Künstliche Intelligenz
Perceptron

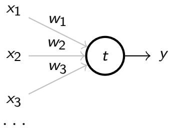
Jun.-Prof. Dr.-Ing. Stefan Lüdtke

Universität Rostock

Institut für Visual & Analytic Computing

Dynamics of a Simple Perceptron (Artificial Threshold Neuron)

- General schema and dynamics:

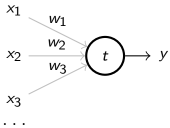


$$y = \begin{cases} 1 & \text{if } \sum_i x_i \cdot w_i \geq t \\ 0 & \text{otherwise} \end{cases}$$

$$y = \Theta(\vec{x} \cdot \vec{w} - t)$$

Dynamics of a Simple Perceptron (Artificial Threshold Neuron)

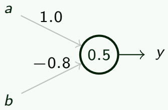
- General schema and dynamics:



$$y = \begin{cases} 1 & \text{if } \sum_i x_i \cdot w_i \geq t \\ 0 & \text{otherwise} \end{cases}$$

$$y = \Theta(\vec{x} \cdot \vec{w} - t)$$

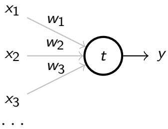
- A small two-input example:



a	b	y
0	0	0
0	1	0
1	0	1
1	1	0

Dynamics of a Simple Perceptron (Artificial Threshold Neuron)

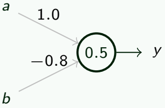
- General schema and dynamics:



$$y = \begin{cases} 1 & \text{if } \sum_i x_i \cdot w_i \geq t \\ 0 & \text{otherwise} \end{cases}$$

$$y = \Theta(\vec{x} \cdot \vec{w} - t)$$

- A small two-input example:

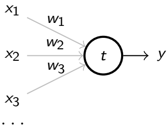


a	b	y
0	0	0
0	1	0
1	0	1
1	1	0

$y = a \wedge \neg b$

Dynamics of a Simple Perceptron (Artificial Threshold Neuron)

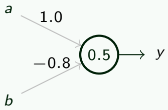
- General schema and dynamics:



$$y = \begin{cases} 1 & \text{if } \sum_i x_i \cdot w_i \geq t \\ 0 & \text{otherwise} \end{cases}$$

$$y = \Theta(\vec{x} \cdot \vec{w} - t)$$

- A small two-input example:



a	b	y
0	0	0
0	1	0
1	0	1
1	1	0

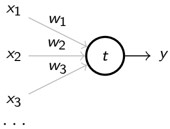
$y = a \wedge \neg b$

a	b	y
0.2	0.0	0
0.0	0.2	0
0.5	0.0	1
5.0	1.0	1

?

Dynamics of a Simple Perceptron (Artificial Threshold Neuron)

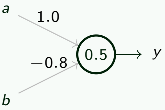
- General schema and dynamics:



$$y = \begin{cases} 1 & \text{if } \sum_i x_i \cdot w_i \geq t \\ 0 & \text{otherwise} \end{cases}$$

$$y = \Theta(\vec{x} \cdot \vec{w} - t)$$

- A small two-input example:



a	b	y
0	0	0
0	1	0
1	0	1
1	1	0

$y = a \wedge \neg b$

a	b	y
0.2	0.0	0
0.0	0.2	0
0.5	0.0	1
5.0	1.0	1

?

Take-away-messages of section: *"Structure and Dynamics of Perceptrons"*

E You should now be able to ...

- explain what a simple perceptron is
- describe the dynamics of a single threshold unit
- name one of the first existing hardware-implementation of a perceptron
- describe the structure, dynamics and adaptation process of the Mark 1 perceptron

Learning objective of section: *"Training Perceptrons"*

In this section we will ...

