Perceptrons

Informatics 1 CG: Lecture 5

Mirella Lapata

School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

21 January, 2016

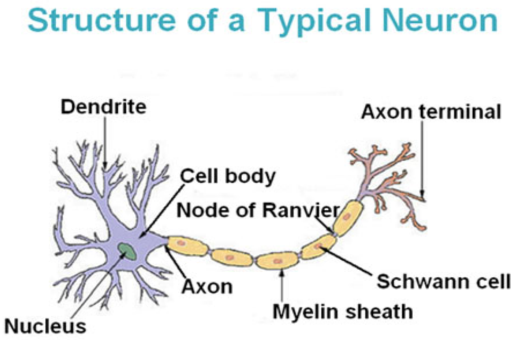
Reading:

Steven Pinker's Words and Rules, Chapter 2
Kevin Gurney's Introduction to Neural Networks, Chapters 2 and 4

Recap: Words and Rules

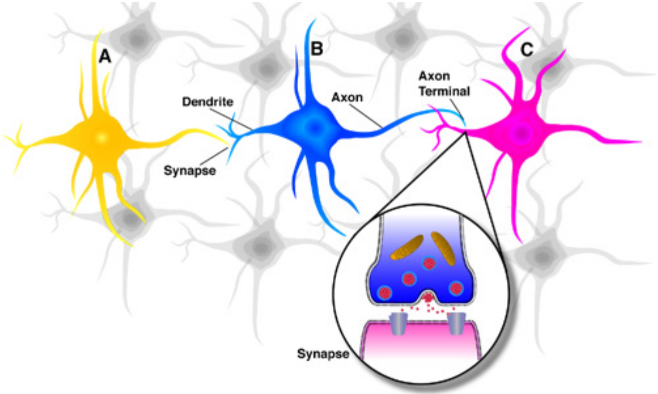
- Does the theory of words and rules explain the dichotomy between regular and irregular verbs?
- Is SPE a plausible theory of how the past tense is formed?
- What does evidence from language development tell us about regular and irregular verbs?
- Maybe a rule is not necessary to explain the past tense.
- Maybe children simply analogise from verbs they already know (e.g., from correct forms like *folded, molded, scolded* to over-regularisations like *holded*).
- All-rules versus all-memory approach.

A Single Neuron



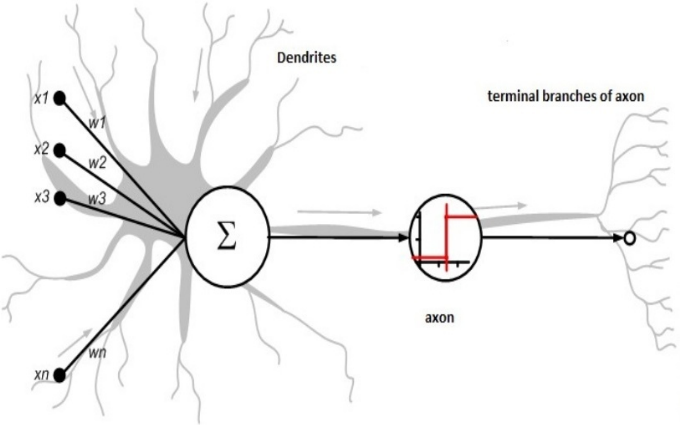
- Neuron receives inputs and combines these in the cell body.
- If the input reaches a threshold, then the neuron may fire (produce an output).
- Some inputs are excitatory, while others are inhibitory.

Biological Neural Networks



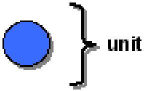
- In biological neural networks, connections are synapses.
- Input connection is conduit through which a member of a network receives information (INPUT)
- Output connection is a conduit through which a member of a network sends information (OUTPUT).

Connectionism is the name for a computer modeling approach based on how information processing occurs in neural networks (connectionist networks are called artificial neural networks).



