Support Vector Machine

Florian Abt

17. Mai 2018

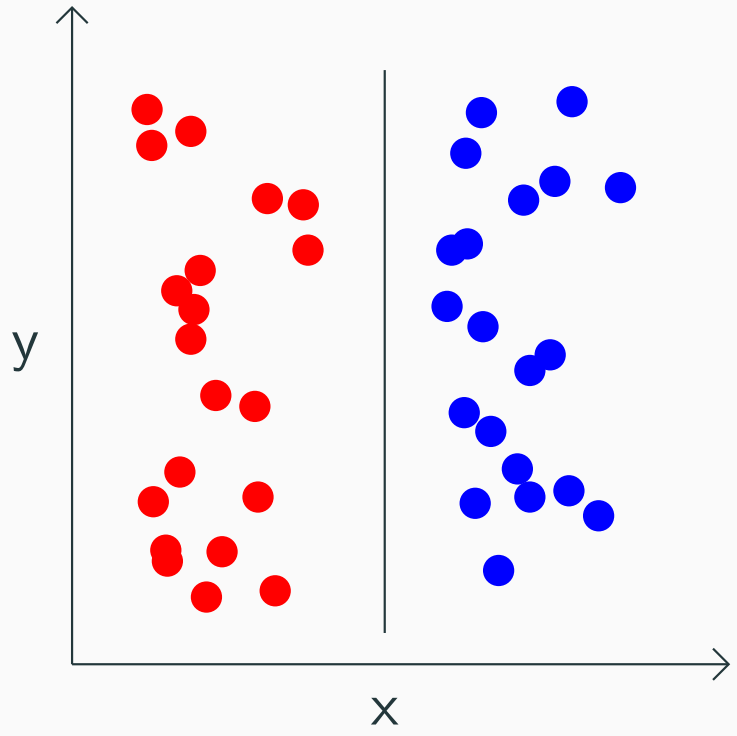
Universität Hamburg - Informatik

Grundlegende Idee

Grundlegende Idee - Das Problem

- Klassifikation von Elementen
- Verteilungen unbekannt
- Elemente befinden sich im n-dimensionalen Raum

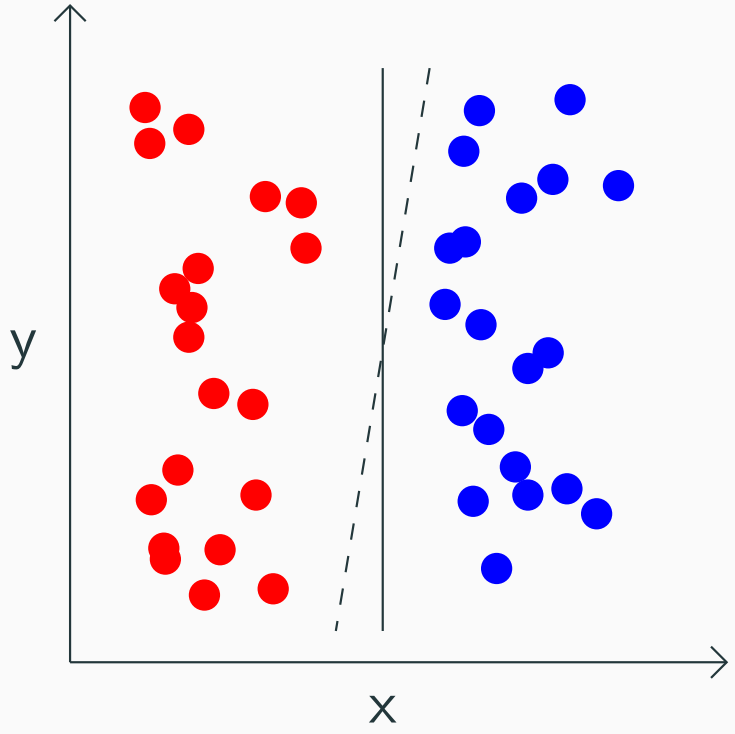
$\Rightarrow$ Platzieren einer Hyperebene zur Klassifikation
(mit minimalem Trainings-Fehler)



Grundlegende Idee - Das Problem

- Klassifikation von Elementen
- Verteilungen unbekannt
- Elemente befinden sich im n-dimensionalen Raum

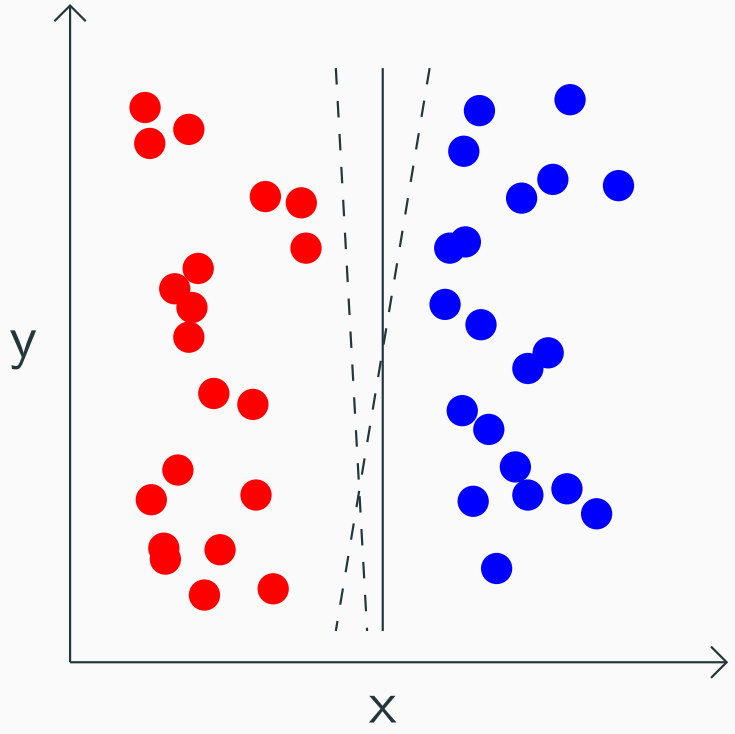
$\Rightarrow$ Platzieren einer Hyperebene zur Klassifikation
(mit minimalem Trainings-Fehler)



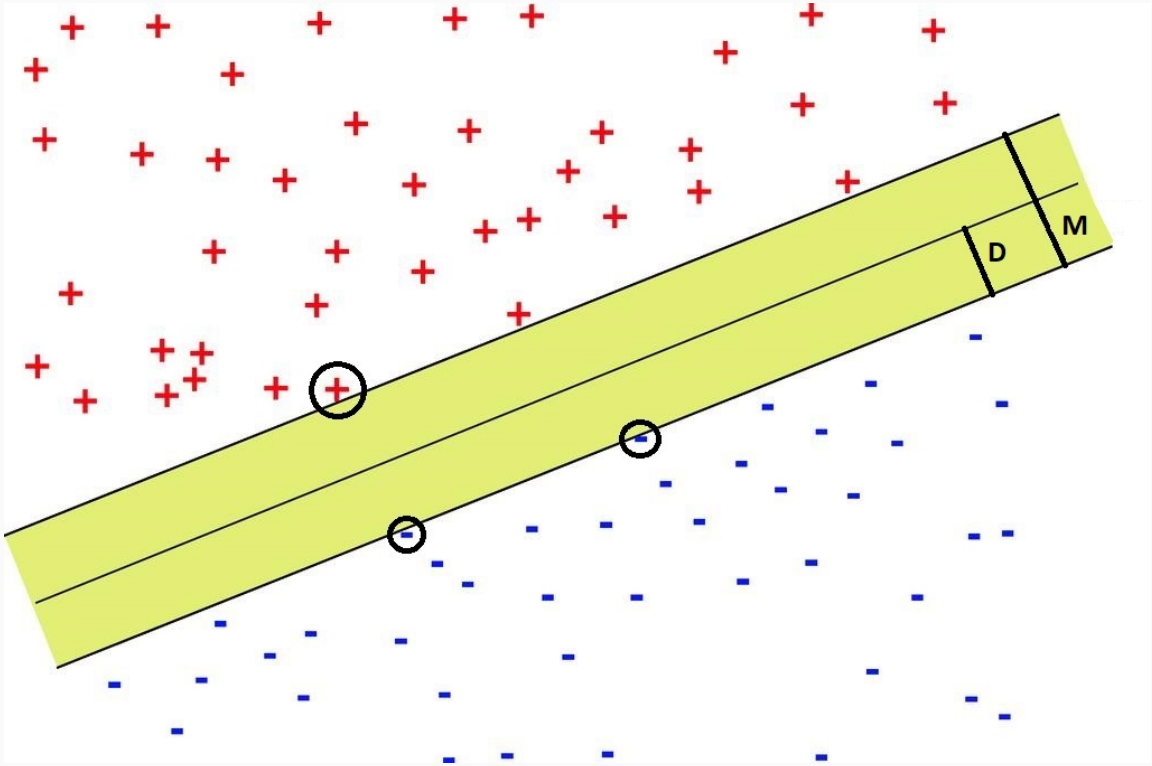
Grundlegende Idee - Das Problem

- Klassifikation von Elementen
- Verteilungen unbekannt
- Elemente befinden sich im n-dimensionalen Raum

$\Rightarrow$ Platzieren einer Hyperebene zur Klassifikation
(mit minimalem Trainings-Fehler)