- discuss the idea behind training a perceptron based on data
- discuss the error function generated by artificial neural networks in general and for threshold units in particular
- introduce a corresponding adaptation schema - the delta rule
- discuss the idea of a hypothesis space
- discuss limitations of single layer perceptrons

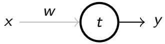
Training a Single Layer Perceptron

Training a Perceptron

Result: A trained perceptron
Input: A perceptron P, a set of training data D

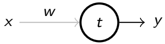
```
Let π_P be the set of parameters of P:
  weights and thresholds
Initialise all parameters π_P, randomly
repeat
    Compute error E wrt. D and current parameters π_P
    Modify parameters π_P such that the error decreases
until E is acceptable
return The modified perceptron P
```

The Simplest Perceptron



$$y_{w,t}(x) = \begin{cases} 1 & \text{if } x \cdot w \geq t \\ 0 & \text{otherwise} \end{cases}$$

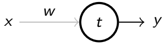
The Simplest Perceptron



$$y_{w,t}(x) = \begin{cases} 1 & \text{if } x \cdot w \geq t \\ 0 & \text{otherwise} \end{cases}$$

1. How many parameters has this perceptron?

The Simplest Perceptron

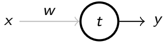


$$y_{w,t}(x) = \begin{cases} 1 & \text{if } x \cdot w \geq t \\ 0 & \text{otherwise} \end{cases}$$

1. How many parameters has this perceptron?
2. Let the following dataset be given. How big is the error E?

x	y	*Id*	*Error*
0	1	*a*	

The Simplest Perceptron

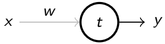


$$y_{w,t}(x) = \begin{cases} 1 & \text{if } x \cdot w \geq t \\ 0 & \text{otherwise} \end{cases}$$

1. How many parameters has this perceptron?
2. Let the following dataset be given. How big is the error E?

x	y	*Id*	*Error*
0	1	*a*	$\lvert y_{w,t}(0) - 1 \rvert$

The Simplest Perceptron

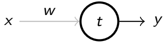


$$y_{w,t}(x) = \begin{cases} 1 & \text{if } x \cdot w \geq t \\ 0 & \text{otherwise} \end{cases}$$

1. How many parameters has this perceptron?
2. Let the following dataset be given. How big is the error E?

x	y	*Id*	*Error*
0	1	a	$\lvert y_{w,t}(0) - 1 \rvert$
1	0	b	$\lvert y_{w,t}(1) - 0 \rvert$

The Simplest Perceptron



$$y_{w,t}(x) = \begin{cases} 1 & \text{if } x \cdot w \geq t \\ 0 & \text{otherwise} \end{cases}$$

1. How many parameters has this perceptron?
2. Let the following dataset be given. How big is the error E?

x	y	*Id*	*Error*
0	1	a	$\lvert y_{w,t}(0) - 1 \rvert$
1	0	b	$\lvert y_{w,t}(1) - 0 \rvert$

E Give the mathematical formula for E wrt. w and t (provide the signature and the definition)!

The Simplest Perceptron – The Error

$$x \xrightarrow{w} t \longrightarrow y$$

x	y	*Id*
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w, t)$$

- Error wrt. datapoint b ($x = 1$):

$$E_b(w, t)$$

- Total error ($E : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$):

$$E(w, t) = E_a(w, t) + E_b(w, t)$$

The Simplest Perceptron – The Error

$$x \xrightarrow{w} (t) \longrightarrow y$$

x	y	*Id*
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w, t) = \begin{cases} 0 & \text{if } w \cdot 0 \geq t \\ 1 & \text{if } w \cdot 0 < t \end{cases}$$

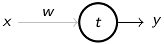
- Error wrt. datapoint b ($x = 1$):

$$E_b(w, t)$$

- Total error ($E : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$):

$$E(w, t) = E_a(w, t) + E_b(w, t)$$

The Simplest Perceptron – The Error



x	y	*ld*
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } w \cdot 0 \geq t \\ 1 & \text{if } w \cdot 0 < t \end{cases} = \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$

