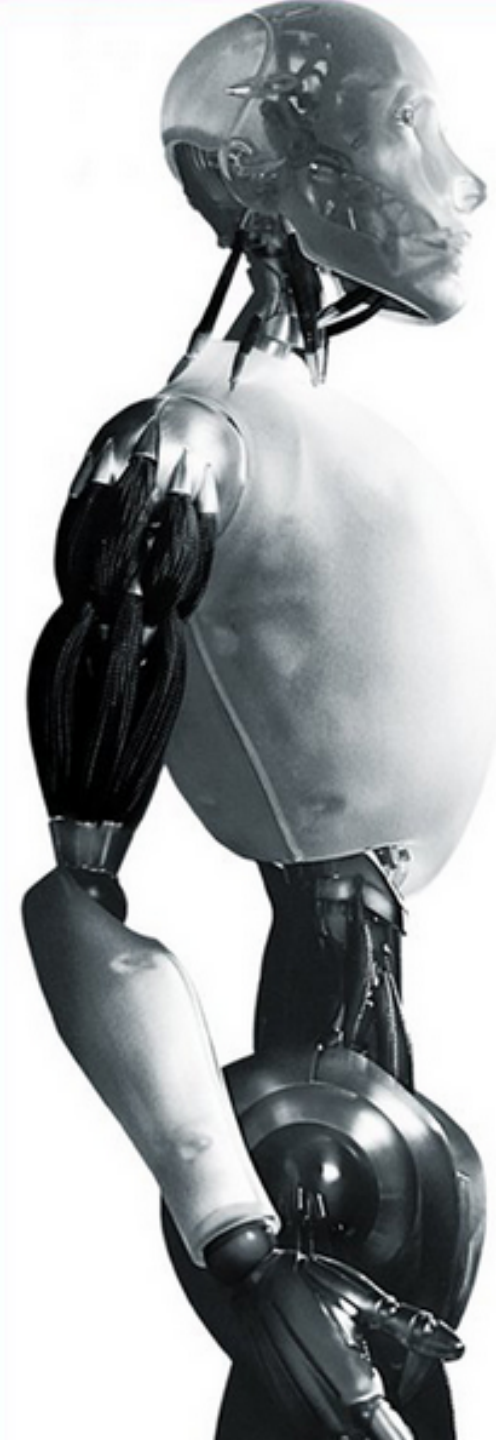
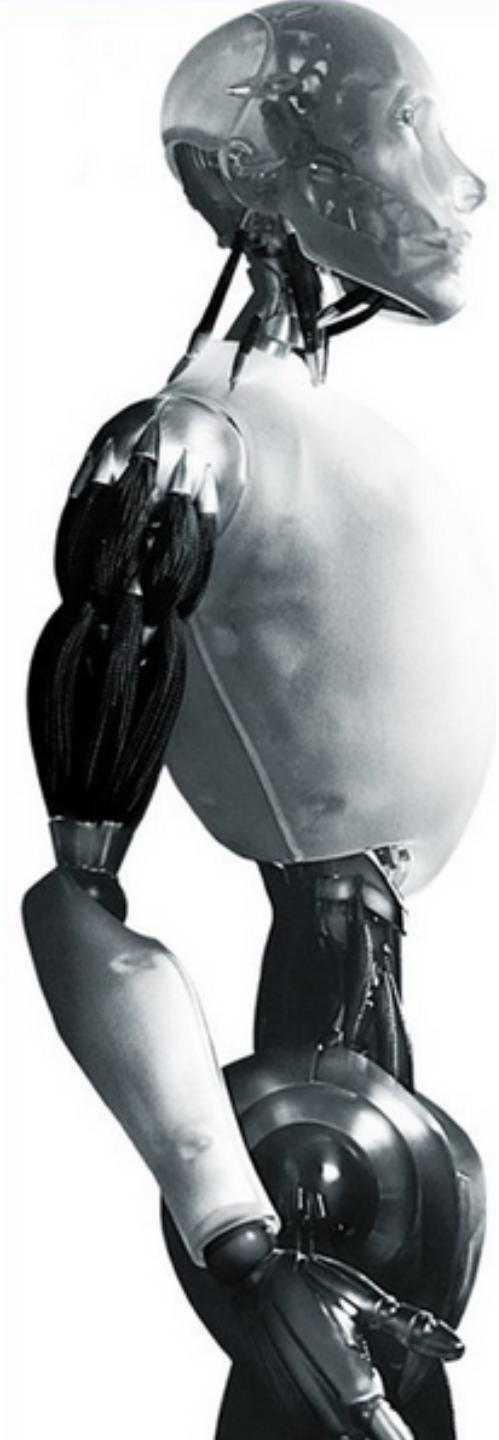
Artificial Neural Networks Backpropagation

Prof Alexiei Dingli



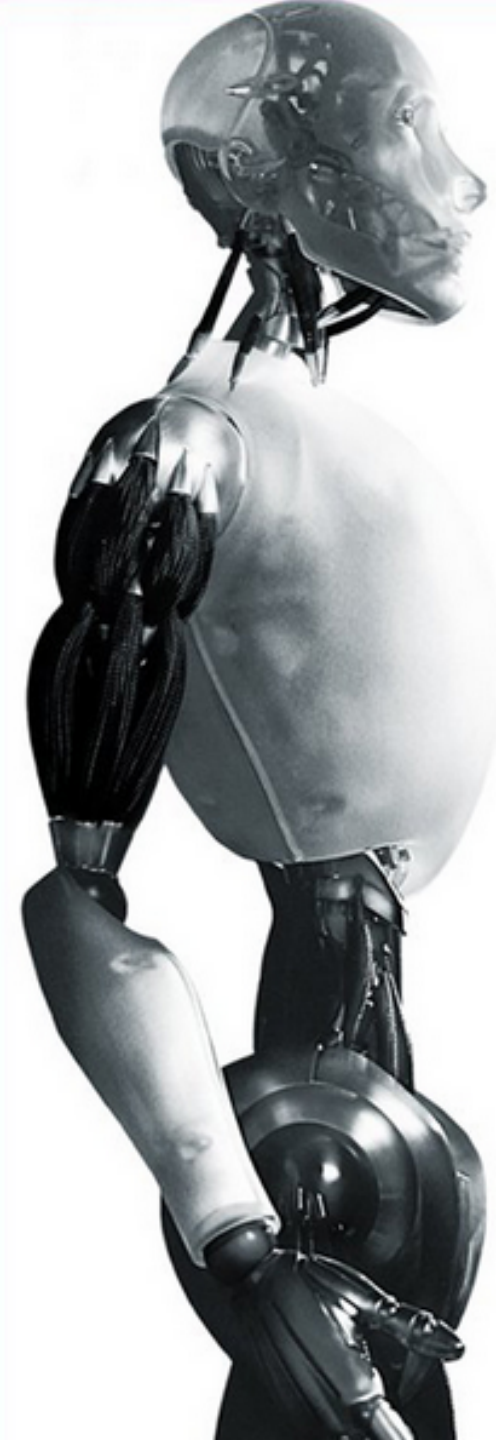
What is Backpropagation?

- Not a network
- A training algorithm
- Uses a simple network such as
  - Feed Forward Network

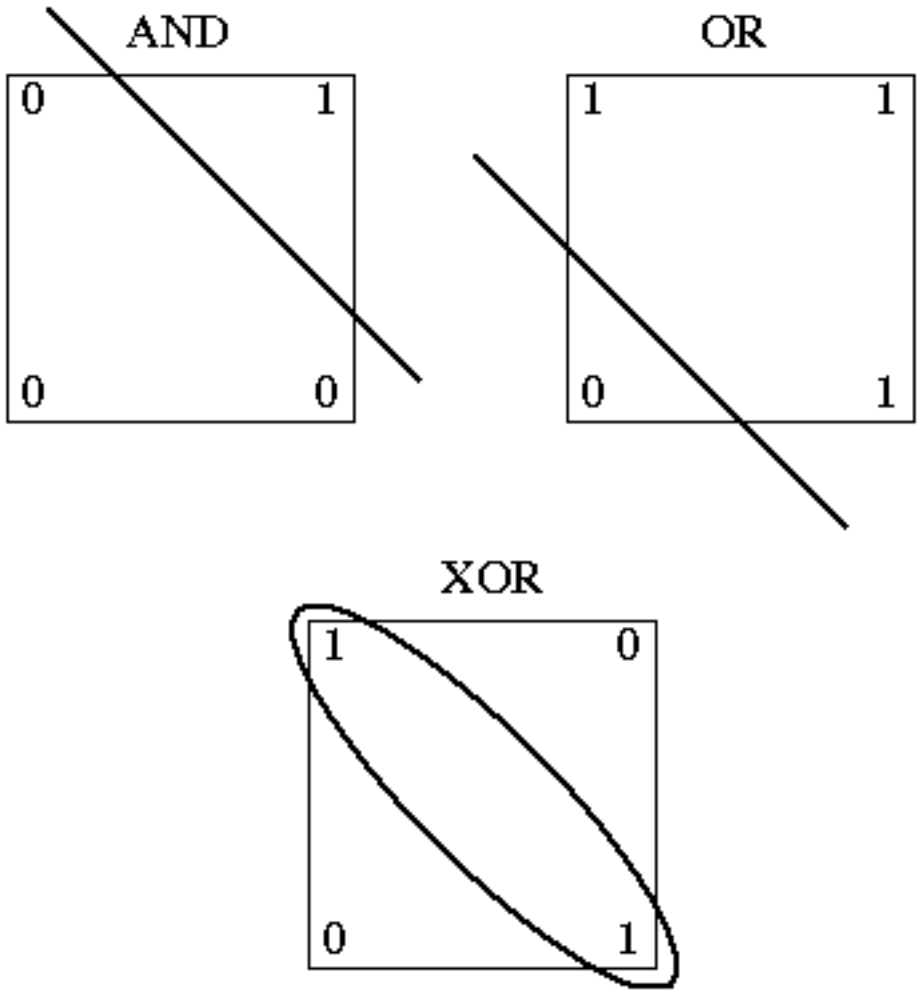


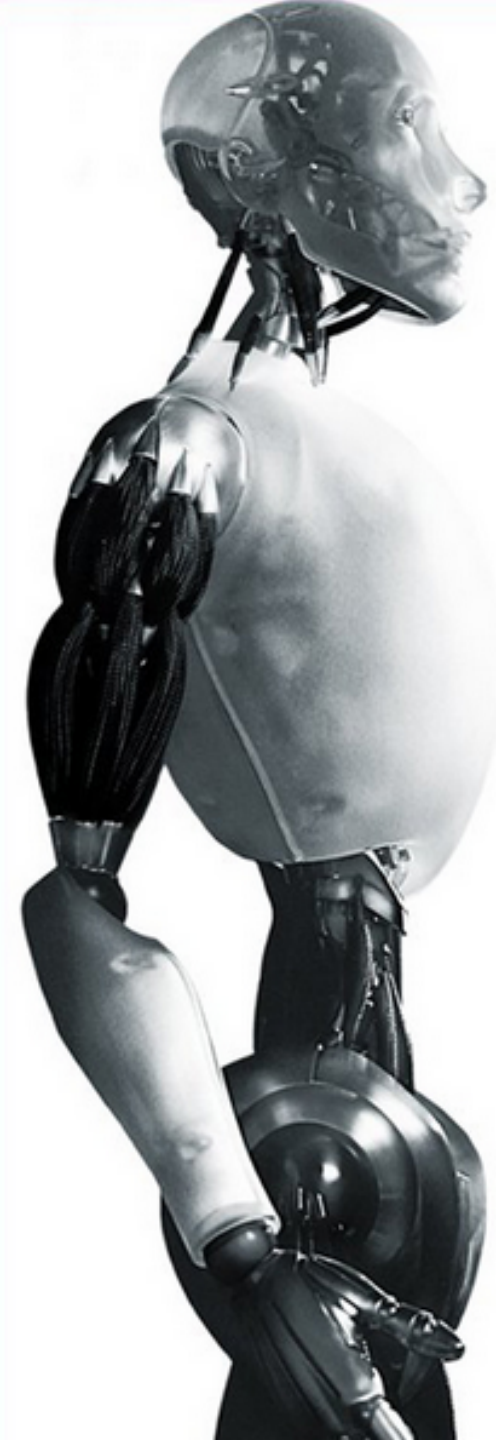
What is Backpropagation?

- Supervised Learning Algorithm
- Used by multi-layer networks
- Uses an output error to adjust the weights
- Neurons are activated using the sigmoid function



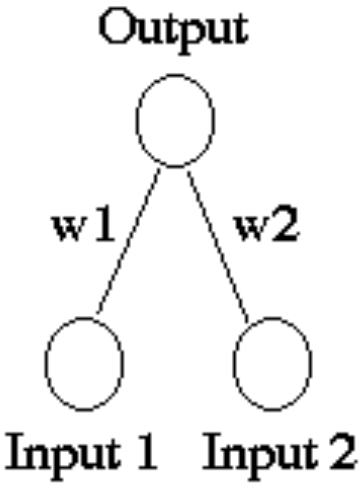
Why hidden layers?

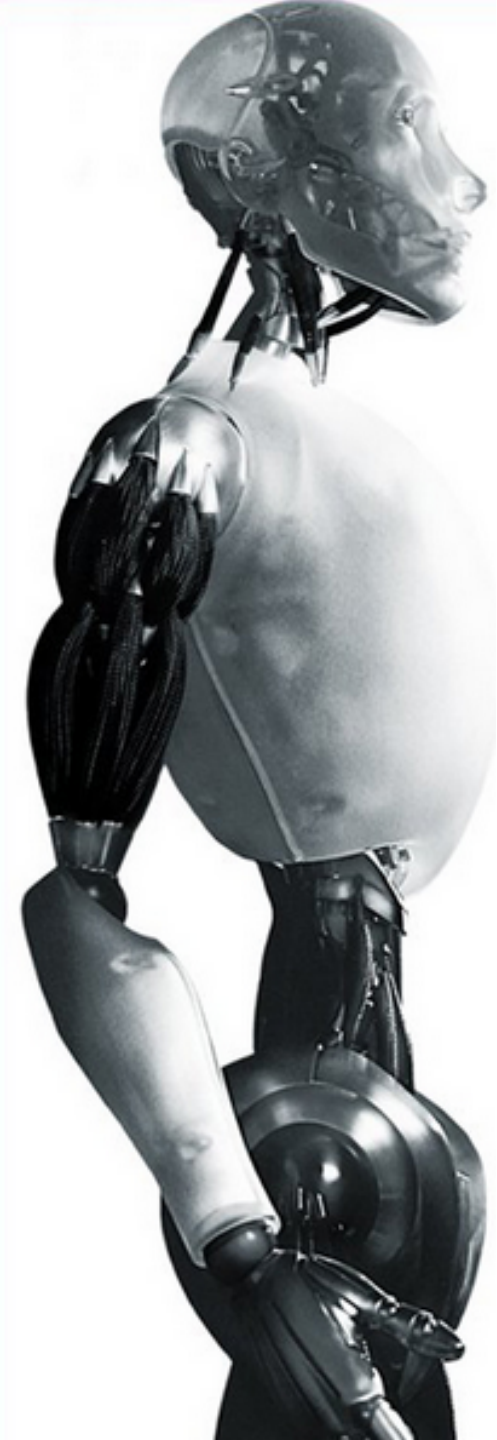




Why hidden layers?

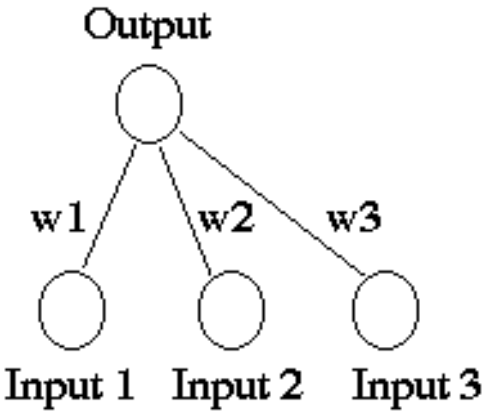
Input 1	Input 2	Output
1	1	0
1	0	1
0	1	1
0	0	0

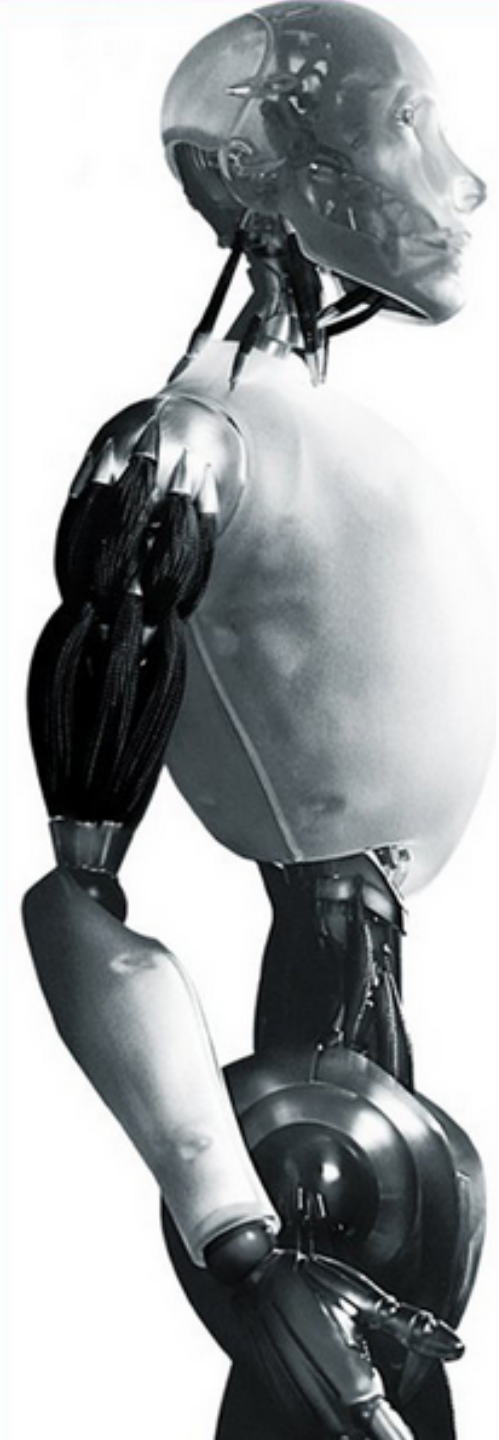




Why hidden layers?

