7. Spezielle Klassifikatoren

7.1. Lineare Diskriminanzfunktionen

Vorgehensweise der Kapitel 3-6: Schätzung der klassenspezifischen WVen → Entscheidungsfunktionen (Diskriminanzfunktionen) $k_i(\mathbf{m})$.

Vorgehensweise hier: Parametrisierte Entscheidungsfunktionen (Diskriminanzfunktionen) $k_i(\mathbf{m};\boldsymbol{\theta})$ werden direkt an die Daten D angepasst, mit dem Ziel, eine möglichst hohe Klassifikationsleistung zu erzielen.
Speziell sollen lineare Diskriminanzfunktionen verwendet werden.

Die folgenden Verfahren benötigen kein Vorwissen über die zugrunde liegenden WVen.

7.1. Lineare Diskriminanzfunktionen

Beispiel: $c = 2$

Diskriminanzfunktion: $k(\mathbf{m}) = \mathbf{w}^{\mathrm{T}}\mathbf{m} + b$

Entscheidung:

$$\hat{\omega} = \hat{\omega}(\mathbf{m}) := \begin{cases} \omega_1 & \text{für } k(\mathbf{m}) > 0 \\ \omega_2 & \text{für } k(\mathbf{m}) < 0 \\ \text{egal} & \text{für } k(\mathbf{m}) = 0 \end{cases}$$

Entscheidungsgrenze H: $k(\mathbf{m}) = 0$ $(d-1)$-dimensionale Hyperebene trennt den Raum in zwei Hälften.

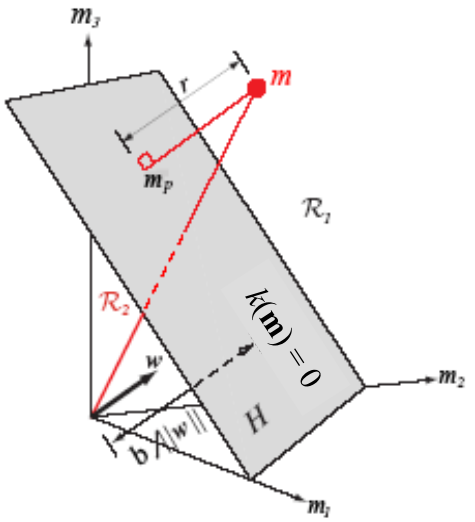
$$\mathrm{H} := \{\mathbf{m} \mid \mathbf{w}^{\mathrm{T}}\mathbf{m} + b = 0\}$$

bestimmt den Ursprungsabstand

bestimmt die Richtung

$$\frac{k(\mathbf{m})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^{\mathrm{T}}}{\|\mathbf{w}\|}\mathbf{m} + \frac{b}{\|\mathbf{w}\|} = \underbrace{D(\mathbf{m}, \mathrm{H})}$$

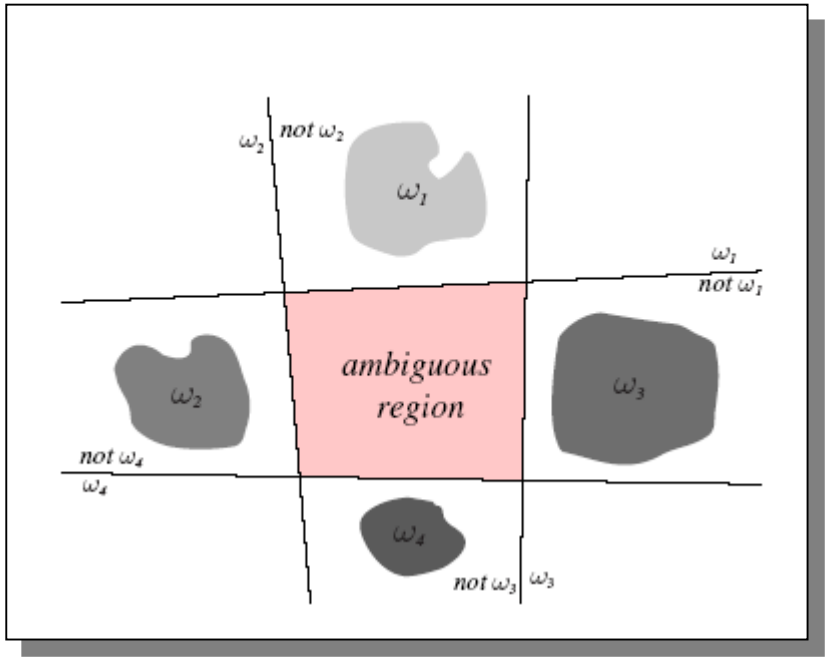
vorzeichen behafteter Abstand von $\mathbf{m}$ zu H

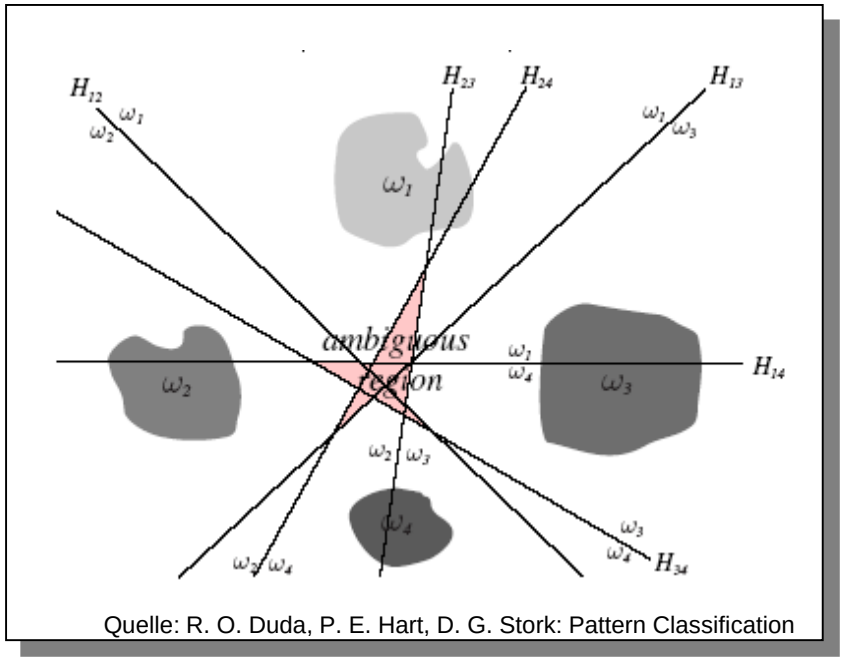


Quelle: R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

7.1. Lineare Diskriminanzfunktionen

Beispiele: $c > 2$





Quelle: R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

Ansatz 1: Lineare Diskriminanz-funktionen zur Unterscheidung zwischen ω_i und *nicht* ω_i.

Ansatz 2: Lineare Diskriminanz-funktionen zur Unterscheidung aller Paare ω_i, ω_j .

→ $c(c-1)/2$ Diskriminanzfunktionen

7.1. Lineare Diskriminanzfunktionen

Ansatz 3: Lineare Maschine

c Diskriminanzfunktionen: $k_i(\mathbf{m}) = \mathbf{w}_i^{\mathrm{T}}\mathbf{m} + b_i,\ i = 1,\ldots,c$

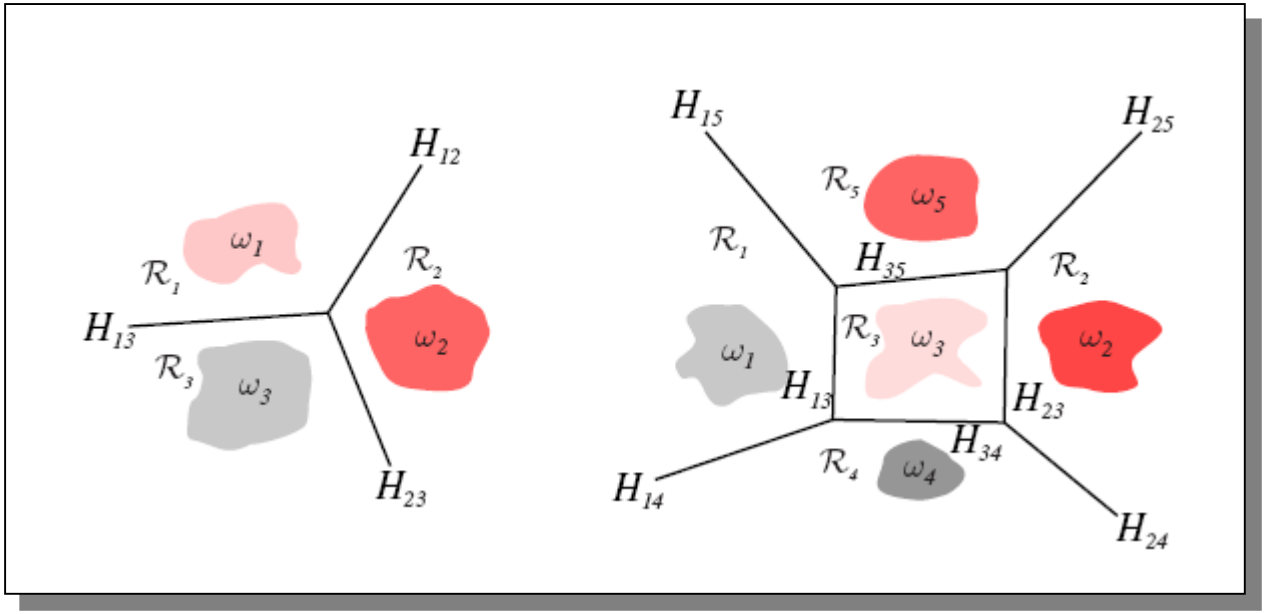
Entscheidung:

$$\hat{\omega} := \omega_i \quad \Leftrightarrow \quad k_i(\mathbf{m}) > k_j(\mathbf{m}),\ \forall i \neq j$$

Bei „Unentschieden“ wird beliebige Entscheidung getroffen.

Entscheidungsgrenzen H_{ij} folgen aus: $k_i(\mathbf{m}) = k_j(\mathbf{m})$

H_{ij} sind Teile von Hyperebenen.



Quelle: R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

7.1. Lineare Diskriminanzfunktionen

Trennflächen: $\mathrm{H}_{ij} = \left\{ \mathbf{m} \middle| \, k_i(\mathbf{m}) = k_j(\mathbf{m}) \wedge k_i(\mathbf{m}) > k_l(\mathbf{m}) \; \forall l \notin \{i, j\} \right\}$

$$\mathrm{H}_{ij} \subseteq \left\{ \mathbf{m} \middle| \left(\mathbf{w}_i - \mathbf{w}_j \right)^{\mathrm{T}} \mathbf{m} + \left(b_i - b_j \right) = 0 \right\}$$

→ Normalenvektor von H_{ij}: $\mathbf{w}_i - \mathbf{w}_j$

→ Distanz zwischen $\mathbf{m}$ und H_{ij}:

$$D(\mathbf{m}, \mathrm{H}_{ij}) = \frac{k_i(\mathbf{m}) - k_j(\mathbf{m})}{\left\| \mathbf{w}_i - \mathbf{w}_j \right\|}$$

Bemerkungen:

- Entscheidungsgebiete sind konvex.
- Entscheidungsgebiete sind einfach zusammenhängend.
- Anzahl der Entscheidungsebenen ist i.d.R. kleiner als $c(c-1)/2$, da i.d.R. nicht alle Klassen aneinander grenzen.

7.1. Lineare Diskriminanzfunktionen

Verallgemeinerung, $c = 2$:

$$k(\mathbf{m}) = \mathbf{w}^{\mathrm{T}}\mathbf{m} + b = b + \sum_{i=1}^{d} w_i m_i$$

$$+ w_{ij} m_i m_j$$

Quadratische Diskriminanzfunktion: $$k(\mathbf{m}) = b + \sum_{i=1}^{d} w_i m_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} m_i m_j$$

Hyperquadriken $k(\mathbf{m}) = 0$ entsprechen den Entscheidungsgrenzen der Bayes‘schen Klassifikation bei klassenbedingt normalverteilten Merkmalen.

Quadratische Diskriminanzfunktion $+ w_{ijk} m_i m_j m_k$

$$+ w_{ijkl} m_i m_j m_k m_l$$

$\vdots$

Polynomiale Diskriminanzfunktionen

allg. Funktionen statt Monome

Verallgemeinerte lineare Diskriminanzfunktionen: $$k(\mathbf{m}) = \sum_{i=0}^{d^*} a_i y_i(\mathbf{m})$$

Verallgemeinerung, $c = 2$**:**

Bemerkungen:

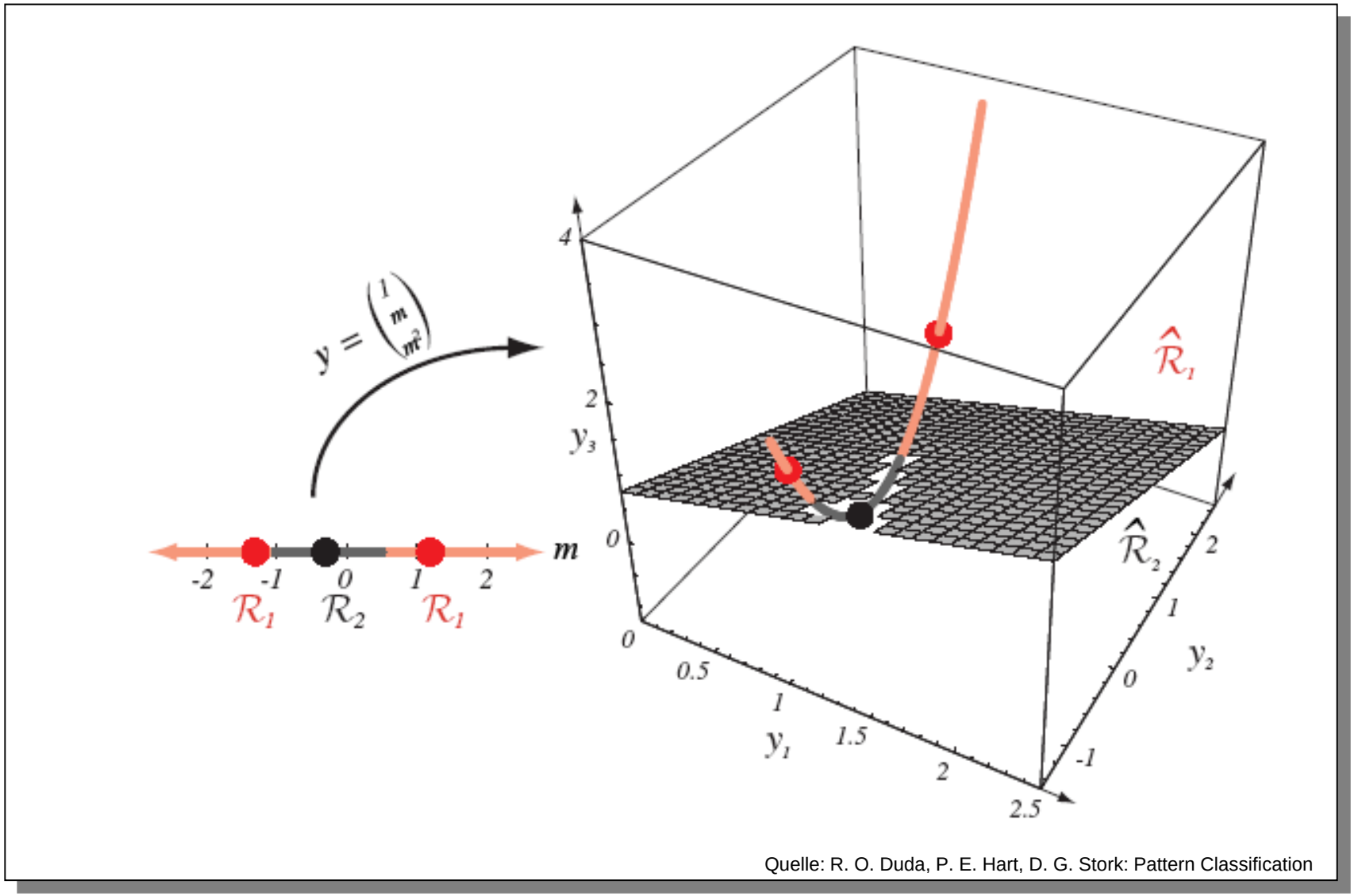
- Die quadratische Diskriminanzfunktion hat zusätzlich $d(d+1)/2$ Parameter.
- Die verallgemeinerte lineare Diskriminanzfunktionen ist zwar nicht linear bezüglich $\mathbf{m}$ aber linear bezüglich $\mathbf{y}$.

$$k(\mathbf{m}) = \sum_{i=0}^{d^*} a_i y_i(\mathbf{m}) = \underbrace{[a_0, \ldots, a_{d^*}]}_{\mathbf{a}^T} \underbrace{\begin{bmatrix} 1 \\ y_1(\mathbf{m}) \\ \vdots \\ y_{d^*}(\mathbf{m}) \end{bmatrix}}_{\mathbf{y}} = \mathbf{a}^T\mathbf{y}$$

- Im Fall $\mathbf{y} = [1, m_1, \ldots, m_d]^T$ spricht man von einem erweiterten Merkmalsvektor (augmented feature vector).

7.1. Lineare Diskriminanzfunktionen

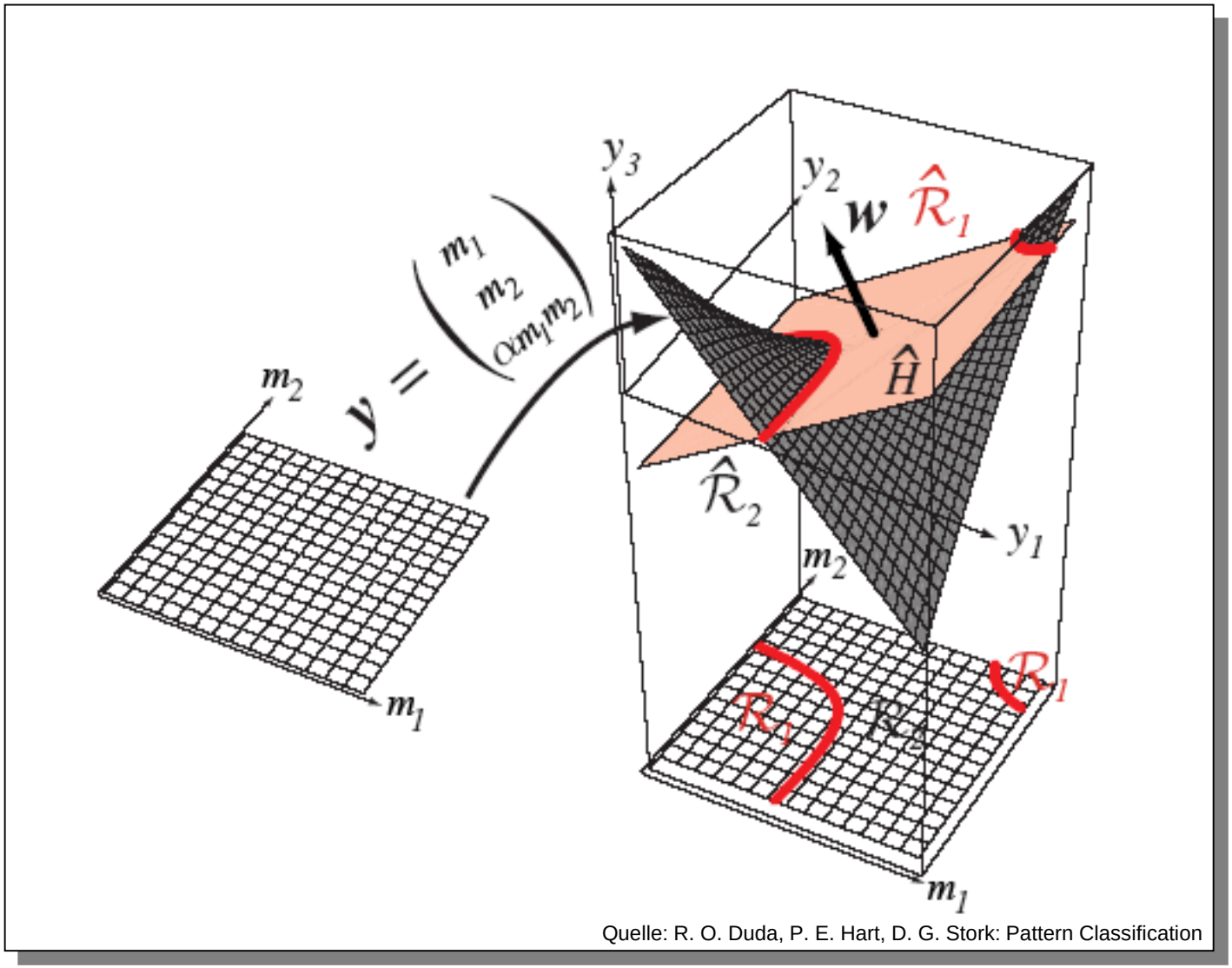
Beispiel: $d = 1, c = 2$



Quelle: R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

7.1. Lineare Diskriminanzfunktionen

Beispiel: $d = 2, c = 2$



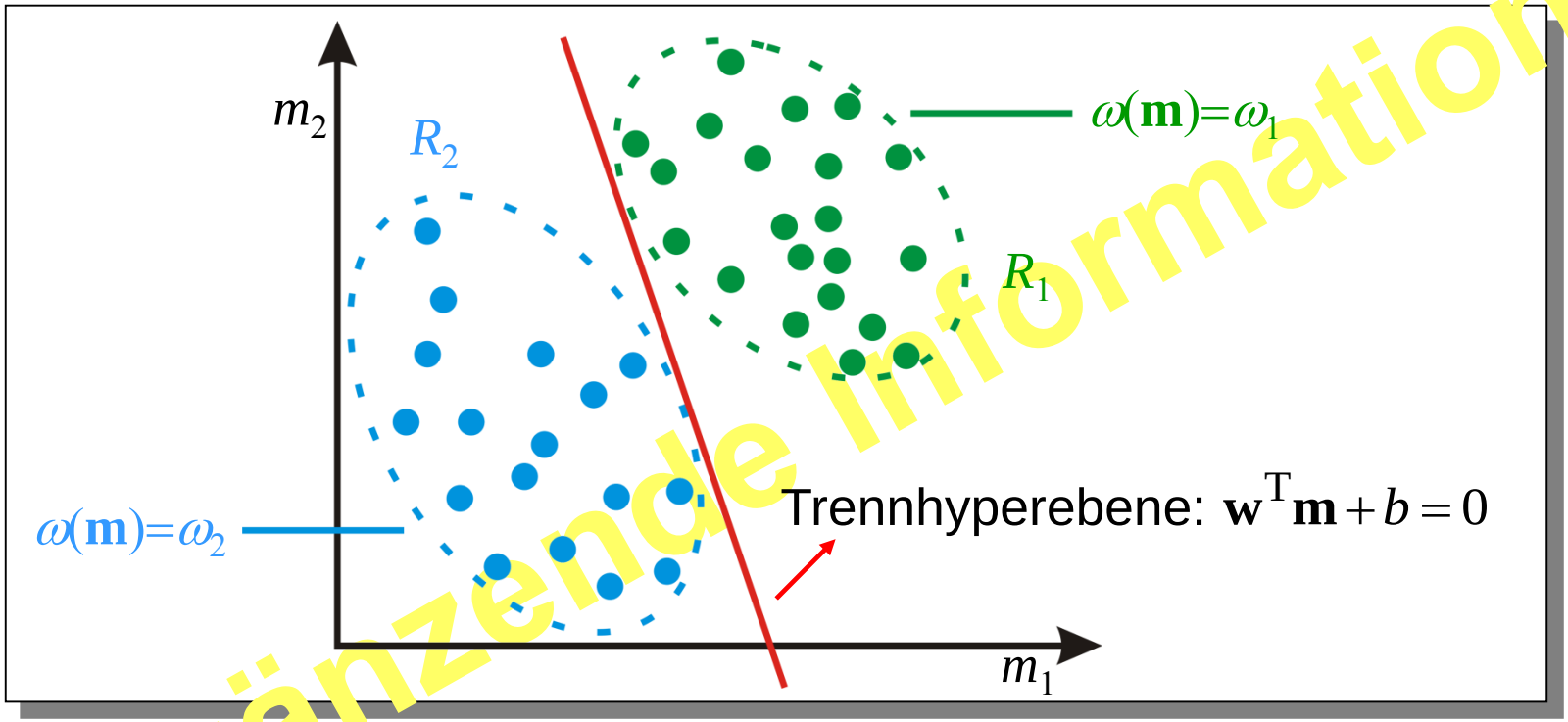
Quelle: R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

Bemerkungen:

- Obwohl die Entscheidungsgebiete im $\mathbf{y}$-Merkmalsraum konvex sind, sind die korrespondierenden Entscheidungsgebiete im $\mathbf{m}$-Merkmalsraum u.U. sehr kompliziert geformt.
- Die Verallgemeinerung des Merkmalsvektros $\mathbf{m} \rightarrow \mathbf{y}$ führt auf eine Dimensionserhöhung $d \rightarrow d^*+1 > d$ mit entsprechenden Konsequenzen für die Größe N der Lernstichprobe.