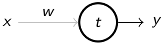
- Error wrt. datapoint b ($x = 1$):

$$E_b(w,t)$$

- Total error ($E : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$):

$$E(w,t) = E_a(w,t) + E_b(w,t)$$

The Simplest Perceptron – The Error



x	y	Id
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w,t) \;= \begin{cases} 0 & \text{if } w \cdot 0 \geq t \\ 1 & \text{if } w \cdot 0 < t \end{cases} \;= \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$

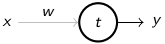
- Error wrt. datapoint b ($x = 1$):

$$E_b(w,t) \;= \begin{cases} 0 & \text{if } w \cdot 1 < t \\ 1 & \text{if } w \cdot 1 \geq t \end{cases}$$

- Total error ($E : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^+$):

$$E(w,t) = E_a(w,t) + E_b(w,t)$$

The Simplest Perceptron – The Error



x	y	Id
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w, t) = \begin{cases} 0 & \text{if } w \cdot 0 \geq t \\ 1 & \text{if } w \cdot 0 < t \end{cases} = \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$

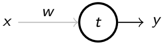
- Error wrt. datapoint b ($x = 1$):

$$E_b(w, t) = \begin{cases} 0 & \text{if } w \cdot 1 < t \\ 1 & \text{if } w \cdot 1 \geq t \end{cases} = \begin{cases} 0 & \text{if } w < t \\ 1 & \text{if } w \geq t \end{cases}$$

- Total error ($E : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$):

$$E(w, t) = E_a(w, t) + E_b(w, t)$$

The Simplest Perceptron – The Error



x	y	Id
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w, t) = \begin{cases} 0 & \text{if } w \cdot 0 \geq t \\ 1 & \text{if } w \cdot 0 < t \end{cases} = \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$

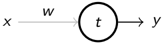
- Error wrt. datapoint b ($x = 1$):

$$E_b(w, t) = \begin{cases} 0 & \text{if } w \cdot 1 < t \\ 1 & \text{if } w \cdot 1 \geq t \end{cases} = \begin{cases} 0 & \text{if } w < t \\ 1 & \text{if } w \geq t \end{cases}$$

- Total error ($E : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$):

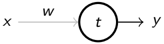
$$E(w, t) = E_a(w, t) + E_b(w, t) = \begin{cases} 0 & \text{if } 0 \geq t \text{ and } w < t \\ 2 & \text{if } 0 < t \text{ and } w \geq t \\ 1 & \text{otherwise} \end{cases}$$

The Simplest Perceptron – The Error Surfaces



x	y	*Id*
0	1	*a*
1	0	*b*

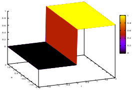
The Simplest Perceptron – The Error Surfaces



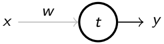
x	y	*Id*
0	1	*a*
1	0	*b*

- Error wrt. datapoint a ($x = 0$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$



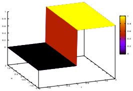
The Simplest Perceptron – The Error Surfaces



x	y	*Id*
0	1	*a*
1	0	*b*

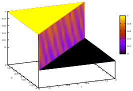
- Error wrt. datapoint a ($x = 0$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$

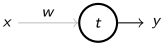


- Error wrt. datapoint b ($x = 1$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } w < t \\ 1 & \text{if } w \geq t \end{cases}$$



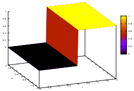
The Simplest Perceptron – The Error Surfaces



x	y	*Id*
0	1	*a*
1	0	*b*

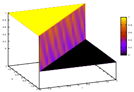
- Error wrt. datapoint a ($x = 0$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } 0 \geq t \\ 1 & \text{if } 0 < t \end{cases}$$



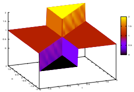
- Error wrt. datapoint b ($x = 1$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } w < t \\ 1 & \text{if } w \geq t \end{cases}$$



- Total error ($E : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$):

$$E_a(w,t) = \begin{cases} 0 & \text{if } 0 \geq t \text{ and } w < t \\ 2 & \text{if } 0 < t \text{ and } w \geq t \\ 1 & \text{if otherwise} \end{cases}$$