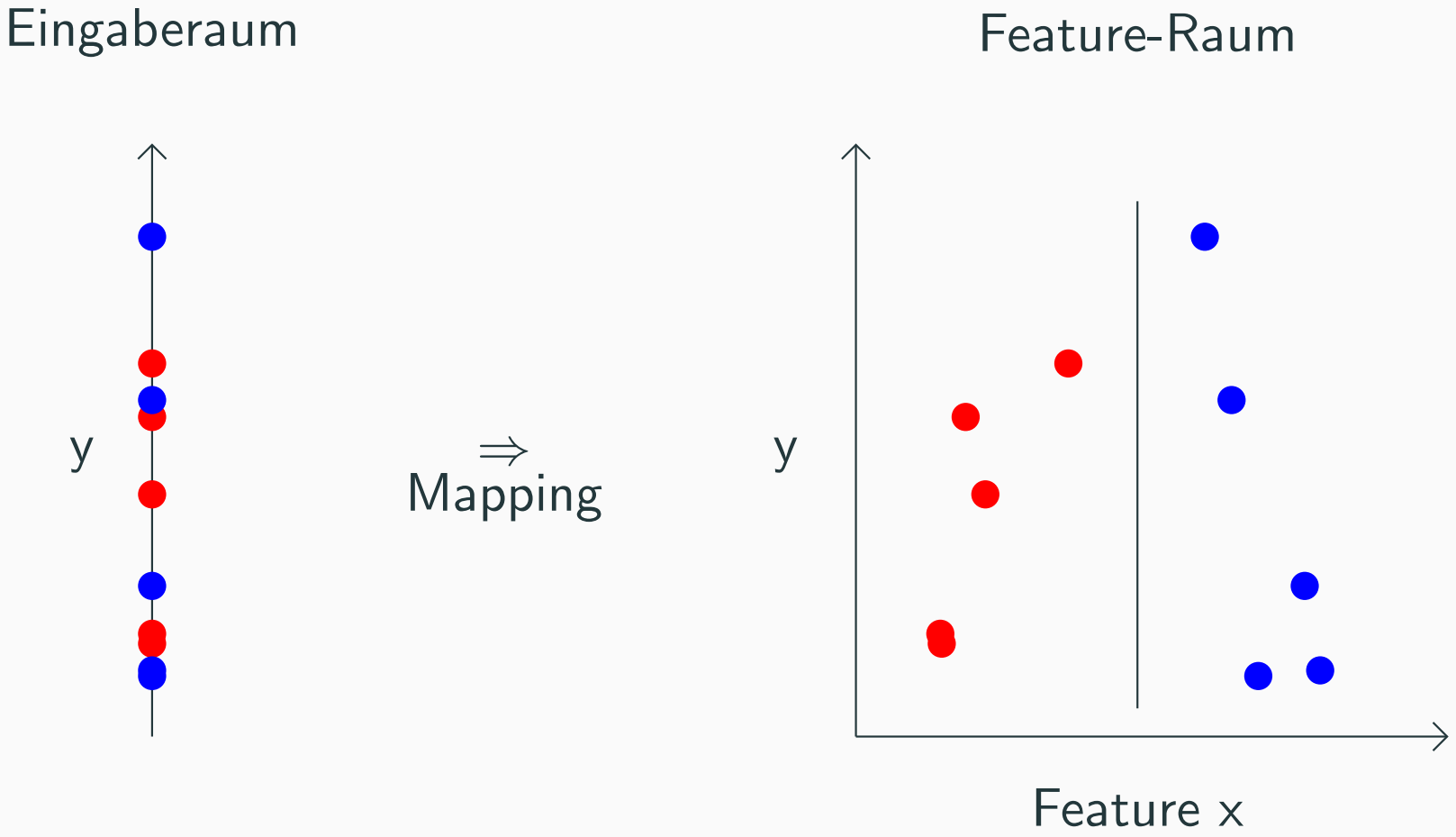


Grundlegende Idee - Vorgehen



- Maximaler Abstand $\Rightarrow$ geringes Fehlerrisiko
- Stützvektoren zur Bestimmung der Hyperebene

Grundlegende Idee - Nicht-lineares Mapping



Algorithmen

Algorithmen - Übersicht

- Linear Maximum Margin Classifier
  - klare lineare Separierung
  - lineare Hyperebene
  - Maximaler Abstand Ebene - Elemente

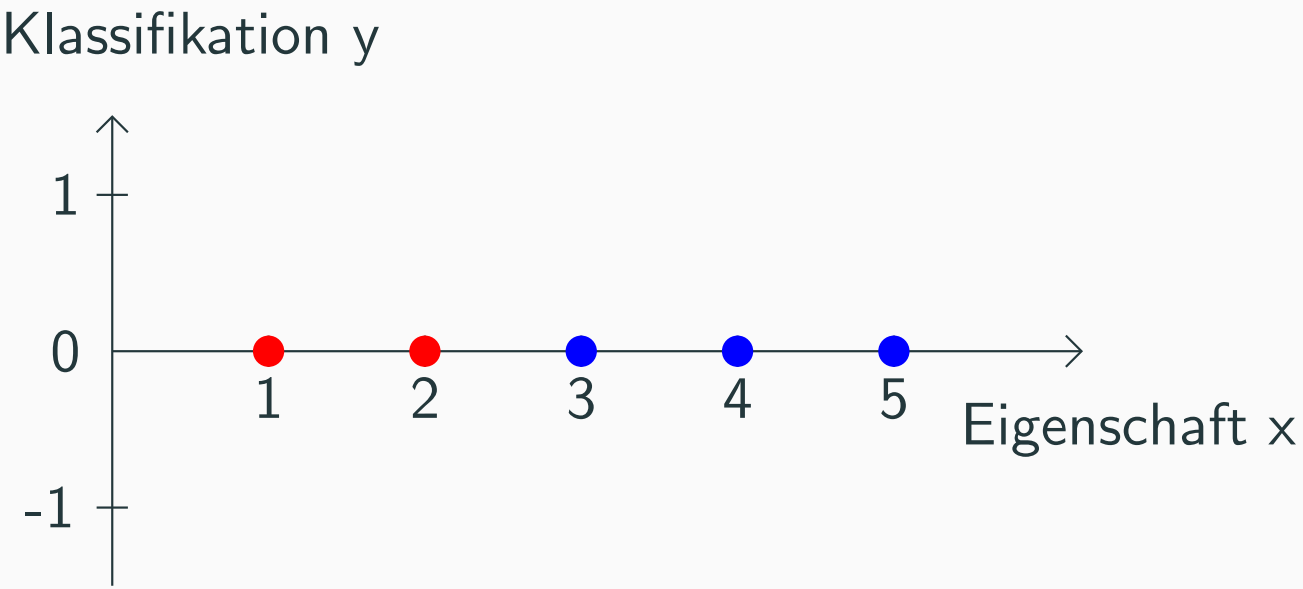
- Linear Soft Margin Classifier
  - nicht klare lineare Separierung
  - lineare Hyperebene
  - bestimmte Anzahl Elemente falsch klassifiziert

- Nonlinear Classifier
  - keine lineare Separierung möglich
  - nicht-lineare Hyperebene
    bzw.
  - nicht-lineares Mapping erforderlich

Algorithmen - Linear Maximum Margin Classifier

Beispiel:

$$(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n), \quad x \in R^1, \quad y \in \{+1, -1\}$$



Algorithmen - Linear Maximum Margin Classifier

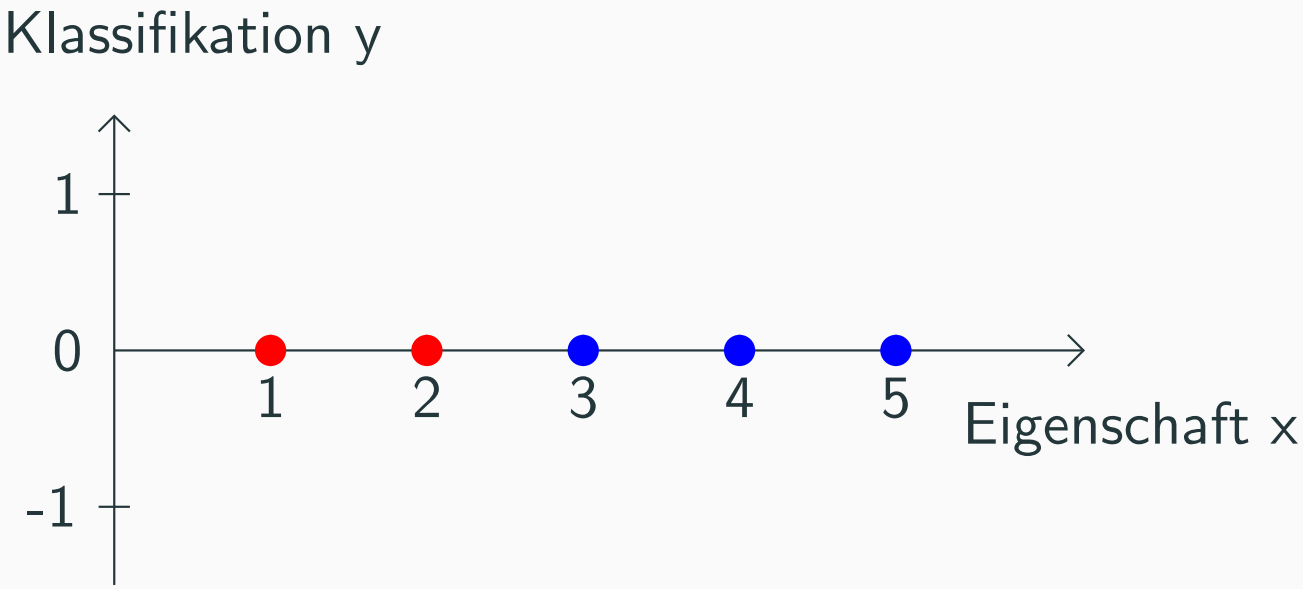
Annahme: Finde Hyperebene mit größtem Abstand
$\Rightarrow$ geringstes Klassifizierungsrisiko

Entscheidungsfunktion:

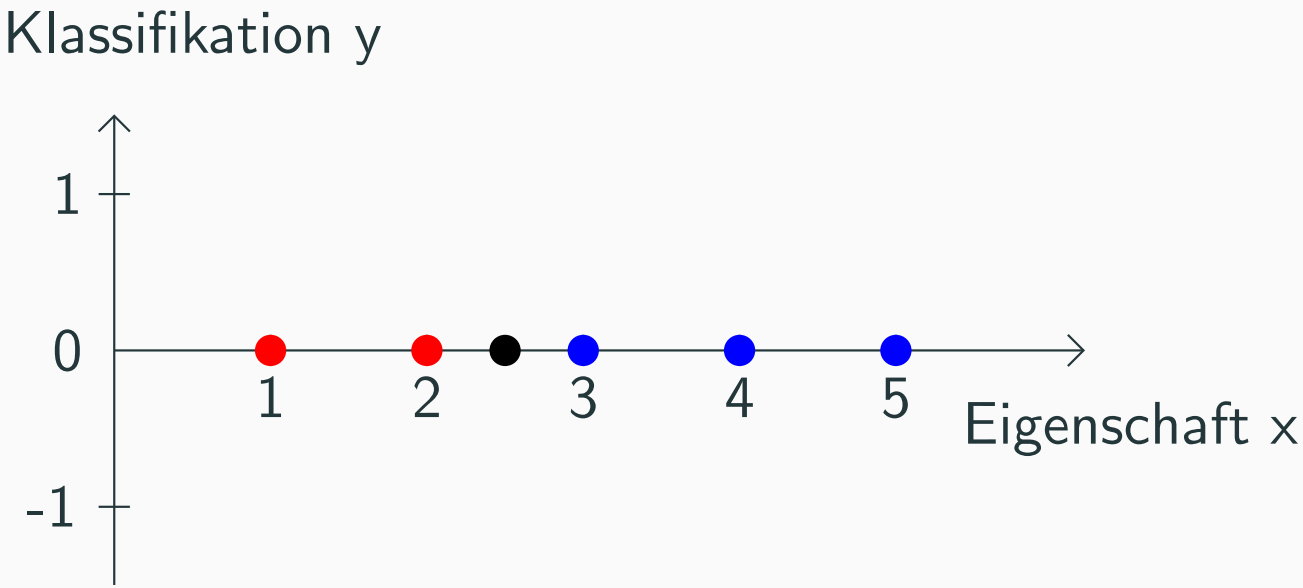
$$d(x, w, b) = w^T \cdot x + b = \sum_{i=1}^{n} w_i \cdot x_i + b$$

Zu bestimmende Parameter der Entscheidungsfunktion (Ebenengleichung) sind $w = [w_1 \quad w_2 \quad \dots \quad w_n]^T$ und b

