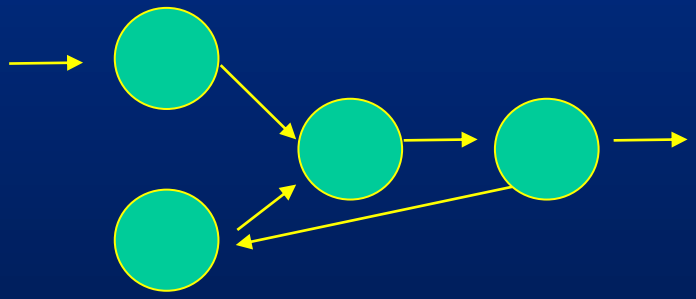


Counterpropagation:



Counterpropagation with various configurations:





Training the perceptron on logical operators:

AND:

x_1	x_2	out
0	0	0
1	0	0
0	1	0
1	1	1

OR:

x_1	x_2	out
0	0	0
1	0	1
0	1	1
1	1	1

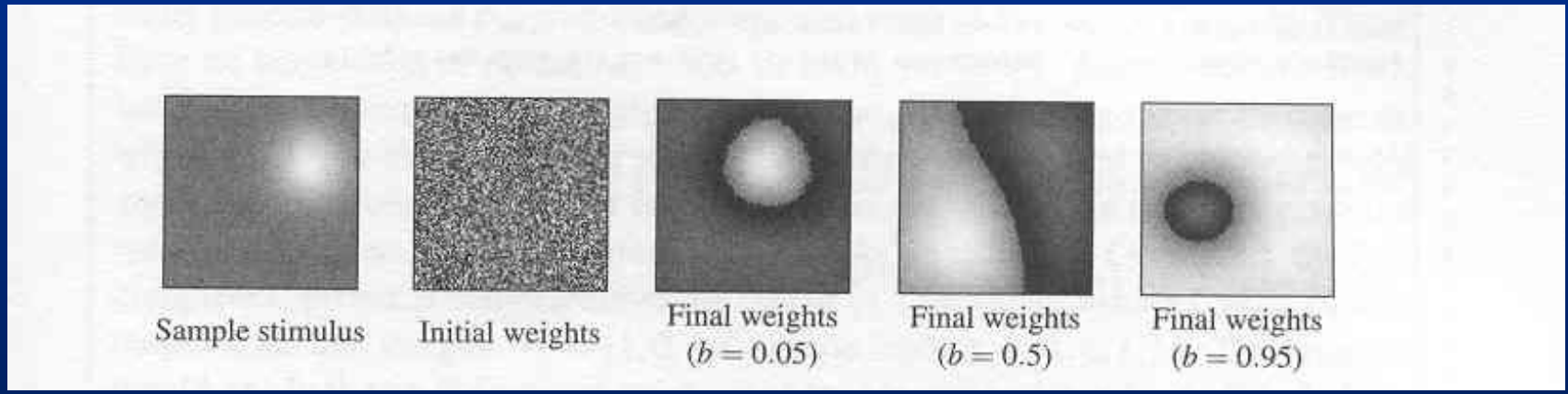
XOR:

x_1	x_2	out
0	0	0
1	0	1
1	0	1
1	1	0

Unsupervized training

Self organization:

Learning rule: $\Delta W_{ij} = \eta\ (z_i - a)(z_j - b)^*$



Enquist and Ghirlanda 2005

Training the perceptron on logical operators:

AND:

x_1	x_2	out
0	0	0
1	0	0
0	1	0
1	1	1

OR:

x_1	x_2	out
0	0	0
1	0	1
0	1	1
1	1	1

XOR:

x_1	x_2	out
0	0	0
1	0	1
1	0	1
1	1	0

Training the perceptron on AND:

AND:

x_1	x_2	out
0	0	0
1	0	0
0	1	0
1	1	1

OR:

x_1	x_2	out
0	0	0
1	0	1
0	1	1
1	1	1

XOR:

x_1	x_2	out
0	0	0
1	0	1
1	0	1
1	1	0

Training the perceptron on AND:

AND:

x_1	x_2	out	result	error
0	0	0	0.35	0.35
1	0	0		
0	1	0		
1	1	1		

Training the perceptron on AND:

AND:

x_1	x_2	out	result	error
0	0	0	0.35	0.35
1	0	0	0.55	0.55
0	1	0		
1	1	1		

Training the perceptron on AND:

AND:

x_1	x_2	out	result	error
0	0	0	0.35	0.35
1	0	0	0.55	0.55
0	1	0	0.05	0.05
1	1	1		

Training the perceptron on AND:

AND:

x_1	x_2	out	result	error
0	0	0	0.35	0.35
1	0	0	0.55	0.55
0	1	0	0.05	0.05
1	1	1	0.25	0.75

Training the perceptron on AND:

AND:

x_1	x_2	out	result	error	
0	0	0	0.35	0.35	new weights
1	0	0	0.55	0.55	new weights
0	1	0	0.05	0.05	new weights
1	1	1	0.25	0.75	new weights

Training the perceptron on AND:

AND:

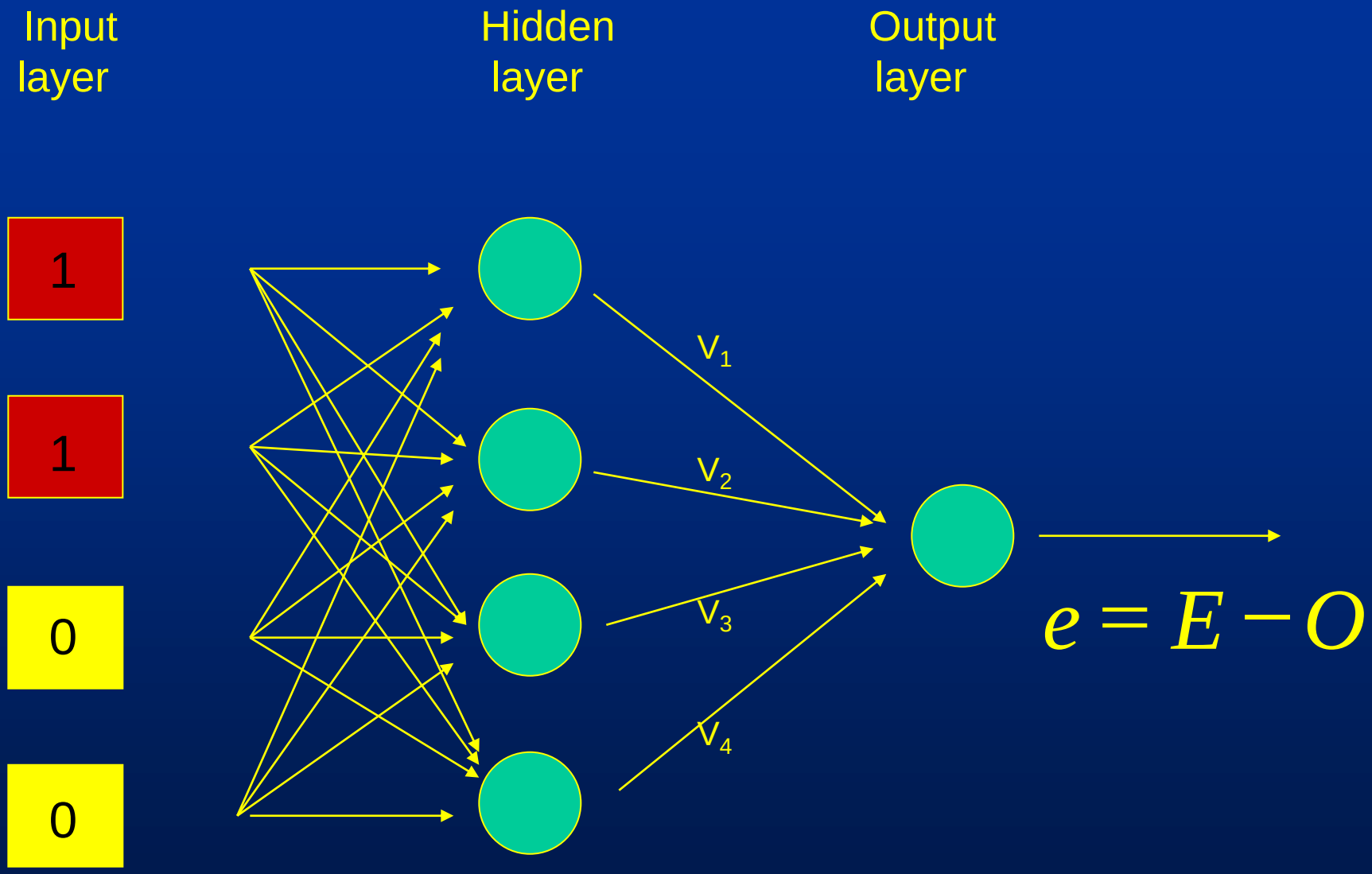
x_1	x_2	out	result	error
0	0	0	0.35	0.35
1	0	0	0.55	0.55
0	1	0	0.05	0.05
1	1	1	0.25	0.75
Average:				0.425

