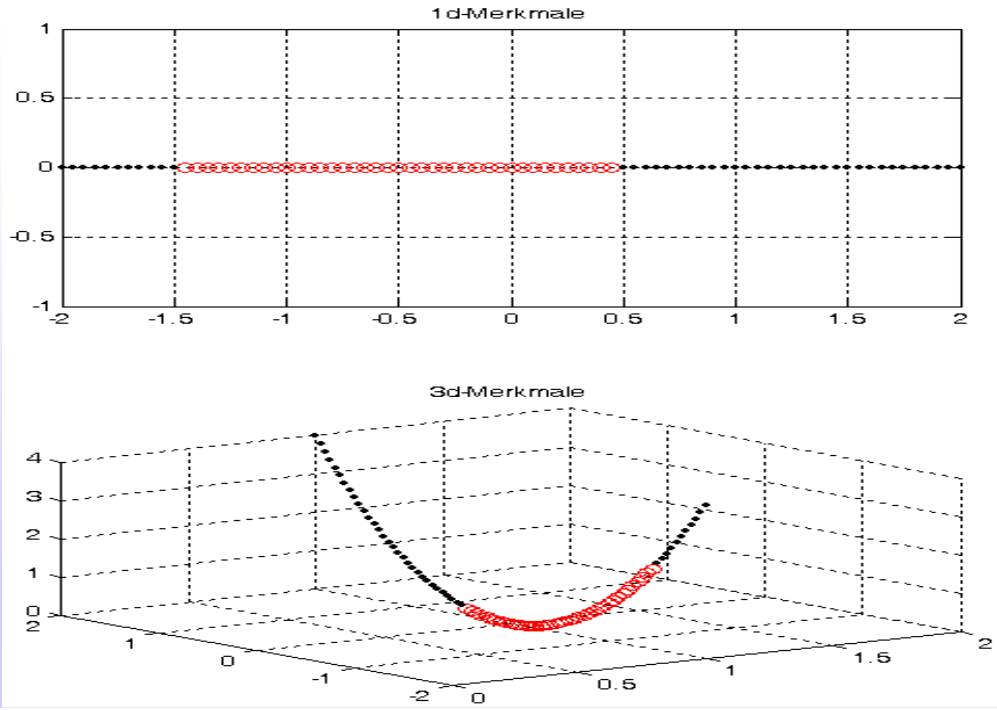
Lineare Diskriminanten

- *Verallgemeinerte Diskriminanten:*

- Beliebige Basisfunktionen

$$g(\mathbf{x}) = \sum_{i=0}^{\hat{d}} w_i \, \varphi_i(\mathbf{x}), \qquad \varphi_0(\mathbf{x}) = 1;\ \hat{d} >> d \qquad \varphi(\mathbf{x}) = \begin{pmatrix} 1 \\ \mathbf{x} \\ \mathbf{x}^2 \end{pmatrix}$$



plot1Dnach3Dsurface.m

Lineare Diskriminanten

- *Verallgemeinerte Diskriminanten:*

$$g(\mathbf{x}) = \sum_{i=0}^{d} w_i \, x_i, \qquad x_0 = 1$$

- Entscheidungsgrenzen müssen nicht linear in den Parametern sein
- Komplexität erfordert oft höher-dimensionale Reihenentwicklungen

$$g(\mathbf{x}) = \sum_{i=0}^{\hat{d}} w_i \, \varphi_i(\mathbf{x}), \qquad \varphi_0(\mathbf{x}) = 1; \; \hat{d} >> d$$

- Häufig polynomialer Ansatz

$$\varphi(\mathbf{x}) = \begin{pmatrix} 1 \\ \mathbf{x} \\ \mathbf{x}^2 \end{pmatrix}$$

Lineare Diskriminanten

- *Verallgemeinerte Diskriminanten:*

$$g(\mathbf{x}) = \sum_{i=0}^{d} w_i \, x_i, \qquad x_0 = 1$$

- Entscheidungsgrenzen müssen nicht linear in den Parametern sein
- Komplexität erfordert oft höher-dimensionale Reihenentwicklungen

$$g(\mathbf{x}) = \sum_{i=0}^{\hat{d}} w_i \, \varphi_i(\mathbf{x}), \qquad \varphi_0(\mathbf{x}) = 1; \; \hat{d} >> d$$

- Häufig Potenzreihe

$$\exp^{\mathbf{x}} = \sum_{n=0}^{\infty} \frac{\mathbf{x}}{n!} \quad \Rightarrow \quad \varphi(\mathbf{x}) = \begin{pmatrix} 1 \\ \mathbf{x} \\ \mathbf{x}^2/2! \\ \mathbf{x}^3/3! \\ \mathbf{x}^4/4! \\ \dots \end{pmatrix}$$

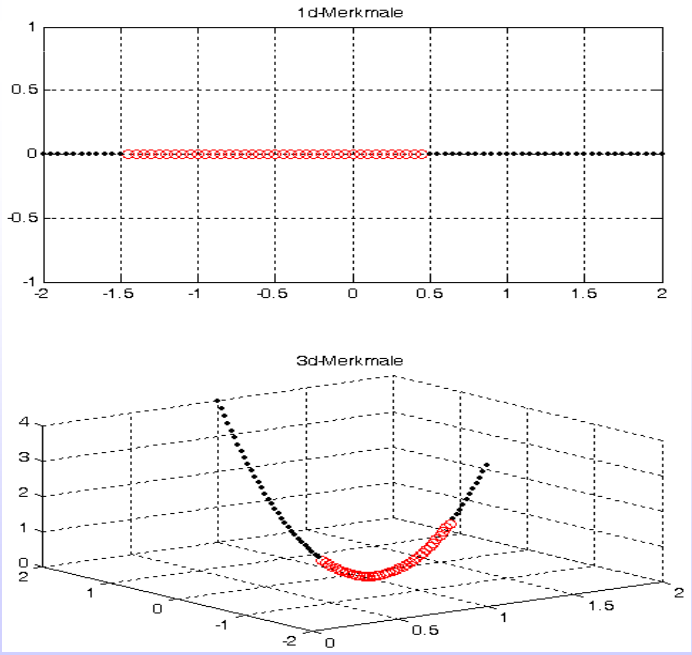
Lineare Diskriminanten

- *Verallgemeinerte Diskriminanten:*

- Beliebige Funktionen
- $\mathbf{x} \in \Re^d, \quad \varphi(\mathbf{x}) \in \Re^{\hat{d}}$

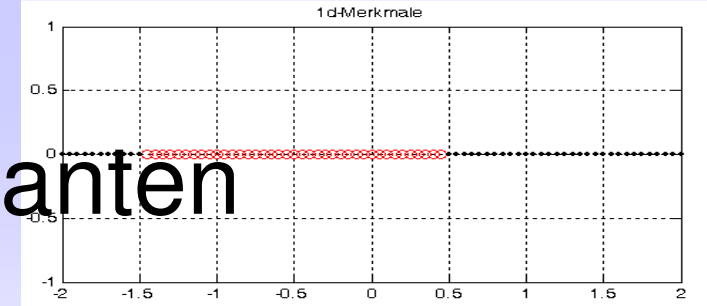
$$g(\mathbf{x}) = \sum_{i=0}^{\hat{d}} w_i \, \varphi_i(\mathbf{x})$$

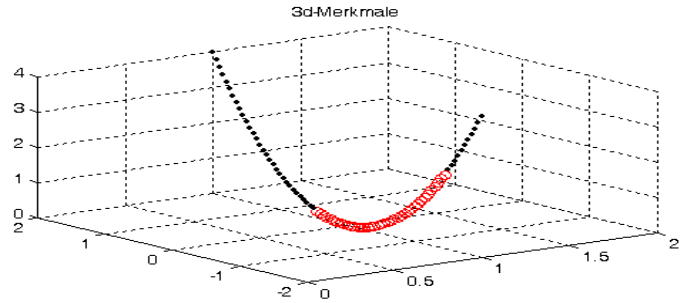
- linear in ϕ
- Hyperebenen



- einfache Diskriminante in $\Re^{\hat{d}}$
  kann sehr komplex in $\Re^d$ sein

Lineare Diskriminanten

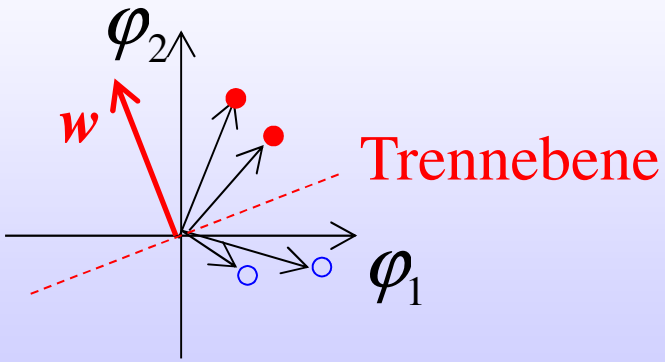


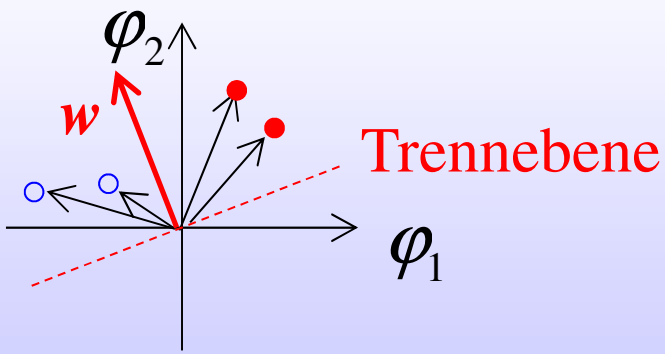


- *Verallgemeinerte Diskriminanten:*
- Berechnung Gewichte
- 2-Klassen, linear separierbar

$$g(\mathbf{x}) = \mathbf{w}^T \cdot \varphi(\mathbf{x})$$

- Merkmal φ_k zu ω_1, wenn $\mathbf{w}^T \cdot \varphi(\mathbf{x}_k) > 0$, sonst zu ω_2
- Normierung: bei allen Merkmale zu ω_2: Vorzeichen tauschen



