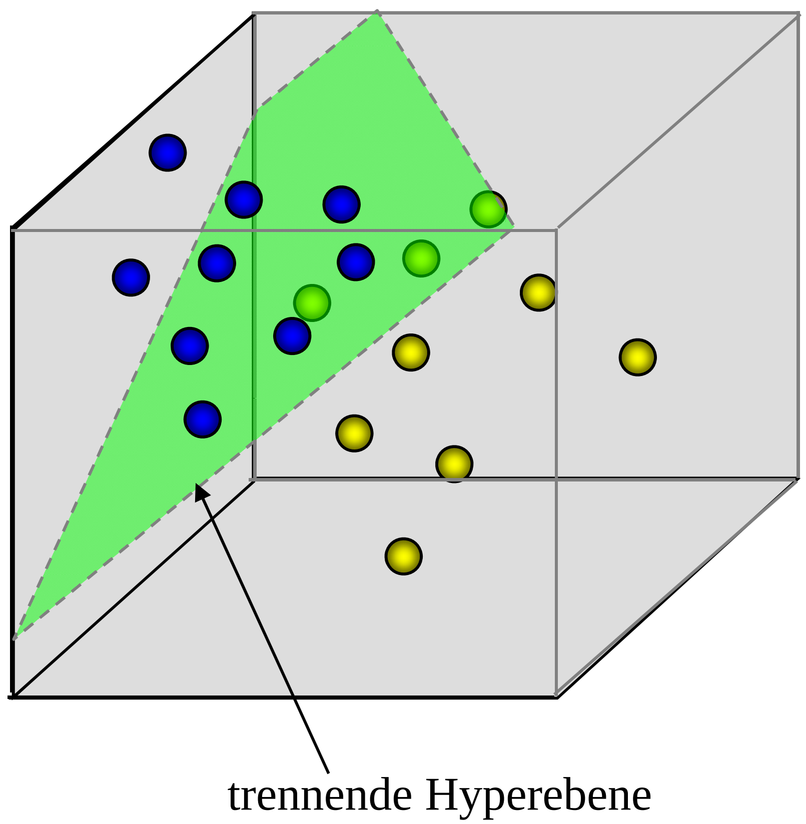
Motivation: Lineare Separation



- Vektoren in R^d repräsentieren Objekte.
- Objekte gehören zu genau einer von je 2 Klassen

Klassifikation durch lineare Separation:

- Suche Hyperebene, die beide Klassen „maximal stabil“ voneinander trennt.

- Ordne unbekannte Elemente der Seite der Ebene zu, auf der sie sich befinden.

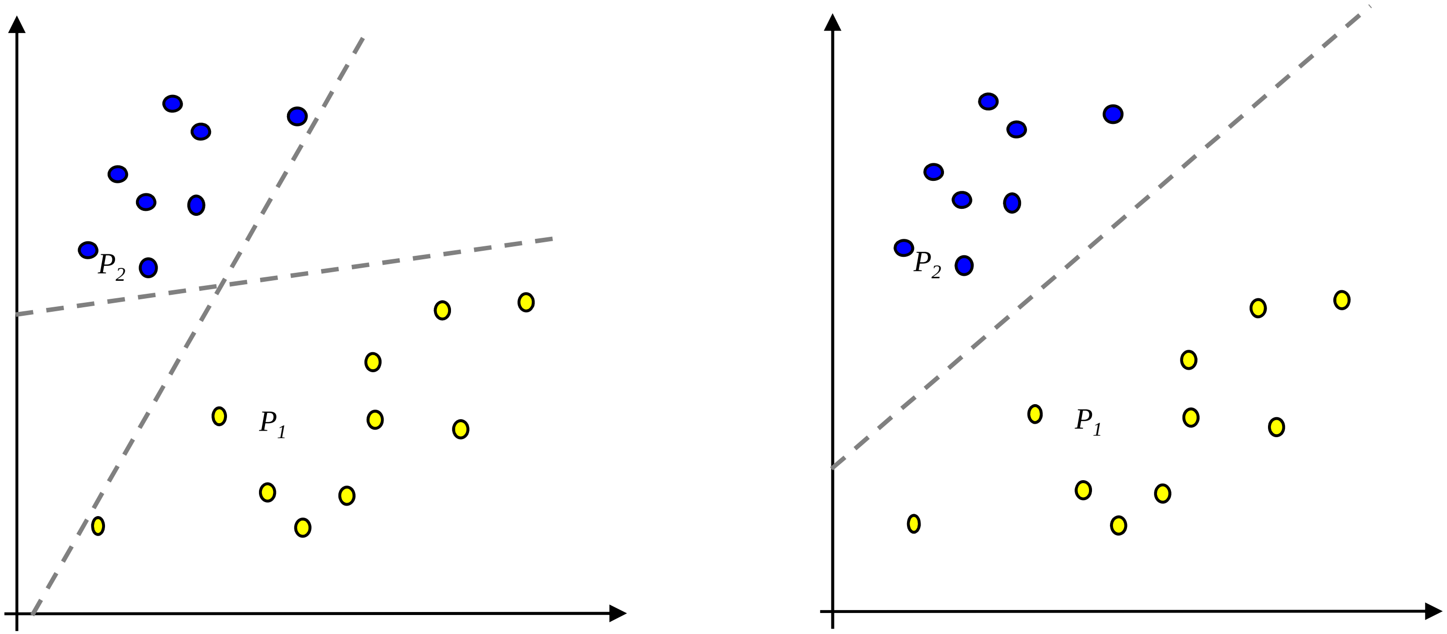
Probleme bei linearer Separation:

- Was ist die „maximal stabile" Hyperebene und wie berechnet man sie effizient?
- Klassen nicht immer linear trennbar.
- Berechnung von Hyperebenen nach Auswahl sehr aufwendig.
- Einschränkung auf 2 Klassen.
- ...

⇨ *Lösungen dieser Probleme mit Support Vector Machines(SVMs) [Vapnik 1979 u. 1995].*

Maximum Margin Hyperplane

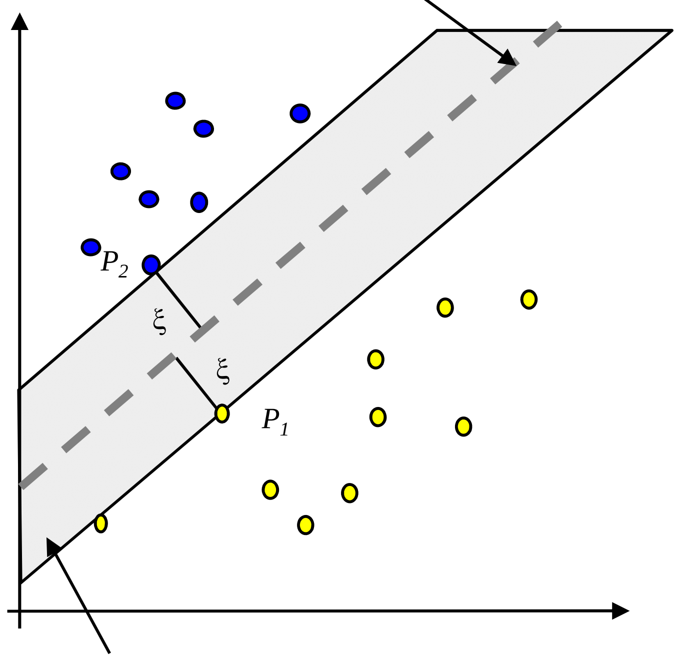
Problem: Hyperebene die P_1 und P_2 trennt ist nicht eindeutig.
$\Rightarrow$ *Welche Hyperebene ist für die Separation die Beste ?*



Kriterien:

- Stabilität beim Einfügen
- Abstand zu den Objekten beider Klassen

Lineare Separation mit der *„Maximum Margin Hyperplane"*



- Abstand zu Punkten aus beiden Mengen ist maximal, d.h. mind. ξ.
- Wahrscheinlichkeit, dass beim Einfügen die trennende Hyperebene verschoben werden muss, ist minimal.
- generalisiert am besten.

⇒ Maximum Margin Hyperplane (MMH) ist „maximal stabil"

MMH ist nur von Punkten P_i abhängig, die Abstand ξ zur Ebene aufweisen.