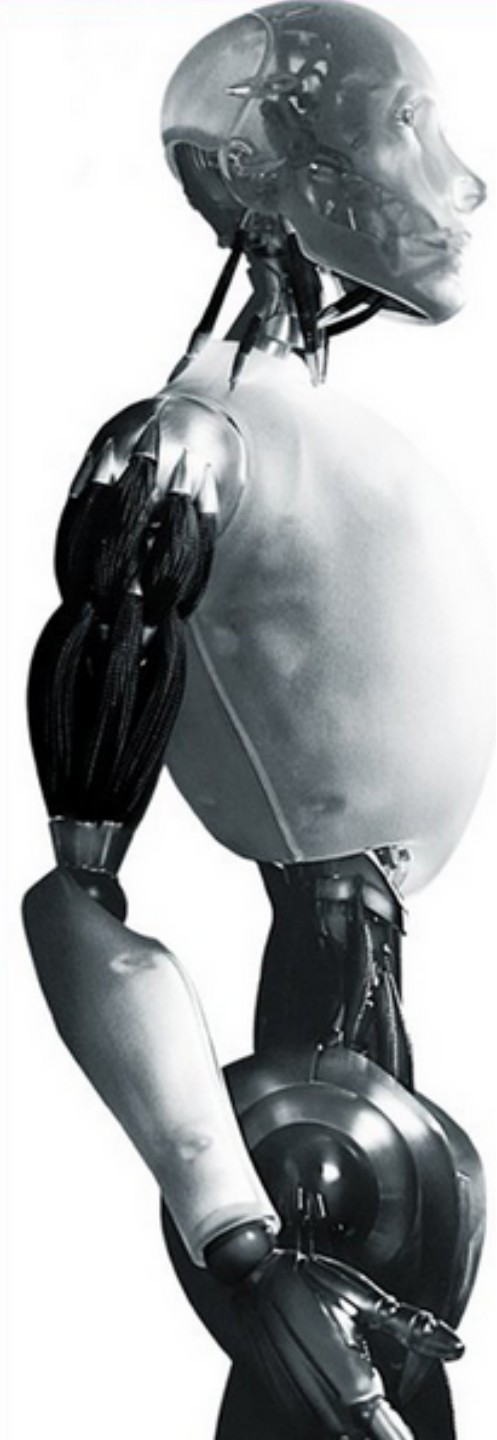
Input 1	Input 2	Input 3	Output
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0



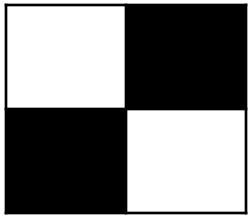
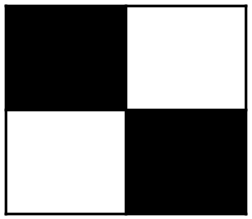
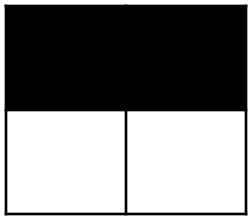


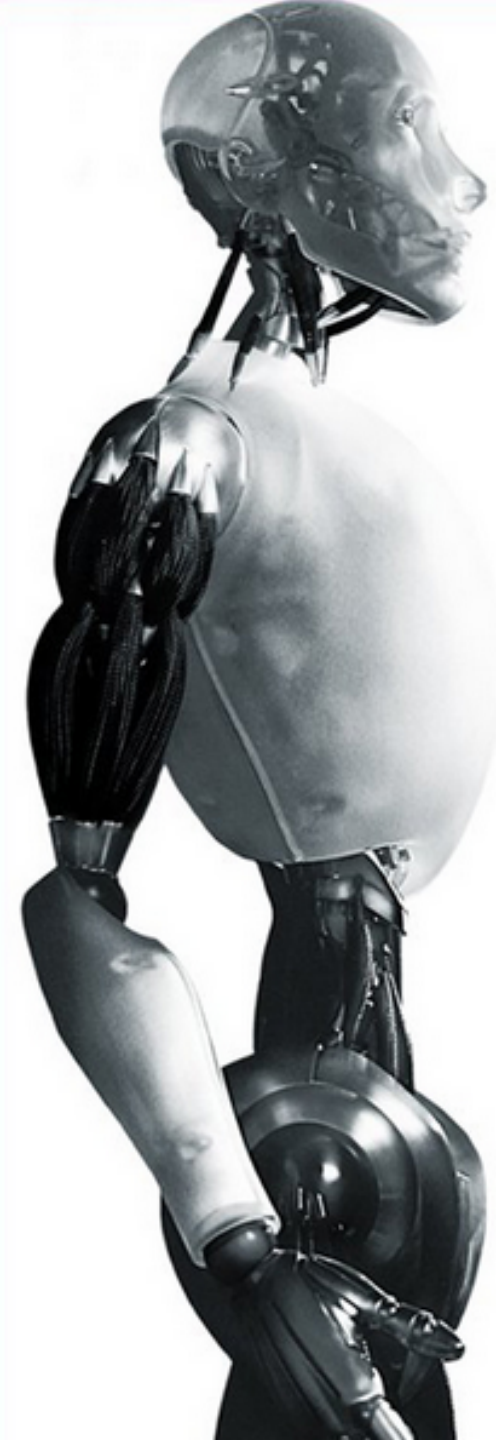
How does it work?

- Learns by example
  - Give examples
  - Adjust weights when training is finished
- Ideal for small Pattern Recognition tasks

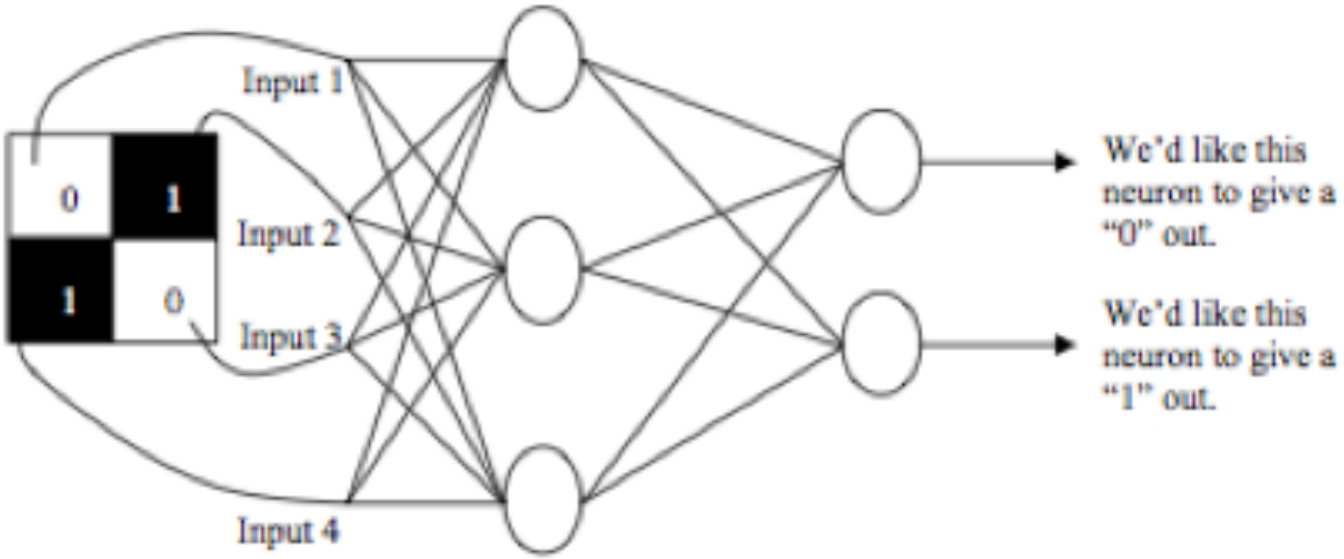


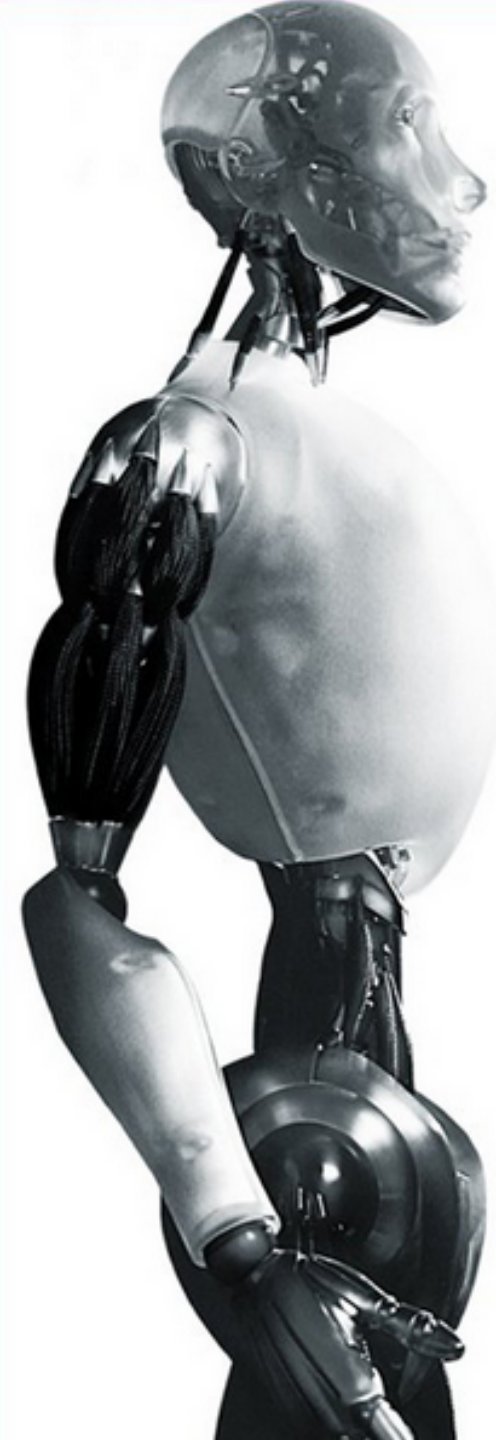
Training Set

Input	Target
	**0 1**
	**1 0**
	**1 1**



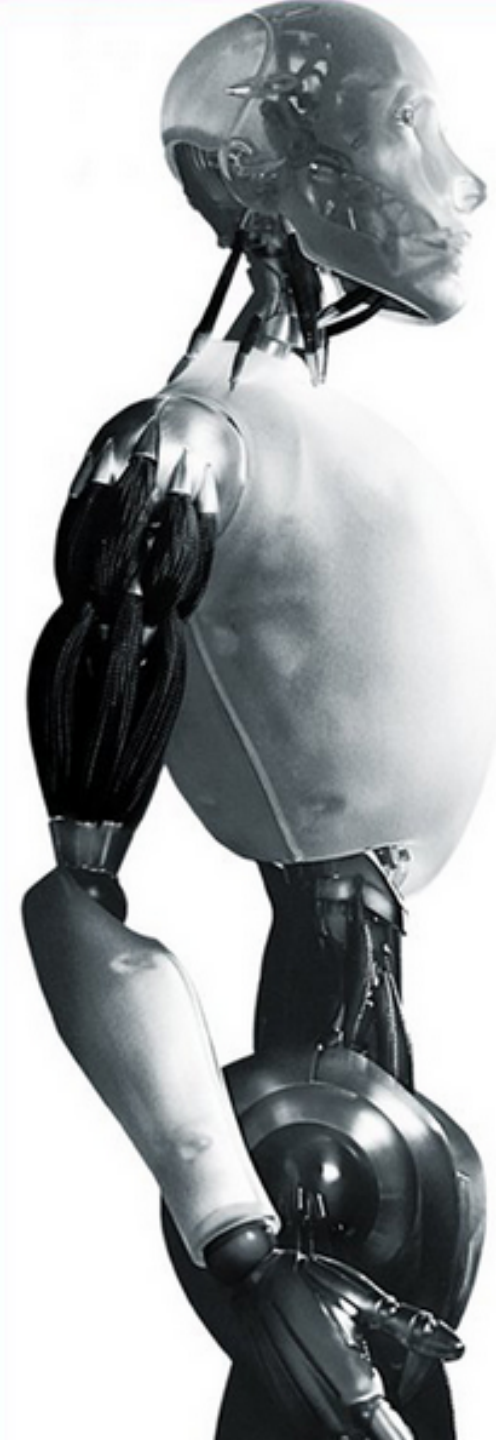
Applying a Training set to a network



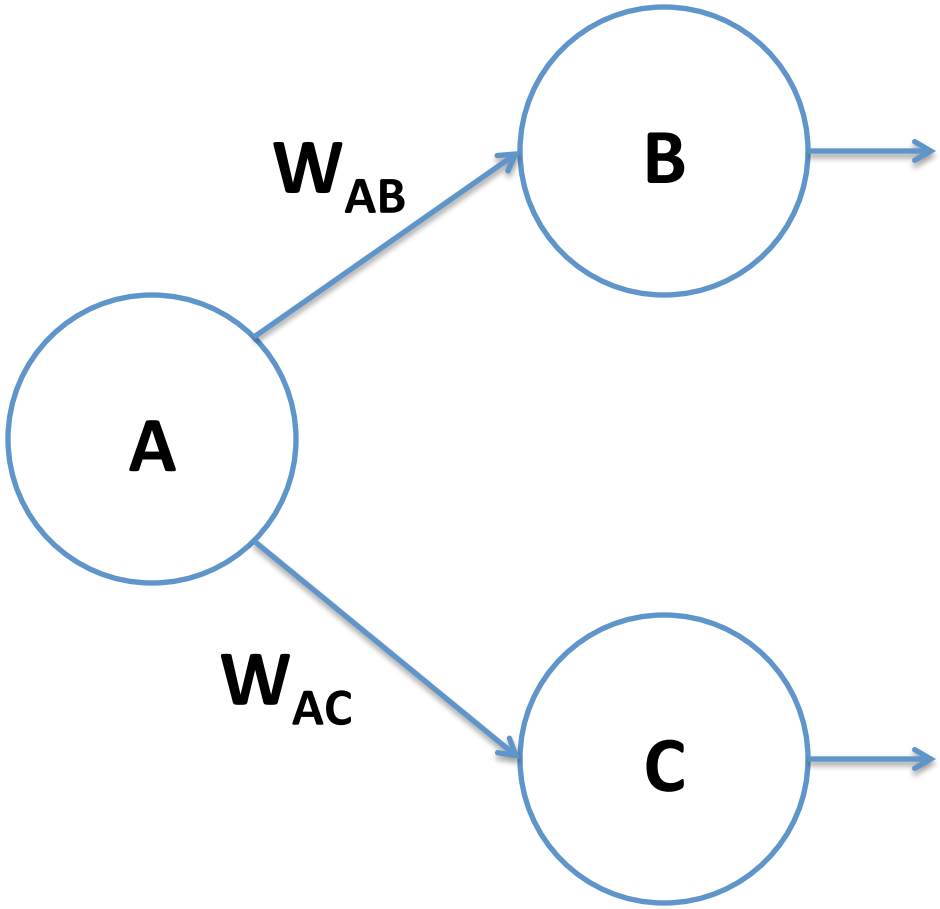


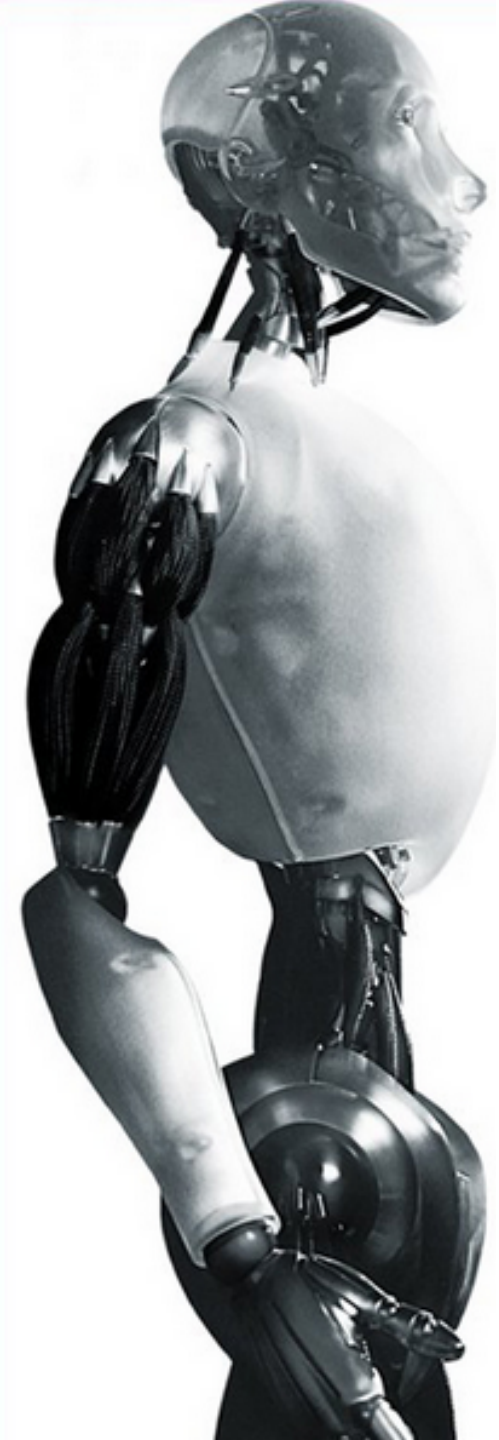
The process

- Network is set using random values
- Between +1 and -1
- A forward pass is applied
- An output is obtained
- Output is compared with the desired Target
- Error is calculated and propagated backwards
- Repeat