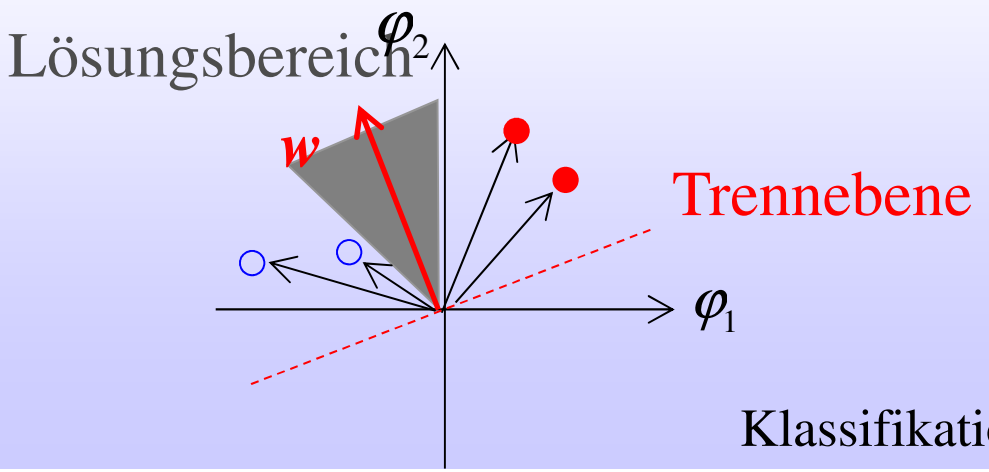


Lineare Diskriminanten

- *Verallgemeinerte Diskriminanten:*
- aus Trainingsdaten: Diskriminanten finden
- Finden der Diskriminanten ⇔ Minimierung von Kriterium J

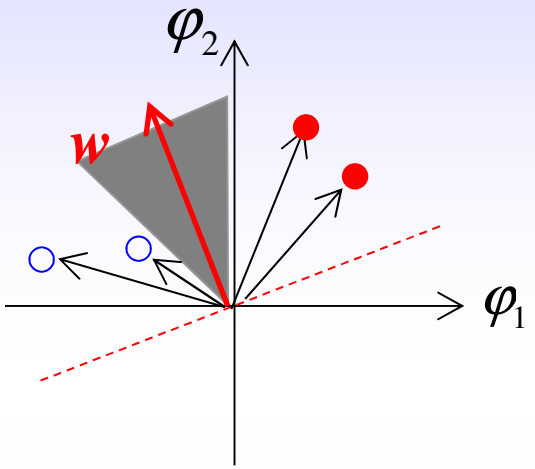
- Gesucht Gewicht ***w***, so dass $\mathbf{w}^T \cdot \varphi_k > 0 \ \forall \varphi_k$

- ***w***: Separierender Vektor, Lösungsvektor auf positiver Seite
- Lösungsbereich: Schnittbereich der *n* Halbräume



Lineare Diskriminanten

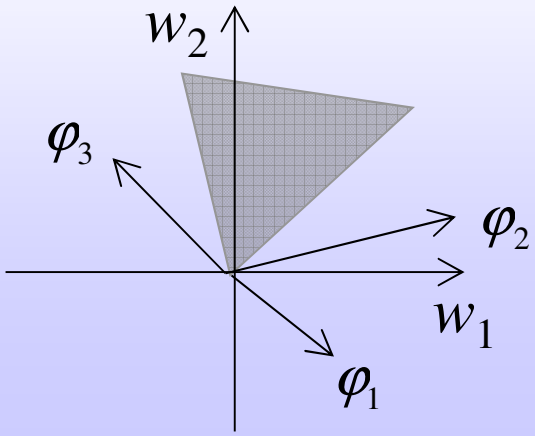
- *Verallgemeinerte Diskriminanten:*
- Berechnung Gewichte
- 2-Klassen, linear separierbar

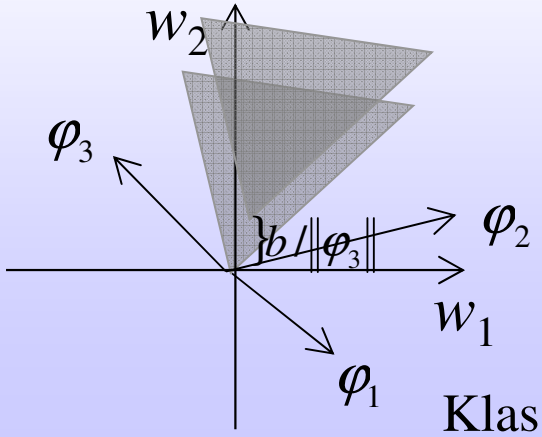


- mehrere Gewichtsvektoren möglich:

1. Maximum $\mathrm{d}_{\min}\left(\varphi_k, H_{ij}\right) \quad \forall \varphi_k$

2. *b* ("Rand"): $\mathbf{w}^T \cdot \varphi_k \geq b > 0 \quad \forall \varphi_k$

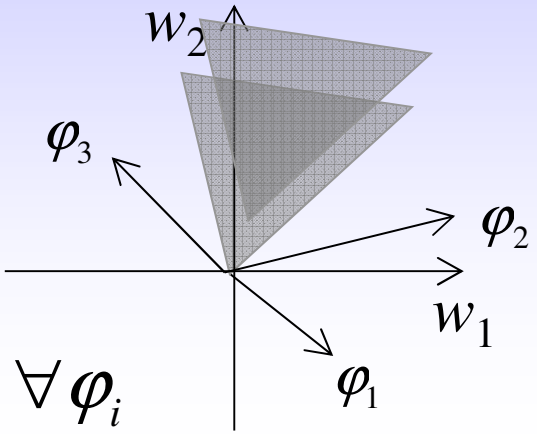




=>Vektor mittig

Lineare Diskriminanten

- *Verallgemeinerte Diskriminanten:*
- Berechnung Gewichte
- 2-Klassen, linear separierbar



- Menge von Ungleichungen $\mathbf{w}^T \cdot \varphi_i \geq b > 0 \quad \forall \varphi_i$

- Vielzahl von Optimierungsverfahren
- Mit unterschiedlicher Konvergenz

Support Vector Machines (SVM)

- Geeignete Abbildung $\varphi(\cdot)$:
  jeder Hyperraum durch Hyperebene separierbar

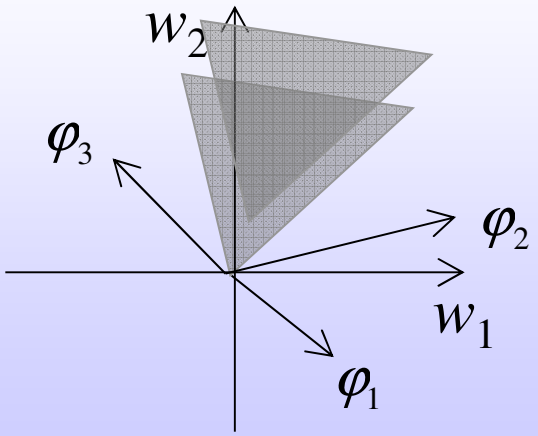
- $g(\varphi(\mathbf{x})) = \mathbf{w}^T \cdot \varphi(\mathbf{x})$ Diskriminante im Hilfsraum

- Labeln: $\varphi_k, k = 1,...,n \quad \mapsto z_k = \begin{cases} +1, \text{wenn } \varphi_k \in \omega_1 \\ -1, \text{wenn } \varphi_k \in \omega_2 \end{cases}$

- Immer: $z_k \cdot g(\varphi_k) \geq 1$

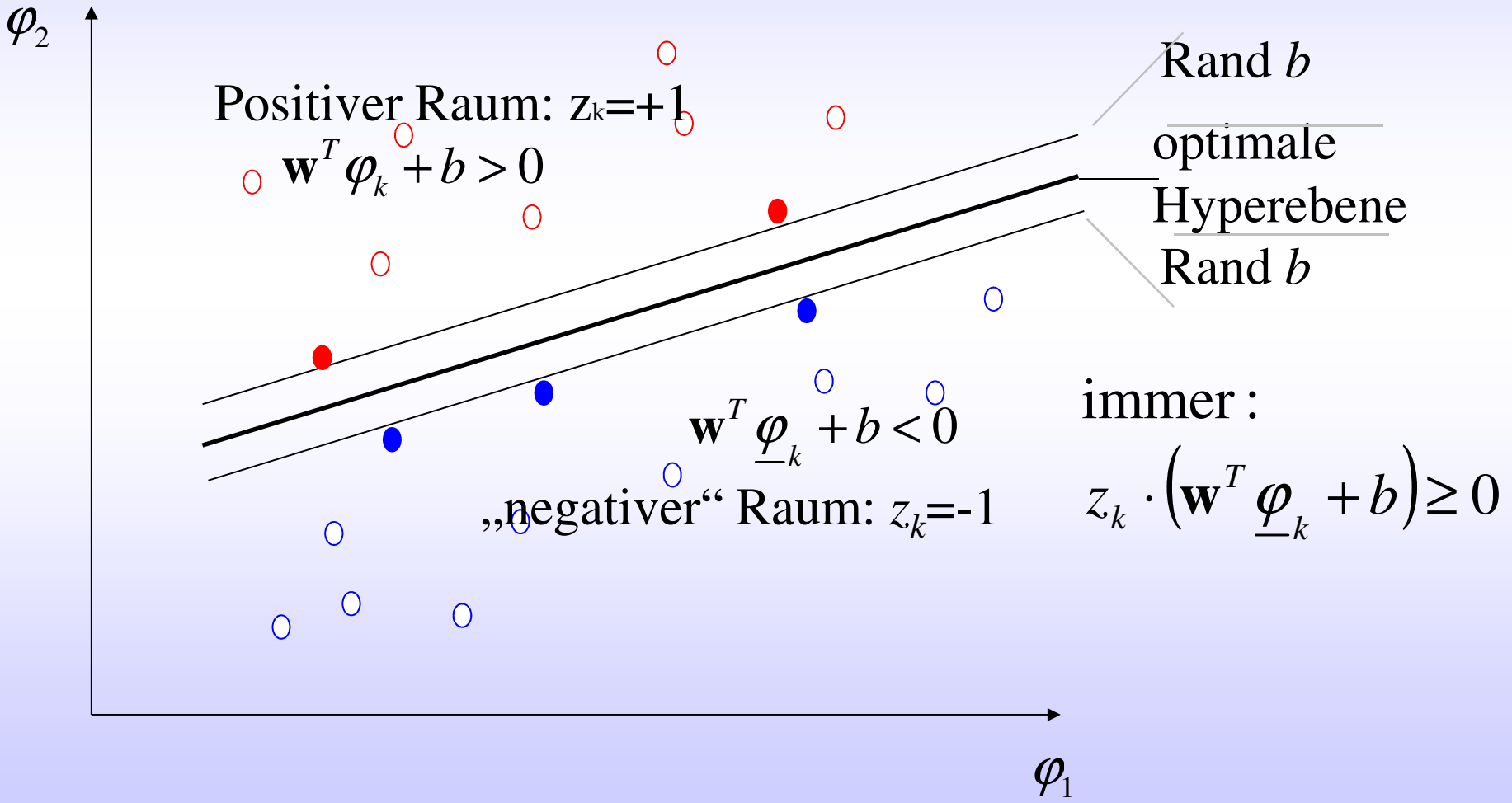
- Größerer Rand b, bessere Generalisierung

$$\frac{z_k \cdot g(\varphi_k)}{\|\mathbf{w}\|} \geq b$$



Support Vector Machines (SVM)

