

history of programming, part III

thursday 17 august 2000

lecture overview:

- database programming
- questions about the assignment
- connectionist programming
- Braitenberg vehicles

1

database programming

- reference:
Date, C.J. (1991) *An Introduction to Database Systems*, Sydney: Addison-Wesley.

- the basics of data manipulation:

- ◊ INSERT
- ◊ DELETE
- ◊ UPDATE
- ◊ SELECT

2

database programming basics: INSERT

```
INSERT  
INTO table [ ( field [, field ] ... ) ]  
VALUES ( literal [, literal ] ... );
```

- for example:

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

```
INSERT  
INTO S  
VALUES ( 'S4','Clark',20,'London' );
```

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London

3

database programming basics: DELETE

```
DELETE  
FROM table  
[ WHERE condition ];
```

- for example:

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London

```
DELETE  
FROM S  
WHERE SNAME = 'S4';
```

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

4

database programming basics: UPDATE

UPDATE table

SET field = scalar-expression

[, field = scalar-expression] ...

[WHERE condition];

- for example:

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

UPDATE S

SET STATUS = 40

WHERE SNAME = 'S3';

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	40	Paris

5

database programming basics: SELECT

SELECT field [, field] ...

FROM table

[WHERE condition];

- for example:

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London

SELECT SNAME

FROM S;

return value =

SNAME
Smith
Jones
Blake
Clark

6

database programming basics: JOIN

table S =

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris

table P =

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

SELECT SNAME, S.CITY, PNAME, COLOR
FROM S, P WHERE S.CITY = P.CITY;

return value =

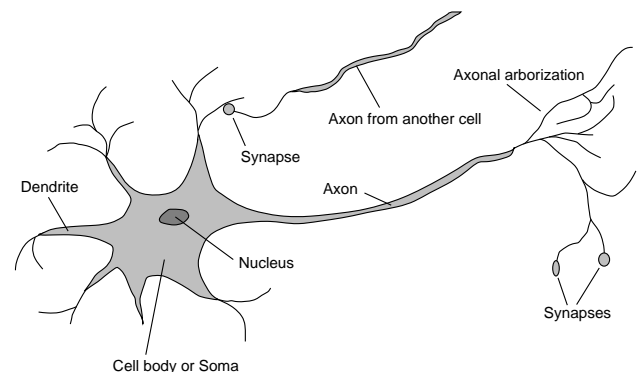
SNAME	S.CITY	PNAME	COLOR
Smith	London	Nut	Red
Smith	London	Screw	Red
Smith	London	Cog	Red
Jones	Paris	Bolt	Green
Jones	Paris	Cam	Blue
Blake	Paris	Bolt	Green
Blake	Paris	Cam	Blue

7

connectionist programming

- neural networks (nn)

- basis in biology



AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

8

nn: Biological neurons

The brain is made up of *neurons* (nerve cells) which have

- ◊ a cell body (soma)
- ◊ *dendrites* (inputs)
- ◊ an *axon* (outputs)
- ◊ *synapses* (connections between cells)

Synapses can be *excitatory* or *inhibitory* and may change over time

When the inputs reach some threshold an *action potential* (electrical pulse) is sent along the axon to the outputs

There are around 10^{11} neurons, 10^{14} synapses; a cycle time of 10^{-3} seconds

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

9

nn: Artificial neural networks

(Artificial) Neural networks are made up of nodes which have

- ◊ inputs edges, each with some *weight*
- ◊ outputs edges (with *weights*)
- ◊ an *activation level* (a function of the inputs)

Weights of edges can be positive or negative and may change over time (learning)

The *input function* is the weighted sum of the activation levels of inputs

The activation level is a non-linear *transfer function* g of this input:

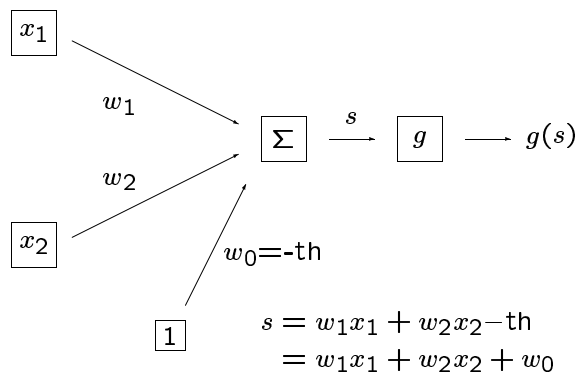
$$a_i = g(in_i) = g\left(\sum a_j w_{j,i}\right)$$

