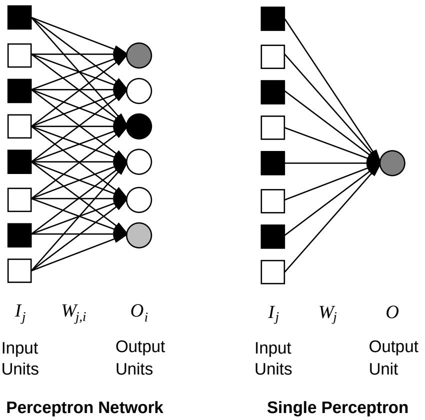
Perceptrons

COMP4211

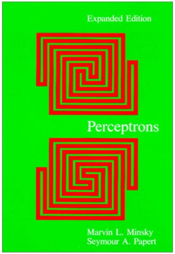
THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

Perceptron

a feed-forward network with only one layer of adjustable (learnable) weights connected to one or more threshold units (as output units)



Perceptron...



- written by Marvin Minsky and Seymour Papert and published in 1969
- introduced by Frank Rosenblatt in 1957

Model

input: $I_1, I_2, \ldots, I_n$

- signals from the other neurons

weights: $w_1, w_2, \ldots, w_n$

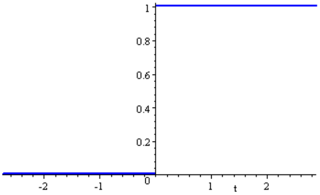
- can be negative

activation function:

- relating the input and output

$$O = \text{step}\left(\sum_{j=1}^{n} w_j I_j - \theta\right)$$

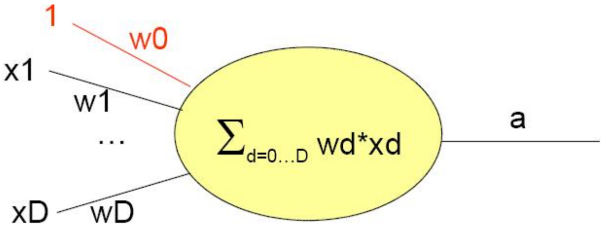
$$\text{step}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \qquad \text{(step function)}$$



Model...

$$O = \text{step}(\sum_{j=1}^{n} w_j I_j - \theta)$$

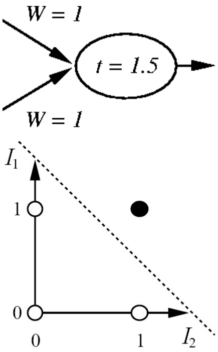
Or, with $w_0 = -\theta$ and $I_0 = 1$



$$O = \text{step}\left(\sum_{j=0}^{n} w_j I_j\right)$$

What Boolean Functions can Perceptrons Represent?

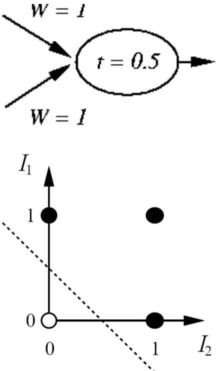
AND?



(a) I_1 **and** I_2

- Perceptron output: $O = \text{step}(\sum_{j=0}^{n} w_j I_j)$
  - decision boundary: $\sum_{j=0}^{n} w_j I_j = 0$

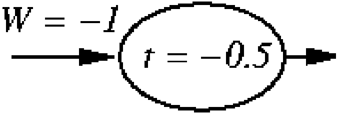
OR?



(b) I_1 **or** I_2

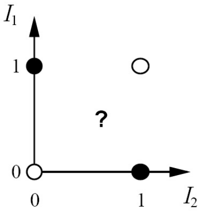
What Boolean Functions can Perceptrons Represent?...

NOT?



What Boolean Functions can Perceptrons Represent?...

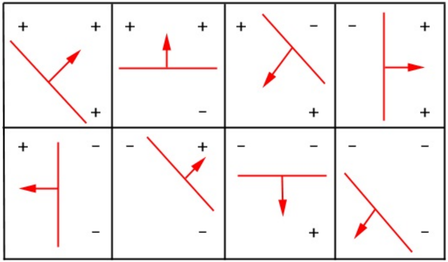
XOR?