Example 1

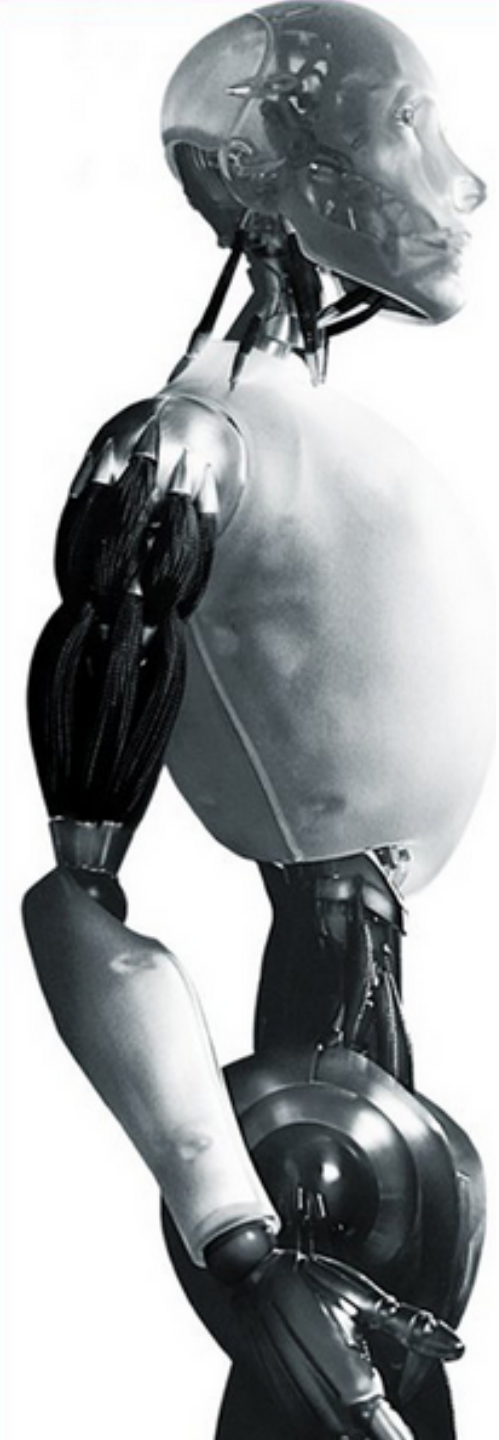




Step 1

- Apply an input to the network
- Get an output

- Since the weights are random, the output can be practically anything

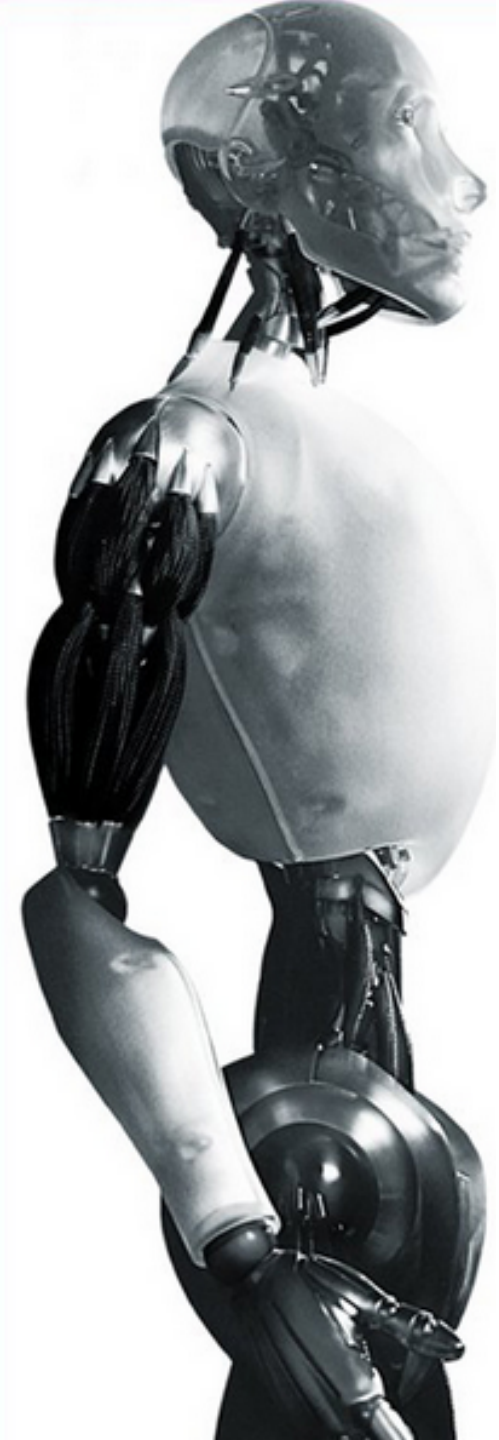


Step 2

- Calculate the error

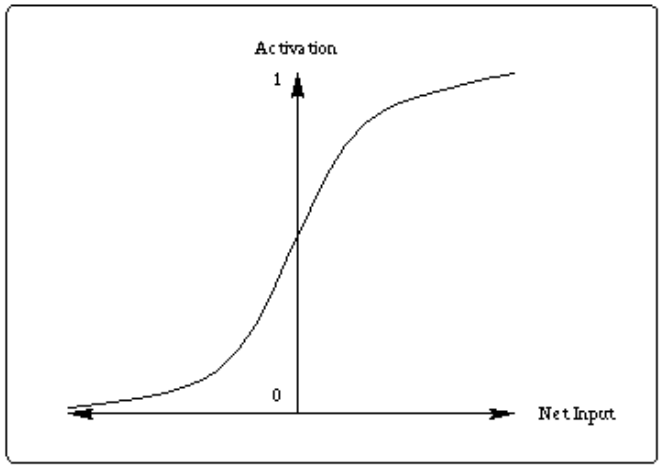
ErrorB = OutputB (1-OutputB)(TargetB – OutputB)

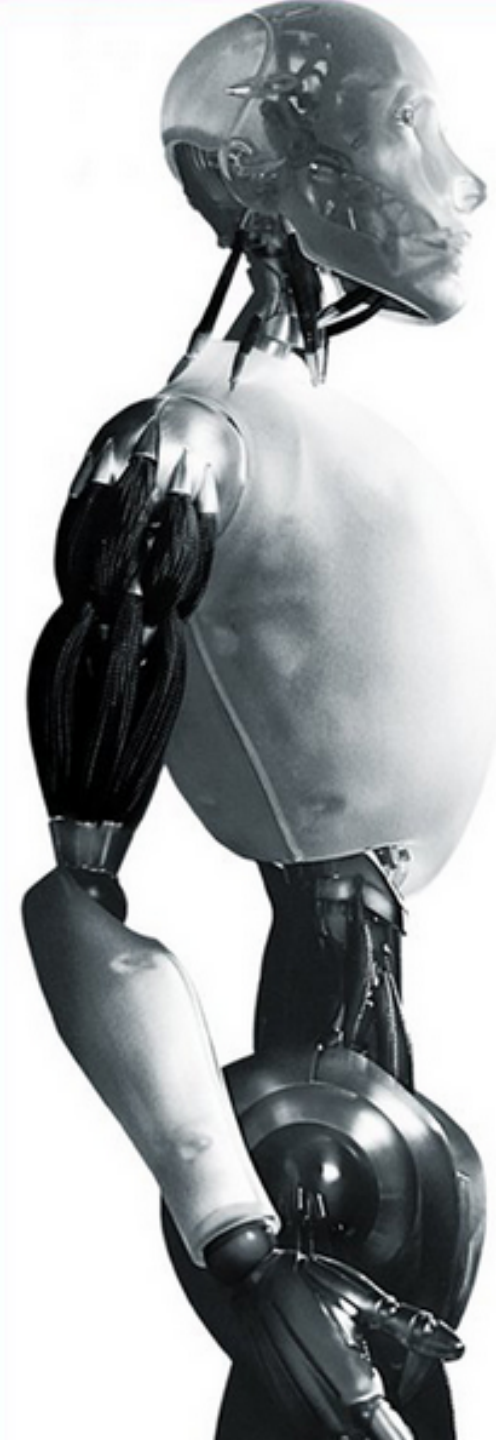
- **Sigmoid function = OutputB (1-OutputB)**
- **Delta = (TargetB – OutputB)**



Sigmoid function

- Normally referred to as a squashing function
- Squashes values between a fixed range between 0 and 1
- Activation occurs if its greater than 0



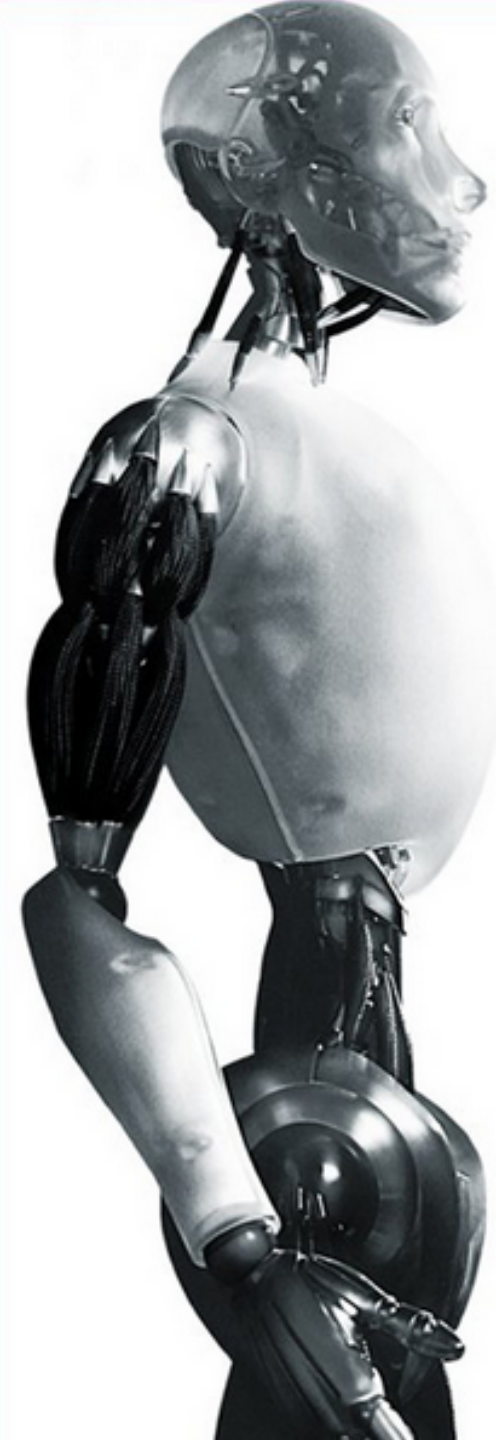


Step 3

- Change the weight

$$W_{AB} = W_{AB} + (\text{ErrorB} \times \text{OutputA})$$

$$W_{AC} = W_{AC} + (\text{ErrorC} \times \text{OutputA})$$

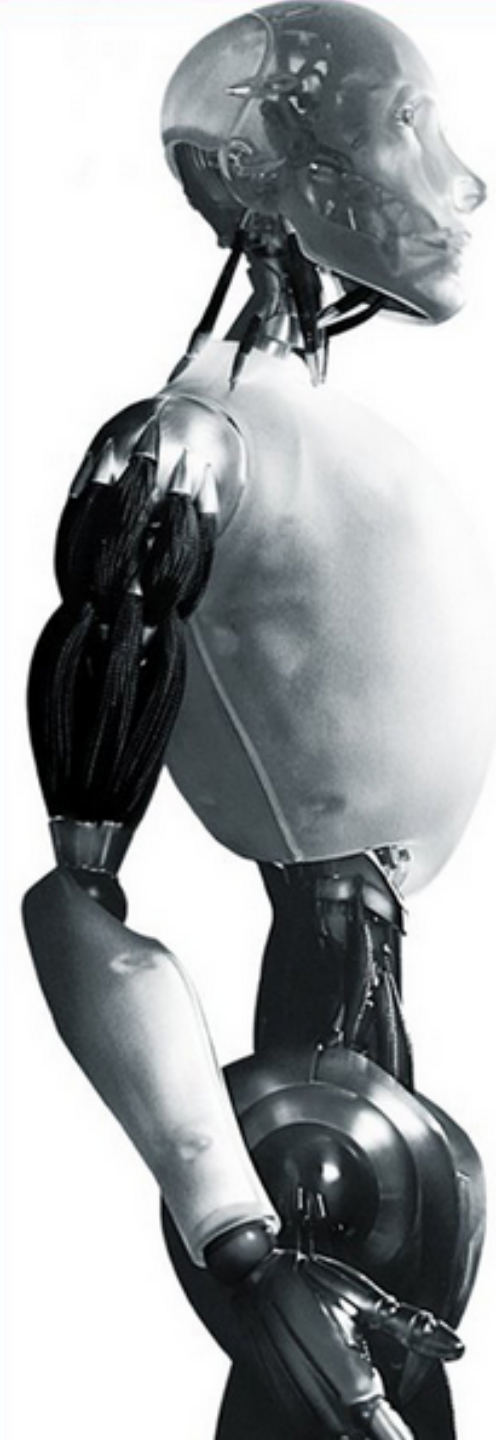


Step 4

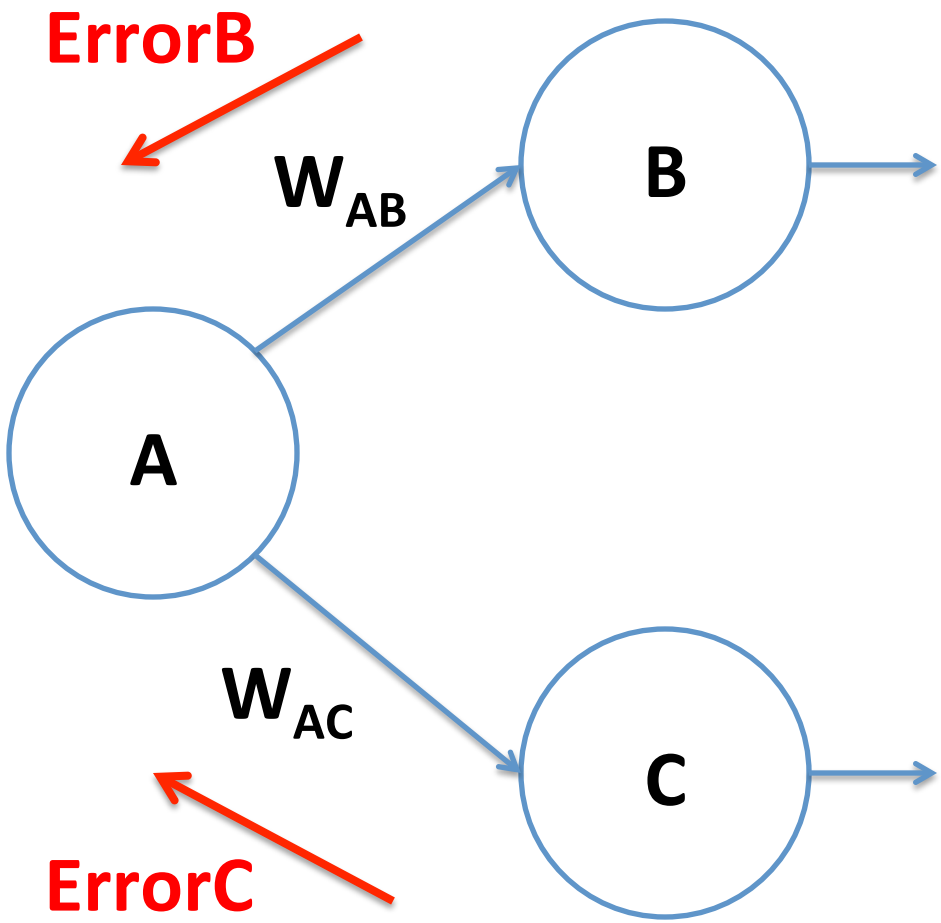
- Propagate backwards
- ErrorA =

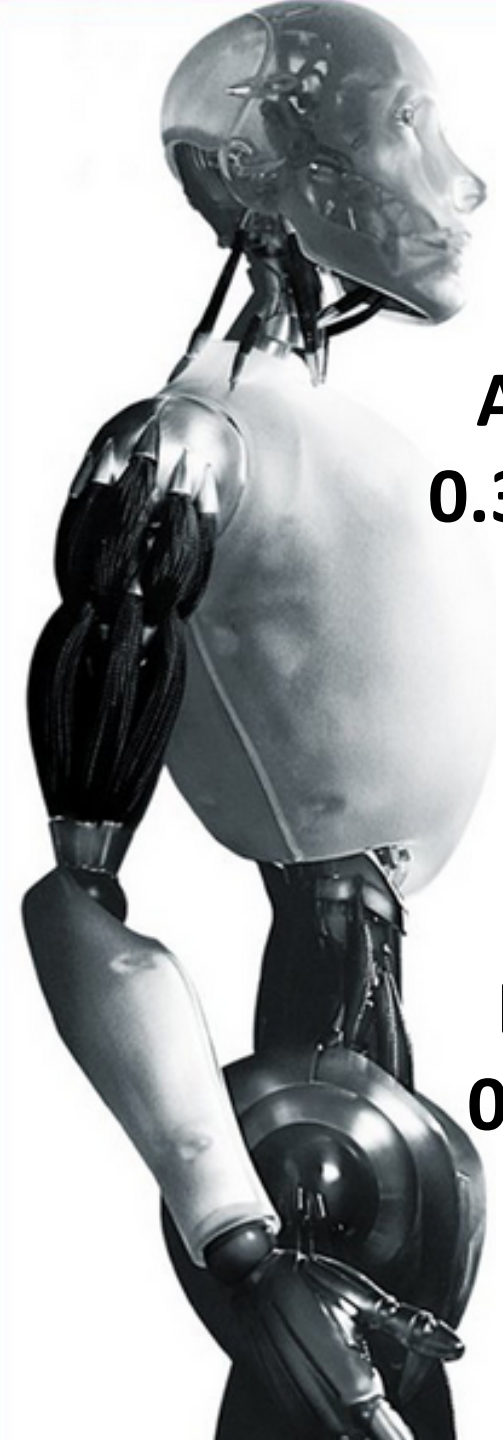
  Output A (1 - Output A)(ErrorB W_{AB} + ErrorC W_{AC})