new weights

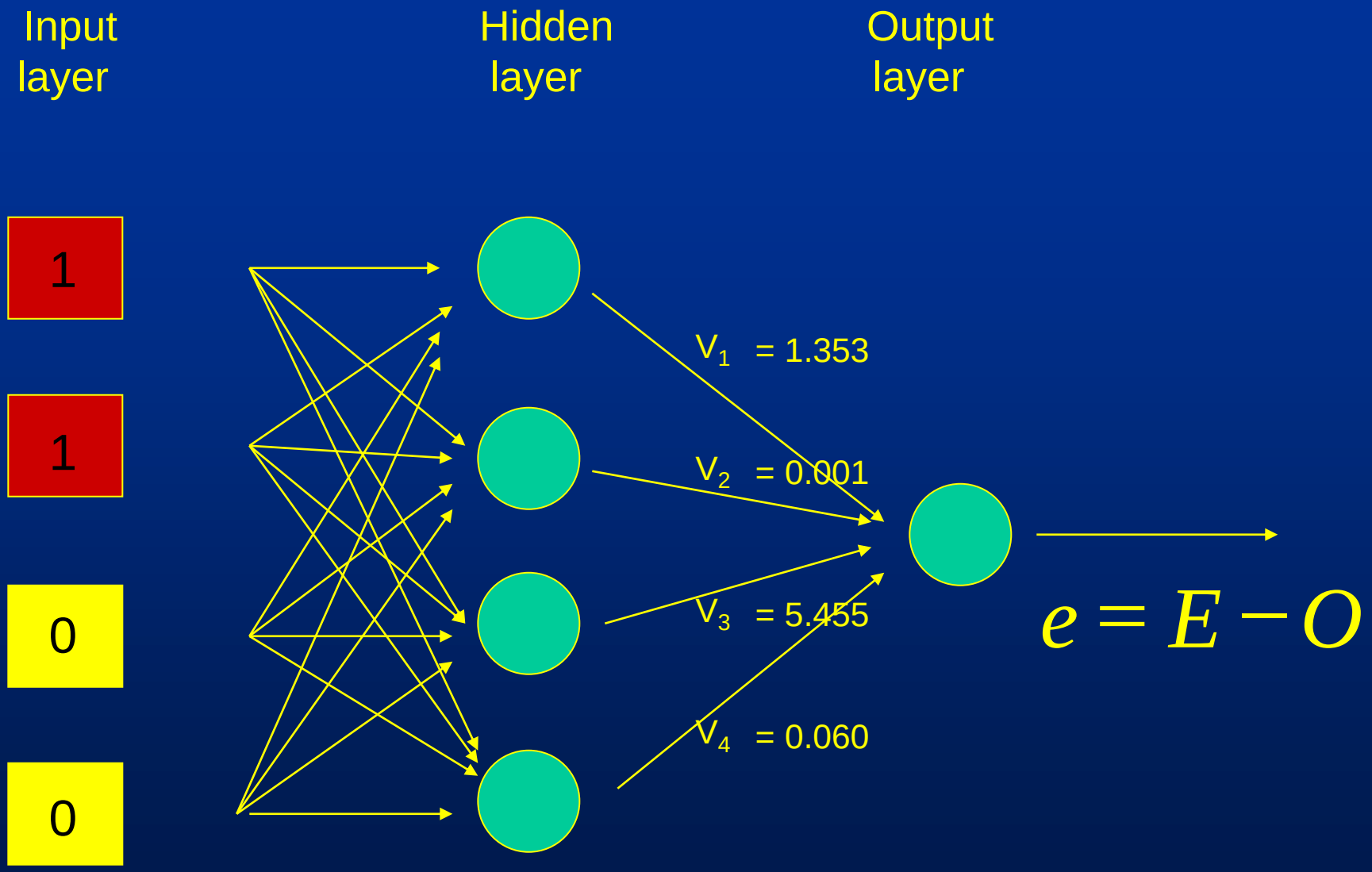
Learning rule:

1. Random choice of new weights
2. Supervised learning such as backpropagation of error

Backpropagation:



Backpropagation:



Learning rules:

Hebb-rule: $\Delta w_{ij} = k \cdot x_i \cdot x_j$

Backpropagation:

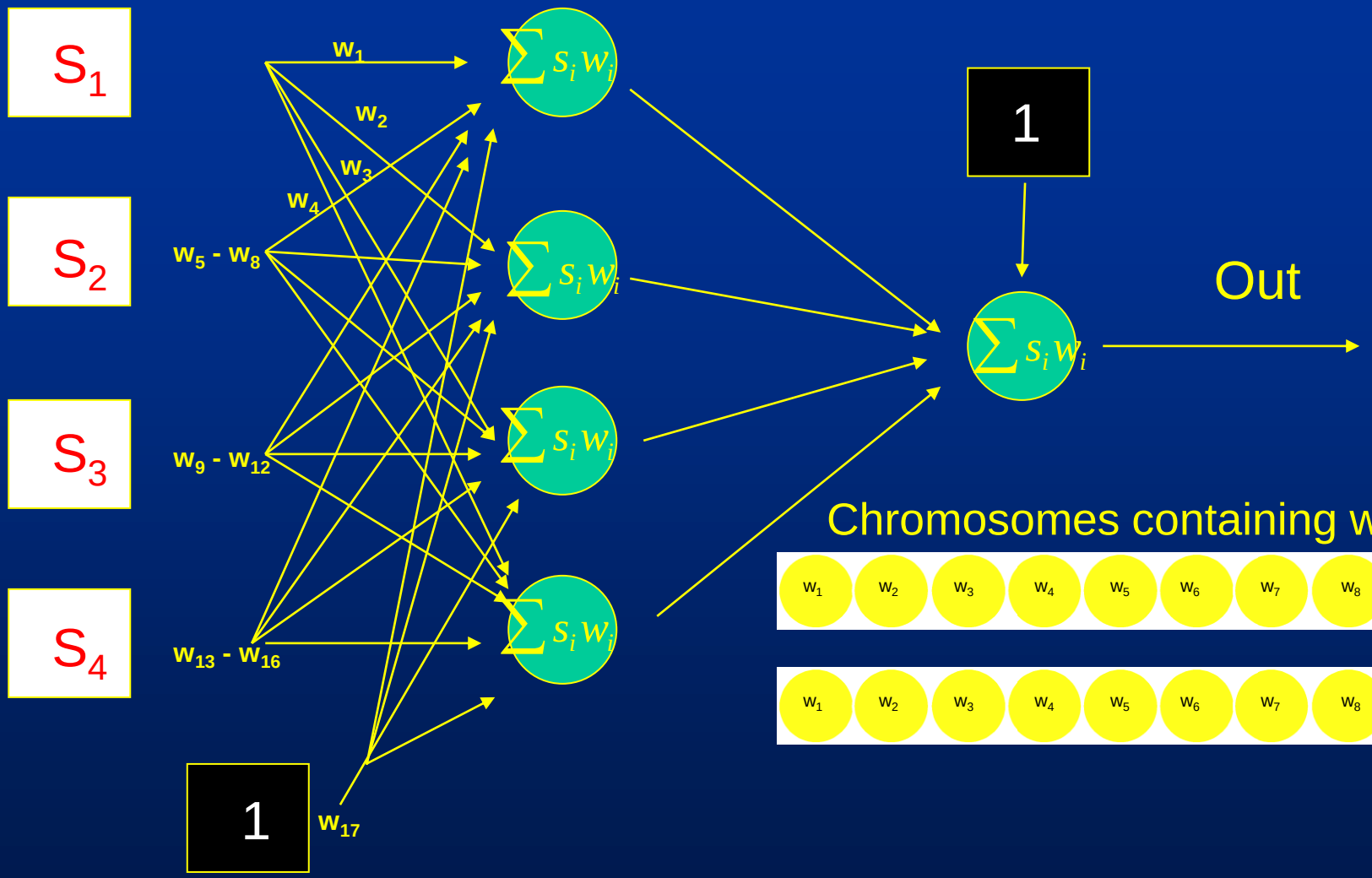
delta-rule: $\Delta w_{ij} = k \cdot x_i \cdot (d_j - x_j) \frac{\partial E}{\partial w_{ij}}$