Anatomy of a Connectionist Model

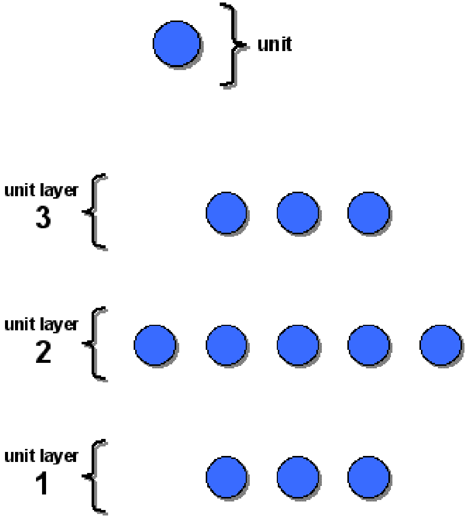
Units are to a connectionist model what neurons are to a biological neural network — the basic information processing structures.



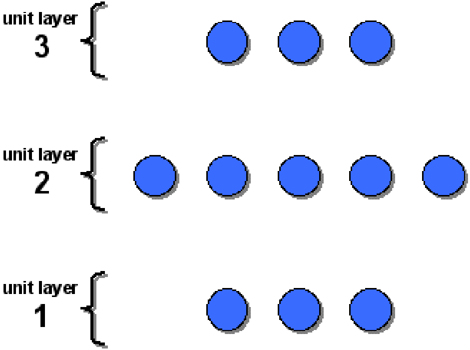
Anatomy of a Connectionist Model

Units are to a connectionist model what neurons are to a biological neural network — the basic information processing structures.

Biological neural networks are organized in layers of neurons. Connectionist models are organized in layers of units, not random clusters.

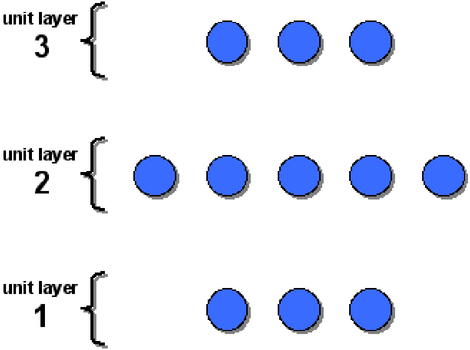


But what you see here still isn't a network. Something is missing.

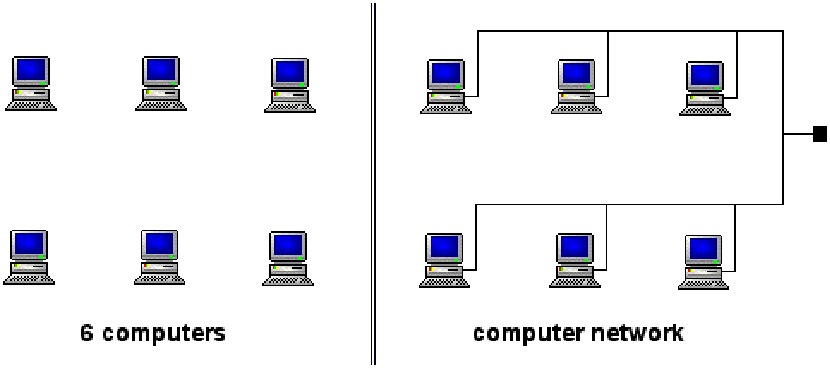


Anatomy of a Connectionist Model

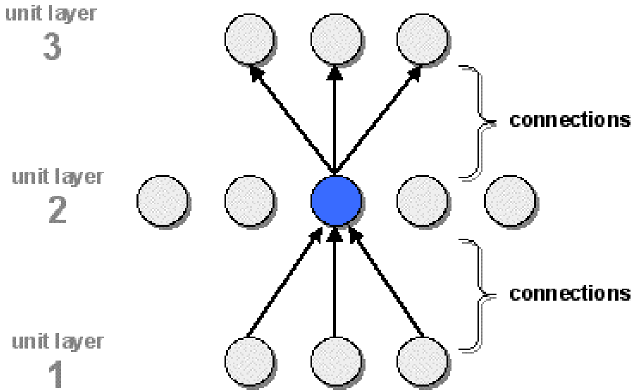
But what you see here still isn't a network. Something is missing. Network connections are conduits through which information flows between members of a network.



Anatomy of a Connectionist Model



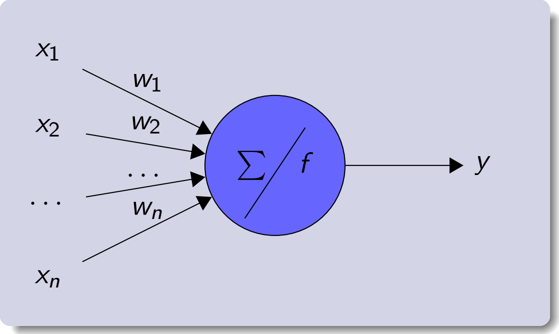
Anatomy of a Connectionist Model



- Connections are represented with lines
- Arrows in a connectionist model indicate the flow of information from one unit to the next.

