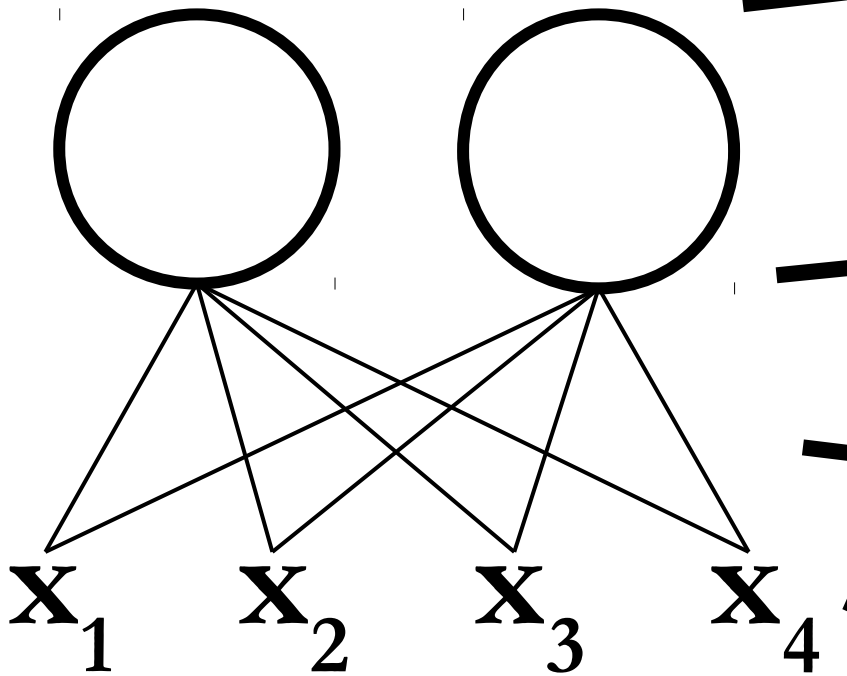
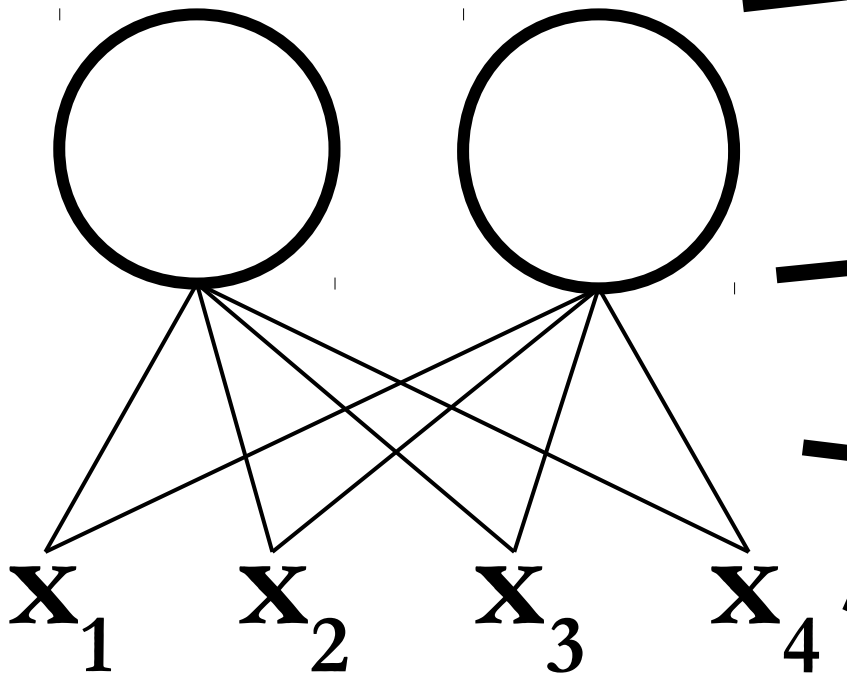



```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21 -1.65]
 [2.32  3.45]
 [0.38  5.30]]
```

```
[[ 0.9  0.8 -1.0 -1.0]
 [-0.5 -0.5  1.5  1.0]]
```

```
[[4.9  3.0  1.4  0.2]
 [6.4  3.2  4.5  1.5]
 [5.8  2.7  5.1  1.9]]
```

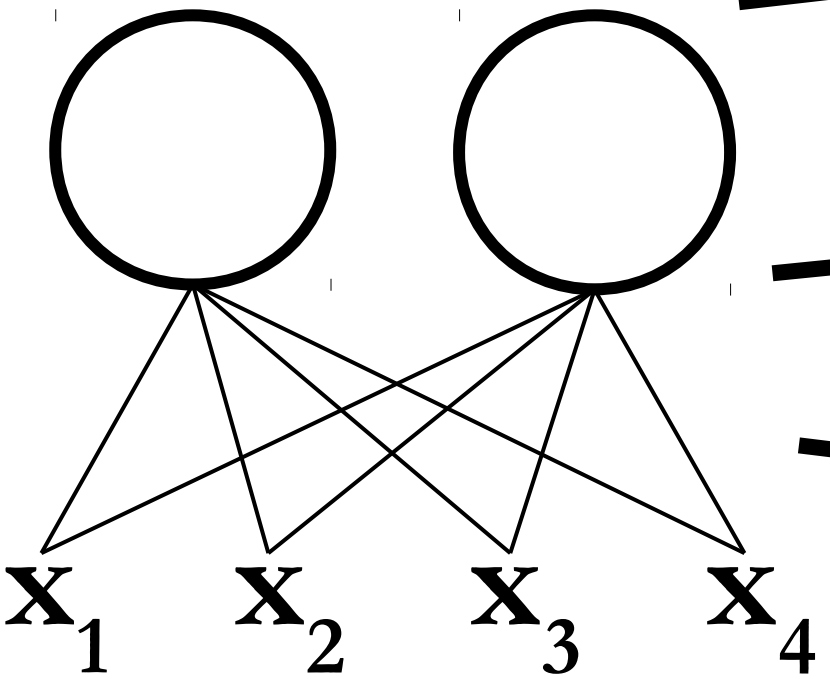



```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21 -1.65]
 [2.32  3.45]
 [0.38  5.30]]
```

```
[[ 0.9   0.8  -1.0  -1.0]
 [-0.5  -0.5   1.5   1.0]]
```

```
[[4.9  3.0  1.4  0.2]
 [6.4  3.2  4.5  1.5]
 [5.8  2.7  5.1  1.9]]
```