perceptron learning rule $\Delta w_{ij} = 2k\, x_i$

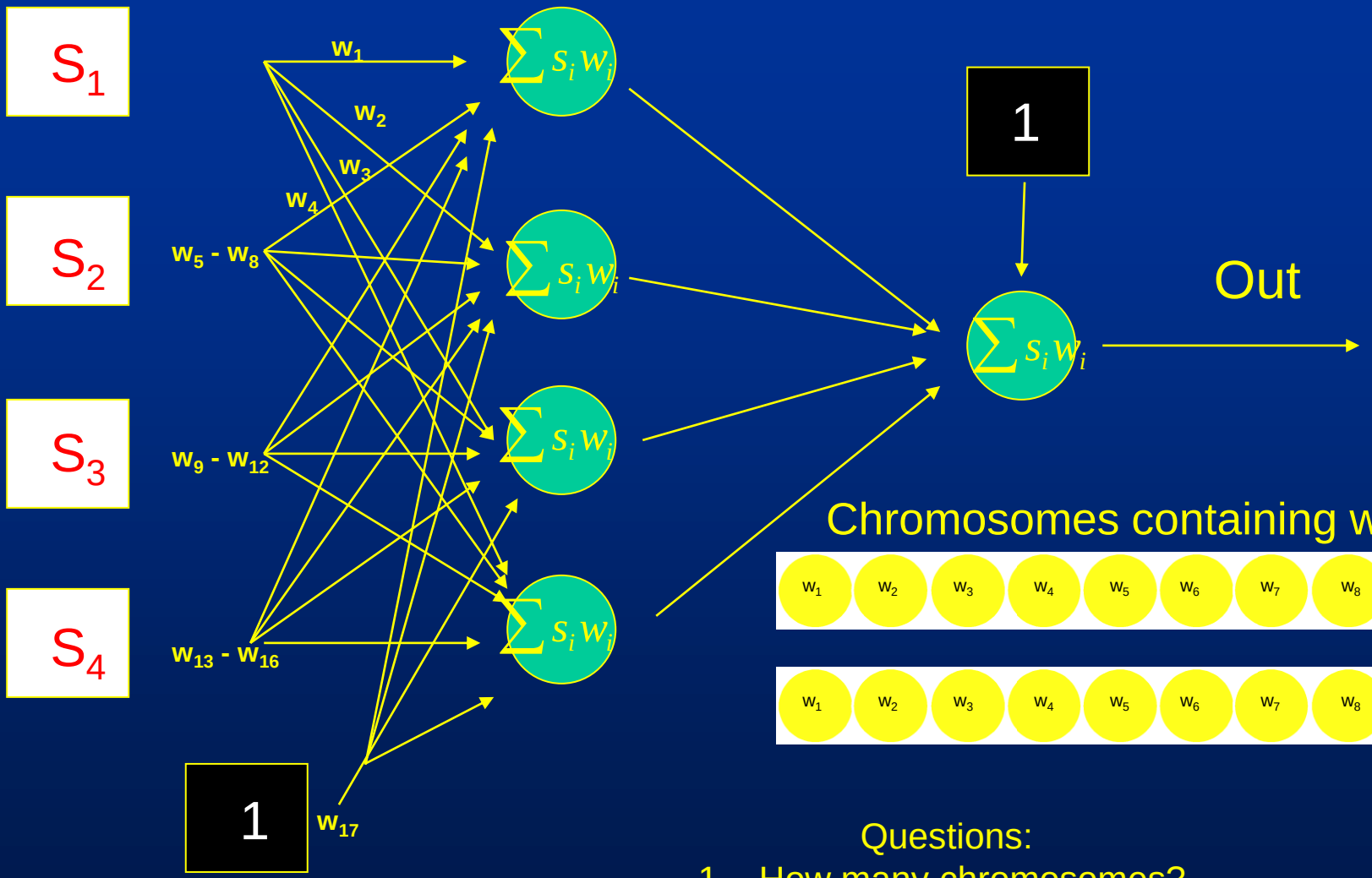
genetic algorithm!

Δwij = change in the weight between neuron *i* och *j*
x_i = signal from of neuron *i*
k = learning coefficient
d = desired output in node *j*
E = error

Neural network optimised by genetic algorithm:



Neural network optimised by genetic algorithm:



Chromosomes containing weights:

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	….
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	….

