Chapter ML:IV

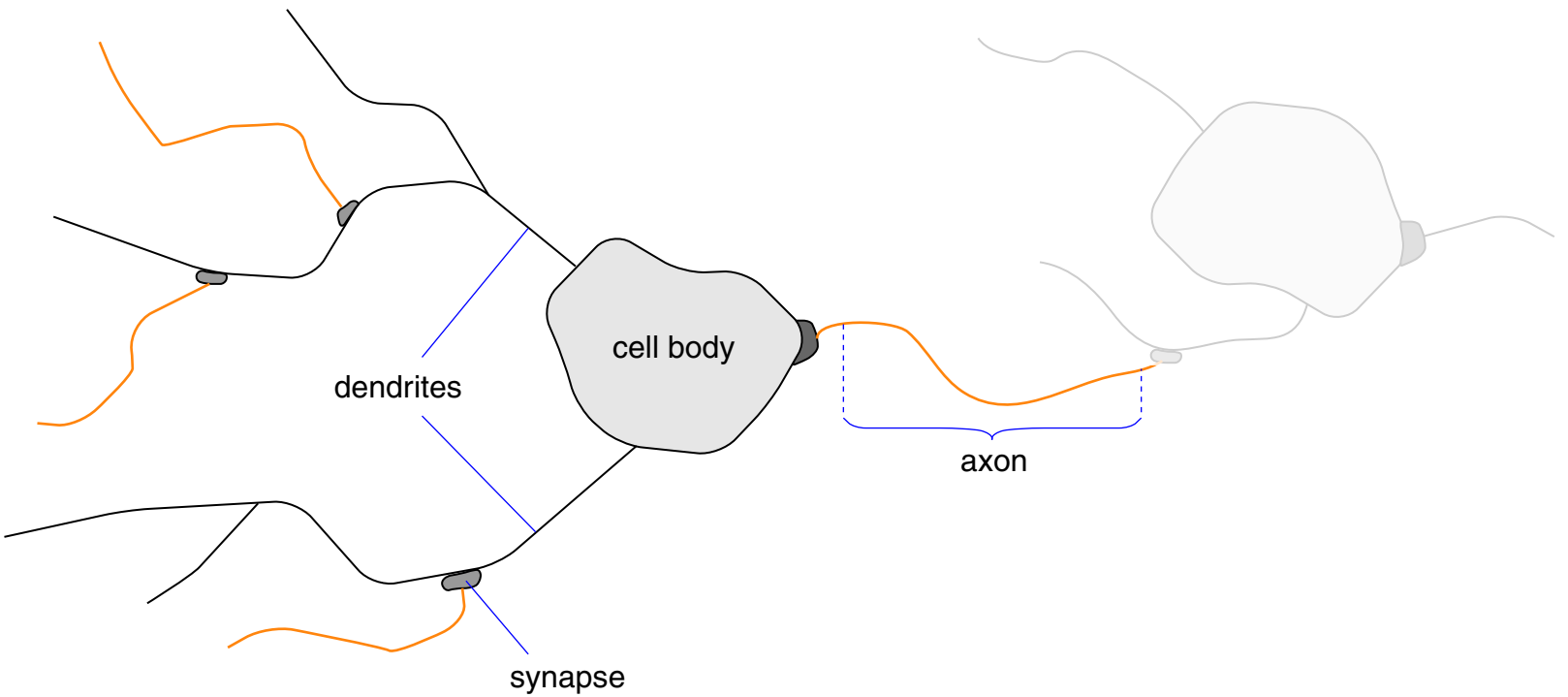
IV. Neural Networks

- Perceptron Learning
- Multilayer Perceptron Basics
- Multilayer Perceptron with Two Layers
- Multilayer Perceptron at Arbitrary Depth
- Advanced MLPs
- Automatic Gradient Computation

Perceptron Learning

Biological Inspiration

A *very* simplified model of a neuron:



Perceptron Learning

Biological Inspiration (continued)

Neuron characteristics:

- The numerous *dendrites* of a neuron serve as its *input* channels for electrical signals.
- At particular contact points, the so-called synapses, electrical signals can be initiated.
- A synapse can initiate signals of different strengths, where the strength is encoded by the frequency of a “pulse train”. Keyword: frequency modulation
- The cell body of a neuron accumulates the incoming signals.
- If a particular stimulus threshold is exceeded, the cell body generates a signal, which is *output* via the *axon*.
- The processing of the signals is unidirectional. (from left to right in the figure)

Remarks (facts about the human brain):

- The human brain is comprised of about 10^{11} neurons.
- The length of a dendrite or an axon is about 100 micron. A micron = 10^{-6} m = 10^{-3} mm.
- The dendrites of a neuron are connected to 100 000 – 200 000 other neurons.
- An axon is connected to up to 10 000 synapses of other neurons.
- The human brain contains about 10^{12} synapses.
- The switching of a neuron is not faster than 10^{-3} s, which is rather slow compared to the 10^{-12} s for electrical circuits. I.e., within the typical human reaction time of about 10^{-1} s only a few hundred neuronal activities can take place.

Perceptron Learning

History

1943 Warren McCulloch and Walter Pitts present a model of the neuron.

1949 Donald Hebb postulates a new learning paradigm: reinforcement only for active neurons. (those neurons involved in a decision process)

1958 Frank Rosenblatt develops the perceptron model. (see next slide)

1962 Rosenblatt proves the perceptron convergence theorem.

1969 Marvin Minsky and Seymour Papert publish a book on the limitations of the perceptron model.

1970

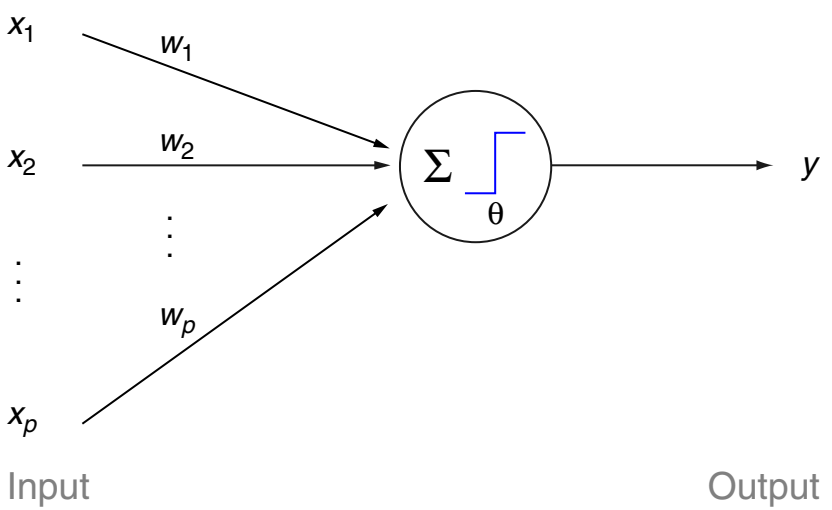
⋮ ANN research paused.

1985

1986 David Rumelhart and James McClelland present the multilayer perceptron.

Perceptron Learning

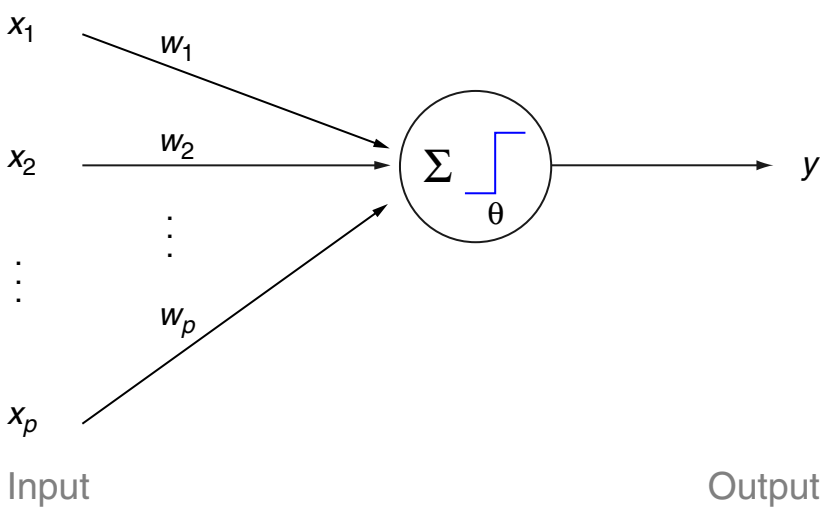
The Perceptron of Rosenblatt (1958)



$$x_j, w_j \in \mathbf{R}, \quad j = 1, \ldots, p$$

Perceptron Learning

The Perceptron of Rosenblatt (1958) (continued)

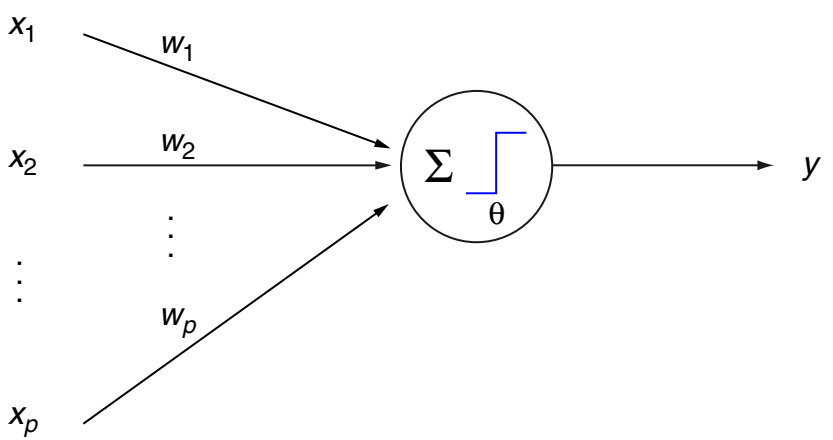


$$x_j, w_j \in \mathbf{R}, \quad j = 1, \ldots, p$$

- Rosenblatt's perceptron relies on the neuron model of McCulloch/Pitts [1943].
- The weights w_j model the reinforcement factor.
- The threshold function models a decision rule.
- The perceptron is a "feed forward system".