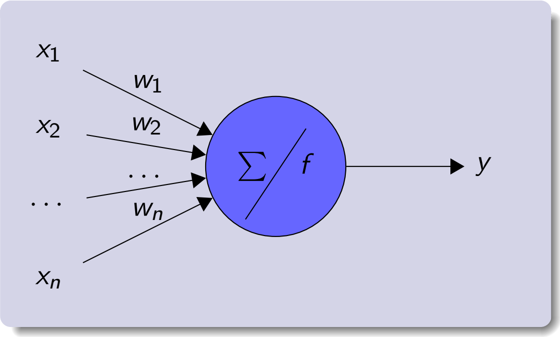
Perceptron: An Artificial Neuron

Perceptron was developed by Frank Rosenblatt in 1957 and can be considered as the simplest artificial neural network.



Perceptron: An Artificial Neuron

Perceptron was developed by Frank Rosenblatt in 1957 and can be considered as the simplest artificial neural network.

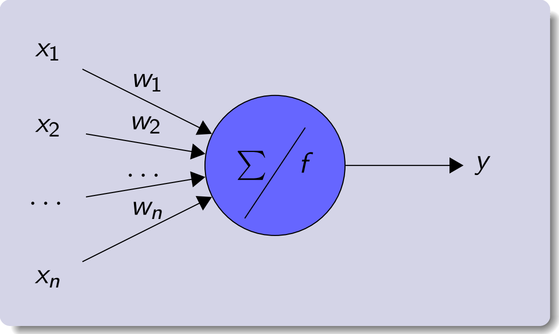


Input function:

$$u(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i$$

Perceptron: An Artificial Neuron

Perceptron was developed by Frank Rosenblatt in 1957 and can be considered as the simplest artificial neural network.



Input function:

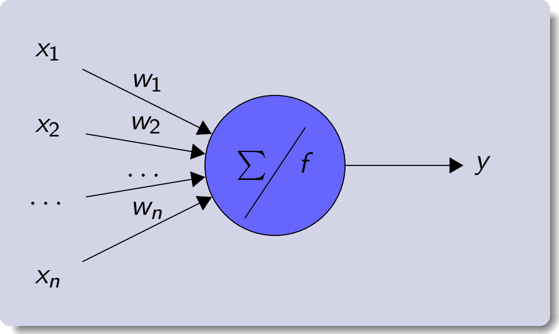
$$u(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i$$

Activation function: threshold

$$y = f(u(\mathbf{x})) = \begin{cases} 1, & \text{if } u(\mathbf{x}) > \theta \\ 0, & \text{otherwise} \end{cases}$$

Perceptron: An Artificial Neuron

Perceptron was developed by Frank Rosenblatt in 1957 and can be considered as the simplest artificial neural network.



Input function:

$$u(\mathbf{x}) = \sum_{i=1}^{n} w_i x_i$$

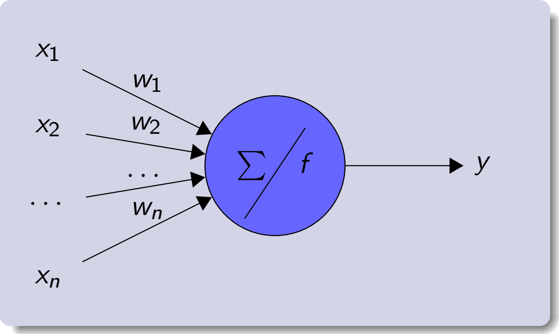
Activation function: threshold

$$y = f(u(\mathbf{x})) = \begin{cases} 1, & \text{if } u(\mathbf{x}) > \theta \\ 0, & \text{otherwise} \end{cases}$$

Activation state:
0 or 1 (-1 or 1)

Perceptron: An Artificial Neuron

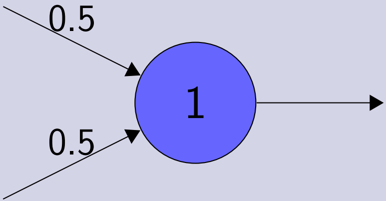
Perceptron was developed by Frank Rosenblatt in 1957 and can be considered as the simplest artificial neural network.