- Goto step 3

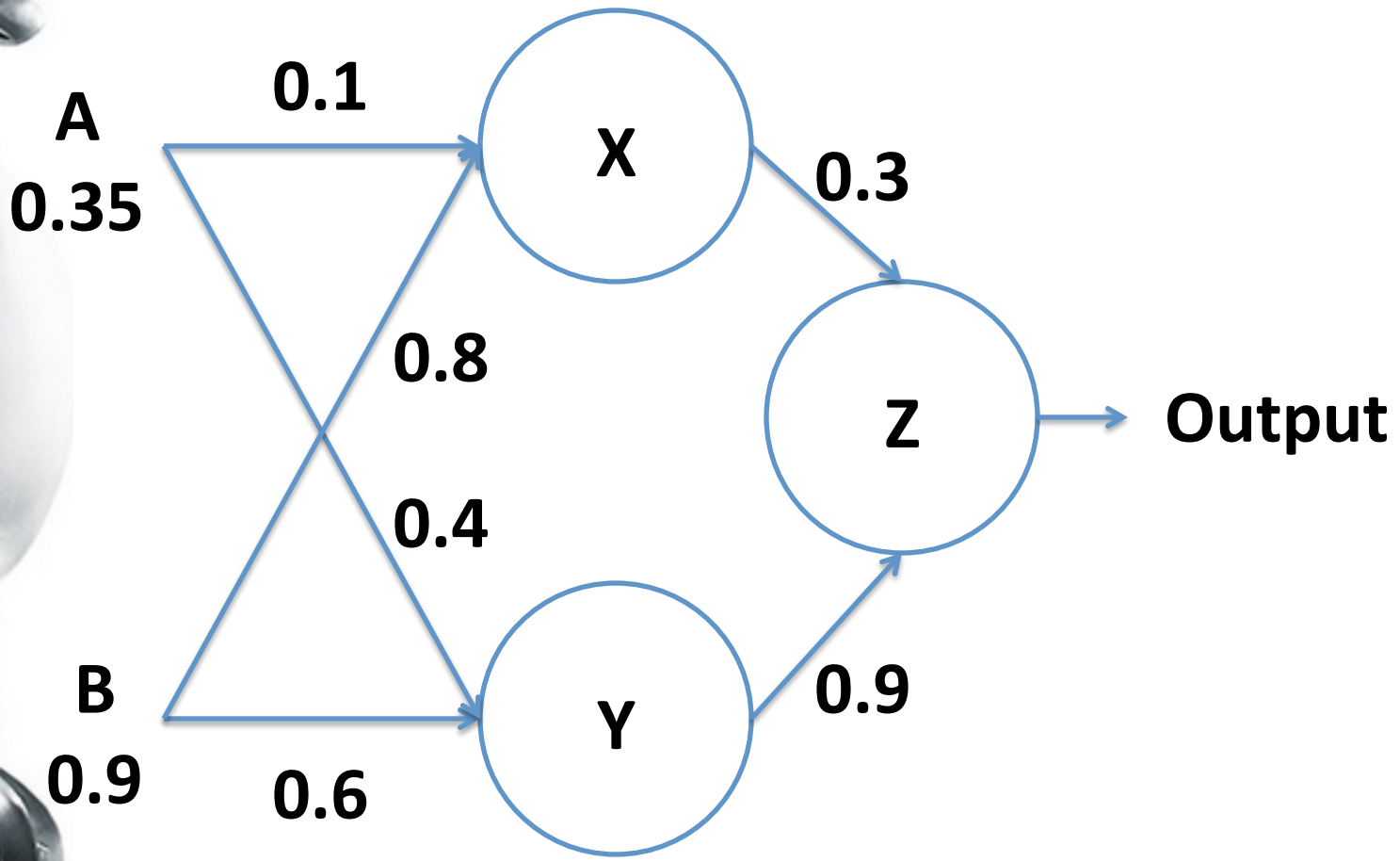


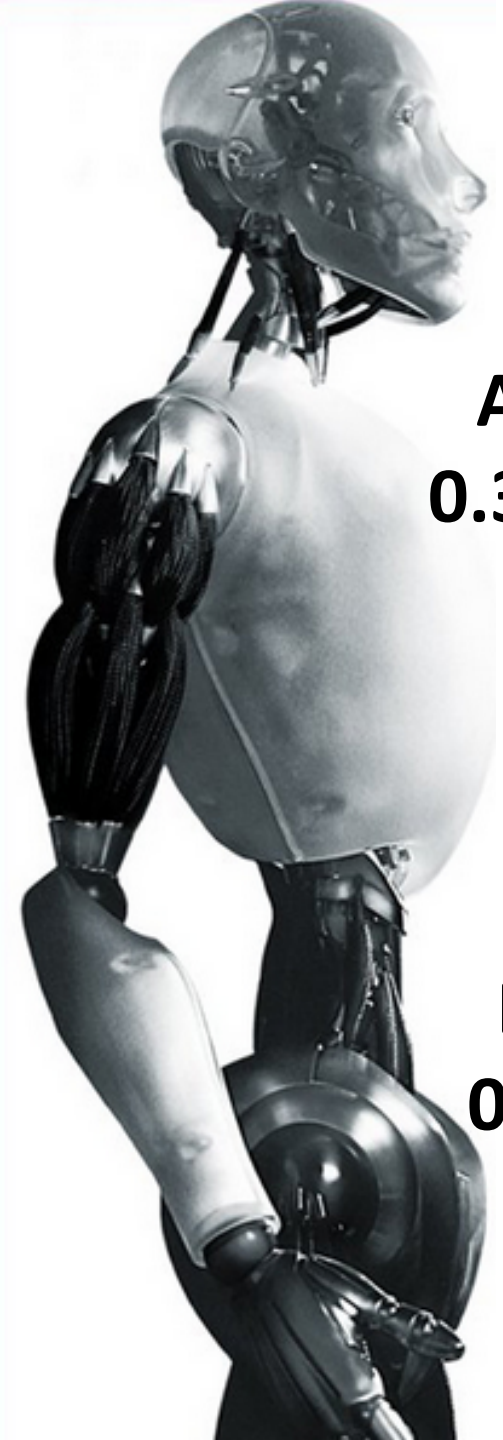
Step 4



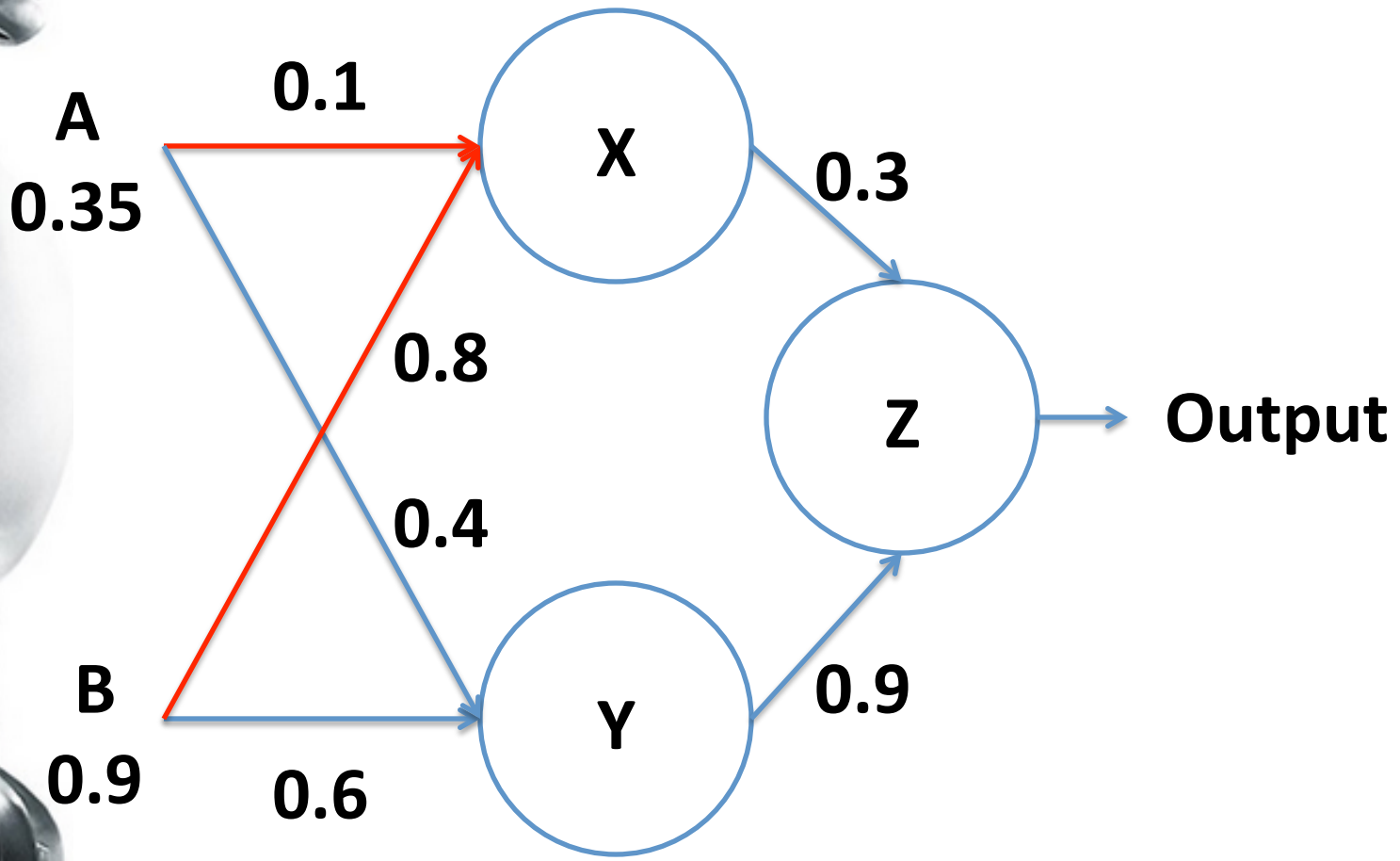


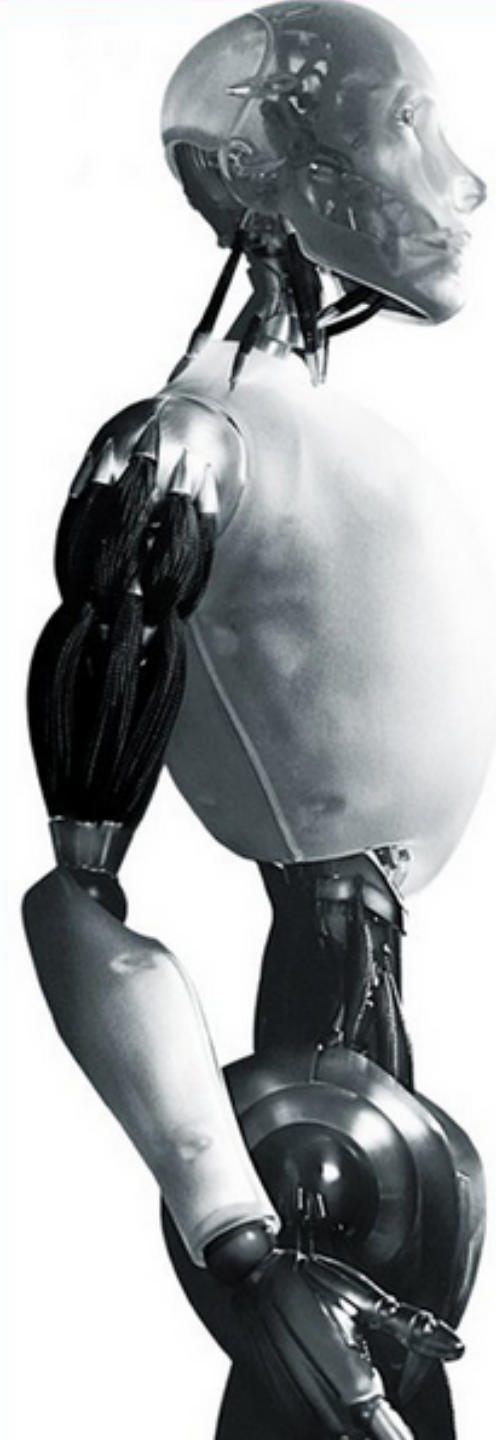
Example 2





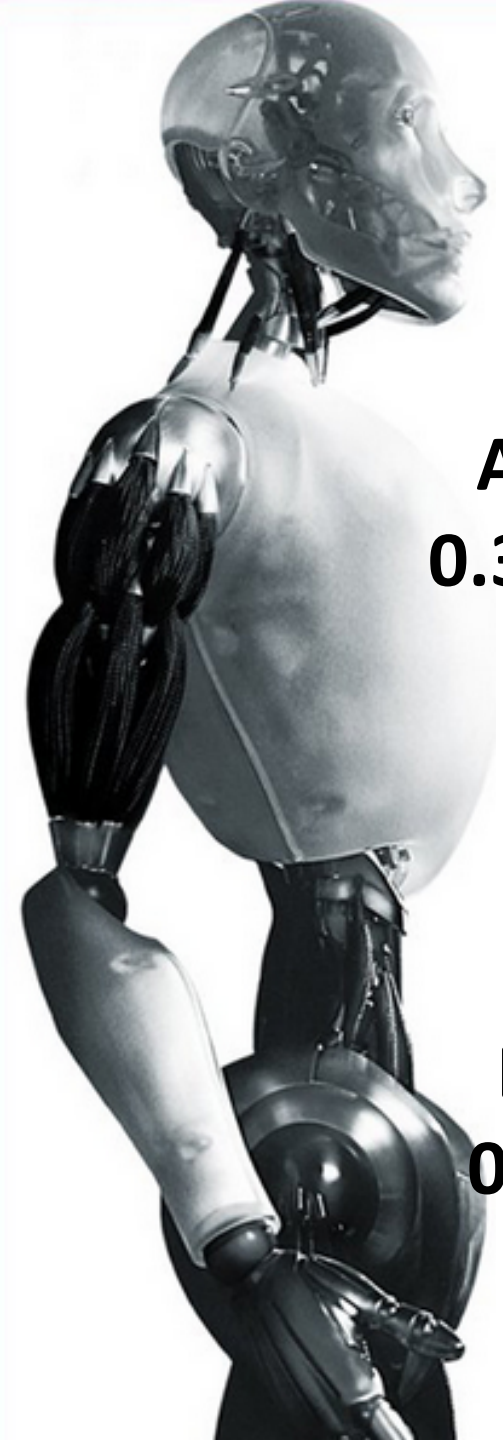
Forward Pass



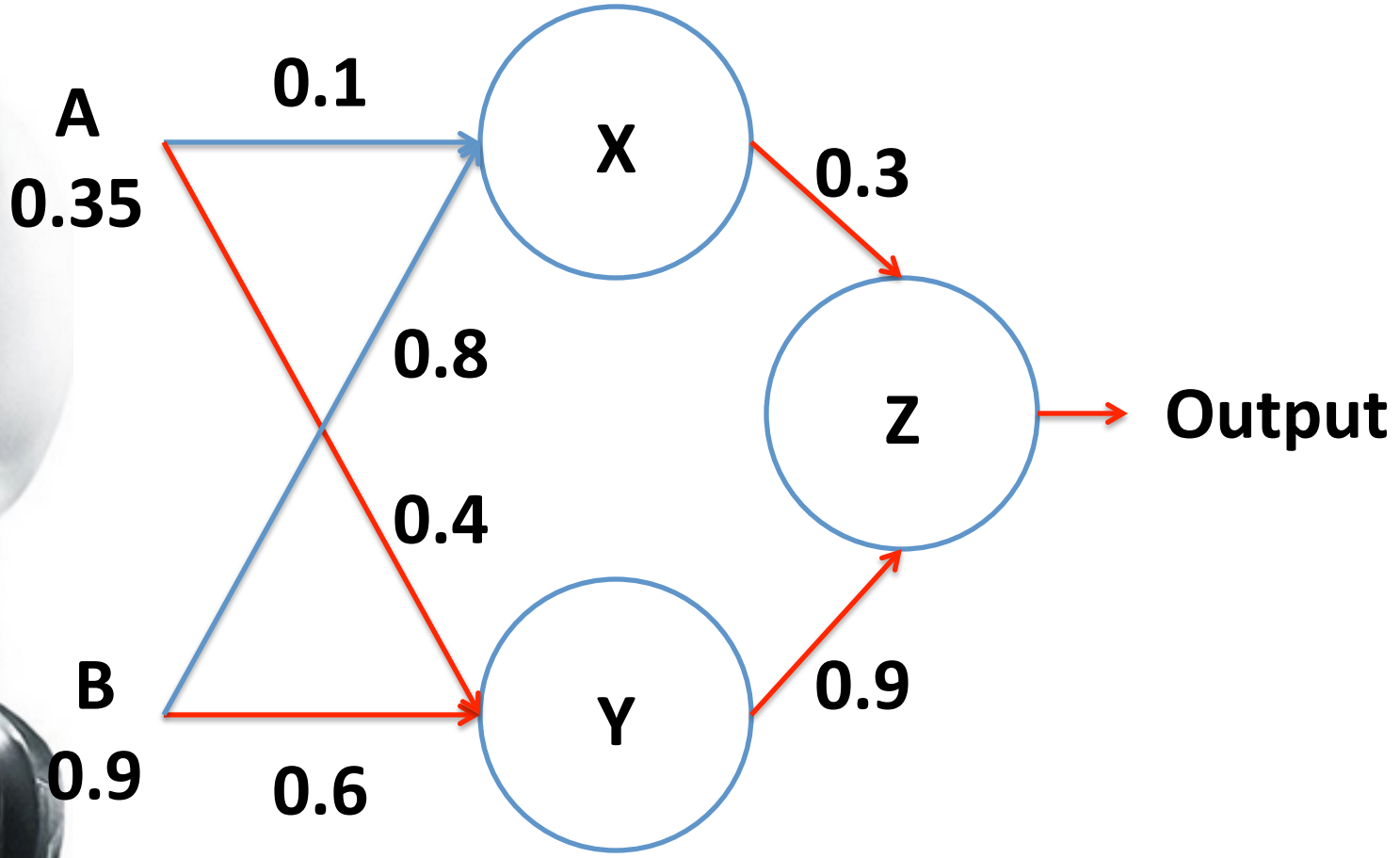


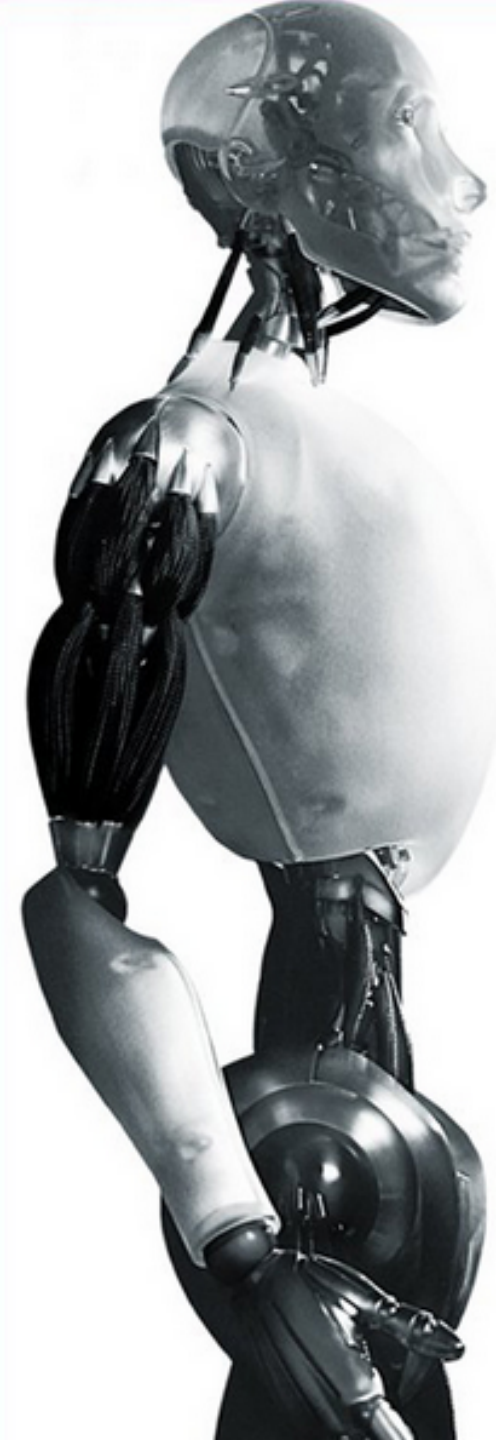
Forward Pass

- X
- (0.35 x 0.1) + (0.9 x 0.8) = 0.755
- Sigmoid $$P(t) = \frac{1}{1 + e^{-t}}$$
- e = Euler's number = 2.71828
- t = Weights x Inputs = 0.755
- Output = 0.68



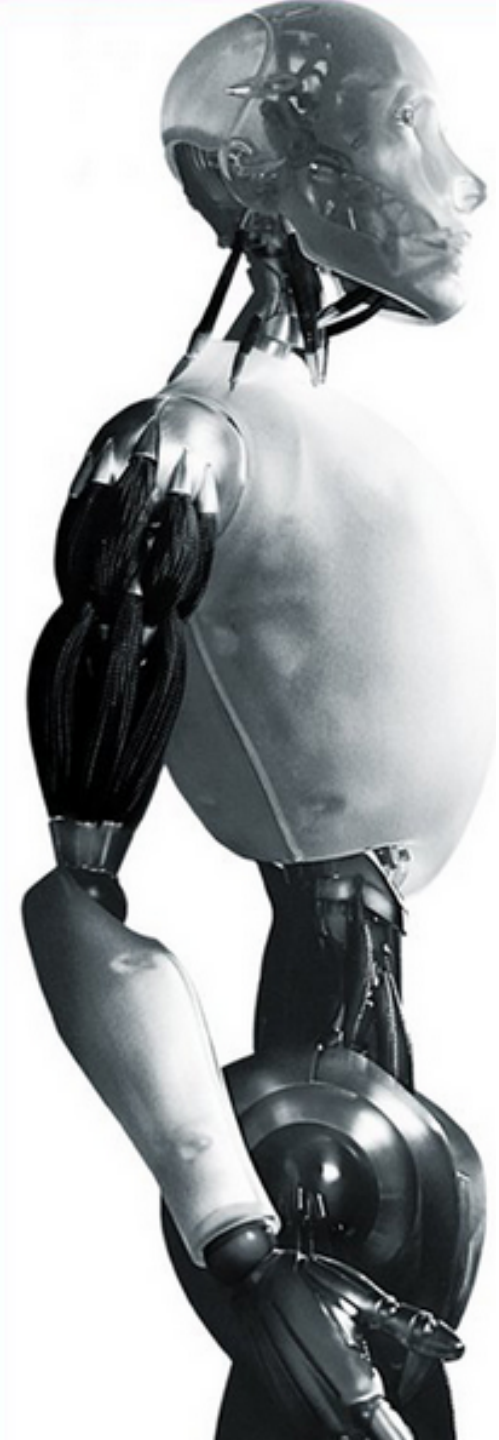
Calculate the remaining Forward pass





Forward Pass

- X
  - (0.35x0.1)+(0.9x0.8)=0.755
  - Out = 0.68
- Y