7.2. Perzeptron

Voraussetzungen: - Zwei Klassen ($c = 2$)
- Lernstichprobe D linear separierbar, |D|=N



Gesucht: Trennhyperebene $\mathbf{w}^T\mathbf{m}+b=0$, welche die Lernstichprobe D korrekt klassifiziert.

Hilfsvariable: $$z := \begin{cases} +1 & \text{für} \quad \omega(\mathbf{m}) = \omega_1 \\ -1 & \text{für} \quad \omega(\mathbf{m}) = \omega_2 \end{cases}$$

Perzeptron-Algorithmus:

Gegeben sei eine linear trennbare Stichprobe D, $c = 2$ und

eine Lernrate $\eta > 0$

$\mathbf{w}_0 := \mathbf{0};\ b_0 := 0;\ k := 0;\ \ R := \max_{1 \le i \le N} \{\|\mathbf{m}_i\|\}$

Repeat

for i =1 to N

if $z_i\left(\mathbf{w}_k^{\mathrm{T}}\,\mathbf{m}_i + b_k\right) \le 0$ then

$\mathbf{w}_{k+1} := \mathbf{w}_k + \eta z_i \mathbf{m}_i$

$b_{k+1} := b_k + \eta z_i R^2$

$k := k + 1$

end if

end for

until (keine Fehler mehr in der for-Schleife)

return ($\mathbf{w}_k, b_k$) (Wobei k die Zahl der beim Lernen aufgetretenen Fehler ist.)

Bemerkungen:

- Unter o.g. Voraussetzungen ist die Konvergenz gesichert.
- Sei das Resultat $\mathbf{w}, b$ normiert, d.h. $\|\mathbf{w}\| = 1$ und sei $\gamma > 0$ mit $z_i\left(\mathbf{w}^{\mathrm{T}}\mathbf{m}_i + b\right) \geq \gamma$ für $i = 1, \ldots N$, dann werden beim Lernen höchstens

$$k \leq \left(\frac{2R}{\gamma}\right)^2$$

  Fehler gemacht (Novikoff).
- Der Perzeptron-Algorithmus macht den Trainingsfehler zu Null.
- Eine kleiner Trainingsfehler garantiert keinen kleinen Testfehler.

7.3. Klassifikation mittels linearer Regression

$$\mathrm{D} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_N\}$$

Ansatz: $k_i(\mathbf{m}) = \mathbf{a}_i^{\mathrm{T}}\mathbf{m} \qquad \mathbf{a}_i \in \mathbb{R}^d, \quad i = 1,\ldots,c$

Wunsch: $k_i(\mathbf{m}) \overset{!}{=} \begin{cases} 1 & \text{für } \omega(\mathbf{m}) = \omega_i \\ 0 & \text{sonst} \end{cases}$

Lernstichprobe: $\mathbf{m}_k^{\mathrm{T}}\mathbf{a}_i \overset{!}{=} \begin{cases} 1 & \text{für } \omega(\mathbf{m}_k) = \omega_i \\ 0 & \text{sonst} \end{cases} \Big\} =: z_{ki} \qquad k = 1,\ldots,N$

$$\underbrace{\begin{bmatrix} \mathbf{m}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{m}_N^{\mathrm{T}} \end{bmatrix}}_{\mathbf{M}} \mathbf{a}_i \overset{!}{=} \underbrace{\begin{bmatrix} z_{1i} \\ \vdots \\ z_{Ni} \end{bmatrix}}_{\text{Zugehörigkeitsvektor } \mathbf{z}_i;\ \text{bekannt für } \mathrm{D}}$$

7.3. Klassifikation mittels linearer Regression

Für $N > d$ ist das Gleichungssystem überbestimmt,
Näherungslösung mit dem *Kleinsten Quadrate Ansatz*:

$$e := \| \mathbf{M}\mathbf{a}_i - \mathbf{z}_i \|^2 \overset{!}{\rightarrow} \text{ minimal}$$

$$e = \mathbf{a}_i^T \mathbf{M}^T \mathbf{M}\mathbf{a}_i - \mathbf{z}_i^T \mathbf{M}\mathbf{a}_i - \mathbf{a}_i^T \mathbf{M}^T \mathbf{z}_i - \mathbf{z}_i^T \mathbf{z}_i$$

$$\nabla_{\mathbf{a}_i} e = 2\mathbf{M}^T \mathbf{M}\mathbf{a}_i - 2\mathbf{M}^T \mathbf{z}_i \overset{!}{=} 0$$

$$\hat{\mathbf{a}}_i = \left(\mathbf{M}^T \mathbf{M}\right)^{-1} \mathbf{M}^T \mathbf{z}_i \qquad i = 1,\ldots,c \qquad \left(\mathbf{M}^T \mathbf{M}\right)^{-1} \mathbf{M}^T : \text{pseudoinverse Matrix von } \mathbf{M}$$

$$k_i(\mathbf{m}) = \mathbf{z}_i^T \mathbf{M}\left(\mathbf{M}^T \mathbf{M}\right)^{-1} \mathbf{m} \qquad i = 1,\ldots,c$$

$$\mathbf{k}(\mathbf{m}) = \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_c^T \end{bmatrix} \mathbf{M}\left(\mathbf{M}^T \mathbf{M}\right)^{-1} \mathbf{m}$$

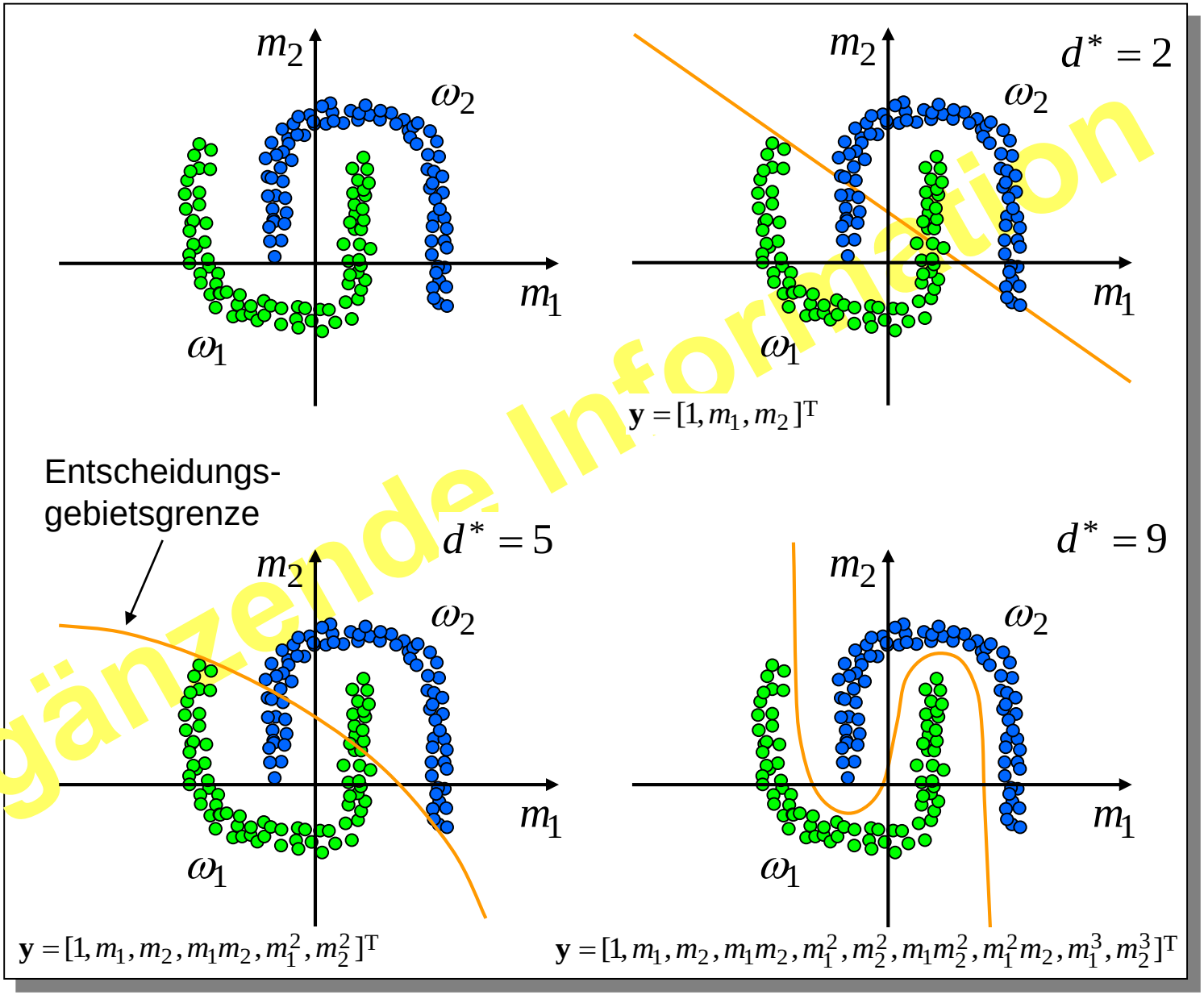
Ersetzt man $\mathbf{m}$ durch einen verallgemeinerten Merkmalsvektor $\mathbf{y}$, ($d \rightarrow d^*+1$), so können Nichtlinearitäten eingefügt werden, ohne die lineare Struktur des Klassifikators aufzugeben.

Bewertung gehonter Zylinderlaufbahnen im Kap. 6 wurde mit diesem Verfahren vorgenommen.

7.3. Klassifikation mittels linearer Regression

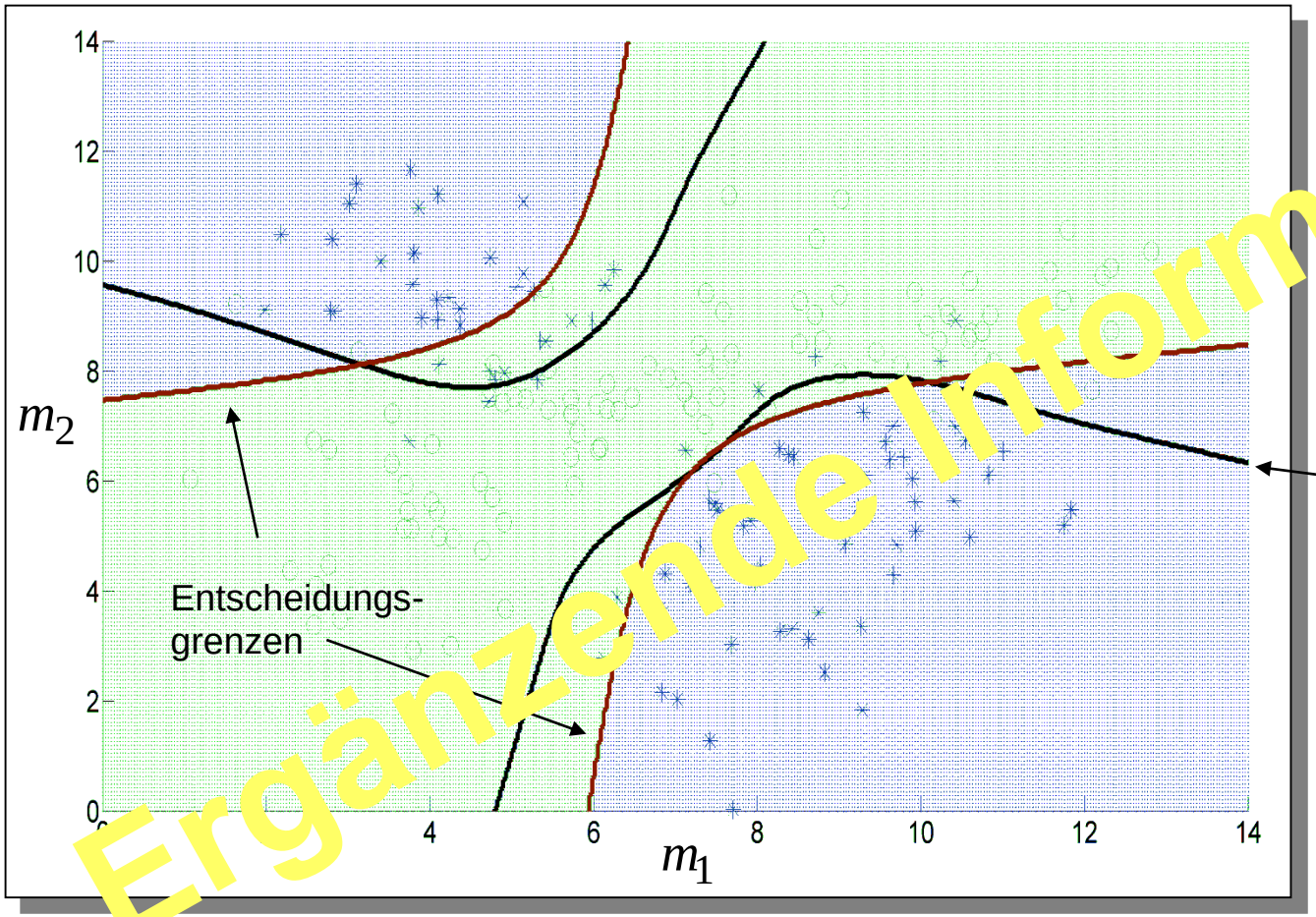
Beispiel:

$d = 2, c = 2$



7.3. Klassifikation mittels linearer Regression

Beispiel: Mit erweitertem Merkmalsvektor $\mathbf{y} = [1, m_1, m_2, m_1 m_2, m_1^2, m_2^2]^{\mathrm{T}}$

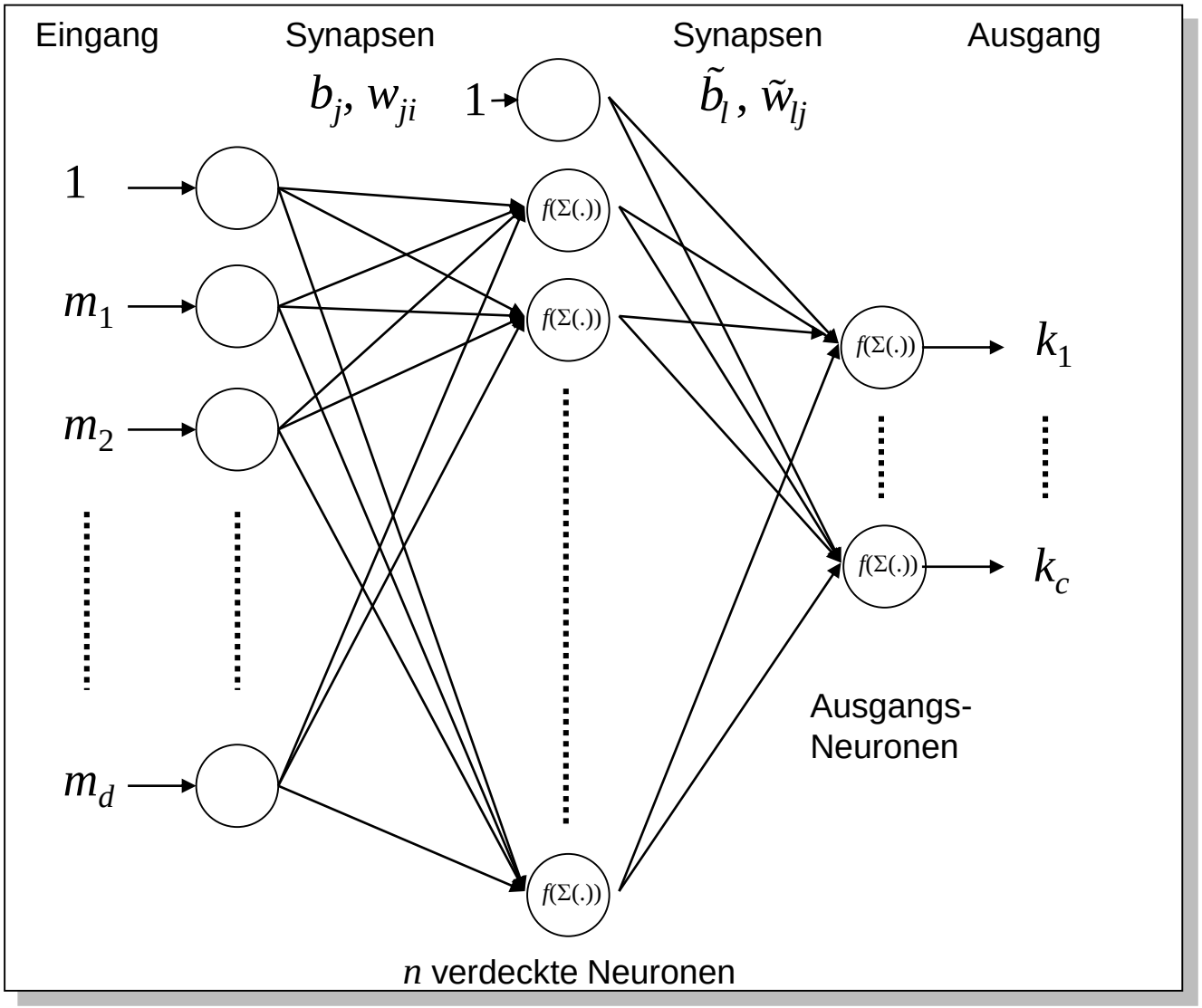


Parameter aus Lernstichprobe geschätzt.

Teststichprobe → Testfehler = 13,5% Asymptotischer Testfehler ≈ 14,96%

7.4. Künstliche Neuronale Netze

Spezielles künstliches Neuronales Netzwerk: **Feed-Forward Netzwerk** mit einer verdeckten Neuronenschicht:



Diskriminanzfunktionen, die durch ein **Feed-Forward-Netzwerk** mit einer verdeckten Neuronenschicht realisiert werden:

$$k_l(\mathbf{m}) = f\left(\sum_{j=1}^{n} \tilde{w}_{lj}\, f\left(\sum_{i=1}^{d} w_{ji} m_i + b_j\right) + \tilde{b}_l\right) \qquad l = 1,\ldots, c$$

$$= f\left(\tilde{\mathbf{w}}_l^{\mathrm{T}} \mathbf{h} + \tilde{b}_l\right) \quad \text{mit} \quad \mathbf{h} := \begin{bmatrix} f\left(\mathbf{w}_1^{\mathrm{T}} \mathbf{m} + b_1\right) \\ \vdots \\ f\left(\mathbf{w}_n^{\mathrm{T}} \mathbf{m} + b_n\right) \end{bmatrix}$$

Gebräuchlich:

$f(\xi) := \frac{1}{1+\mathrm{e}^{-\xi}}$

„Fermifunktion“

Es gilt: Jede stetige Funktion $\mathbf{m} \mapsto \mathbf{k}$ kann durch ein Feed-Forward-Netzwerk mit einer verdeckten Schicht dargestellt werden. Das kann anhand eines Satzes von Kolmogorov bewiesen werden, der besagt: Eine auf $[0, 1]^d$, $d > 1$ stetige Funktion $g(\mathbf{m})$ lässt sich darstellen in der Form:

$$g(\mathbf{m}) = \sum_{j=1}^{2d+1} \Xi_j \left(\sum_{i=1}^{d} \psi_{ij}(m_i)\right)$$

Ξ_j und ψ_{ij} sind i.Allg. nichtlineare Funktionen. → Feed-Forward-Netzwerke sind universelle Approximatoren für stetige Funktionen.