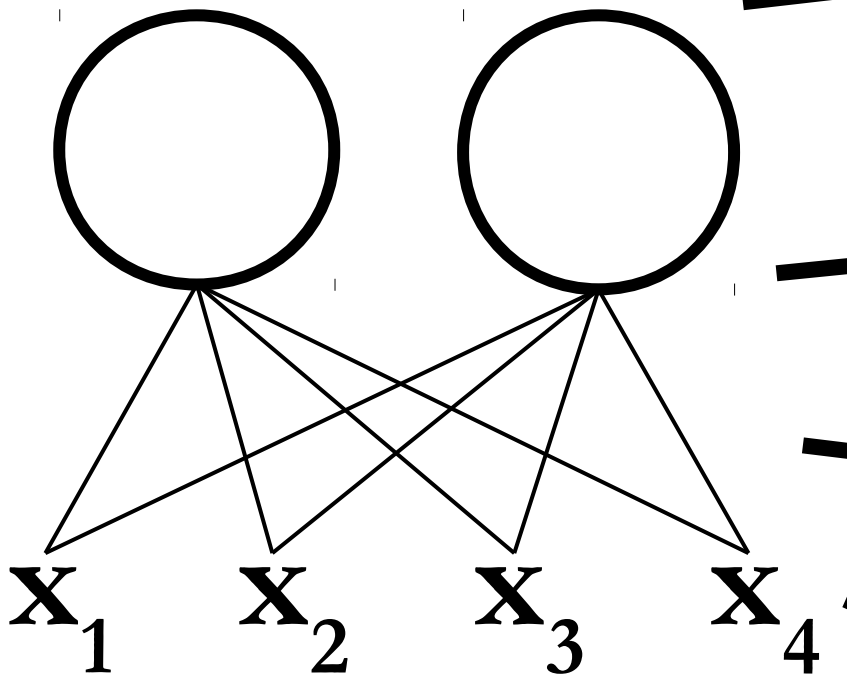



```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

```
[[ 0.9   0.8  -1.0  -1.0]
 [-0.5  -0.5   1.5   1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```

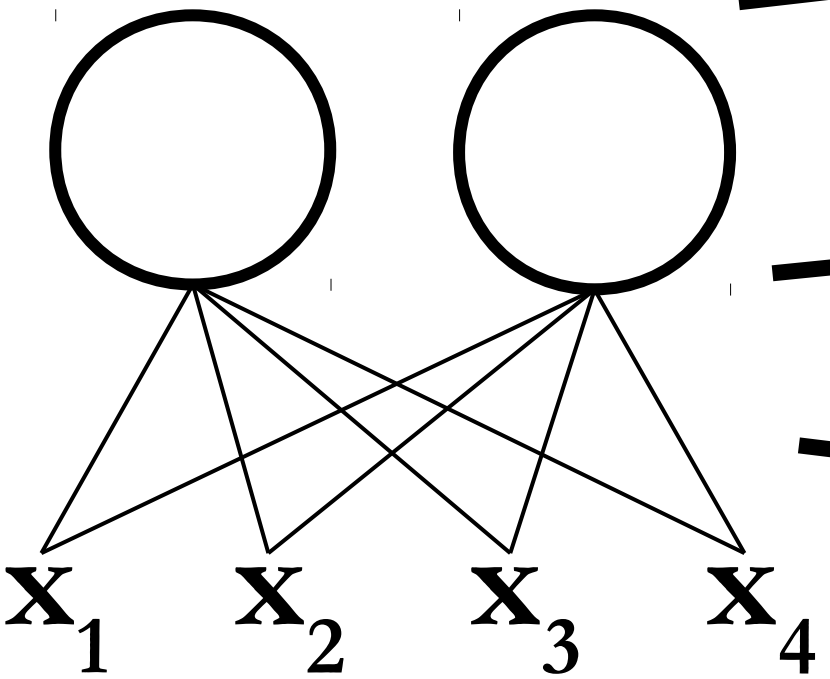



```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

```
[[ 0.9   0.8  -1.0  -1.0]
 [-0.5  -0.5  1.5    1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```

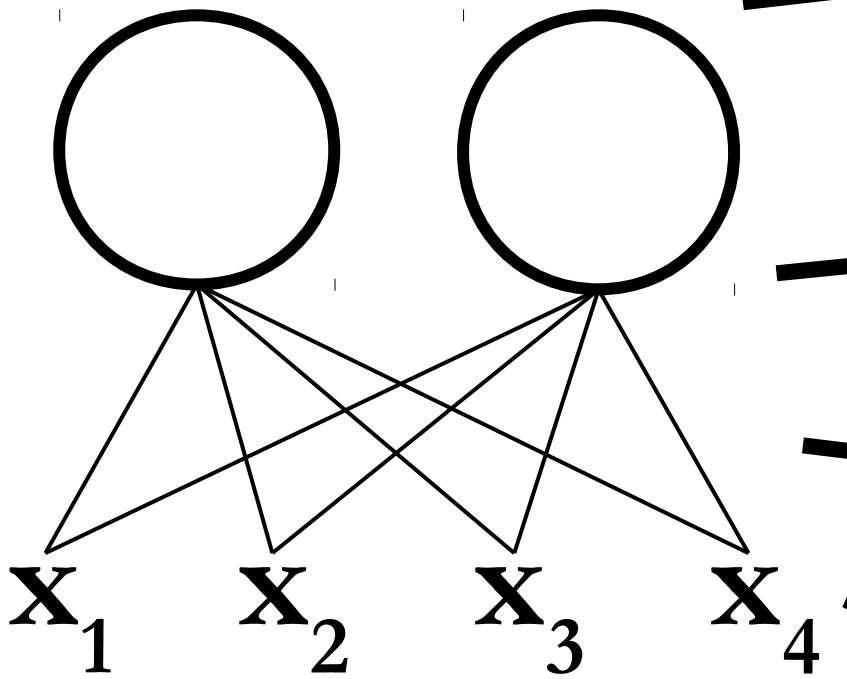



```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