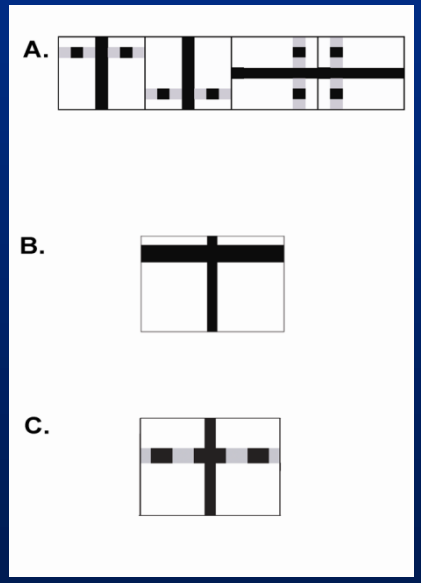
Questions:

1. How many chromosomes?
2. How many genes in each?
3. What type of reproduction?

Neural network optimised by genetic algorithm

= self learning in machines!

Examples of the combined approach:



Hybridization in the crow

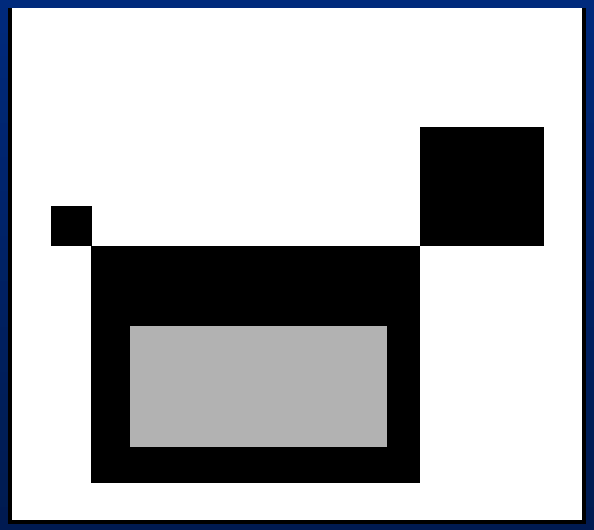