Training von Feed-Forward-Netzwerken: Iterative Minimierung des Fehlers:

$$e := \sum_{\nu=1}^{N} \| \mathbf{k}(\mathbf{m}_\nu) - \boldsymbol{\omega}(\mathbf{m}_\nu) \|^2$$

Gradientenverfahren:

$$\begin{cases} \Delta w_{ji} = -\alpha \dfrac{\partial e}{\partial w_{ji}} \\ \Delta \tilde{w}_{lj} = -\alpha \dfrac{\partial e}{\partial \tilde{w}_{lj}} \end{cases}$$

→ Backpropagation Algorithmus;

Details siehe z.B. R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

Vorteile:

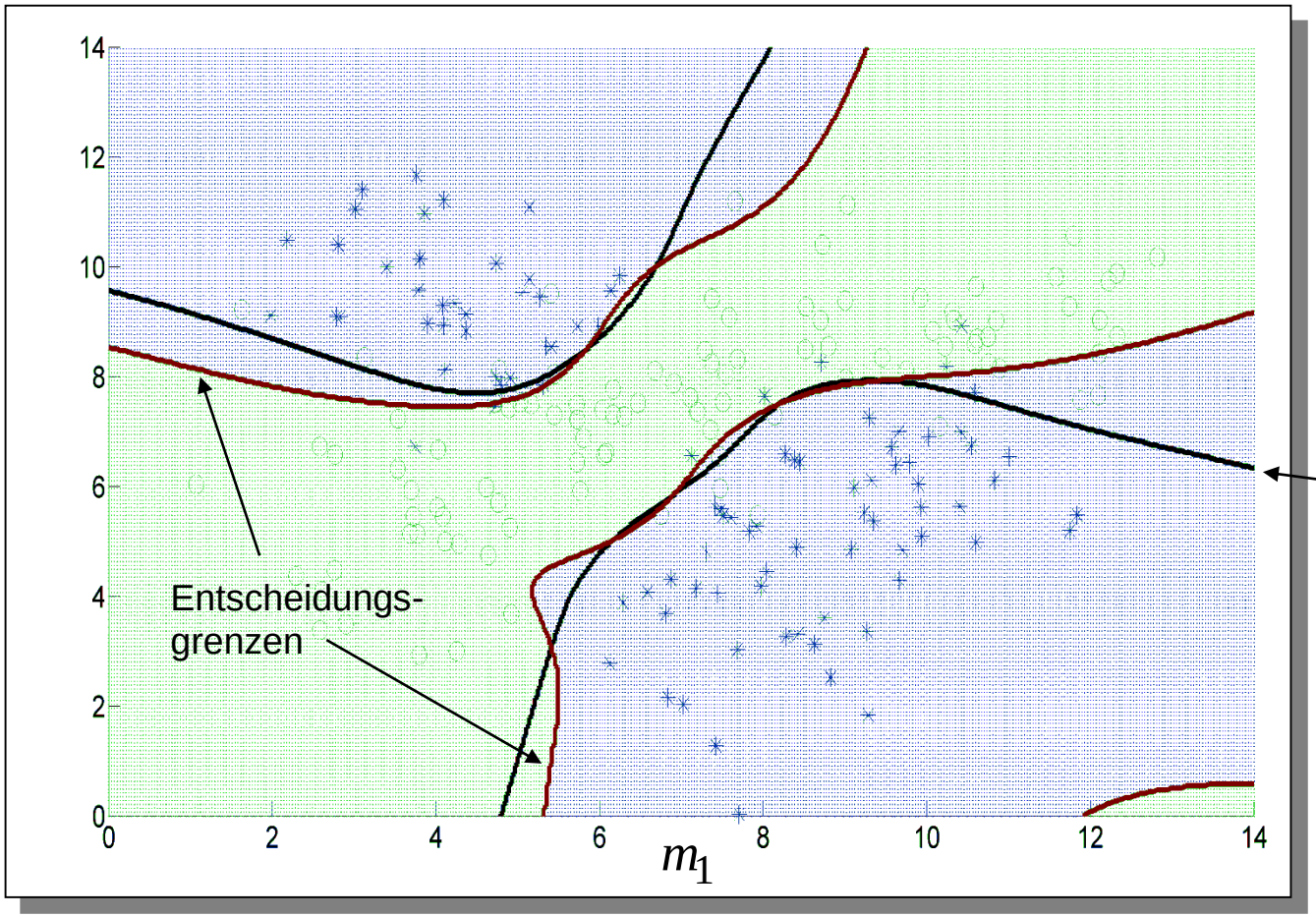
- Einfach zu konfigurieren, kein Vorwissen erforderlich.
- Beliebige Entscheidungsfunktionen lassen sich approximieren.
- Schnelle Klassifikation.

Nachteile:

- Große KNN ⟶ viele Parameter ⟶ große Lernstichprobe notwendig.
- Neigt zu Overfitting.
- Training großer KNN ist aufwändig.
- Fehlende Anhaltspunkte für die Dimensionierung des KNN
- KNN ist mathematisch weitgehend eine „Black Box“.
- Gefahr der Konvergenz gegen lokale Minima.

7.4. Künstliche Neuronale Netze

Beispiel: *Feed-forward-Netzwerk* mit 2 verdeckten Schichten mit je 10 Neuronen



$$f(\xi) = \frac{2}{1+e^{-2\xi}} - 1$$

Netzwerk mit Lernstichprobe trainiert.

Teststichprobe → Testfehler = 8,5% Asymptotischer Testfehler ≈ 12,42%

Beispiel: Kodierungsproblem; Autoencoder.

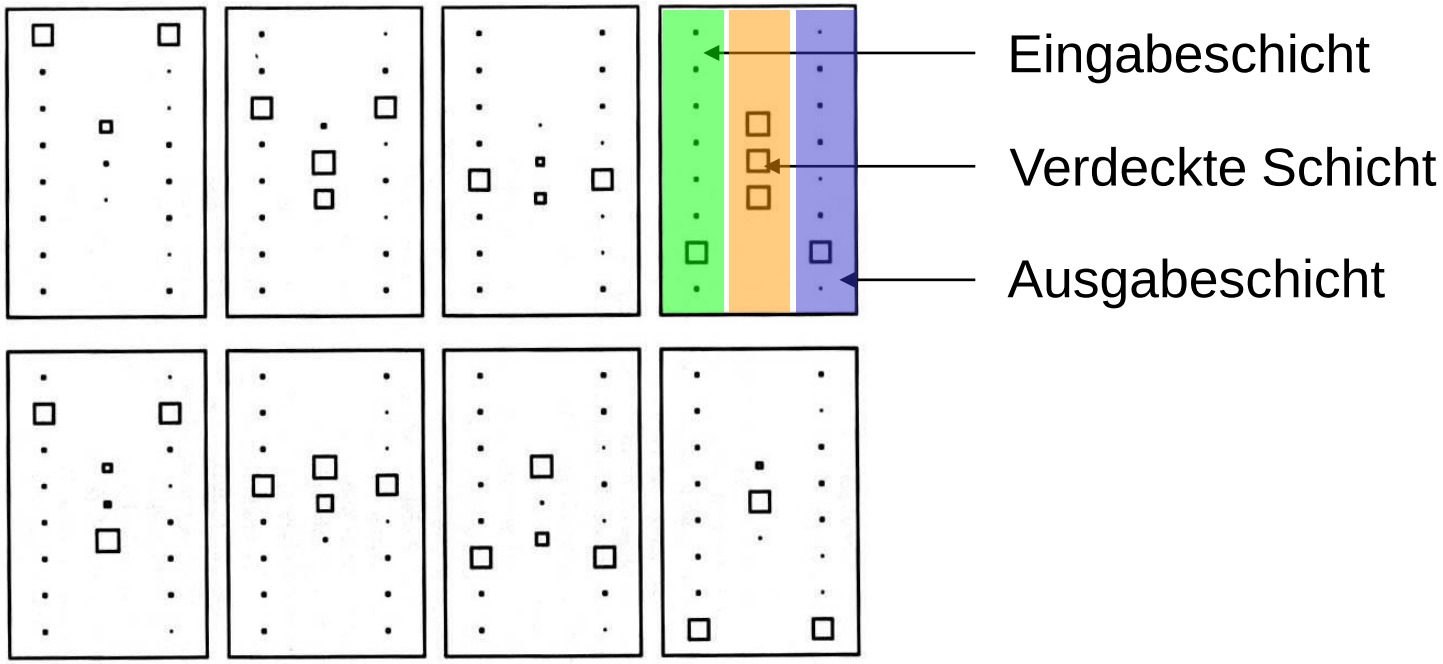
Quelle: H. Ritter, Th. Martinez, K. Schulten: Neuronale Netze, Addison-Wesley 1990

$$D = \{\mathbf{m}_1,\ldots,\mathbf{m}_8\} \qquad \mathbf{m}_i \in \left\{\tfrac{1}{10},\tfrac{9}{10}\right\}^8 \qquad \mathbf{m}_i := \left[\tfrac{1}{10},\ldots,\underbrace{\tfrac{9}{10}}_{i-\text{te Stelle}},\ldots,\tfrac{1}{10}\right]^T$$

Feed-Forward-Netzwerk mit einer verdeckten Schicht mit $n = 3$ Neuronen

Ziel: Ausgabeschicht soll Eingangsvektor reproduzieren

Problem: Innere Schicht stellt Engpass dar und muss eine Kodierung finden



7.4. Künstliche Neuronale Netze – Tiefe Netze

Gründe für tiefere Neuronale Netze:

- Tiefere Netze ermöglichen gleiche Approximationsfähigkeit wie flache Netze bei weniger Parametern: Es existieren Funktionen, die ein Netz mit n Schichten mit polynomieller Parameteranzahl (bezüglich der Zahl der Eingabeneuronen) darstellen kann, während bei n-1 Schichten eine exponentielle Parameteranzahl nötig wäre.
- Prinzipielle Möglichkeit geringerer Parameteranzahl → bessere Generalisierungsfähigkeit bzw. weniger Overfitting.
- Tiefe Netze können hierarchische Teil-Objekt-Beziehungen modellieren.
- Der menschliche visuelle Kortex ist ähnlich in mehreren hierarchischen Schichten aufgebaut.

Unterschied zu flachen neuronalen Netzen:

Es existiert keine scharfe Grenze, ab wann eine Netz tief ist.
1 verdeckte Schicht gilt als flach, 10 Schichten gelten bereits als sehr tief.

Quelle: J. Schmidhuber: Deep Learning in Neural Networks: An Overview

Schwierigkeiten und historische Probleme:

- Backpropagation Algorithmus problematisch bei tiefen Netzen.
  - Gradient verschwindet oder explodiert beim Rückpropagieren wegen exponentieller Änderung von Schicht zu Schicht.
  - Optimierung konvergiert oft in Sattelpunkten oder lokalen Optima.
- Tiefe Netze haben tendenziell viele Parameter und benötigen daher sehr viele Trainingsdaten.
- Zunächst fehlende Konzepte, wie praktische Probleme vorteilhaft in tiefen neuronalen Netzen repräsentiert werden können.
- Zunächst besser verständliche alternative Verfahren verfügbar, die ähnlich performant waren, insbesondere SVM.

Erfolgsgründe:

Größere Datenmengen vorhanden.

Höhere verfügbare Rechenleistung (GPUs): Durch die sehr hohe mögliche Anzahl an Trainingsschritten kann auch ein verschwindend kleiner Gradient ausreichen.

Verbesserung der Lernstrategie

- Unüberwachtes Vortrainieren.
- Stochastischer Gradientenabstieg mit Momentum und Gewichtsverfall.
- ReLU-Aktivierungsfunktion.

Spezialisierung der Architektur

- Für sequentielle Daten (z.B. Sprach- oder Handschrifterkennung): LSTM (*Long Short-Term Memory*), löst Problem des verschwindenden/ explodierenden Gradienten.
  Details siehe S. Hochreiter, J. Schmidhuber: Long short-term memory
- Für mehrdimensionale Daten (z.B. Bilddaten): *Convolutional Neural Network* (CNN) mit trainierbaren Filtern zur Faltung der Daten.

7.4. Künstliche Neuronale Netze – Tiefe Netze

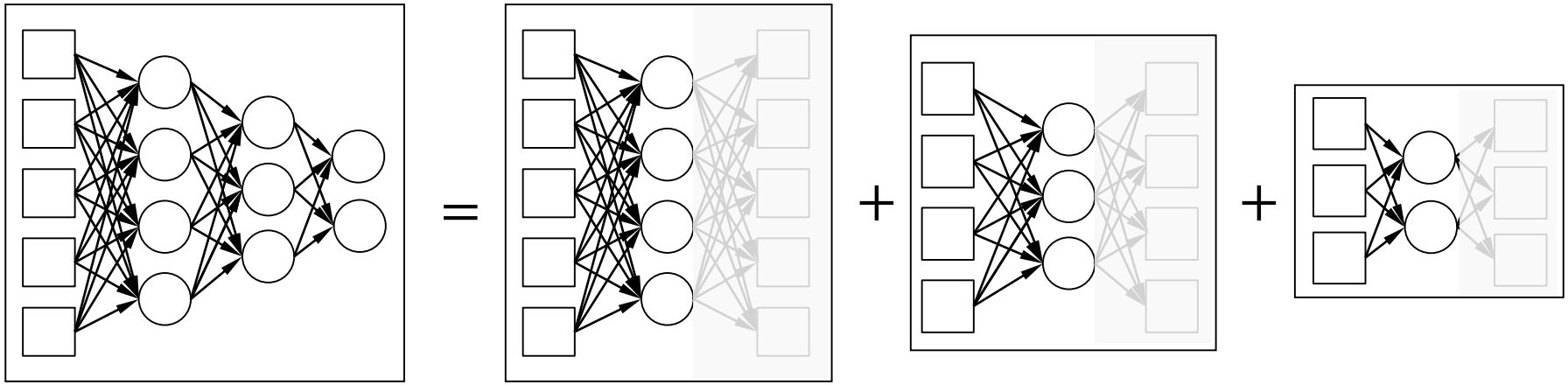
Unüberwachtes Vortrainieren: Kodierer (Autoencoder) für das Vortraining jeder Schicht.

Ziel: Gute Initialisierung für tiefes Netz finden.

Problem: Nicht genügend annotierte Daten für überwachtes Lernen.

Tiefes Netz wird schichtweise aufgeteilt und nacheinander unüberwacht vortrainiert. Jede Schicht wird als Kodierer für vorherigen Schicht trainiert.

Vermeidet Probleme durch lokale Optima, da beim Training der nächsten Schicht bereits eine gute Initialisierung vorhanden ist.



Hinweis: Bei sehr großer Menge annotierter Daten kann ein tiefes Netz auch ohne Vortraining direkt gelernt werden.

Stochastischer Gradientenabstieg:

Parameteroptimierung mit teilmengenbasiert *stochastischem Gradientenabstieg* mit *Momentum* (vermeidet Konvergenz in Sattelpunkten oder lokalen Minima) und *Gewichtsverfall* (vermeidet Overfitting). Aktualisierung des Parametervektors $\boldsymbol{w}$ bei Zielfunktion e:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha\left(\Delta\boldsymbol{w}_t + D \cdot \boldsymbol{w}_t\right) \quad , \quad \Delta\boldsymbol{w}_t = M \cdot \Delta\boldsymbol{w}_{t-1} + \sum_{\nu \in B_t} \frac{\partial e_\nu}{\partial \boldsymbol{w}_t}$$

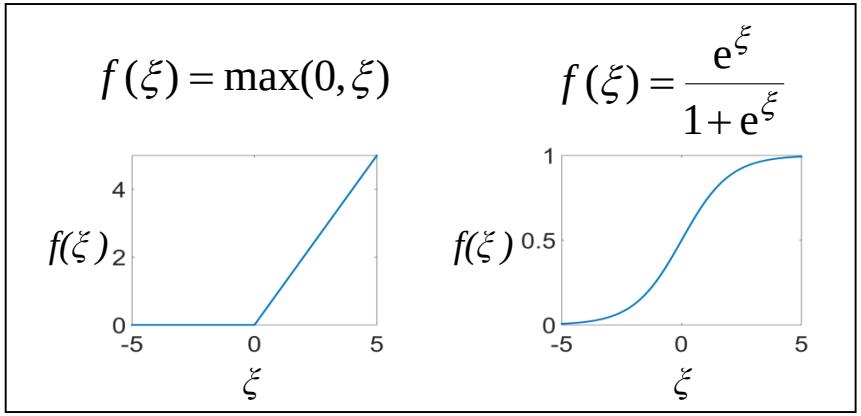
Lernrate — Gewichtsverfall — Momentum

Gradient für eine zufällig gewählte kleine Teilmenge B_t aus Trainingsdaten.

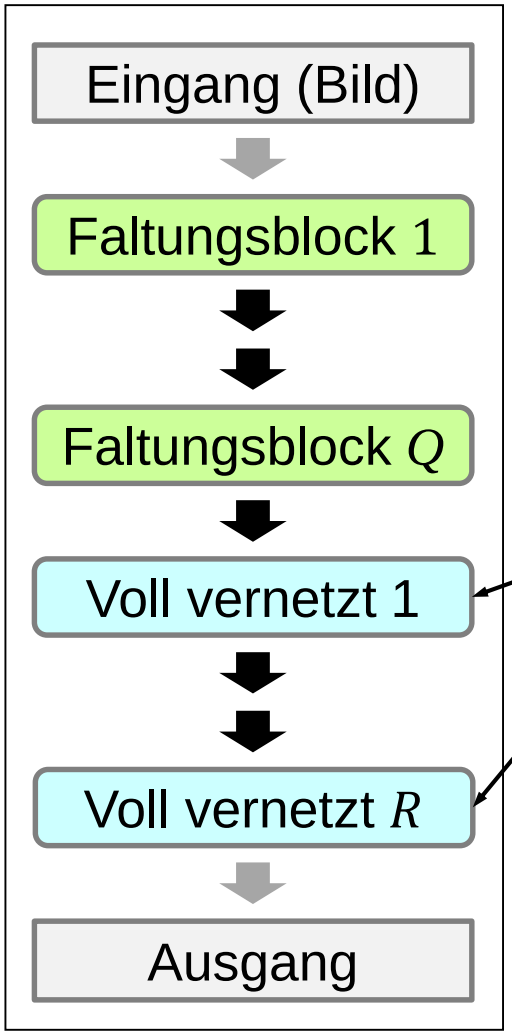
ReLU:

Rectified Linear Unit (ReLU) anstatt Sigmoidfunktion als nichtlineare Aktivierungsfunktion $f(\xi)$

Vorteil: Gradient verschwindet für große ξ nicht → beschleunigt Training.

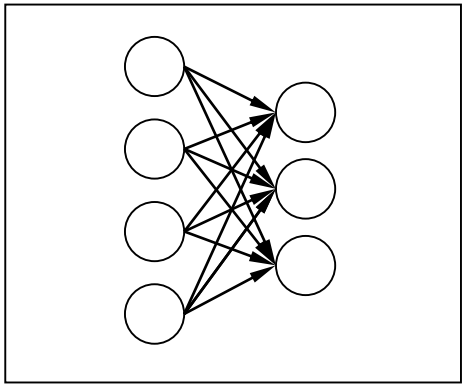


Convolutional Neural Network: Typischer Aufbau für Bildklassifikation.

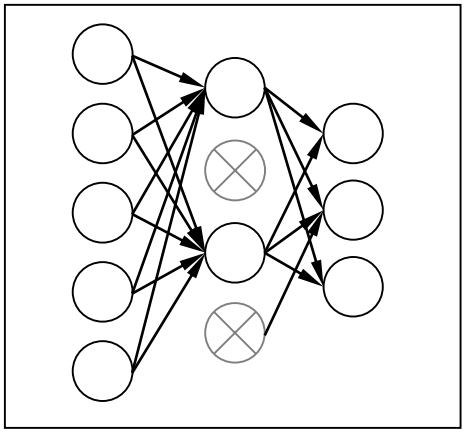


Meist $Q >> R$

Voll vernetzt: wie bei klassischem neuronalen Netz.
Klassifiziert die Merkmale der Faltungsblöcke.



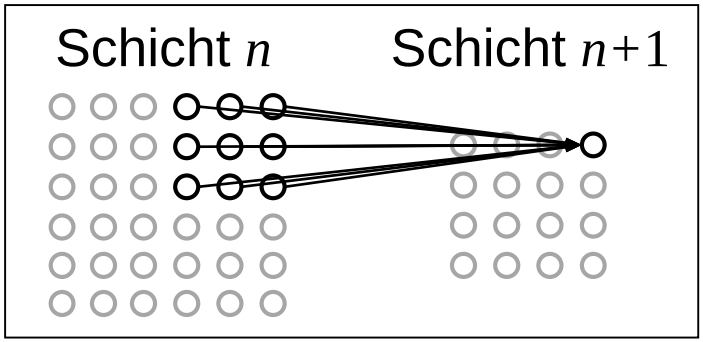
Trainingsstrategie für voll vernetzte Schichten: *dropout*. Deaktiviert zufällig Neuronen während des Trainings. Vermeidet Overfitting.



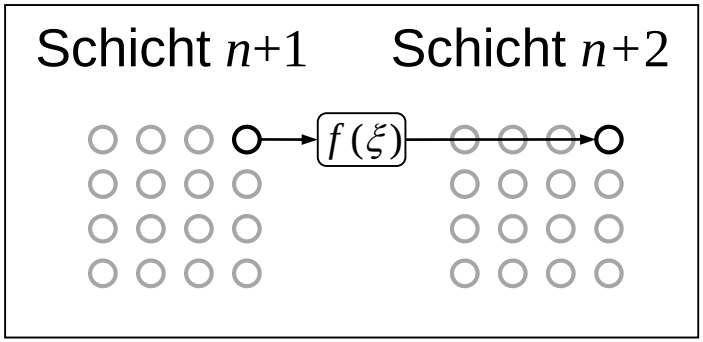
7.4. Künstliche Neuronale Netze – Tiefe Netze

Faltungs-block

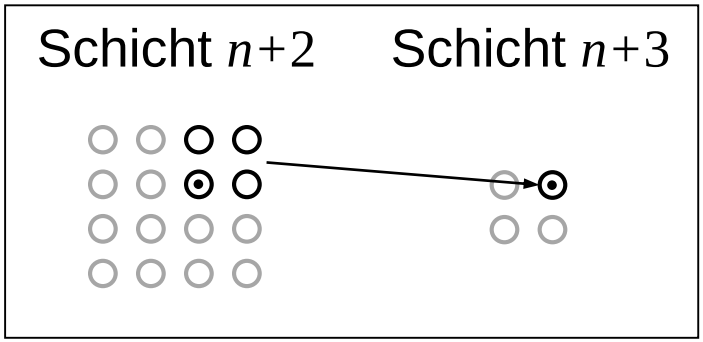
Faltung (*Convolution*) zur Filterung der Daten.
Gewichte entsprechen gleicher Faltungsmatrix für alle Verschiebungen → LVI-Filter.
Lernt mehrere LVI-Filter parallel.



Aktivierungsfunktion für Nichtlinearität, meist *ReLU*.



Unterabtastung zur räumlichen Aggregation von Merkmalen, meist *Max-Pooling*.
Dient der Translationsinvarianz und Rauschtoleranz.

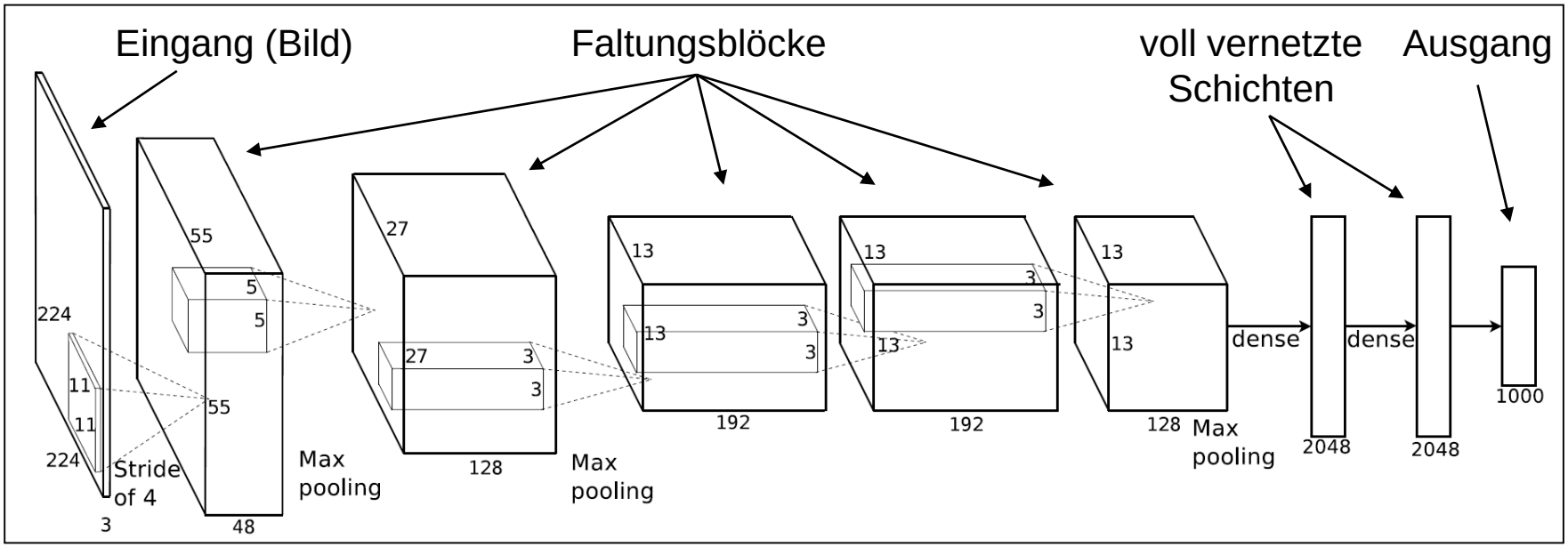


7.4. Künstliche Neuronale Netze – Tiefe Netze

Beispiel: Bildkategorisierung (ImageNet), 1000 Kategorien.
Trainingsdatensatz: ca. 15 Millionen Bilder.