- Support-Vektoren suchen



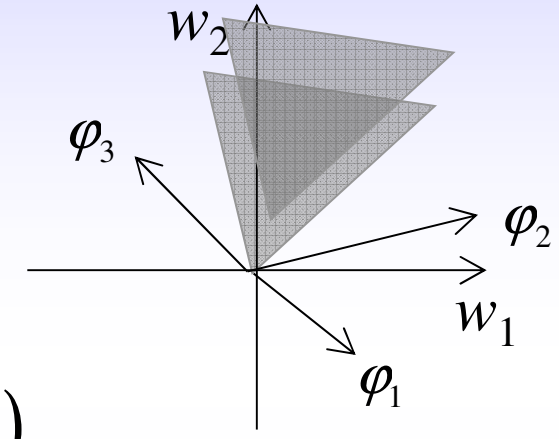
Support Vector Machines (SVM)

- bisher:

$$z_k \cdot \mathrm{g}(\varphi_k) \geq 1 \qquad \mathbf{w}^T \varphi_k + b > 0$$

$$z_k \cdot \mathrm{g}(\varphi_k) - 1 \geq 0$$

$$\frac{z_k \cdot \mathrm{g}(\varphi_k)}{\|\mathbf{w}\|} \geq b \qquad \mathrm{g}(\varphi_k(\mathbf{x})) = \mathbf{w}^T \cdot \varphi_k(\mathbf{x})$$



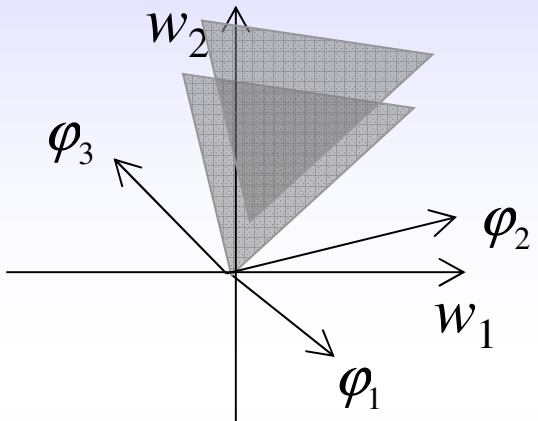
- Ges. ist die Hyperebene:

$$\mathrm{g}_k(\mathbf{w}, b) = -\left(z_k \cdot \mathbf{w}^T \varphi_k + b - 1\right) = 0 \qquad \text{NB: } \mathbf{w}^T \varphi_k + b > 0$$

=> Wie groß ist b am Rand?

Support Vector Machines (SVM)

$$\frac{z_k \cdot \mathrm{g}(\varphi_k)}{\|\mathbf{w}\|} \geq b$$



- Ziel: *w* finden, welche *b* maximiert
  - Minimiere $\|\mathbf{w}\|$

NB: $\mathrm{g}_k(\mathbf{w},b) = -\left(z_k \cdot \mathbf{w}^T \varphi_k + b - 1\right) = 0$

- Lagrange-Multiplikator

$$\mathrm{L}(\mathbf{w},b,\lambda) = \mathrm{f}(\mathbf{w}) + \sum_{k=1}^{n} \lambda_k \, \mathrm{g}_k(\mathbf{w},b)$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{k=1}^{n} \lambda_k \left[z_k \cdot \mathbf{w}^T \varphi_k + b - 1\right]$$

Support Vector Machines (SVM)

- Minimierung des Lagrange-Multiplikators

$$\mathrm{L}(\mathbf{w}, b, \lambda) = \mathrm{f}(\mathbf{w}) + \sum_{k=1}^{n} \lambda_k \, \mathrm{g}_k(\mathbf{w})$$

$$= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{k=1}^{n} \lambda_k \left[z_k \cdot \mathbf{w}^T \varphi_k + b - 1 \right]$$

$$\mathrm{L}(\mathbf{w}, b, \lambda) = \frac{1}{2} \sum_{i=1}^{d} w_i^2 - \sum_{k=1}^{n} \lambda_k \left[z_k \cdot \sum_{i=1}^{d} w_i \, \varphi_k^i + b - 1 \right]$$

$$\frac{\partial \mathrm{L}(\mathbf{w}, b, \lambda)}{\partial w_i} = 0$$

Support Vector Machines (SVM)

- Minimierung des Lagrange-Multiplikators

$$\frac{\partial \mathrm{L}(\mathbf{w},\lambda)}{\partial w_i} = w_i - \sum_{k=1}^{n} \lambda_k \cdot z_k \cdot \varphi_k^i = 0$$

$$\mathbf{w} = \sum_{k=1}^{n} \lambda_k \cdot z_k \cdot \varphi_k$$

- Einsetzen in Lagrange-Multiplikator

$$\mathrm{L}(\mathbf{w},\lambda) = \frac{1}{2}\left(\sum_{j=1}^{n} \lambda_j z_j \varphi_j\right)^2 - \sum_{k=1}^{n} \lambda_k \cdot \left(z_k \cdot \left(\sum_{j=1}^{n} \lambda_j z_j \varphi_j\right) \cdot \varphi_k - 1\right)$$

$$= \sum_{k=1}^{n} \lambda_k - \frac{1}{2} \sum_{j,k=1}^{n} \lambda_j \lambda_k z_j z_k \left\langle \varphi_j, \varphi_k \right\rangle$$

Support Vector Machines (SVM)

- Minimierung des Lagrange-Multiplikators (Gleichungen)

=> Maximierung des Kuhn-Tucker-Konstrukts (Ungleichungen)

$$L(\lambda) = \sum_{k=1}^{n} \lambda_k - \frac{1}{2} \sum_{j,k=1}^{n} \lambda_j \lambda_k z_j z_k \left\langle \varphi_j , \varphi_k \right\rangle$$

- Nebenbedingungen:

$$\sum_{k=1}^{n} \lambda_k z_k = 0 \qquad \lambda_k \geq 0, \;\; k = 1, \ldots, n$$

Support Vector Machines (SVM)

- Minimierung des Lagrange-Multiplikators (Gleichungen)

=> Maximierung des Kuhn-Tucker-Konstrukts (Ungleichungen)

- Multiplikatoren für Stützvektoren

$$\lambda_k \neq 0, \;\; k \in SV$$

- Datenkompression

Support Vector Machines (SVM)

- Lösungsvektor

$$\mathbf{w}^* = \sum_{k \in SV} \left(\lambda_k^* \cdot z_k \right) \cdot \varphi_k$$

- Rand

$$b^* = \frac{\max\left\{ \left\langle \mathbf{w}^*, \varphi_k \right\rangle \Big|_{z_k = -1} \right\} + \min\left\{ \left\langle \mathbf{w}^*, \varphi_k \right\rangle \Big|_{z_k = 1} \right\}}{2}$$

- Klassifikation

$$\mathrm{f}\left(\varphi_{neu}\right) = \mathrm{sign}\left(\left\langle \mathbf{w}, \varphi_{neu} \right\rangle + b \right)$$

$$\mathrm{f}\left(\varphi_{neu}\right) = \mathrm{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot \left\langle \varphi_k, \varphi_{neu} \right\rangle + b \right)$$

Support Vector Machines (SVM)

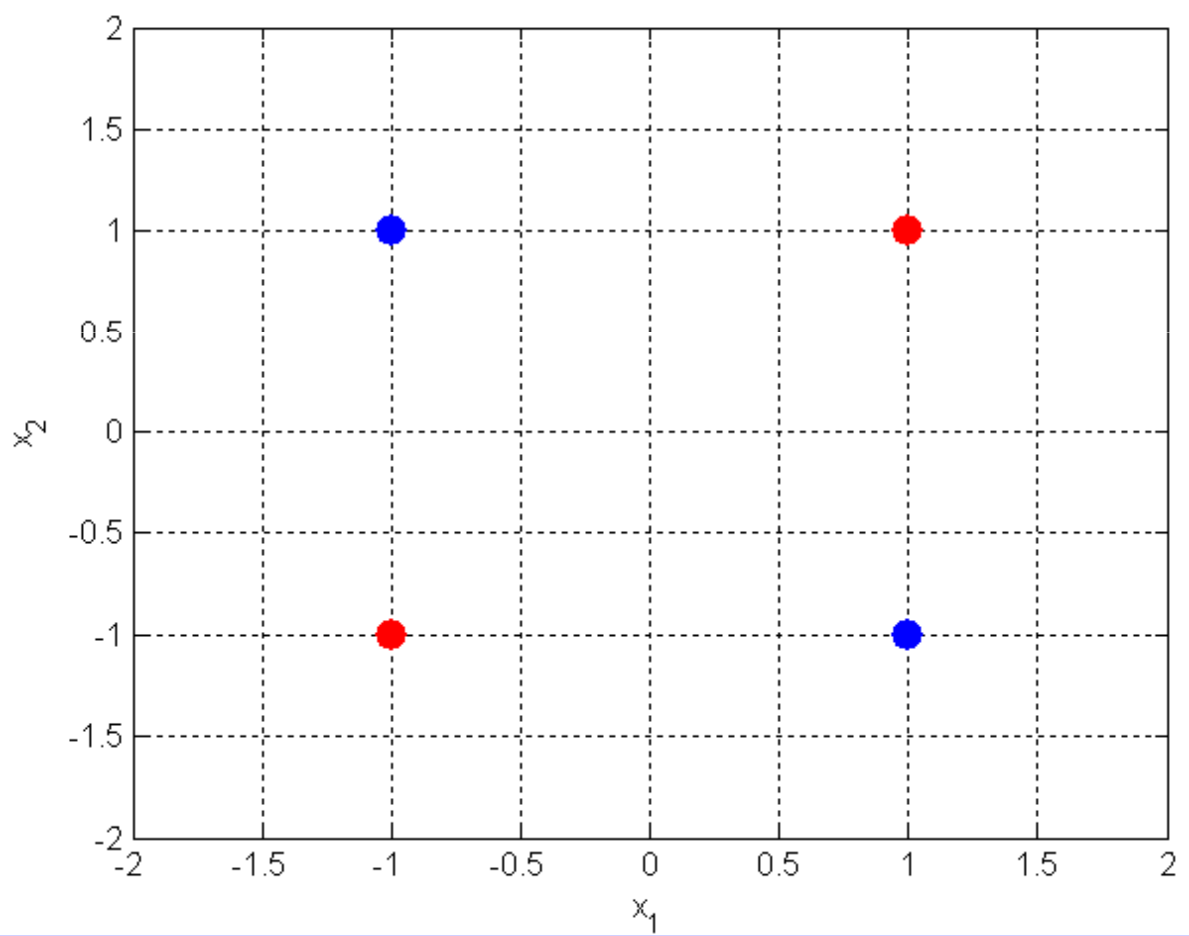
- Klassifikation

$$\mathrm{f}(\varphi_{neu}) = \mathrm{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot \langle \varphi_k, \varphi_{neu} \rangle + b \right)$$