Training a Perceptron

Training a Perceptron

Result: A trained perceptron
Input: A perceptron P, a set of training data D

```
Let π_P be the set of parameters of P:
  weights and thresholds
Initialise all parameters π_P, randomly
repeat
    Compute error E wrt. D and current parameters π_P
    Modify parameters π_P such that the error decreases
until E is acceptable
return The modified perceptron P
```

Training a Perceptron

Training a Perceptron

Result: A trained perceptron
Input: A perceptron P, a set of training data D

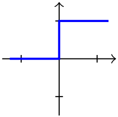
Let π_P be the set of parameters of P:
weights and thresholds
Initialise all parameters π_P, randomly
repeat
  Compute error E wrt. D and current parameters π_P
  Modify parameters π_P such that the error decreases
   But, how can we modify the parameters?
until *E is acceptable*
return *The modified perceptron P*

From plateaus to slopes

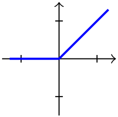
- The error function for the step function is not well suited for training

From plateaus to slopes

- The error function for the step function is not well suited for training
- Instead of using crisp decisions (step function), we will use a smoother version (ramp function) as activation function:



threshold

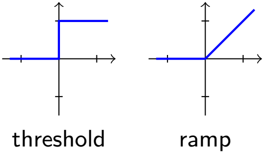


ramp

(The ramp function is called ReLU nowadays)

From plateaus to slopes

- The error function for the step function is not well suited for training
- Instead of using crisp decisions (step function), we will use a smoother version (ramp function) as activation function:



(The ramp function is called ReLU nowadays)

- I.e., instead of error $= 1$, we will use the distance from the weighted sum to the threshold

The Simplest Perceptron – The Error-Surface again

T Draw the error-surfaces for the **ramp** function!

(gnuplot PerceptronRamp.gnuplot)

Training a Perceptron

Training a Perceptron

Result: A trained perceptron
Input: A perceptron P, a set of training data D

Let π_P be the set of parameters of P:
  weights and thresholds
Initialise all parameters π_P, randomly
repeat
    Compute error E wrt. D and current parameters π_P
    Modify parameters π_P such that the error decreases
      But, how can we modify the parameters?
until *E is acceptable*
return *The modified perceptron P*

Training a Perceptron

Training a Perceptron

Result: A trained perceptron
Input: A perceptron P, a set of training data D

```
Let π_P be the set of parameters of P:
  weights and thresholds
Initialise all parameters π_P, randomly
repeat
    Compute error E wrt. D and current parameters π_P
    Modify parameters π_P such that the error decreases
      But, how can we modify the parameters?
      By going down-hill
until E is acceptable
return The modified perceptron P
```

Training a Perceptron: The Delta Rule

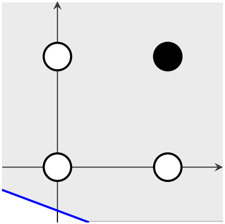
Delta Rule

Input: A set P of n-dimensional positive examples and a set N of n-dimensional negative examples

Result: n-dim. weights $\vec{w}$ and a threshold t, such that $\vec{w} \cdot \vec{p} \geq t$ for all $\vec{p} \in P$ and $\vec{w} \cdot \vec{n} < t$ for all $\vec{n} \in N$

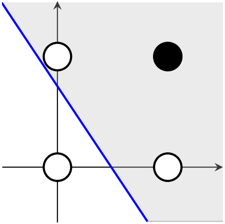
Initialise the weight vector $\vec{w}_0$ randomly
repeat
    select a sample $\vec{x} \in P \cup N$ randomly
    if $\vec{x} \in P$ **then**
        if $\vec{w} \cdot \vec{x} < t$ **then** set $\vec{w} := \vec{w} + \vec{x}$ and $t := t - 1$
    else
        if $\vec{w} \cdot \vec{x} \geq t$ **then** set $\vec{w} := \vec{w} - \vec{x}$ and $t := t + 1$
    end
until E *is acceptable or maximal number of iterations*
return *The modified perceptron, i.e.,* $\vec{w}$ *and* t