```
[[ 0.9   0.8  -1.0  -1.0]
 [-0.5  -0.5   1.5   1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```




```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

```
[[ 0.9   0.8  -1.0  -1.0]
 [-0.5  -0.5  1.5    1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2 0.8] [6.4 3.2 4.5 1.5 4.7] [5.8 2.7 5.1 1.9 6.2]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
	3 x 4	4 x 2	

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
	3 x 4	4 x 2	

	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
Matrix Multiplication:	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 -0.5] [0.8 -0.5] [-1.0 1.5] [-1.0 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]
	3 x 4	4 x 2	3 x 2

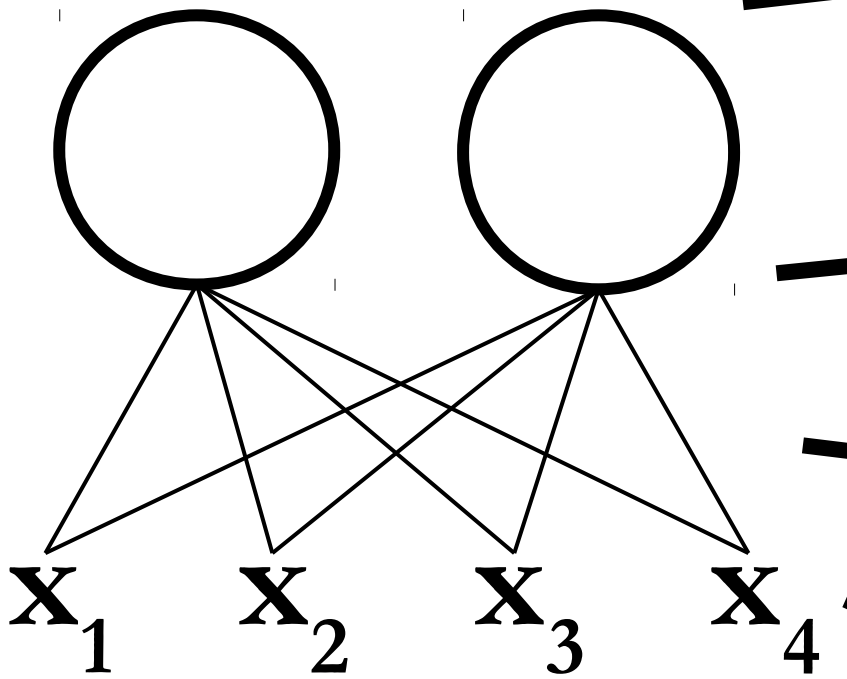
	Inputs	Weights	Result
determine_layer_outputs():	[[4.9 3.0 1.4 0.2] [6.4 3.2 4.5 1.5] [5.8 2.7 5.1 1.9]]	[[0.9 0.8 -1.0 -1.0] [-0.5 -0.5 1.5 1.0]]	[[5.21 -1.65] [2.32 3.45] [0.38 5.30]]

Matrix Multiplication:

```
[[4.9 3.0 1.4 0.2 0.8]
 [6.4 3.2 4.5 1.5 4.7]
 [5.8 2.7 5.1 1.9 6.2]]

[[ 0.9 -0.5]
 [ 0.8 -0.5]
 [-1.0  1.5]
 [-1.0  1.0]]
```

3 x 5 x 4 x 2

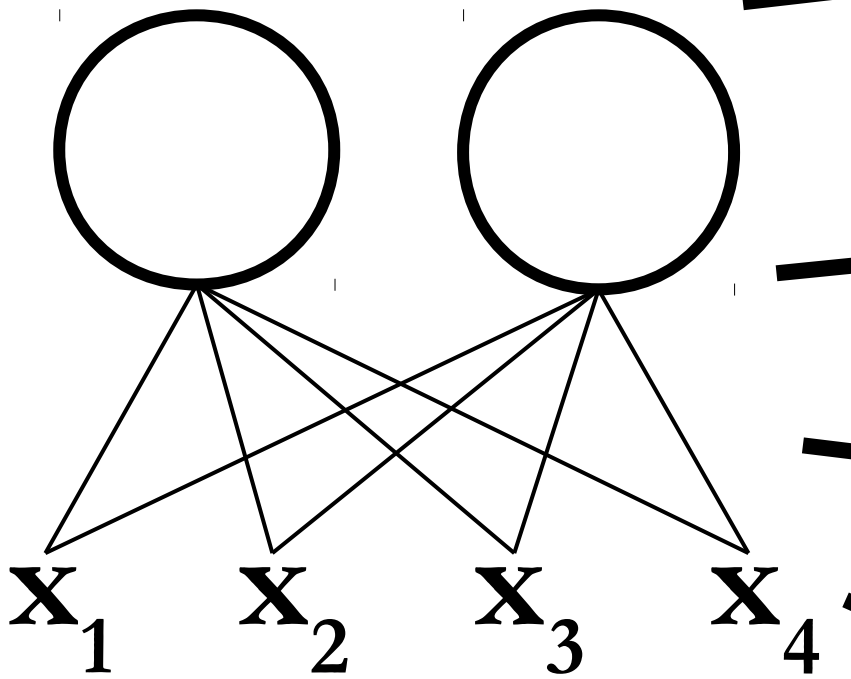



```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

```
[[ 0.9   0.8  -1.0  -1.0]
 [-0.5  -0.5  1.5    1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```




```
[[1  0]
 [1  1]
 [0  1]]
```

