Artificial Intelligence

Neural Networks

Lesson 3:
Training Threshold Logic Units

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Training threshold logic units
- Delta rule
- Training examples
- Convergence

Training Threshold Logic Units (1)

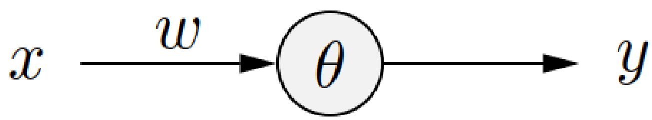
- Geometric interpretation provides a way to construct threshold logic units with 2 and 3 inputs, but:
  - Not an automatic method (human visualization needed).
  - Not feasible for more than 3 inputs.

Training Threshold Logic Units (2)

- Automatic training:
  - Start with random values for weights and threshold.
  - Determine the error of the output for a set of training patterns.
  - Error is a function of the weights and the threshold: $e = e(w_1, \dots, w_n, \theta)$.
  - Adapt weights and threshold so that the error becomes smaller.
  - Iterate adaptation until the error vanishes.

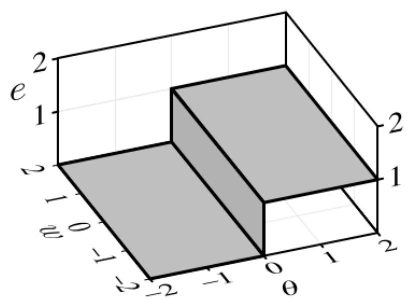
Training Threshold Logic Units (3)

Single input threshold logic unit for the negation $\neg x$

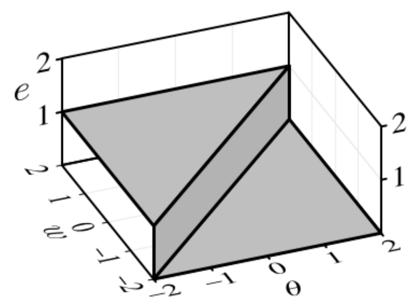


x	y
0	1
1	0

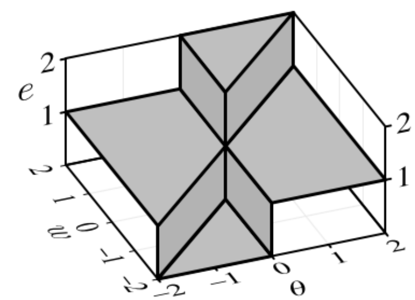
Output error as a function of weight and threshold



error for $x = 0$



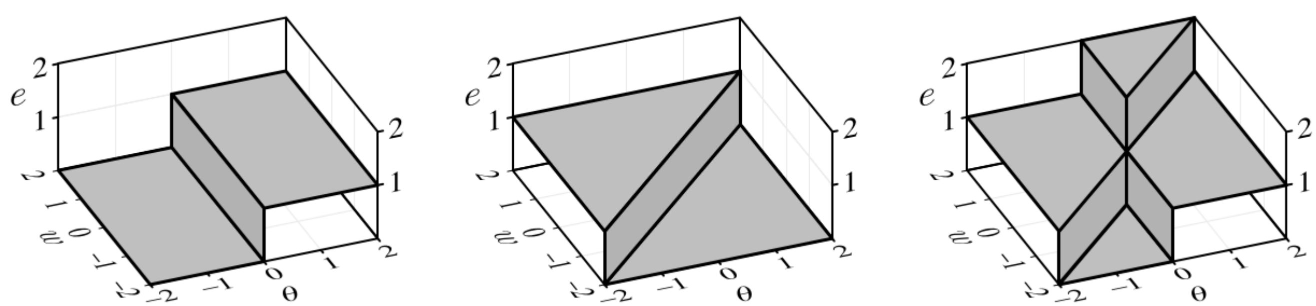
error for $x = 1$



sum of errors

Training Threshold Logic Units (4)