Klassifikator	Fehlerrate
CNN	37,5%
beste Alternative (SIFT + Fisher-vektor + SVM)	45,7%

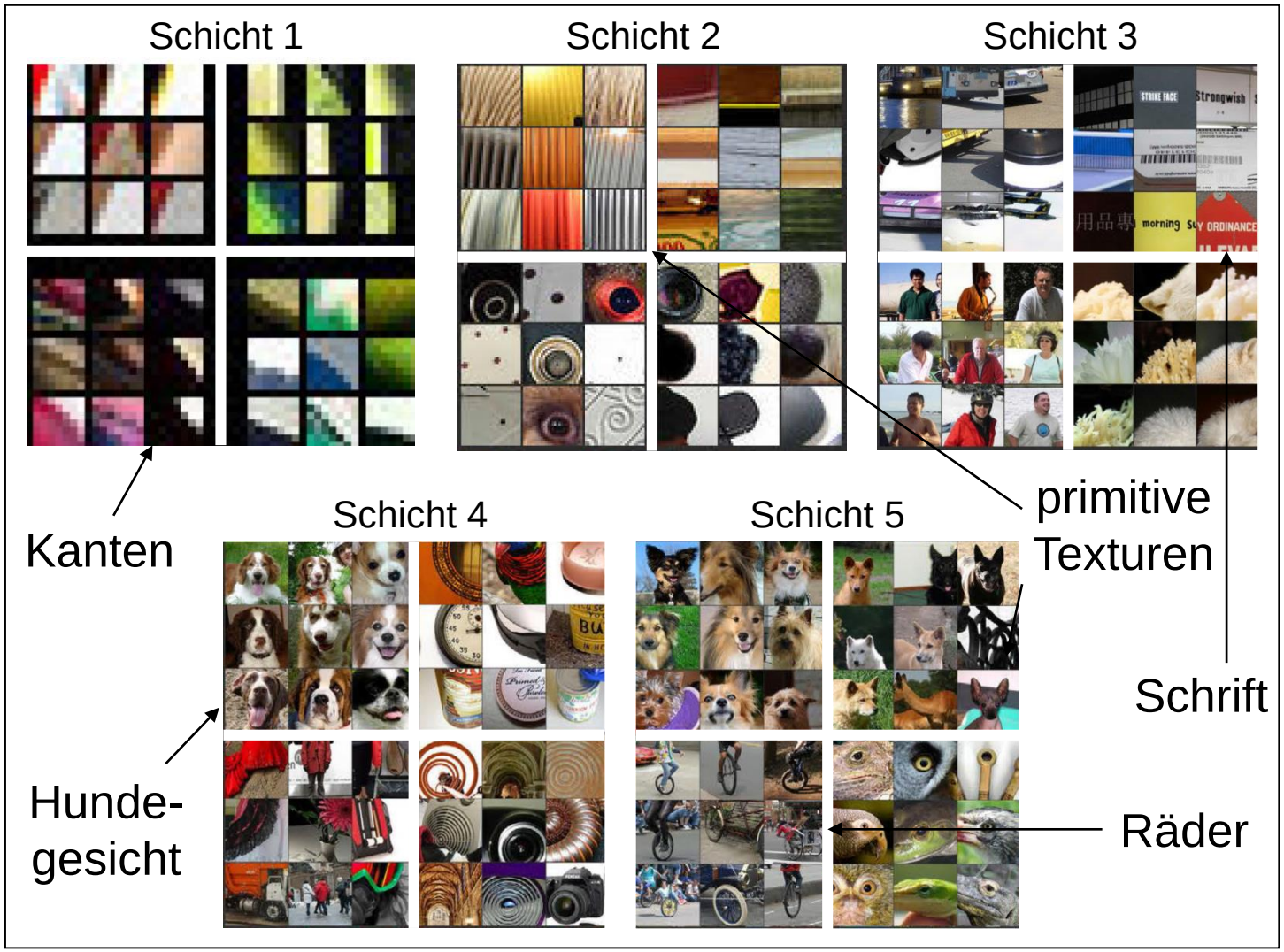
Netzaufbau:



Quelle: A. Krizhevsky et al.: ImageNet Classification with Deep Convolutional Neural Networks

7.4. Künstliche Neuronale Netze – Tiefe Netze

Beiträge der verschiedenen Schichten des Netzes.



Bildausschnitte mit der höchsten Aktivierung für jeweils 4 ausgewählte Merkmale (Filter).

Je tiefer die Schicht im Netz, desto höher die Abstraktion.

Quelle: M. D. Zeiler et al.: Visualizing and Understanding Convolutional Networks

7.5. Support Vektor Maschine

Einfacher, extrem leistungsfähiger Klassifikator, der auf mehreren fundamentalen Ideen basiert und diese geschickt kombiniert.

(A) Lineare Trennung mit maximalen Abstand (*Margin*) der Trennebene zu den nächstgelegenen Stichproben (Support Vektoren).

(B) Duale Formulierung des linearen Klassifikators.

(C) Nichtlineare Abbildung der primären Merkmale in einen hochdimensionalen Merkmalsraum Φ.

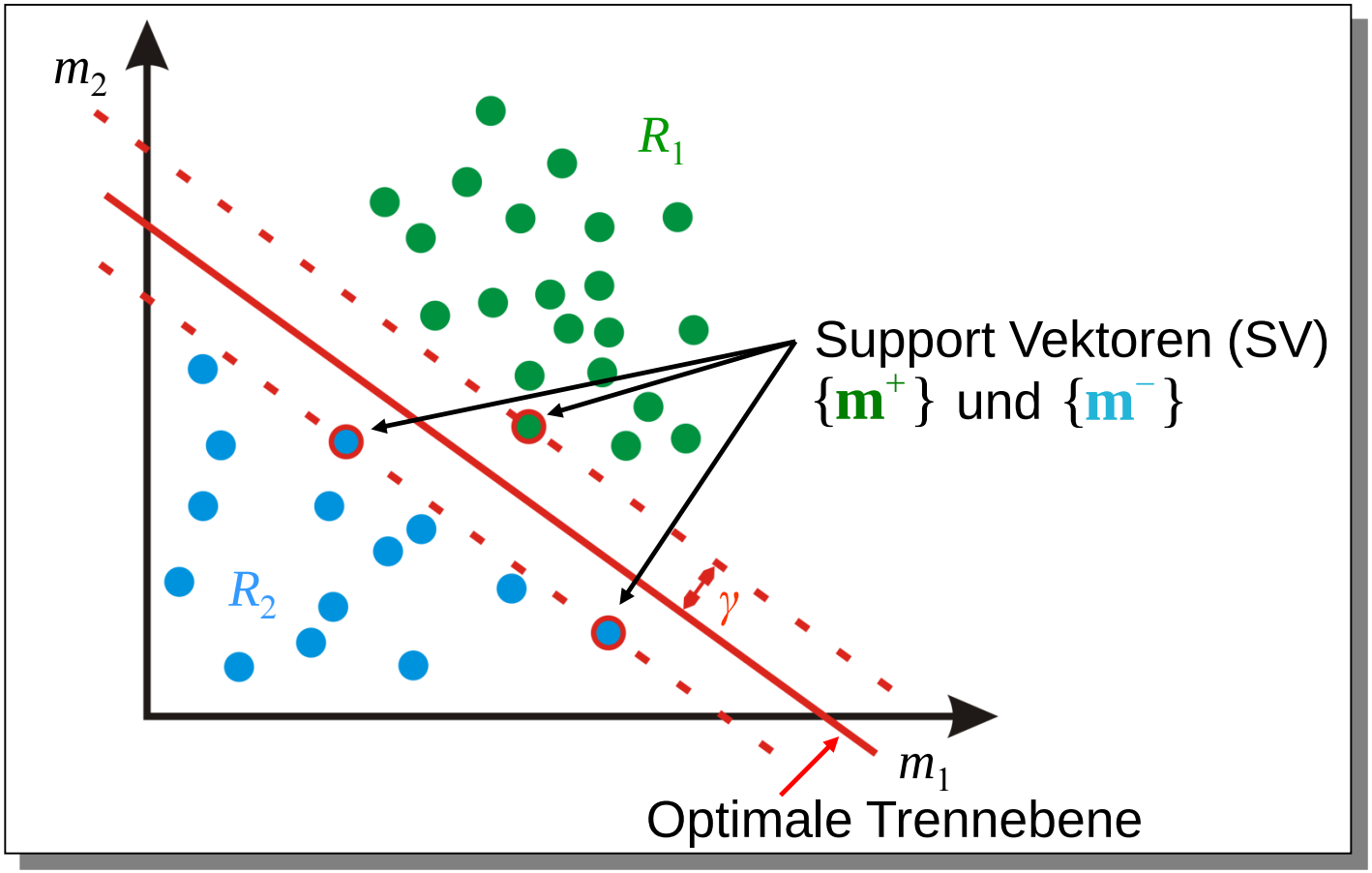
(D) Implizite Nutzung des u.U. ∞-dimensionalen Eigenfunktionsraumes einer so genannten Kernfunktion K als transformierten Merkmalsraum Φ. Dabei müssen die transformierten Merkmale nicht explizit berechnet werden und der Klassifikator hat trotz der hohen Dimension von Φ nur eine niedrige Zahl von freien Parametern (Kernel-Trick).

(E) Relaxation der Forderung nach linearer Trennbarkeit durch Einführung von Schlupfvariablen (*slack variables*).

7.5. Support Vektor Maschine – (A)

Voraussetzungen: - Zweiklassenproblem: $c = 2$
- Datensatz $D = \{\mathbf{m}_1, ..., \mathbf{m}_N\}$ sei linear trennbar
- $\mathbf{m} \in \mathbb{R}^d$

(A)



γ : maximaler Abstand (Rand) zu den nächstgelegenen Stichproben.

7.5. Support Vektor Maschine – (A)

Man wählt die Trennhyperebene $\mathbf{w}^{\mathrm{T}}\mathbf{m}+b=0$ aus, für die der Abstand γ (Rand, Margin) maximal wird.

$$(\mathbf{w},b)=\arg\max_{\mathbf{w}',b'}\{\gamma(\mathbf{w}',b',\mathrm{D})\}$$

Lineare Klassifikation: $k(\mathbf{m})=\mathbf{w}^{\mathrm{T}}\mathbf{m}+b$

$$k(\mathbf{m})>0 \quad : \quad \hat{\omega}=\omega_1$$
$$k(\mathbf{m})<0 \quad : \quad \hat{\omega}=\omega_2$$
$$k(\mathbf{m})=0 \quad : \quad \text{egal}$$

Hoffnung: Je größer γ, desto besser sind die Generalisierungseigenschaften des Klassifikators.

Bemerkung: Die Lage der Trennebene hängt nur von den Support Vektoren ab. Die SVM „konzentriert sich“ nur auf den Grenzbereich zwischen den Clustern und somit auf die schwierigsten Beispiele.

7.5. Support Vektor Maschine – (A)

Linearer Klassifikator mit maximalem Rand (Margin):

Da das Vorzeichen der Entscheidungsfunktion $\text{sign}(k(\mathbf{m}))$ sich durch eine Skalierung $(\mathbf{w}, b) \rightarrow (\beta\mathbf{w}, \beta b), \beta > 0$ nicht ändert, kann o.B.d.A. für die SV $\mathbf{m}^+$ von ω_1, für die SV $\mathbf{m}^-$ von ω_2 und für die Trennebene $\mathbf{w}^T\mathbf{m} + b = 0$ mit maximalem Rand (Margin) geschrieben werden:

$$\left.\begin{aligned} \mathbf{w}^T\mathbf{m}^+ + b &= 1 \\ \mathbf{w}^T\mathbf{m}^- + b &= -1 \end{aligned}\right\} \Rightarrow \mathbf{w}^T\mathbf{m}^+ - \mathbf{w}^T\mathbf{m}^- = 2$$

Normierung des Vektors $\mathbf{w}$ liefert:

$$\left.\begin{aligned} \frac{\mathbf{w}^T}{\|\mathbf{w}\|}\mathbf{m}^+ + \frac{b}{\|\mathbf{w}\|} &= \gamma \\ \frac{\mathbf{w}^T}{\|\mathbf{w}\|}\mathbf{m}^- + \frac{b}{\|\mathbf{w}\|} &= -\gamma \end{aligned}\right\} \Rightarrow \frac{1}{\|\mathbf{w}\|}\left(\mathbf{w}^T\mathbf{m}^+ - \mathbf{w}^T\mathbf{m}^-\right) = 2\gamma \quad \Rightarrow \quad \text{Rand } \gamma = \frac{1}{\|\mathbf{w}\|}$$

Der Rand γ wird maximal $\Leftrightarrow \|\mathbf{w}\|$ wird minimal.

7.5. Support Vektor Maschine – (A)

Linearer Klassifikator mit maximalem Rand (Margin):

Gegeben sei eine linear trennbare Lernstichprobe D.

Die Hyperebene $\mathbf{w}^{\mathrm{T}}\mathbf{m}+b=0$, die das Optimierungsproblem:

Minimiere: $\|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle = \mathbf{w}^{\mathrm{T}}\mathbf{w}$

Nebenbedingung (NB): $z_i\left(\langle \mathbf{w}, \mathbf{m}_i \rangle + b\right) \geq 1 \qquad i = 1,\ldots,N$

löst, realisiert den linearen Klassifikator mit maximalem Rand.

Indikatorvariable: $z_i := \begin{cases} 1 & \text{für } \mathbf{m}_i \in \omega_1 \\ -1 & \text{für } \mathbf{m}_i \in \omega_2 \end{cases}$

7.5. Support Vektor Maschine – (A)

Optimierung nach Lagrange:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^{N} \alpha_i \left[z_i \left(\langle \mathbf{w}, \mathbf{m}_i \rangle + b \right) - 1 \right] \overset{!}{\rightarrow} \text{minimal}$$

mit Lagrangefaktoren: $\alpha_i \geq 0 \quad i = 1, \ldots, N$

Stationarität:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} z_i \alpha_i \mathbf{m}_i \overset{!}{=} 0 \qquad \frac{\partial L}{\partial b} = \sum_{i=1}^{N} z_i \alpha_i \overset{!}{=} 0$$

$$\rightarrow \quad \mathbf{w} = \sum_{i=1}^{N} z_i \alpha_i \mathbf{m}_i \quad \text{und} \quad \sum_{i=1}^{N} z_i \alpha_i = 0$$

7.5. Support Vektor Maschine – (B)

(B) Duale Formulierung des linearen Klassifikators:

Setzt man $\mathbf{w} = \sum_{j=1}^{N} \alpha_j z_j \mathbf{m}_j$ in die *primale* Form des Klassifikators

$k(\mathbf{m}) = \mathbf{w}^{\mathrm{T}}\mathbf{m} + b = \sum_{i=1}^{d} w_i m_i + b = \langle \mathbf{w}, \mathbf{m} \rangle + b$ ein, so folgt:

Duale Form:

$$k(\mathbf{m}) = \langle \mathbf{w}, \mathbf{m} \rangle + b = \left\langle \sum_{j=1}^{N} \alpha_j z_j \mathbf{m}_j, \mathbf{m} \right\rangle + b = \sum_{j=1}^{N} \alpha_j z_j \langle \mathbf{m}_j, \mathbf{m} \rangle + b \quad (*)$$

Bemerkungen:

- Zahl der freien Parameter in dieser Formulierung: N ; sie hängt nicht von der Dimension d ab!
- Im Klassifikator nach $(*)$ treten die Merkmalsvektoren nicht isoliert, sondern nur innerhalb von Skalarprodukten auf.
- Das aus dem Perzeptronalgorithmus resultierende $\mathbf{w}$ ist ebenfalls eine Linearkombination der Vektoren $\mathbf{m}_i \in \mathrm{D}$.

7.5. Support Vektor Maschine – (B)

Optimierung nach Lagrange:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle - \sum_{i=1}^{N} \alpha_i \left[z_i \left(\langle \mathbf{w}, \mathbf{m}_i \rangle + b \right) - 1 \right] \overset{!}{\rightarrow} \text{minimal}$$

mit Lagrangefaktoren: $\alpha_i \geq 0 \quad i = 1, \ldots, N$

Stationarität:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} z_i \alpha_i \mathbf{m}_i \overset{!}{=} 0 \qquad \frac{\partial L}{\partial b} = \sum_{i=1}^{N} z_i \alpha_i \overset{!}{=} 0$$

$\mathbf{w} = \sum_{i=1}^{N} z_i \alpha_i \mathbf{m}_i$ und $\sum_{i=1}^{N} z_i \alpha_i = 0$ in L einsetzen:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} z_i z_j \alpha_i \alpha_j \langle \mathbf{m}_i, \mathbf{m}_j \rangle \overset{!}{\rightarrow} \text{maximal}$$

Das ist die duale Form der Lagrangefunktion. Sie hängt nur von den dualen Variablen ab und muss maximiert werden.

7.5. Support Vektor Maschine – (B)

Linearer Klassifikator mit maximalem Rand (Margin):

Gegeben sei eine linear trennbare Lernstichprobe D.

Falls α^* das folgende quadratische Optimierungsproblem mit Gleichungs- und Ungleichungsnebenbedingungen löst:

Maximiere: $$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} z_i z_j \alpha_i \alpha_j \langle \mathbf{m}_i, \mathbf{m}_j \rangle$$

Nebenbedingung (NB): $$\sum_{i=1}^{N} z_i \alpha_i = 0,\ \alpha_i \geq 0,\ i = 1,...,N$$

dann realisiert der Vektor $\mathbf{w}^* = \sum_{i=1}^{N} z_i \alpha_i^* \mathbf{m}_i$ den linearen Klassifikator mit maximalem Rand $\gamma = \left\| \mathbf{w}^* \right\|^{-1}$.

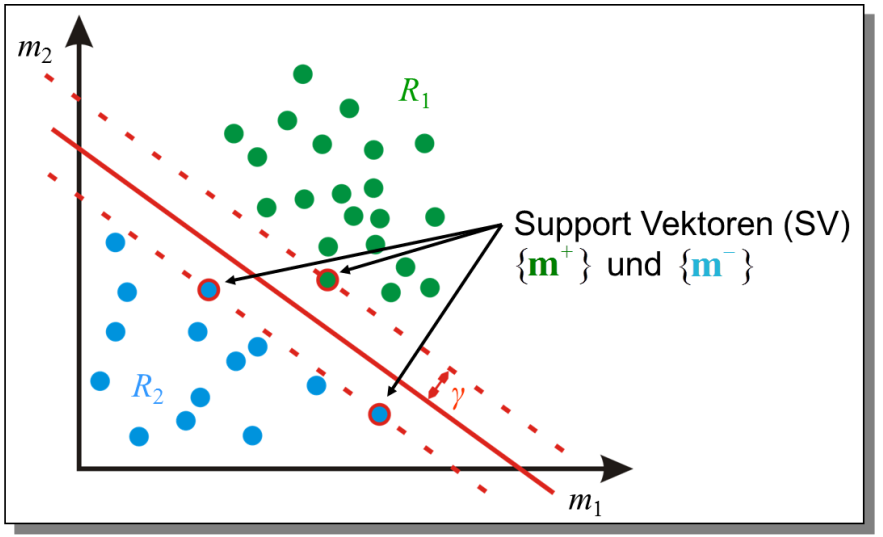
Für b^* folgt: $$b^* = -\frac{\max\limits_{z_i=-1} \left\{ \langle \mathbf{w}^*, \mathbf{m}_i \rangle \right\} + \min\limits_{z_i=1} \left\{ \langle \mathbf{w}^*, \mathbf{m}_i \rangle \right\}}{2}$$

7.5. Support Vektor Maschine – (B)

Linearer Klassifikator mit maximalem Rand (Margin):

Bemerkungen:

- Für solche quadratischen Optimierungsprobleme mit den gegebenen Nebenbedingungen gibt es effiziente Lösungsverfahren.
- Lernstichproben $\mathbf{m}_i$, die nicht SV sind, beeinflussen das Ergebnis nicht.
  → Die SV alleine definieren die Trennebene.
  → Bei der Optimierung verschwinden alle α_i^* , die nicht zu den SV gehören. Nur die zu den SV gehörenden α_i^* sind $\neq 0$! (Daher Namensgebung SV).
- Die Zahl der nichtverschwindenden Parameter des Klassifikators ist gleich #SV (Anzahl der SV).



7.5. Support Vektor Maschine – (C)

(C) Nichtlineare Abbildung der primären Merkmale in einen hochdimensionalen Vektorraum Φ

$$\mathbf{m} \mapsto \boldsymbol{\varphi}(\mathbf{m}) = \left(\varphi_1(\mathbf{m}),...,\varphi_{d^*}(\mathbf{m})\right)^{\mathrm{T}} \qquad \mathbb{M} \subset \mathbb{R}^{d}$$
$$\mathbb{M} \to \Phi \qquad \boldsymbol{\varphi} \in \Phi \subset \mathbb{R}^{d^*}$$

Die $\varphi_i(\cdot)$ sind im Allgemeinen nichtlineare Funktionen.

Φ sei ausgestattet mit einem Skalarprodukt: $\langle \boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2 \rangle$

Nichtlineare Entscheidungsfunktion (Grundform)**:**

$$k(\mathbf{m}) = \sum_{i=1}^{d^*} w_i \varphi_i(\mathbf{m}) + b = \mathbf{w}^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{m}) + b = \langle \mathbf{w}, \boldsymbol{\varphi}(\mathbf{m}) \rangle + b \qquad \mathbf{w} \in \mathbb{R}^{d^*}$$

- Zahl der freien Parameter des Klassifikators in dieser Formulierung: d^*
- Lineare Trennung mit maximalen Abstand der Trennebene zu den nächstgelegenen Stichproben $\{\boldsymbol{\varphi}(\mathbf{m}_i)\}$ kann nun in Φ erfolgen.
- Für $d^* > d$ bildet $\boldsymbol{\varphi}(\mathbf{m})$ eine d-dimensionale Untermannigfaltigkeit in Φ.

7.5. Support Vektor Maschine – (C)

Bemerkung: Hat man für ein Zweiklassenproblem $c = 2$ einen Datensatz $\mathrm{D} = \{\mathbf{m}_1, ..., \mathbf{m}_N\}$ d-dimensionaler Merkmalsvektoren, so lassen sich diese immer linear, d.h. durch eine Hyperebene trennen, wenn $d \geq N - 1$ ist und die $\{\mathbf{m}_i\}$ nicht in einem $d-1$-dimensionalen Unterraum zu liegen kommen.

Nichtlineare Entscheidungsfunktion (Duale Form):

$$k(\mathbf{m}) = \sum_{i=1}^{N} \alpha_i z_i \left\langle \boldsymbol{\varphi}(\mathbf{m}_i), \boldsymbol{\varphi}(\mathbf{m}) \right\rangle + b$$

Zahl der freien Parameter des Klassifikators in dieser Formulierung: N.

Die Dimension d^* tritt in der dualen Formulierung nicht in Erscheinung!