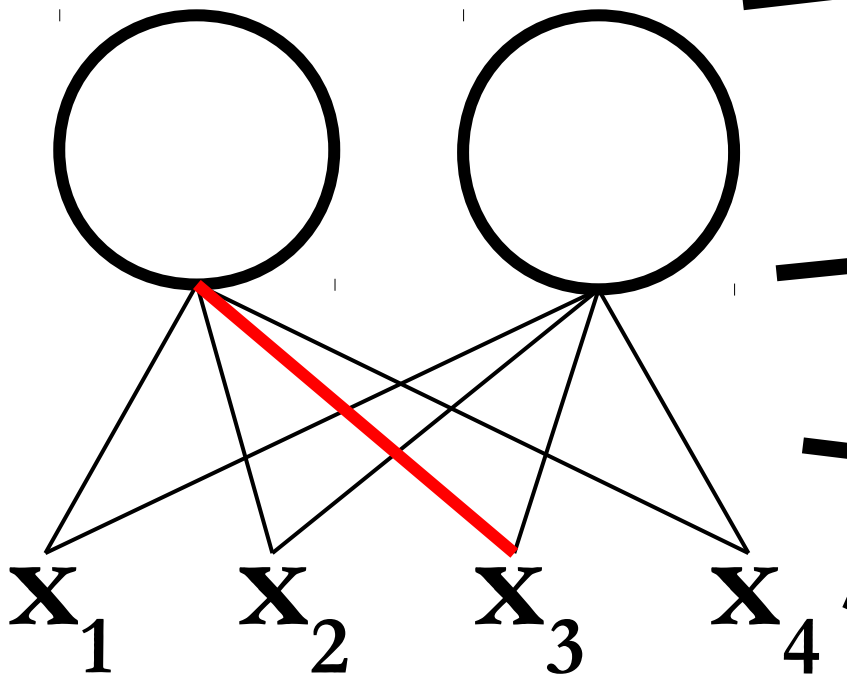
```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

to

from

```
[[ 0.9  -0.5]
 [ 0.8  -0.5]
 [-1.0   1.5]
 [-1.0   1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```




```
[[1  0]
 [1  1]
 [0  1]]
```

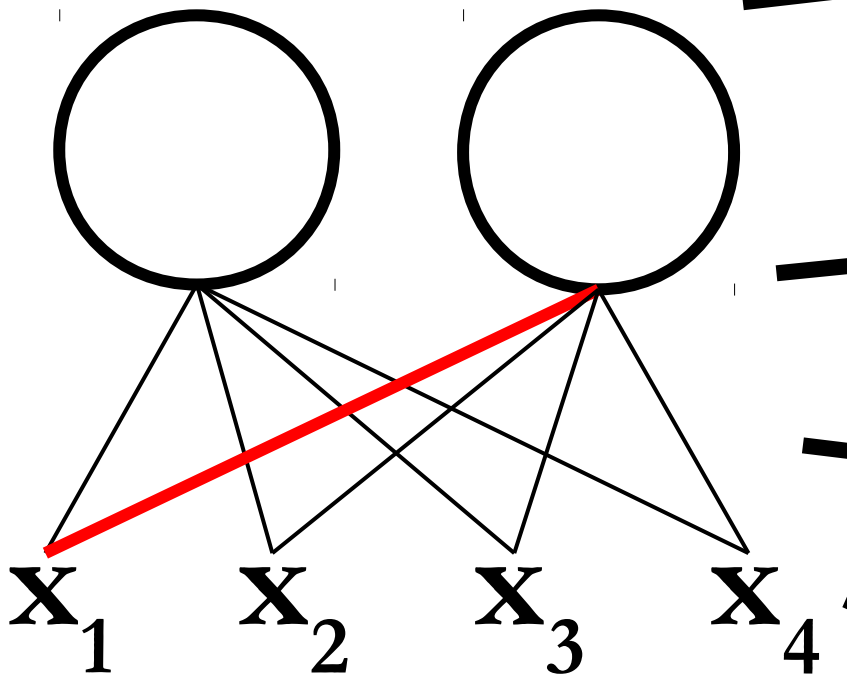
```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

to

from

```
[[ 0.9  -0.5]
 [ 0.8  -0.5]
 [-1.0   1.5]
 [-1.0   1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```




```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

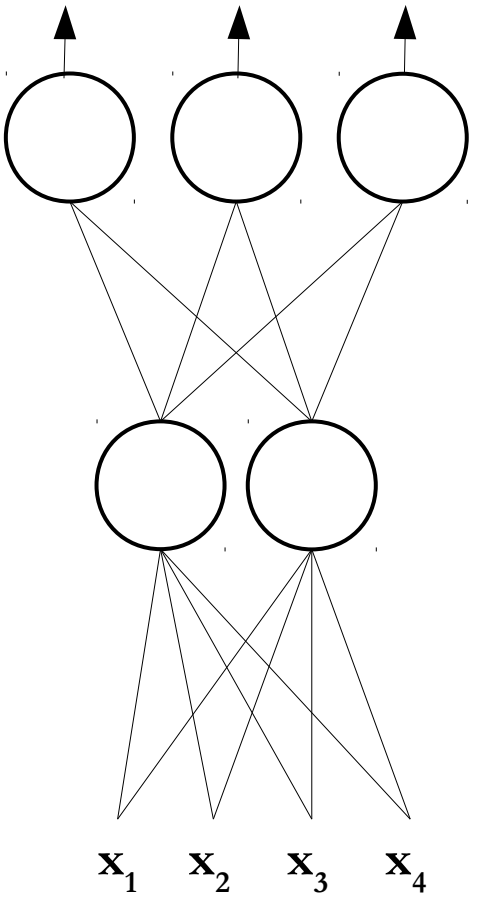
to

from

```
[[ 0.9  -0.5]
 [ 0.8  -0.5]
 [-1.0   1.5]
 [-1.0   1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```

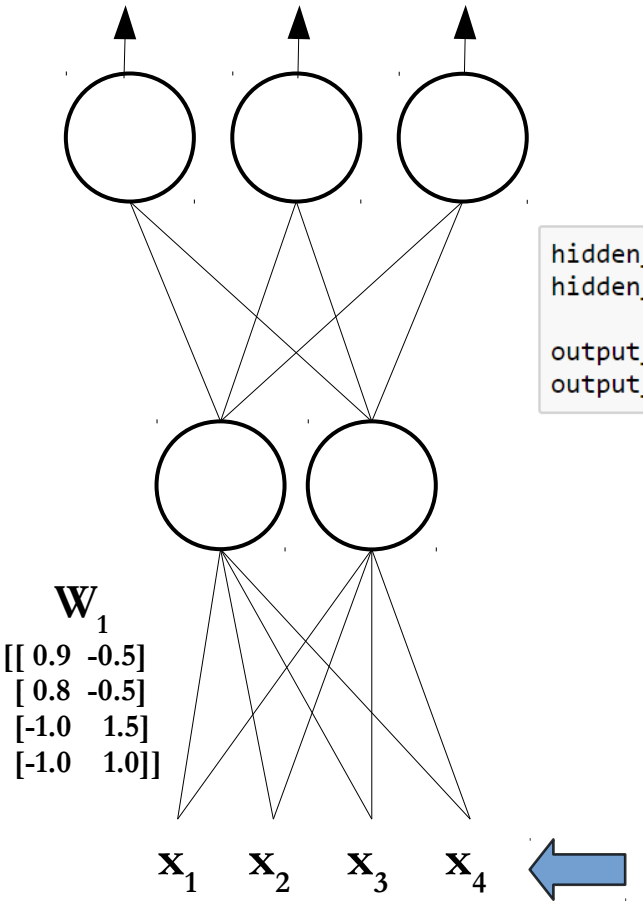
Feedforward




```
hidden_layer_inputs = np.dot(inputs, weights_1)
hidden_layer_outputs = step_function(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = step_function(output_layer_inputs)
```