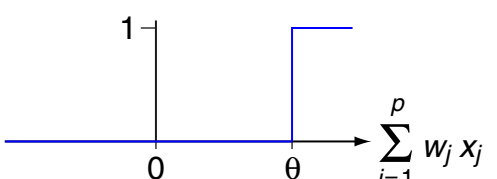
Perceptron Learning

The Perceptron of Rosenblatt (1958) (continued)



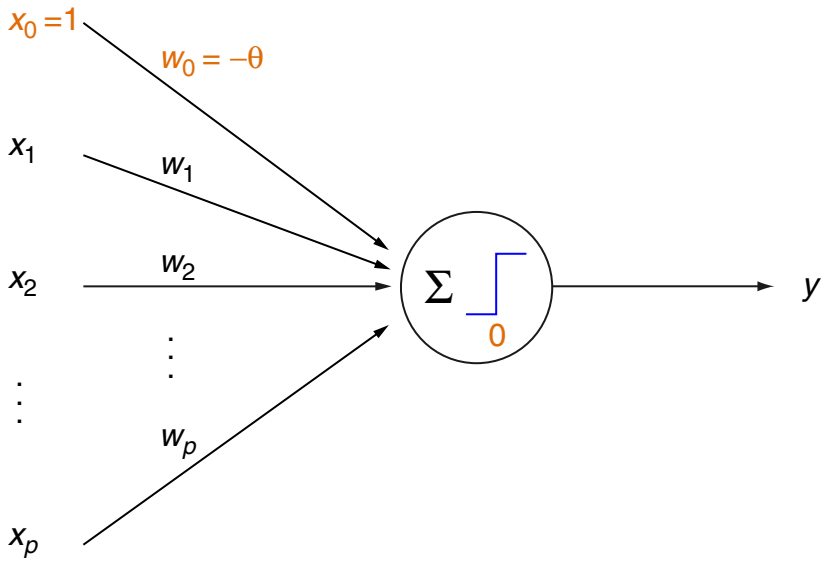
$$x_j, w_j \in \mathbf{R}, \quad j = 1, \ldots, p$$

$$y(\mathbf{x}) = \begin{cases} 1, & \text{if} \quad \sum_{j=1}^{p} w_j x_j \geq \theta \\ 0, & \text{if} \quad \sum_{j=1}^{p} w_j x_j < \theta \end{cases}$$



Perceptron Learning

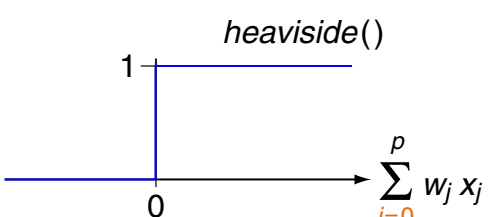
The Perceptron of Rosenblatt (1958) (continued)



$$x_j, w_j \in \mathbf{R}, \quad j = 0, \ldots, p$$

$$y(\mathbf{x}) = \textit{heaviside}\Big(\sum_{j=1}^{p} w_j x_j - \theta\Big)$$

$$= \textit{heaviside}\Big(\sum_{j=0}^{p} w_j x_j\Big), \quad \text{with } w_0 = -\theta,\ x_0 = 1$$



Remarks:

- By extending both the weight vector by $w_0 = -\theta$ and the feature vectors by the constant feature $x_0 = 1$, the learning algorithm gets a canonical form.
- A single hypothesis is determined by $(w_0, w_1, \ldots, w_p)$.
- Since the range of the model function $y(\mathbf{x})$ is $\{0, 1\}$, the *heaviside* function is used for discretization (= class assignment). In section From Regression to Classification of part Machine Learning Basics the range of the model function is $\{-1, 1\}$ and the sign function does the discretization, which in principle is no difference.
- The *heaviside* function is named after the mathematician Oliver Heaviside.
  [Wikipedia: step function, Oliver Heaviside]
- The threshold function is also called “activation function”.

Perceptron Learning

Binary Classification Problems

Setting:

- X is a multiset of feature vectors from an inner product space $\mathbf{X}$, $\mathbf{X} \subseteq \mathbf{R}^p$.
- $C = \{0, 1\}$ is a set of two classes. Similarly: $\{-1, 1\}$, $\{\ominus, \oplus\}$, $\{\text{no}, \text{yes}\}$, etc.
- $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_n, c_n)\} \subseteq X \times C$ is a multiset of examples.

Learning task:

- Fit D using a perceptron $y()$.

Perceptron Learning

The PT Algorithm

[algorithms: LMS BGD_σ PT BGD IGD]

Algorithm:	PT	Perceptron Training
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{0, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

PT(D, η)

1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$
4. $(\mathbf{x}, c) =$ *random_select*(D)
5. $y(\mathbf{x}) \stackrel{(\star)}{=}$ *heaviside*$(\mathbf{w}^T \mathbf{x})$
6. $\delta = c - y(\mathbf{x})$ // $y(\mathbf{x}) \in \{0, 1\}$, $c \in \{0, 1\} \rightsquigarrow \delta \in \{0, 1, -1\}$
7. $\Delta\mathbf{w} \stackrel{(\star)}{=} \eta \cdot \delta \cdot \mathbf{x}$
8. $\mathbf{w} = \mathbf{w} + \Delta\mathbf{w}$
9. **UNTIL**(*convergence*$(D, y(), t)$)
10. *return*$(\mathbf{w})$

Perceptron Learning

The PT Algorithm

[algorithms: LMS BGD_σ PT BGD IGD]

Algorithm:	PT	Perceptron Training
Input:	D	Multiset of examples $(\mathbf{x}, c)$ with $\mathbf{x} \in \mathbf{R}^p$, $c \in \{0, 1\}$.
	η	Learning rate, a small positive constant.
Output:	$\mathbf{w}$	Weight vector from $\mathbf{R}^{p+1}$. (= hypothesis)

PT(D, η)

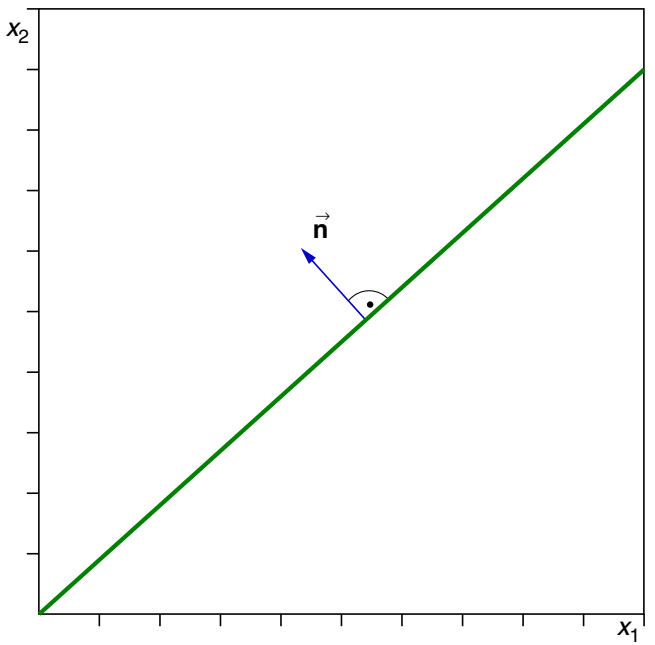
1. *initialize_random_weights*$(\mathbf{w})$, $t = 0$
2. **REPEAT**
3. $t = t + 1$
4. $(\mathbf{x}, c) =$ *random_select*(D)
5. `Model function evaluation.`
6. `Calculation of indicator for true/false hyperplane side.`
7. `Calculation of weight correction.`
8. `Parameter vector update.`
9. **UNTIL**(*convergence*$(D, y(), t)$)
10. *return*$(\mathbf{w})$

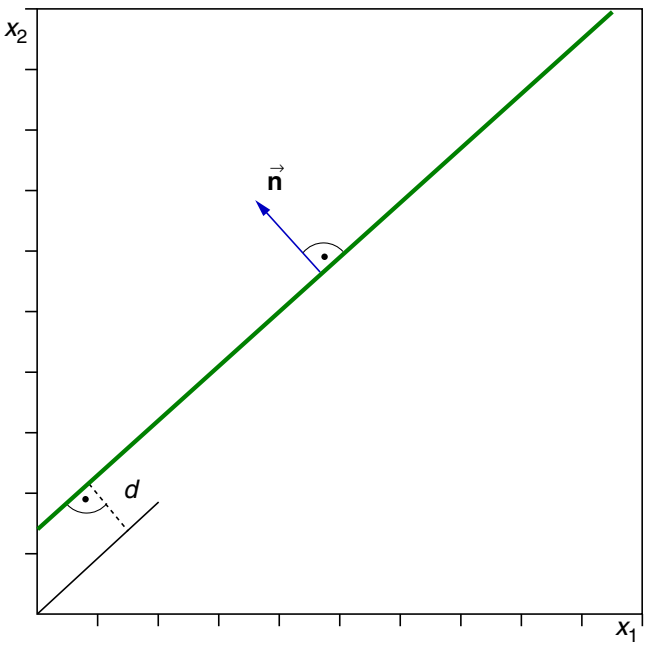
Remarks:

- $(\star)$ Recap. We consider the feature vector $\mathbf{x}$ in its extended form when used as operand in a scalar product with the weight vector, $\mathbf{w}^T\mathbf{x}$, and consequently, when noted as argument of the model function, $y(\mathbf{x})$. I.e., $\mathbf{x} = (1, x_1, \ldots, x_p)^T \in \mathbf{R}^{p+1}$, and $x_0 = 1$.
- The variable t denotes the time. The learning algorithm gets an example presented at each point in time and may adapt the weight vector.
- For an example $(\mathbf{x}, c) \in D$ the weight adaptation rule compares the target class c (the ground truth) to the class computed by the perceptron. In case of a wrong classification of $\mathbf{x}$, δ is either -1 or $+1$, regardless of the exact numeric difference between c and $\mathbf{w}^T\mathbf{x}$.
- Recap. The function *convergence*() checks for misclassified examples (= analyzes the $0/1$ loss). Consider in this regard the vectors of observed and computed classes, $D|_c$ and $y(D|_{\mathbf{x}})$ respectively. In addition, the function may check via t an upper bound on the number of iterations.