Algorithmen - Linear Maximum Margin Classifier

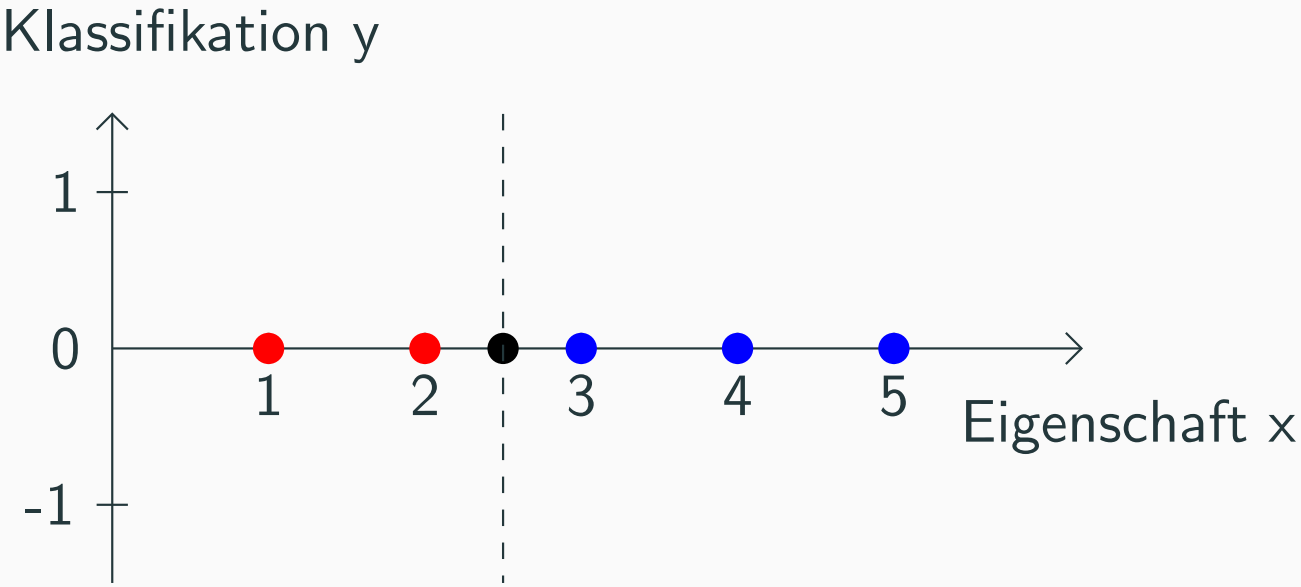


Algorithmen - Linear Maximum Margin Classifier



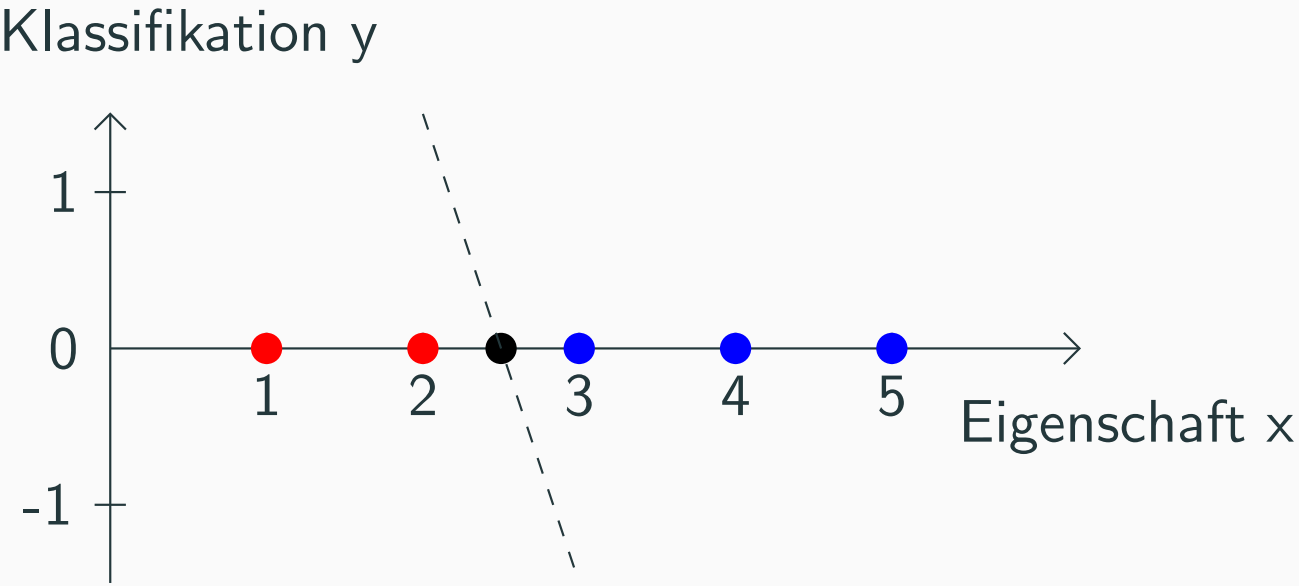
Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)



Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

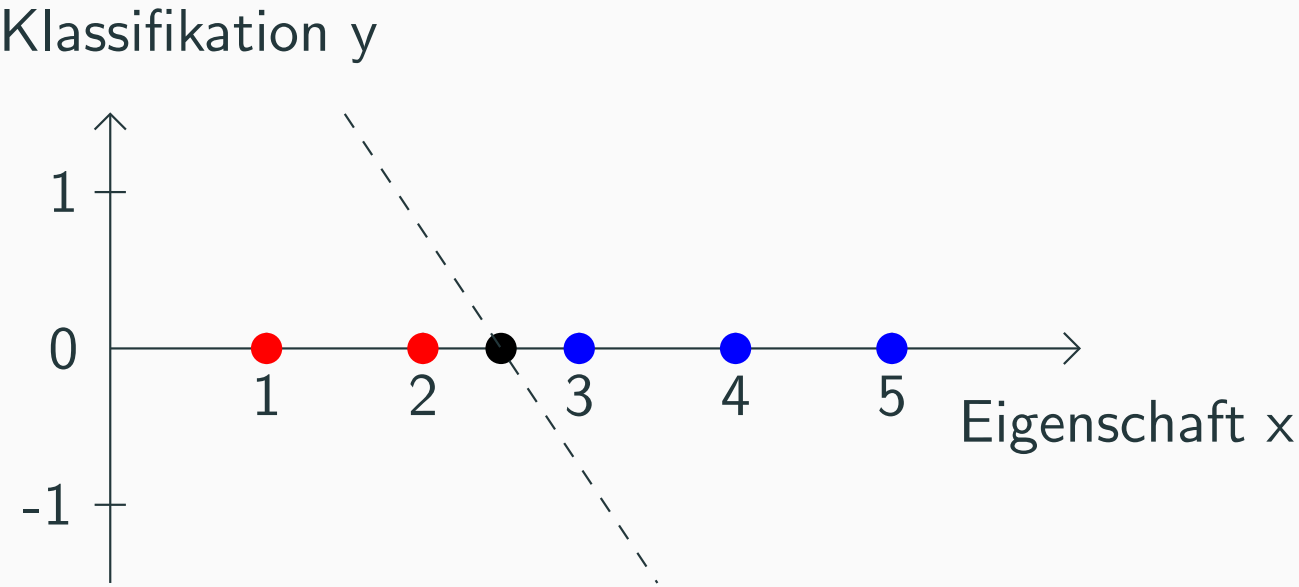


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

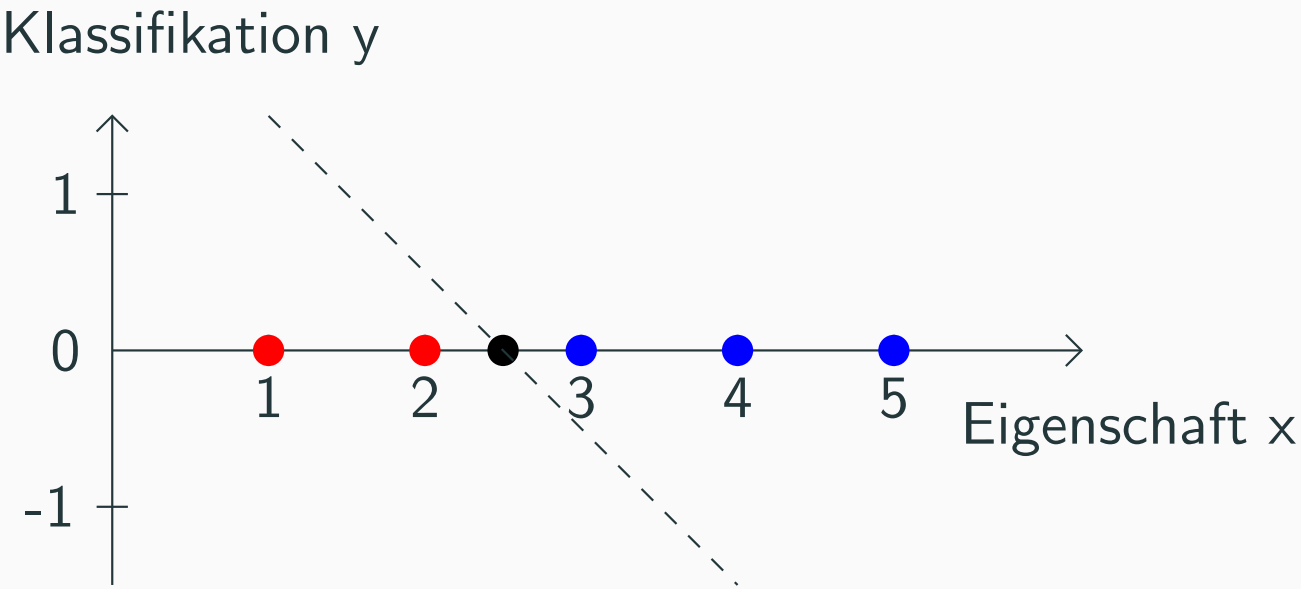


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

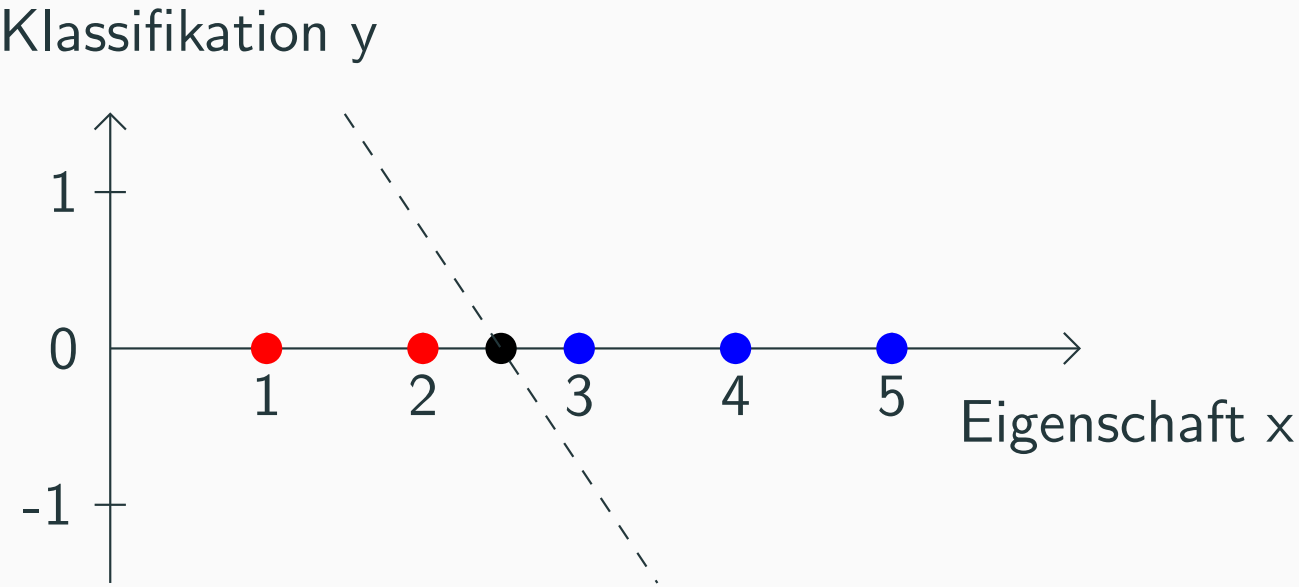


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

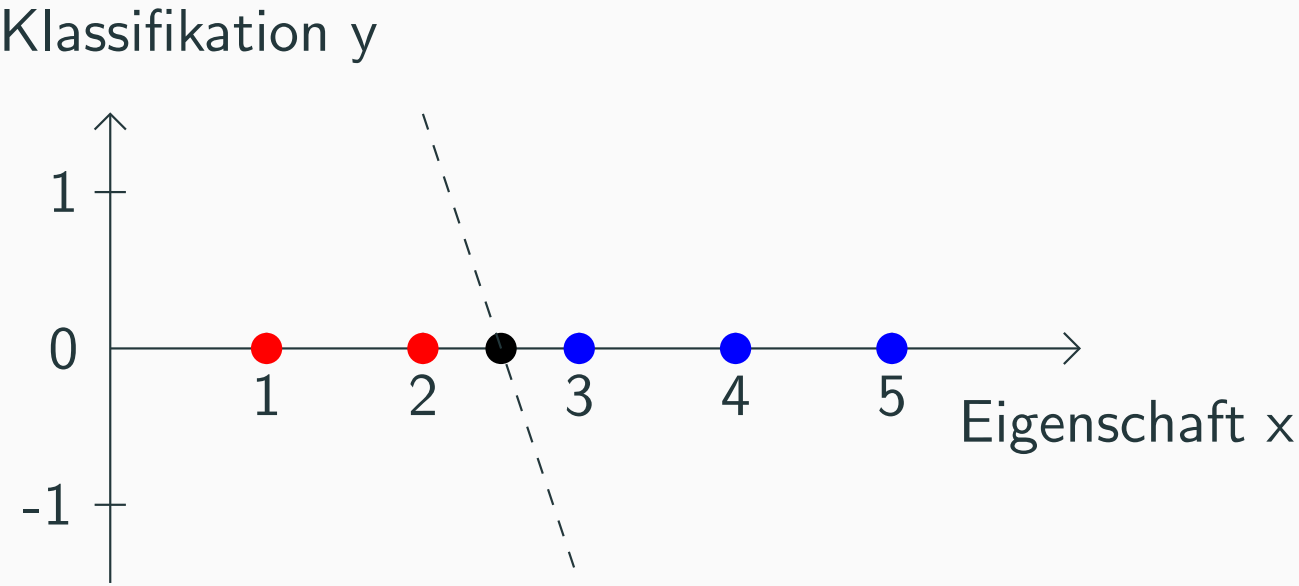


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

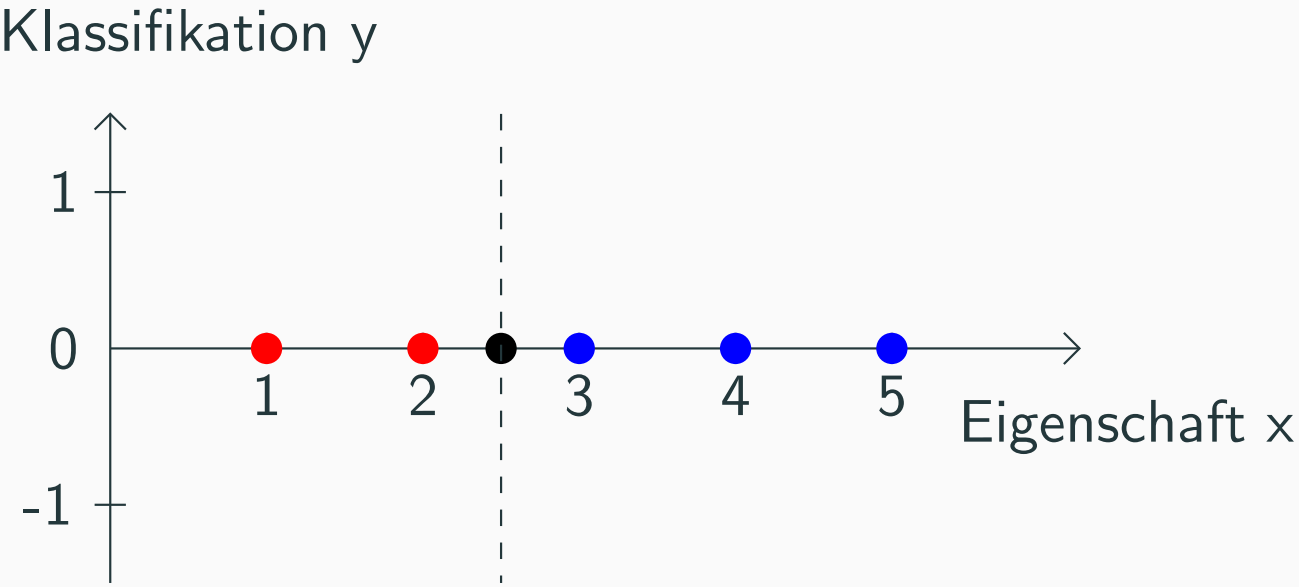


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

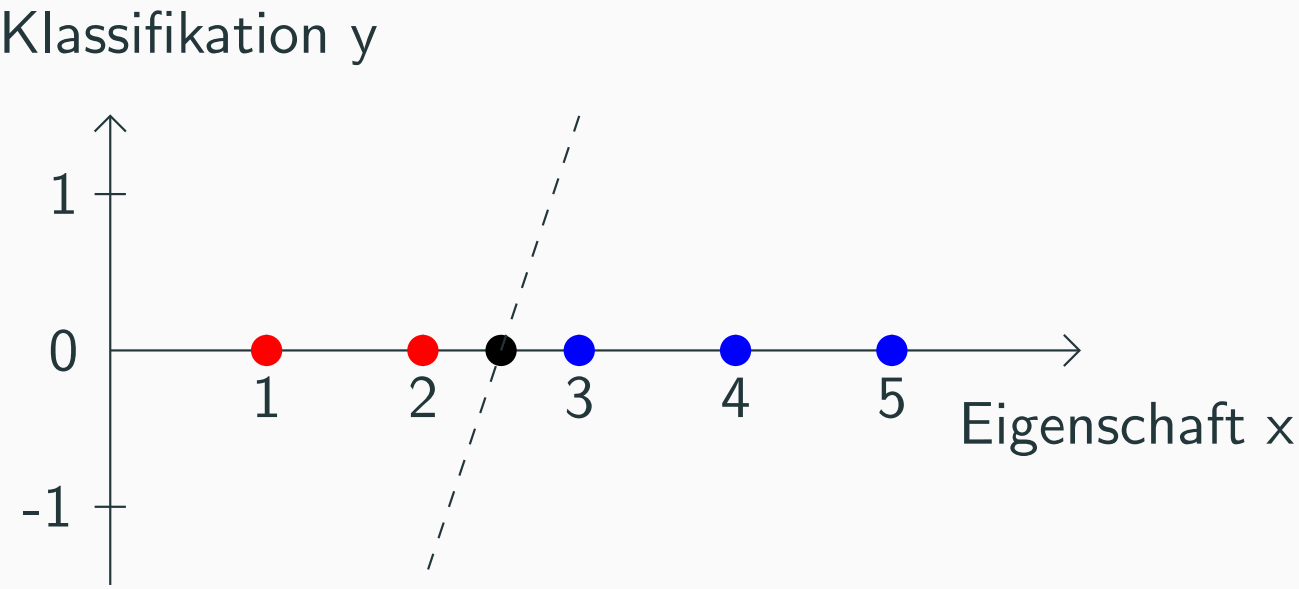


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

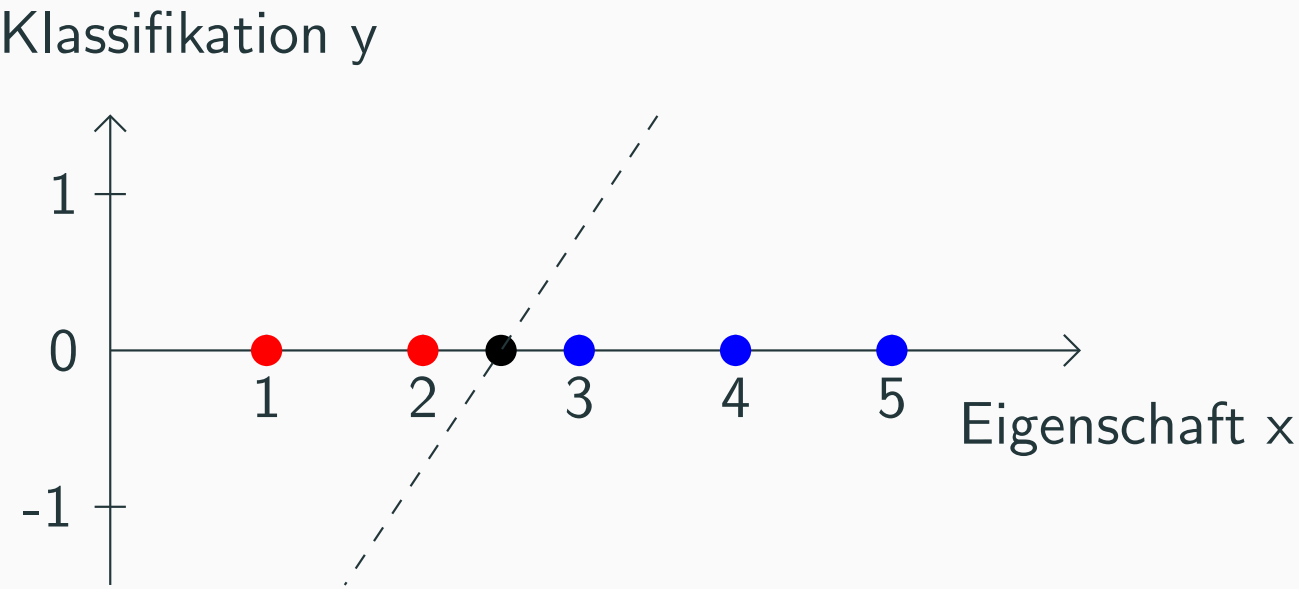


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)