Feedforward



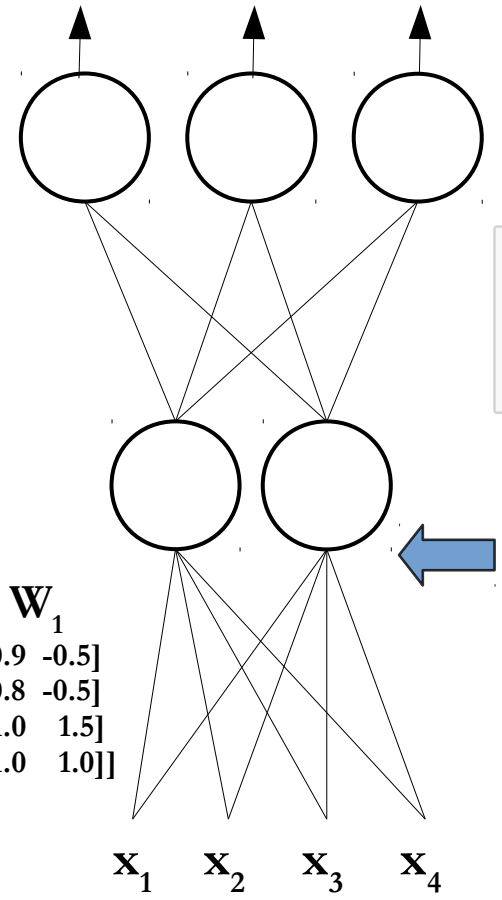

```
hidden_layer_inputs = np.dot(inputs, weights_1)
hidden_layer_outputs = step_function(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = step_function(output_layer_inputs)
```

X

```
[[ 4.9  3.0  1.4  0.2]
 [ 6.4  3.2  4.5  1.5]
 [ 5.8  2.7  5.1  1.9]]
```

Feedforward




```
hidden_layer_inputs = np.dot(inputs, weights_1)
hidden_layer_outputs = step_function(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = step_function(output_layer_inputs)
```

H_{in}

```
[[ 5.21 -1.65]
 [ 2.32  3.45]
 [ 0.38  5.30]]
```

X

```
[[ 4.9  3.0  1.4  0.2]
 [ 6.4  3.2  4.5  1.5]
 [ 5.8  2.7  5.1  1.9]]
```

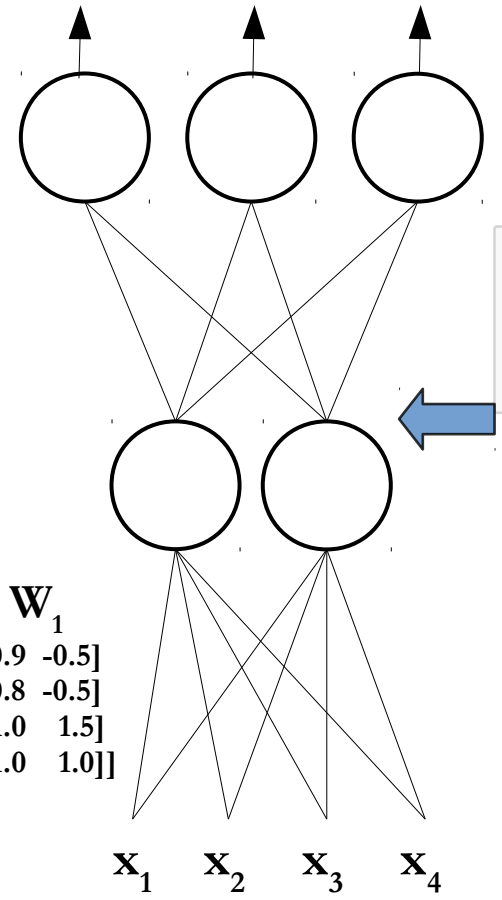
Feedforward

$$H_{in}$$

[[5.21 -1.65]
 [2.32 3.45]
 [0.38 5.30]]

$$H_{out}$$

[[1 0]
 [1 1]
 [0 1]]




```
hidden_layer_inputs = np.dot(inputs, weights_1)
hidden_layer_outputs = step_function(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = step_function(output_layer_inputs)
```

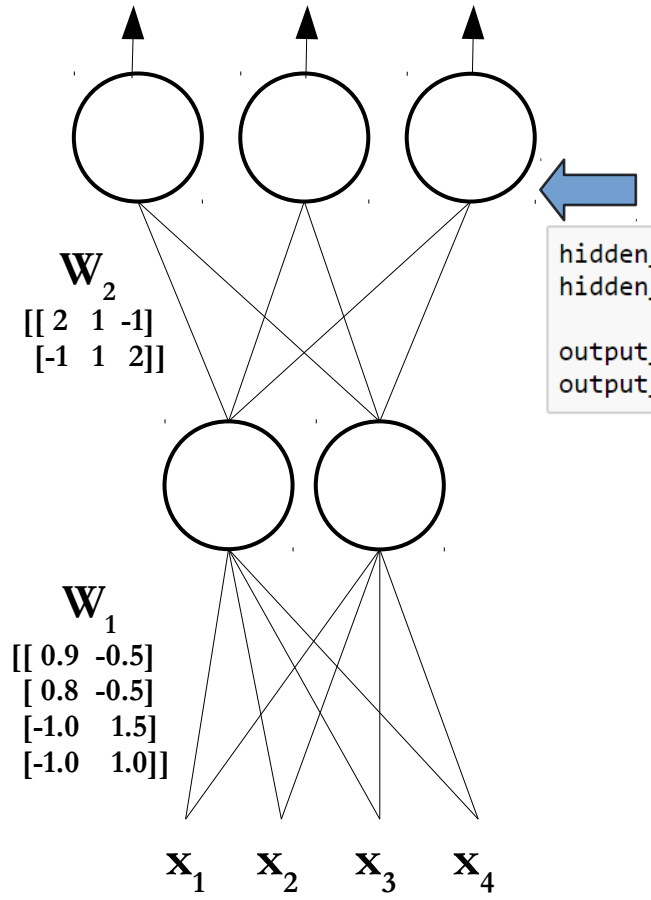
$$X$$

[[4.9 3.0 1.4 0.2]
 [6.4 3.2 4.5 1.5]
 [5.8 2.7 5.1 1.9]]