Perceptron Learning

Weight Adaptation: Illustration in Input Space



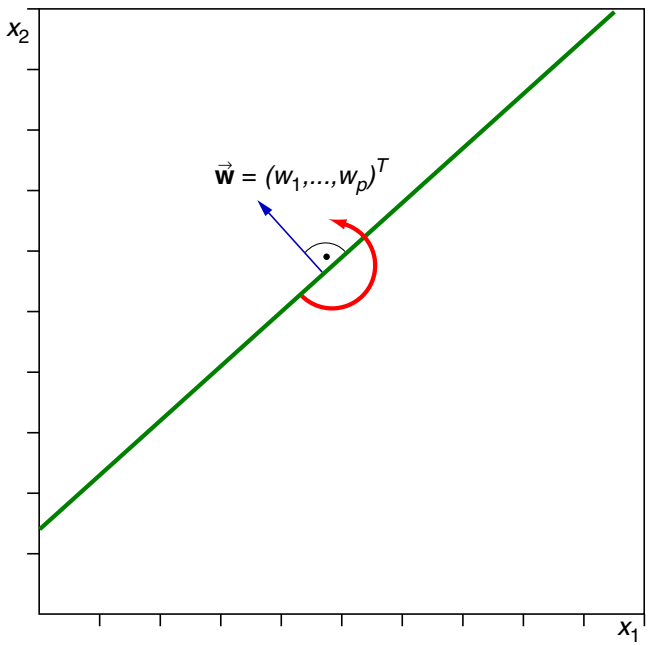


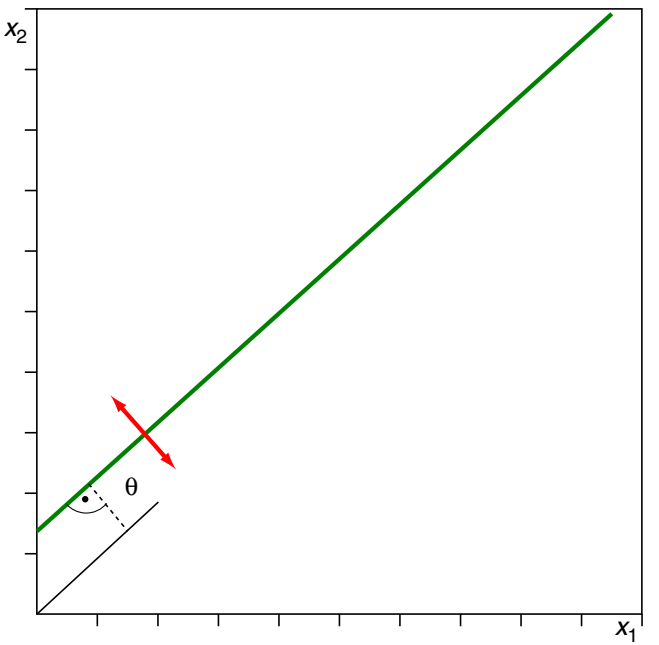
Definition of an (affine) hyperplane: $N = \left\{ \mathbf{x} \mid \vec{\mathbf{n}}^T \mathbf{x} = d \right\}$ [Wikipedia]

- $\vec{\mathbf{n}}$ is a normal vector of (a vector perpendicular to) the hyperplane N.
- If $||\vec{\mathbf{n}}|| = 1$ then $|\vec{\mathbf{n}}^T \mathbf{x} - d|$ gives the (geometric) distance of a point $\mathbf{x}$ to N.
- If $\text{sign}(\vec{\mathbf{n}}^T \mathbf{x_1} - d) = \text{sign}(\vec{\mathbf{n}}^T \mathbf{x_2} - d)$, then $\mathbf{x_1}$ and $\mathbf{x_2}$ are on the same side of the hyperplane.

Perceptron Learning

Weight Adaptation: Illustration in Input Space (continued)





Definition of an (affine) hyperplane: $\mathbf{w}^T\mathbf{x} = 0 \Leftrightarrow \sum_{j=1}^{p} w_j x_j = \theta = -w_0$

Hyperplane definition as before, with notation taken from the classification problem setting.

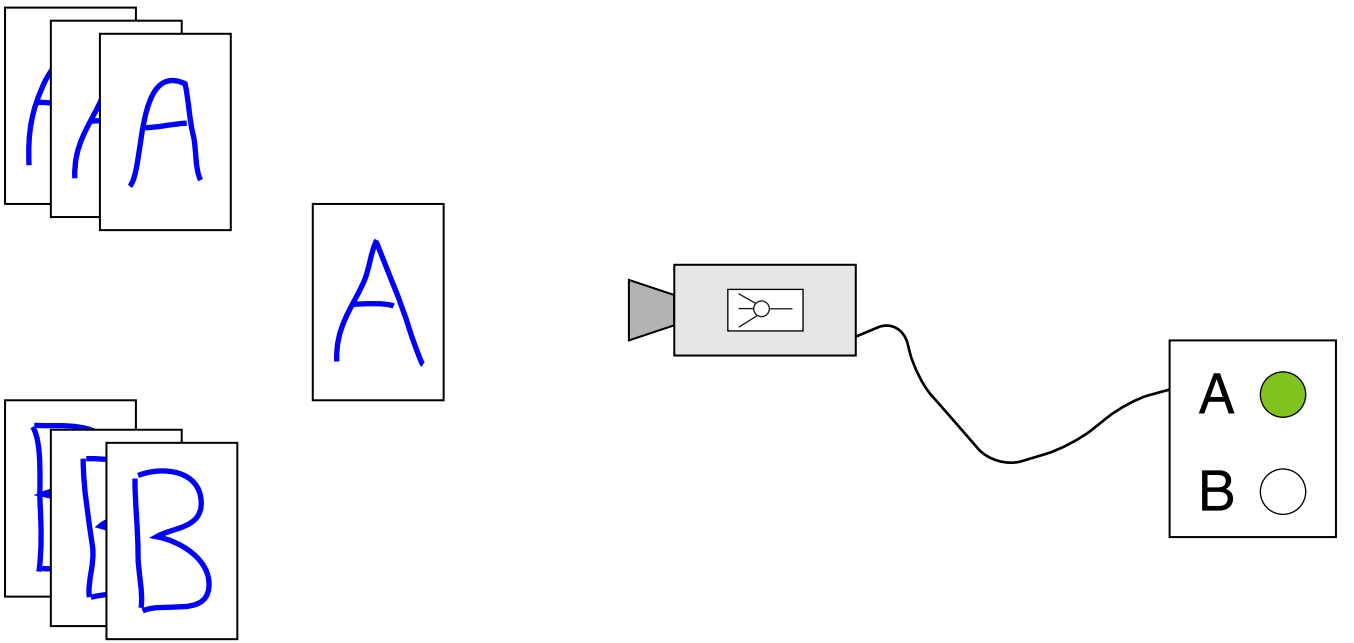
Remarks:

- Recap. Distinguish between the p-dimensional direction vector $\vec{\mathbf{w}} = (w_1, \ldots, w_p)^T$, and the $(p+1)$-dimensional hypothesis $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$.
- A perceptron defines a hyperplane that is perpendicular (= normal) to $\vec{\mathbf{w}} = (w_1, \ldots, w_p)^T$.
- θ or $-w_0$ defines the offset of the hyperplane from the origin, along $\vec{\mathbf{w}}$ and as multiple of $1/||\vec{\mathbf{w}}||$.
- The set of possible weight vectors $\mathbf{w} = (w_0, w_1, \ldots, w_p)^T$, $\mathbf{w} \in \mathbf{R}^{p+1}$, forms the hypothesis space H.
- Weight adaptation means learning, and the shown learning paradigm is supervised. Consider in this regard the lines 6–7 of the PT algorithm: If some x_j is zero, Δw_j will be zero as well. Keyword: Hebbian learning [Wikipedia: Hebbian theory, Donald Hebb]
- Note that here (and in the following illustrations) the hyperplane movement is not the result of solving a regression problem in the $(p+1)$-dimensional input-output-space, where the sum of the residuals is to be minimized.

  Rather, the PT algorithm takes each missclassified example $\mathbf{x}$ as an event to update the hyperplane's normal vector by a fixed amount that is proportional to $\mathbf{x}$. In particular, the update, $\Delta\mathbf{w}$, does not exploit the residual associated with $\mathbf{x}$ at time t, i.e., the absolute value of the distance of $\mathbf{x}$ from the hyperplane is disregarded.

Perceptron Learning

Example



- The examples are presented to the perceptron.
- The perceptron computes a value that is interpreted as class label.

Perceptron Learning

Example (continued)

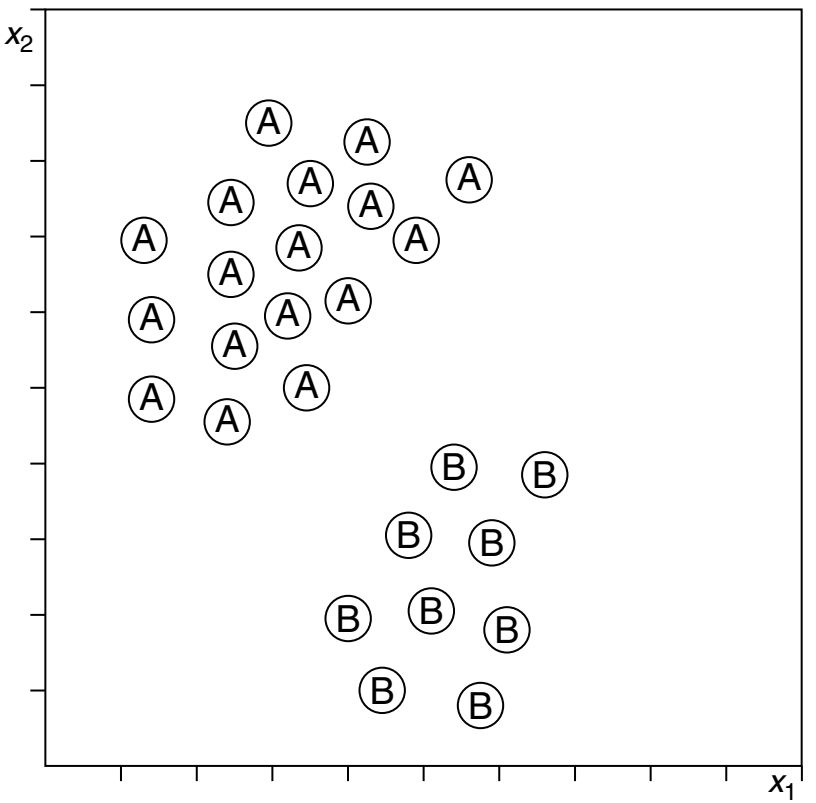
Encoding:

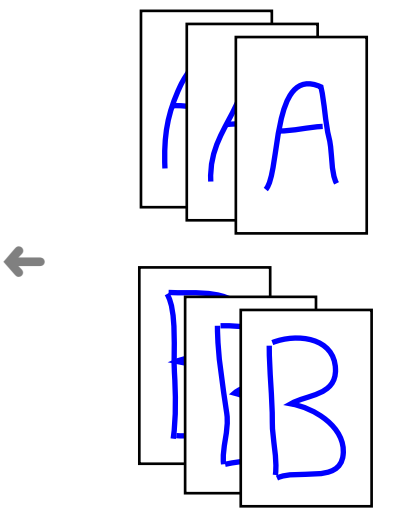
- The encoding of the examples is based on features such as the number of line crossings, most acute angle, longest line, etc.
- The class label, c, is encoded as a number: examples from A are labeled with 1, examples from B are labeled with 0.

$$\underbrace{\begin{pmatrix} x_{1_1} \\ x_{1_2} \\ \vdots \\ x_{1_p} \end{pmatrix} \cdots \begin{pmatrix} x_{k_1} \\ x_{k_2} \\ \vdots \\ x_{k_p} \end{pmatrix}}_{\text{Class } A \;\widehat{=}\; c = 1} \qquad \underbrace{\begin{pmatrix} x_{l_1} \\ x_{l_2} \\ \vdots \\ x_{l_p} \end{pmatrix} \cdots \begin{pmatrix} x_{m_1} \\ x_{m_2} \\ \vdots \\ x_{m_p} \end{pmatrix}}_{\text{Class } B \;\widehat{=}\; c = 0}$$

Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]