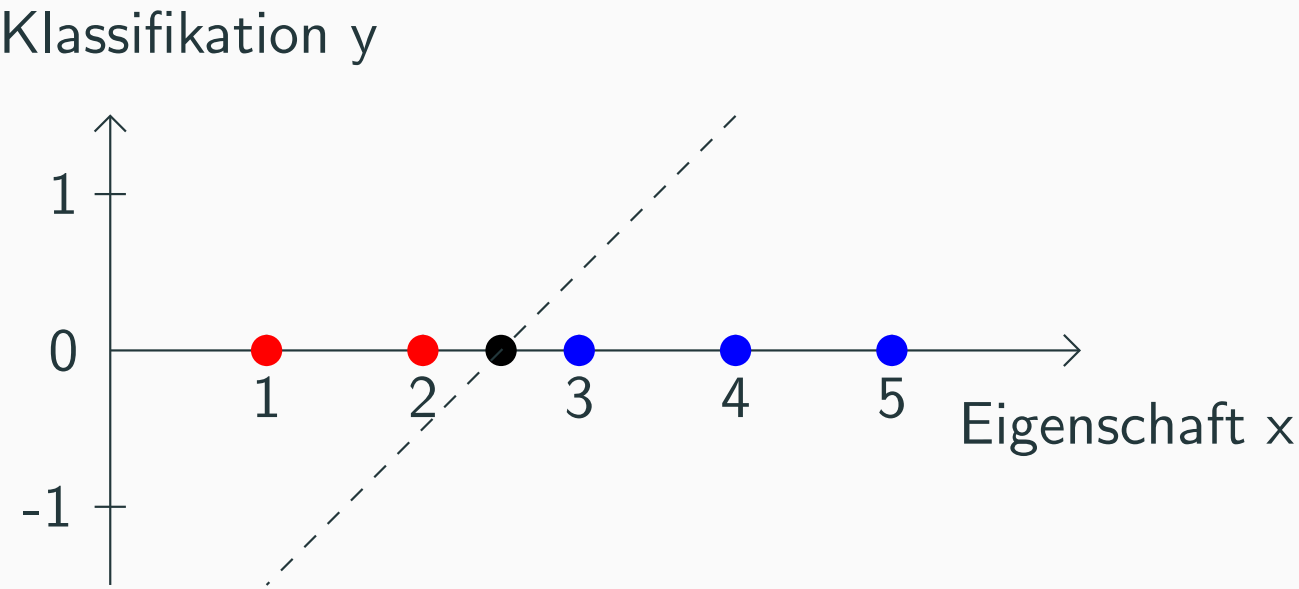


$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

$d(x, w, b)$ beschreibt kanonische Hyperebene (Hypergerade)



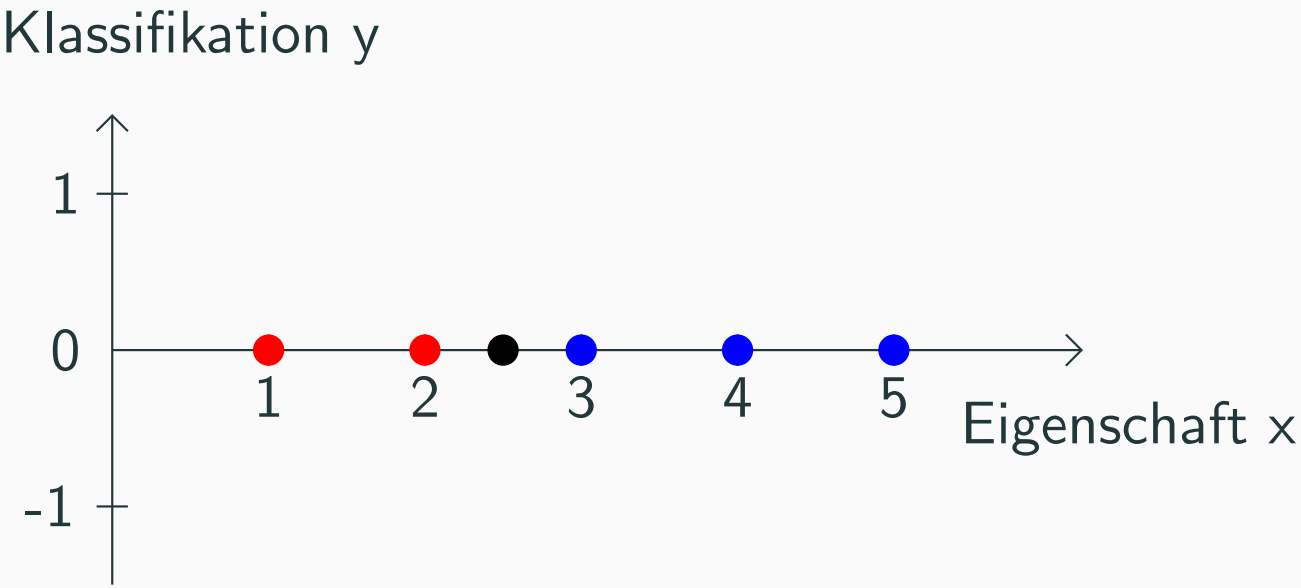
$d(x, k \cdot w, k \cdot b), \quad k \in \mathbb{N}$

beschreibt alle möglichen Hyperebenen

Algorithmen - Linear Maximum Margin Classifier

Indikatorfunktion i_F (Abbildung der Entscheidungsfunktion auf $y \in \{1, -1\}$)

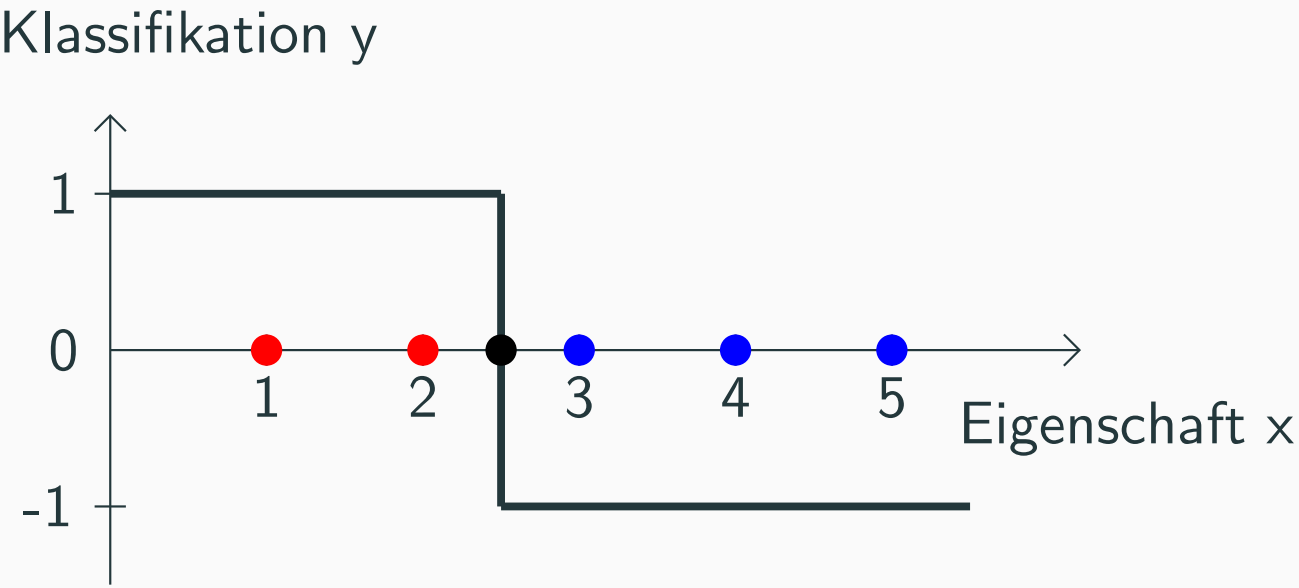
$$i_F = sign(d(x_p, w, b)) := \begin{cases} 1, & \text{if } d(x_p, w, b) > 0 \quad (y_p = 1) \\ -1, & \text{if } d(x_p, w, b) < 0 \quad (y_p = -1) \end{cases}$$



Algorithmen - Linear Maximum Margin Classifier

Indikatorfunktion i_F (Abbildung der Entscheidungsfunktion auf $y \in \{1, -1\}$)

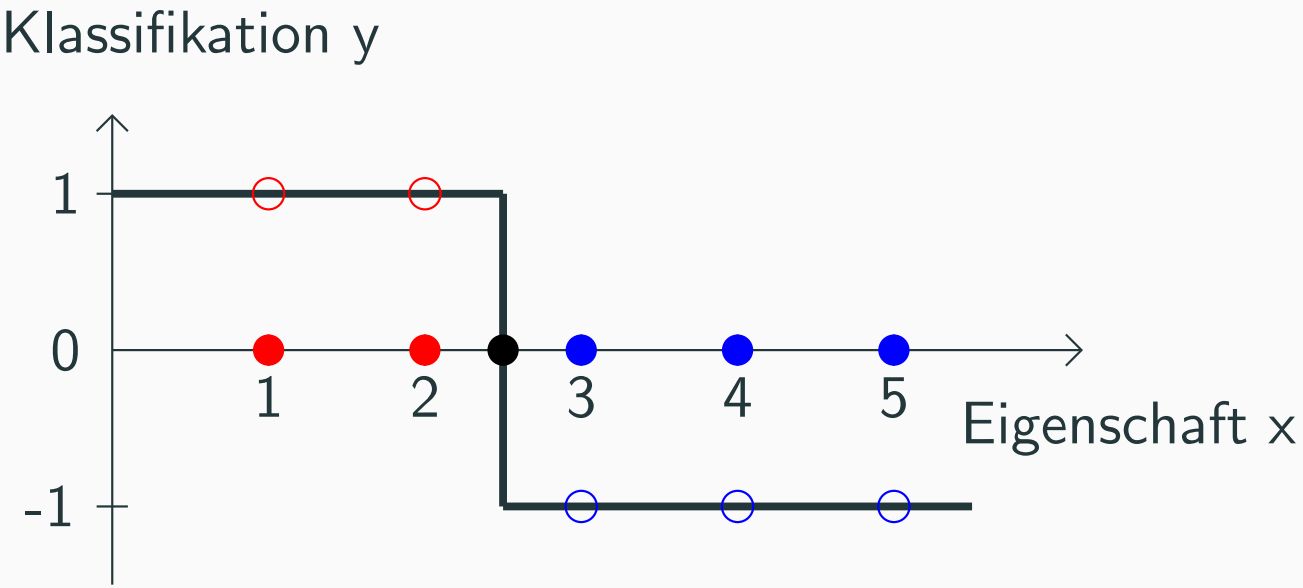
$$i_F = sign(d(x_p, w, b)) := \begin{cases} 1, & \text{if } d(x_p, w, b) > 0 \quad (y_p = 1) \\ -1, & \text{if } d(x_p, w, b) < 0 \quad (y_p = -1) \end{cases}$$



Algorithmen - Linear Maximum Margin Classifier

Indikatorfunktion i_F (Abbildung der Entscheidungsfunktion auf $y \in \{1, -1\}$)

