

المتحكمات العصبونية و العائمة

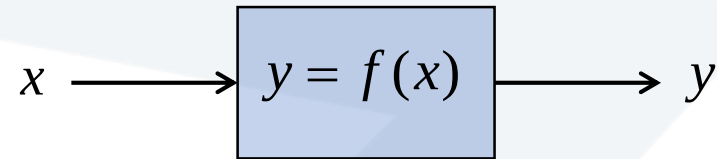
محاضرة 3

## أمثلة عن تطبيقات الشبكات العصبية الاصطناعية

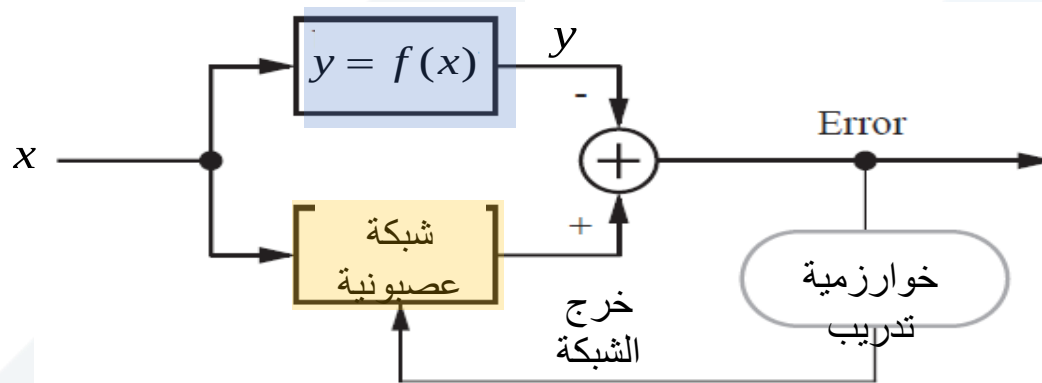
د. فادي متوج

# Function Approximation

- لدينا تابع غير معروف نريد مقاربته بواسطة شبكة عصبونية.



- يتم ذلك من خلال ضبط بارامترات الشبكة (الأوزان) بحيث تنتج نفس استجابة التابع الغير معروف إذا تم تطبيق نفس الدخل لكلا النظامين.



• تم إثبات أن شبكة عصبونية ذات تغذية أمامية مؤلفة من طبقة خفية واحدة، بتوابع تفعيل **sigmoid** في الطبقة الخفية، و توابع تفعيل **خطية** في طبقة الخرج هي قادرة على مقارنة أي تابع مستمر بشرط أن تحتوي على عدد كافي من العصبونات في الطبقة الخفية

المراجع: *Cybenko (1989) and Hornik et al. (1989)*

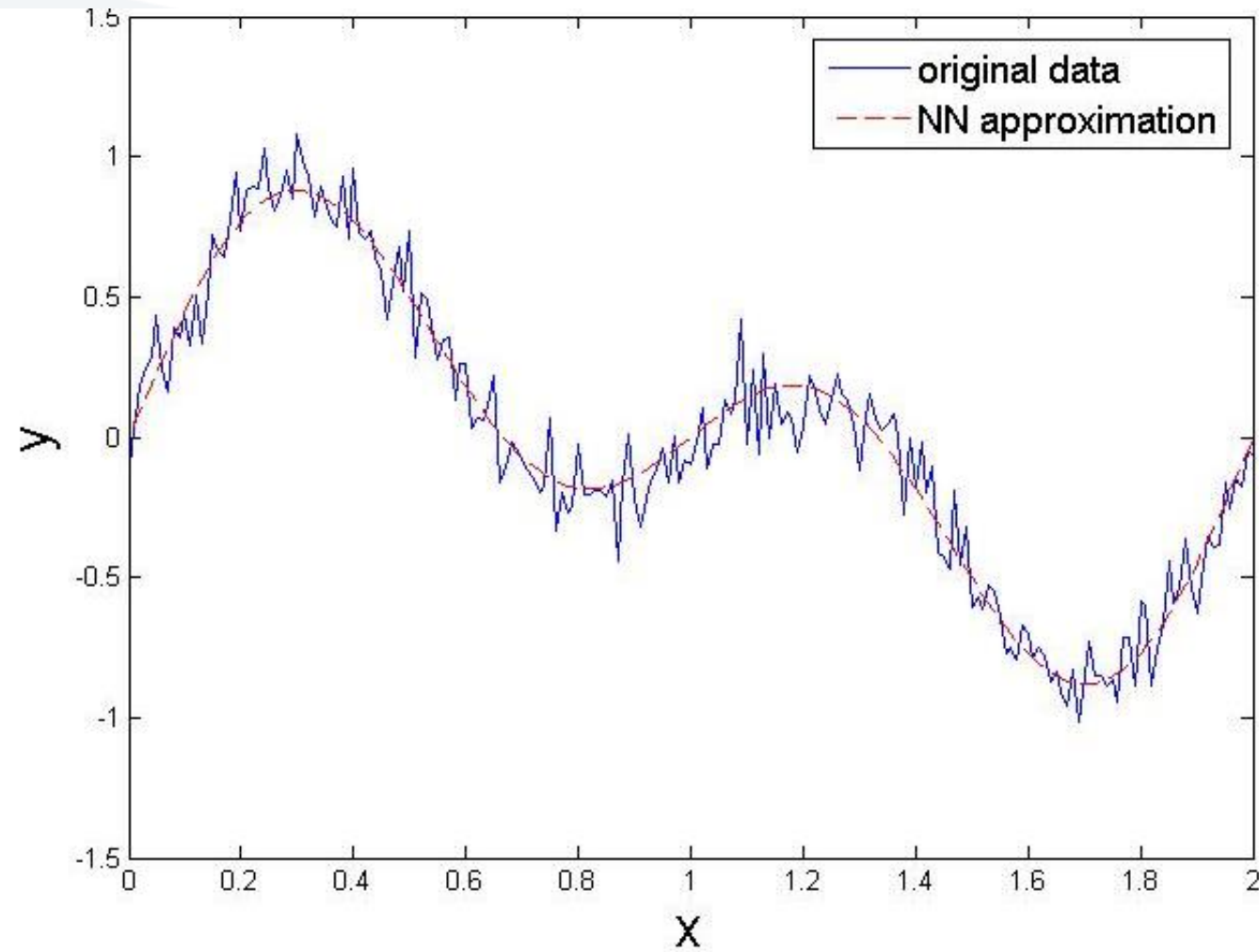
- التابع المطلوب مقاربته هو :