Wenn es einen Weg gäbe, die inneren Produkte $\left\langle \boldsymbol{\varphi}(\mathbf{m}_i), \boldsymbol{\varphi}(\mathbf{m}) \right\rangle$ direkt als eine Funktion der Merkmalsvektoren $\mathbf{m}_i$ und $\mathbf{m}$ zu berechnen, könnten die beiden Schritte hin zu einer nichtlinearen Entscheidungsfunktion (Transformation $\boldsymbol{\varphi}(\cdot)$ und Skalarprodukt in Φ) auf einmal durchgeführt werden.

(D) Kernfunktionen

Definition: Kernfunktion (Kernel):
Eine Funktion K heißt Kernfunktion, falls für alle $\mathbf{m}, \mathbf{m}' \in \mathbb{M}$ gilt:

$$K(\mathbf{m}, \mathbf{m}') = \langle \varphi(\mathbf{m}), \varphi(\mathbf{m}') \rangle$$

wobei $\varphi(\cdot)$ eine Abbildung von $\mathbb{M}$ auf Φ ist.

Entscheidungsfunktion in Kernfunktionsformulierung:

$$k(\mathbf{m}) = \sum_{i=1}^{N} \alpha_i z_i K(\mathbf{m}_i, \mathbf{m}) + b \qquad (**)$$

7.5. Support Vektor Maschine – (D)

Theorem von Mercer: Eine symmetrische Funktion K des L_2 besitzt eine Entwicklung:

$$K(\mathbf{m},\mathbf{m}') = \sum_{j=1}^{\infty} \lambda_j \varphi_j(\mathbf{m}) \varphi_j(\mathbf{m}')$$

mit positiven Koeffizienten $\lambda_j > 0$ (d.h. K bezeichnet ein Skalarprodukt in einem mit K assoziierten Merkmalsraum Φ) genau dann, wenn:

$$\int_{\mathbb{M}} \int_{\mathbb{M}} K(\mathbf{m},\mathbf{m}') f(\mathbf{m}) f(\mathbf{m}') \mathrm{d}\mathbf{m} \mathrm{d}\mathbf{m}' > 0$$

gilt für alle $f \not\equiv 0$ mit $\int f^2(\mathbf{m}) \mathrm{d}\mathbf{m} < \infty$

Dabei sind λ_i und $\varphi_i(\cdot)$ die Lösungen des Eigenwertsproblems:

$$\int_{\mathbb{M}} K(\mathbf{m},\mathbf{m}') \varphi(\mathbf{m}') \mathrm{d}\mathbf{m}' = \lambda \varphi(\mathbf{m})$$

7.5. Support Vektor Maschine – (D)

Ist eine Funktion K gegeben, die Mercers Theorem erfüllt und mithin eine Kernfunktion ist, und arbeitet man mit der dualen Formulierung des Klassifikators (**), so nutzt man implizit den u.U. ∞-dimensionalen transformierten Merkmalsraum Φ, ohne dass die zugehörigen Merkmale $\{\varphi_j\}$ explizit berechnet werden müssen!

Die Kernfunktion K induziert den Merkmalsraum Φ.

Obwohl man also einen Merkmalsvektor $\boldsymbol{\varphi}(\mathbf{m})$ mit gegebenenfalls sehr hoher, u.U. sogar unendlicher Dimension d^* hat, besitzt der Klassifikator (**) nur N freie Parameter.

7.5. Support Vektor Maschine – (D)

Kernfunktionen: Beispiel 1

$$K(\mathbf{m},\mathbf{u}) := \langle \mathbf{m},\mathbf{u} \rangle^2 = \left(\sum_{i=1}^{d} m_i u_i \right)^2 = \left(\sum_{i=1}^{d} m_i u_i \right) \left(\sum_{j=1}^{d} m_j u_j \right) =$$

$$= \sum_i \sum_j m_i m_j u_i u_j = \sum_{(i,j)=(1,1)}^{(d,d)} (m_i m_j)(u_i u_j)$$

Das ist äquivalent zum inneren Produkt für transformierte Merkmalsvektoren der Form: $\varphi(\mathbf{m}) = \left(m_i m_j \right)_{(i,j)=(1,1)}^{(d,d)} = \left(m_1^2, m_1 m_2, m_2^2, m_1 m_3, m_2 m_3, \ldots, m_d^2 \right)^{\mathrm{T}}$

also dem Vektor aller Monome zweiten Grades der $\{m_i\}$.

$d^* = \frac{1}{2}(d+1)d$ ($m_i m_j$ für $i \neq j$ kommt je zweimal vor)

7.5. Support Vektor Maschine – (D)

Kernfunktionen: Beispiel 2

$$K(\mathbf{m},\mathbf{u}) := \left(\langle \mathbf{m},\mathbf{u} \rangle + c\right)^2 = \left(\sum_{i=1}^{d} m_i u_i + c\right)\left(\sum_{j=1}^{d} m_j u_j + c\right) =$$

$$= \sum_i \sum_j m_i m_j u_i u_j + 2c\sum_{i=1}^{d} m_i u_i + c^2 = \sum_{(i,j)=(1,1)}^{(d,d)} (m_i m_j)(u_i u_j) + \sum_{i=1}^{d} (\sqrt{2c}\, m_i)(\sqrt{2c} u_i) + c^2$$

Der implizite Vektor $\boldsymbol{\varphi}(\mathbf{m})$ enthält hier alle Monome der $\{m_i\}$ vom Grad ≤ 2

$$d^* = \binom{d+2}{2} = \frac{1}{2}(d+2)(d+1)$$

Kernfunktionen: Beispiel 3

$$K(\mathbf{m},\mathbf{u}) := \langle \mathbf{m},\mathbf{u} \rangle^q$$

Die transformierten Merkmale sind alle Monome $m_i m_j ... m_l$ vom Grad q.

$$d^* = \binom{d+q-1}{q}$$

7.5. Support Vektor Maschine – (D)

Kernfunktionen: Beispiel 4

$$K(\mathbf{m},\mathbf{u}) := \left(\langle \mathbf{m},\mathbf{u} \rangle + c\right)^q$$

Die transformierten Merkmale sind alle Monome von Grad $\leq q$.

Diese Kernfunktionen nennt man auch „Polynomiale Kerne“.

Kernfunktionen: Beispiel 5

Oft eingesetzte Kernfunktion (Gaußfunktion)

$$K(\mathbf{m},\mathbf{m}') = \exp\left\{-\frac{\|\mathbf{m}-\mathbf{m}'\|^2}{\sigma^2}\right\}$$

ist ein Spezialfall für so genannte Radiale Basisfunktionen (RBF).

7.5. Support Vektor Maschine – (D)

Klassifikator mit maximalem Rand im durch eine Kernfunktion K induzierten Merkmalsraum Φ

Gegeben sei eine im Raum Φ linear trennbare Lernstichprobe D und eine Kernfunktion K, die diesen Raum induziert.

α^* und b^* lösen das Optimierungsproblem:

Maximiere:	$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} z_i z_j \alpha_i \alpha_j K(\mathbf{m}_i, \mathbf{m}_j)$
Nebenbedingung (NB):	$\sum_{i=1}^{N} z_i \alpha_i = 0,\ \alpha_i \geq 0,\ i = 1,\ldots,N$

Die Entscheidungsfunktion $k(\mathbf{m}) = \sum_{i=1}^{N} z_i \alpha_i^* K(\mathbf{m}_i, \mathbf{m}) + b^*$ ist äquivalent zur Hyperebene in Φ mit maximalem Rand: $\gamma = \frac{1}{\sqrt{\sum_{i \in SV} \alpha_i^*}}$

7.5. Support Vektor Maschine – (D)

Bemerkung: Aufgrund des Theorems von Mercer ist die Matrix $\left(K\left(\mathbf{m}_i, \mathbf{m}_j\right)\right)_{i,j=1}^{N}$ positiv definit. Damit wird das Optimierungsproblem konvex und liefert als eindeutige Lösung das globale Optimum.

Fehlerwahrscheinlichkeit:

Abschätzung der Fehlerwahrscheinlichkeit für die Klassifikation von Stichproben $\mathbf{m} \notin \mathrm{D}$ (Generalisierungsfähigkeit):

$$P(\hat{\omega}(\mathbf{m}) \neq \omega(\mathbf{m}), \mathbf{m} \notin \mathrm{D}) \approx \frac{\#\mathrm{SV}}{N}$$

Leaving-one-out-Argumentation:
Einen Vektor von D auslassen. Ist er nicht SV, wird er richtig klassifiziert. Ist er SV, könnte er falsch klassifiziert werden. Zyklische Permutation über alle $\mathbf{m} \in \mathrm{D}$

⟶ Je weniger SV, desto besser! (Bestätigung „Ockhams Rasiermesser“)

(E) SVM für nicht linear trennbare Stichproben D

Einführung so genannter Slack-Variablen $\xi_i \geq 0,\ i = 1,\ldots,N,$ die ein „Eindringen“ von Vektoren $\mathbf{m}_i$ um ξ_i in den Streifen zwischen den SV bewerten.
(engl. Slack: dt. Schlupf)

Für den linearen Klassifikator mit maximalem Rand folgt:

Minimieren: $\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{N} \xi_i^2$

Nebenbedingung (NB): $z_i \left(\langle \mathbf{w}, \mathbf{m}_i \rangle + b \right) \geq 1 - \xi_i, \qquad i = 1,\ldots,N$

$C > 0$: Designparameter

Das führt in der dualen Formulierung und bei Verallgemeinerung von $\langle \mathbf{w}, \mathbf{m} \rangle$ durch eine Kernfunktion K auf das Optimierungsproblem: Soft Margin SVM

7.5. Support Vektor Maschine – (E)

Soft-Margin SVM

Gegeben: Lernstichprobe $\mathbb{D}$

Maximiere: $$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} z_i z_j \alpha_i \alpha_j \left(K(\mathbf{m}_i, \mathbf{m}_j) + \frac{1}{C} \delta_{ij} \right)$$

Nebenbedingung (NB): $\sum_{i=1}^{N} z_i \alpha_i = 0,\ \alpha_i \geq 0,\ i = 1,\ldots, N$

Lösung: $\boldsymbol{\alpha}^*$

Entscheidungsfunktion: $k(\mathbf{m}) = \sum_{i=1}^{N} z_i \alpha_i^* K(\mathbf{m}_i, \mathbf{m}) + b^*$, wobei b^* so gewählt wird, dass $z_i k(\mathbf{m}) = 1 - \frac{\alpha_i^*}{C}$ für alle i mit $\alpha_i^* \neq 0$ gilt.

Beweis siehe z.B. Cristianini, Shawe-Taylor.

Weiterführende Literatur: N. Cristianini, J. Shawe-Taylor: An Introduction to Support Vector Machines and other Kernel-Based Learning Methods. Cambridge University Press 2000.

Diskussion SVM

Vorteile:

- Sehr leistungsfähig
- Die Komplexität der SVM wird durch die Zahl der SV bestimmt und nicht durch die Dimension des Merkmalsraumes. Die SVM neigt daher weniger zu Overfitting als viele andere Klassifikatoren.
- SVM liefert globales Optimum (KNN liefern meistens suboptimale Lösungen)
- Geometrisch anschauliche Funktionsweise.
- Leicht anwendbar; kein a priori Wissen notwendig
- Gut entwickelte Theorie liegt zugrunde (Vapnik).

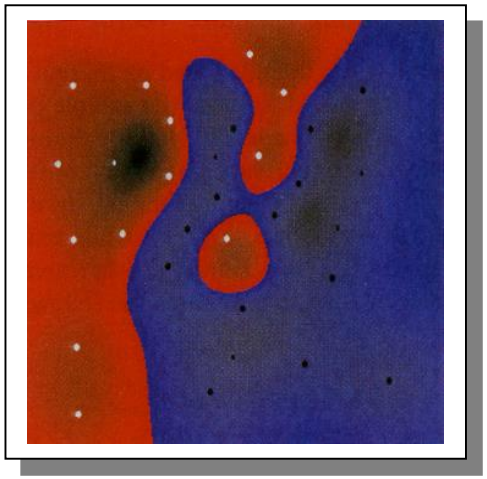
Nachteile:

- Multiklassenansatz aufwändig: eine SVM pro Klasse.

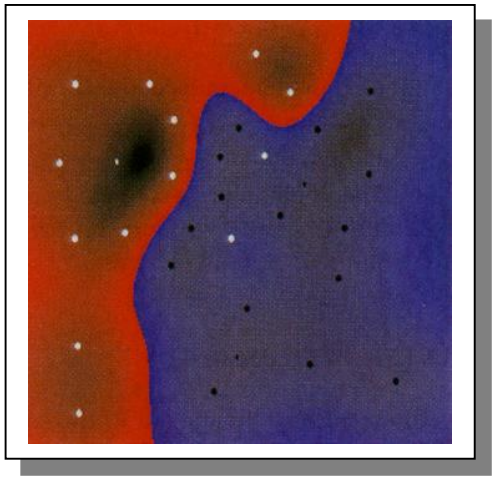
7.5. Support Vektor Maschine

Beispiel: Kernfunktion = Gaußfunktion mit festem σ .

- $c = 2$
- Klasse ω_1 = {weiße Punkte}, rotes Gebiet
- Klasse ω_2 = {schwarze Punkte}, blaues Gebiet
- SV = fette Punkte



$$K(\mathbf{m},\mathbf{m}') = \exp\left\{-\frac{\|\mathbf{m}-\mathbf{m}'\|^2}{\sigma^2}\right\}$$



SVM mit maximalem Rand (Hard-Margin):

- gute Klassifikation
- komplizierte Entscheidungsgrenze.

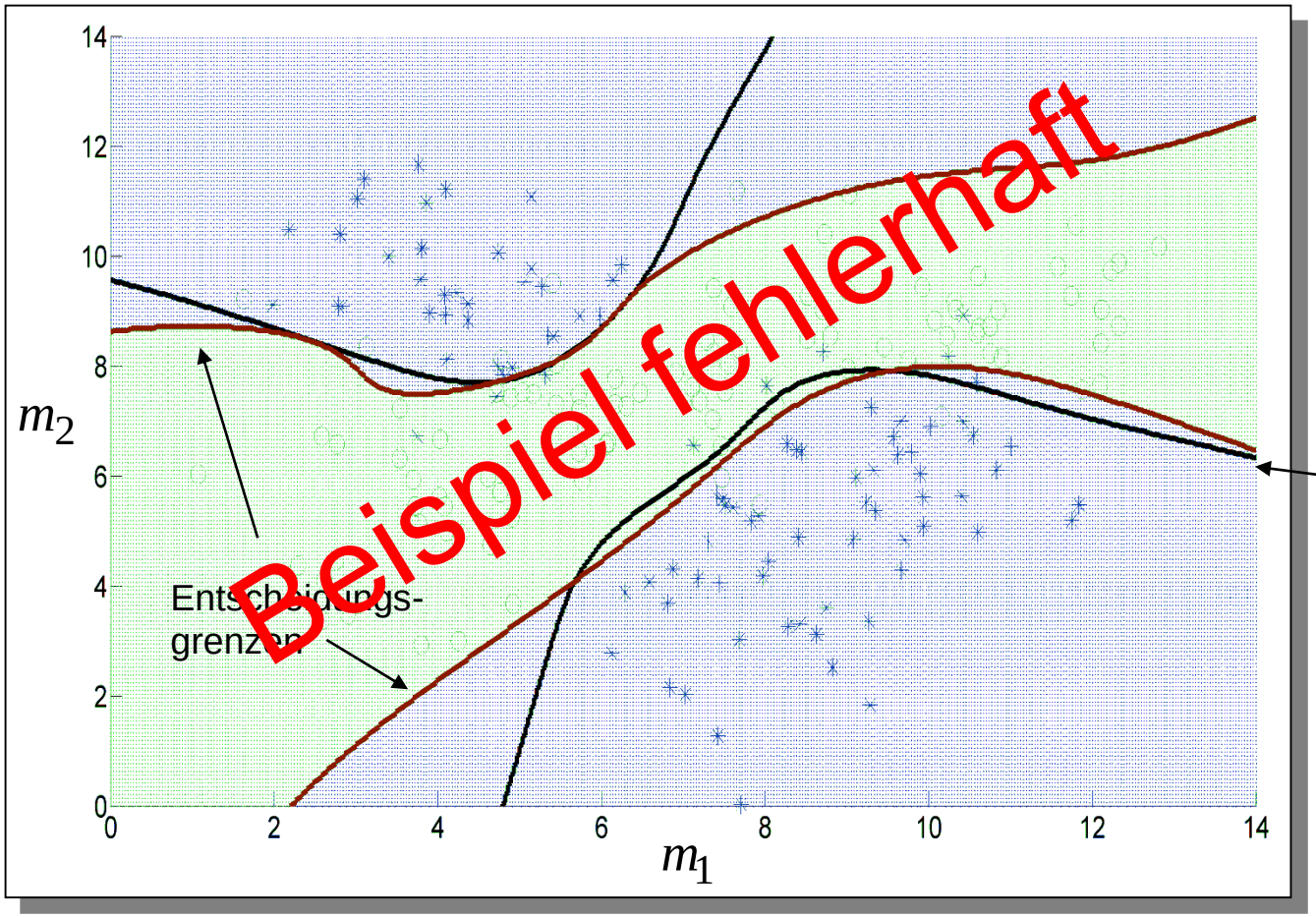
SVM mit Soft-Margin:

- erlaubt falsche Klassifikation
- glattere Entscheidungsgrenze

Quelle: N. Cristianini, J. Shawe-Taylor: An Introduction to Support Vector Machines

7.5. Support Vektor Maschine

Beispiel: *Hard-Margin* SVM, Kernfunktion = Gaußfunktion $\sigma = 4$.



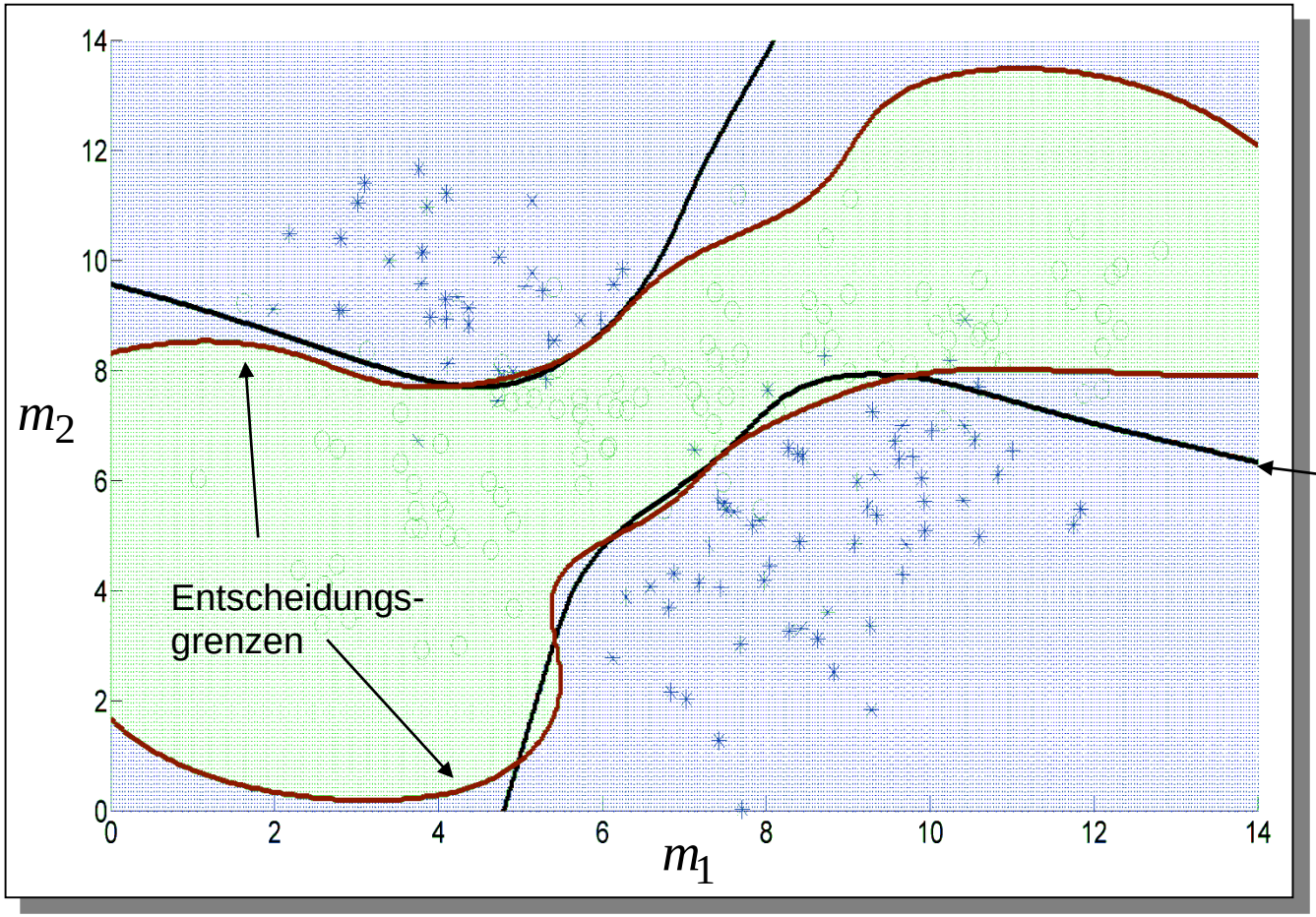
$$K(\mathbf{m},\mathbf{m}') = \exp\left\{-\frac{\|\mathbf{m}-\mathbf{m}'\|^2}{\sigma^2}\right\}$$

SVM mit Lernstichprobe erstellt.

Teststichprobe → Testfehler = 8% Asymptotischer Testfehler ≈ 11,68%

7.5. Support Vektor Maschine

Beispiel: *Soft-Margin* SVM, Kernfunktion = Gaußfunktion mit $\sigma = 0,5$
Designparameter $C = 1$



$$K(\mathbf{m},\mathbf{m}') = \exp\left\{-\frac{\|\mathbf{m}-\mathbf{m}'\|^2}{\sigma^2}\right\}$$

Bayesscher
Optimalklassifikator

SVM mit Lernstichprobe erstellt.