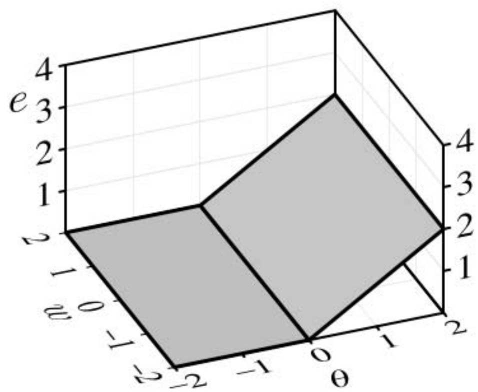
- The error function cannot be used directly, because it consists of plateaus.
- Solution: If the computed output is wrong, take into account how far the weighted sum is from the threshold
  - "how wrong" the relation of weighted sum and threshold is).



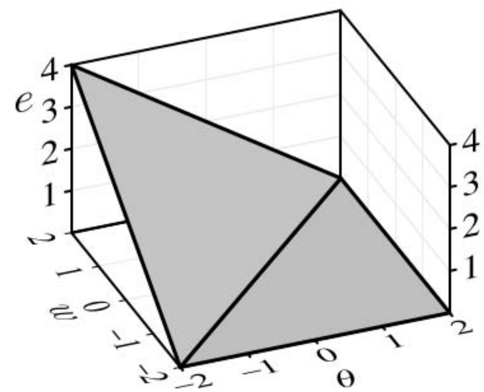
error for $x = 0$ error for $x = 1$ sum of errors

Training Threshold Logic Units (5)

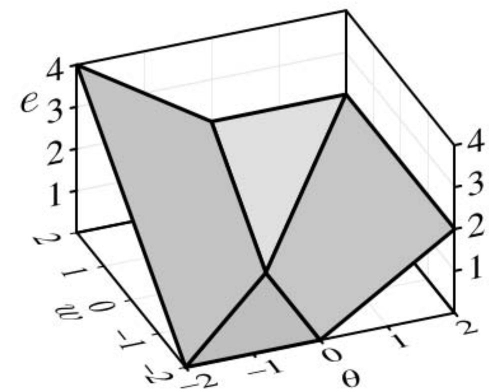
Modified output error as a function of weight and threshold