$$y = 0.5\sin(\pi x) + 0.5\sin(2\pi x), \quad x \in [0, 2]$$

- يتم توليد المعطيات في نقاط متساوية البعد عن بعضها
- يتم إضافة بعض الضجيج noise إلى المعطيات

```
x = 0:0.01:2.0;  
y = 0.5*sin(pi*x)+0.5*sin(2*pi*x)+0.1*randn(size(x));
```

- يحتوي كل من شعاع الدخل x و شعاع الخرج y على 201 نقطة



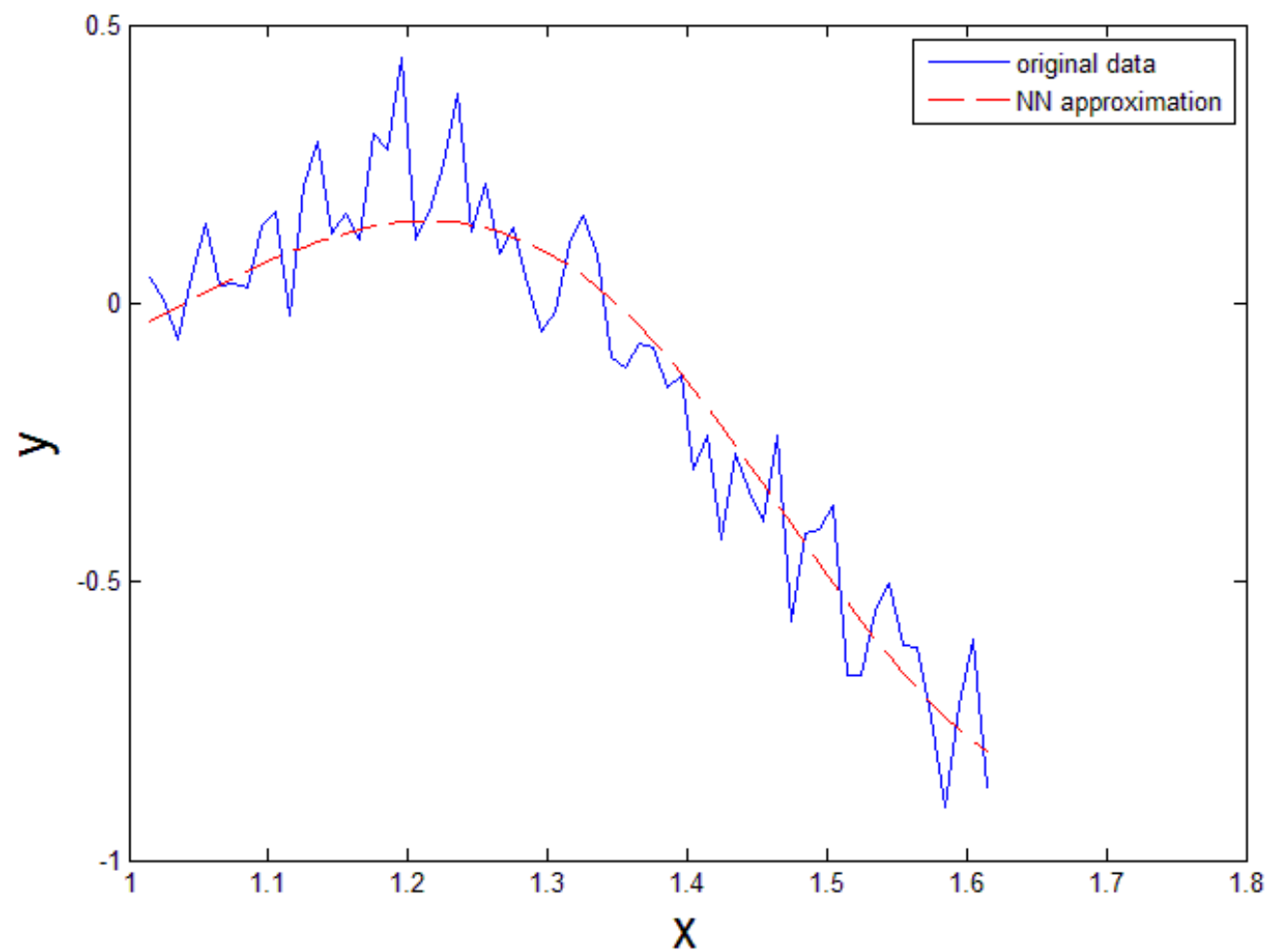
## • إنشاء الشبكة وتدريبها

```
net=newff(minmax(x),[4 1],{'tansig','purelin'},'trainlm');  
net.trainParam.epochs = 150;  
net.trainParam.show = 10;  
net = init(net);  
[net, tr] = train(net,x,y);
```