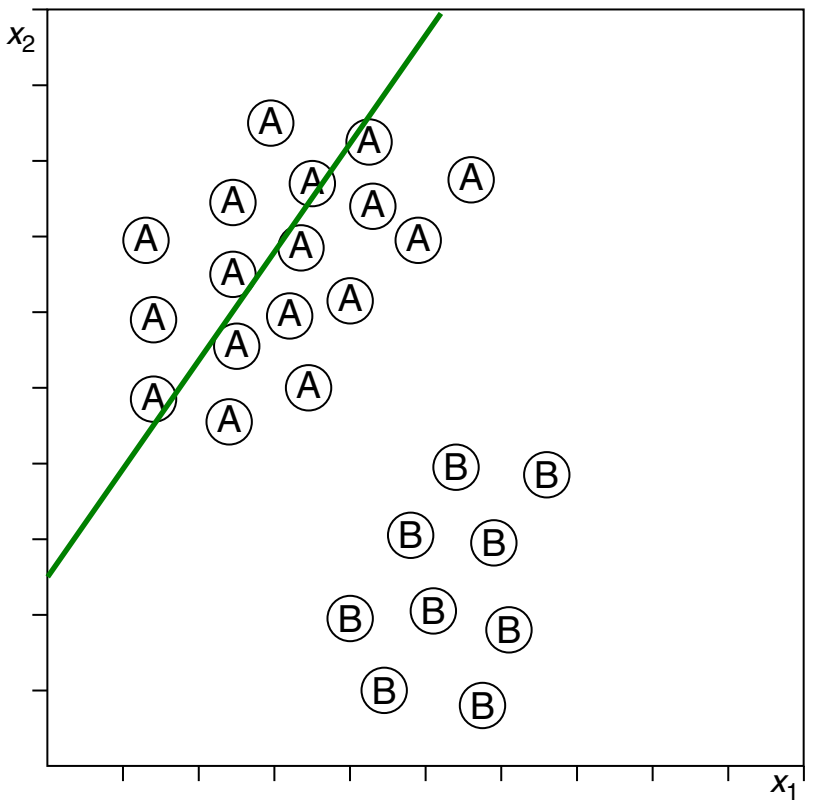




A possible configuration of encoded objects (feature vectors) in the input space $\mathbf{X}$.

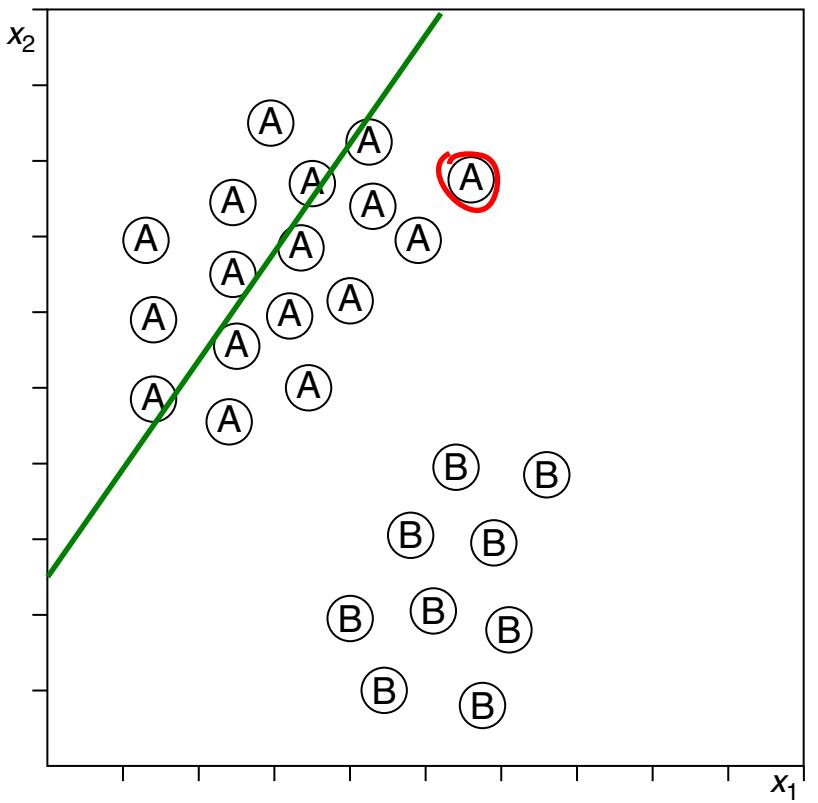
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



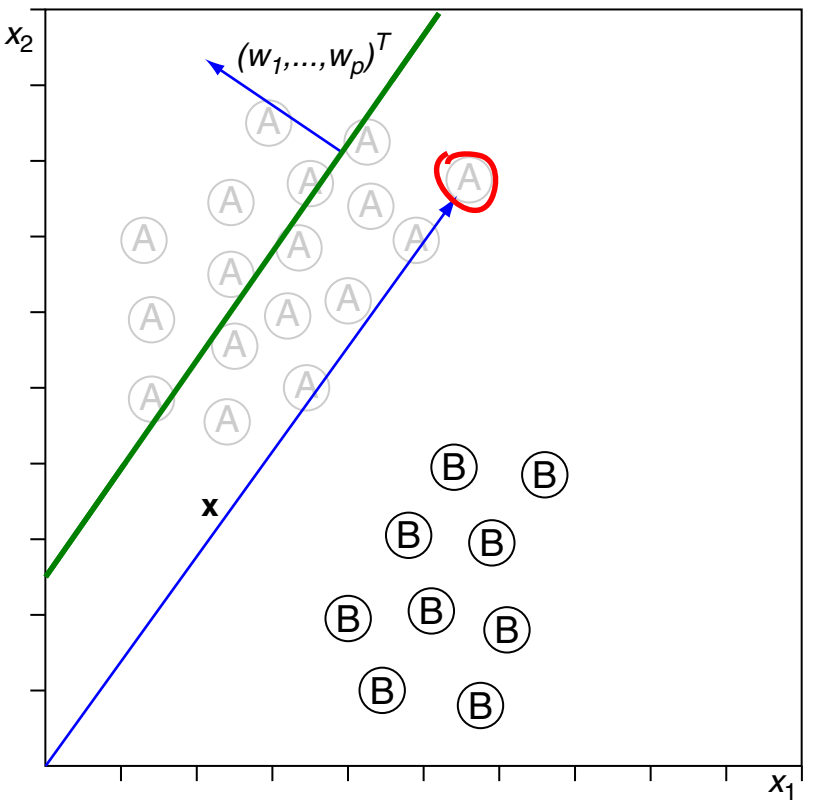
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



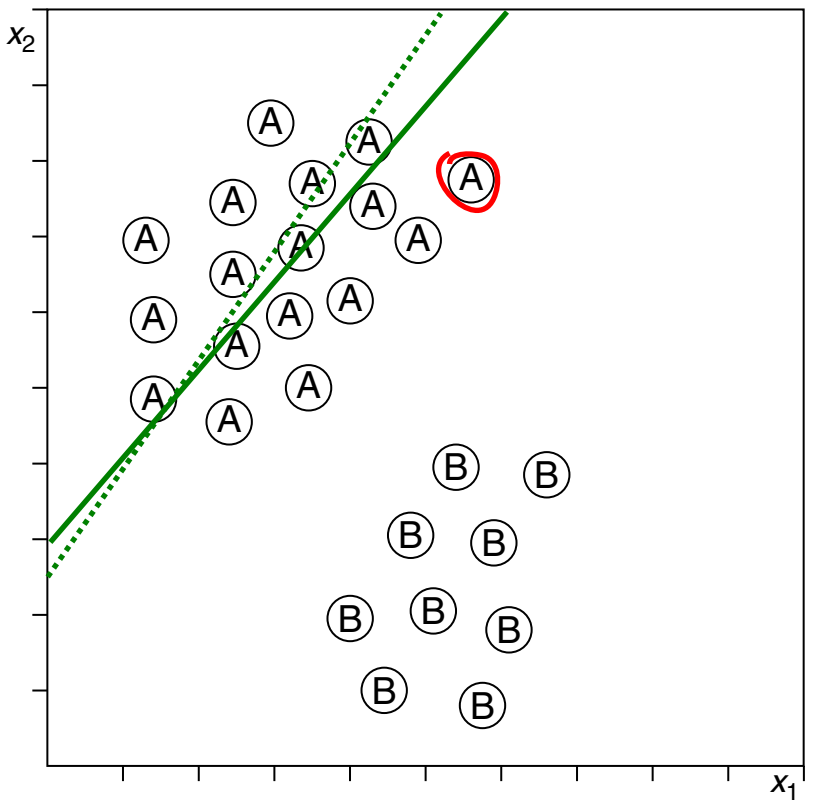
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



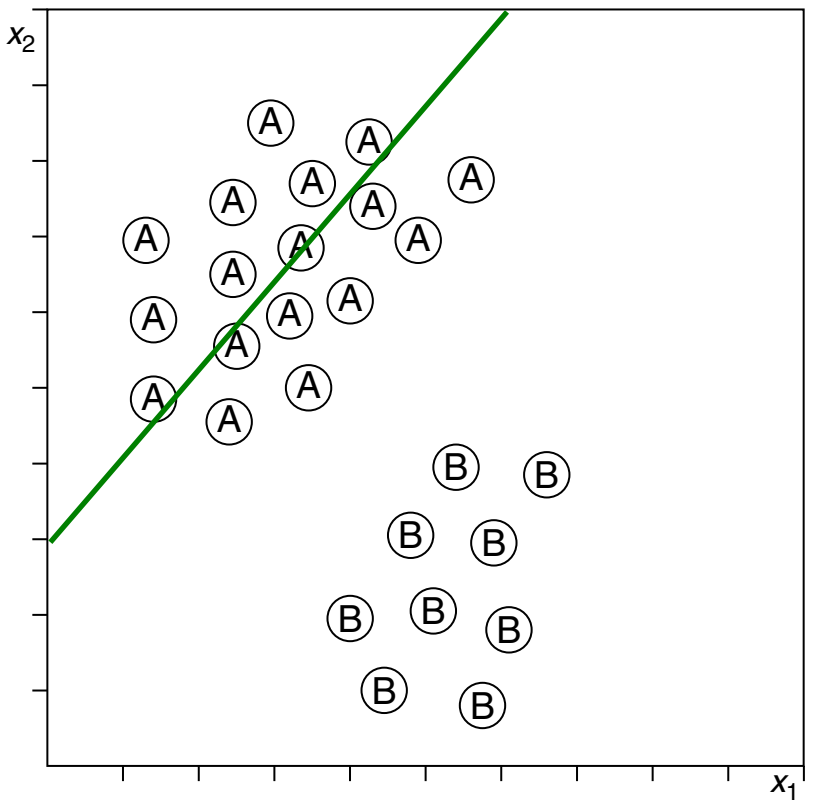
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