  - (0.9x0.6)+(0.35x0.4) = 0.68
  - Out = 0.6637
- Z
  - (0.3x0.68)+(0.9x0.6637) = 0.80133
  - Out = 0.69

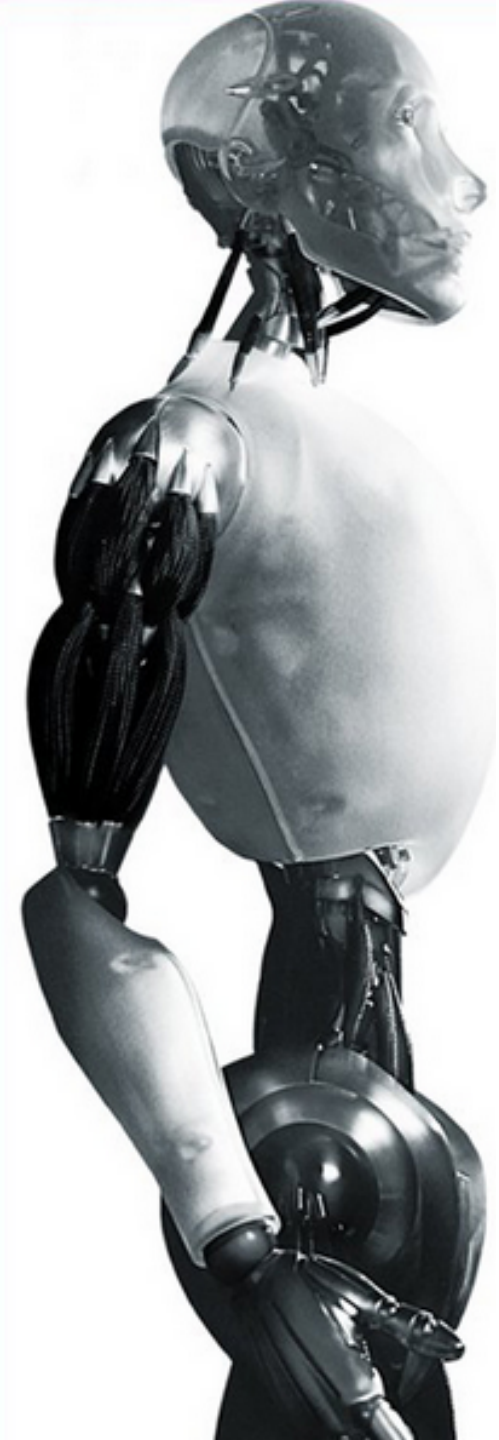


Reverse Pass (assuming target is 0.5)

- Calculate the error …

ErrorZ = OutputZ (1-OutputZ)(TargetZ – OutputZ)

0.69(1-0.69)(0.5-0.69) = -0.0406



Adjust the weights

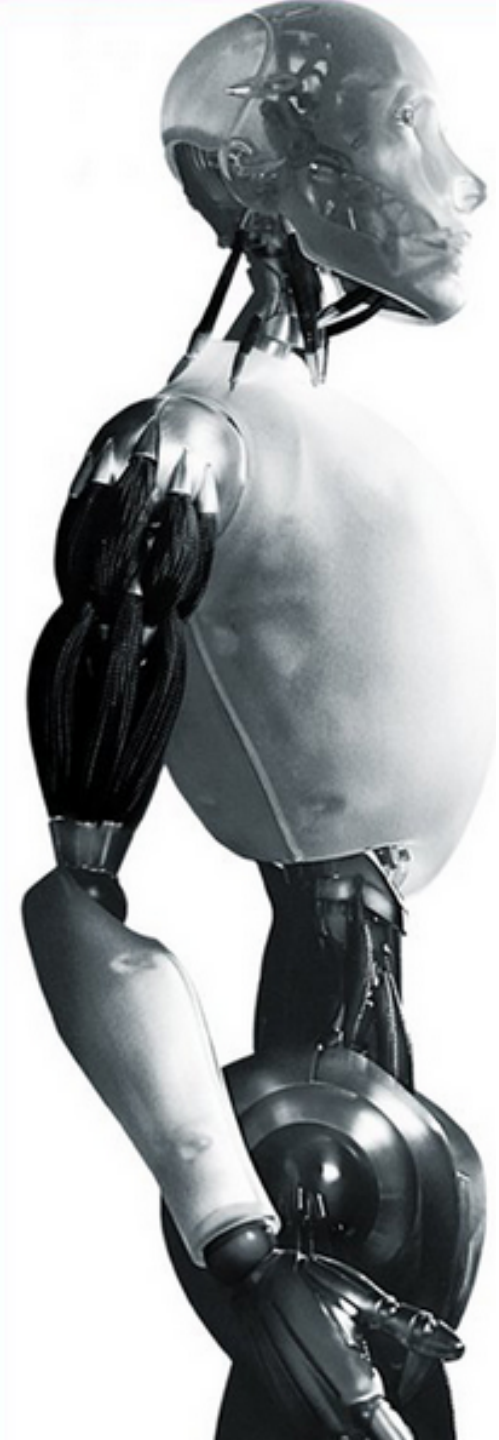
W = W + (Error x Output)

X

0.3 + (-0.0406x0.68) = 0.272392

Y

0.9 + (-0.0406x0.6637) = 0.87305



Calculate the error of hidden layers

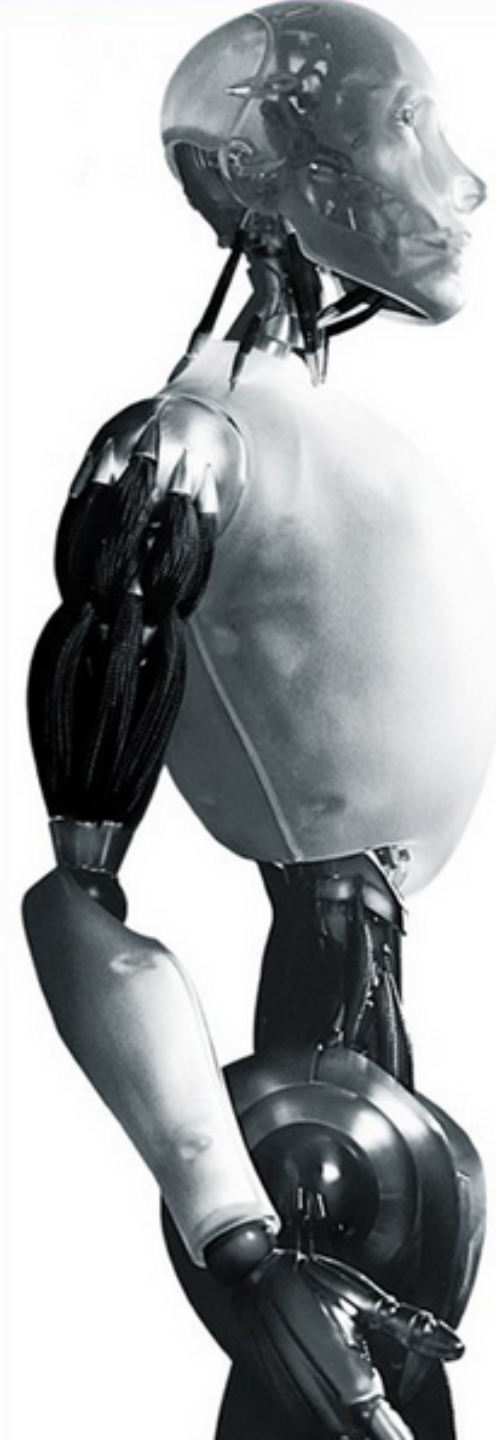
- Error =

  (Error x W)

  (1 – Output) Output

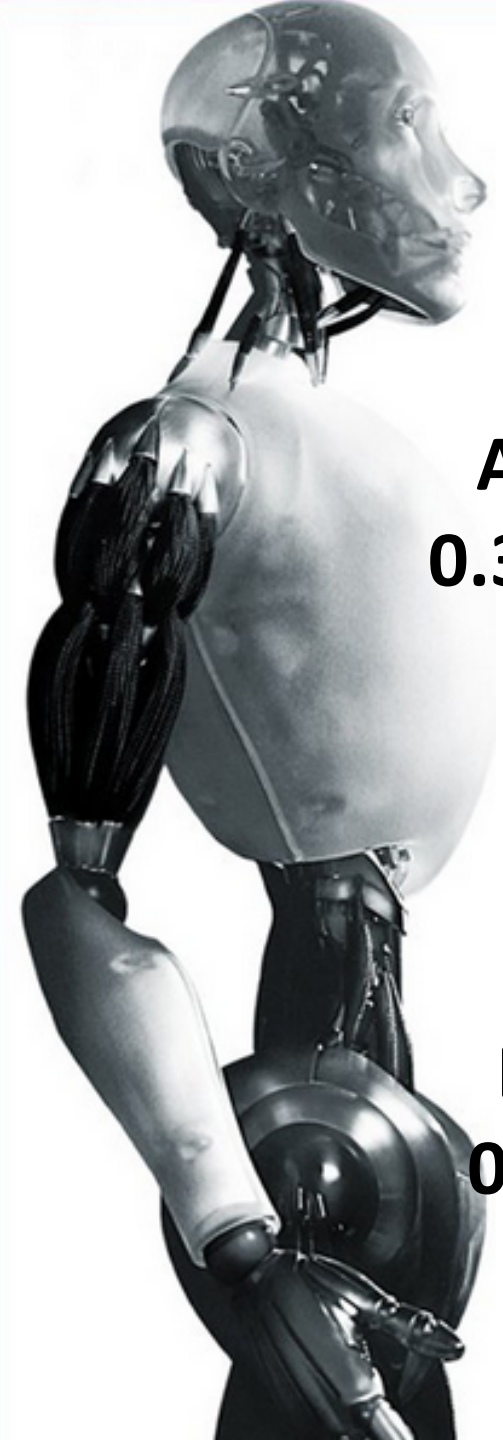
- -0.0406 x 0.272392 x (1-0.68) 0.68 = -2.406x10-3