## • اختبار الشبكة :

لاختبار الشبكة يتم توليد نقاط جديدة ضمن المجال الذي تم اختياره لتدريب الشبكة

```
x1 = 1.015:0.01:1.615;  
y1 = 0.5*sin(pi*x1)+0.5*sin(2*pi*x1)+0.1*randn(size(x1));  
out1 = sim(net,x1);  
plot(x1, y1,'b',x1,out1,'r');
```

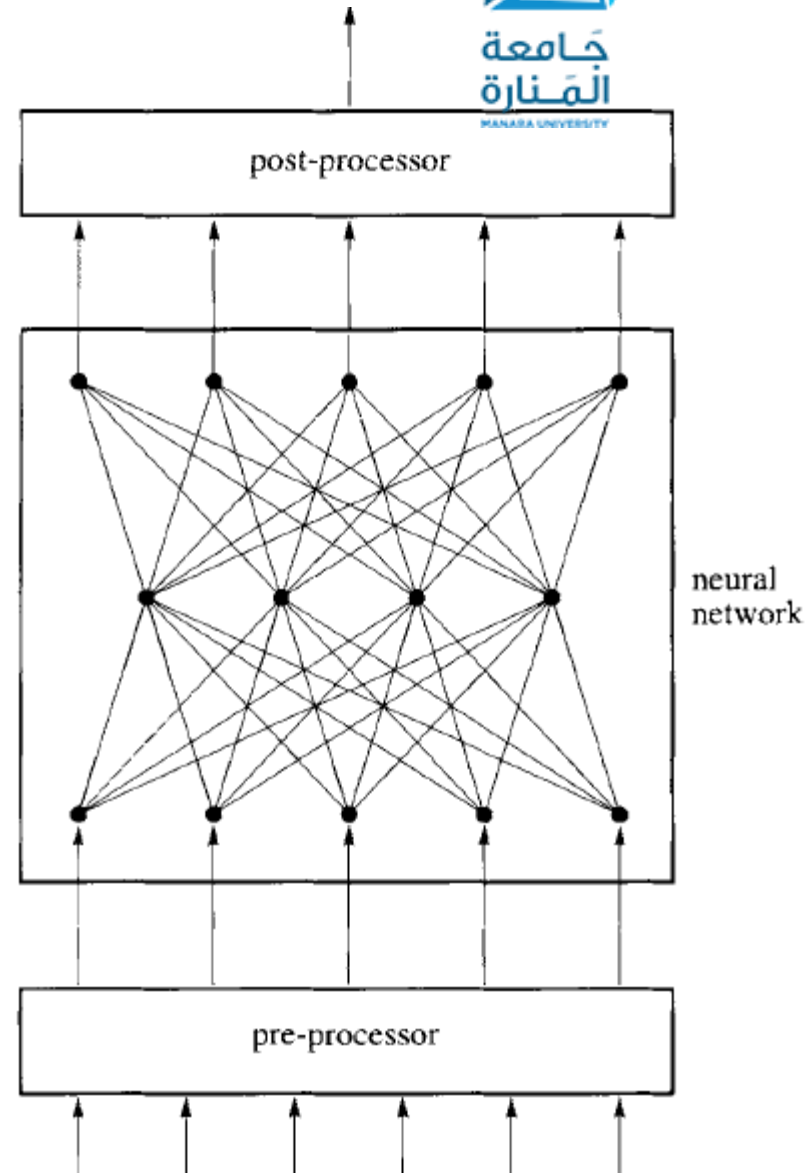




# Optical Character Recognition (OCR)

## قبل الشروع بعملية التعرف على المحارف يجب القيام ببعض الخطوات **لمعالجة الصورة**

- تحويل الصورة إلى صورة ذات تدرج رمادي
- تحويل الصورة ذات التدرج الرمادي إلى صورة ثنائية حيث يتم تقسيم الصورة إلى خلايا (pixels) ثم تمثل كل خلية بالرقم الثنائي **1** إذا كانت تحتوي على نص بينما تمثل بالرقم **0** إذا كانت بيضاء



- استخدام الشبكة العصبية يتم من خلال تدريب الشبكة على أمثلة من مجموعة أحرف خاصة، مثل خط طباعة font خاص.
- خلال عملية التدريب، جميع الحروف في ذلك الخط ستعرض على الشبكة، وعندما يصل الخطأ إلى قيمة منخفضة بشكل مناسب تكون الشبكة قد تدربت.
- عرض أي حرف من ذلك الخط مرة أخرى على الشبكة يجب أن ينتج الاستجابة الصحيحة التي من شأنها تحديد الحرف.

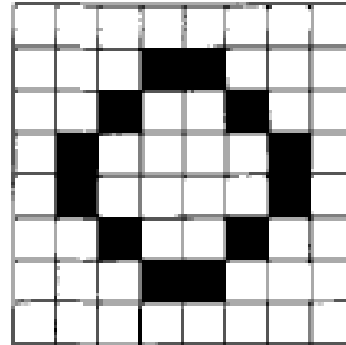
## Example

### التعرف على الحارف من 0 إلى 9

- يمكن تمثيل كل محرف برمز مؤلف من  $8 \times 8$  تسلسل للأصفر و الواحدات يمثل خلايا الصورة البيضاء والسوداء و الذي يشكل دخل الشبكة العصبونية.