- Inputs are in the range [0, 1], where 0 is "off" and 1 is "on".
- Weights can be any real number (positive or negative).

Perceptrons for Logic

Perceptron for AND

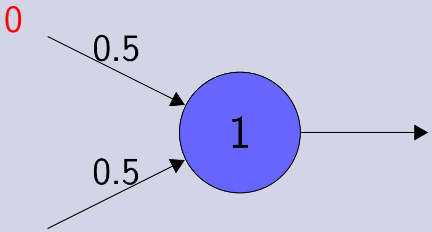


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

Perceptron for AND

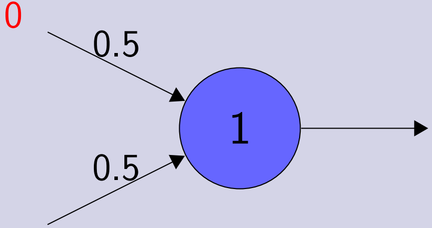


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

Perceptron for AND



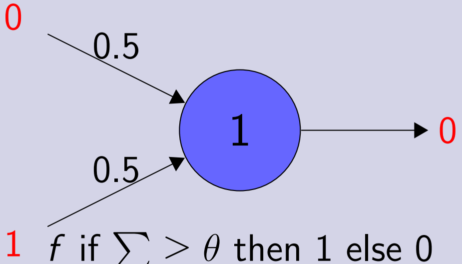
f if $\sum \geq \theta$ then 1 else 0

$0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

Perceptron for AND



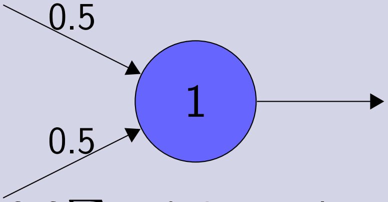
f if $\sum \geq \theta$ then 1 else 0

$$0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$$

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

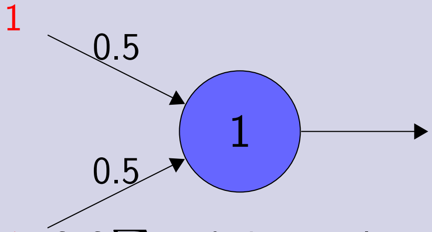
Perceptron for AND



f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptron for AND

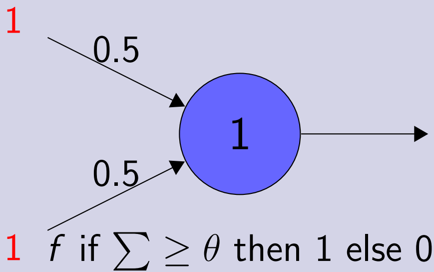


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

Perceptron for AND



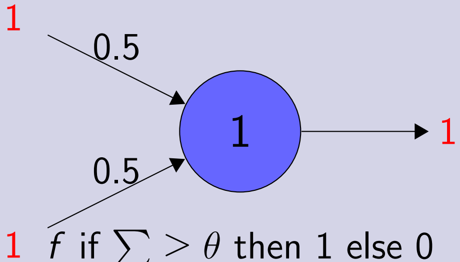
f if $\sum \geq \theta$ then 1 else 0

$$1 \cdot 0.5 + 1 \cdot 0.5 = 1$$

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

Perceptron for AND



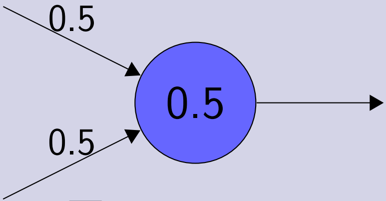
f if $\sum \geq \theta$ then 1 else 0

$$1 \cdot 0.5 + 1 \cdot 0.5 = 1$$

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Perceptrons for Logic

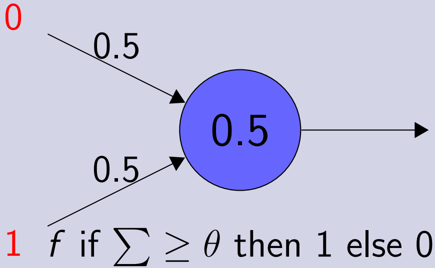
Perceptron for OR



f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

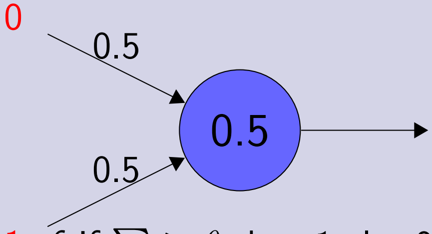
Perceptron for OR



x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