Input signals from ”pixel” images

$S_1 = -1 =$ $S_2 = 0 =$ $S_3 = 1 =$

14 x 14

45 x 45

200 x 200

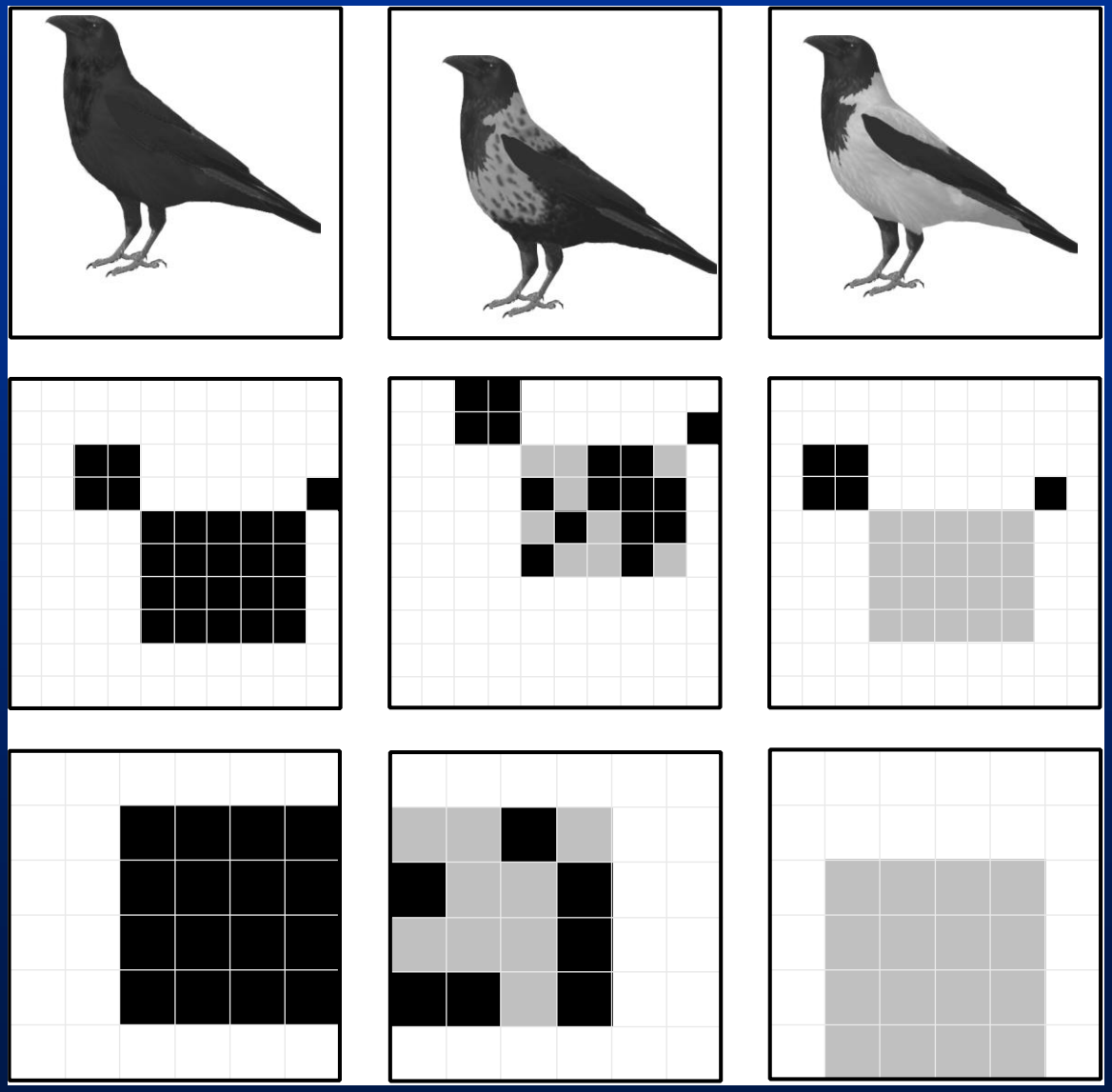


196

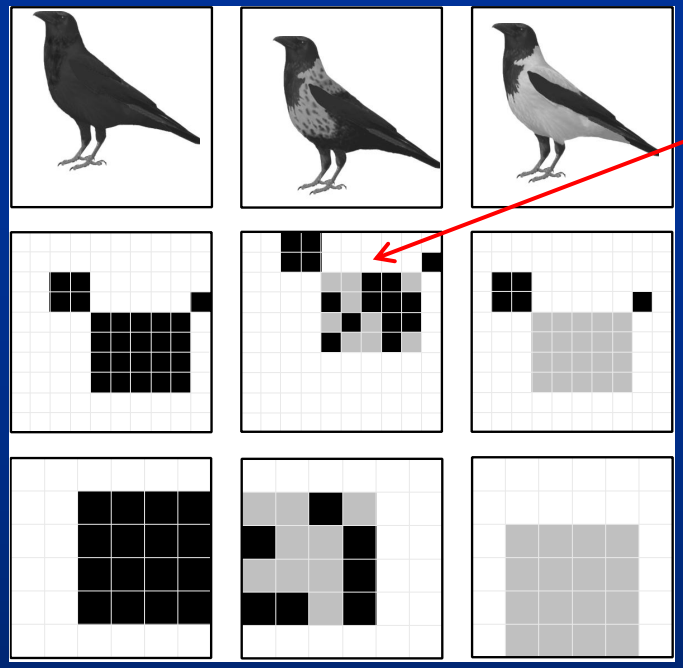
2025

40000

Generalisation:



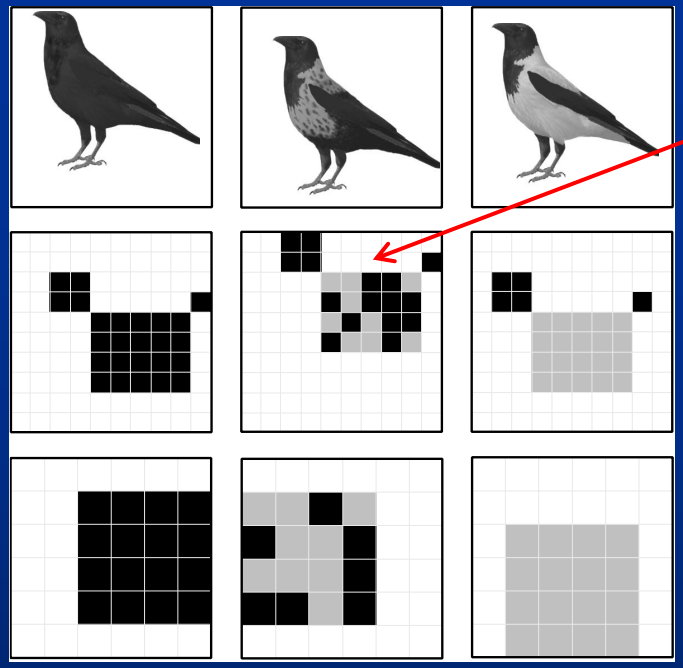
What information is the network learning?