error for $x = 0$



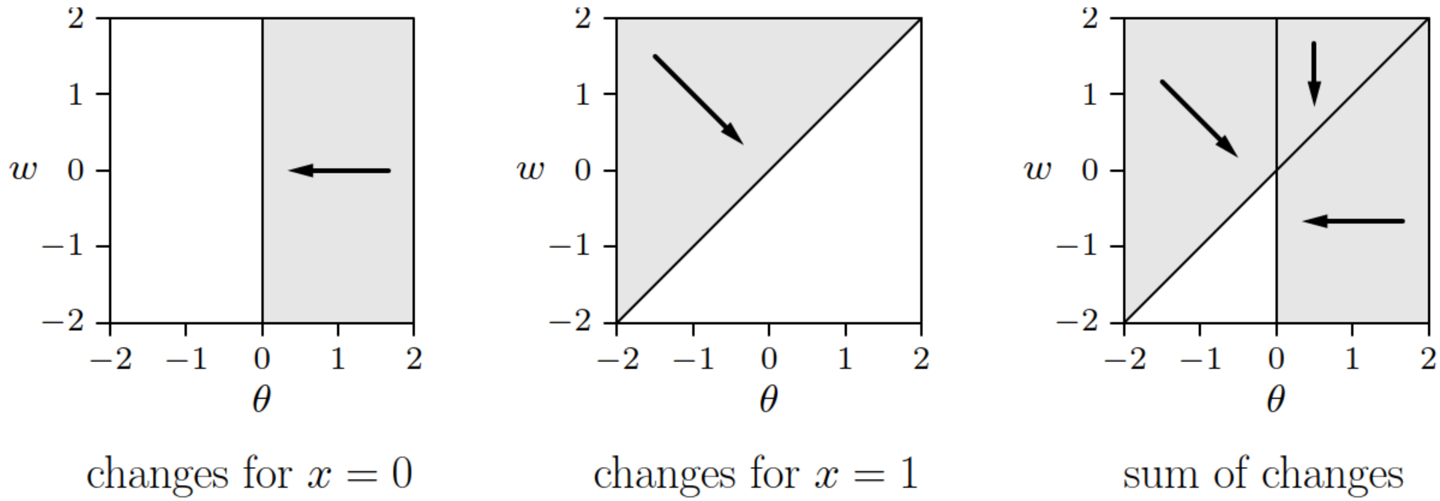
error for $x = 1$



sum of errors

Training Threshold Logic Units (6)

Parameter changes:



- Start at a random point.
- Iteratively adapt parameters according to the direction corresponding to the current point.
- Stop if the error vanishes

Training Threshold Logic Units (7)

- **Online learning**
  - Acquire one learning pattern at a time
  - Compute parameter corrections for this learning pattern
  - Apply parameter corrections

- **Batch learning**
  - Collect a sequence of learning patterns during a learning epoch
  - Compute the cumulated parameter corrections for the entire set of learning patterns
  - Apply the parameter corrections

Delta Rule (1)

Training Rule: Delta Rule (Widrow-Hoff)

Let $\vec{x} = (x_1, \ldots, x_n)^T$ be an input vector of a threshold logic unit, o the desired output for this input vector and y the actual output of the threshold logic unit. If $y \neq o$, then the threshold θ and the weight vector $\vec{w} = (w_1, \ldots, w_n)^T$ are adapted as follows in order to reduce the error:

$\forall i \in \{1, \ldots, n\}$:

$\theta^{(new)} = \theta^{(old)} + \Delta\theta$, with $\Delta\theta = -\eta(o - y)$,

$w_i^{(new)} = w_i^{(old)} + \Delta w_i$, with $\Delta w_i = \eta(o - y)x_i$,

Where η is the learning rate

Delta Rule (2)

- **Online Training**: Adapt parameters after each training pattern.

epoch	x	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw	θ	w
								1.5	2
1	0	1	-1.5	0	1	-1	0	0.5	2
	1	0	1.5	1	-1	1	-1	1.5	1
2	0	1	-1.5	0	1	-1	0	0.5	1
	1	0	0.5	1	-1	1	-1	1.5	0
3	0	1	-1.5	0	1	-1	0	0.5	0
	1	0	0.5	0	0	0	0	0.5	0
4	0	1	-0.5	0	1	-1	0	-0.5	0
	1	0	0.5	1	-1	1	-1	0.5	-1
5	0	1	-0.5	0	1	-1	0	-0.5	-1
	1	0	-0.5	0	0	0	0	-0.5	-1
6	0	1	0.5	1	0	0	0	-0.5	-1
	1	0	-0.5	0	0	0	0	-0.5	-1

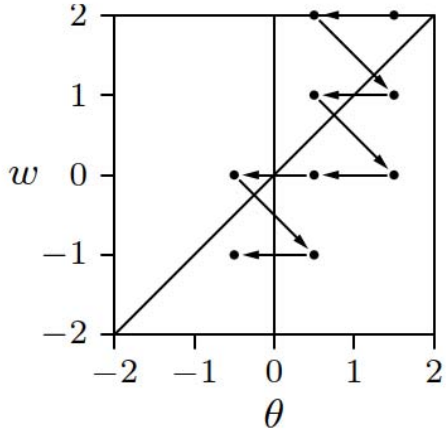
Delta Rule (3)

- **Batch Training**: Adapt parameters only at the end of each epoch, that is, after a traversal of all training patterns.