Feedforward

O_{in}

```
[[ 2  1 -1]
 [ 1  2  1]
 [-1  1  2]]
```




```
hidden_layer_inputs = np.dot(inputs, weights_1)
hidden_layer_outputs = step_function(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = step_function(output_layer_inputs)
```

H_{in}

```
[[ 5.21 -1.65]
 [ 2.32  3.45]
 [ 0.38  5.30]]
```

H_{out}

```
[[1 0]
 [1 1]
 [0 1]]
```

X

```
[[ 4.9  3.0  1.4  0.2]
 [ 6.4  3.2  4.5  1.5]
 [ 5.8  2.7  5.1  1.9]]
```

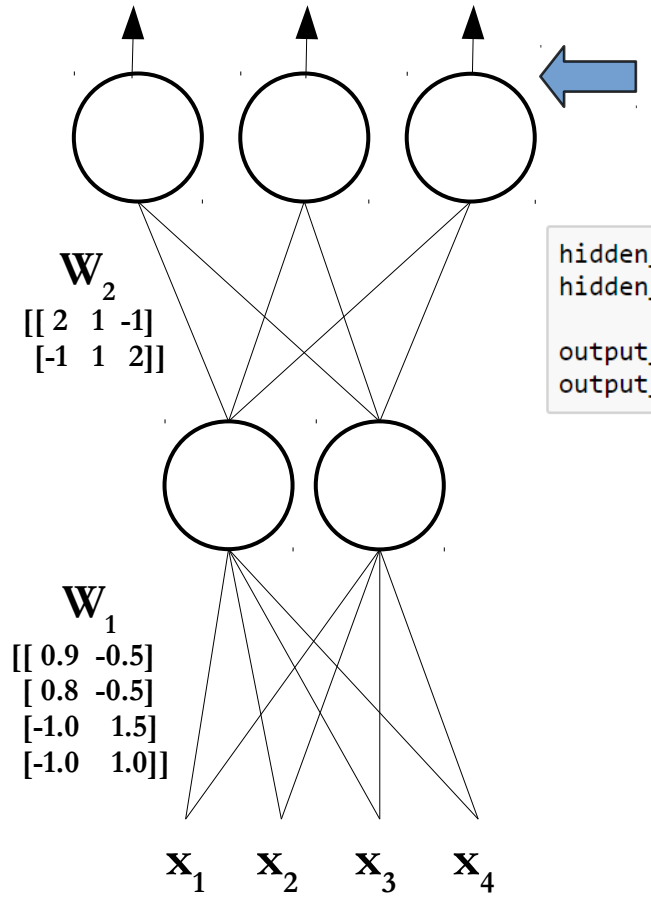
Feedforward

O_{in}

$$\begin{bmatrix} 2 & 1 & -1 \\ 1 & 2 & 1 \\ -1 & 1 & 2 \end{bmatrix}$$

O_{out}

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$




```
hidden_layer_inputs = np.dot(inputs, weights_1)
hidden_layer_outputs = step_function(hidden_layer_inputs)

output_layer_inputs = np.dot(hidden_layer_outputs, weights_2)
output_layer_outputs = step_function(output_layer_inputs)
```

H_{in}

$$\begin{bmatrix} 5.21 & -1.65 \\ 2.32 & 3.45 \\ 0.38 & 5.30 \end{bmatrix}$$

H_{out}

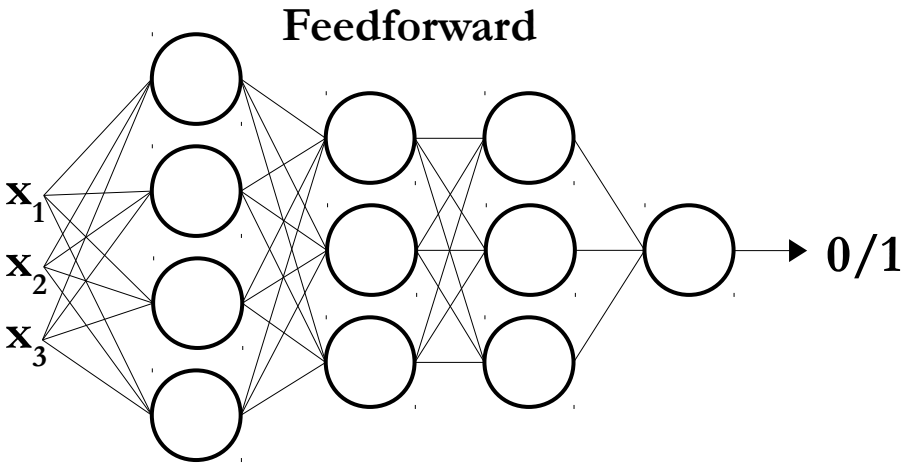
$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