- -0.0406 x 0.87305 x (1-0.6637)0.6637 = -7.91x10-3

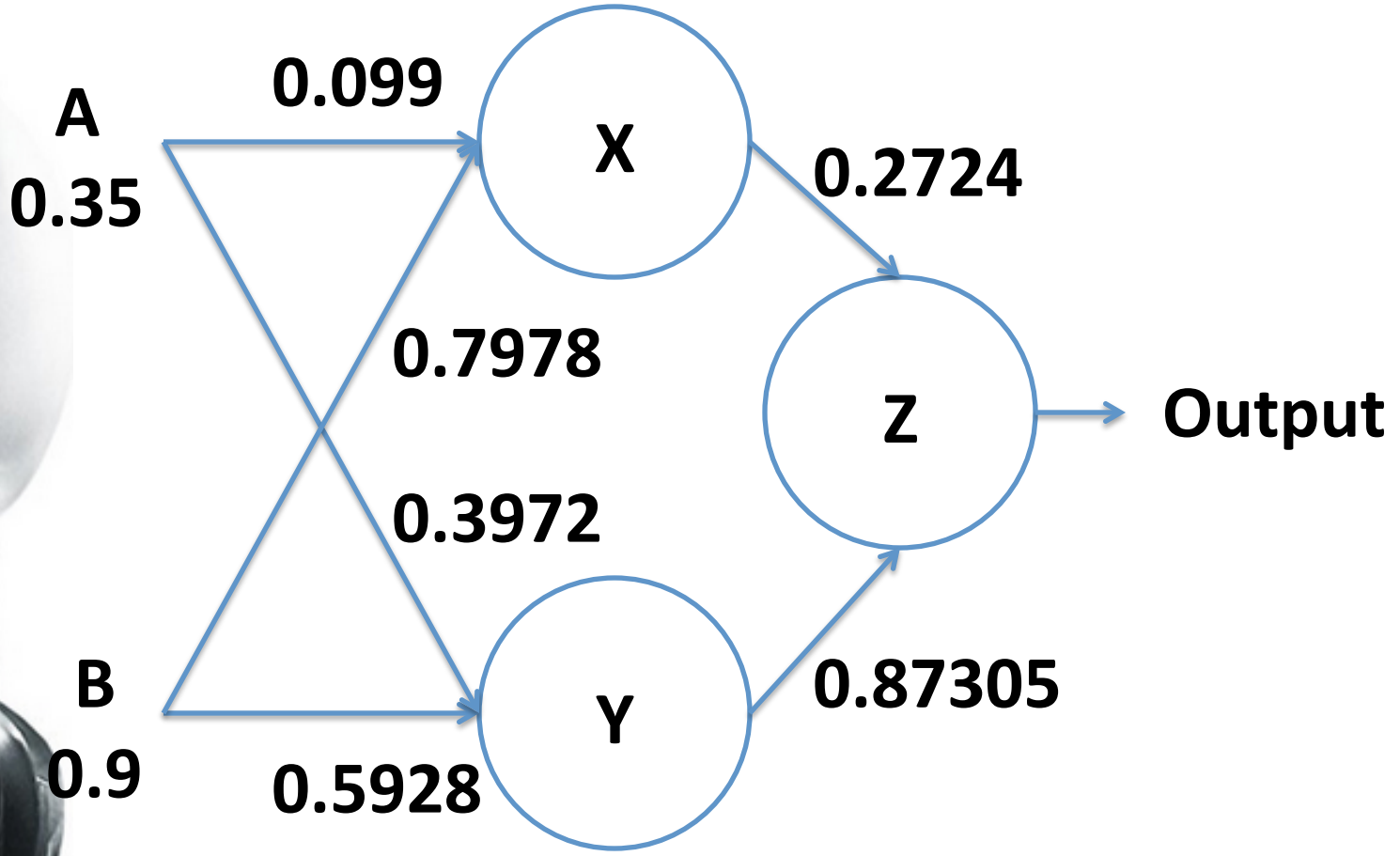


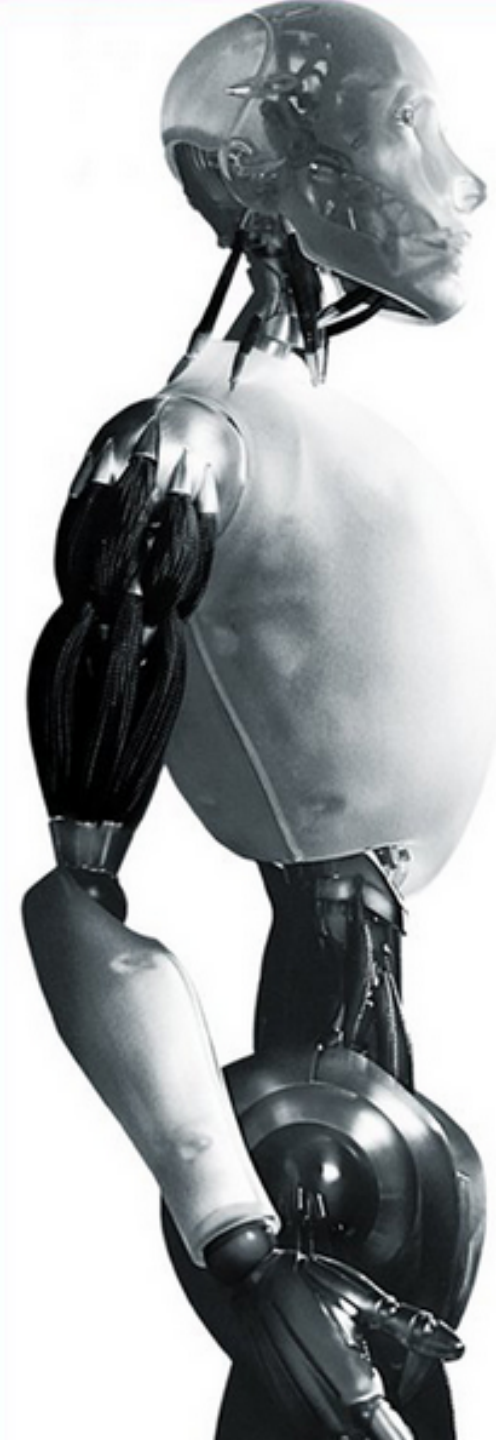
Adjust the weights

- **W = W + (Error x Output)**
- $W = 0.1 + (-2.406 \times 10^{-3} \times 0.35) = 0.09916$
- $W = 0.8 + (-2.406 \times 10^{-3} \times 0.9) = 0.7978$
- $W = 0.4 + (-7.91 \times 10^{-3} \times 0.35) = 0.3972$
- $W = 0.6 + (-7.91 \times 10^{-3} \times 0.9) = 0.5928$



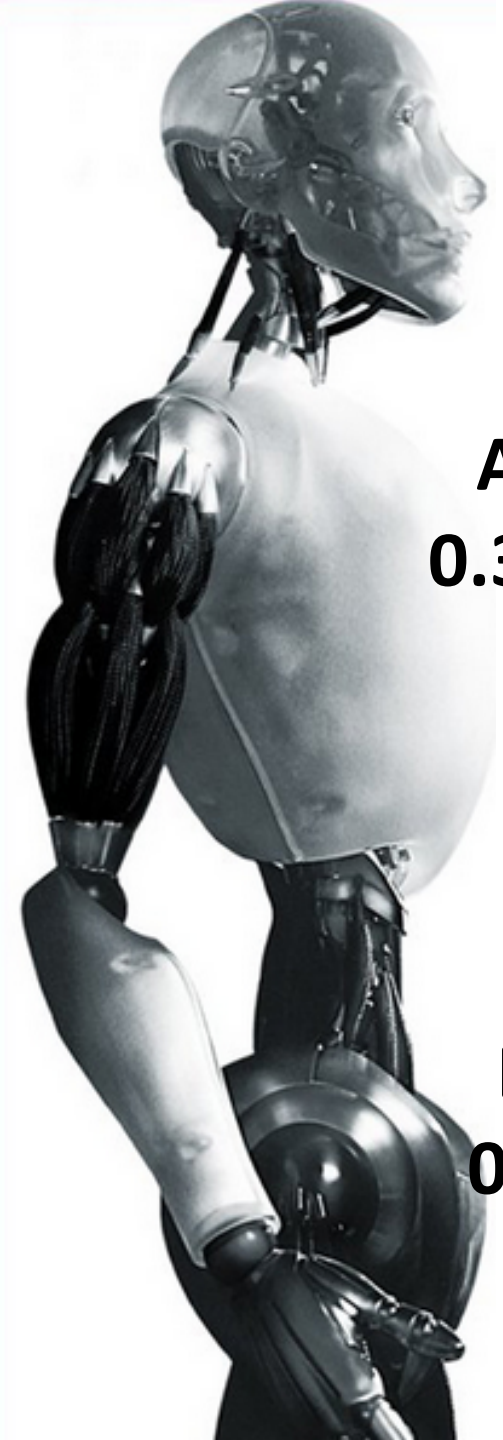
New Network



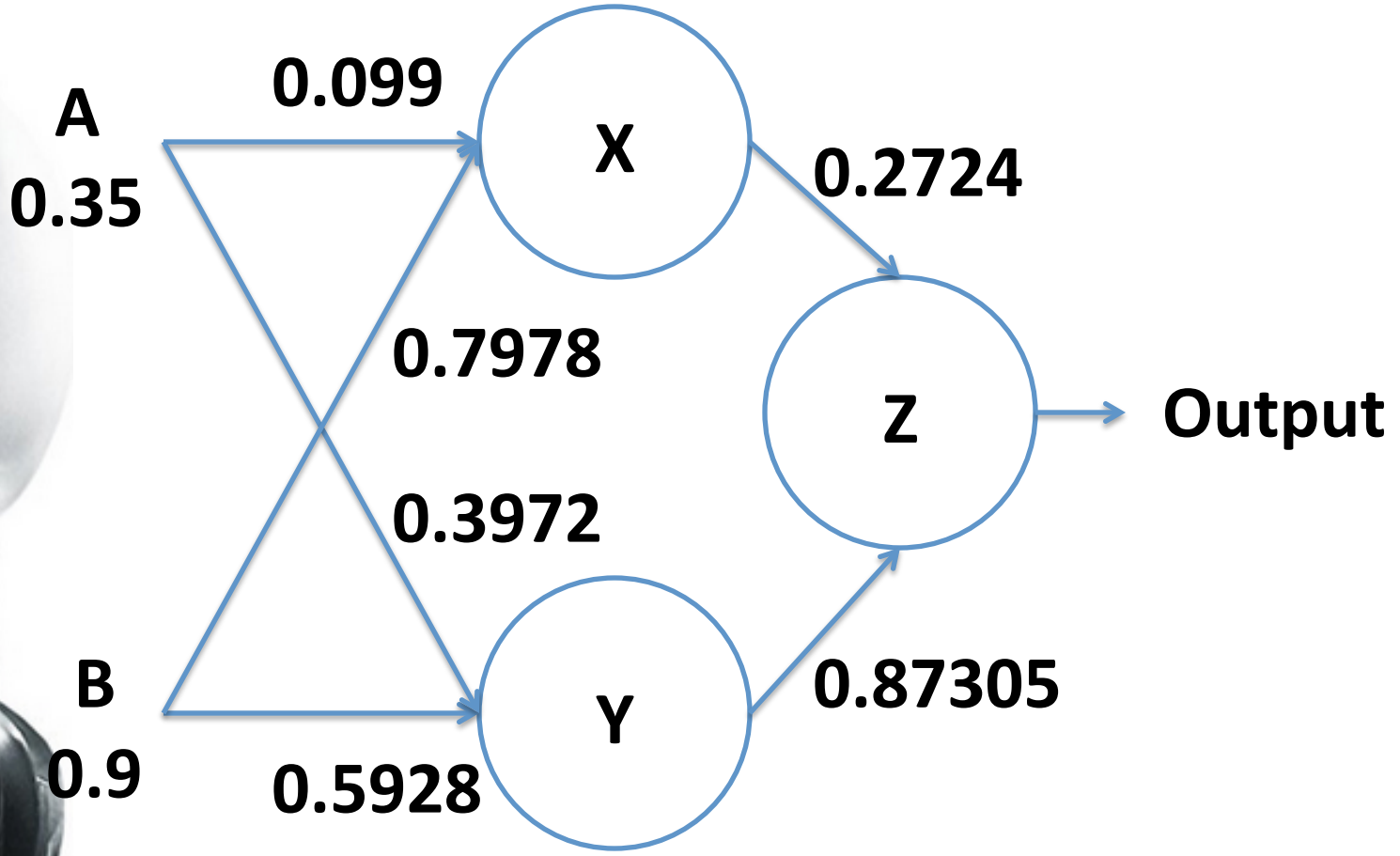


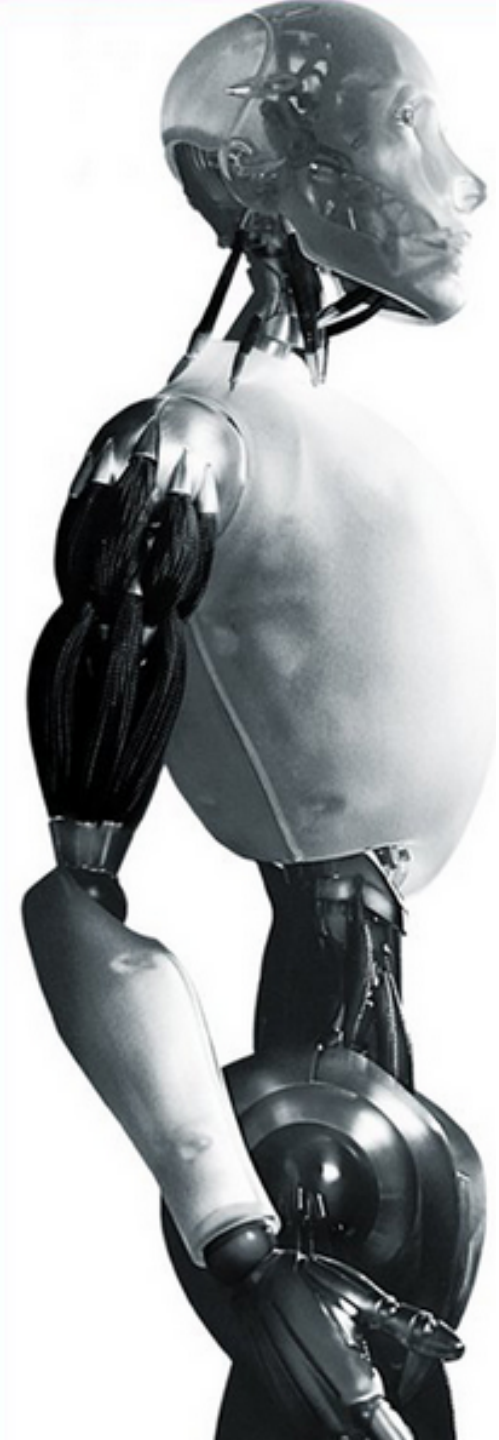
Forward Pass

- X
  - (0.35 x 0.099) + (0.9 x 7978) = 0.753
- Sigmoid $$P(t) = \frac{1}{1 + e^{-t}}$$
- e = Euler's number = 2.71828
- t = Weights x Inputs = 0.753
- Output = 0.6798