Some nodes are inputs (perception), some are outputs (action)

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

10

nn: Rosenblatt Perceptron



x_1, x_2 are inputs

w_1, w_2 are synaptic weights

th is a threshold

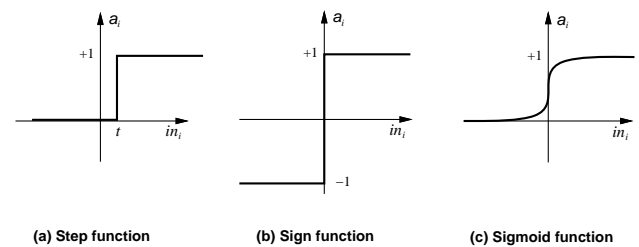
w_0 is a **bias** weight

g is transfer function

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

11

nn: Transfer functions



transfer function g may be step function

$$g(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ 0, & \text{if } s < 0 \end{cases}$$

or sign function

$$g(s) = \begin{cases} 1, & \text{if } s \geq 0 \\ -1, & \text{if } s < 0 \end{cases}$$

or sigmoid function

$$g(s) = \frac{1}{1 + e^{-s}}$$

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

12

nn: Linear Separability

Q: what kind of functions can a perceptron compute?

A: linearly separable functions

Examples include:

AND $w_1 = w_2 = 1.0, w_0 = -1.5$
 OR $w_1 = w_2 = 1.0, w_0 = -0.5$
 NOR $w_1 = w_2 = -1.0, w_0 = 0.5$

Q: How do we train it to learn a new function?

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Perceptron Learning Rule

Adjust the weights as each input is presented
 recall: $s = w_1x_1 + w_2x_2 + w_0$, $\eta > 0$ is called the **learning rate**

if $g(s) = 0$ but should be 1,

$$\begin{aligned} w_k &\leftarrow w_k + \eta x_k \\ w_0 &\leftarrow w_0 + \eta \end{aligned}$$

$$\text{so } s \leftarrow s + \eta \left(1 + \sum_k x_k^2\right)$$

if $g(s) = 1$ but should be 0,

$$\begin{aligned} w_k &\leftarrow w_k - \eta x_k \\ w_0 &\leftarrow w_0 - \eta \end{aligned}$$

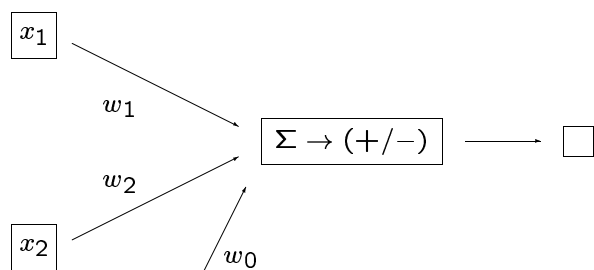
$$\text{so } s \leftarrow s - \eta \left(1 + \sum_k x_k^2\right)$$

otherwise, weights are unchanged

Theorem: This will learn to classify the data correctly, as long as they are **linearly separable**

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Perceptron Learning Example



$$w_1 x_1 + w_2 x_2 + w_0 > 0$$

learning rate $\eta = 0.1$

begin with random weights

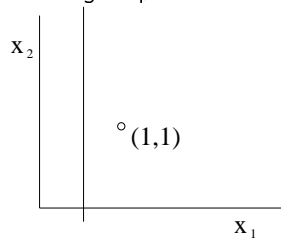
$$\begin{aligned} w_1 &= 0.2 \\ w_2 &= 0.0 \\ w_0 &= -0.1 \end{aligned}$$

$$0.2 x_1 + 0.0 x_2 - 0.1 > 0$$

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Training Steps 1 & 2

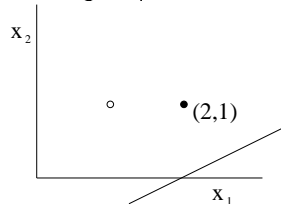
training step 1



$$\begin{aligned} w_1 &\leftarrow w_1 - \eta x_1 = 0.1 \\ w_2 &\leftarrow w_2 - \eta x_2 = -0.1 \\ w_0 &\leftarrow w_0 - \eta = -0.2 \end{aligned}$$