Teststichprobe → Testfehler = 10% Asymptotischer Testfehler ≈ 11,65%

7.5. Support Vektor Maschine

Beispiel: Klassifikation handgeschriebener Ziffern

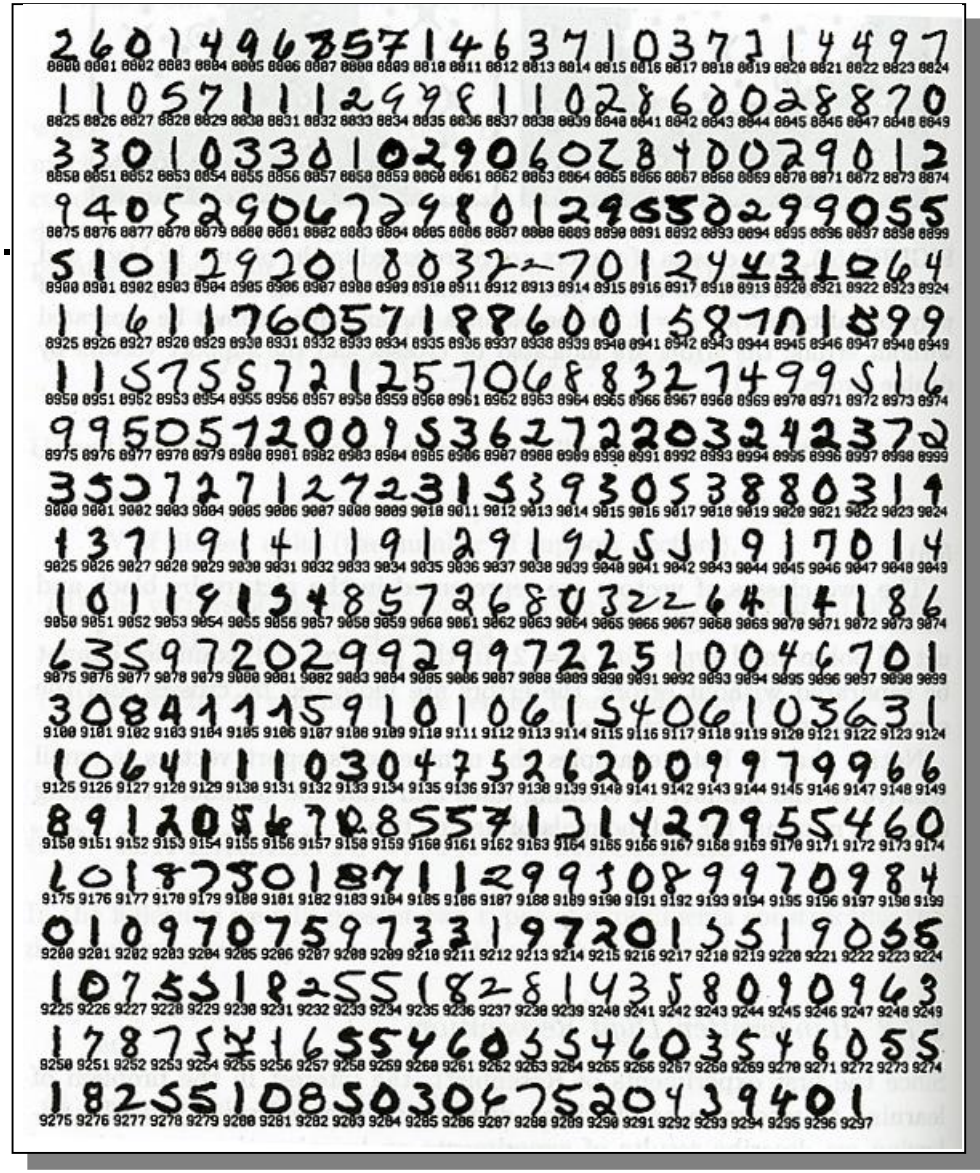
Daten:

- U.S. Postal Service Datenbasis
- Lernstichprobe D: $N = 7300$ Bsp.
- Teststichprobe T: 2000 Bsp.

(LeCun et al. 1990)

Merkmalsvektor := Muster:

- 16 x 16 Pixel Grauwertbild



Quelle für das Beispiel: V. N. Vapnik: The Nature of Statistical Learning

7.5. Support Vektor Maschine

Ansatz: Je eine SVM mit Soft-Margin für jede der 10 Klassen

$$K(\mathbf{m},\mathbf{m}') := \left(\frac{\mathbf{m}^T\mathbf{m}'}{256}\right)^q \qquad q = 1,\ldots,7$$

Klassifikator	Fehlerrate %
Menschliche Leistung	2,5
Entscheidungs-baum C4.5	16,2
Bestes zweischichtiges KNN	5,9
Fünfschichtiges KNN (LeNet 1)	5,1

Tabelle gibt *State-of-the-art* Stand Ende der 90er Jahre wieder.

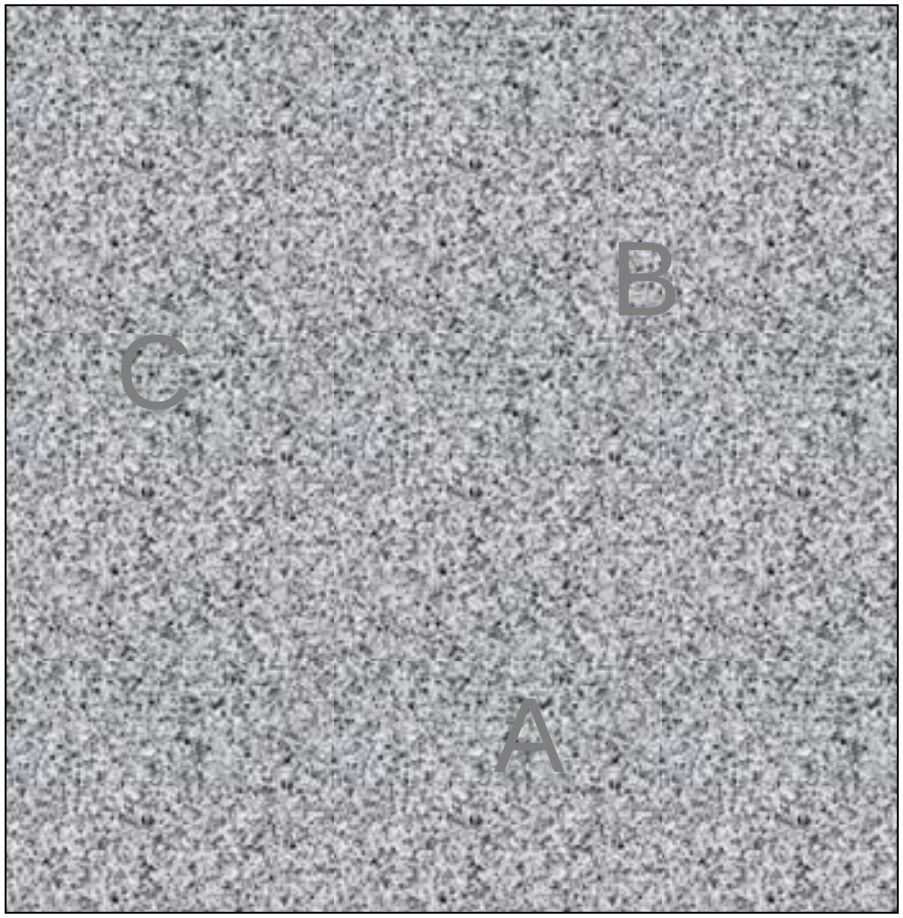
q	d^*	#SV	Fehlerrate %
1	256	282	8,9
2	≈ 33000	227	4,7
3	≈ 1×10^6	274	4,0
4	≈ 1×10^9	321	4,2
5	≈ 1×10^{12}	374	4,3
6	≈ 1×10^{14}	377	4,5
7	≈ 1×10^{16}	422	4,5

Quelle für das Beispiel: V. N. Vapnik: The Nature of Statistical Learning

7.6. Matched-Filter

Aufgabe:
In einem Signal befinden sich zu klassifizierende Objekte sowie Störungen.

Bsp.: Bildsignal



Matched-Filter in diskreter Schreibweise:

$$\mathbf{g}_{mn} = \mathbf{o}_{mn} + \mathbf{r}_{mn}$$

Bild-ausschnitt — Objekt — Rauschen

$$\mathbf{g}_{mn} := (\ldots, g_{m-i,n-j}, \ldots)^{\mathrm{T}} \qquad \forall (i,j) \in \mathrm{U}$$

$\mathbf{o}_{mn}, \mathbf{r}_{mn}$ entsprechend

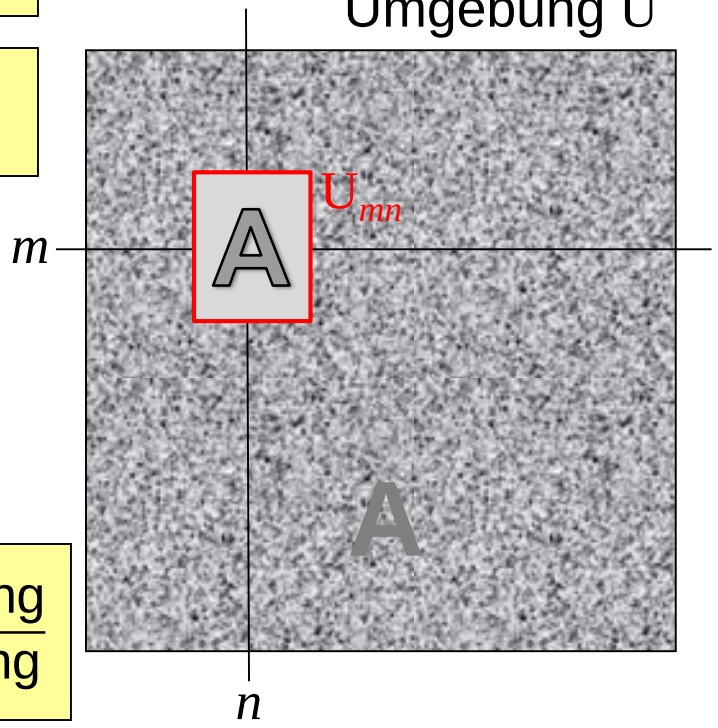
U: Umgebung des Ursprungs

U_{mn}: um m,n verschobene Umgebung U

Filter : $\mathbf{v} := (\ldots, v_{ij}, \ldots)^{\mathrm{T}}$

$$k_{mn} = \mathbf{v}^{\mathrm{T}} \mathbf{g}_{mn}$$

$$\mathbf{g}_{mn}, \mathbf{o}_{mn}, \mathbf{r}_{mn}, \mathbf{v} \in \mathrm{IR}^{|\mathrm{U}|}$$



Im Folgenden werden die Indizes mn zur Vereinfachung der Schreibweise unterdrückt:

$$k = \mathbf{v}^{\mathrm{T}} \mathbf{o} + \mathbf{v}^{\mathrm{T}} \mathbf{r}$$

Signal-to-Noise Ratio

$$\mathrm{SNR} := \frac{P_1}{P_2} = \frac{\text{Nutzsignalleistung}}{\text{Störsignalleistung}}$$

7.6. Matched-Filter

$$P_1 := (\mathbf{v}^T \mathbf{o})^2$$

$$P_2 = \mathrm{E}\{(\mathbf{v}^T \mathbf{r})^2\} = \mathrm{E}\{(\mathbf{v}^T \mathbf{r})(\mathbf{r}^T \mathbf{v})\} = \mathbf{v}^T \underbrace{\mathrm{E}\{\mathbf{r}\mathbf{r}^T\}}_{=: \mathbf{K}_{rr}} \mathbf{v}$$

← Kovarianzmatrix des Rauschens $\mathbf{r}$

$$\mathrm{SNR} = \frac{(\mathbf{v}^T \mathbf{o})^2}{\mathbf{v}^T \mathbf{K}_{rr} \mathbf{v}} \xrightarrow{!} \mathrm{Max.}$$

$\mathbf{K}_{rr} = \mathbf{Q}^T \mathbf{Q}$ ← Geht bei positiv definiten, symmetrischen Matrizen.

$$\mathbf{K}_{rr}^{-1} = \mathbf{Q}^{-1} (\mathbf{Q}^T)^{-1}$$

$\mathbf{Q}\,\mathbf{v} =: \mathbf{w}$ $\qquad$ $\mathbf{v} = \mathbf{Q}^{-1}\mathbf{w}$

$$\mathbf{v}^T = \mathbf{w}^T (\mathbf{Q}^{-1})^T$$

$$\mathrm{SNR} = \frac{(\mathbf{w}^T (\mathbf{Q}^{-1})^T \mathbf{o})^2}{\mathbf{w}^T \mathbf{w}}$$

o. B. d. A.: $\|\mathbf{w}\| = 1$

7.6. Matched-Filter

$$\mathrm{SNR} = (\mathbf{w}^{\mathrm{T}}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{o})^2 \xrightarrow{!} \text{Max.} \Leftrightarrow \mathbf{w} \parallel (\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{o} \Leftrightarrow \mathbf{w} = \text{const.}\,(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{o}$$

Diskretes Matched-Filter:

$$\mathbf{v} = \text{const.}\,\mathbf{Q}^{-1}(\mathbf{Q}^{-1})^{\mathrm{T}}\mathbf{o} = \text{const.}\,\mathbf{K}_{rr}^{-1}\mathbf{o}\,, \quad \text{const.} := 1$$

Spezialfall: $\mathbf{K}_{rr} \propto \mathbf{I}$, d.h. „weißes“ Rauschen, folgt: $\mathbf{v} \propto \mathbf{o}$.

Interpretation:

$$k_{mn} = \mathbf{v}^{\mathrm{T}}\mathbf{g}_{mn} = \mathbf{o}^{\mathrm{T}}\,\mathbf{Q}^{-1}\underbrace{(\mathbf{Q}^{-1})^{\mathrm{T}}}\,\mathbf{g}_{mn}$$

„Whitening“-Filter

Modifikation von $\mathbf{o}$ aufgrund des „Whitening“

Bemerkung: I. Allg. werden Detektionsprobleme durch Rotation, Skalierung und Verzerrung der gesuchten, bekannten Objekte erschwert. Matched-Filter sind hiergegen empfindlich.→ Vorher normalisieren.

7.6. Matched-Filter

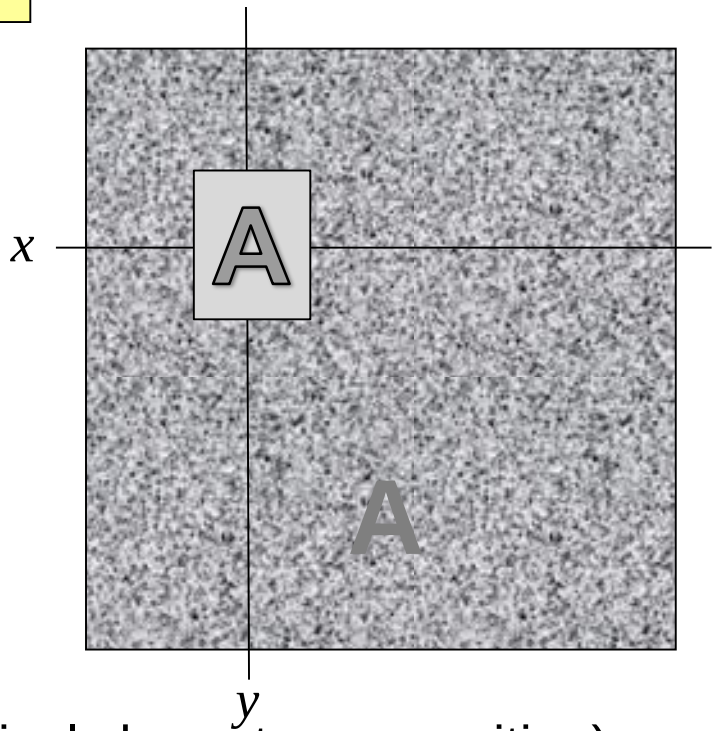
Klassifikation mit Matched-Filtern:

Für jede Klasse wird ein Matched-Filter $\mathbf{v}_i$ erstellt.

Merkmalsvektor:

$$\mathbf{m}(\mathbf{x}) := \mathrm{col}\left\{\left\{g(\mathbf{x}') \mid \mathbf{x}' = \mathbf{x} - \boldsymbol{\alpha}\,,\ \forall \boldsymbol{\alpha} \in \mathrm{U}\right\}\right\}$$

$$\mathbf{x} = (m\Delta x, n\Delta y)^{\mathrm{T}}$$



Entscheidungsfunktionen:

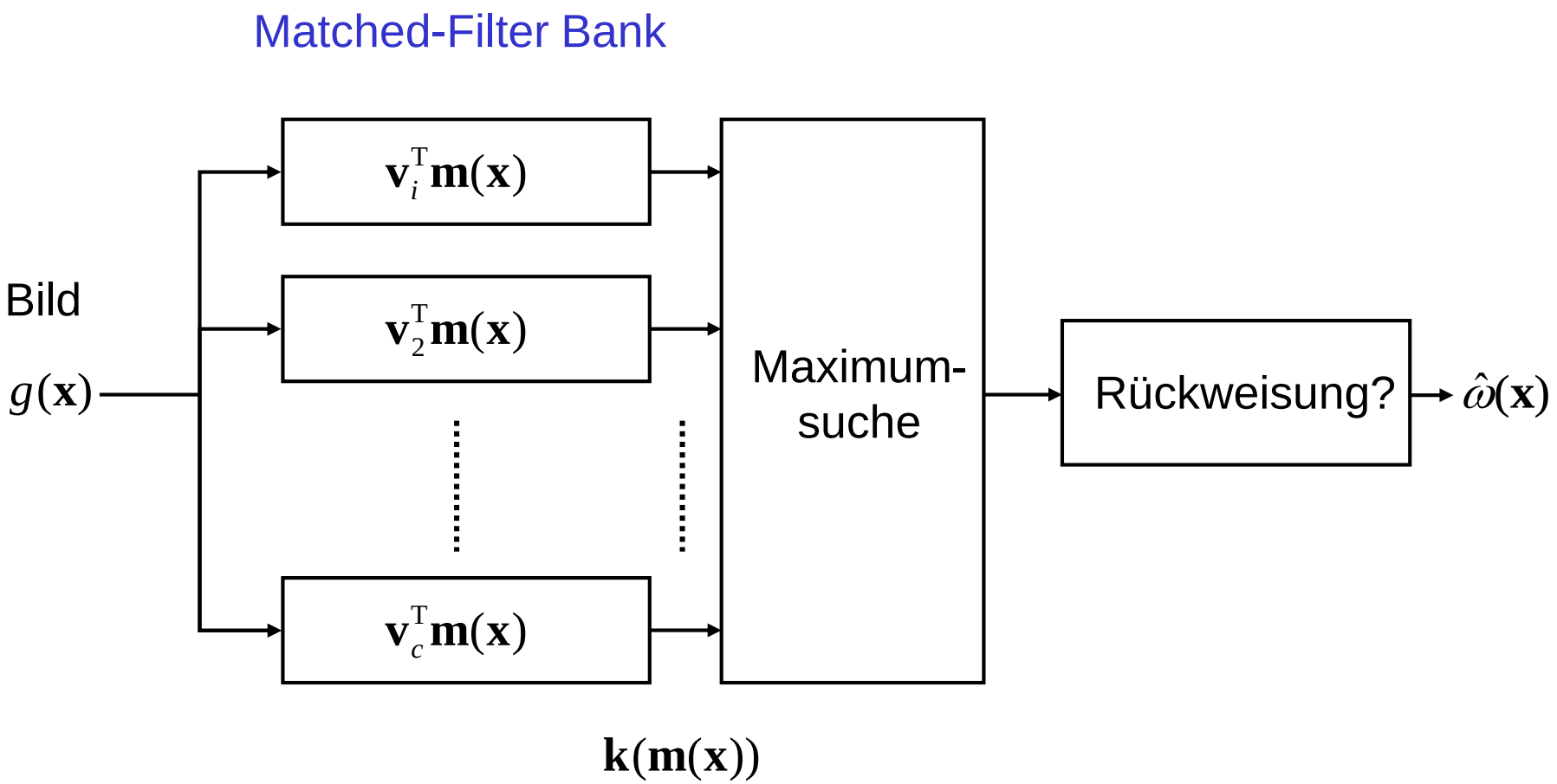
$$k_i(\mathbf{m}(\mathbf{x})) = \mathbf{v}_i^{\mathrm{T}}\mathbf{m}(\mathbf{x}) \qquad i = 1,\ldots,c$$

Entscheidungsvektor:

$$\mathbf{k}(\mathbf{m}(\mathbf{x})) = \begin{bmatrix} \mathbf{v}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{v}_c^{\mathrm{T}} \end{bmatrix} \mathbf{m}(\mathbf{x})$$

Bsp.: Automatisches Zeichenlesen (OCR optical character recognition)

Klassifikation mit Matched-Filtern:

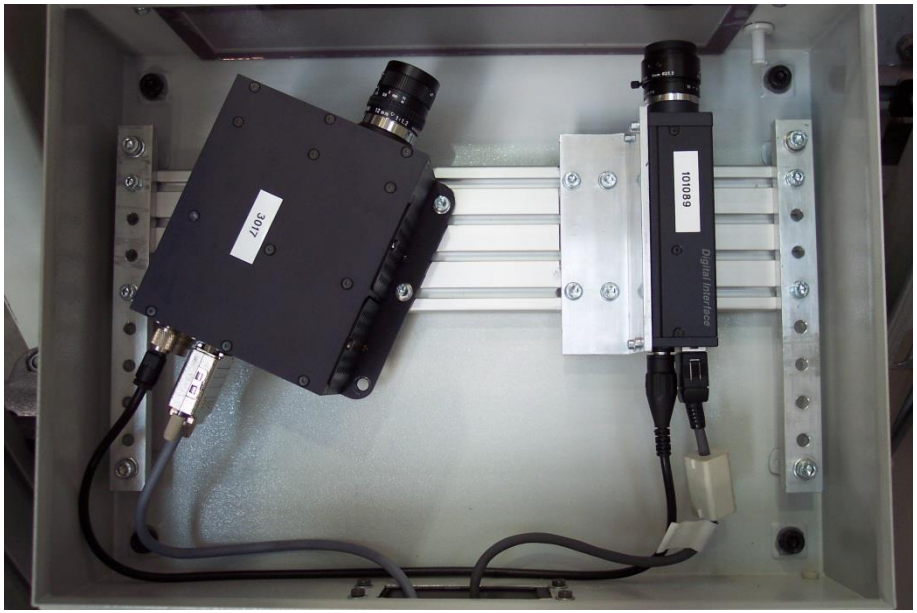


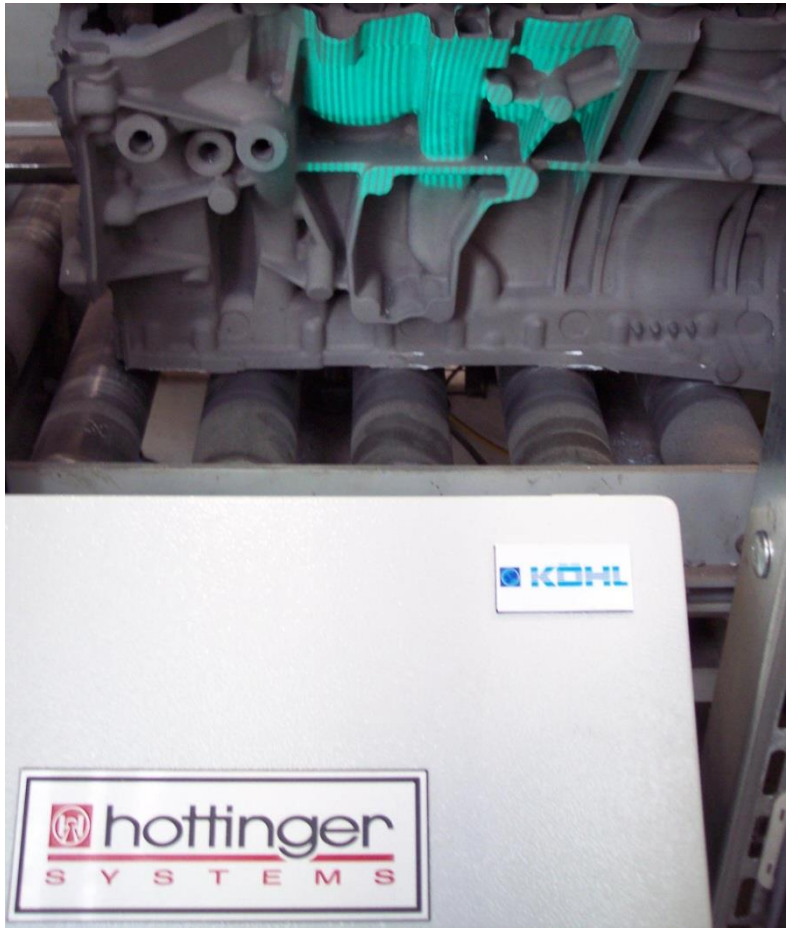
Anwendungsbeispiel: OCR einer Reliefschrift mit Matched-Filtern (1)