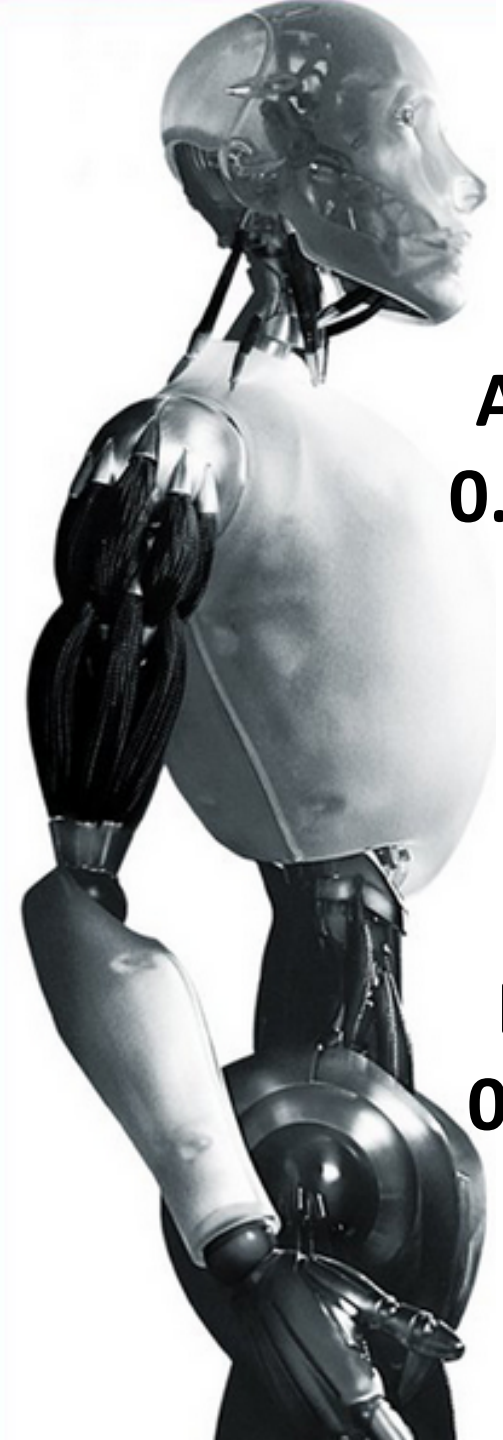
Calculate the remaining Forward pass



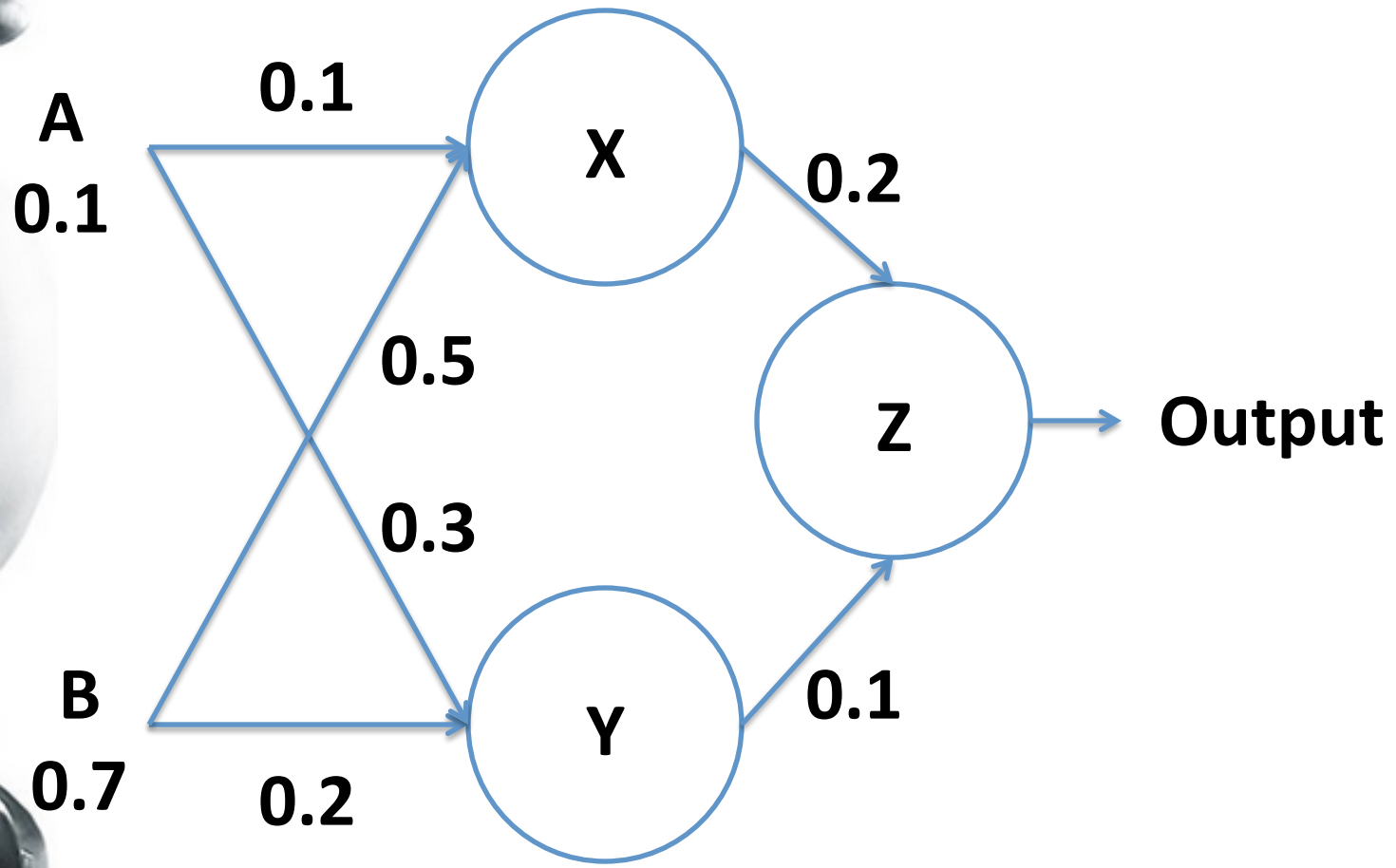


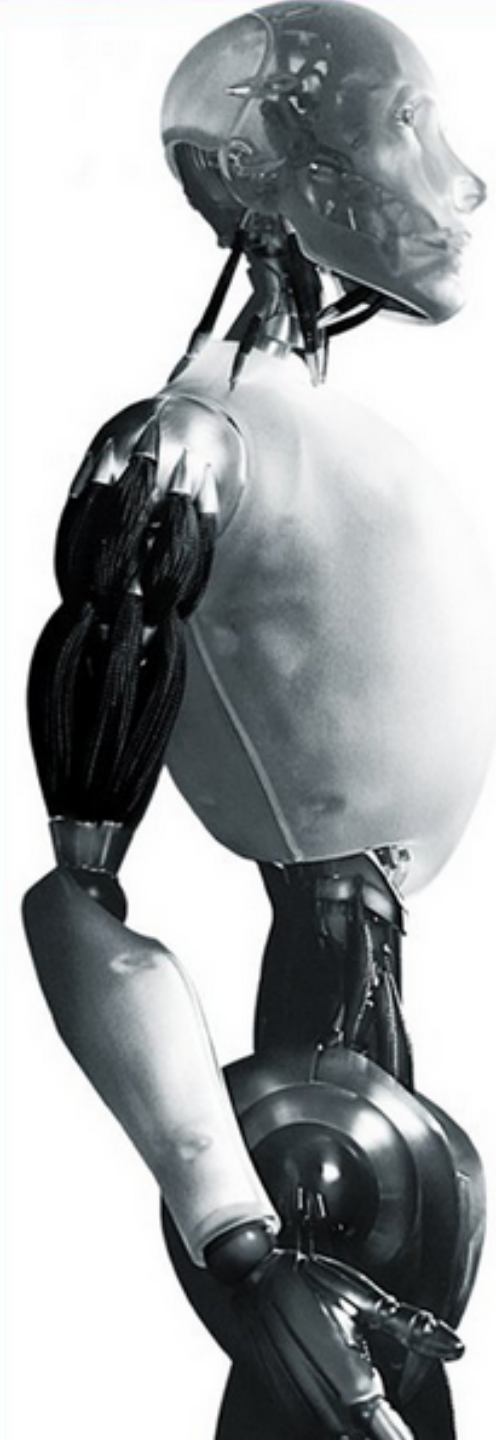
New Forward Pass

- X
  - (0.35x0.099)+(0.9x0.7978)=0.753
  - Out = 0.6798
- Y
  - (0.9x0.5928)+(0.35x0.3972) = 0.67254
  - Out = 0.6621
- Z
  - (0.2724x0.6798)+(0.87305x0.6621) = 0.763
  - Out = 0.682
- Old Error = 0.5 – 0.69 = -0.19
- New Error = 0.5 – 0.682 = -0.182

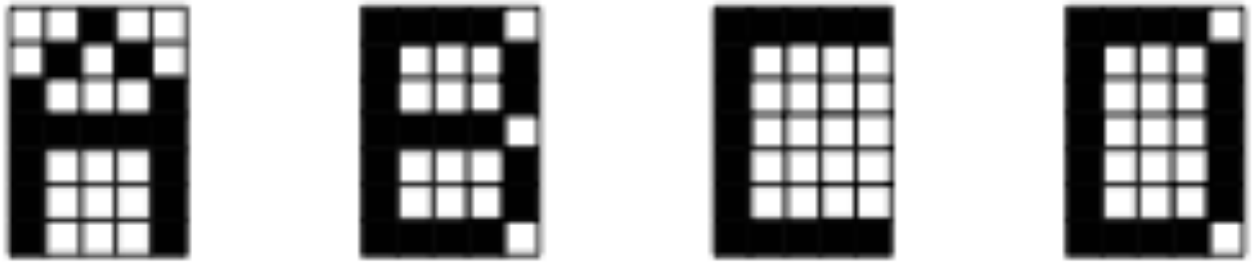


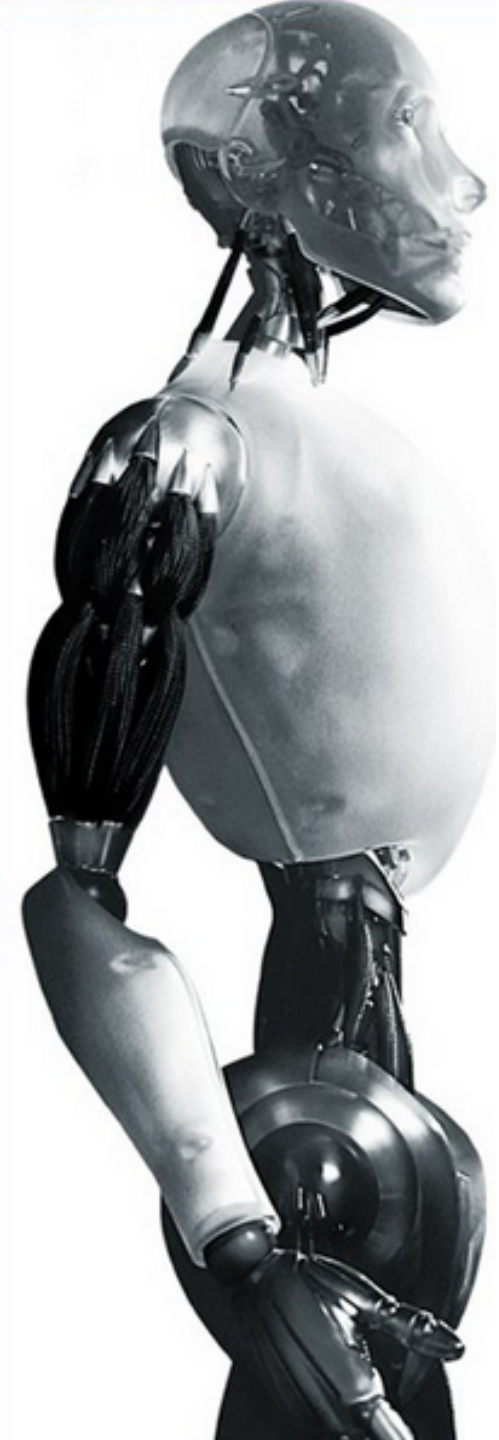
Exercise: Target output 1



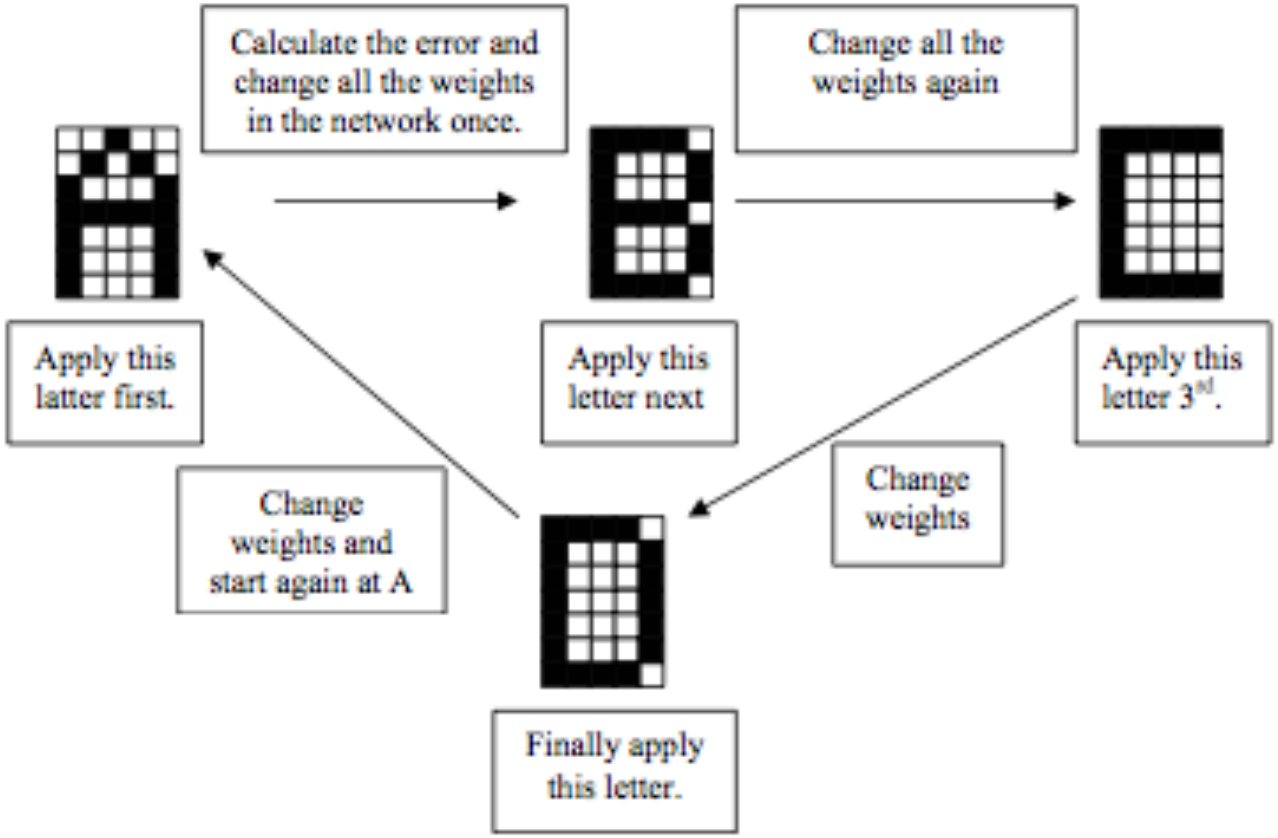


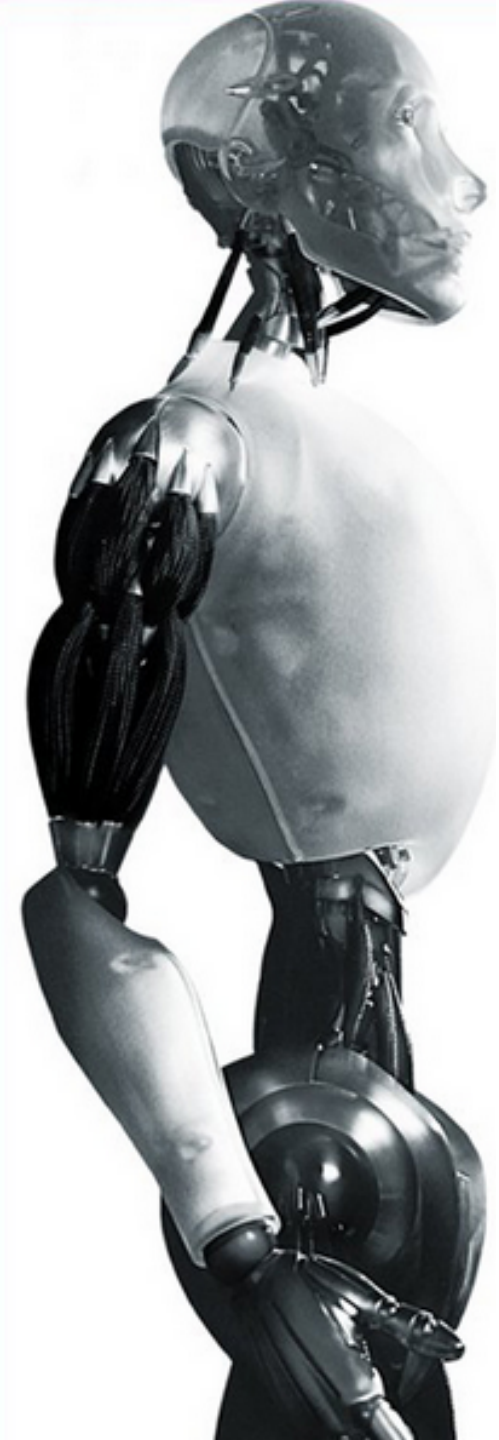
How do we create an OCR?





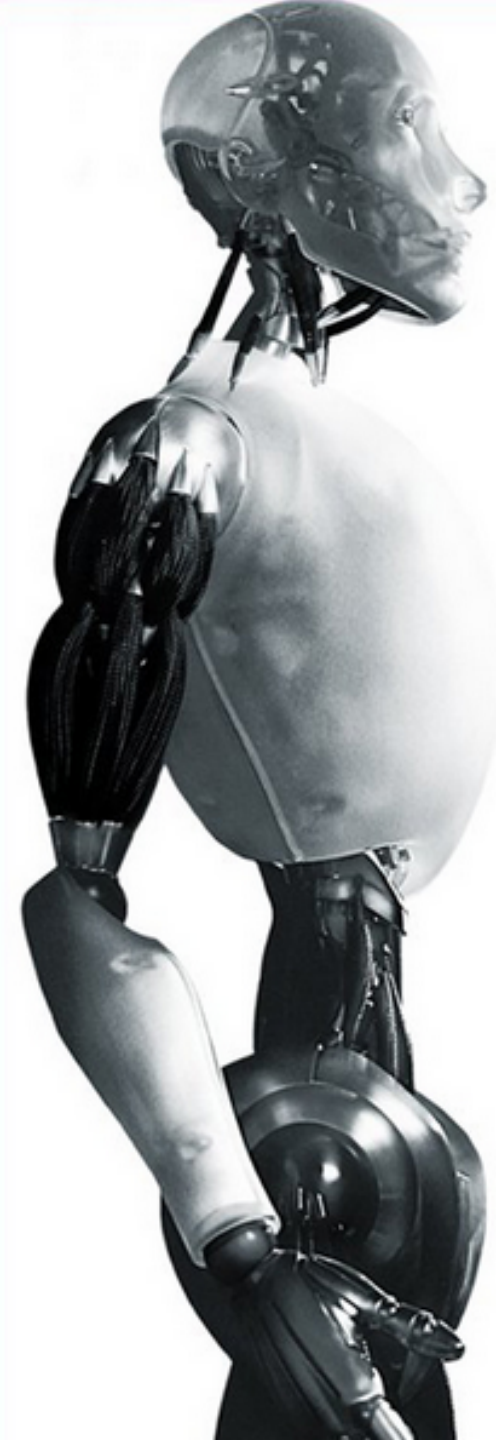
Training


Calculate the error and change all the weights in the network once.
Change all the weights again
Apply this latter first.
Apply this letter next
Apply this letter 3rd.
Change weights and start again at A
Change weights
Finally apply this letter.