Perceptrons for Logic

Perceptron for OR



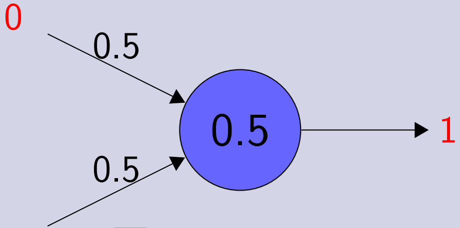
f if $\sum \geq \theta$ then 1 else 0

$0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

Perceptrons for Logic

Perceptron for OR



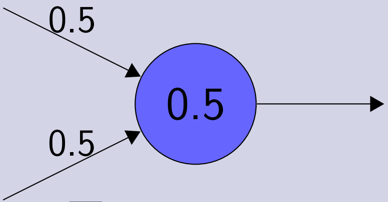
f if $\sum \geq \theta$ then 1 else 0

$0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

Perceptrons for Logic

Perceptron for XOR

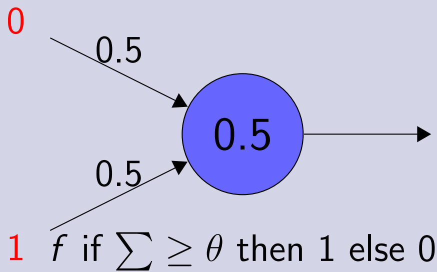


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

XOR is an exclusive OR because it only returns a true value of 1 if the two values are exclusive, i.e., they are both different.

Perceptron for XOR

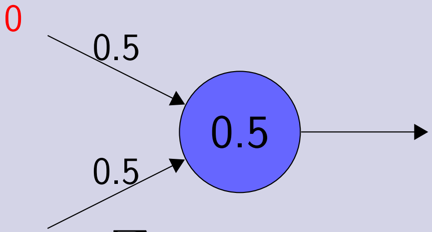


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

XOR is an exclusive OR because it only returns a true value of 1 if the two values are exclusive, i.e., they are both different.

Perceptron for XOR



f if $\sum \geq \theta$ then 1 else 0

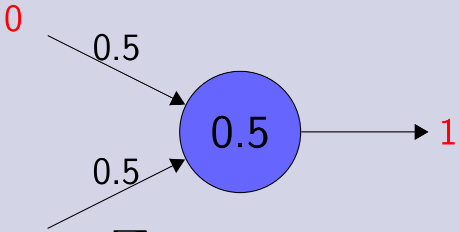
$$0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$$

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

XOR is an exclusive OR because it only returns a true value of 1 if the two values are exclusive, i.e., they are both different.

Perceptrons for Logic

Perceptron for XOR



f if $\sum \geq \theta$ then 1 else 0

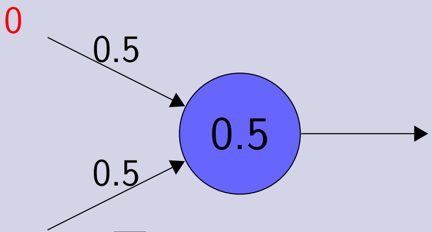
$$0 \cdot 0.5 + 1 \cdot 0.5 = 0.5$$

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

XOR is an exclusive OR because it only returns a true value of 1 if the two values are exclusive, i.e., they are both different.