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 0 1 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0
```



- لدينا نمط الخرج المطلوب المقابل لـ 0 هو السلسلة التالية من عشرة أرقام ثنائية (بت):

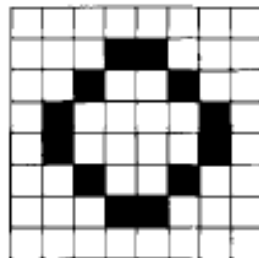
*Desired output pattern*

```
1 0 0 0 0 0 0 0 0
```

**Numeral: 0**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 0 1 0 0 1 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0
```



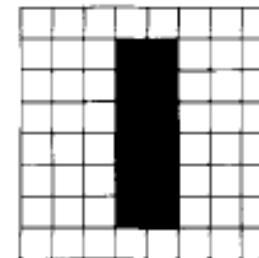
*Desired output pattern*

```
1 0 0 0 0 0 0 0 0 0
```

**Numeral: 1**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0
```



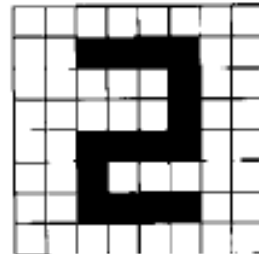
*Desired output pattern*

```
0 1 0 0 0 0 0 0 0 0
```

**Numeral: 2**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



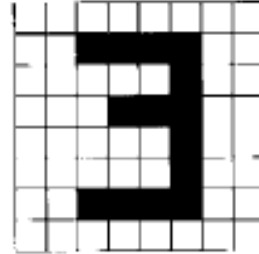
*Desired output pattern*

```
0 0 1 0 0 0 0 0 0 0
```

**Numeral: 3**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



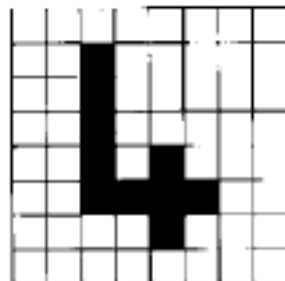
*Desired output pattern*

```
0 0 0 1 0 0 0 0 0 0
```

**Numerals: 4**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 1 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
```



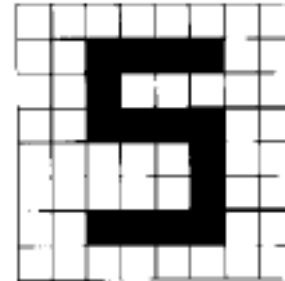
*Desired output pattern*

```
0 0 0 0 1 0 0 0 0 0
```

**Numerals: 5**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



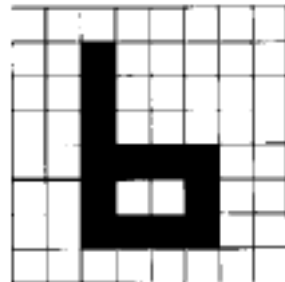
*Desired output pattern*

```
0 0 0 0 0 1 0 0 0 0
```

**Numerals: 6**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



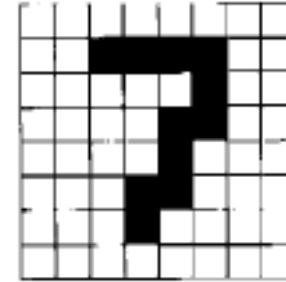
*Desired output pattern*

```
0 0 0 0 0 0 1 0 0 0
```

**Numerals: 7**

*Input pattern*

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 1 0 0 0
0 0 0 1 1 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
```



*Desired output pattern*

```
0 0 0 0 0 0 0 1 0 0
```

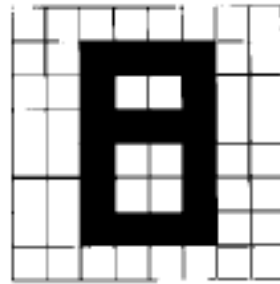


جامعة  
المنارة  
MANARA UNIVERSITY

Numeral: 8

Input pattern

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



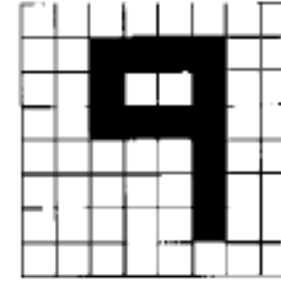
Desired output pattern

```
0 0 0 0 0 0 0 0 1 0
```

Numeral: 9

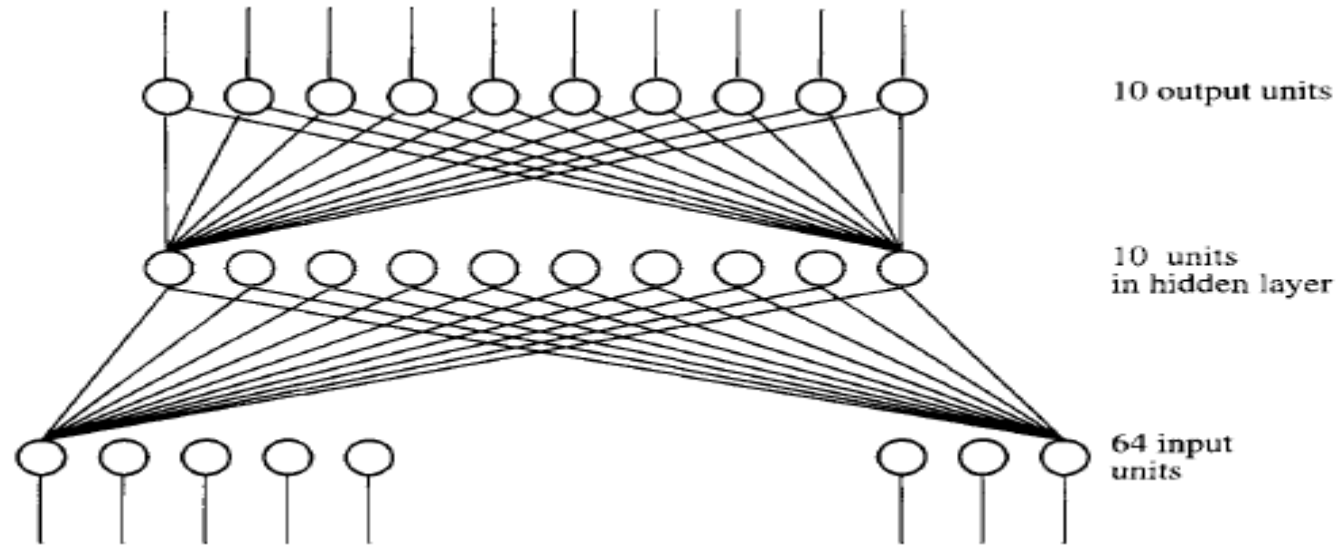
Input pattern