Training a Perceptron: The Delta Rule (visualisation)



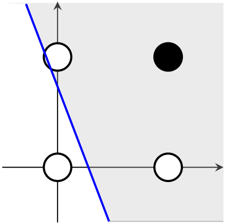
t	w_1	w_2
-0.35	0.33	0.89

Training a Perceptron: The Delta Rule (visualisation)



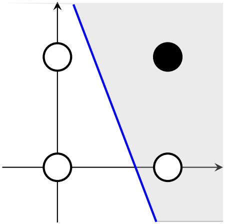
t	w_1	w_2
-0.35	0.33	0.89
0.65	1.33	0.89

Training a Perceptron: The Delta Rule (visualisation)



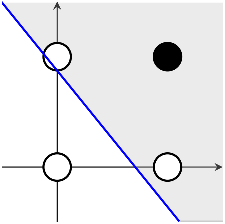
t	w_1	w_2
-0.35	0.33	0.89
0.65	1.33	0.89
0.65	2.33	0.89

Training a Perceptron: The Delta Rule (visualisation)



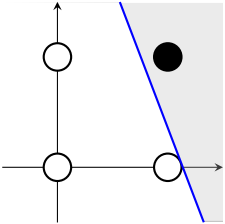
t	w_1	w_2
-0.35	0.33	0.89
0.65	1.33	0.89
0.65	2.33	0.89
1.65	2.33	0.89

Training a Perceptron: The Delta Rule (visualisation)



t	w_1	w_2
-0.35	0.33	0.89
0.65	1.33	0.89
0.65	2.33	0.89
1.65	2.33	0.89
1.65	2.33	1.89

Training a Perceptron: The Delta Rule (visualisation)



t	w_1	w_2
-0.35	0.33	0.89
0.65	1.33	0.89
0.65	2.33	0.89
1.65	2.33	0.89
1.65	2.33	1.89
2.65	2.33	0.89

Supervised Learning

recap

Based on given data, a *supervised learning algorithm* has to learn a mapping from inputs to correct outputs.

- The training data consists of pairs: $(\text{input}, \text{desired output})$
- Let a set D of n input-output data-pairs be given:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

  with y_i being generated by some unknown function f, i.e., $y_i = f(x_i) + \mathit{noise}$.
- Then, a supervised learning procedure determines a function h from the set of possible hypotheses $h \in \mathcal{H}$ which performs well on D.

Perceptron: Hypothesis Space

- What is the hypothesis space $\mathcal{H}$ of a perceptron?

Perceptron: Hypothesis Space

- What is the hypothesis space $\mathcal{H}$ of a perceptron?
- The perceptron implements a linear decision!

Perceptron: Hypothesis Space

- What is the hypothesis space $\mathcal{H}$ of a perceptron?
- The perceptron implements a linear decision!
- Therefore, only linearly separable classification tasks can be solved

Perceptron: Hypothesis Space

- What is the hypothesis space $\mathcal{H}$ of a perceptron?
- The perceptron implements a linear decision!
- Therefore, only linearly separable classification tasks can be solved
- I.e., the hypothesis space $\mathcal{H}$ of a perceptron with n inputs, is the set of all linear separable binary classification functions over n inputs

Perceptron: Hypothesis Space

- What is the hypothesis space $\mathcal{H}$ of a perceptron?
- The perceptron implements a linear decision!
- Therefore, only linearly separable classification tasks can be solved
- I.e., the hypothesis space $\mathcal{H}$ of a perceptron with n inputs, is the set of all linear separable binary classification functions over n inputs

- Will the delta-rule always find a solution?

Perceptron: Hypothesis Space

- What is the hypothesis space $\mathcal{H}$ of a perceptron?
- The perceptron implements a linear decision!
- Therefore, only linearly separable classification tasks can be solved
- I.e., the hypothesis space $\mathcal{H}$ of a perceptron with n inputs, is the set of all linear separable binary classification functions over n inputs

- Will the delta-rule always find a solution?
- If yes, will it always find the same solution?