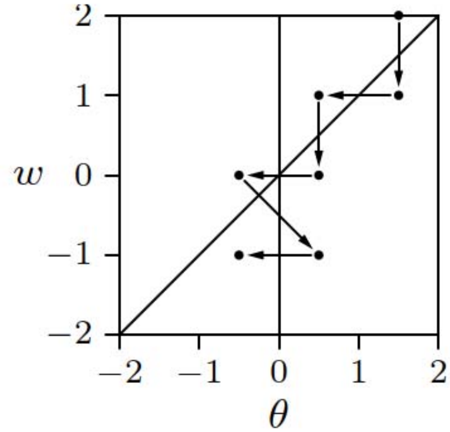
epoch	x	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw	θ	w
								1.5	2
1	0	1	-1.5	0	1	-1	0		
	1	0	0.5	1	-1	1	-1	1.5	1
2	0	1	-1.5	0	1	-1	0		
	1	0	-0.5	0	0	0	0	0.5	1
3	0	1	-0.5	0	1	-1	0		
	1	0	0.5	1	-1	1	-1	0.5	0
4	0	1	-0.5	0	1	-1	0		
	1	0	-0.5	0	0	0	0	-0.5	0
5	0	1	0.5	1	0	0	0		
	1	0	0.5	1	-1	1	-1	0.5	-1
6	0	1	-0.5	0	1	-1	0		
	1	0	-1.5	0	0	0	0	-0.5	-1
7	0	1	0.5	1	0	0	0		
	1	0	-0.5	0	0	0	0	-0.5	-1

Training Examples (1)

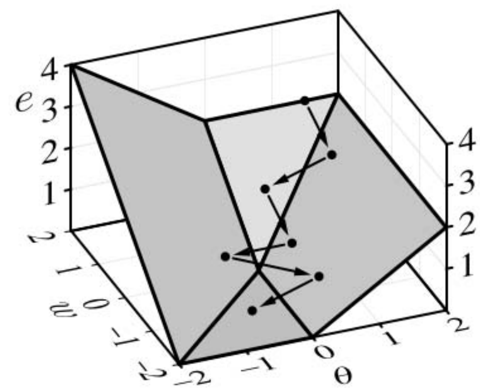
Example of online and batch training



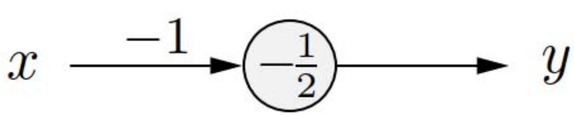
Online Training

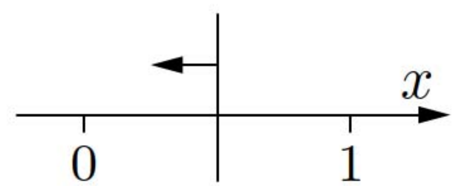


Batch Training



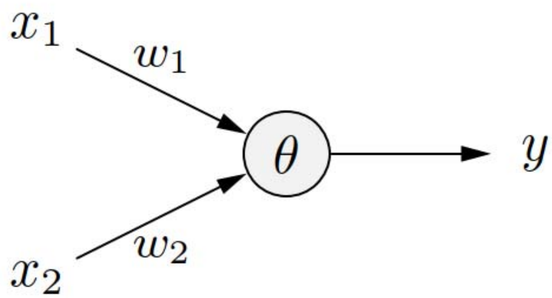
Batch Training



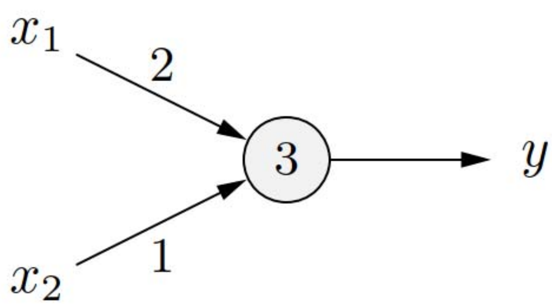


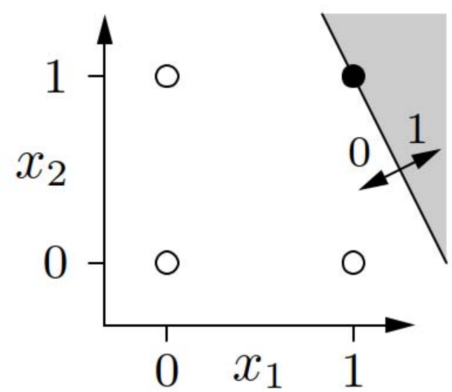
Training Examples (2)