```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0
```



Desired output pattern

```
0 0 0 0 0 0 0 0 0 1
```



بنية الشبكة العصبونية المستخدمة في مثالنا



• بعد عرض بيانات التدريب **100** مرة على الشبكة

| Numeral | Outputs     |             |             |             |             |             |             |             |             |             |
|---------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0       | <b>0.46</b> | 0.02        | 0.17        | 0.13        | 0.08        | 0.09        | 0.02        | 0.13        | 0.19        | 0.15        |
| 1       | 0.06        | <b>0.72</b> | 0.07        | 0.10        | 0.12        | 0.10        | 0.00        | 0.31        | 0.06        | 0.18        |
| 2       | 0.12        | 0.05        | <b>0.22</b> | 0.15        | 0.04        | 0.10        | 0.10        | 0.05        | 0.14        | 0.07        |
| 3       | 0.09        | 0.06        | 0.12        | <b>0.17</b> | 0.02        | 0.15        | 0.02        | 0.09        | 0.13        | 0.14        |
| 4       | 0.11        | 0.11        | 0.09        | 0.06        | <b>0.67</b> | 0.07        | 0.14        | 0.18        | 0.09        | 0.12        |
| 5       | 0.08        | 0.04        | 0.10        | 0.14        | 0.03        | <b>0.14</b> | 0.02        | 0.10        | 0.13        | 0.16        |
| 6       | 0.07        | 0.03        | 0.17        | 0.11        | 0.23        | 0.12        | <b>0.82</b> | 0.05        | 0.14        | 0.07        |
| 7       | 0.13        | 0.17        | 0.05        | 0.08        | 0.11        | 0.10        | 0.00        | <b>0.33</b> | 0.09        | 0.27        |
| 8       | 0.15        | 0.02        | 0.13        | 0.12        | 0.03        | 0.12        | 0.04        | 0.07        | <b>0.16</b> | 0.15        |
| 9       | 0.11        | 0.04        | 0.06        | 0.09        | 0.04        | 0.12        | 0.01        | 0.19        | 0.12        | <b>0.26</b> |

• بعد عرض بيانات التدريب **200** مرة على الشبكة

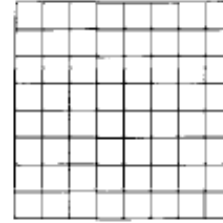
| Numeral | Outputs     |             |             |             |             |             |             |             |             |             |
|---------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0       | <b>0.85</b> | 0.00        | 0.06        | 0.05        | 0.07        | 0.00        | 0.00        | 0.11        | 0.13        | 0.08        |
| 1       | 0.01        | <b>0.84</b> | 0.05        | 0.07        | 0.03        | 0.04        | 0.00        | 0.11        | 0.00        | 0.04        |
| 2       | 0.07        | 0.08        | <b>0.76</b> | 0.18        | 0.01        | 0.08        | 0.03        | 0.01        | 0.18        | 0.01        |
| 3       | 0.05        | 0.08        | 0.13        | <b>0.74</b> | 0.00        | 0.25        | 0.00        | 0.06        | 0.06        | 0.04        |
| 4       | 0.06        | 0.04        | 0.02        | 0.00        | <b>0.84</b> | 0.01        | 0.08        | 0.04        | 0.06        | 0.04        |
| 5       | 0.00        | 0.02        | 0.03        | 0.14        | 0.02        | <b>0.60</b> | 0.02        | 0.01        | 0.16        | 0.18        |
| 6       | 0.01        | 0.02        | 0.09        | 0.02        | 0.11        | 0.16        | <b>0.90</b> | 0.00        | 0.18        | 0.05        |
| 7       | 0.09        | 0.10        | 0.01        | 0.09        | 0.06        | 0.03        | 0.00        | <b>0.82</b> | 0.02        | 0.17        |
| 8       | 0.07        | 0.00        | 0.10        | 0.02        | 0.04        | 0.11        | 0.05        | 0.01        | <b>0.68</b> | 0.20        |
| 9       | 0.01        | 0.01        | 0.00        | 0.01        | 0.06        | 0.16        | 0.00        | 0.10        | 0.14        | <b>0.76</b> |

| Numeral | Outputs     |             |             |             |             |             |             |             |             |             |
|---------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0       | <b>0.94</b> | 0.00        | 0.01        | 0.03        | 0.04        | 0.00        | 0.00        | 0.03        | 0.04        | 0.03        |
| 1       | 0.00        | <b>0.95</b> | 0.03        | 0.01        | 0.01        | 0.01        | 0.00        | 0.04        | 0.00        | 0.01        |
| 2       | 0.01        | 0.03        | <b>0.94</b> | 0.04        | 0.00        | 0.01        | 0.01        | 0.01        | 0.04        | 0.00        |
| 3       | 0.03        | 0.01        | 0.03        | <b>0.93</b> | 0.00        | 0.06        | 0.00        | 0.02        | 0.01        | 0.01        |
| 4       | 0.03        | 0.03        | 0.01        | 0.00        | <b>0.94</b> | 0.00        | 0.03        | 0.01        | 0.01        | 0.00        |
| 5       | 0.00        | 0.01        | 0.01        | 0.04        | 0.00        | <b>0.91</b> | 0.02        | 0.00        | 0.04        | 0.04        |
| 6       | 0.00        | 0.01        | 0.03        | 0.00        | 0.05        | 0.06        | <b>0.95</b> | 0.00        | 0.04        | 0.01        |
| 7       | 0.03        | 0.03        | 0.01        | 0.03        | 0.01        | 0.00        | 0.00        | <b>0.94</b> | 0.00        | 0.03        |
| 8       | 0.03        | 0.00        | 0.03        | 0.00        | 0.01        | 0.02        | 0.02        | 0.00        | <b>0.92</b> | 0.03        |
| 9       | 0.01        | 0.00        | 0.00        | 0.00        | 0.02        | 0.04        | 0.00        | 0.03        | 0.03        | <b>0.94</b> |

All 0s

Input pattern

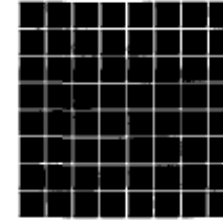
```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```