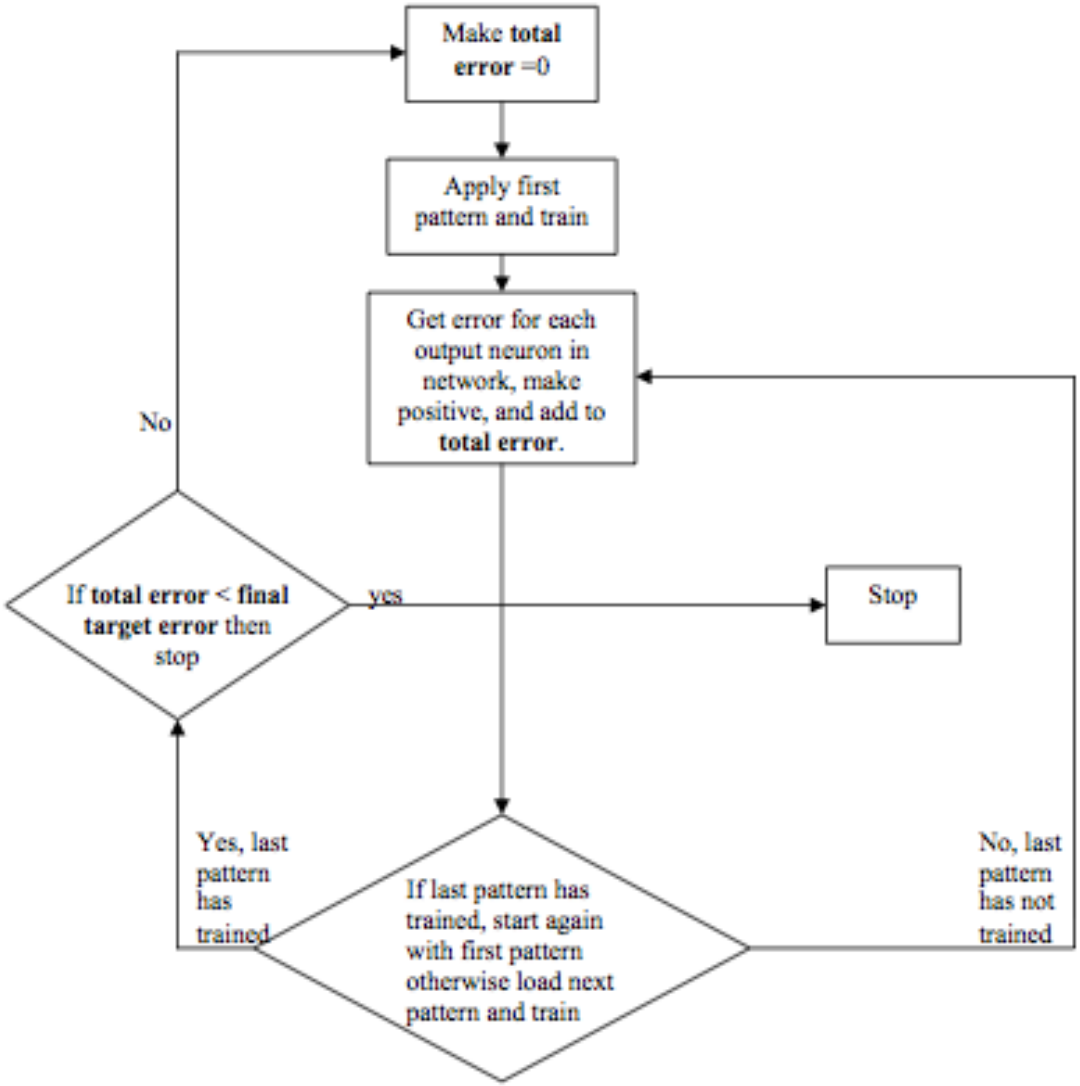


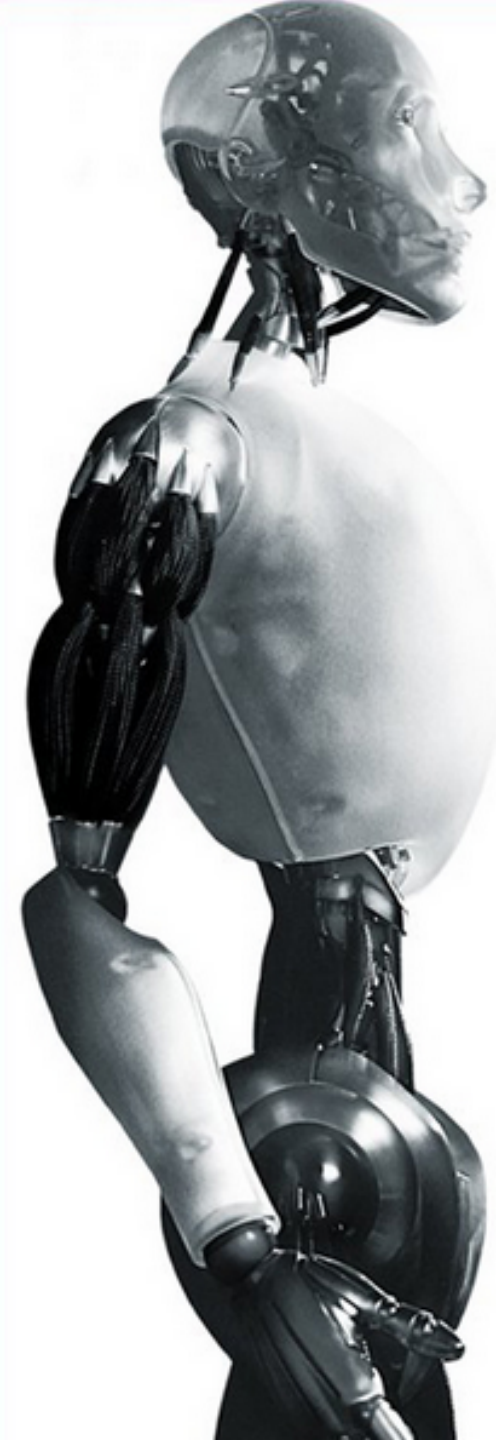
When to stop the learning?

- When all the letters will be learnt?
- When the error level is low.



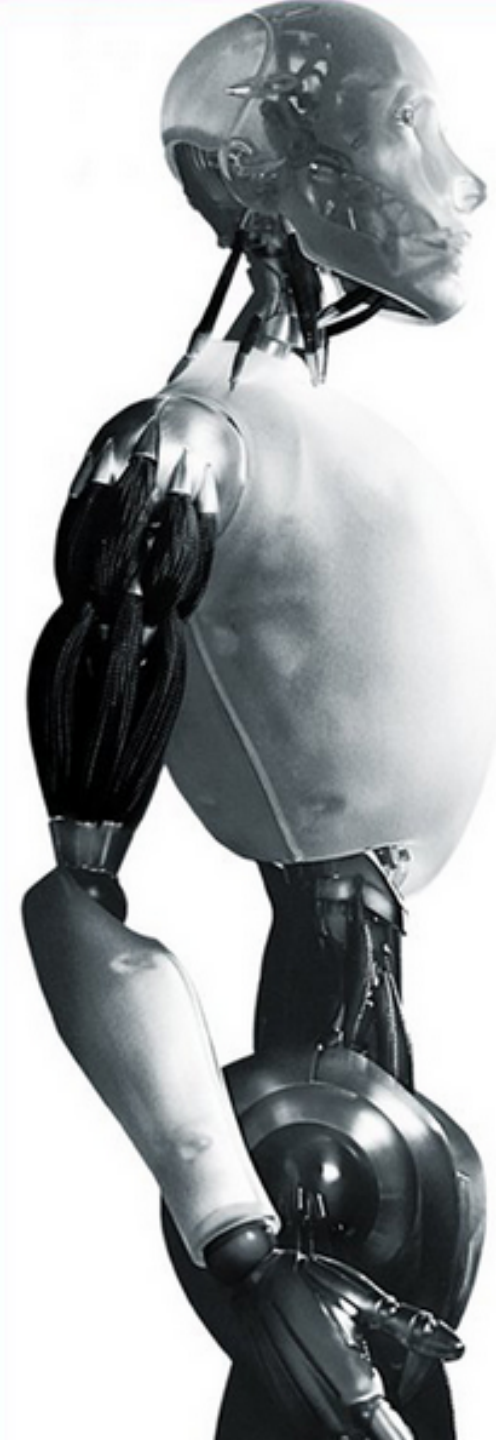
Calculating Total Error





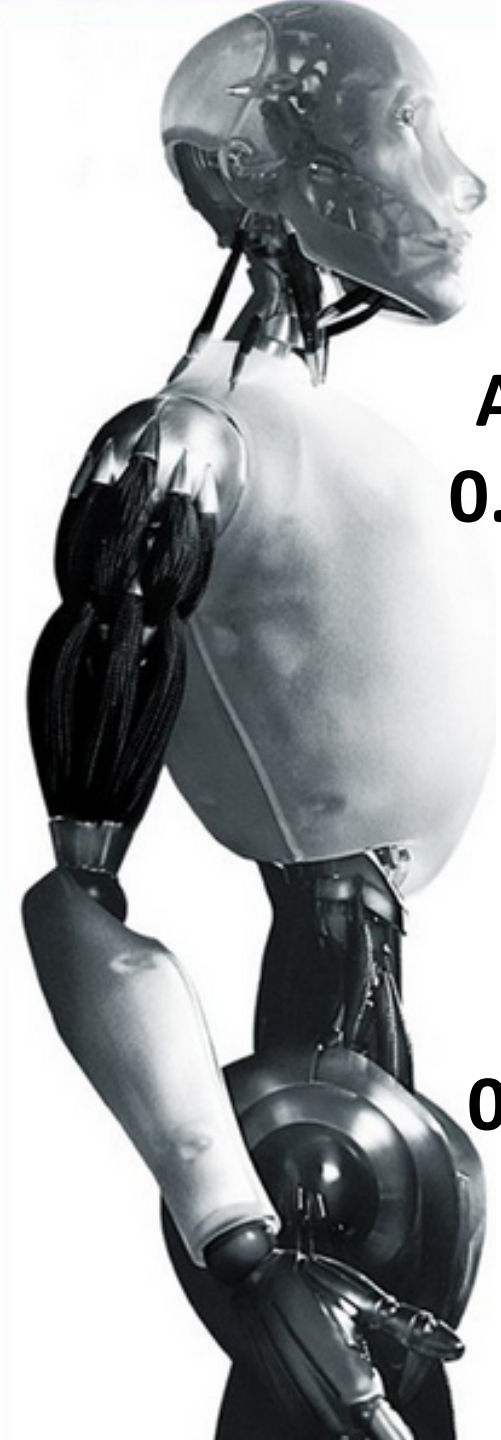
Problems of Backpropagation

- Reaching a local minima (Getting stuck)
- Possible solution is to reset the weights and restart

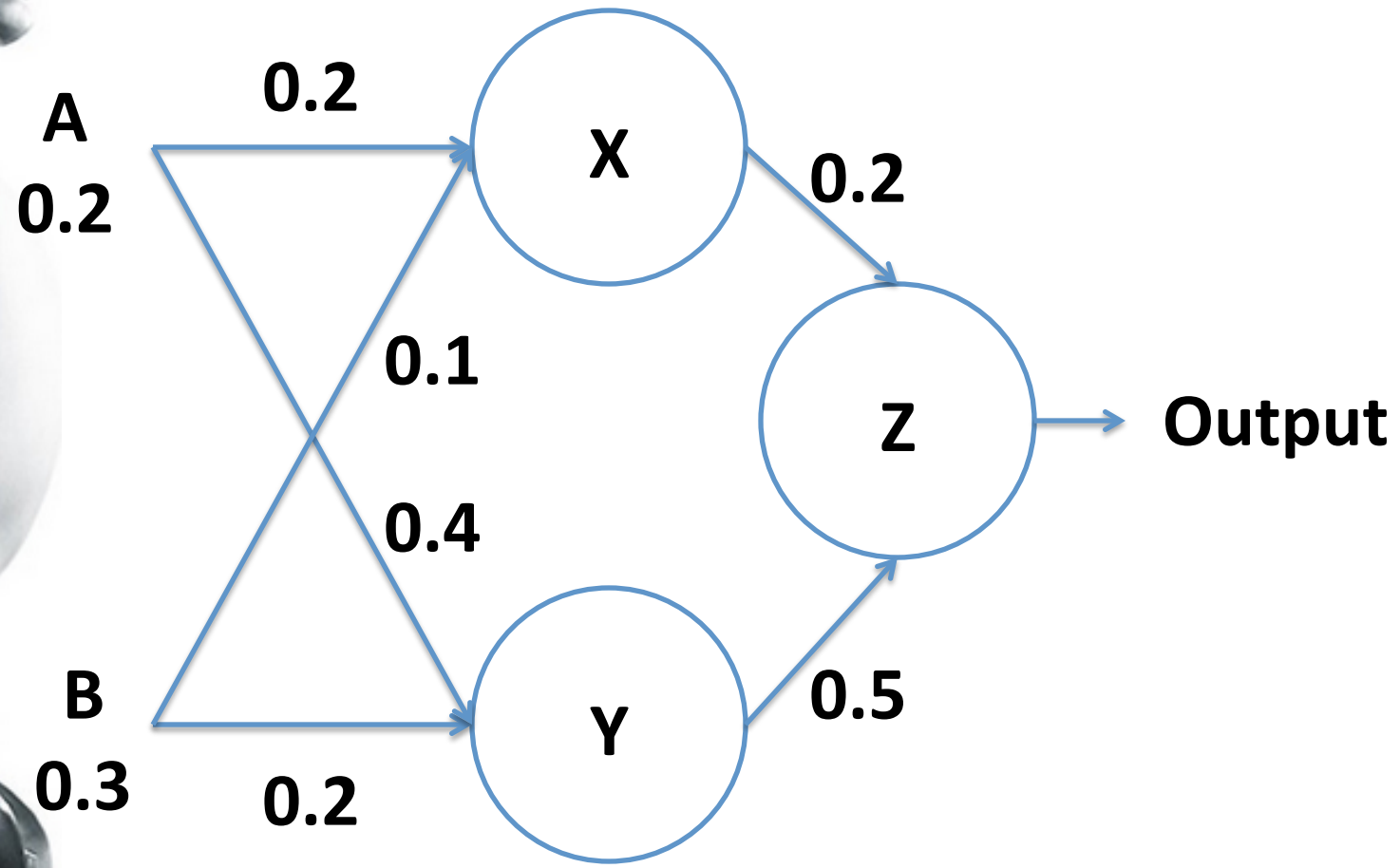


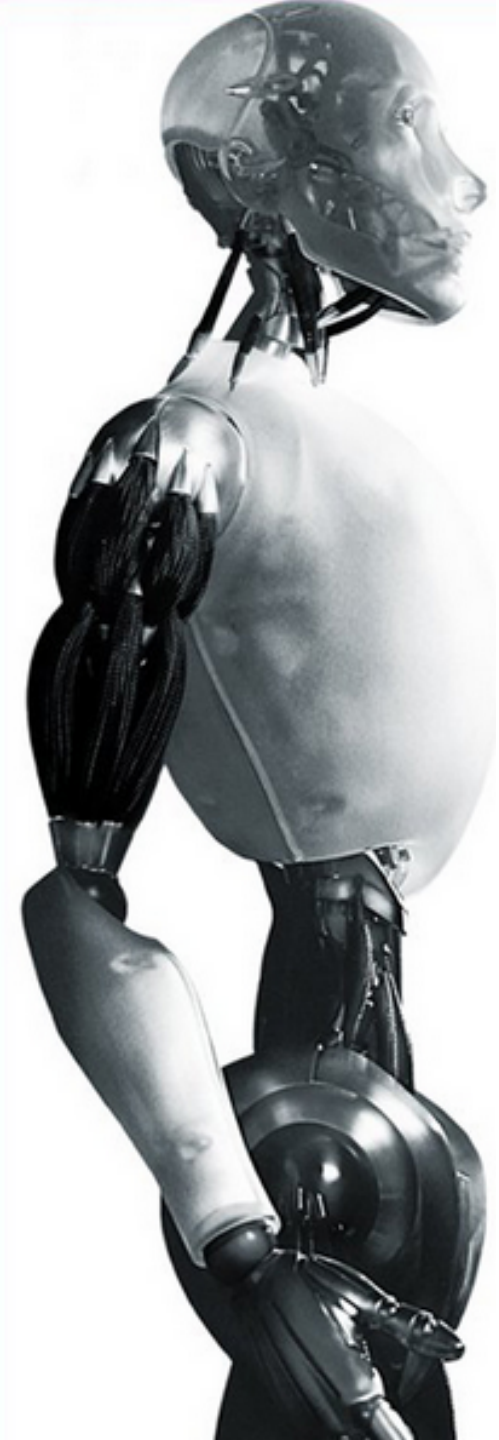
How many hidden layers?

- No specific rule
- Deduced through experimentation



Exercise: Target output 0.6





Questions?