0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 -1 -1 …

0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 …

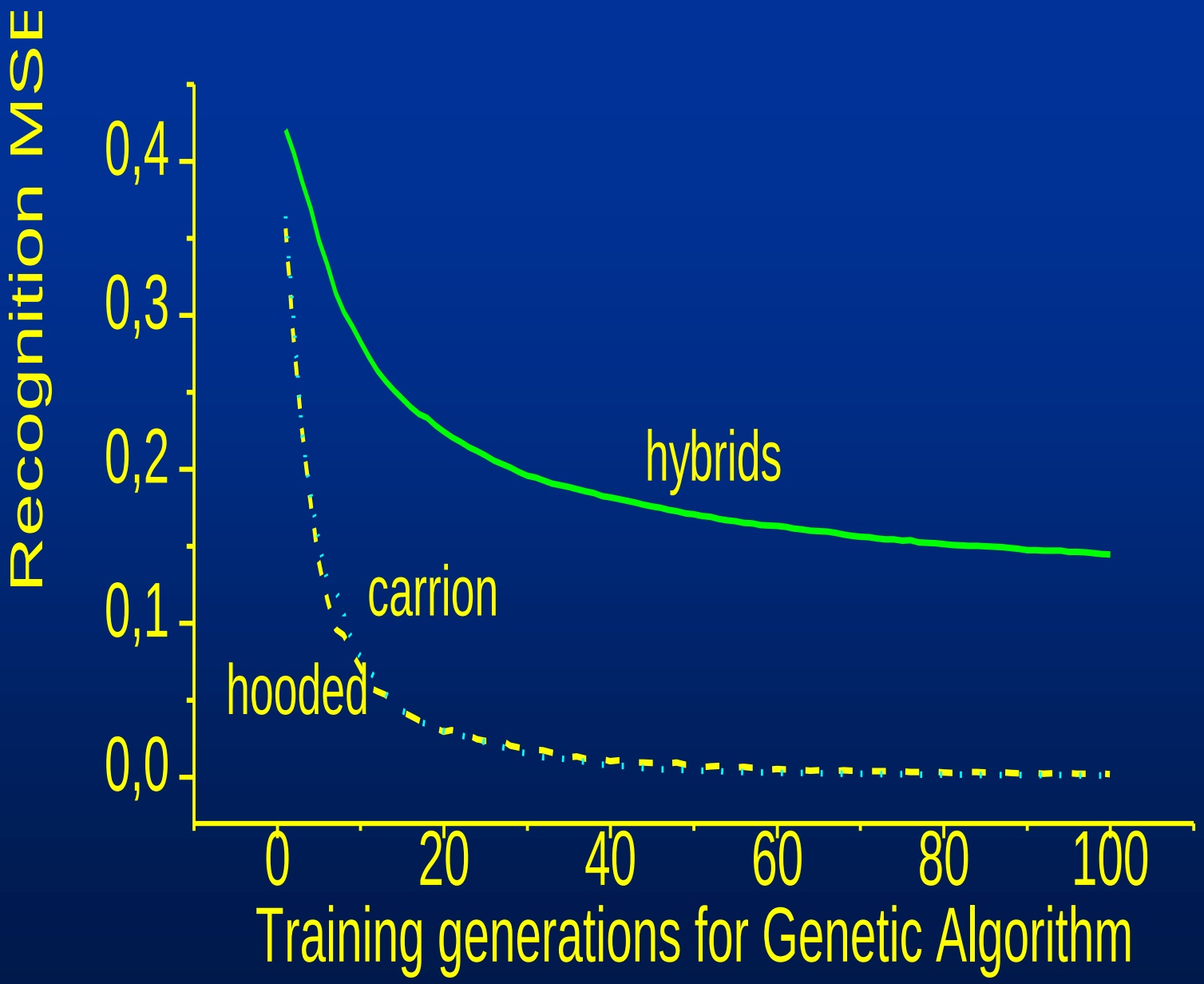
What information is the network learning?