Perceptrons for Logic

Perceptron for XOR

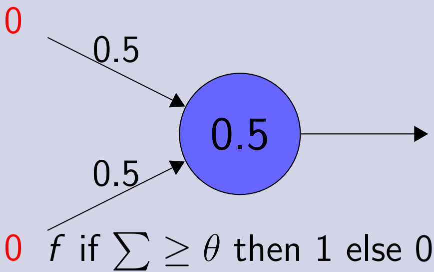


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptrons for Logic

Perceptron for XOR



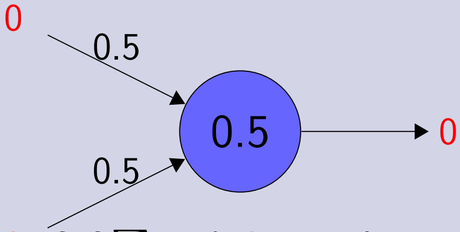
f if $\sum \geq \theta$ then 1 else 0

$$0 \cdot 0.5 + 0 \cdot 0.5 = 0$$

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptrons for Logic

Perceptron for XOR

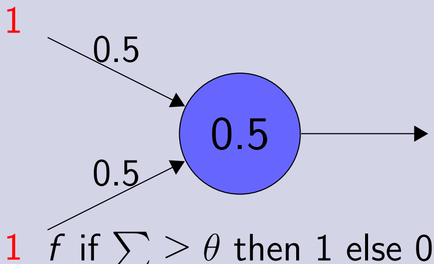


f if $\sum \geq \theta$ then 1 else 0

$$0 \cdot 0.5 + 0 \cdot 0.5 = 0$$

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptron for XOR

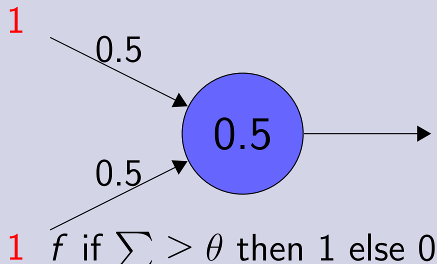


f if $\sum \geq \theta$ then 1 else 0

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptrons for Logic

Perceptron for XOR



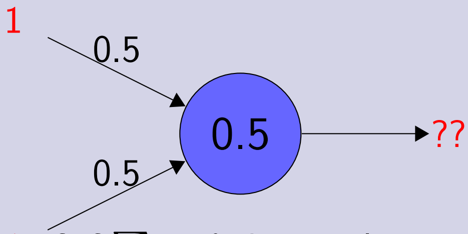
f if $\sum \geq \theta$ then 1 else 0

$$1 \cdot 0.5 + 1 \cdot 0.5 = 1$$

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptrons for Logic

Perceptron for XOR



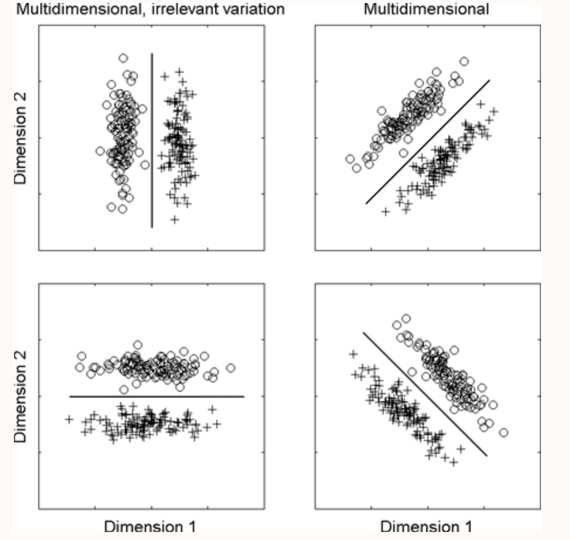
1 f if $\sum \geq \theta$ then 1 else 0

$$1 \cdot 0.5 + 1 \cdot 0.5 = 1$$

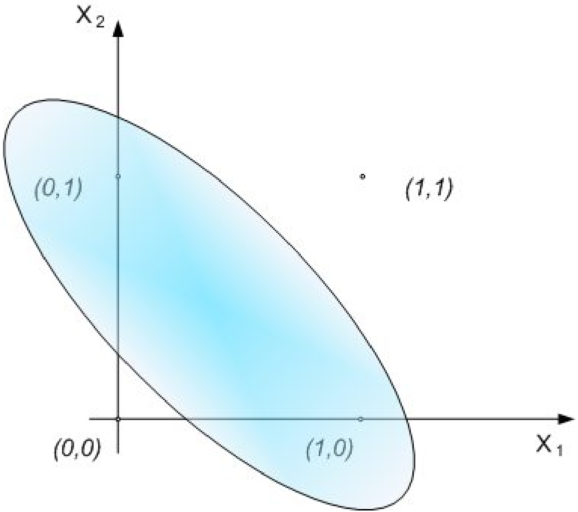
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Perceptrons as Classifiers

Perceptrons are linear classifiers, i.e., they can only separate points with a hyperplane (a straight line).