X

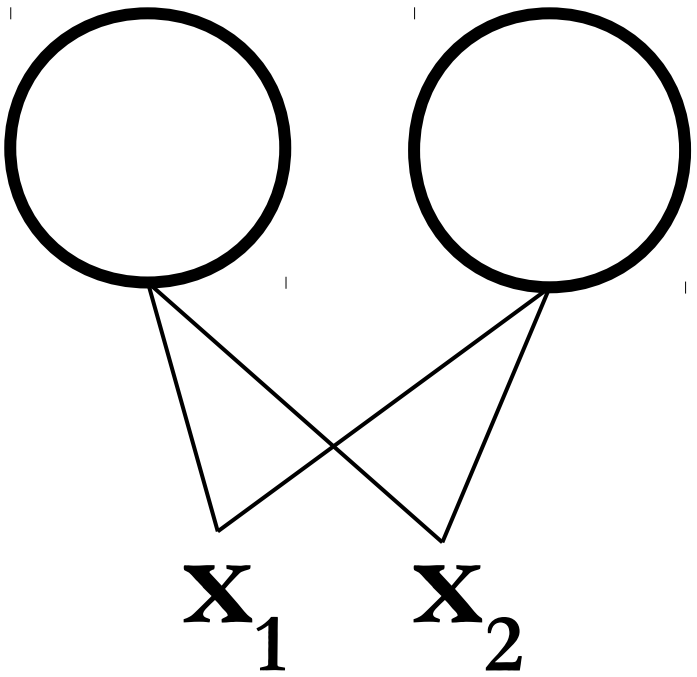
$$\begin{bmatrix} 4.9 & 3.0 & 1.4 & 0.2 \\ 6.4 & 3.2 & 4.5 & 1.5 \\ 5.8 & 2.7 & 5.1 & 1.9 \end{bmatrix}$$

Deep Learning

How does the algorithm make a decision?



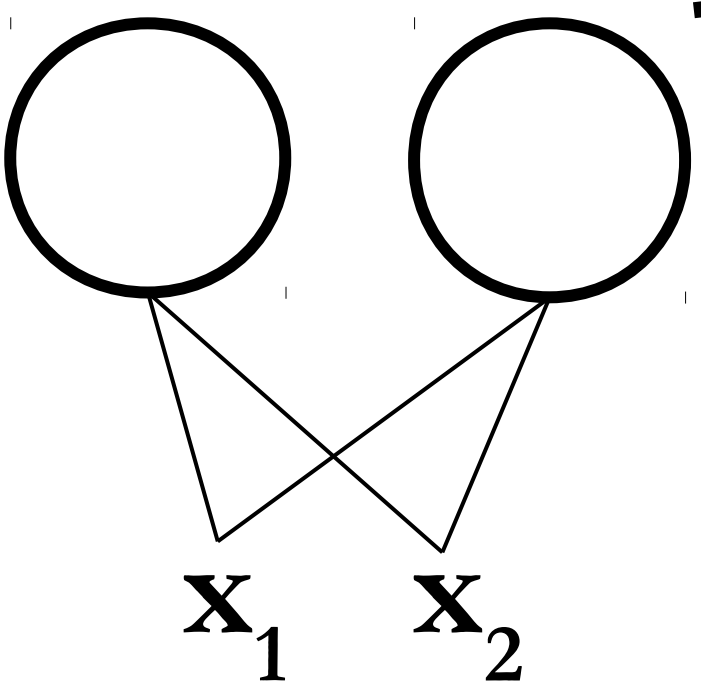
How do you determine the right parameters for the algorithm?



to

from [[0.9 -0.5]
[0.8 -0.5]]

[[4.9 3.0]
[6.4 3.2]]

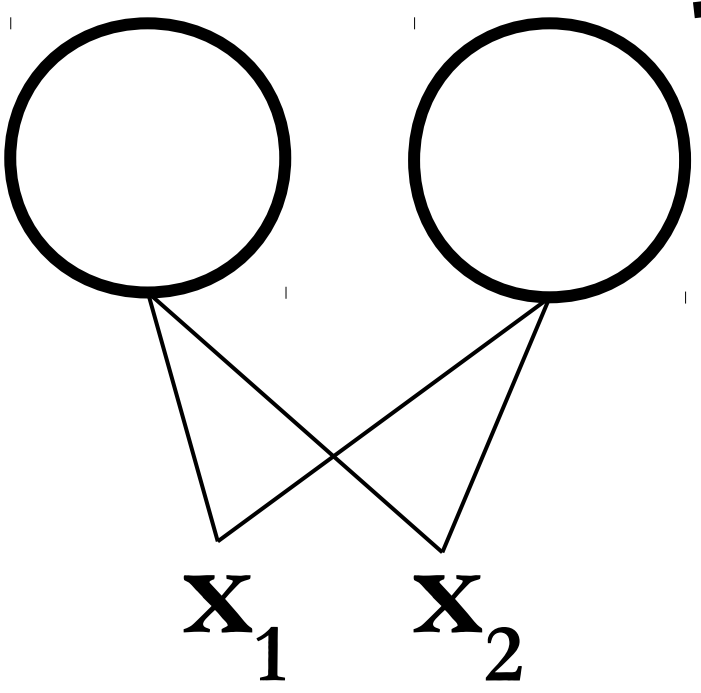


[[1 0]
[1 0]]

[[6.81 -3.95]
[8.32 -4.80]]

to
from [[0.9 -0.5]
[0.8 -0.5]]

[[4.9 3.0]
[6.4 3.2]]

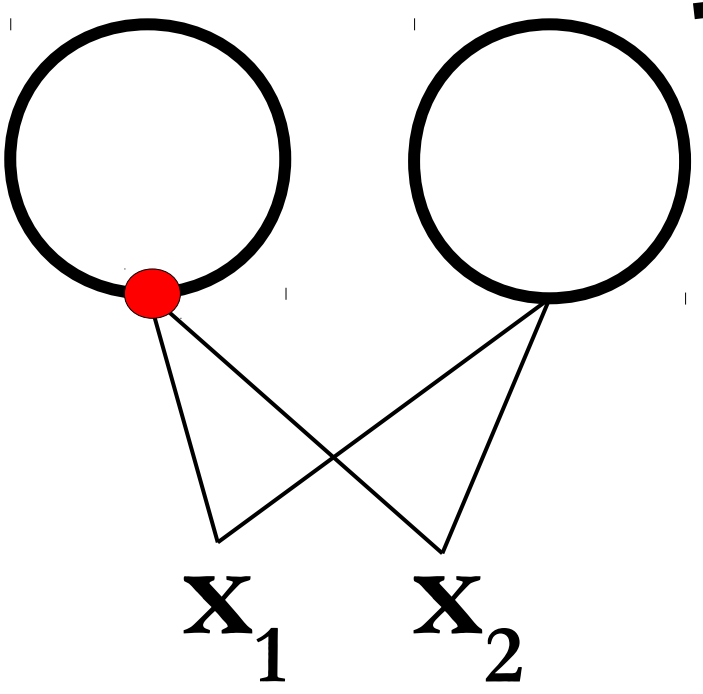


[[0 0]
[0 0]]