Reliefschrift auf einem Gussteil

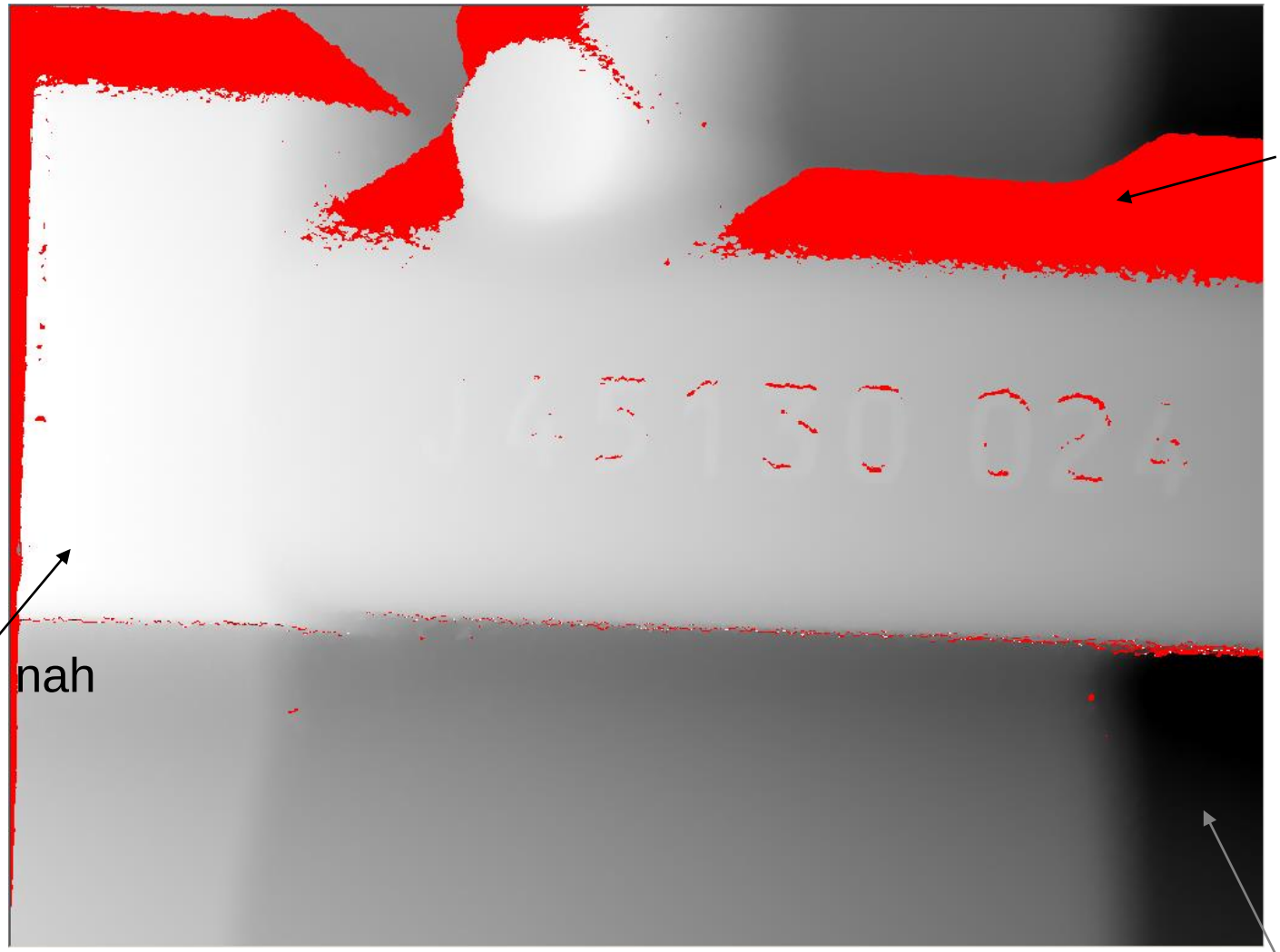
Anwendungsbeispiel: OCR einer Reliefschrift mit Matched-Filtern (2)

Reliefaufnahme mittels Streifenprojektion

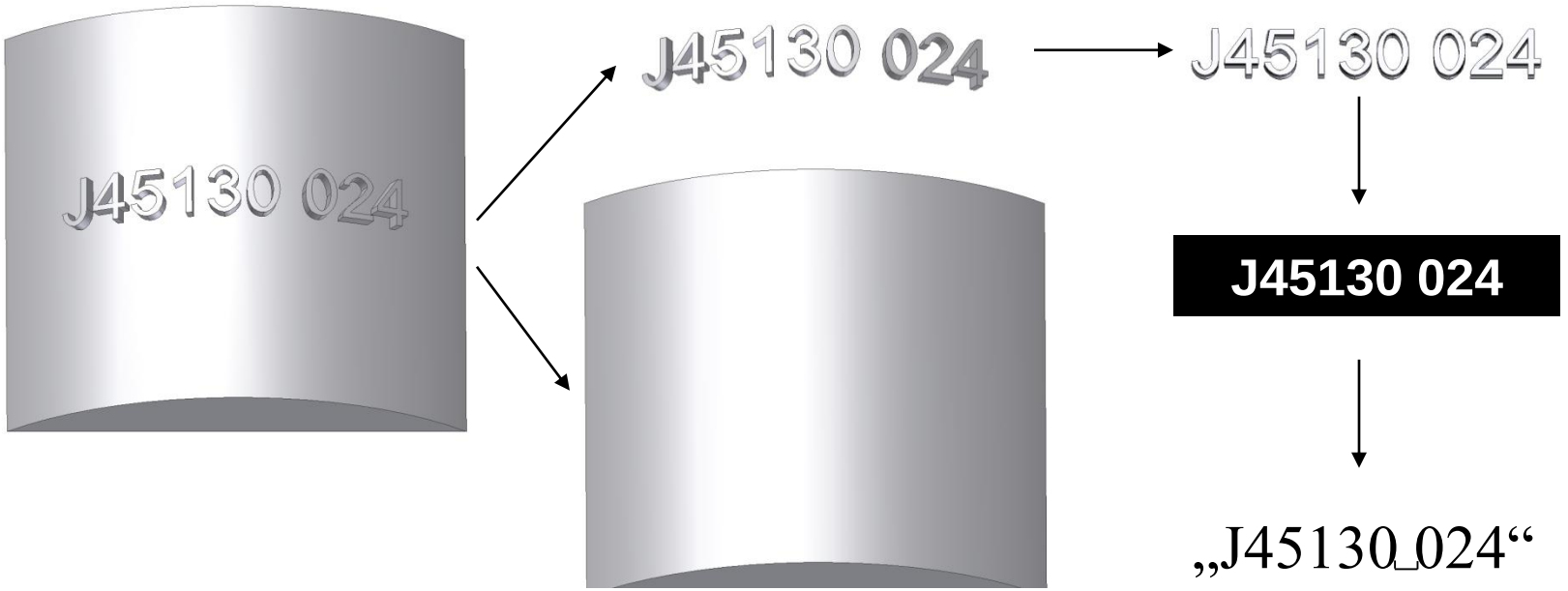




Anwendungsbeispiel: OCR einer Reliefschrift mit Matched-Filtern (3)



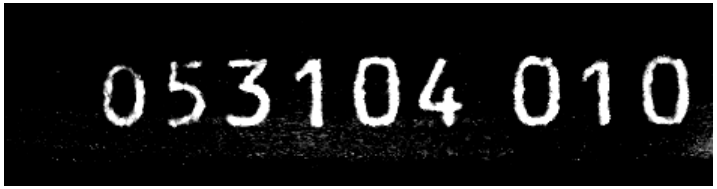
Anwendungsbeispiel: OCR einer Reliefschrift mit Matched-Filtern (4)



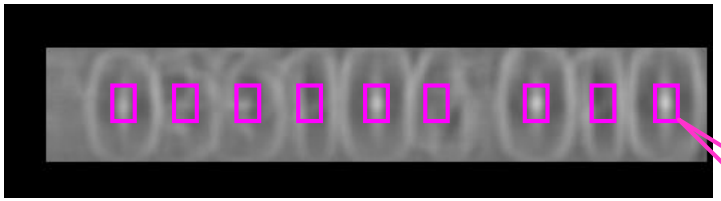
- Separation der Reliefinformation
- Abwicklung/Projektion auf Referenzebene
- Grauwertkodiertes Tiefenbild
- 2D-Zeichenerkennung

7.6. Matched-Filter

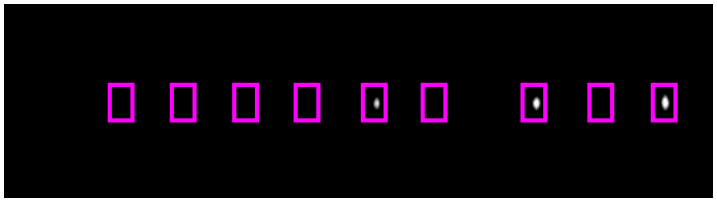
Anwendungsbeispiel:
OCR mit Matched-Filtern (5)



Rohdaten

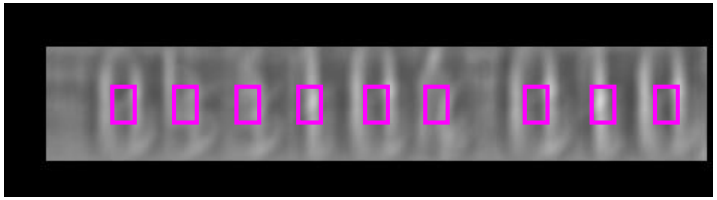


Matched-Filterung mit „0“

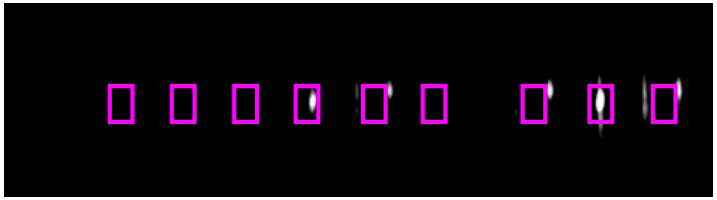


nach Schwellwertbildung

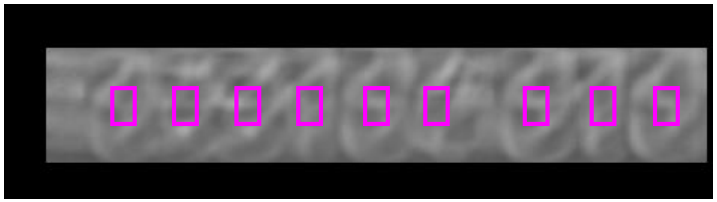
Regions of Interest (RoI)



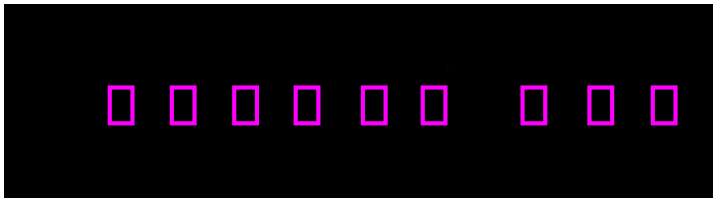
Matched-Filterung mit „1“



nach Schwellwertbildung



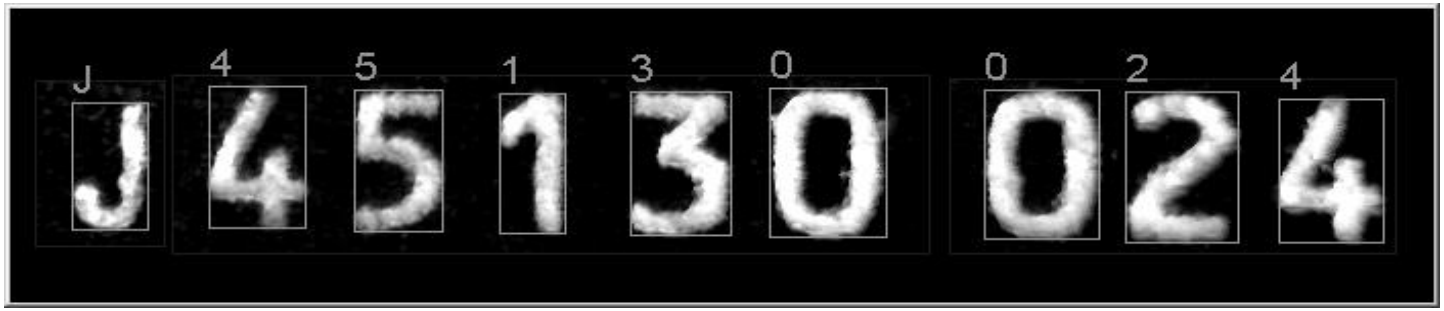
Matched-Filterung mit „2“



nach Schwellwertbildung

OCR-Resultat

Anwendungsbeispiel: OCR einer Reliefschrift mit Matched-Filtern (6)



7.7. Klassifikation von Sequenzen

Bsp.: Erkennung gesprochener Wörter bzw. von Buchstaben

Jeder Buchstabe stellt eine Klasse dar: $\omega_1 \hat{=} \text{A},\ \omega_2 \hat{=} \text{B},\ \omega_3 \hat{=} \text{C}, \ldots$ usw.

Wörter werden von einer Quelle generiert, die sequentiell diese Klassen als Zustände annimmt.

Die Buchstaben in einem Wort sind nicht unabhängig von ihren Nachbarbuchstaben. → Eine Buchstabenerkennung wird leistungsfähiger sein, wenn diese Abhängigkeiten modelliert und berücksichtigt werden.

Nicht die Buchstaben selbst werden beobachtet – sie werden als unbeobachtbar angesehen – sondern nur die mit ihnen assoziierten Laute (nach einer entsprechenden Signalauswertung). Beobachtungen:

$v_1 \hat{=} [\Lambda], v_2 \hat{=} [\text{æ}], v_3 \hat{=} [a:]$,....usw.

Ziel: Erkennung von Wörtern, d.h. der zugehörigen Zustandssequenzen (Buchstabensequenzen) anhand der beobachteten Lautsequenzen.

7.7. Klassifikation von Sequenzen

Diskretes Markov-Modell l-ter Ordnung:

$$P(\omega(t+1) \mid \omega(t), \omega(t-1), \omega(t-2), \ldots) = P(\omega(t+1) \mid \omega(t), \omega(t-1), \ldots, \omega(t-l+1))$$

$$\forall t \in \square \quad \omega(t) \in \Omega/\sim = \{\omega_1, \ldots, \omega_c\}$$

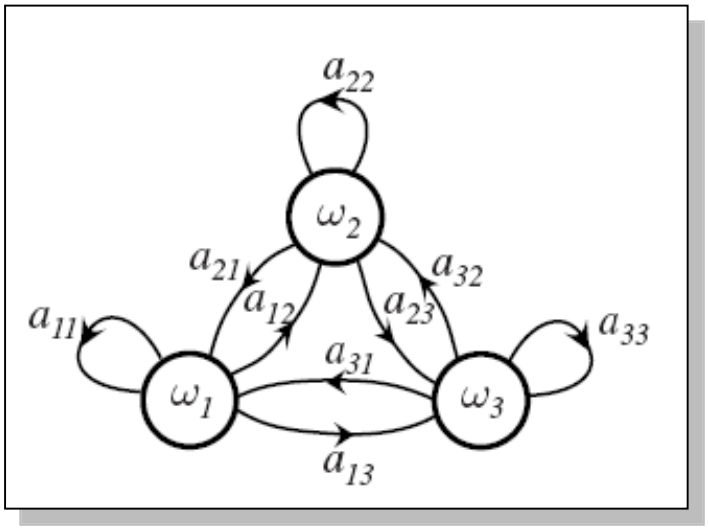
Diskretes Markov-Modell erster Ordnung:

$$P(\omega(t+1) \mid \omega(t), \omega(t-1), \omega(t-2), \ldots) = P(\omega(t+1) \mid \omega(t))$$

Übergangswahrscheinlichkeiten: $a_{ij} := P(\omega_j(t+1) \mid \omega_i(t))$

A Priori Wahrscheinlichkeiten: $P(\omega_i)$

Beispiel für ein Markov-Modell erster Ordnung mit 3 Zuständen:



Quelle: R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification

7.7. Klassifikation von Sequenzen

Bsp.: Buchstabensequenzen von Markov-Modellen unterschiedlicher Ordnung:

Quelle: R. Hoffmann: Signalanalyse und –erkennung. Springer 1998

Ordnung der MQ	Realisierungsbeispiele
	Deutsch (KÜPFMÜLLER 1954[a])
0	EME GKNEET ERS TITBL BTZENFNDBGD EAI E LASZ BETEATR IASMIRCH EGEOM
1	AUSZ KEINU WONDINGLIN DUFRN ISAR STEISBERER ITEHM ANORER
2	PLANZEUDGES PHIN INE UNDEN VEBEICHT GES AUF ES SO UNG GAN DICH WANDERSO
3	ICH FOLGEMAESZIG BIS STEHEN DISPONIN SEELE NAMEN
	Englisch (SHANNON 1948[b])
0	OCRO HLI RGWR NMIELWIS EULL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH BRL
1	ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEA-MY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANOY TOBE SEACE CTISBE
2	IN NO IST LAT WHEY CRACTICT FROURE BIRS GRO-CID PONDENOME OF DEMONSTURES OF THE REP-TAGIN IS REGOACTIONA OF CRE
	Russisch (DOBRUŠIN 1961[c])
0	EYNT CIJA'A OERV ODNG 'UEMLOLJK Z-JA ENVTŠA
1	UMARONO KAČ VSVANNYJ ROSJA NYCH KOVKROV NEDARE
2	POKAK POT DURNOSKAKA NAKONEPNO ZNE STVO-LOVIL SE TVOJ O-NIL'
3	VESEL VRAT'SJA NE SUCHOM I NEPO I KORKO

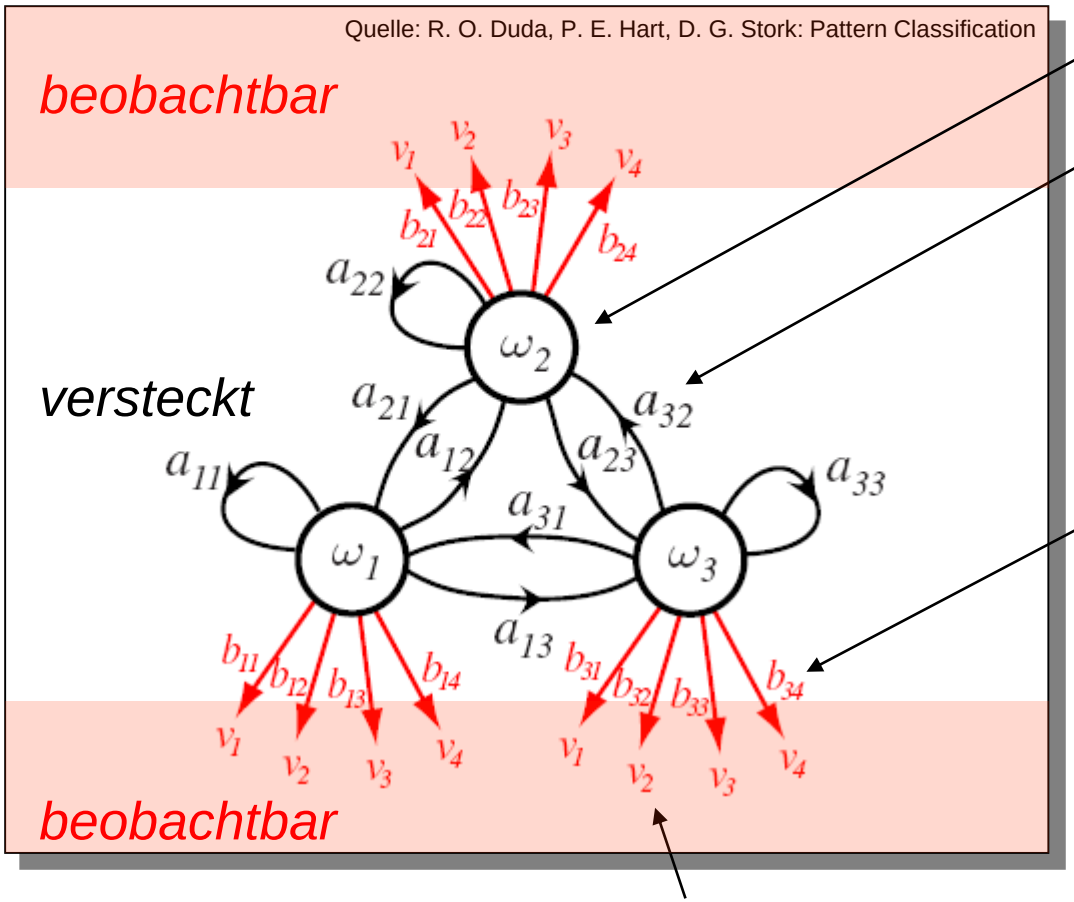
[a] KÜPFMÜLLER, K.: Die Entropie der deutschen Sprache. FTZ 7 (1954) 6, S. 265 - 272.

[b] SHANNON, C. E.: Collected papers, ed. by N. J. A. SLOANE and A. D. WYNER. New York: IEEE 1993, S. 14.

[c] In bibliothekarischer Transliteration zitiert nach JAGLOM, A. M.; JAGLOM, I. M.: Wahrscheinlichkeit und Information (Übers. a. d. Russ.). Berlin: Dt. Verl. der Wissenschaften, 3. Aufl. 1967, S. 200 - 204.

Hidden Markov-Models (HMM)

Bsp. für ein diskretes HMM erster Ordnung:



Beobachtungen

Zustände; A Priori WV $P(\omega)$

$a_{ij} := P(\omega_j(t+1) \mid \omega_i(t))$

Übergangswahrscheinlich-keiten zwischen Zuständen.

$$\sum_j a_{ij} = 1 \quad \forall i$$

$b_{jk} := P(v_k(t) \mid \omega_j(t))$

Eintrittswahrscheinlichkeiten der Beobachtungen gegeben der Zustände.

$$\sum_k b_{jk} = 1 \quad \forall j$$

7.7. Klassifikation von Sequenzen

Wichtigste Aufgabenstellungen bei HMM:

Auswertung (Vorwärtsproblem):
Gegeben sei ein HMM mit allen Wahrscheinlichkeiten a_{ij} und b_{jk}.

Wie groß ist die Wahrscheinlichkeit einer bestimmten Sequenz von Beobachtungen $v(1),\ldots,v(T)$?

Dekodierung (Rückwärtsproblem)*:
Gegeben sei ein HMM mit allen Wahrscheinlichkeiten a_{ij} und b_{jk} sowie eine Sequenz von Beobachtungen $v(1),\ldots,v(T)$.