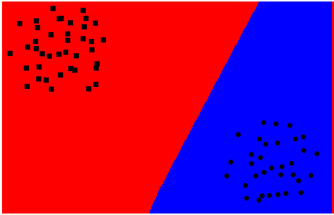
(c) I_1 **xor** I_2

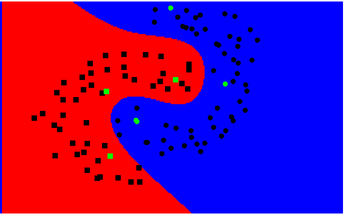
Linearly Separable Functions

- a function can be represented by a single perceptron if and only if the function is linearly separable



Three points in a plane shattered by a half-space.





What Boolean Functions can Neural Networks Represent?

Theorem

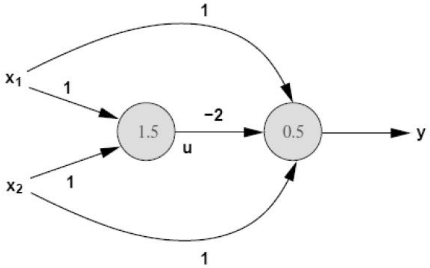
With more layers of sufficiently many perceptrons, any Boolean function can be represented

- any Boolean function can be represented in either DNF (disjunctive normal form) or CNF (conjunctive normal form)
- CNF: conjunction of disjuncts

$$(a \vee c) \wedge (b \vee c)$$
$$(a \vee b) \wedge (\neg b \vee c \vee \neg d) \wedge (d \vee \neg e)$$

- DNF: disjunction of conjuncts

$$(a \wedge c) \vee (b \wedge c)$$
$$(a \wedge \neg b \wedge \neg c) \vee (\neg d \wedge e \wedge f)$$



x_1	x_2	$w_1x_1 + w_2x_2 - 1.5$	u	$w_1x_1 + w_2x_2 + u(-2) - 0.5$	y	class
0	0	-1.5	0	-0.5	0	no
0	1	-0.5	0	0.5	1	yes
1	0	-0.5	0	0.5	1	yes
1	1	0.5	1	-0.5	0	no

Learning Linearly Separable Functions

how to find the appropriate weights?

supervised learning $\Rightarrow$ training examples

Example

apples

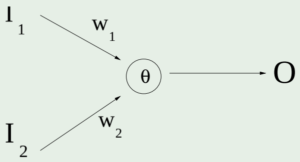
not apples

Learning Linearly Separable Functions...

- convert into features

Example

	I_1	I_2	T
e_1 :	5	1	0
e_2 :	2	1	0
e_3 :	1	1	1
e_4 :	3	3	1
e_5 :	4	2	0
e_6 :	2	3	1



$$O = \text{step}(w_0 + w_1 I_1 + w_2 I_2)$$

Basic Algorithm

- start with some initial values for the weights
- use the perceptron to classify training examples
- modify the weights when errors occur

```
function NEURAL-NETWORK-LEARNING(examples) returns network

  network ← a network with randomly assigned weights
  repeat
     for each e in examples do
        O ← NEURAL-NETWORK-OUTPUT(network, e)
        T ← the observed output values from e
        update the weights in network based on e, O, and T
     end
  until all examples correctly predicted or stopping criterion is reached
  return network
```

- an iterative algorithm

How to Update the Weights?

idea

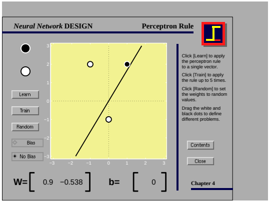
if the observed output (T) is different from the predicted one (O), then make small adjustments in the weights to reduce the difference

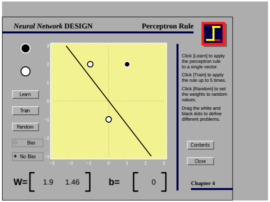
- if the error $T - O$ is positive, then we need to increase O
- if the error is negative, then we need to decrease O
- each input unit contributes $w_j I_j$ to the total input, so if I_j is positive, an increase in w_j will tend to increase O
- if I_j is negative, an increase in w_j will tend to decrease O