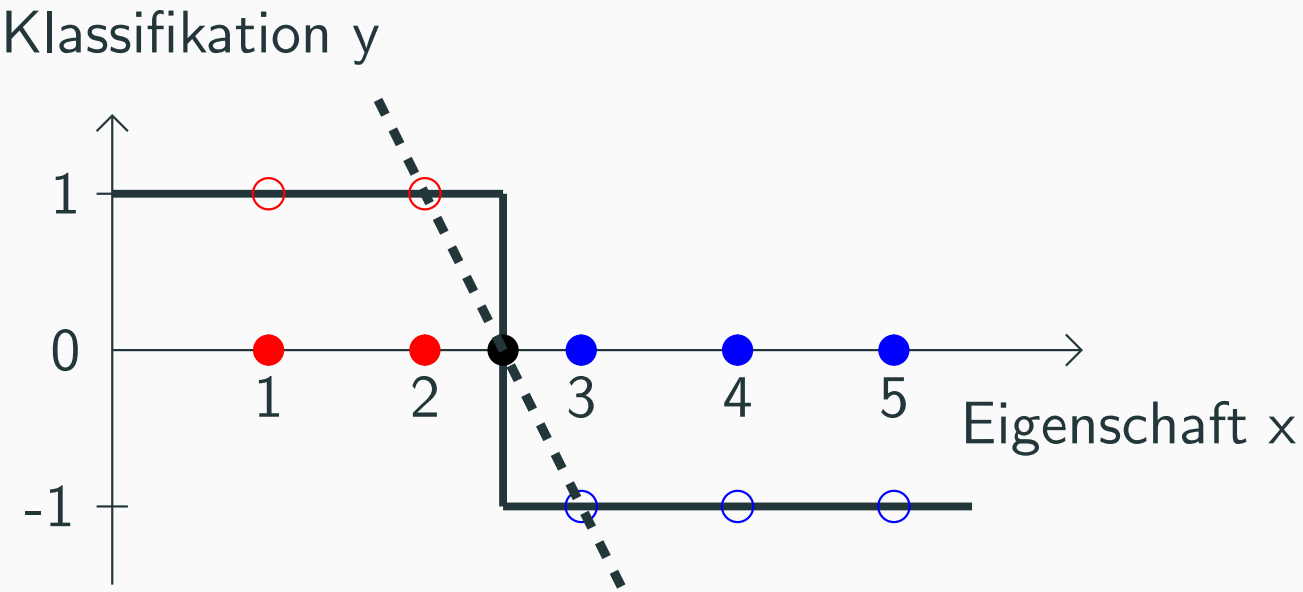
$$i_F = sign(d(x_p, w, b)) := \begin{cases} 1, & \text{if } d(x_p, w, b) > 0 \quad (y_p = 1) \\ -1, & \text{if } d(x_p, w, b) < 0 \quad (y_p = -1) \end{cases}$$



Algorithmen - Linear Maximum Margin Classifier

Indikatorfunktion i_F (Abbildung der Entscheidungsfunktion auf $y \in \{1, -1\}$)

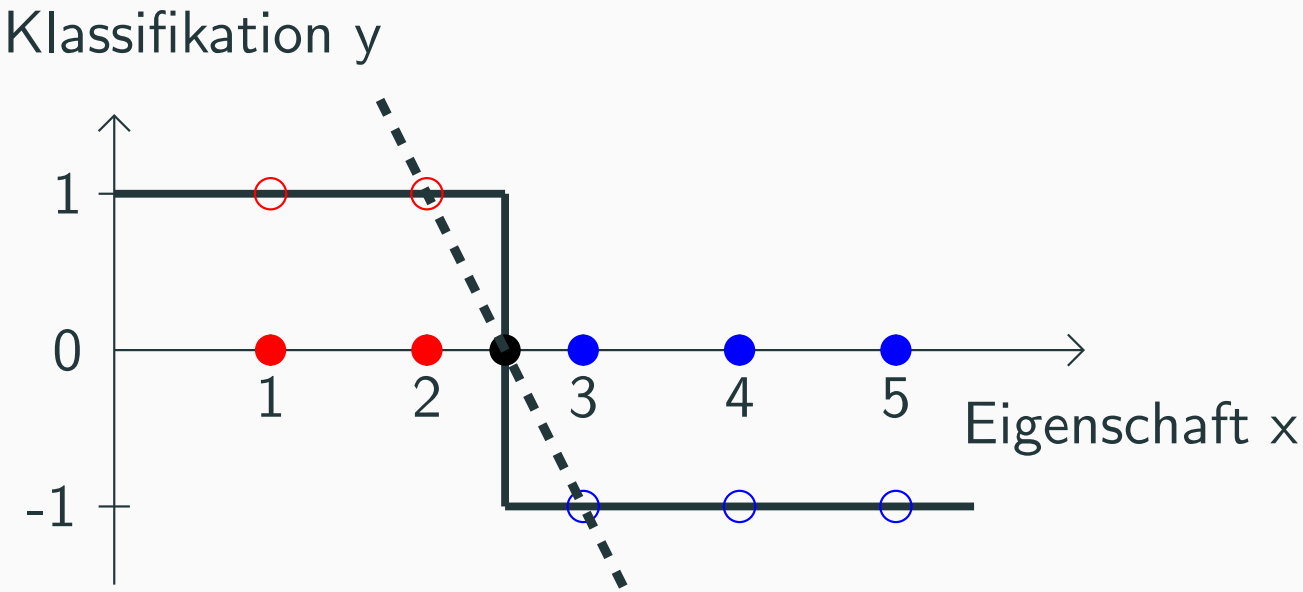
$$i_F = sign(d(x_p, w, b)) := \begin{cases} 1, & \text{if } d(x_p, w, b) > 0 \quad (y_p = 1) \\ -1, & \text{if } d(x_p, w, b) < 0 \quad (y_p = -1) \end{cases}$$



Algorithmen - Linear Maximum Margin Classifier

Indikatorfunktion i_F (Abbildung der Entscheidungsfunktion auf $y \in \{1, -1\}$)

$$i_F = sign(d(x_p, w, b)) := \begin{cases} 1, & \text{if } d(x_p, w, b) > 0 \quad (y_p = 1) \\ -1, & \text{if } d(x_p, w, b) < 0 \quad (y_p = -1) \end{cases}$$

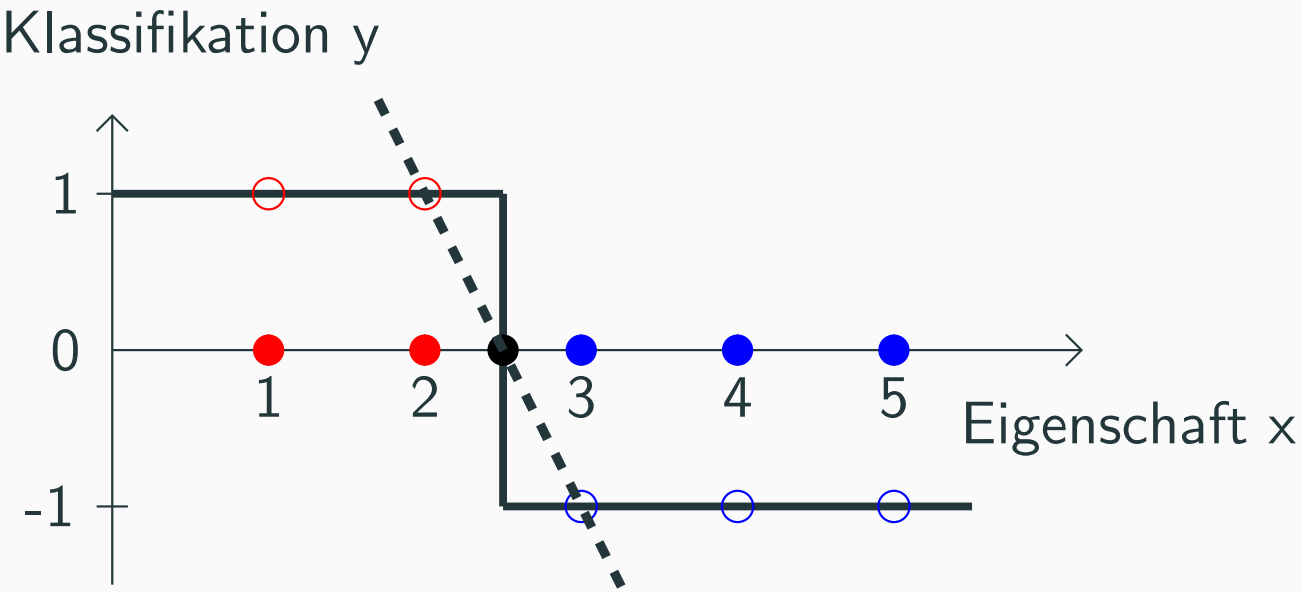


Für Stützvektoren gilt: $d(x, w, b) = i_F = 1$

Algorithmen - Linear Maximum Margin Classifier

Indikatorfunktion i_F (Abbildung der Entscheidungsfunktion auf $y \in \{1, -1\}$)

$$i_F = sign(d(x_p, w, b)) := \begin{cases} 1, & \text{if } d(x_p, w, b) > 0 \quad (y_p = 1) \\ -1, & \text{if } d(x_p, w, b) < 0 \quad (y_p = -1) \end{cases}$$



Für Stützvektoren gilt: $d(x, w, b) = i_F = 1$

Entscheidungsgrenze: $d(x, w, b) = 0$ (im Eingaberaum)

Algorithmen - Linear Maximum Margin Classifier

Kanonische Hyperebene:

- für alle Trainingselemente x gilt: $|d| >= |i_F|$
- R^1 Eingaberaum $\Rightarrow$ nur 1 kanonische Hyperebene
- R^n Eingaberaum $\Rightarrow$ n kanonische Hyperebenen

Optimale kanonische Hyperebenen im R^n
$\Rightarrow$ Maximaler Abstand D zu den Stützvektoren

Algorithmen - Linear Maximum Margin Classifier

Abstand Punkt p zur Hyperebene:

$$D = \frac{|(wx_p) \pm b|}{||w||} = \frac{|w_1 x_{1p} + w_2 x_{2p} + \cdots + w_n x_{np} \pm b|}{\sqrt{w_1^2 + w_2^2 + \cdots + w_3^2}} \quad (1)$$

Algorithmen - Linear Maximum Margin Classifier

Abstand Punkt p zur Hyperebene:

$$D = \frac{|(wx_p) \pm b|}{||w||} = \frac{|w_1 x_{1p} + w_2 x_{2p} + \cdots + w_n x_{np} \pm b|}{\sqrt{w_1^2 + w_2^2 + \cdots + w_3^2}} \quad (1)$$

Da Punkt p ein Stützvektor gilt:

$$d(x_p, w, b) = w^T x_p + b = i_F = 1 \quad (2)$$

Algorithmen - Linear Maximum Margin Classifier

Abstand Punkt p zur Hyperebene:

$$D = \frac{|(wx_p) \pm b|}{||w||} = \frac{|w_1 x_{1p} + w_2 x_{2p} + \cdots + w_n x_{np} \pm b|}{\sqrt{w_1^2 + w_2^2 + \cdots + w_3^2}} \tag{1}$$

Da Punkt p ein Stützvektor gilt:

$$d(x_p, w, b) = w^T x_p + b = i_F = 1 \tag{2}$$

Es folgt:

$$D = \frac{1}{\sqrt{w_1^2 + w_2^2 + \cdots + w_3^2}} \tag{3}$$

Algorithmen - Linear Maximum Margin Classifier

Abstand Punkt p zur Hyperebene:

$$D = \frac{|(wx_p) \pm b|}{||w||} = \frac{|w_1 x_{1p} + w_2 x_{2p} + \cdots + w_n x_{np} \pm b|}{\sqrt{w_1^2 + w_2^2 + \cdots + w_3^2}} \quad (1)$$

Da Punkt p ein Stützvektor gilt:

$$d(x_p, w, b) = w^T x_p + b = i_F = 1 \quad (2)$$

Es folgt:

$$D = \frac{1}{\sqrt{w_1^2 + w_2^2 + \cdots + w_3^2}} \quad (3)$$

Die **Minimierung von $\mathbf{w^T w}$** (Äquivalent zu Minimierung von $\sqrt{w^T w}$) unter der Bedingung $y_i[w^T x_i + b] \geq 1$ für jedes Trainingselement, **maximiert** den Abstand **D**.