All 1s

Input pattern

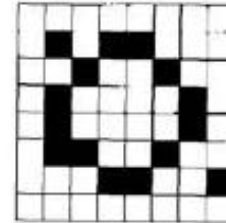
```
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
```



0 ?

Input pattern

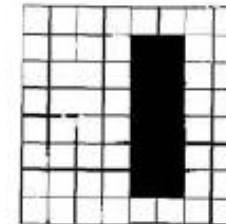
```
0 0 0 0 0 0 0 0
0 1 0 1 1 0 0 0
0 0 1 0 0 1 0 0
0 1 0 0 0 0 1 0
0 1 0 0 0 0 1 0
0 1 1 0 0 1 0 0
0 0 0 1 1 0 0 1
0 0 0 0 0 0 0 0
```



1 ?

Input pattern

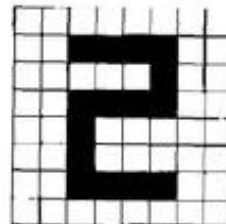
```
0 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0
```



2 ?

Input pattern

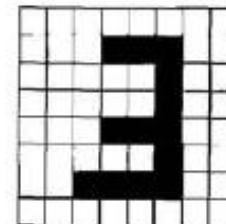
```
0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



3 ?

Input pattern

```
0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0
```



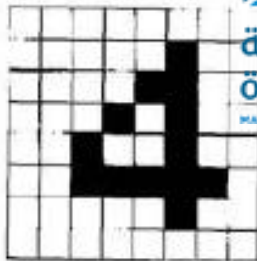
4?

Input pattern

```

0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 1 0 0
0 0 0 1 0 1 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 1 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0

```



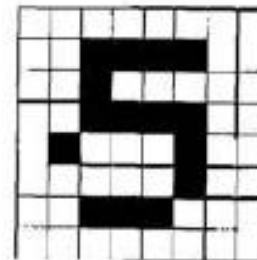
5?

Input pattern

```

0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 1 0 0
0 1 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 1 1 1 0 0 0
0 0 0 0 0 0 0 0

```



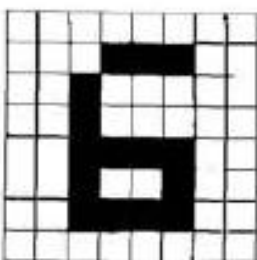
6?

Input pattern

```

0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0

```



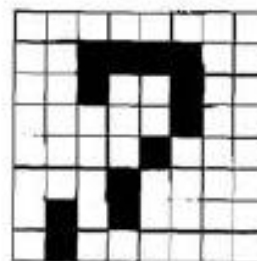
7?

Input pattern

```

0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0
0 0 0 1 0 0 0 0
0 1 0 1 0 0 0 0
0 1 0 0 0 0 0 0

```



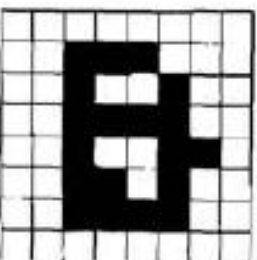
8?

Input pattern

```

0 0 0 0 0 0 0 0
0 0 1 1 1 0 0 0
0 0 1 0 0 1 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 1 0
0 0 1 1 0 1 0 0
0 0 1 1 1 1 0 0
0 0 0 0 0 0 0 0

```



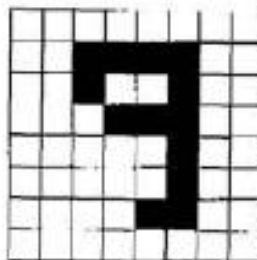
9?

Input pattern

```

0 0 0 0 0 0 0 0
0 0 1 1 1 1 0 0
0 0 1 0 0 1 0 0
0 0 0 1 1 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0