- Nachteil: Transformation $\varphi(\cdot)$ explizit berechnen

Support Vector Machines (SVM)

- Beispiel XOR



Support Vector Machines (SVM)

- Beispiel XOR
- Polynomialentwicklung

$$\varphi(\mathbf{x}) = \begin{pmatrix} 1 \\ \sqrt{2} \cdot x_1 \\ \sqrt{2} \cdot x_2 \\ \sqrt{2} \cdot x_1 x_2 \\ x_1^2 \\ x_2^2 \end{pmatrix}$$

Support Vector Machines (SVM)

- Beispiel XOR

$$\sum_{k=1}^{n} \lambda_k z_k = 0$$

$$\lambda_1 - \lambda_2 + \lambda_3 - \lambda_4 = 0$$

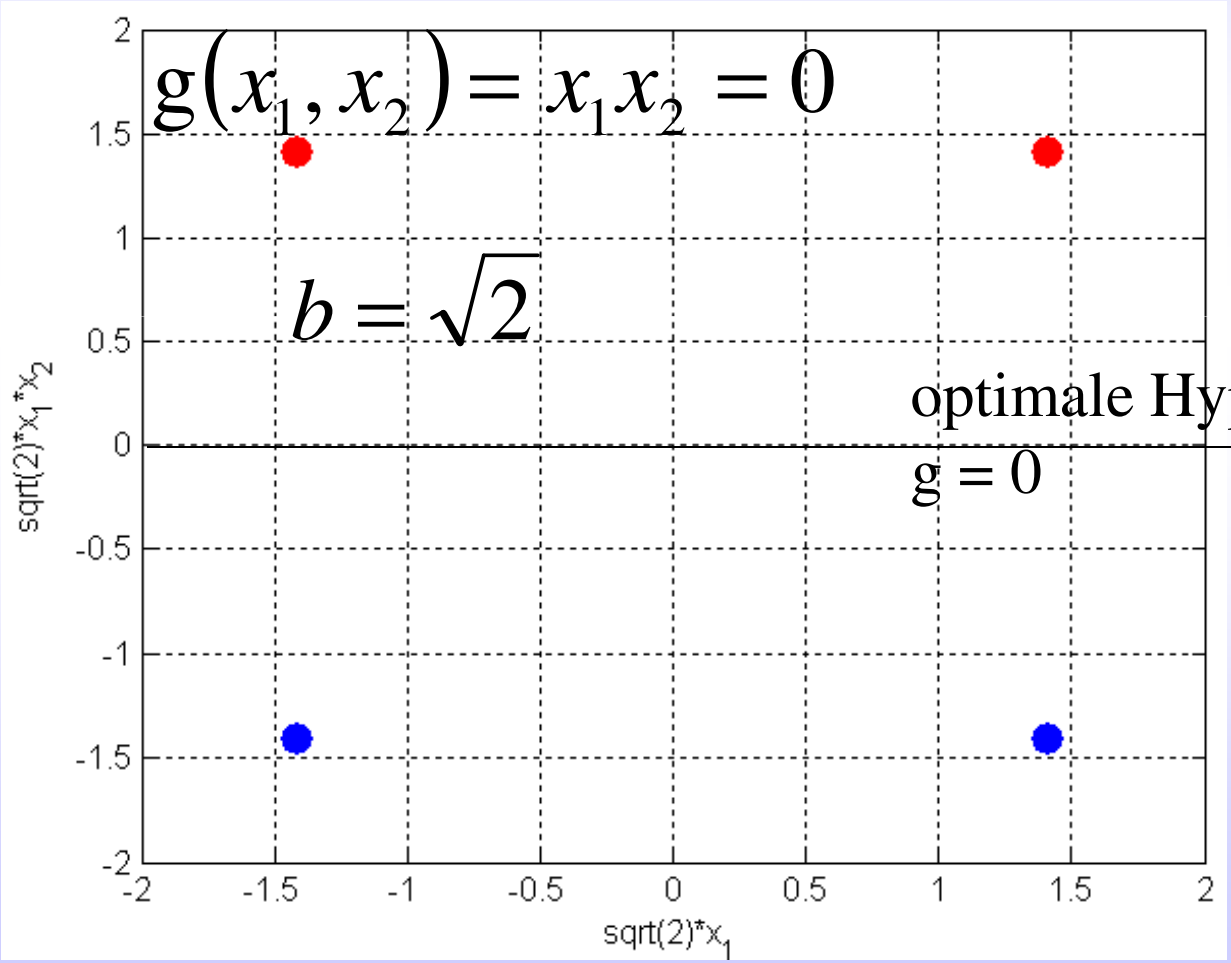
$$\lambda_1 = \lambda_3 \qquad \lambda_2 = \lambda_4$$

- Analytisch:

$$\mathbf{w}_k = \frac{1}{8}, k = 1,...,4 \qquad b = \sqrt{2}$$

Support Vector Machines (SVM)

- Beispiel XOR

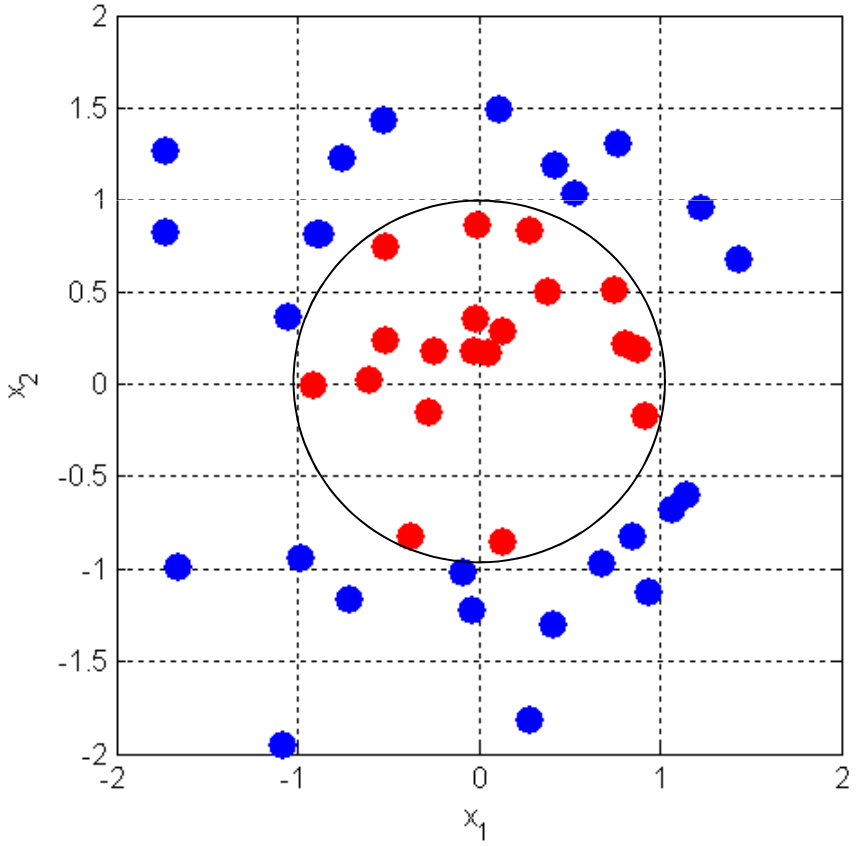


Support Vector Machines (SVM)

- Beispiel 2

$$\varphi : \Re^2 \to \Re^3$$

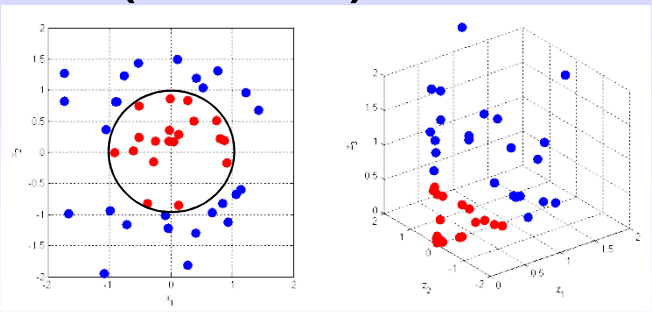
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



Support Vector Machines (SVM)

- Klassifikation

$$\mathrm{f}(\varphi_{neu}) = \mathrm{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot \langle \varphi_k, \varphi_{neu} \rangle + b \right)$$



- allg. Abstände:

$$\langle \mathbf{x}, \mathbf{y} \rangle^d = (\mathbf{x} \cdot \mathbf{y})^d$$

$$= \left(\sum_{k=1}^{n} x_k \cdot y_k \right)^d$$

$$\varphi : \Re^2 \rightarrow \Re^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := \left(x_1^2, \sqrt{2} x_1 x_2, x_2^2\right)$$

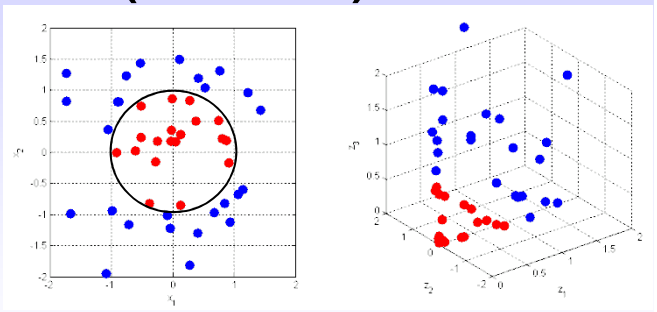
- *d* verzerrt Abstandmaß
- wenn polynomiale Raumverzerrung definiert:

$$= (\varphi(\mathbf{x}) \cdot \varphi(\mathbf{y}))$$

Support Vector Machines (SVM)

- Klassifikation

$$f(\varphi_{neu}) = \text{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot \langle \varphi_k, \varphi_{neu} \rangle + b \right)$$



- allg.: $K(\mathbf{x}, \mathbf{y}) := (\varphi(\mathbf{x}) \cdot \varphi(\mathbf{y}))$