[[1.21 1.10]
[0.72 0.80]]

to
from [[0.9 -0.5]
[0.8 -0.5]]

[[4.9 3.0]
[6.4 3.2]]

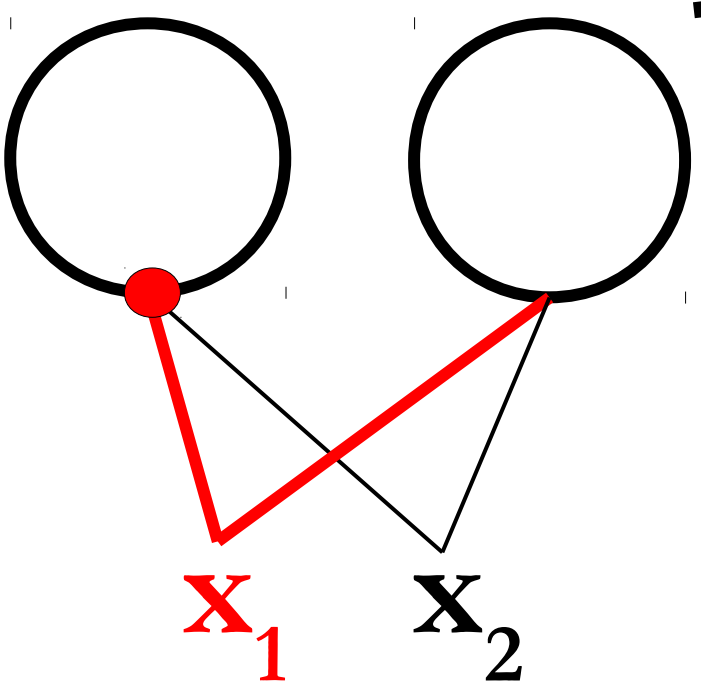


[[0 0]
[0 0]]

[[1.21 1.10]
[0.72 0.80]]

to
from [[0.9 -0.5]
[0.8 -0.5]]

[[4.9 3.0]
[6.4 3.2]]

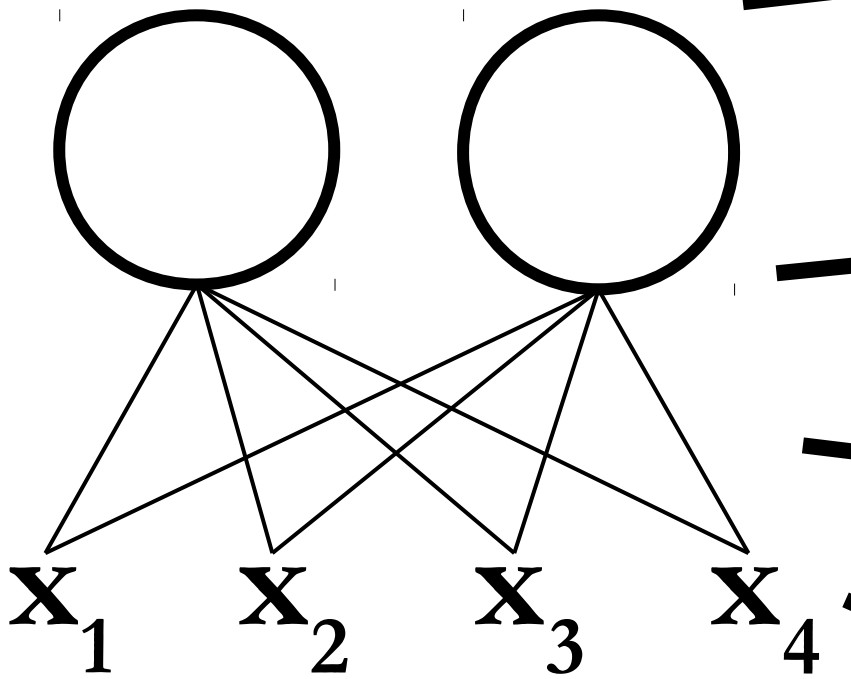


[[0 0]
[0 0]]

[[1.21 1.10]
[0.72 0.80]]

from [[0.9 -0.5]
[0.8 -0.5]]
to

[[4.9 3.0]
[6.4 3.2]]




```
[[1  0]
 [1  1]
 [0  1]]
```

```
[[5.21  -1.65]
 [2.32   3.45]
 [0.38   5.30]]
```

to

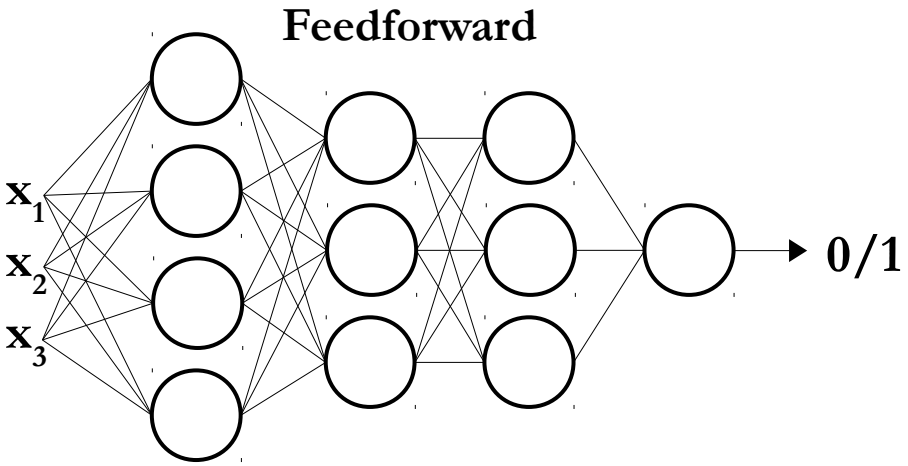
from

```
[[ 0.9  -0.5]
 [ 0.8  -0.5]
 [-1.0   1.5]
 [-1.0   1.0]]
```

```
[[4.9   3.0   1.4   0.2]
 [6.4   3.2   4.5   1.5]
 [5.8   2.7   5.1   1.9]]
```

Deep Learning

How does the algorithm make a decision?



How do you determine the right parameters for the algorithm?