The XOR probem again



What is the Perceptron Really Seeing?

Sequence of exemplars presented to the Perceptron:

N	input x	target t
1	(0,1,0,0)	1
2	(1,0,0,0)	0
3	(0,1,1,1)	0
4	(1,0,1,0)	0
5	(1,1,1,1)	1
6	(0,1,0,0)	1
...	...	...

- The above Perceptron has 4 inputs (binary) $\approx$ feature vector representing each exemplar.
- The Perceptron sees 6 exemplars or training items
- We know what the right answer is $\approx$ target
- What would happen if we used random weights/threshold?

What is the Perceptron Really Seeing?

Sequence of exemplars presented to the Perceptron:

N	input x	target t	output o
1	(0,1,0,0)	1	0
2	(1,0,0,0)	0	0
3	(0,1,1,1)	0	1
4	(1,0,1,0)	0	1
5	(1,1,1,1)	1	0
6	(0,1,0,0)	1	1
...	...	...	...

- The above Perceptron has 4 inputs (binary) $\approx$ feature vector representing each exemplar.
- The Perceptron sees 6 exemplars or training items
- We know what the right answer is $\approx$ target
- What would happen if we used random weights/threshold?

Learning

Q_1: But... choosing weights and threshold θ for the perceptron is not easy! How to learn the weights and threshold from examples?

A_1: We can use a learning algorithm that adjusts the weights and threshold based on examples.

`http://www.youtube.com/watch?v=vGwemZhPlsA&feature=youtu.be`

Learning: A trick to learn θ

$$\sum_{i=1}^{n} w_i x_i > \theta$$

Learning: A trick to learn θ

$$\sum_{i=1}^{n} w_i x_i > \theta$$

$$\sum_{i=1}^{n} w_i x_i - \theta > 0$$

Learning: A trick to learn θ

$$\sum_{i=1}^{n} w_i x_i > \theta$$

$$\sum_{i=1}^{n} w_i x_i - \theta > 0$$

$$w_1 x_1 + w_2 x_2 + \ldots w_n x_n - \theta > 0$$

$$\sum_{i=1}^{n} w_i x_i > \theta$$

$$\sum_{i=1}^{n} w_i x_i - \theta > 0$$

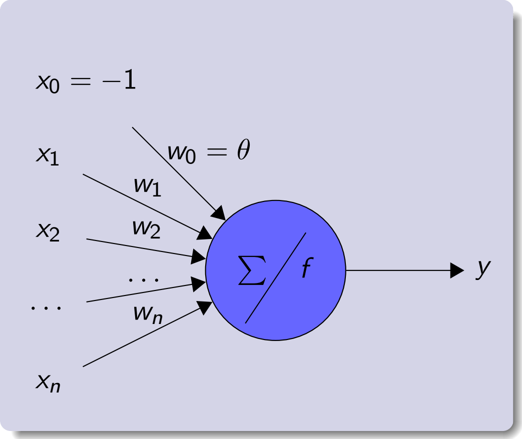
$$w_1 x_1 + w_2 x_2 + \ldots w_n x_n - \theta > 0$$

$$w_1 x_1 + w_2 x_2 + \ldots w_n x_n + \theta(-1) > 0$$

Learning: A trick to learn θ

$$\sum_{i=1}^{n} w_i x_i > \theta$$

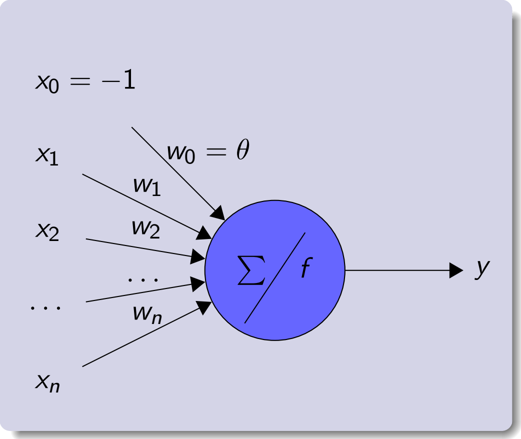
$$\sum_{i=1}^{n} w_i x_i - \theta > 0$$



$$w_1 x_1 + w_2 x_2 + \dots w_n x_n - \theta > 0$$

$$w_1 x_1 + w_2 x_2 + \dots w_n x_n + \theta(-1) > 0$$

Learning: A trick to learn θ



- We can consider θ as a weight to be learnt!
- The input is fixed as -1. The activation function is then:

$$y = f(u(\mathbf{x})) = \begin{cases} 1, & \text{if } u(\mathbf{x}) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Learning Rule