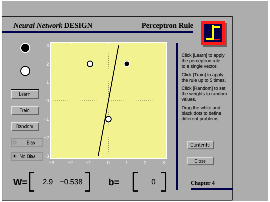
weight update rule

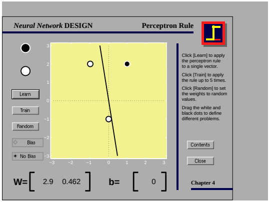
$$w_j \leftarrow w_j + \alpha I_j (T - O), \quad \forall j$$

- α: learning rate


Neural Network DESIGN
Perceptron Rule
Learn
Train
Random
Bias
No Bias
Click [Learn] to apply the perceptron rule to a single vector.
Click [Train] to apply the rule up to 5 times.
Click [Random] to set the weights to random values.
Drag the white and black dots to define different problems.
Contents
Close
W= [0.9 −0.538]
b= [0]
Chapter 4


Neural Network DESIGN
Perceptron Rule
Learn
Train
Random
Bias
No Bias
Click [Learn] to apply the perceptron rule to a single vector.
Click [Train] to apply the rule up to 5 times.
Click [Random] to set the weights to random values.
Drag the white and black dots to define different problems.
Contents
Close
W= [1.9 1.46]
b= [0]
Chapter 4


Neural Network DESIGN
Perceptron Rule
Learn
Train
Random
Bias
No Bias
Click [Learn] to apply the perceptron rule to a single vector.
Click [Train] to apply the rule up to 5 times.
Click [Random] to set the weights to random values.
Drag the white and black dots to define different problems.
Contents
Close
W= [2.9 −0.538]
b= [0]
Chapter 4


Neural Network DESIGN
Perceptron Rule
Learn
Train
Random
Bias
No Bias
Click [Learn] to apply the perceptron rule to a single vector.
Click [Train] to apply the rule up to 5 times.
Click [Random] to set the weights to random values.
Drag the white and black dots to define different problems.
Contents
Close
W= [2.9 0.462]
b= [0]
Chapter 4

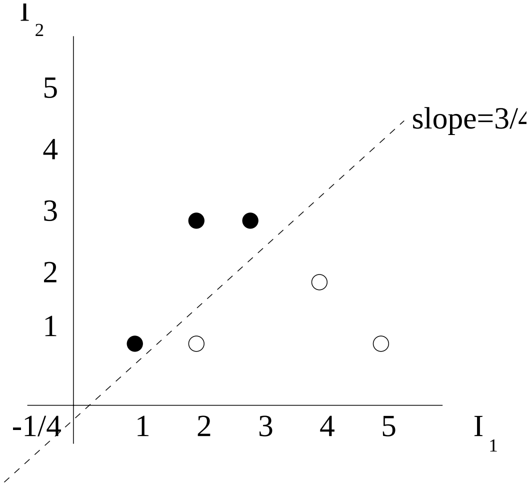
demo

Trace of The Learning Process

Take $\alpha = 1$ and the initial weight values to be 0

Iteration	w^{old}	I	T	O	T = O?	w^{new}
1	(0 0 0)	(1 5 1)	0	1	no	(-1 -5 -1)
2	(-1 -5 -1)	(1 2 1)	0	0	yes	(-1 -5 -1)
3	(-1 -5 -1)	(1 1 1)	1	0	no	(0 -4 0)
4	(0 -4 0)	(1 3 3)	1	0	no	(1 -1 3)
5	(1 -1 3)	(1 4 2)	0	1	no	(0 -5 1)
6	(0 -5 1)	(1 2 3)	1	0	no	(1 -3 4)
7	(1 -3 4)	(1 5 1)	0	0	yes	(1 -3 4)
8	(1 -3 4)	(1 2 1)	0	0	yes	(1 -3 4)
9	(1 -3 4)	(1 1 1)	1	1	yes	(1 -3 4)
10	(1 -3 4)	(1 3 3)	1	1	yes	(1 -3 4)
11	(1 -3 4)	(1 4 2)	0	0	yes	(1 -3 4)
12	(1 -3 4)	(1 2 3)	1	1	yes	(1 -3 4)

Geometric Interpretation of Solution



Perceptron Convergence Theorem

if the training examples are linearly separable, then applying the perceptron weight updating rule can

- always converge to some solution (i.e. a set of weights)
- in a finite number of steps for any initial choice of weights

what if the examples are not linearly separable?

- perceptron may fail to converge