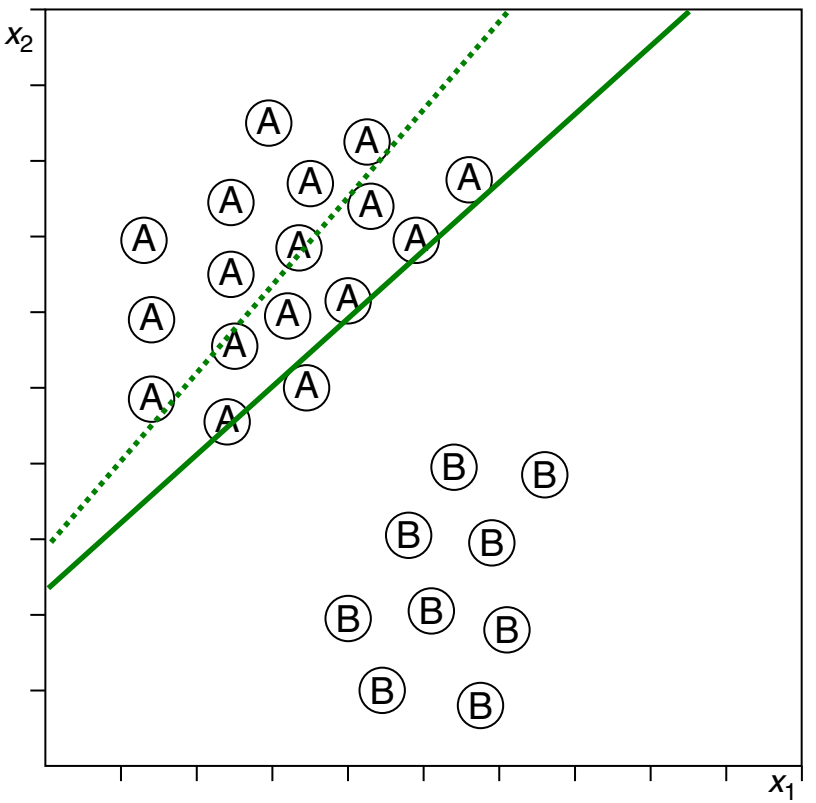
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



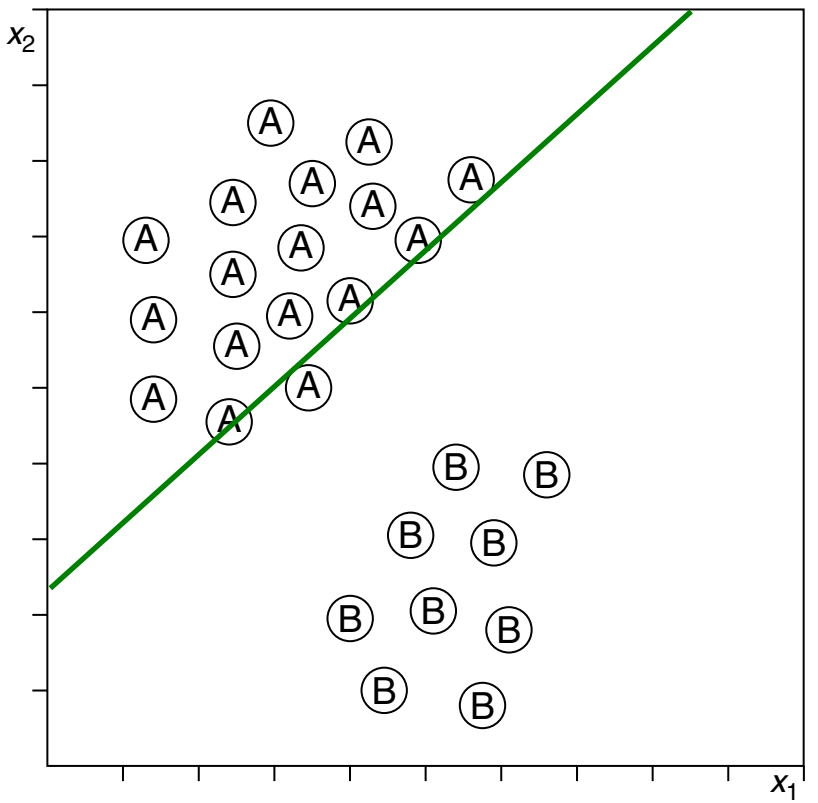
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



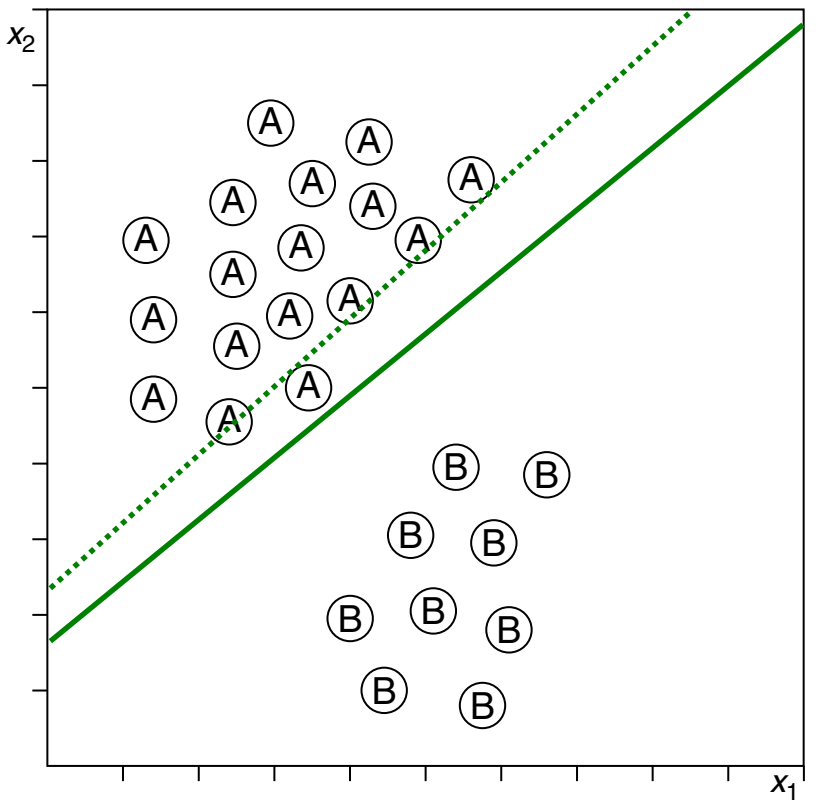
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



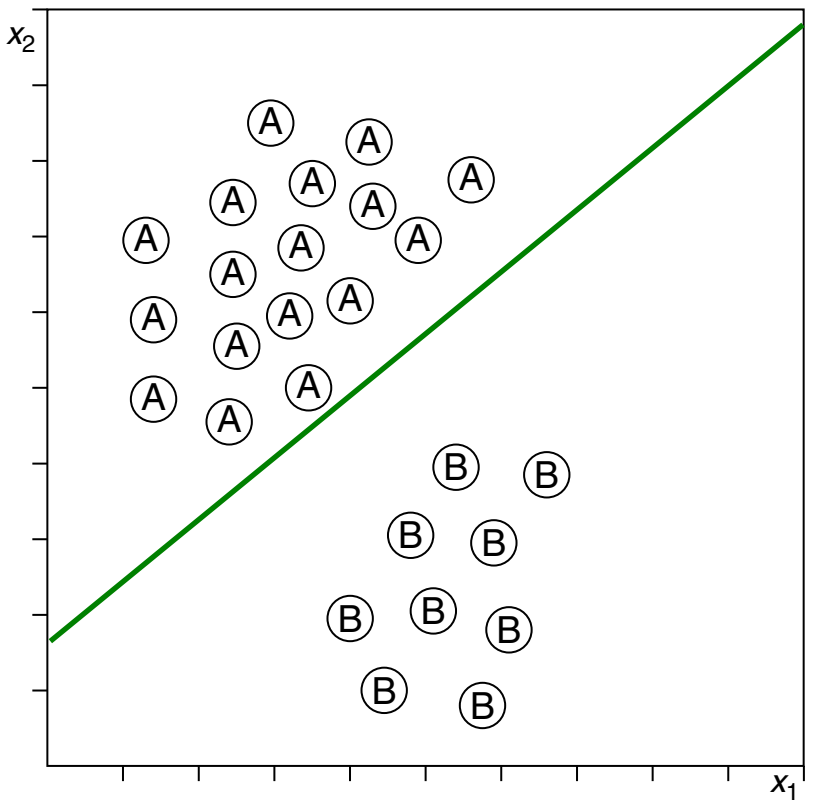
Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



Perceptron Learning

Example: Illustration in Input Space (continued) [PT algorithm]



Perceptron Learning

Perceptron Convergence Theorem [discussion]

Questions:

1. Which kind of learning tasks can be addressed with the functions in the hypothesis space H?
2. Can the PT algorithm construct such a function for a given task?

Perceptron Learning

Perceptron Convergence Theorem (continued) [discussion]

Questions:

1. Which kind of learning tasks can be addressed with the functions in the hypothesis space H?
2. Can the PT algorithm construct such a function for a given task?

Theorem 1 (Perceptron Convergence [Rosenblatt 1962]**)**

Let X_0 and X_1 be two finite sets with vectors of the form $\mathbf{x} = (1, x_1, \ldots, x_p)^T$, let $X_1 \cap X_0 = \emptyset$, and let $\widehat{\mathbf{w}}$ define a separating hyperplane with respect to X_0 and X_1. Moreover, let D be a set of examples of the form $(\mathbf{x}, 0)$, $\mathbf{x} \in X_0$ and $(\mathbf{x}, 1)$, $\mathbf{x} \in X_1$.

Then holds:

If the examples in D are processed with the PT algorithm, the constructed weight vector $\mathbf{w}$ will converge within a finite number of iterations.

Perceptron Learning

Perceptron Convergence Theorem: Proof

Preliminaries:

- The sets X_1 and X_0 are separated by a hyperplane $\widehat{\mathbf{w}}$. The proof requires that for all $\mathbf{x} \in X_1$ the inequality $\widehat{\mathbf{w}}^T\mathbf{x} > 0$ holds. This condition is always fulfilled, as the following consideration shows.
  Let $\mathbf{x}' \in X_1$ with $\widehat{\mathbf{w}}^T\mathbf{x}' = 0$. Since X_0 is finite, the members $\mathbf{x} \in X_0$ have a minimum positive distance δ with regard to the hyperplane $\widehat{\mathbf{w}}$. Hence, $\widehat{\mathbf{w}}$ can be moved by $\frac{\delta}{2}$ towards X_0, resulting in a new hyperplane $\widehat{\mathbf{w}}'$ that still fulfills $(\widehat{\mathbf{w}}')^T\mathbf{x} < 0$ for all $\mathbf{x} \in X_0$, but that now also fulfills $(\widehat{\mathbf{w}}')^T\mathbf{x} > 0$ for all $\mathbf{x} \in X_1$.