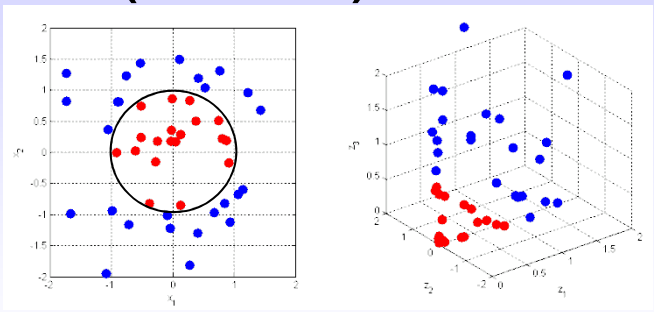
  Kern („kernel“) der SVM

- Kern ersetzt das Skalarprodukt und bewirkt nichtlineares Abstandsmaß

Support Vector Machines (SVM)

- Klassifikation

$$f(\varphi_{neu}) = \text{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot \langle \varphi_k, \varphi_{neu} \rangle + b \right)$$

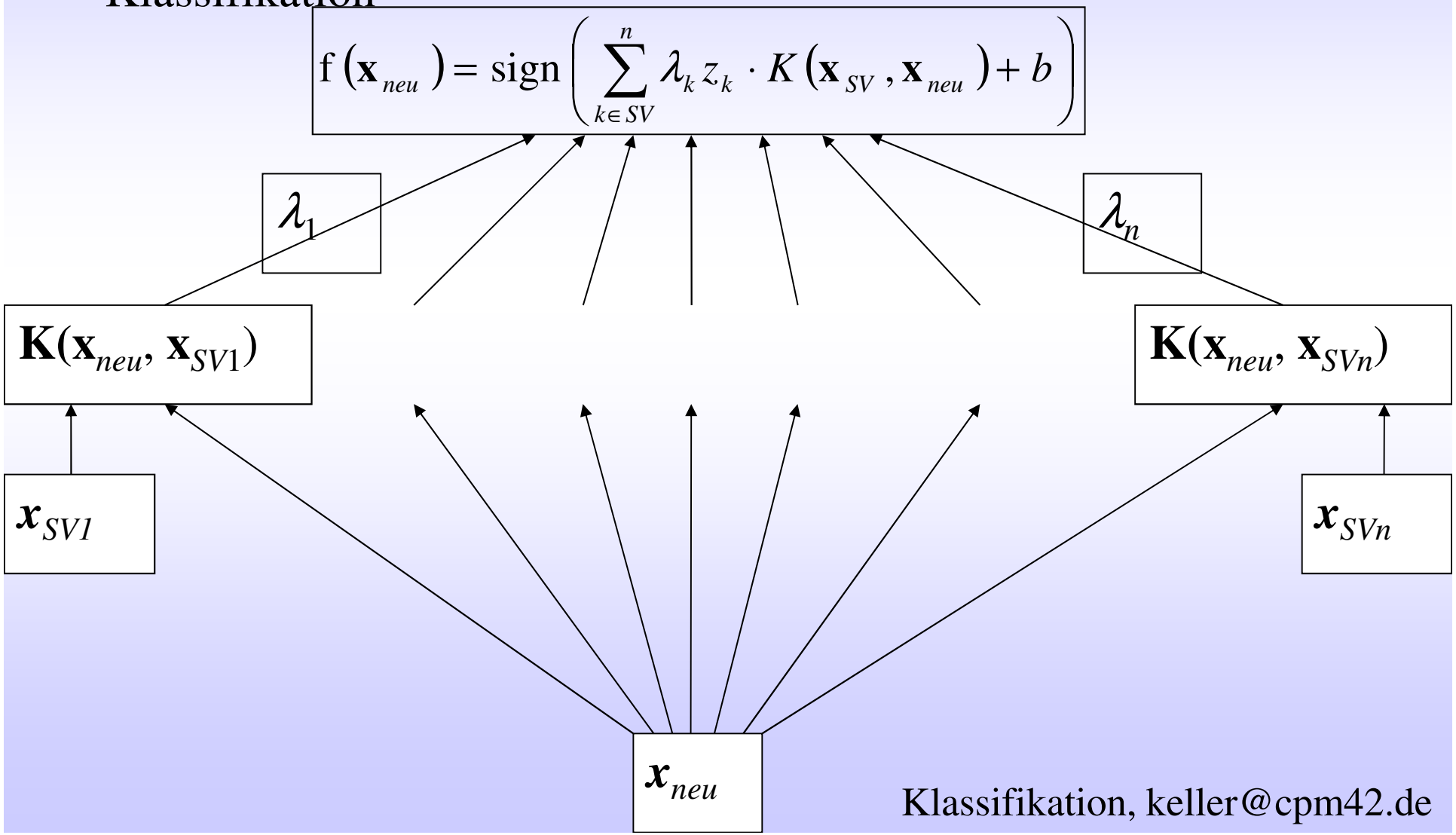


- Klassifikation

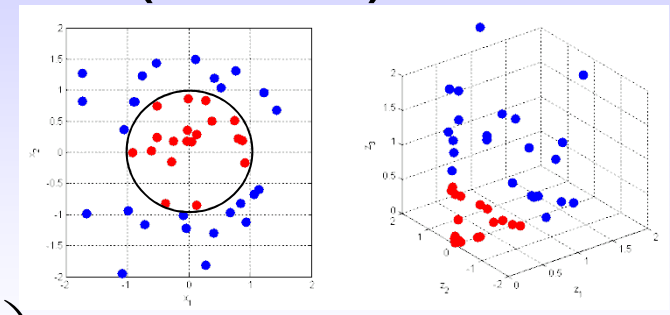
$$f(\mathbf{x}_{neu}) = \text{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot K(\mathbf{x}_{SV}, \mathbf{x}_{neu}) + b \right)$$

Support Vector Machines (SVM)

- Klassifikation



Support Vector Machines (SVM)



- Klassifikation

$$f(\varphi_{neu}) = \text{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot \langle \varphi_k, \varphi_{neu} \rangle + b \right)$$

$$f(\mathbf{x}_{neu}) = \text{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot K(\mathbf{x}_k, \mathbf{x}_{neu}) + b \right)$$

- Alternative kernels

  - Polynom $K(\mathbf{x}, \mathbf{y}) := (\mathbf{x} \cdot \mathbf{y} + c)^d$
  - Sigmoide $K(\mathbf{x}, \mathbf{y}) := \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta)$ MLP
  - RBF $K(\mathbf{x}, \mathbf{y}) := \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \right)$ RBF-Netz

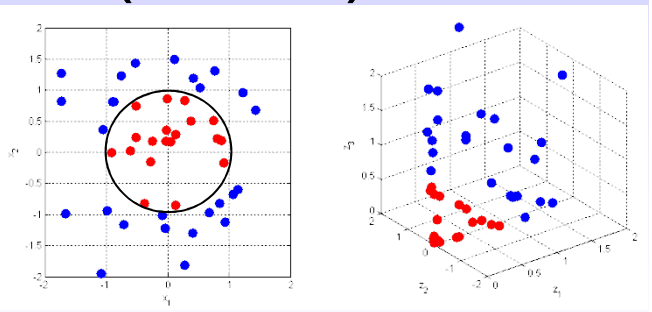
=> Wo steckt die Dimensionaliät?

=> Reihenentwicklung

Support Vector Machines (SVM)

- Klassifikation

$$f(\mathbf{x}_{neu}) = \operatorname{sign}\left(\sum_{k \in SV}^{n} \lambda_k z_k \cdot K(\mathbf{x}_k, \mathbf{x}_{neu}) + b \right)$$



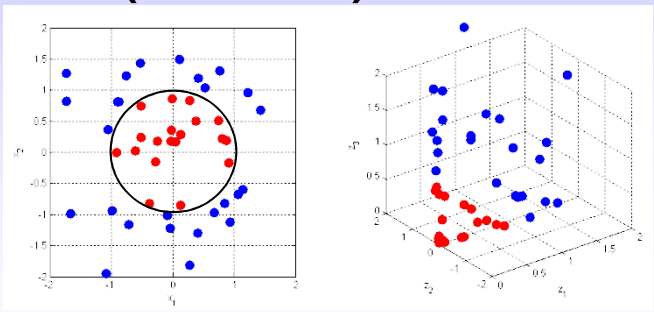
$$K(\mathbf{x}, \mathbf{y}) := \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \right)$$

$$\exp^{\mathbf{x}} = \sum_{n=0}^{\infty} \frac{\mathbf{x}}{n!} \quad \Leftarrow \quad \varphi(\mathbf{x}) = \begin{pmatrix} 1 \\ \mathbf{x} \\ \mathbf{x}^2/2! \\ \mathbf{x}^3/3! \\ \mathbf{x}^4/4! \\ \dots \end{pmatrix}$$

Support Vector Machines (SVM)

- Hauptachsen-Transformation (PCA)

$$\mathbf{K} = \mathrm{E}\left\{\left\langle (\mathbf{x} - \bar{\mathbf{x}}), (\mathbf{x} - \bar{\mathbf{x}}) \right\rangle\right\}$$



- Kernel-based PCA

$$\mathbf{K} = \frac{1}{n} \sum_{k=1}^{n} \mathrm{K}(\mathbf{x}_k, \mathbf{x}_k)$$

- Einfache Herleitung der transformierten Eigenvektoren

Multilayer Perceptron (MLP)

- Lineare Hyperebenen sehr leistungsfähig
- Nichtlineare Transformationen: alles wird separierbar
- Problem: Geeignete Nichtlinearität

- Kenntnis Merkmalsverteilung
- Polynome: wenige Merkmale für viele Parameter

- Neuronale Netze:
  - Training der *Nichtlinearität*
  - Training der *linearen Diskriminante*

- Neuronale Netze primär:
  - Abbildung von Eingangsräumen auf *beliebige* Ausgangsräume