Delta-Rule Convergence Theorem

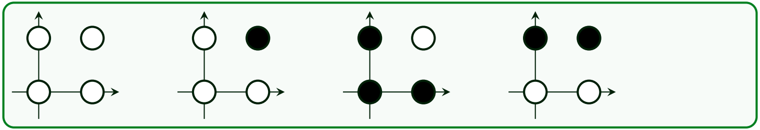
Theorem (Delta-Rule Convergence Theorem)

If there is a weight vector $\vec{w}^*$ such that $\Theta(\vec{w}^* \cdot \vec{x} - t) = y$ for all $(\vec{x}, y)$, (i.e., the problem is linearly separable) then for any starting vector $\vec{w}$, the delta rule will converge to an updated weight vector $\vec{w}$ (not necessarily unique and not necessarily $\vec{w}^*$) that gives the correct response for all training patterns, and it will do so in a finite number of steps.

`www.cs.ubbcluj.ro/~csatol/kozgaz_mestint/4_neuronhalo/PerceptConvProof.pdf`

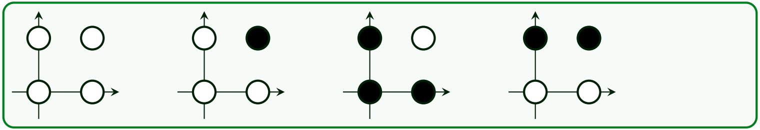
Perceptron: Some linearly (in)separable problems

- Linear separable problems in 2d (4 out of 14 in total):



Perceptron: Some linearly (in)separable problems

- Linear separable problems in 2d (4 out of 14 in total):



- Linear inseparable problems in 2d:

Perceptron: Number of linearly (in)separable problems

- For two inputs, there are only 2 inseparable binary problems, but 14 separable ones, i.e., no problem?

Perceptron: Number of linearly (in)separable problems

- For two inputs, there are only 2 inseparable binary problems, but 14 separable ones, i.e., no problem?
- Unfortunately in higher dimensions there is a problem:

# var	# fns	# Lin.-sep.functions
2	16	14
3	256	104
4	65536	1882
5	4.2910^{10}	$9.45 \cdot 10^5$
6	1.8410^{19}	$1.50 \cdot 10^8$
7	3.4010^{38}	$8.37 \cdot 10^{10}$
8	1.1510^{77}	$1.75 \cdot 10^{14}$
9	1.3410^{154}	$1.44 \cdot 10^{18}$

N. Gruzling. *Linear separability of the vertices of an n-dimensional hypercube* (2007)

Summary

- A single perceptron can be trained ...
  - using the delta rule
  - to compute any linearly separable boolean function
    (because they compute a linear decision)
- Networks of perceptrons ...
  - can compute any boolean function
  - can not be trained using the delta rule!
    It is not applicable for non-output units, because the target mapping of the hidden units is unknown.

Take-away-messages of section: *“Training Perceptrons”*

E You should now be able to ...

- explain how to train a single layer perceptron

Take-away-messages of section: *"Training Perceptrons"*

E You should now be able to ...

- explain how to train a single layer perceptron
- describe, compute and draw the error surface of a single threshold unit

Take-away-messages of section: *"Training Perceptrons"*

E You should now be able to ...

- explain how to train a single layer perceptron
- describe, compute and draw the error surface of a single threshold unit
- describe the ideas behind the *Delta Rule*

Take-away-messages of section: *"Training Perceptrons"*

E You should now be able to ...

- explain how to train a single layer perceptron
- describe, compute and draw the error surface of a single threshold unit
- describe the ideas behind the *Delta Rule*
- explain the concept of a hypothesis space by example for a perceptron

Take-away-messages of section: *"Training Perceptrons"*

E You should now be able to ...

- explain how to train a single layer perceptron
- describe, compute and draw the error surface of a single threshold unit
- describe the ideas behind the *Delta Rule*
- explain the concept of a hypothesis space by example for a perceptron
- know what caused the first AI winter