- By defining $X' = X_1 \cup \{-\mathbf{x} \mid \mathbf{x} \in X_0\}$, the searched $\mathbf{w}$ fulfills $\mathbf{w}^T\mathbf{x} > 0$ for all $\mathbf{x} \in X'$. Then, with $c = 1$ for all $\mathbf{x} \in X'$, $\delta \in \{0, 1\}$ (instead of $\{0, 1, -1\}$). [PT algorithm, Line 5]

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

Preliminaries:

- The sets X_1 and X_0 are separated by a hyperplane $\widehat{\mathbf{w}}$. The proof requires that for all $\mathbf{x} \in X_1$ the inequality $\widehat{\mathbf{w}}^T\mathbf{x} > 0$ holds. This condition is always fulfilled, as the following consideration shows.
  Let $\mathbf{x}' \in X_1$ with $\widehat{\mathbf{w}}^T\mathbf{x}' = 0$. Since X_0 is finite, the members $\mathbf{x} \in X_0$ have a minimum positive distance δ with regard to the hyperplane $\widehat{\mathbf{w}}$. Hence, $\widehat{\mathbf{w}}$ can be moved by $\frac{\delta}{2}$ towards X_0, resulting in a new hyperplane $\widehat{\mathbf{w}}'$ that still fulfills $(\widehat{\mathbf{w}}')^T\mathbf{x} < 0$ for all $\mathbf{x} \in X_0$, but that now also fulfills $(\widehat{\mathbf{w}}')^T\mathbf{x} > 0$ for all $\mathbf{x} \in X_1$.

- By defining $X' = X_1 \cup \{-\mathbf{x} \mid \mathbf{x} \in X_0\}$, the searched $\mathbf{w}$ fulfills $\mathbf{w}^T\mathbf{x} > 0$ for all $\mathbf{x} \in X'$. Then, with $c = 1$ for all $\mathbf{x} \in X'$, $\delta \in \{0, 1\}$ (instead of $\{0, 1, -1\}$). [PT algorithm, Line 5]

- The PT Algorithm performs a number of iterations, where $\mathbf{w}(t)$ denotes the weight vector for iteration t, which form the basis for the weight vector $\mathbf{w}(t+1)$. $\mathbf{x}(t) \in X'$ denotes the feature vector chosen in round t. The first (and randomly chosen) weight vector is denoted as $\mathbf{w}(0)$.

- Recall the Cauchy-Schwarz inequality: $||\mathbf{a}||^2 \cdot ||\mathbf{b}||^2 \geq (\mathbf{a}^T\mathbf{b})^2$, where $||\mathbf{x}|| := \sqrt{\mathbf{x}^T\mathbf{x}}$ denotes the Euclidean norm.

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

Line of argument:

(a) We state a lower bound for how much $||\mathbf{w}||$ has changed from its initial value after n iterations. The derivation of this lower bound exploits the presupposed linear separability of X_0 and X_1.

(b) We state an upper bound for how much $||\mathbf{w}||$ can change from its initial value after n iterations. The derivation of this upper bound exploits the finiteness of X_0 and X_1, which in turn guarantees the existence of an upper bound for the norm of the maximum feature vector.

(c) We observe that the lower bound grows quadratically in n, whereas the upper bound grows linearly. To satisfy the relation "lower bound $<$ upper bound", n must be finite.

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

1. The PT algorithm computes in iteration t the scalar product $\mathbf{w}(t)^T\mathbf{x}(t)$. If classified correctly, $\mathbf{w}(t)^T\mathbf{x}(t) > 0$ and $\mathbf{w}$ is unchanged. Otherwise, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$ [Line 5-7].

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

1. The PT algorithm computes in iteration t the scalar product $\mathbf{w}(t)^T\mathbf{x}(t)$. If classified correctly, $\mathbf{w}(t)^T\mathbf{x}(t) > 0$ and $\mathbf{w}$ is unchanged. Otherwise, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$ [Line 5-7].

2. A sequence of n incorrectly classified feature vectors, $(x(t))$, along with the weight adaptation, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$, results in the series $\mathbf{w}(n)$:

$$\mathbf{w}(1) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0)$$
$$\mathbf{w}(2) = \mathbf{w}(1) + \eta \cdot \mathbf{x}(1) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \eta \cdot \mathbf{x}(1)$$
$$\vdots$$
$$\mathbf{w}(n) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \ldots + \eta \cdot \mathbf{x}(n-1)$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

1. The PT algorithm computes in iteration t the scalar product $\mathbf{w}(t)^T\mathbf{x}(t)$. If classified correctly, $\mathbf{w}(t)^T\mathbf{x}(t) > 0$ and $\mathbf{w}$ is unchanged. Otherwise, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$ [Line 5-7].

2. A sequence of n incorrectly classified feature vectors, $(x(t))$, along with the weight adaptation, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$, results in the series $\mathbf{w}(n)$:

$$\begin{aligned}
\mathbf{w}(1) &= \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) \\
\mathbf{w}(2) &= \mathbf{w}(1) + \eta \cdot \mathbf{x}(1) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \eta \cdot \mathbf{x}(1) \\
&\vdots \\
\mathbf{w}(n) &= \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \ldots + \eta \cdot \mathbf{x}(n-1)
\end{aligned}$$

3. The hyperplane defined by $\widehat{\mathbf{w}}$ separates X_1 and X_0: $\forall \mathbf{x} \in X' : \widehat{\mathbf{w}}^T\mathbf{x} > 0$
Let $\delta := \min_{\mathbf{x} \in X'} \widehat{\mathbf{w}}^T\mathbf{x}$. Recall from the preliminaries that $\delta > 0$ holds.

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

1. The PT algorithm computes in iteration t the scalar product $\mathbf{w}(t)^T\mathbf{x}(t)$. If classified correctly, $\mathbf{w}(t)^T\mathbf{x}(t) > 0$ and $\mathbf{w}$ is unchanged. Otherwise, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$ [Line 5-7].

2. A sequence of n incorrectly classified feature vectors, $(x(t))$, along with the weight adaptation, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$, results in the series $\mathbf{w}(n)$:

$$\begin{aligned}
&\mathbf{w}(1) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) \\
&\mathbf{w}(2) = \mathbf{w}(1) + \eta \cdot \mathbf{x}(1) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \eta \cdot \mathbf{x}(1) \\
&\quad\vdots \\
&\mathbf{w}(n) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \ldots + \eta \cdot \mathbf{x}(n-1)
\end{aligned}$$

3. The hyperplane defined by $\widehat{\mathbf{w}}$ separates X_1 and X_0: $\forall \mathbf{x} \in X' : \widehat{\mathbf{w}}^T\mathbf{x} > 0$
Let $\delta := \min_{\mathbf{x} \in X'} \widehat{\mathbf{w}}^T\mathbf{x}$. Recall from the preliminaries that $\delta > 0$ holds.

4. Analyze the scalar product of $\mathbf{w}(n)$ and $\widehat{\mathbf{w}}$:

$$\begin{aligned}
& \widehat{\mathbf{w}}^T\mathbf{w}(n) && = && \widehat{\mathbf{w}}^T\mathbf{w}(0) + \eta \cdot \widehat{\mathbf{w}}^T\mathbf{x}(0) + \ldots + \eta \cdot \widehat{\mathbf{w}}^T\mathbf{x}(n-1) \\
\Rightarrow\ & \widehat{\mathbf{w}}^T\mathbf{w}(n) && \geq && \widehat{\mathbf{w}}^T\mathbf{w}(0) + \eta \cdot n\delta \ \geq\ 0 \quad \text{// For } n \geq n_0 \text{ with sufficiently large } n_0 \in \mathbb{N}. \\
\Rightarrow\ & (\widehat{\mathbf{w}}^T\mathbf{w}(n))^2 && \geq && (\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2
\end{aligned}$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

1. The PT algorithm computes in iteration t the scalar product $\mathbf{w}(t)^T\mathbf{x}(t)$. If classified correctly, $\mathbf{w}(t)^T\mathbf{x}(t) > 0$ and $\mathbf{w}$ is unchanged. Otherwise, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$ [Line 5-7].

2. A sequence of n incorrectly classified feature vectors, $(x(t))$, along with the weight adaptation, $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$, results in the series $\mathbf{w}(n)$:

$$\begin{aligned}
\mathbf{w}(1) &= \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) \\
\mathbf{w}(2) &= \mathbf{w}(1) + \eta \cdot \mathbf{x}(1) = \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \eta \cdot \mathbf{x}(1) \\
&\vdots \\
\mathbf{w}(n) &= \mathbf{w}(0) + \eta \cdot \mathbf{x}(0) + \ldots + \eta \cdot \mathbf{x}(n-1)
\end{aligned}$$

3. The hyperplane defined by $\widehat{\mathbf{w}}$ separates X_1 and X_0 : $\forall \mathbf{x} \in X' : \widehat{\mathbf{w}}^T\mathbf{x} > 0$
Let $\delta := \min_{\mathbf{x} \in X'} \widehat{\mathbf{w}}^T\mathbf{x}$. Recall from the preliminaries that $\delta > 0$ holds.

4. Analyze the scalar product of $\mathbf{w}(n)$ and $\widehat{\mathbf{w}}$:

$$\begin{aligned}
& \widehat{\mathbf{w}}^T\mathbf{w}(n) && = && \widehat{\mathbf{w}}^T\mathbf{w}(0) + \eta \cdot \widehat{\mathbf{w}}^T\mathbf{x}(0) + \ldots + \eta \cdot \widehat{\mathbf{w}}^T\mathbf{x}(n-1) \\
\Rightarrow\ & \widehat{\mathbf{w}}^T\mathbf{w}(n) && \geq && \widehat{\mathbf{w}}^T\mathbf{w}(0) + \eta \cdot n\delta \ \geq\ 0 \quad \text{// For } n \geq n_0 \text{ with sufficiently large } n_0 \in \mathbb{N}. \\
\Rightarrow\ & (\widehat{\mathbf{w}}^T\mathbf{w}(n))^2 && \geq && (\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2
\end{aligned}$$

5. Apply the Cauchy-Schwarz inequality:

$$||\widehat{\mathbf{w}}||^2 \cdot ||\mathbf{w}(n)||^2 \ \geq\ (\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2 \quad \Leftrightarrow \quad ||\mathbf{w}(n)||^2 \ \geq\ \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2}$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

6. Consider again the weight adaptation $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$:

$$
\begin{aligned}
||\mathbf{w}(t+1)||^2 &= ||\mathbf{w}(t) + \eta \cdot \mathbf{x}(t)||^2 \\
&= (\mathbf{w}(t) + \eta \cdot \mathbf{x}(t))^T (\mathbf{w}(t) + \eta \cdot \mathbf{x}(t)) \\
&= \mathbf{w}(t)^T \mathbf{w}(t) + \eta^2 \cdot \mathbf{x}(t)^T \mathbf{x}(t) + 2\eta \cdot \mathbf{w}(t)^T \mathbf{x}(t) \\
&\leq ||\mathbf{w}(t)||^2 + ||\eta \cdot \mathbf{x}(t)||^2 \quad \text{// Since } \mathbf{w}(t)^T \mathbf{x}(t) < 0.
\end{aligned}
$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