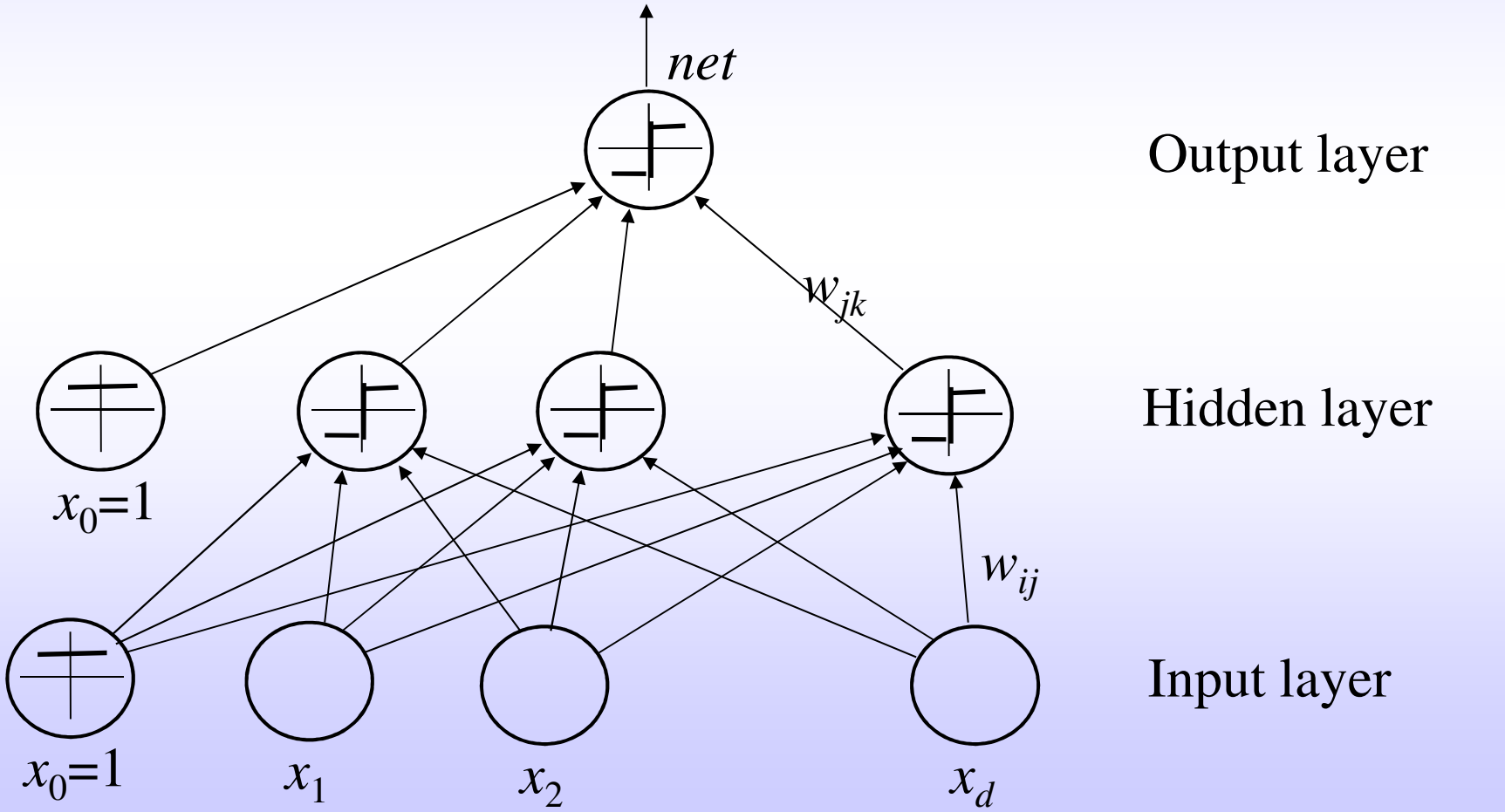
Multilayer Perceptron (MLP)

- Unüberwachtes Lernen: SOM
- Überwachtes Lernen: RBF, Multilyaer Perceptron (MLP)

- Bisher: 2 Schichten
- Nun: mind. 3 Schichten

Multilayer Perceptron (MLP)

- Aufbau Multilyaer Perceptron (MLP)



Multilayer Perceptron (MLP)

- Feedforward Operation

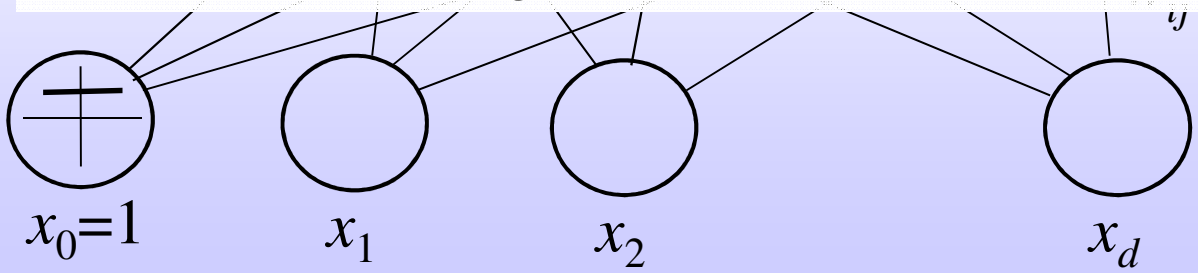
$$z_k = \mathrm{f}(net_k)$$ Nichtlineare Abbildung

$$net_k = \sum_{j=0}^{n_H} w_{jk} \cdot y_j = \mathbf{w}_k^T \cdot \mathbf{y}$$ Output layer

$$y_i = \mathrm{f}(net_j)$$ Nichtlineare Abbildung

$$g_j(\mathbf{x}) = net_j = \sum_{i=0}^{d} w_{ij} \cdot x_i = \mathbf{w}_j^T \cdot \mathbf{x}$$ Hidden layer

Aktivierungsfunktion



Input layer

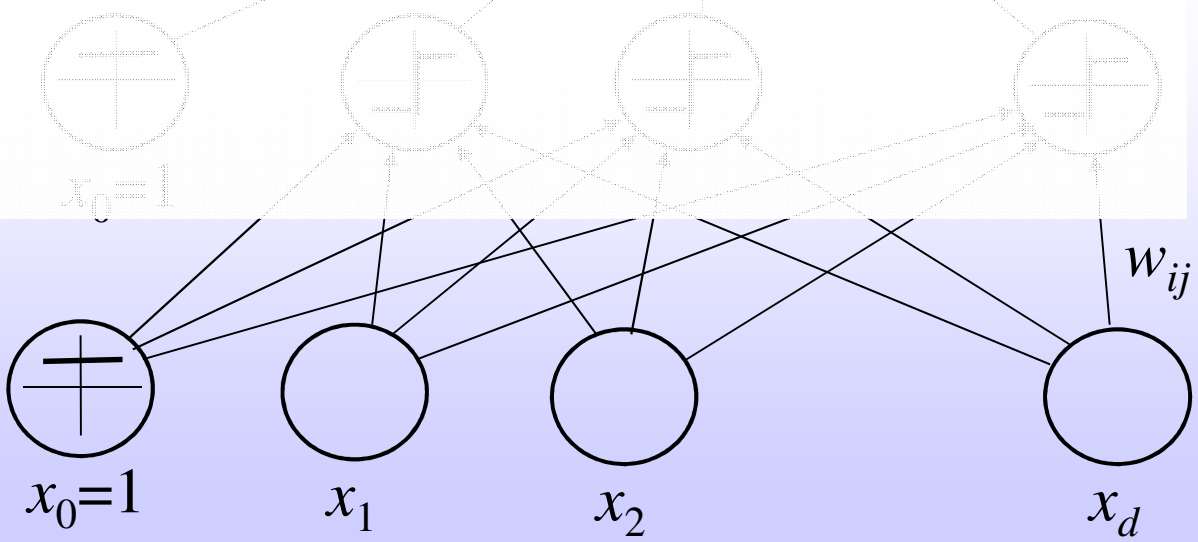
Multilayer Perceptron (MLP)

- Feedforward Operation:

$$z_k = \mathrm{f}(net_k)$$

$$g_k(\mathbf{x}) = z_k = \mathrm{f}\left(\sum_{j=1}^{n_H} w_{jk} \cdot \mathrm{f}\left(\sum_{i=1}^{d} w_{ij} \cdot x_i + w_{j0}\right) + w_{k0}\right)$$

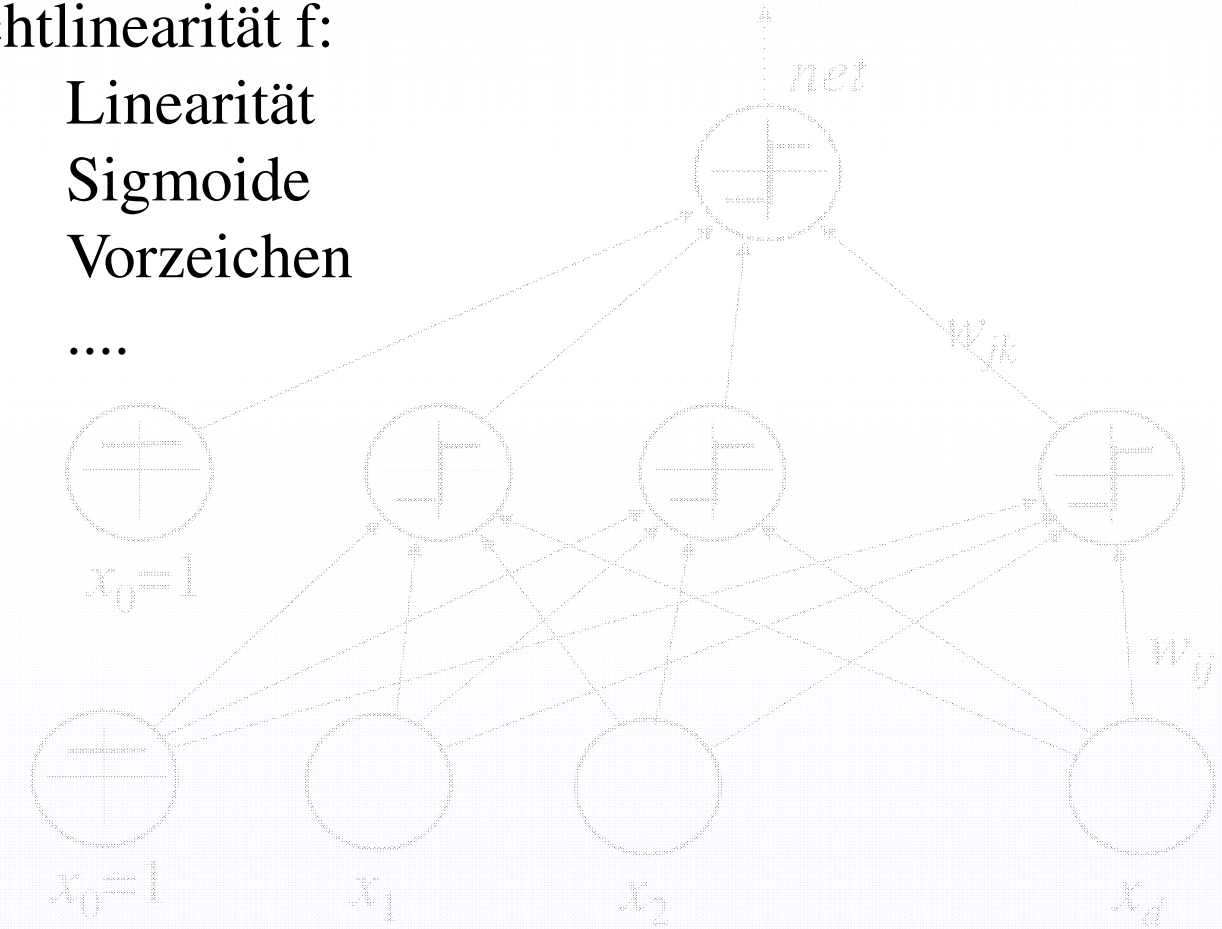
-Klassifikation: c Ausgabeneuronen k=1, ..., c



Multilayer Perceptron (MLP)

- Feedforward Operation:

- Nichtlinearität f:
  - Linearität
  - Sigmoide
  - Vorzeichen
  - ....