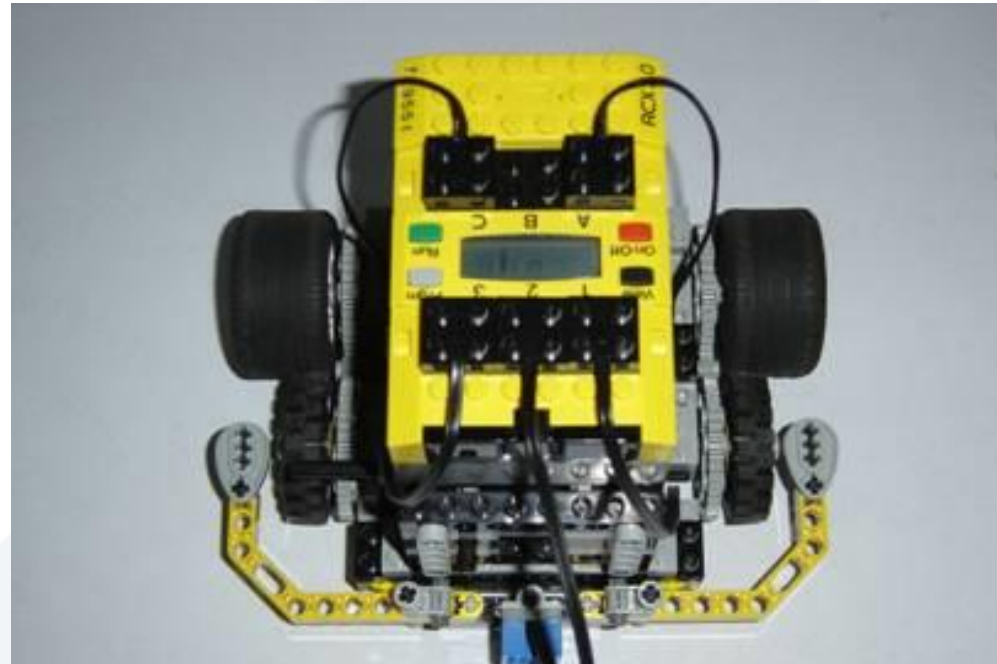
```





| Input pattern | Outputs     |             |             |             |             |             |             |             |             |             |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| All 0s        | 0.04        | 0.12        | 0.02        | 0.03        | 0.08        | 0.01        | 0.02        | 0.08        | 0.01        | 0.08        |
| All 1s        | 0.00        | 0.01        | 0.00        | 0.00        | 0.00        | 0.00        | 0.00        | 0.03        | 0.00        | 0.22        |
| 0             | <b>0.93</b> | 0.00        | 0.01        | 0.02        | 0.04        | 0.00        | 0.00        | 0.03        | 0.03        | 0.02        |
| 1             | 0.04        | <b>0.11</b> | 0.01        | 0.32        | 0.01        | 0.00        | 0.00        | 0.18        | 0.00        | 0.01        |
| 2             | 0.05        | 0.00        | <b>0.08</b> | 0.01        | 0.00        | 0.01        | 0.01        | 0.01        | <b>0.82</b> | 0.01        |
| 3             | 0.06        | 0.06        | 0.89        | <b>0.33</b> | 0.00        | 0.01        | 0.01        | 0.01        | 0.01        | 0.00        |
| 4             | 0.25        | 0.00        | 0.01        | 0.00        | <b>0.22</b> | 0.00        | 0.00        | 0.06        | 0.16        | 0.04        |
| 5             | 0.00        | 0.01        | 0.01        | 0.04        | 0.00        | <b>0.84</b> | 0.01        | 0.00        | 0.04        | 0.03        |
| 6             | 0.00        | 0.02        | 0.17        | 0.00        | 0.03        | 0.12        | <b>0.61</b> | 0.00        | 0.08        | 0.00        |
| 7             | 0.01        | 0.01        | 0.00        | 0.01        | 0.02        | 0.00        | 0.00        | <b>0.81</b> | 0.01        | 0.05        |
| 8             | 0.09        | 0.00        | 0.03        | 0.00        | 0.05        | 0.01        | 0.01        | 0.01        | <b>0.89</b> | 0.01        |
| 9             | 0.04        | 0.00        | 0.00        | 0.17        | 0.00        | 0.01        | 0.00        | 0.04        | 0.02        | <b>0.44</b> |

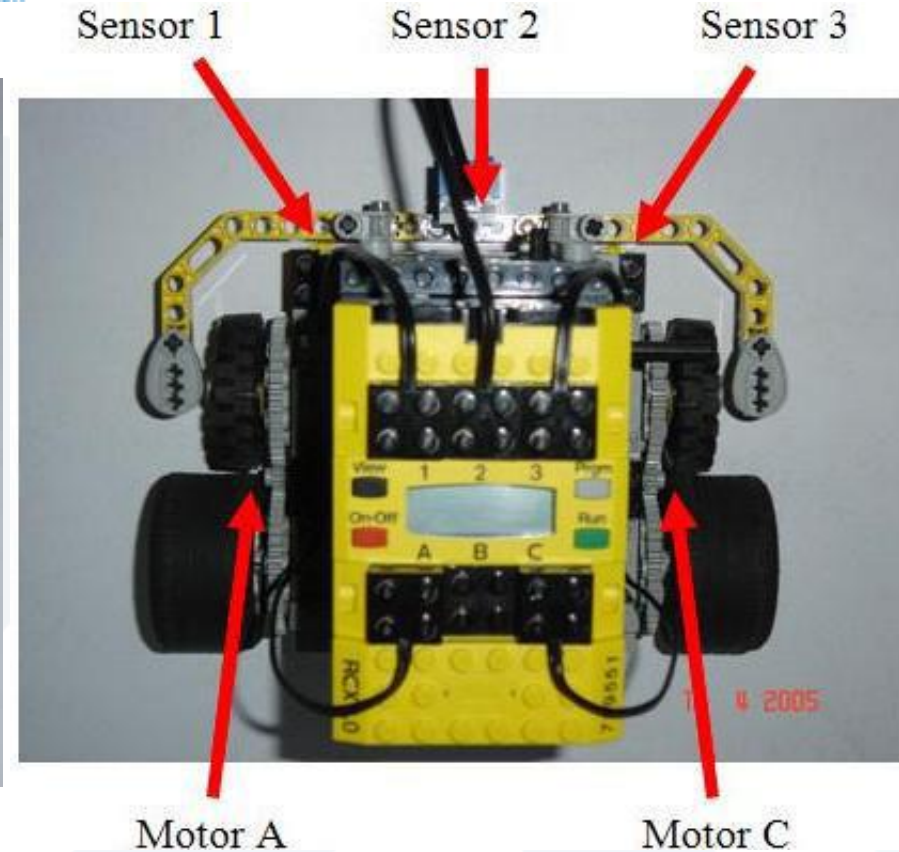
# A neural network for Lego robots



- This example shows how to develop a robot that can learn by using a neural network, and implementing it on a **Lego robot**.
- Using the neural network, the Lego robot vehicle can learn some basic rules for moving forward, backward, left, and right.