6. Consider again the weight adaptation $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$:

$$\begin{aligned}
||\mathbf{w}(t+1)||^2 &= ||\mathbf{w}(t) + \eta \cdot \mathbf{x}(t)||^2 \\
&= (\mathbf{w}(t) + \eta \cdot \mathbf{x}(t))^T (\mathbf{w}(t) + \eta \cdot \mathbf{x}(t)) \\
&= \mathbf{w}(t)^T \mathbf{w}(t) + \eta^2 \cdot \mathbf{x}(t)^T \mathbf{x}(t) + 2\eta \cdot \mathbf{w}(t)^T \mathbf{x}(t) \\
&\leq ||\mathbf{w}(t)||^2 + ||\eta \cdot \mathbf{x}(t)||^2 \quad \text{// Since } \mathbf{w}(t)^T \mathbf{x}(t) < 0.
\end{aligned}$$

7. Consider the series $\mathbf{w}(n)$ from Step 2:

$$\begin{aligned}
||\mathbf{w}(n)||^2 &\leq ||\mathbf{w}(n-1)||^2 + ||\eta \cdot \mathbf{x}(n-1)||^2 \\
&\leq ||\mathbf{w}(n-2)||^2 + ||\eta \cdot \mathbf{x}(n-2)||^2 + ||\eta \cdot \mathbf{x}(n-1)||^2 \\
&\leq ||\mathbf{w}(0)||^2 + ||\eta \cdot \mathbf{x}(0)||^2 + \ldots + ||\eta \cdot \mathbf{x}(n-1)||^2 \\
&= ||\mathbf{w}(0)||^2 + \sum_{j=0}^{n-1} ||\eta \cdot \mathbf{x}(i)||^2
\end{aligned}$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

6. Consider again the weight adaptation $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t)$:

$$\begin{aligned} ||\mathbf{w}(t+1)||^2 &= ||\mathbf{w}(t) + \eta \cdot \mathbf{x}(t)||^2 \\ &= (\mathbf{w}(t) + \eta \cdot \mathbf{x}(t))^T (\mathbf{w}(t) + \eta \cdot \mathbf{x}(t)) \\ &= \mathbf{w}(t)^T \mathbf{w}(t) + \eta^2 \cdot \mathbf{x}(t)^T \mathbf{x}(t) + 2\eta \cdot \mathbf{w}(t)^T \mathbf{x}(t) \\ &\leq ||\mathbf{w}(t)||^2 + ||\eta \cdot \mathbf{x}(t)||^2 \quad \text{// Since } \mathbf{w}(t)^T \mathbf{x}(t) < 0. \end{aligned}$$

7. Consider the series $\mathbf{w}(n)$ from Step 2:

$$\begin{aligned} ||\mathbf{w}(n)||^2 &\leq ||\mathbf{w}(n-1)||^2 + ||\eta \cdot \mathbf{x}(n-1)||^2 \\ &\leq ||\mathbf{w}(n-2)||^2 + ||\eta \cdot \mathbf{x}(n-2)||^2 + ||\eta \cdot \mathbf{x}(n-1)||^2 \\ &\leq ||\mathbf{w}(0)||^2 + ||\eta \cdot \mathbf{x}(0)||^2 + \ldots + ||\eta \cdot \mathbf{x}(n-1)||^2 \\ &= ||\mathbf{w}(0)||^2 + \sum_{j=0}^{n-1} ||\eta \cdot \mathbf{x}(i)||^2 \end{aligned}$$

8. With $\varepsilon := \max_{\mathbf{x} \in X'} ||\mathbf{x}||^2$ follows $||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

9. Both inequalities (see Step 5 and Step 8) must be fulfilled:

$$||\mathbf{w}(n)||^2 \geq \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \quad \text{and} \quad ||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

$$\Rightarrow \quad \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \leq ||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

$$\Rightarrow \quad \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

9. Both inequalities (see Step 5 and Step 8) must be fulfilled:

$$||\mathbf{w}(n)||^2 \geq \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \quad \text{and} \quad ||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

$$\Rightarrow \quad \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \leq ||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

$$\Rightarrow \quad \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0) + n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

Set $\mathbf{w}(0) = \mathbf{0}$:

$$\Rightarrow \quad \frac{n^2\eta^2\delta^2}{||\widehat{\mathbf{w}}||^2} \leq n\eta^2\varepsilon$$

$$\Leftrightarrow \quad n \leq \frac{\varepsilon}{\delta^2} \cdot ||\widehat{\mathbf{w}}||^2$$

Perceptron Learning

Perceptron Convergence Theorem: Proof (continued)

9. Both inequalities (see Step 5 and Step 8) must be fulfilled:

$$||\mathbf{w}(n)||^2 \geq \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0)+n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \quad \text{and} \quad ||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

$$\Rightarrow \quad \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0)+n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \leq ||\mathbf{w}(n)||^2 \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

$$\Rightarrow \quad \frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0)+n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \leq ||\mathbf{w}(0)||^2 + n\eta^2\varepsilon$$

Set $\mathbf{w}(0)=\mathbf{0}$:

$$\Rightarrow \quad \frac{n^2\eta^2\delta^2}{||\widehat{\mathbf{w}}||^2} \leq n\eta^2\varepsilon$$

$$\Leftrightarrow \quad n \leq \frac{\varepsilon}{\delta^2}\cdot||\widehat{\mathbf{w}}||^2$$

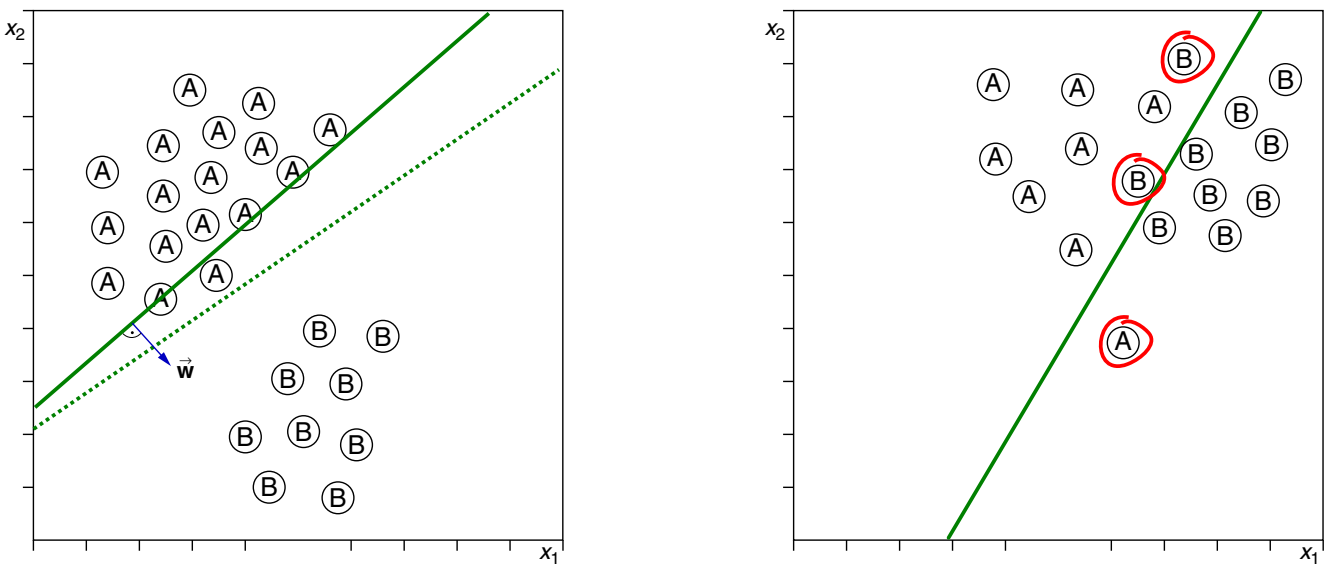
$\rightsquigarrow$ The PT algorithm terminates within a finite number of iterations.

Observe: $\frac{(\widehat{\mathbf{w}}^T\mathbf{w}(0)+n\eta\delta)^2}{||\widehat{\mathbf{w}}||^2} \in \Theta(n^2)$ and $||\mathbf{w}(0)||^2 + n\eta^2\varepsilon \in \Theta(n)$

Perceptron Learning

Perceptron Convergence Theorem: Discussion [theorem]

Given some $\mathbf{w}$, the PT algorithm checks if the examples $(\mathbf{x}, c) \in D$ are on the correct *hyperplane side* and possibly adapts $\mathbf{w}$ (left). Goal is to find a separating hyperplane $\mathbf{w}$.



If the classes are linearly separable (left), the PT algorithm will converge. If no such hyperplane exists, convergence cannot be guaranteed (right).

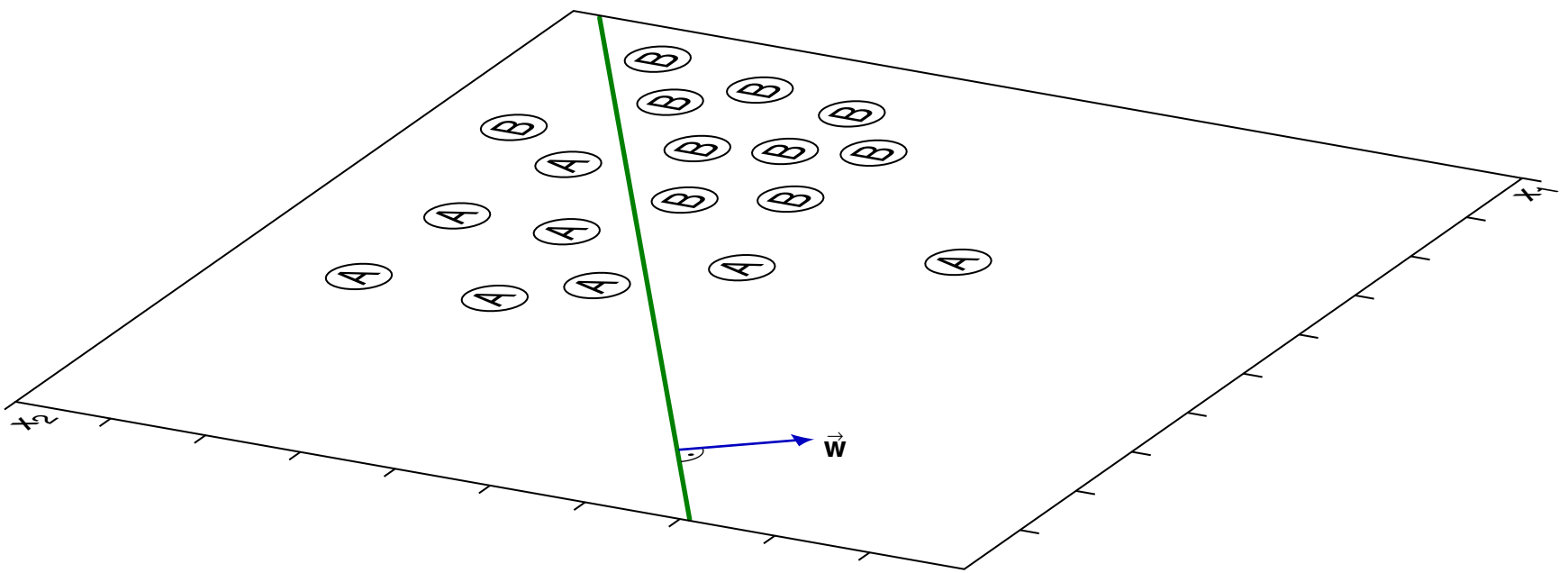
Remarks:

- The PT algorithm may require an exponential number of iterations to find a separating hyperplane. With linear programming a separating hyperplane can be found in polynomial time.
- Owed to the intialization of the random weights, but also to the randomized encounter of wrongly classified examples, different runs of the PT algorithm on the same set of training examples D can lead to different solutions (different separating hyperplanes).