Threshold logic unit with two inputs for the conjunction



x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1





Training Examples (3)

epoch	x_1	x_2	o	$\vec{x}\vec{w}$	y	e	$\Delta\theta$	Δw_1	Δw_2	θ	w_1	w_2
										0	0	0
1	0	0	0	0	1	−1	1	0	0	1	0	0
	0	1	0	−1	0	0	0	0	0	1	0	0
	1	0	0	−1	0	0	0	0	0	1	0	0
	1	1	1	−1	0	1	−1	1	1	0	1	1
2	0	0	0	0	1	−1	1	0	0	1	1	1
	0	1	0	0	1	−1	1	0	−1	2	1	0
	1	0	0	−1	0	0	0	0	0	2	1	0
	1	1	1	−1	0	1	−1	1	1	1	2	1
3	0	0	0	−1	0	0	0	0	0	1	2	1
	0	1	0	0	1	−1	1	0	−1	2	2	0
	1	0	0	0	1	−1	1	−1	0	3	1	0
	1	1	1	−2	0	1	−1	1	1	2	2	1
4	0	0	0	−2	0	0	0	0	0	2	2	1
	0	1	0	−1	0	0	0	0	0	2	2	1
	1	0	0	0	1	−1	1	−1	0	3	1	1
	1	1	1	−1	0	1	−1	1	1	2	2	2
5	0	0	0	−2	0	0	0	0	0	2	2	2
	0	1	0	0	1	−1	1	0	−1	3	2	1
	1	0	0	−1	0	0	0	0	0	3	2	1
	1	1	1	0	1	0	0	0	0	3	2	1
6	0	0	0	−3	0	0	0	0	0	3	2	1
	0	1	0	−2	0	0	0	0	0	3	2	1
	1	0	0	−1	0	0	0	0	0	3	2	1
	1	1	1	0	1	0	0	0	0	3	2	1

Convergence (1)

Convergence Theorem:

Let $L = \{(\overrightarrow{x_1}, o_1), \ldots (\overrightarrow{x_m}, o_m)\}$ be a set of training patterns, each consisting of an input vector $\overrightarrow{x_i} \in \mathbb{R}^n$ and a desired output $o_i \in \{0, 1\}$.

Furthermore, let $L_0 = \{(\vec{x}, o) \in L \mid o = 0\}$ and $L_1 = \{(\vec{x}, o) \in L \mid o = 1\}$.

If L_0 and L_1 are linearly separable, that is, if $\vec{w} \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ exist such that

$\forall(\vec{x}, 0) \in L_0: \vec{w}^T \vec{x} < \theta$, and

$\forall(\vec{x}, 1) \in L_0: \vec{w}^T \vec{x} \geq \theta$,

then online as well as batch training terminate.

Convergence (2)

- The algorithms terminate only when the error vanishes.
- The resulting threshold and weights solve the problem.
- For not linearly separable problems the algorithms do not terminate
  - oscillation, repeated computation of same non-solving $\vec{w}$ and θ.

Convergence (3)

- Parameter correction depends on the encoding of the Boolean values.
- Using *false=0* and *true=1* may result in less opportunities for correcting the parameters by means of the Delta Rule.

- To speed up learning more frequent correction opportunities can be achieved by adopting a different encoding scheme

ADALINE (ADAptive LINear Element)

uses *false=-1* and *true=1*

Convergence (4)

- Single threshold logic units can only compute linearly separable functions.
  - Training single threshold logic units with the delta rule is easy and fast and guaranteed to find a solution, if one exists.
- Networks of threshold logic units can compute arbitrary Boolean functions.
  - Networks of threshold logic units cannot be trained
    - there are no desired values for the neurons of the first layer(s),
    - the problem can usually be solved with several different functions computed by the neurons of the first layer(s) (non-unique solution).