Algorithmen - Linear Maximum Margin Classifier

Nicht-lineares Optimierungsproblem mit Ungleichungsbedingung:

$$\textit{Minimiere } f(x)$$

$$\textit{unter der Nebenbedingung } g(x) \leq 0$$

Algorithmen - Linear Maximum Margin Classifier

Nicht-lineares Optimierungsproblem mit Ungleichungsbedingung:

$$\textit{Minimiere } f(x)$$

$$\textit{unter der Nebenbedingung } g(x) \leq 0$$

Lagrange-Dualität mit Lagrange-Multiplikatoren α_i:

$$L(w, b, \alpha) = \underbrace{\frac{1}{2} w^T w}_{f(x)=M^{-1}=(2D)^{-1}} - \sum_{i=1}^{I} \alpha_i \underbrace{\{y_i[w^T x_i + b] - 1\}}_{g(x)=y_i[w^T x_i+b]-1\geq 0} \quad (4)$$

Algorithmen - Linear Maximum Margin Classifier

Nicht-lineares Optimierungsproblem mit Ungleichungsbedingung:

$$\textit{Minimiere } f(x)$$

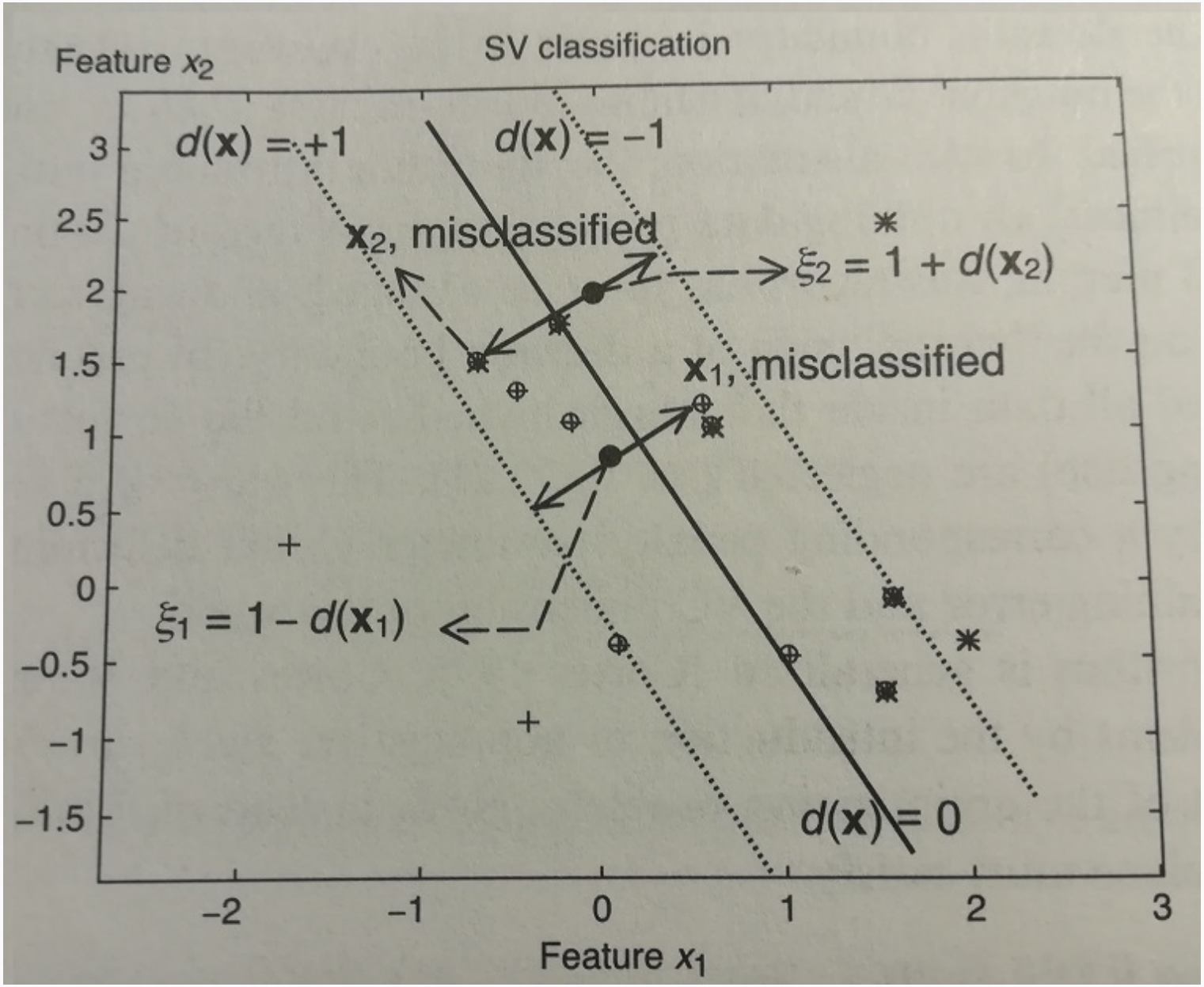
$$\textit{unter der Nebenbedingung } g(x) \leq 0$$

Lagrange-Dualität mit Lagrange-Multiplikatoren α_i:

$$L(w, b, \alpha) = \underbrace{\frac{1}{2} w^T w}_{f(x)=M^{-1}=(2D)^{-1}} - \sum_{i=1}^{l} \alpha_i \underbrace{\{y_i[w^T x_i + b] - 1\}}_{g(x)=y_i[w^T x_i+b]-1\geq 0} \quad (4)$$

Der Sattelpunkt von $L(w, b, \alpha)$ ist das gesuchte Minimum von $f(x)$ und damit der größte Abstand D.

Algorithmen - Linear Soft Margin Classifier



Algorithmen - Linear Soft Margin Classifier

Erweiterung der Bedingung $y_i[w^T x_i + b] \geq 1$ mit lockeren Variablen $\xi_i \quad (i = 0, l)$:

$$y_i[w^T x_i + b] \geq 1 - \xi_i, \qquad i = 1, l \tag{5}$$

$$\xi_i \geq 0 \tag{6}$$

Algorithmen - Linear Soft Margin Classifier

Erweiterung der Bedingung $y_i[w^T x_i + b] \geq 1$ mit lockeren Variablen $\xi_i \quad (i = 0, l)$:

$$y_i[w^T x_i + b] \geq 1 - \xi_i, \qquad i = 1, l \tag{5}$$

$$\xi_i \geq 0 \tag{6}$$

Einführung eines frei wählbaren Strafparameters C (Überlappungsfehler) in der zu minimierenden Funktion:

$$J(w, \xi) = \frac{1}{2} w^T w + C \left(\sum_{i=1}^{l} \xi \right) \tag{7}$$

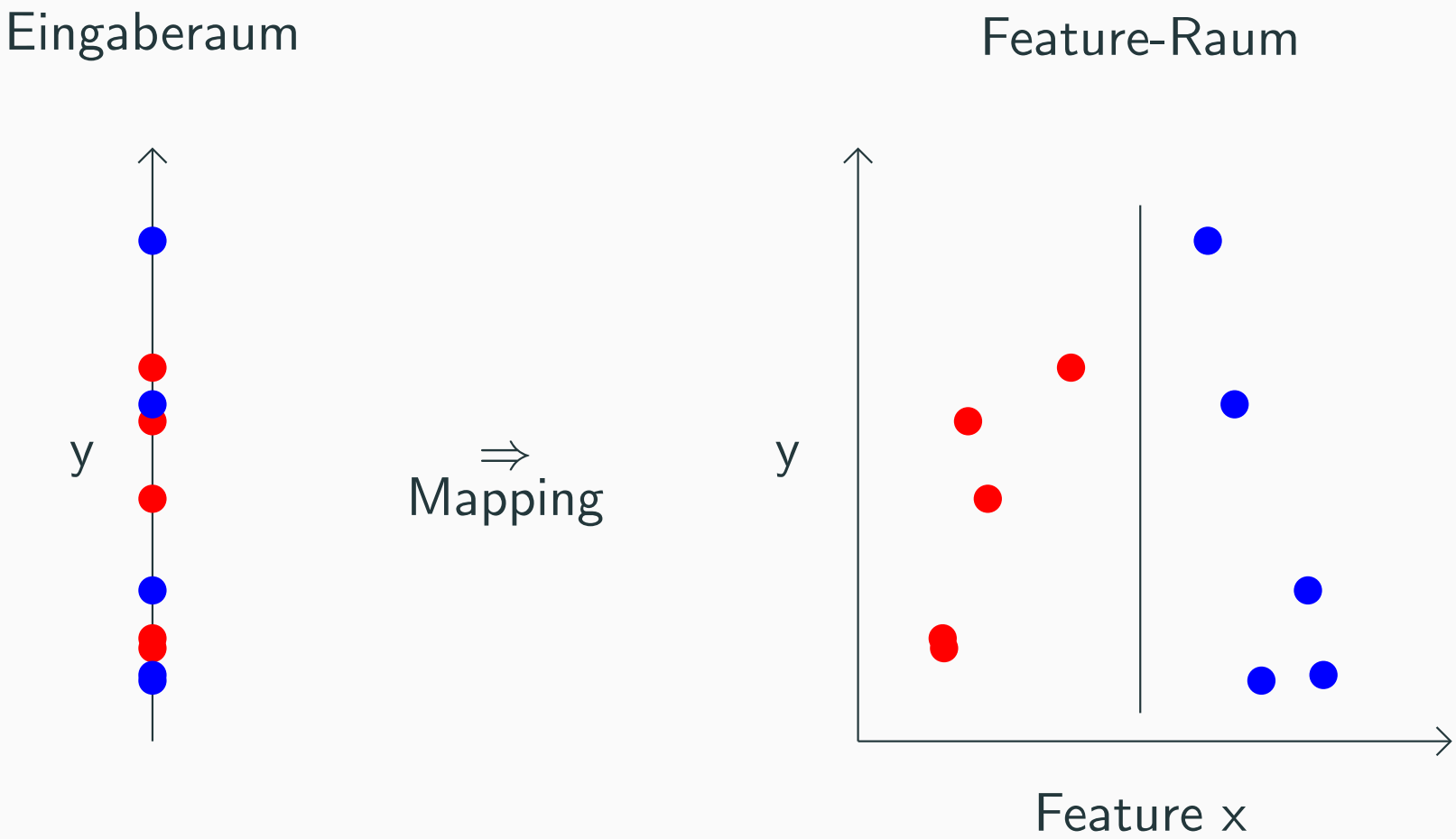
Algorithmen - Linear Soft Margin Classifier

Lagrange-Dualität mit Lagrange-Multiplikatoren α_i und β_i

$$\begin{aligned} L(w, b, \xi, \alpha, \beta) =& \frac{1}{2} w^T w + C \left(\sum_{i=1}^{I} \xi \right) \\ & - \sum_{i=1}^{I} \alpha_i \{ [w^T x_i + b] - 1 + \xi \} \\ & - \sum_{i=1}^{I} \beta_i \xi_i \end{aligned} \tag{8}$$

Der Sattelpunkt von $(w, b, \xi, \alpha, \beta)$ ist das gesuchte Minimum von $f(x)$ und damit der größte Abstand D.

Algorithmen - Nonlinear Classifier



Erhöhung der Dimension!