Perceptron Learning

PT Algorithm versus Regression

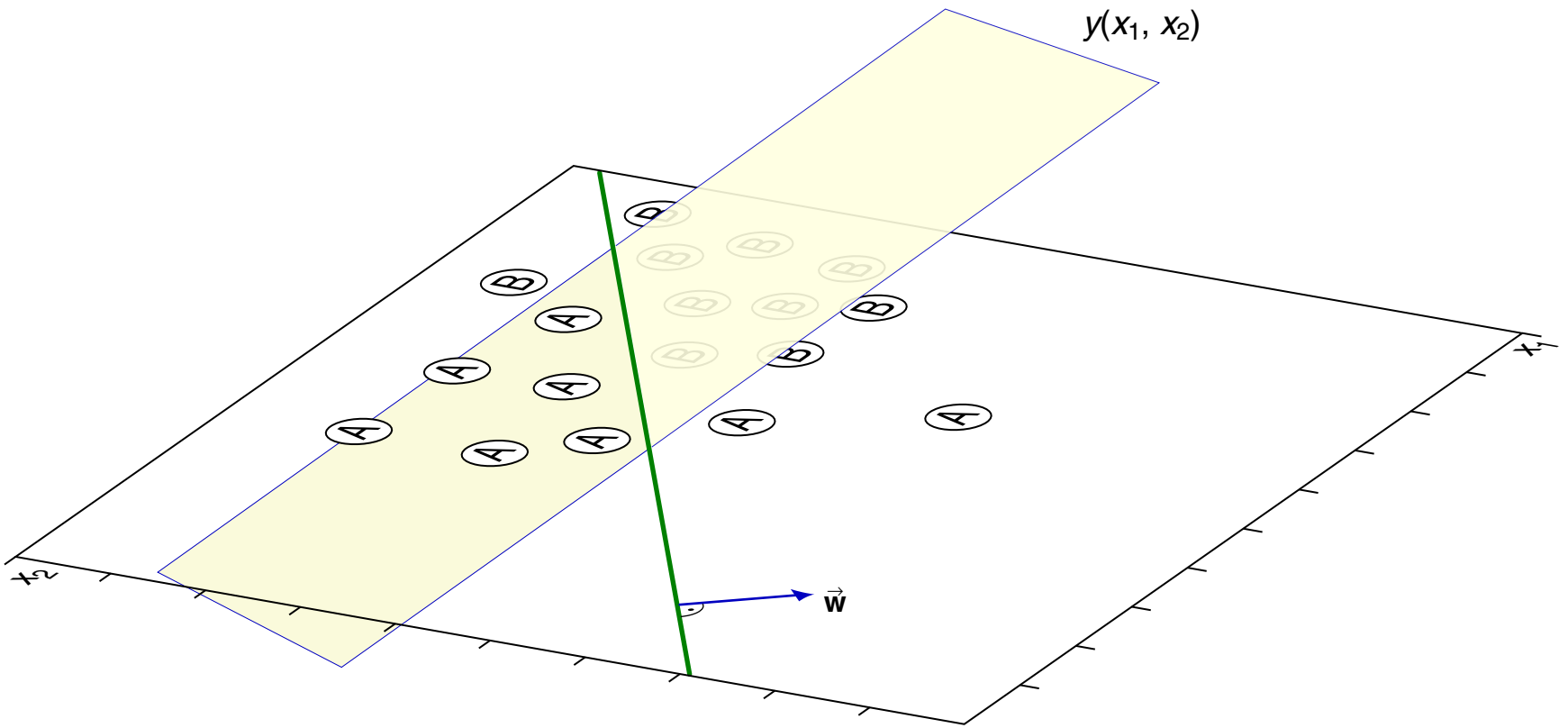
Given some $\mathbf{w}$, regression methods compute a loss, quantifying the "grade of misclassification", by exploiting both the *hyperplane side and the distance*, given the examples $(\mathbf{x}, c) \in D$. Goal is to find a loss-minimum hyperplane $\mathbf{w}$.



Perceptron Learning

PT Algorithm versus Regression (continued)

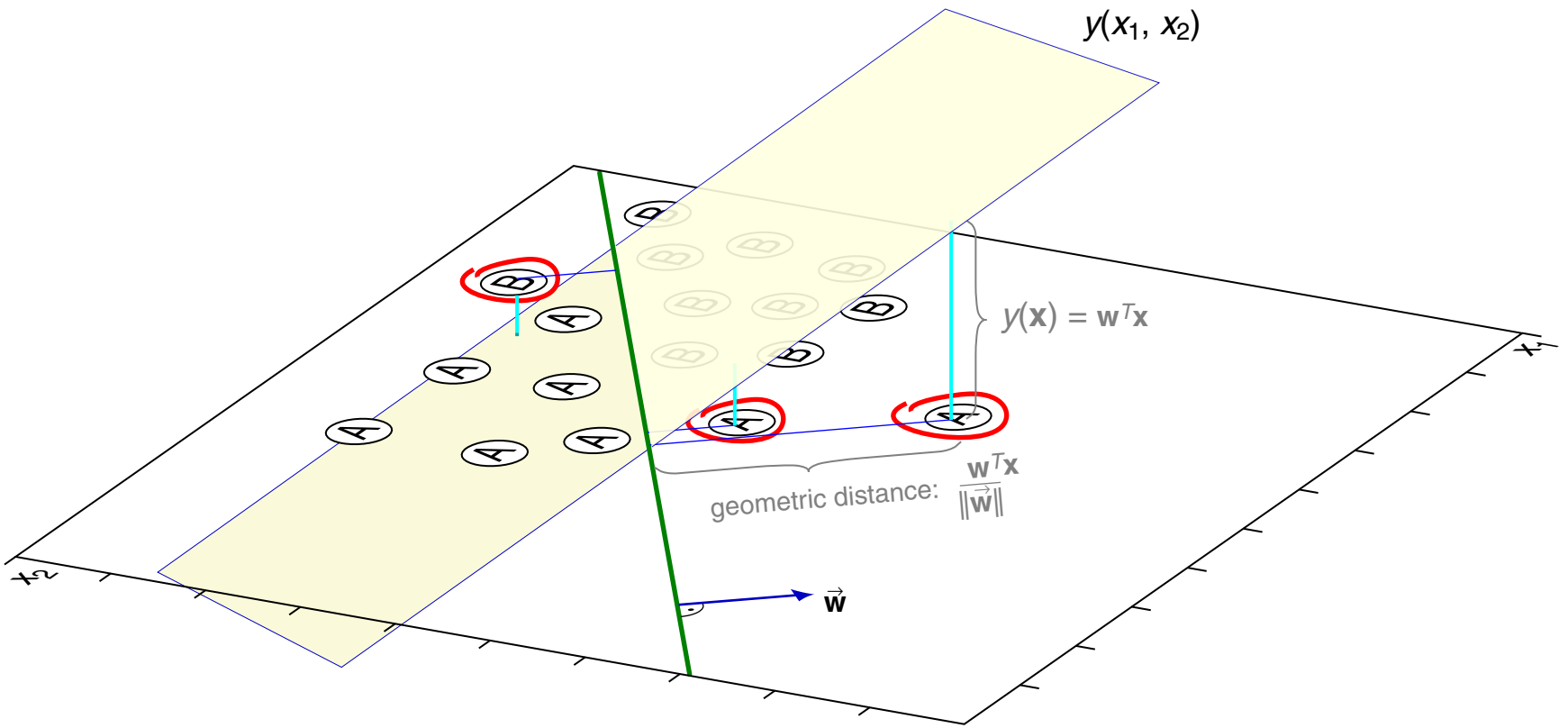
Given some $\mathbf{w}$, regression methods compute a loss, quantifying the "grade of misclassification", by exploiting both the *hyperplane side and the distance*, given the examples $(\mathbf{x}, c) \in D$. Goal is to find a loss-minimum hyperplane $\mathbf{w}$.



Perceptron Learning

PT Algorithm versus Regression (continued)

Given some $\mathbf{w}$, regression methods compute a loss, quantifying the "grade of misclassification", by exploiting both the *hyperplane side and the distance*, given the examples $(\mathbf{x}, c) \in D$. Goal is to find a loss-minimum hyperplane $\mathbf{w}$.



Remarks:

- Both approaches, the PT algorithm and regression, determine a hyperplane (= weights $\mathbf{w}$) in the p-dimensional input space. Those $\mathbf{x} \in \mathbf{R}^p$ that fullfill the identity $\mathbf{w}^T\mathbf{x} = 0$ form the hyperplane. See the illustrations for (1) the PT algorithm and (2) linear regression.

Remarks: (continued)

- The PT algorithm looks similar to the incremental gradient descent algorithm, IGD, since these algorithms differ only in the computation of $y(\mathbf{x})$ (Line 5) where the former applies the *heaviside* function to $\mathbf{w}^T\mathbf{x}$. However, this subtle "syntactic difference" is a significant conceptual difference, entailing a number of consequences:
  - The PT algorithm is not based on residuals (in the $(p+1)$-dimensional input-output-space) but refers to the input space only, where it simply evaluates the side of the hyperplane as a binary feature (correct side or not).
  - Gradient descent is a *regression* approach which exploits the residuals provided by a loss function of choice, whose differential is evaluated to guide hyperplane search.
  - Provided linear separability, the PT algorithm will converge within a finite number of iterations, which, however, cannot be guaranteed for gradient descent. [theorem]
  - Gradient descent may converge even if the data is not linearly separable. However, this convergence process is of an asymptotic nature and no finite iteration bound can be stated.
  - Data sets can be constructed whose classes are linearly separable, but where gradient descent will not determine a hyperplane that classifies all examples correctly (whereas the PT algorithm of course does).