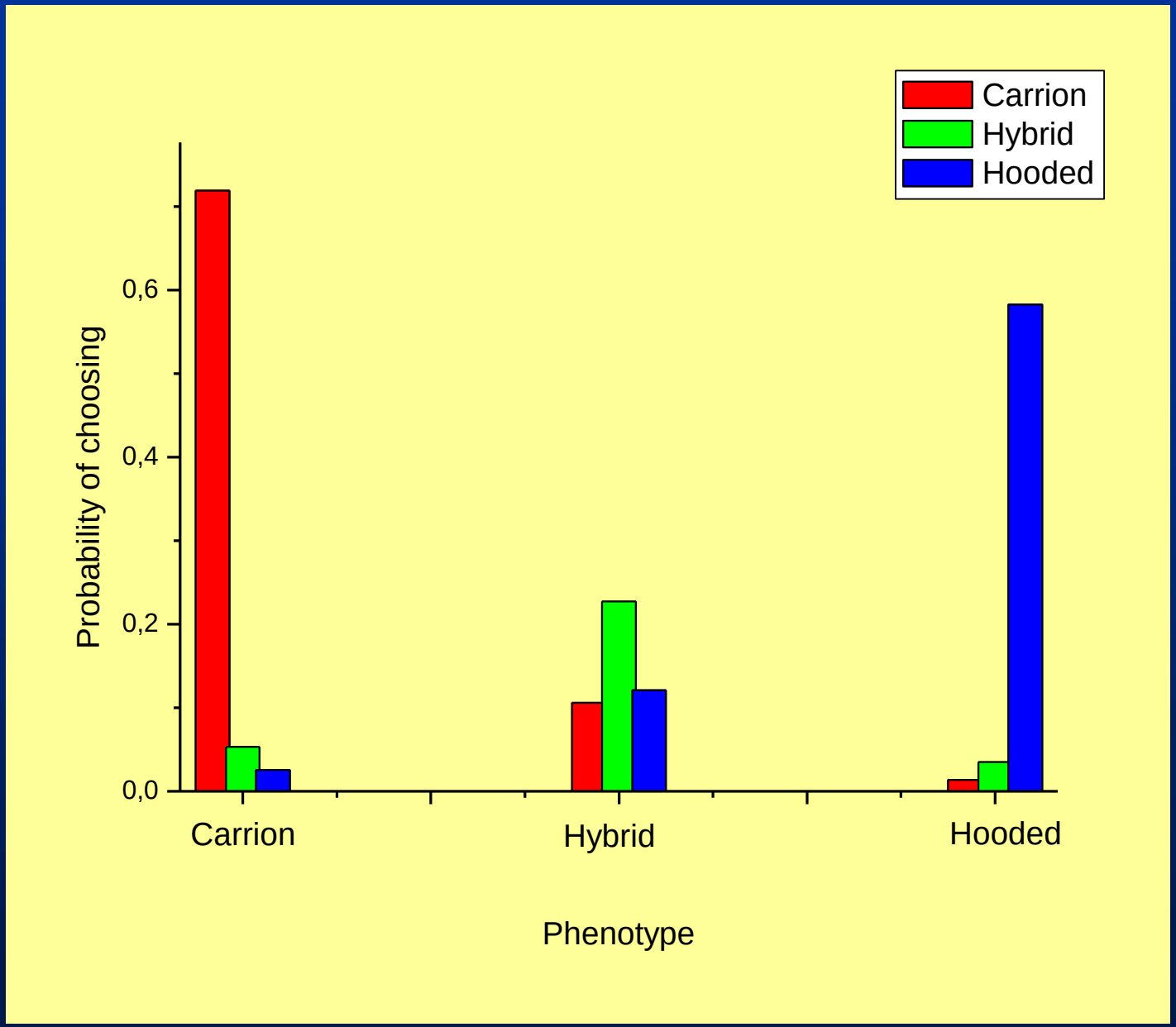


0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 -1 -1 …

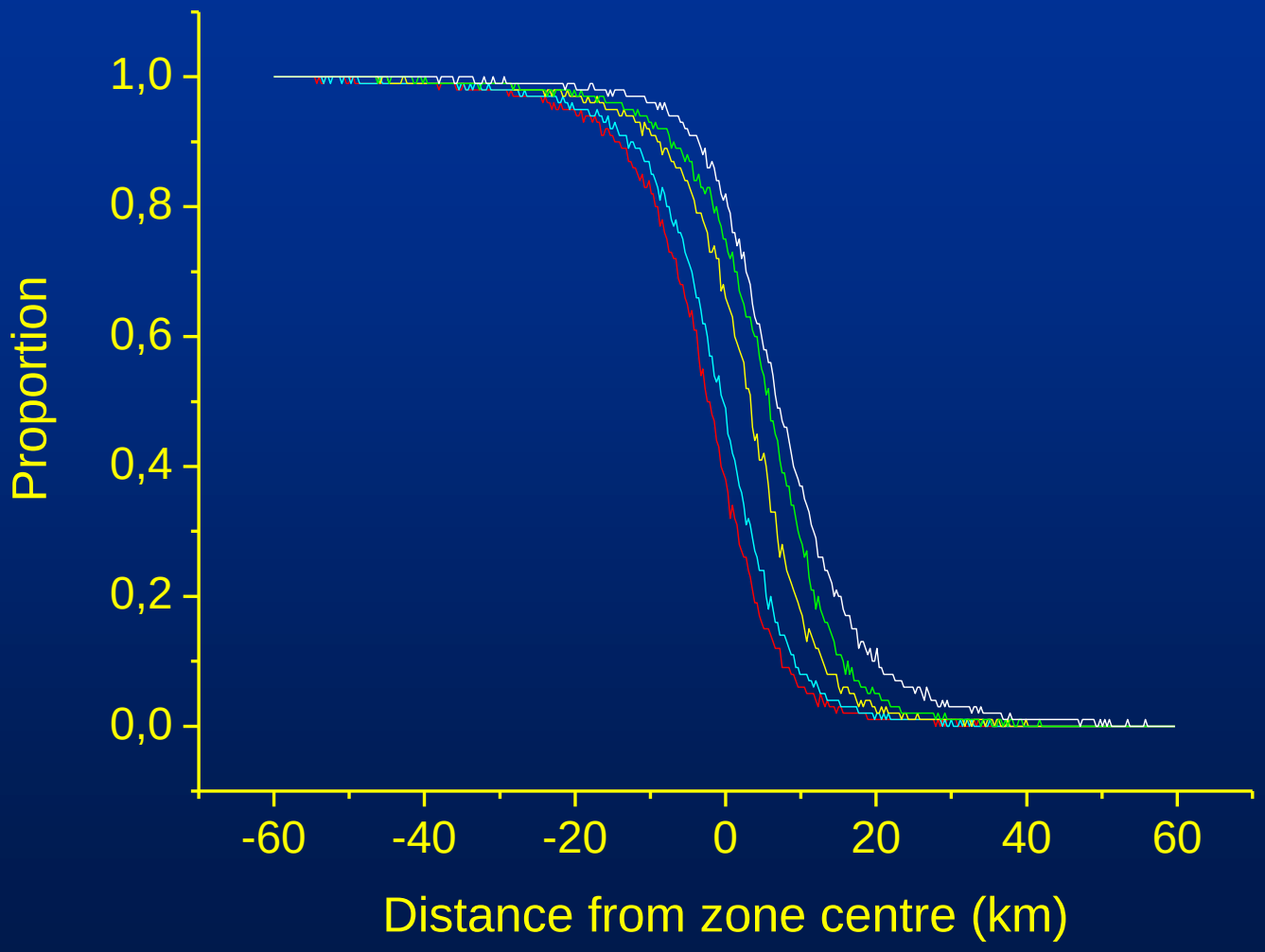
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 …

What would the picture 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 …
look like?

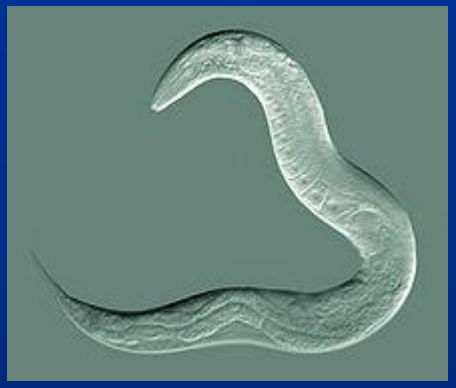

Recognition MSE
0,4
0,3
0,2
0,1
0,0
hybrids
carrion
hooded
0
20
40
60
80
100
Training generations for Genetic Algorithm


Carrion
Hybrid
Hooded
Probability of choosing
0,0
0,2
0,4
0,6
Carrion
Hybrid
Hooded
Phenotype

Zone movement during introgression



Caenorhabditis elegans:



302 neurons!