Algorithmen - Nonlinear Classifier

Erweiterung Eingaberaum auf Feature-Raum

$$R^n \rightarrow R^f \quad \text{mit Mapping} \quad \Phi(x) \tag{9}$$

$$\text{für Vektor} \quad x = [x_1, x_2, x_3]$$

Algorithmen - Nonlinear Classifier

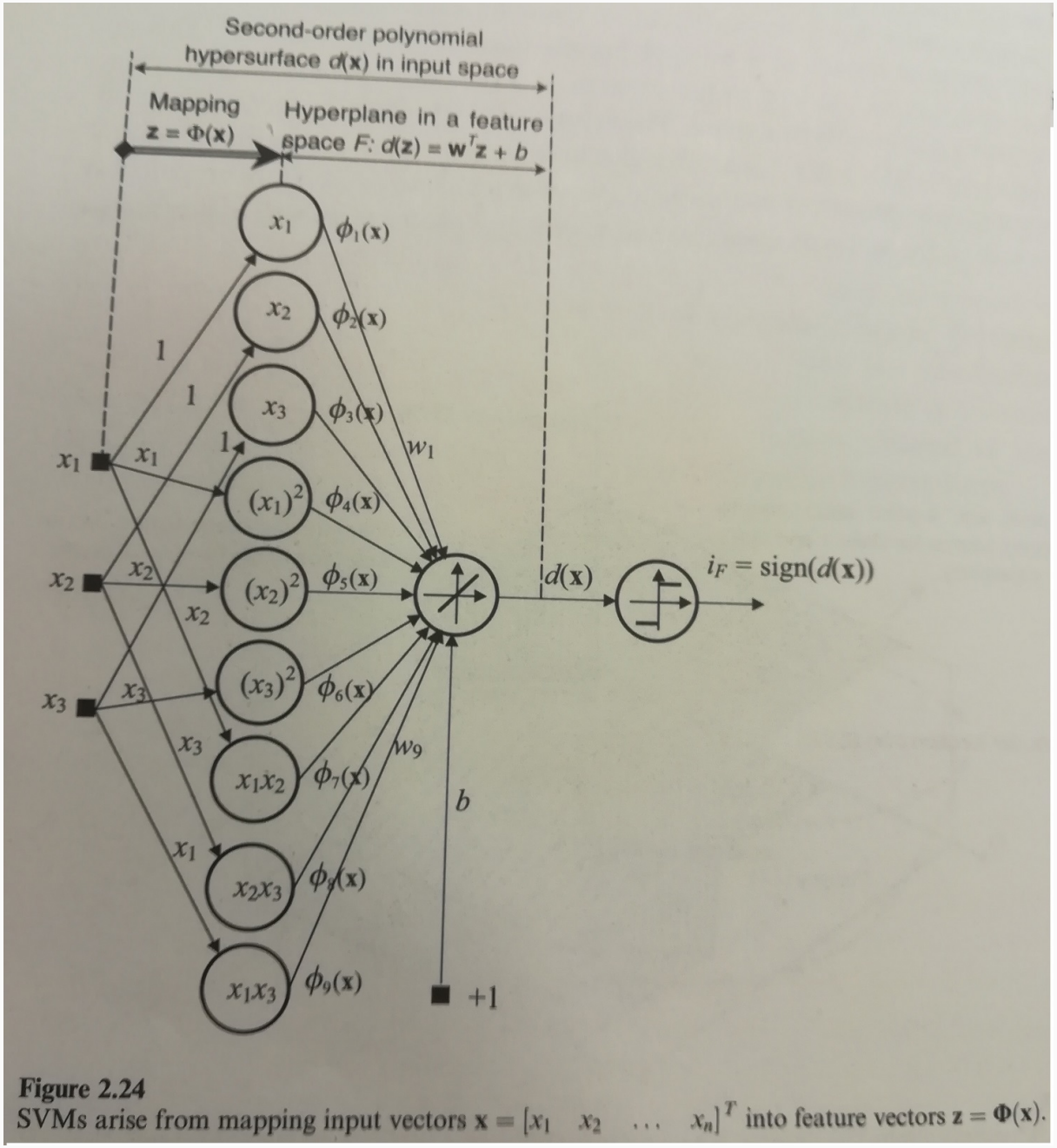


Figure 2.24
SVMs arise from mapping input vectors $\mathbf{x} = [x_1 \quad x_2 \quad \ldots \quad x_n]^T$ into feature vectors $\mathbf{z} = \Phi(\mathbf{x})$.

Algorithmen - Nonlinear Classifier

Erweiterung Eingaberaum auf Feature-Raum

$$R^n \rightarrow R^f \quad \text{mit Mapping} \quad \Phi(x) \tag{9}$$

$$\text{für Vektor} \quad x = [x_1, x_2, x_3]$$

- Lineare Hyperebene im Feature-Raum
- Nicht-lineare Hyperoberfläche im Eingaberaum

Algorithmen - Nonlinear Classifier

Erweiterung Eingaberaum auf Feature-Raum

$$R^n \rightarrow R^f \quad \text{mit Mapping} \quad \Phi(x) \tag{9}$$

$$\text{für Vektor} \quad x = [x_1, x_2, x_3]$$

- Lineare Hyperebene im Feature-Raum
- Nicht-lineare Hyperoberfläche im Eingaberaum

Probleme:

- Finden von $\Phi(x)$
- Explosion der Dimension

Algorithmen - Nonlinear Classifier

Beispiel: $\Phi(x)$

Algorithmen - Nonlinear Classifier

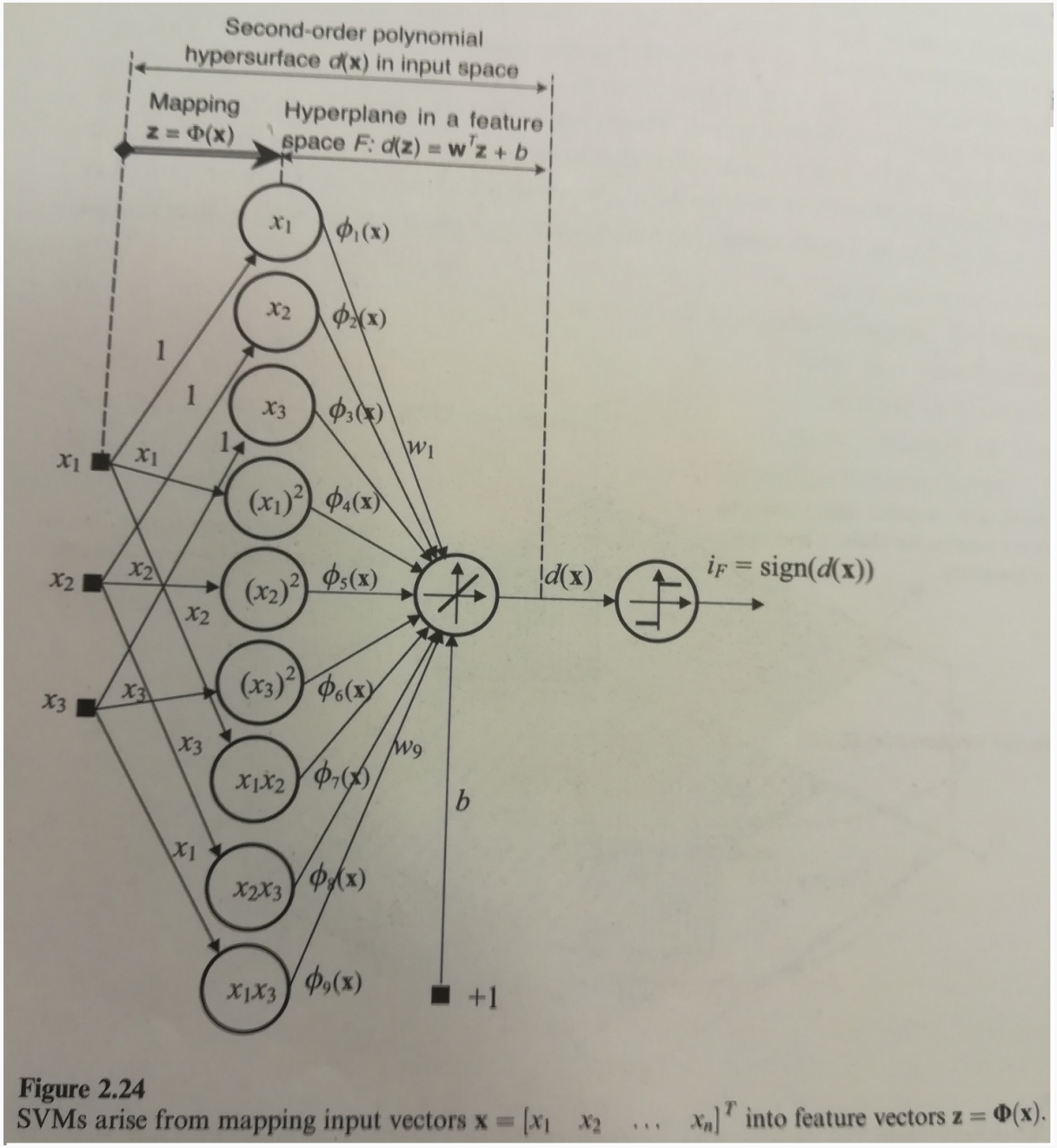


Figure 2.24
SVMs arise from mapping input vectors $\mathbf{x} = [x_1 \quad x_2 \quad \ldots \quad x_n]^T$ into feature vectors $\mathbf{z} = \Phi(\mathbf{x})$.

Algorithmen - Nonlinear Classifier

Beispiel: $\Phi(x)$

Algorithmen - Nonlinear Classifier

Beispiel: $\Phi(x)$

für Eingaberaum R^{256}

ergibt sich Feature-Raum $R^{33.152}$

Algorithmen - Nonlinear Classifier

Beispiel: $\Phi(x)$

für Eingaberaum R^{256}

ergibt sich Feature-Raum $R^{33.152}$

$$L(w, b, \alpha) = \underbrace{\frac{1}{2} w^T w}_{\text{Funktion in } R^n} - \sum_{i=1}^{l} \alpha_i \cdot \text{Nebenbedingung}$$

Algorithmen - Nonlinear Classifier

Beispiel: $\Phi(x)$

für Eingaberaum R^{256}

ergibt sich Feature-Raum $R^{33.152}$

$$L(w, b, \alpha) = \underbrace{\frac{1}{2} w^T w}_{\text{Funktion in } R^n} - \sum_{i=1}^{l} \alpha_i \cdot \text{Nebenbedingung}$$

Kernel Funktionen $K(x_i, x_j)$ als Funktion im Eingaberaum:

$$K(x_i, x_j) = \Phi^T(x_i)\Phi(x_j)$$

Anwendung

Anwendung - Bezug zum Task 7

SemEval-2018 task 7 Semantic Relation Extraction and Classification in Scientific Papers

- Klassifikation von semantischen Beziehungen
- Analyse Abstracts wissenschaftlicher Paper
- 6 verschiedene Klassen
- 2 Teil Task
  - Task a mit bereits extrahierten gelabelten semantischen Beziehungen
  - Task b mit Extraktion der semantischen Beziehungen

Anwendung

Problem:

1. Wörter in Vektoren konvertieren
   - word2Vec
   - elmo
2. Welche Feature zum Platzieren im Feature-Raum?

Zwei frameworks für SVM's:

- LIBSVM `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`
  - Datenformat: <label> <index1>:<value1> <index2>:<value2> ...
- SVM^{light} `http://svmlight.joachims.org/`

Ende

Ende

Danke für eure Aufmerksamkeit

Fragen?

Gibt es noch Fragen?

Literatur

Literatur

- Kecman, V. (2001). Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. MIT press.
- http://www.statsoft.com/Textbook/Support-Vector-Machines (Stand: 09.05.2018)
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/ (Stand:16.05.2018)
- http://svmlight.joachims.org/ (Stand:16.05.2018)