Multilayer Perceptron (MLP)

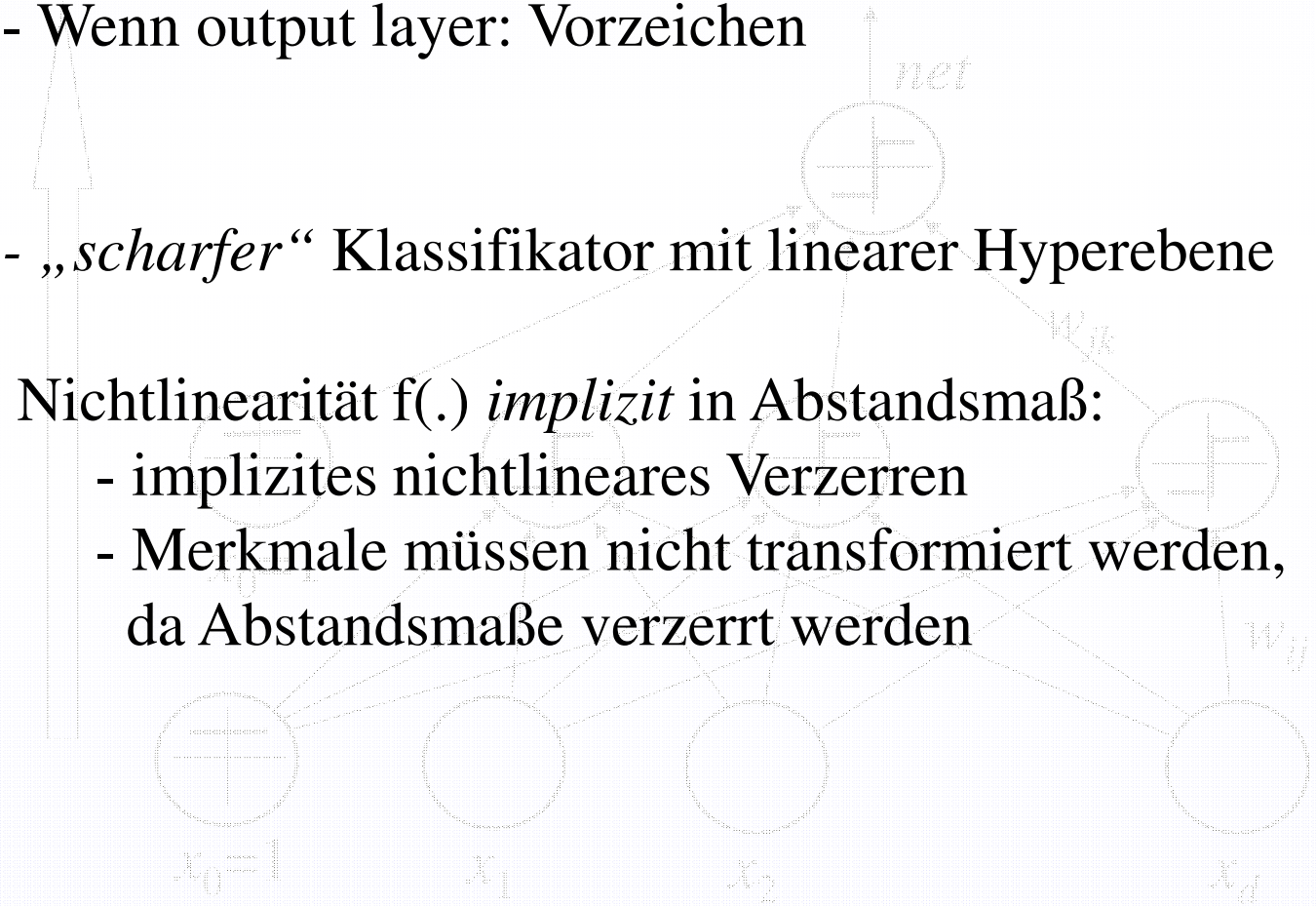
- Feedforward Operation:

- Wenn output layer: Vorzeichen

- *„scharfer“* Klassifikator mit linearer Hyperebene

Nichtlinearität f(.) *implizit* in Abstandsmaß:

- implizites nichtlineares Verzerren
- Merkmale müssen nicht transformiert werden, da Abstandsmaße verzerrt werden



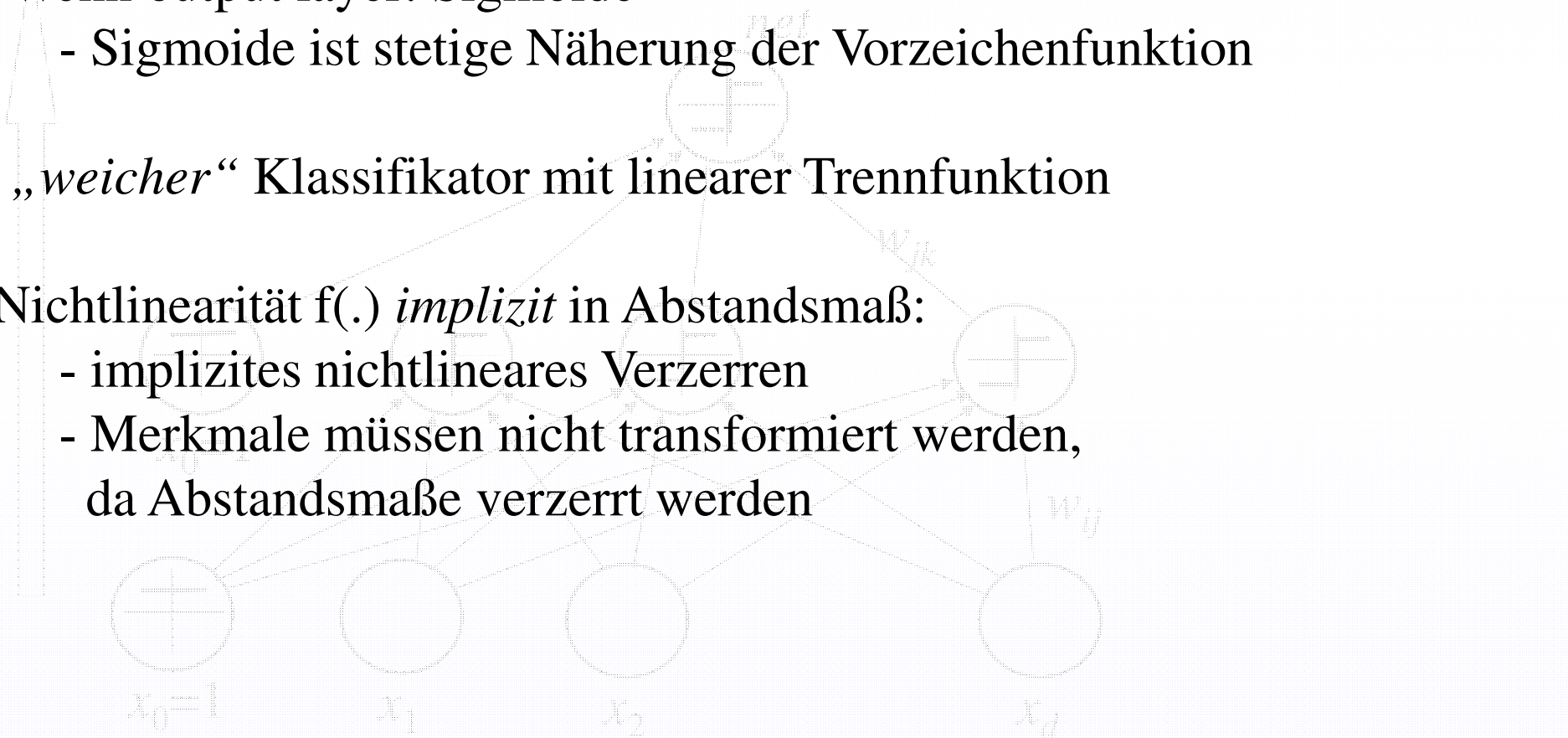
Multilayer Perceptron (MLP)

- Feedforward Operation:

- Wenn output layer: Sigmoide
  - Sigmoide ist stetige Näherung der Vorzeichenfunktion

- *„weicher“* Klassifikator mit linearer Trennfunktion

Nichtlinearität f(.) *implizit* in Abstandsmaß:
- implizites nichtlineares Verzerren
- Merkmale müssen nicht transformiert werden, da Abstandsmaße verzerrt werden



Multilayer Perceptron (MLP)

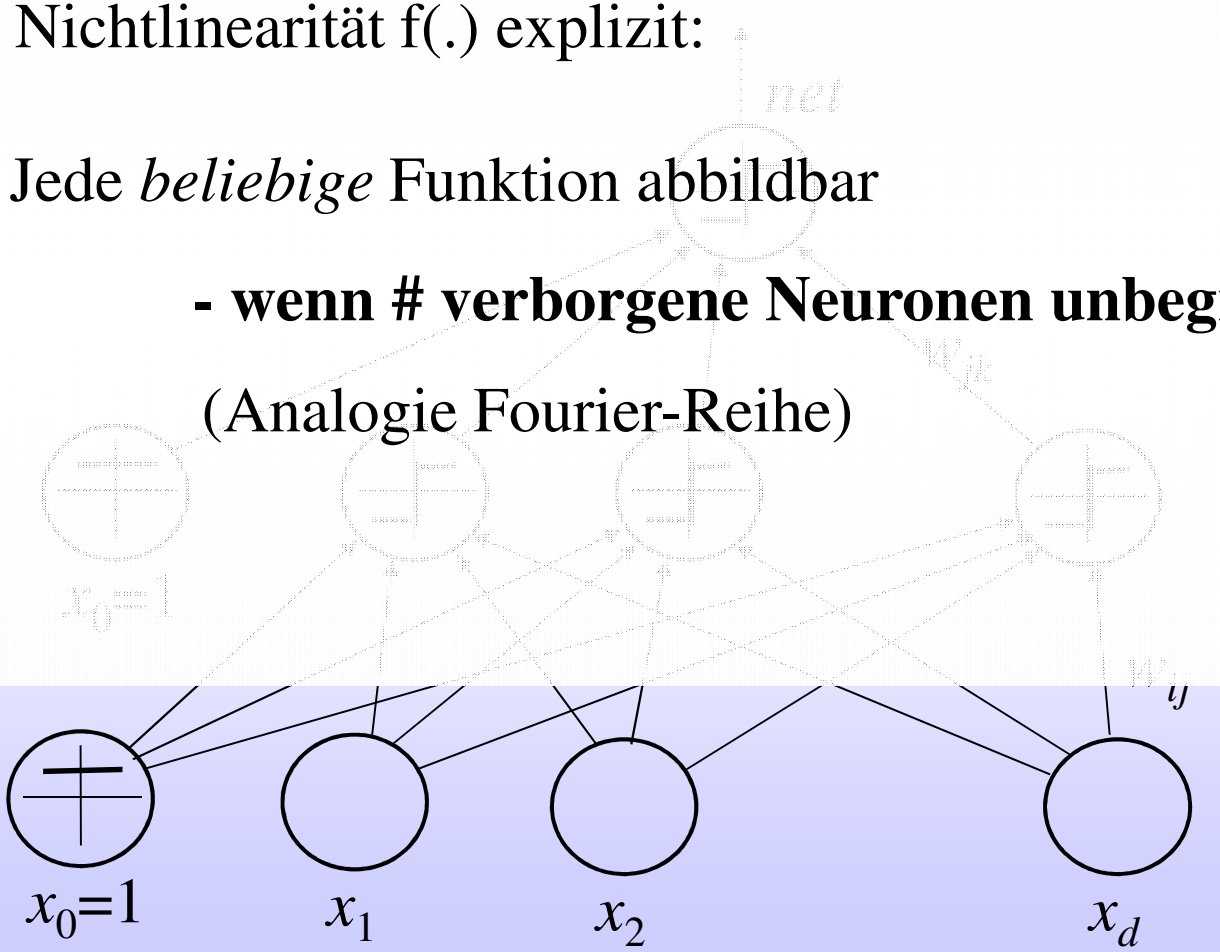
- Feedforward Operation:

- Nichtlinearität f(.) explizit:

Jede *beliebige* Funktion abbildbar

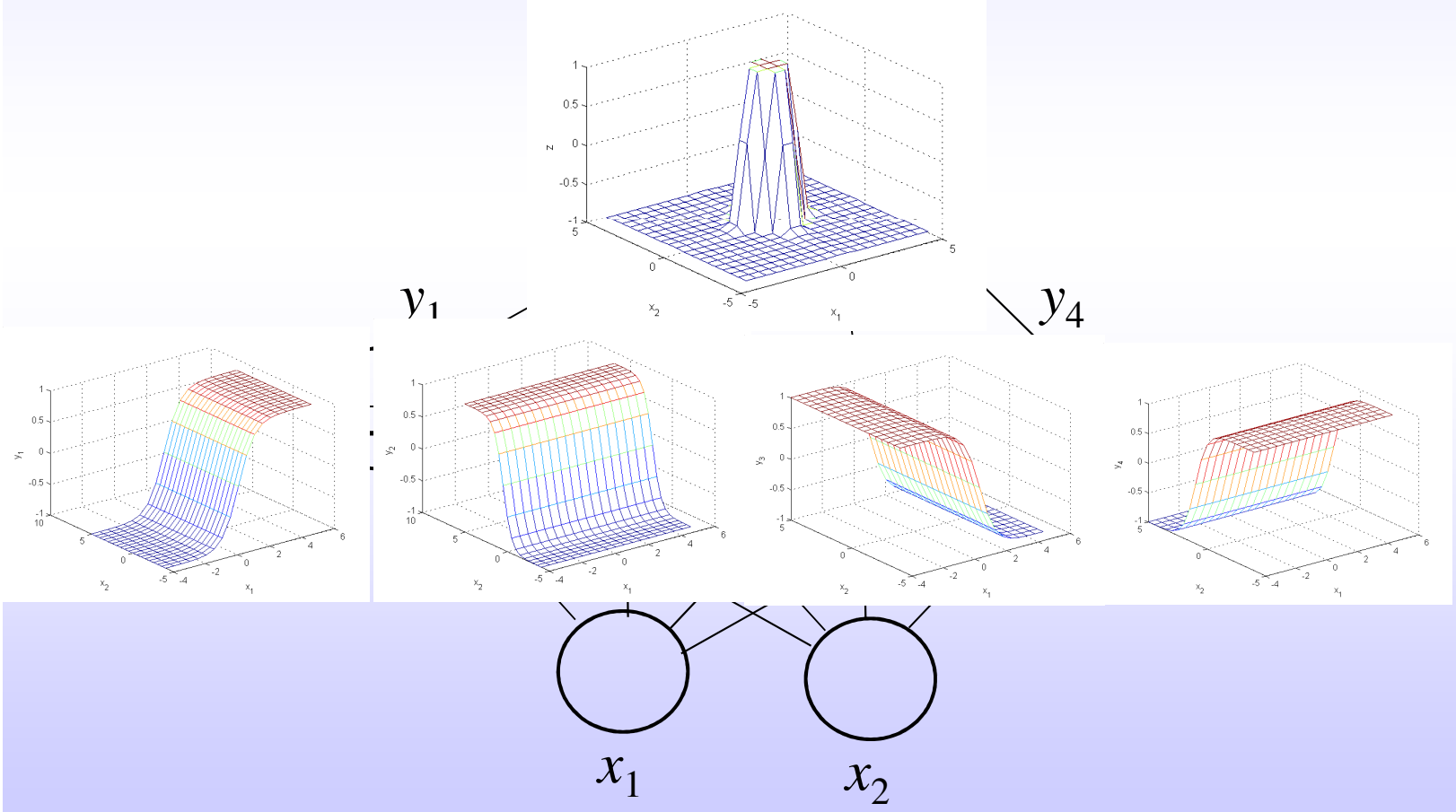
- wenn # verborgene Neuronen unbegrenzt!

(Analogie Fourier-Reihe)



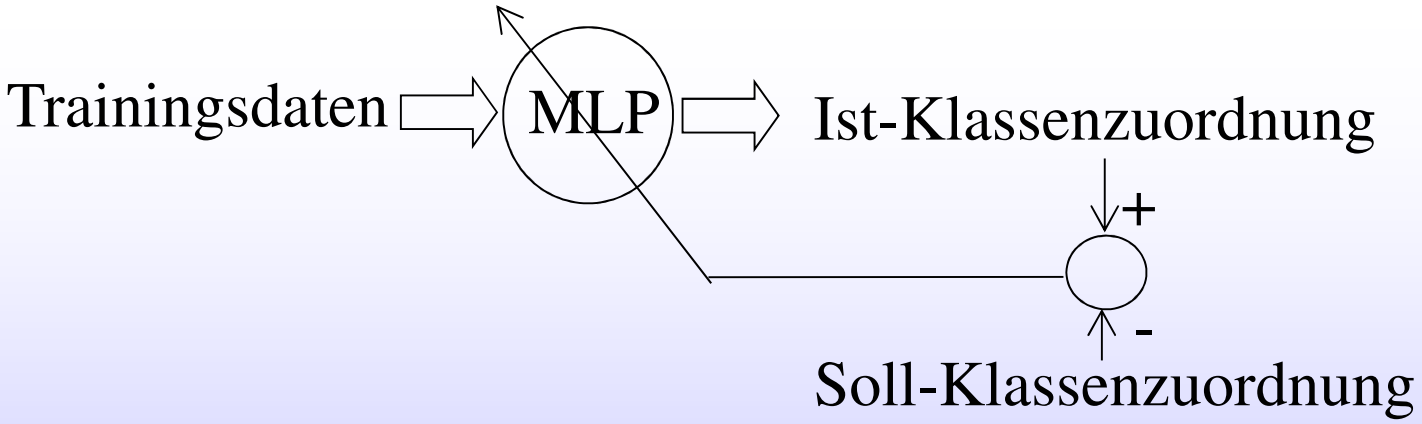
Multilayer Perceptron (MLP)

- Feedforward Operation:
  - Beispiel: Nichtlinearität f(.) *explizit* dargestellt:



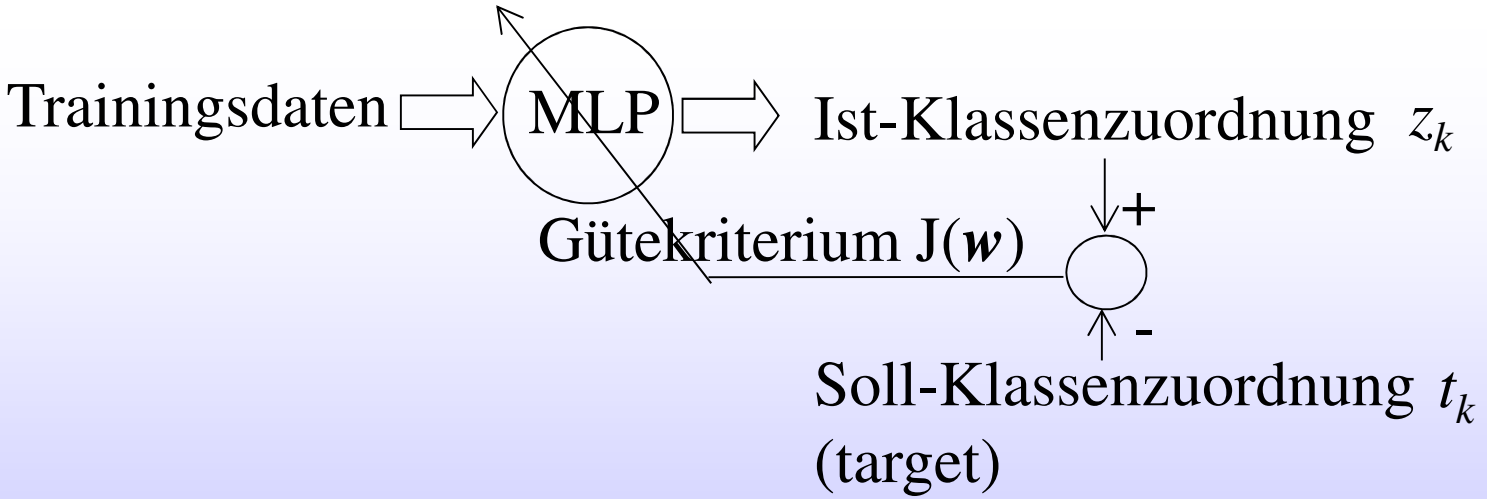
Multilayer Perceptron (MLP)

- Training:
  - Backpropagation (des Fehlers)
  - div. Optimierungsverf., meist Gradientenverfahren LMS



Multilayer Perceptron (MLP)

- Training:
  - Backpropagation (des Fehlers)
  - div. Optimierungsverf., meist Gradientenverfahren LMS



Multilayer Perceptron (MLP)

- Training:

  - Backpropagation (des Fehlers)

Gütekriterium, Trainingsfehler

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|\mathbf{t} - \mathbf{z}\|^2$$

Gradientenverfahren

$$\nabla J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$