Wie lautet die wahrscheinlichste Zustandssequenz $\omega(1),\ldots,\omega(T)$, welche die gegebene Beobachtungssequenz erzeugt haben könnte?
→ Viterbi-Algorithmus

Lernen (Parameterschätzung)*:
Gegeben sei die Anzahl der Zustände, die Menge der möglichen Beobachtungen des HMM, sowie eine Menge von Trainingssequenzen $\{v(1),\ldots,v(T)\}$.

Wie können die Parameter a_{ij} und b_{jk} geschätzt werden?
→ ML-Schätzung mit Expectation Maximation (EM)

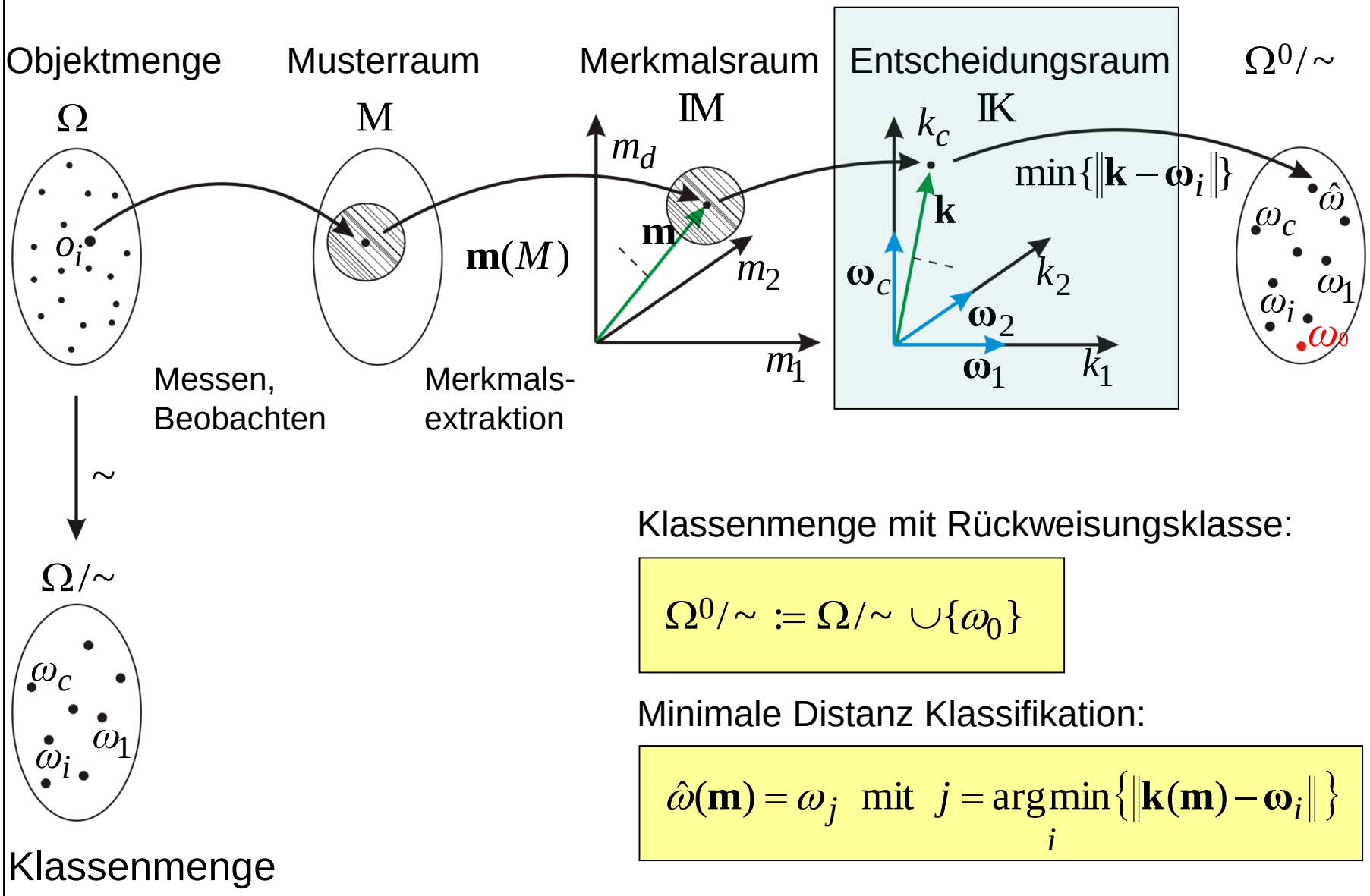
*Besondere Bedeutung für die Mustererkennung

7.7. Klassifikation von Sequenzen

Bemerkungen:

- Erfolgreich eingesetzt in Handschriften-, Gesten- und Spracherkennung
- *Brute Force* Methoden zur Lösung der drei wichtigsten Aufgaben im Zusammenhang mit HMM scheitern an der exponentiell mit der Länge T der Zustandssequenzen steigenden kombinatorischen Komplexität, da die Zahl möglicher Sequenzen c^T beträgt.
- Weiterführende Literatur:
  - T. K. Moon, W. C. Stirling:
    Mathematical Methods and Algorithms. Prentice Hall 2000.
  - G.A. Fink:
    Mustererkennung mit Markov-Modellen – Theorie – Praxis – Anwendungsgebiete. Teubner 2003

7.8. Klassifikation mit Rückweisung



Klassenmenge mit Rückweisungsklasse:

$$\Omega^0/\sim \;:=\; \Omega/\sim \cup\{\omega_0\}$$

Minimale Distanz Klassifikation:

$$\hat{\omega}(\mathbf{m}) = \omega_j \quad \text{mit} \quad j = \arg\min_i \left\{ \|\mathbf{k}(\mathbf{m}) - \boldsymbol{\omega}_i\| \right\}$$

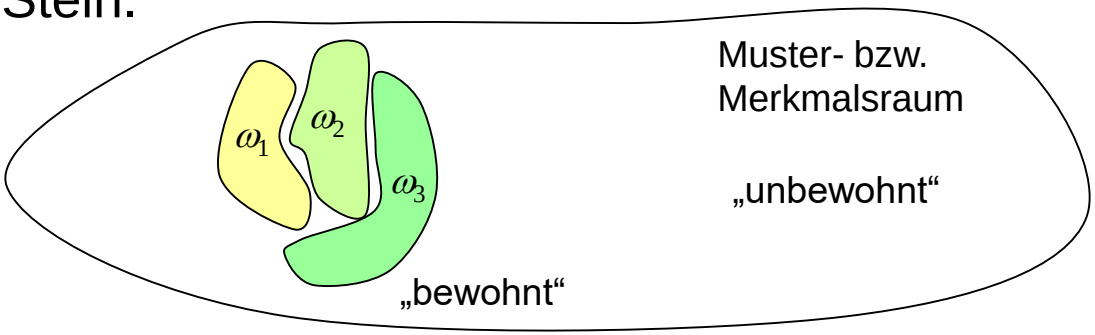
7.8. Klassifikation mit Rückweisung

Der Beschreibung der Klassen mittels $\Omega/\sim = \{\omega_1, \ldots, \omega_c\}$ liegt eine „Abgeschlossene Welt Annahme“ (*Closed World Assumption*) zugrunde. Man beschreibt, was man kennt und lässt alles andere unberücksichtigt.

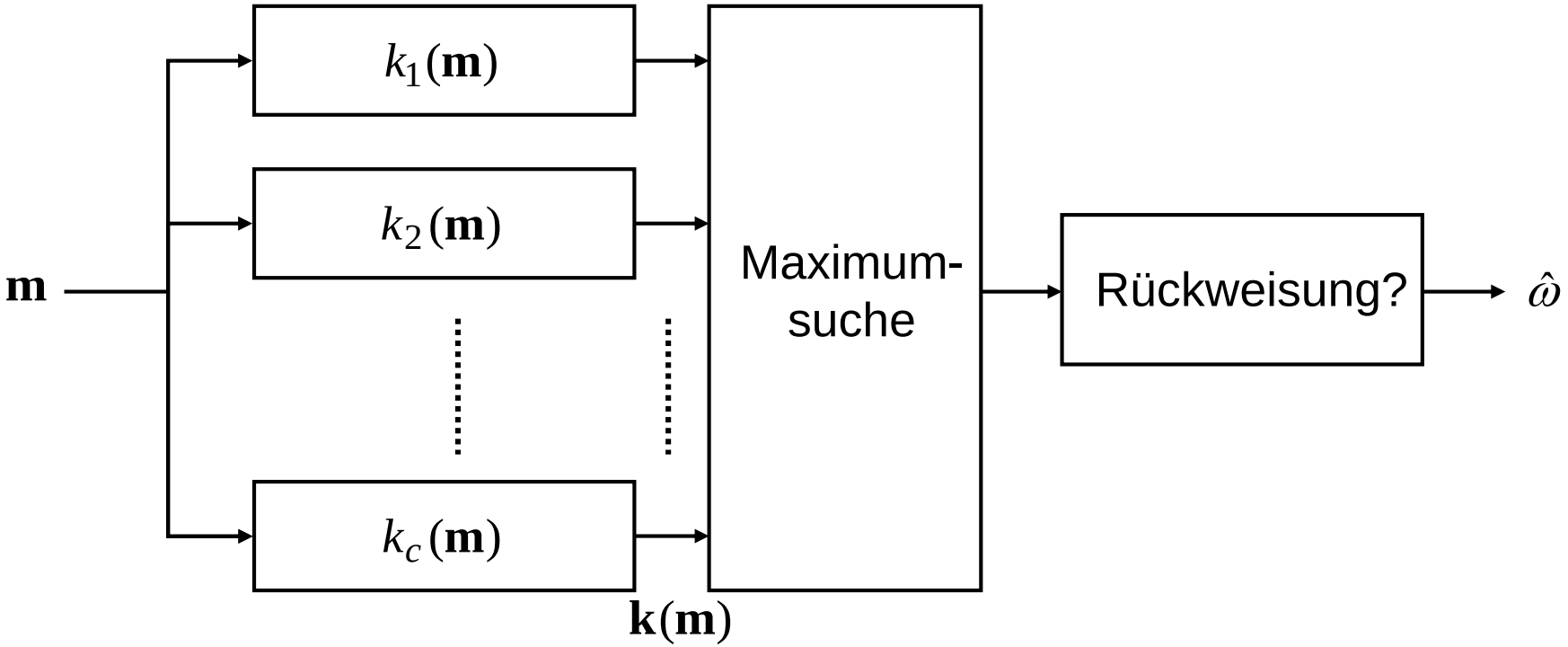
Bsp.: Ein Klassifikator für Obst und Gemüse an einer intelligenten Waage lässt andere Objekte der Welt unberücksichtigt.

Rückweisungsgründe:

- Unentschiedene Situation: die k_i bilden kein signifikantes Maximum aus.
  Beispiel „intelligente Waage“: Birne, die wie ein Apfel aussieht.
- Vollkommen unbekanntes Objekt jenseits der mittels $\Omega/\sim$ beschriebenen Domäne liegt vor.
  Beispiel „intelligente Waage“: Objekt, das weder Obst noch Gemüse ist, wird dargeboten, z.B. ein Stein.



7.8. Klassifikation mit Rückweisung

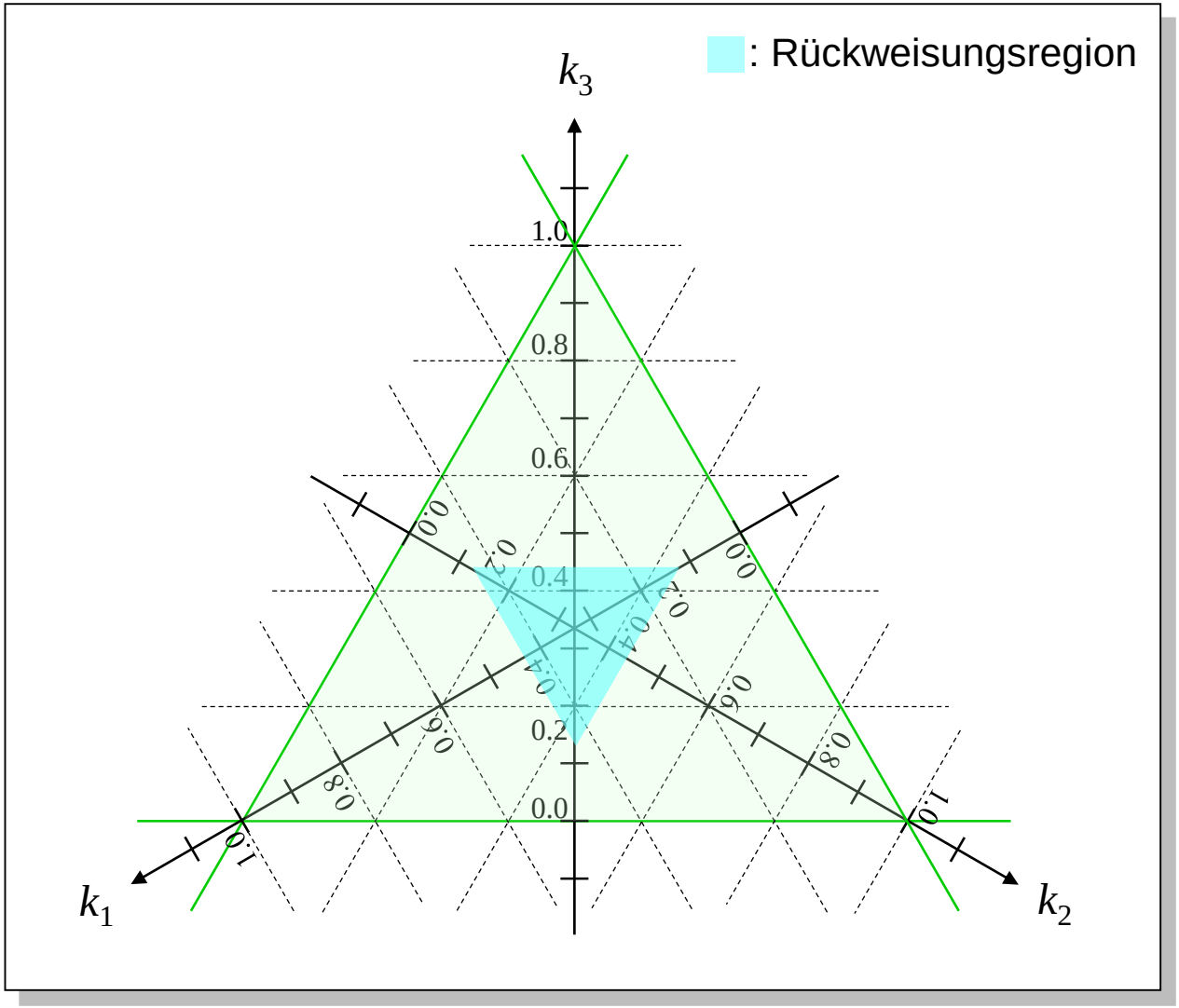


Rückweisungskriterien:

- Maximum-Kriterium: Rückweisung falls $\max\{k_i\} <$ Schwellwert
- Differenz-Kriterium:
  Rückweisung falls $\max\{k_i\} - \max\{\{k_j\} \backslash \max\{k_i\}\} <$ Schwellwert
- Abstands-Kriterium:
  Rückweisung falls $\min\{|| \mathbf{k} - \boldsymbol{\omega}_i ||\} >$ Schwellwert
- Minimum-Kriterium: Rückweisung falls $\min\{k_i\} <$ Schwellwert < 0

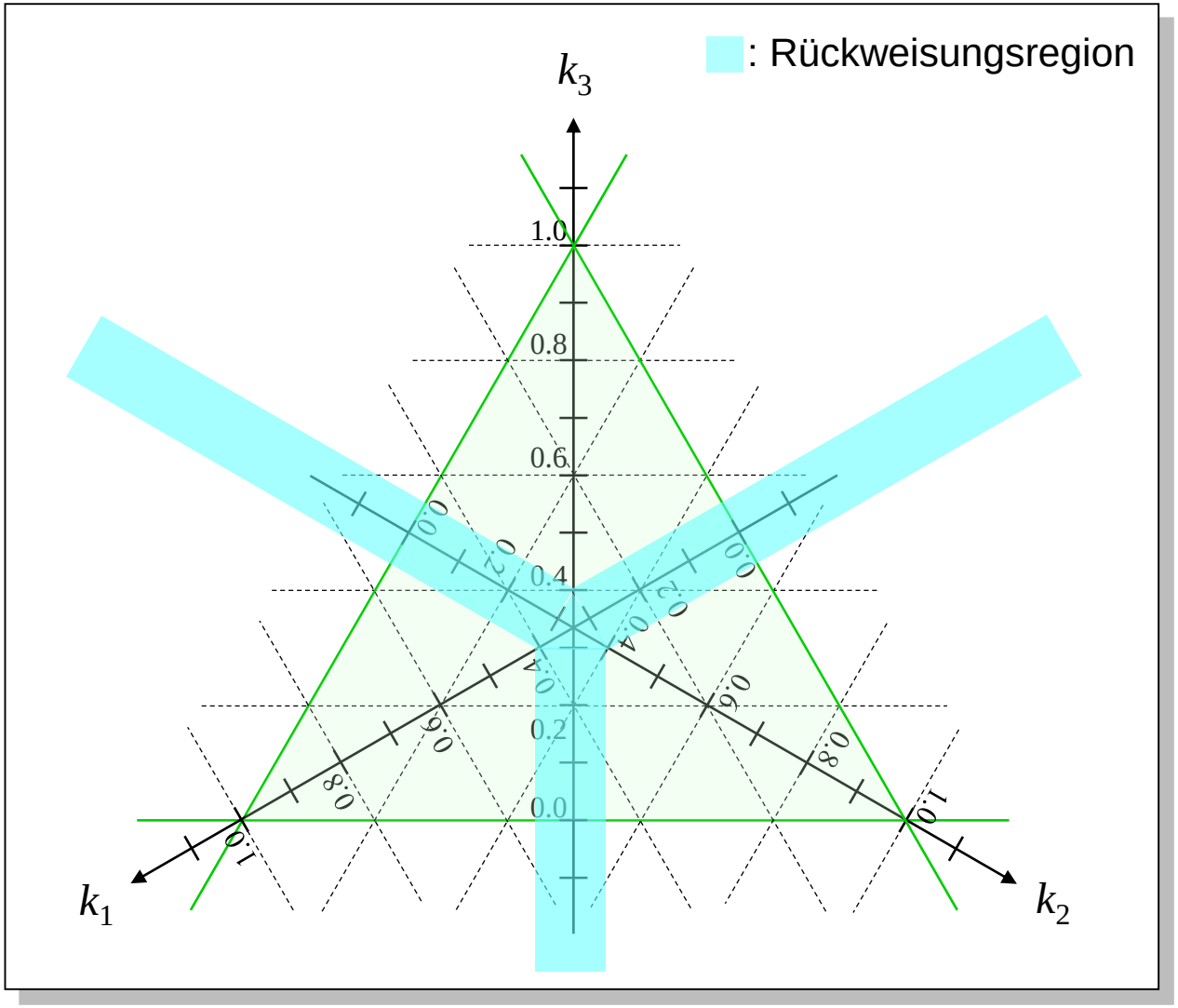
7.8. Klassifikation mit Rückweisung

Maximum-Kriterium: Rückweisungsregion (→ ω_0) im Entscheidungsraum



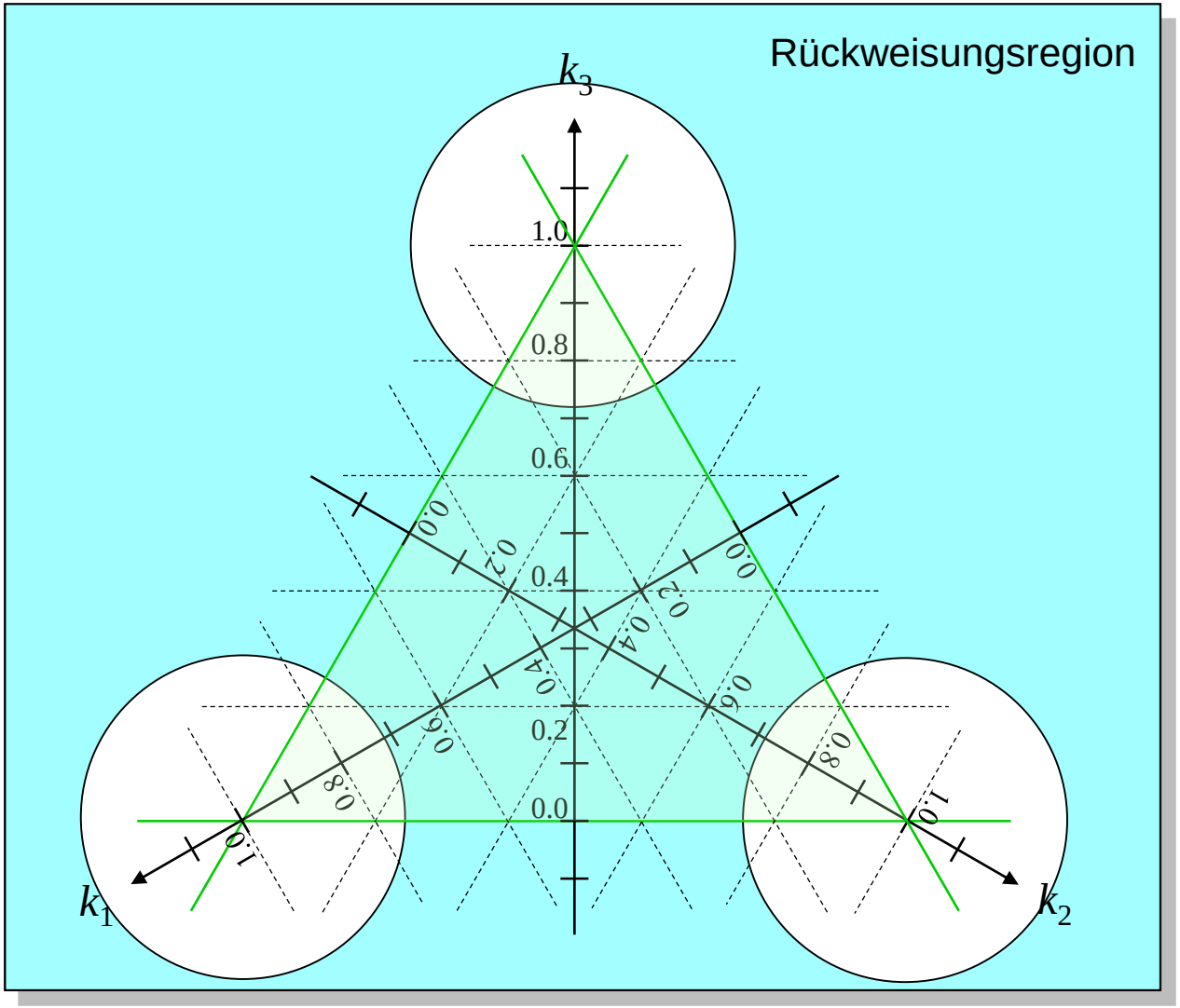
7.8. Klassifikation mit Rückweisung

Differenz-Kriterium: Rückweisungsregion (→ ω_0) im Entscheidungsraum



7.8. Klassifikation mit Rückweisung

Abstands-Kriterium: Rückweisungsregion (→ ω_0) im Entscheidungsraum



7.8. Klassifikation mit Rückweisung

Minimum-Kriterium: Rückweisungsregion (→ ω_0) im Entscheidungsraum