$$0.1 x_1 - 0.1 x_2 - 0.2 > 0$$

training step 2



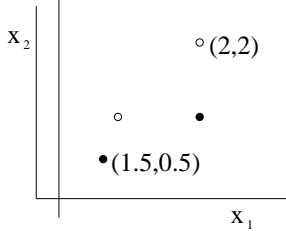
$$\begin{aligned} w_1 &\leftarrow w_1 + \eta x_1 = 0.3 \\ w_2 &\leftarrow w_2 + \eta x_2 = 0.0 \\ w_0 &\leftarrow w_0 + \eta = -0.1 \end{aligned}$$

$$0.3 x_1 + 0.0 x_2 - 0.1 > 0$$

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Training Steps 3 and beyond

training step 3



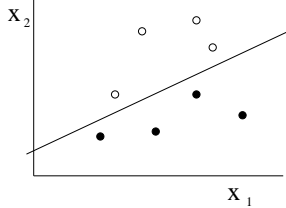
3rd point correctly classified, so no change

4th point:

$$\begin{aligned} w_1 &\leftarrow w_1 - \eta x_1 = 0.1 \\ w_2 &\leftarrow w_2 - \eta x_2 = -0.2 \\ w_0 &\leftarrow w_0 - \eta = -0.2 \end{aligned}$$

$$0.1 x_1 - 0.2 x_2 - 0.2 > 0$$

training step N

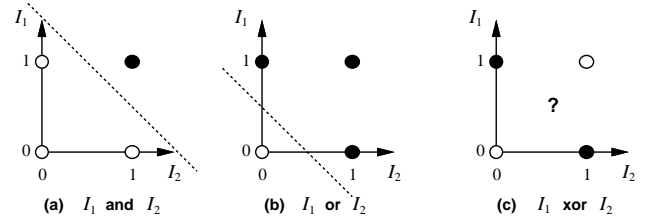


eventually, all the data will be correctly classified (provided it is linearly separable)

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Limitations

Problem: many useful functions are not linearly separable (e.g. XOR)



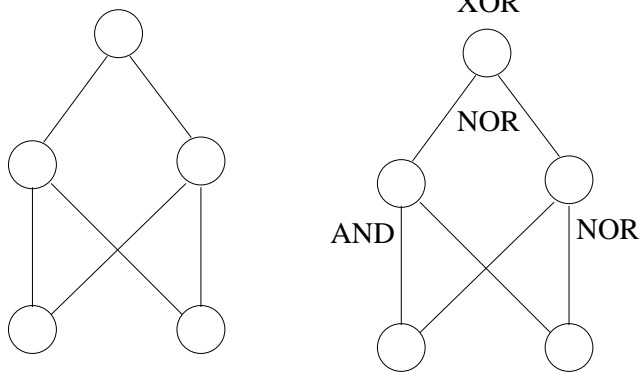
Possible solution:

$x_1 \text{ XOR } x_2$ can be written as: $(x_1 \text{ AND } x_2) \text{ NOR } (x_1 \text{ NOR } x_2)$

Recall that AND, OR and NOR can be implemented by perceptrons

AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Multi-Layer Neural Networks



AIMA Slides ©Russell & Norvig, 1998, Blair 2000, Sklar 2000

nn: Summary

- Neural networks are biologically inspired
- Perceptron can learn any linearly separable function
- Multi-layer neural networks can learn other functions

nn: so what about XOR?

Braitenberg vehicles

- Valentino Braitenberg
- *Vehicles: Experiments in Synthetic Psychology* (1984)
- application of connectionist ideas to physical vehicles
- personified with human emotions

21

22

for next lecture:

- read Malone, T. (1981) *What Makes Computer Games Fun?*, Byte, December 1981.
- read Papert, S. (1980) *Mindstorms: Children, Computers, and Powerful Ideas*, BasicBooks. Chapter 3.
- assignment due Thursday 31 August
- *there will be no additional reading for 31 August... yippee!!!*

23