Learning happens by adjusting weights. The threshold can be considered as a weight.

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

- η, $0 < \eta \leq 1$ is a constant called learning rate.
- t is the target output of the current example.
- o is the output obtained by the Perceptron.

Learning Rule

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

$o = 1$ and $t = 1$
$o = 0$ and $t = 1$

- Learning rate η is positive; controls how big changes Δw_i are.
- If $x_i > 0$, $\Delta w_i > 0$. Then w_i increases in an attempt to make $w_i x_i$ become larger than θ.
- If $x_i < 0$, $\Delta w_i < 0$. Then w_i reduces.

Learning Rule

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

$$\begin{aligned} o = 1 \text{ and } t = 1 \quad \Delta w_i &= \eta(t-o)x_i = \eta(1-1)x_i = 0 \\ o = 0 \text{ and } t = 1 \quad \Delta w_i &= \eta(t-o)x_i = \eta(1-0)x_i = \eta x_i \end{aligned}$$

- Learning rate η is positive; controls how big changes Δw_i are.
- If $x_i > 0$, $\Delta w_i > 0$. Then w_i increases in an attempt to make $w_i x_i$ become larger than θ.
- If $x_i < 0$, $\Delta w_i < 0$. Then w_i reduces.

Learning Rule: Exercise

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i \;= \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1, t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

Learning Rule: Exercise

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i \;= \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1, t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

- Determine the output of the Perceptron for the input -1:

Learning Rule: Exercise

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1, t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

- Determine the output of the Perceptron for the input -1:
  $w_1 x_1 = 0.5(-1) = -0.5 \leq \theta \rightarrow o = 0$
- The new weight w_1 after applying the learning rule:

Learning Rule: Exercise

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1, t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

- Determine the output of the Perceptron for the input -1:
  $w_1 x_1 = 0.5(-1) = -0.5 \leq \theta \rightarrow o = 0$
- The new weight w_1 after applying the learning rule:
  $\Delta w_1 = 0.6(1 - 0)(-1) = -0.6 \rightarrow w_1 = 0.5 - 0.6 = -0.1$

Learning Rule: Exercise

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i \;= \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1, t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

- Determine the output of the Perceptron for the input -1:
  $w_1 x_1 = 0.5(-1) = -0.5 \leq \theta \rightarrow o = 0$
- The new weight w_1 after applying the learning rule:
  $\Delta w_1 = 0.6(1 - 0)(-1) = -0.6 \rightarrow w_1 = 0.5 - 0.6 = -0.1$
- The new output of the Perceptron for the input -1:

Learning Rule: Exercise

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i \;= \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1, t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

- Determine the output of the Perceptron for the input -1:
  $w_1 x_1 = 0.5(-1) = -0.5 \leq \theta \rightarrow o = 0$
- The new weight w_1 after applying the learning rule:
  $\Delta w_1 = 0.6(1 - 0)(-1) = -0.6 \rightarrow w_1 = 0.5 - 0.6 = -0.1$
- The new output of the Perceptron for the input -1:
  $w_1 x_1 = -0.1(-1) = 0.1 \geq \theta \rightarrow o = 1$

Learning Algorithm

```
Initialize all weights randomly.
repeat
    for each training example do
        Apply the learning rule.
    end for
until the error is acceptable or a certain number
of iterations is reached
```

This algorithm is guaranteed to find a solution with zero error in a limited number of iterations as long as the examples are linearly separable.

Summary

What does this have to do with the words versus rules debate?

- Connectionism is a computer modeling approach inspired by neural networks.
- Anatomy of a connectionist model: units, connections
- The Perceptron as a linear classifier.
- A learning algorithm for Perceptrons

Next lecture: multilayer perceptrons (neural networks).