# Lego robots

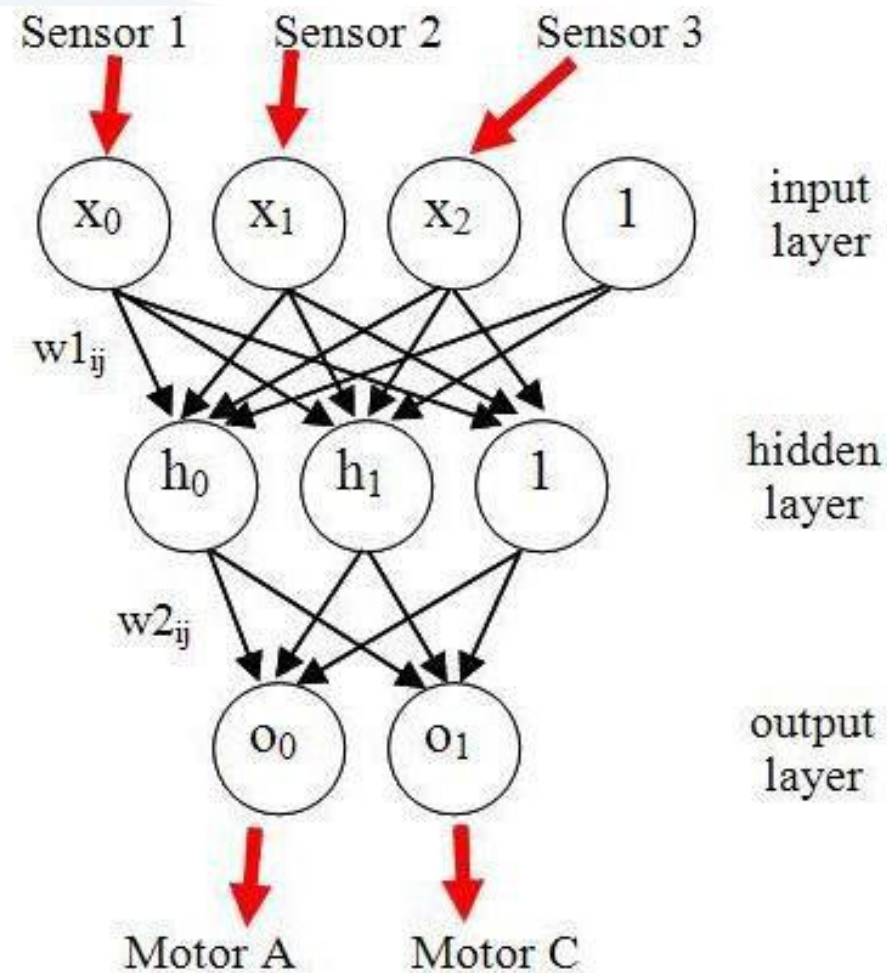


- The robot has **three inputs** (two touch sensors and one light sensor) and **two outputs** (the two motors)



جامعة  
المنارة  
MANARA UNIVERSITY

## بنية الشبكة العصبونية المستخدمة لتدريب الروبوت



- **Moving forward:** If **Sensor 1** is **off**, and **Sensor 2** is over a **white floor**, and **Sensor 3** is **off**, then Motor A and Motor C go forward (robot goes forward).
- **Moving right:** If Sensor 1 is on, then Motor A goes forward, and Motor C goes backward (robot turns right).
- **Moving left:** If Sensor 3 is on, then Motor A goes backward, and Motor C goes forward (robot turns left).
- **Moving backward:** If Sensor 2 is over a black floor, then Motor A and Motor C go backward (robot goes backward).

# Training Data

| Rules to learn |          |          |          |          |          |
|----------------|----------|----------|----------|----------|----------|
| Rule           | Sensor 1 | Sensor 2 | Sensor 3 | Motor A  | Motor C  |
| 1              | Off      | White    | Off      | Forward  | Forward  |
| 2              | On       | White    | Off      | Forward  | Backward |
| 3              | Off      | White    | On       | Backward | Forward  |
| 4              | Off      | Black    | Off      | Backward | Backward |

Physical inputs

Physical outputs

| Training examples |    |    |     |     |
|-------------------|----|----|-----|-----|
| S1                | S2 | S3 | M-A | M-C |
| 0                 | 0  | 0  | 1   | 1   |
| 1                 | 0  | 0  | 1   | 0   |
| 0                 | 0  | 1  | 0   | 1   |
| 0                 | 1  | 0  | 0   | 0   |

Input vectors

Output vectors

دخل الشبكة العصبونية

خرج الشبكة العصبونية

S1 = Sensor 1

M-A = Motor A

## Neural Network Test

- The input-output vector pairs are the examples we use to train neural network. So, based on its sensor states, our robot will learn to move forward, right, left, and backward.
- But what would happen **if both touch sensors were on?** The robot would not learn that case (rule or example), but the neural network would give it an emergent behavior.
- **What emergent behavior?** Will the robot go forward, backward, left, or right? We will get the answer by compiling the program, downloading it to the robot, and pressing the Run button to run the program and see the robot behavior.