⇒ *P_i heißt Support Vector*

Maximum Margin Hyperplane

Zusammenfassung der Schreibweisen der benötigten algebraischen Konstrukte für Featurespace *FS*:

Skalarprodukt zweier Vektoren: $\left\langle \vec{x}, \vec{y} \right\rangle, \vec{x}, \vec{y} \in FS$

z.B. $\left\langle \vec{x}, \vec{y} \right\rangle = \sum_{i=1}^{d} \left(x_i \cdot y_i \right)$ kanonisches Skalarprodukt

Beschreibung einer Hyperebene: $H\left(\vec{w}, b\right) = \left\{ \vec{x} \in FS \;\middle|\; 0 = \left\langle \vec{w}, \vec{x} \right\rangle + b \right\}$

Abstand eines Vectors zur Ebene: $dist\left(\vec{x}, H(\vec{w}, b)\right) = \left| \frac{1}{\sqrt{\left\langle \vec{w}, \vec{w} \right\rangle}} \left\langle \vec{w}, \vec{x} \right\rangle + b \right|$

Berechnung der Maximum Margin Hyperplane

1. Bedingung: kein Klassifikationsfehler (Klasse1: y_i=1,Klasse 2:y_i=-1)

$$\left.\begin{array}{r}(y_i = -1) \Rightarrow [\langle \vec{w}, \vec{x}_i \rangle + b] < 0 \\ (y_i = 1) \Rightarrow [\langle \vec{w}, \vec{x}_i \rangle + b] > 0\end{array}\right\} \Leftrightarrow y_i \left[\langle \vec{w}, \vec{x}_i \rangle + b\right] > 0$$

2. Bedingung: Maximaler Rand (Margin)

maximiere: $\xi = \min_{\vec{x}_i \in TR} \left| \frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \right|$ (Abstand von x_i zur Ebene $H(\vec{w},b)$)

oder

maximiere: ξ , so dass $\left[y_i \left(\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} \left[\langle \vec{w}, \vec{x}_i \rangle + b \right] \right) \geq \xi \right]$ für ∀i ∈ [1..n]

Maximum Margin Hyperplane

maximiere ξ in $\left[y_i \left(\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} \left[\langle \vec{w}, \vec{x}_i \rangle + b \right] \right) \geq \xi \right]$; für ∀i ∈ [1..n]

Setze $\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} = \xi$: max. $\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}}$, mit $\left(y_i \cdot \left(\xi \cdot \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \right) \geq \xi \right)$ ∀i ∈ [1..n]

⇒ max. $\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}}$, mit $\left(y_i \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \geq 1 \right)$ ∀i ∈ [1..n]

Statt $\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}}$ invertiere, quadriere und minimiere das Ergebnis:

Primäres OP: minimiere $J(\vec{w}, b) = \langle \vec{w}, \vec{w} \rangle$

unter Nebenbedingung für ∀i ∈ [1..n] sei $\left(y_i \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \geq 1 \right)$

Maximum Margin Hyperplane

Zur Berechnung wird das primäre Optimierungsproblem in ein duales OP überführt.
(Umformulierung in Form mit Langrange Multiplikatoren nach Karush-Kuhn-Tucker)

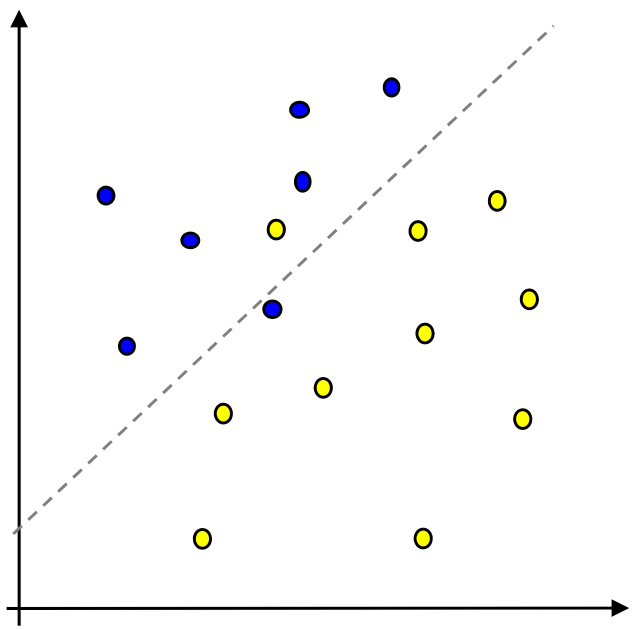
Duales OP: maximiere $L(\vec{\alpha}) = \left(\sum_{i=1}^{n} \alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \left\langle \vec{x}_i, \vec{x}_j \right\rangle$

unter Bedingung $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$, $0 \le \alpha_i$ und $\vec{\alpha} \in \Re^n$

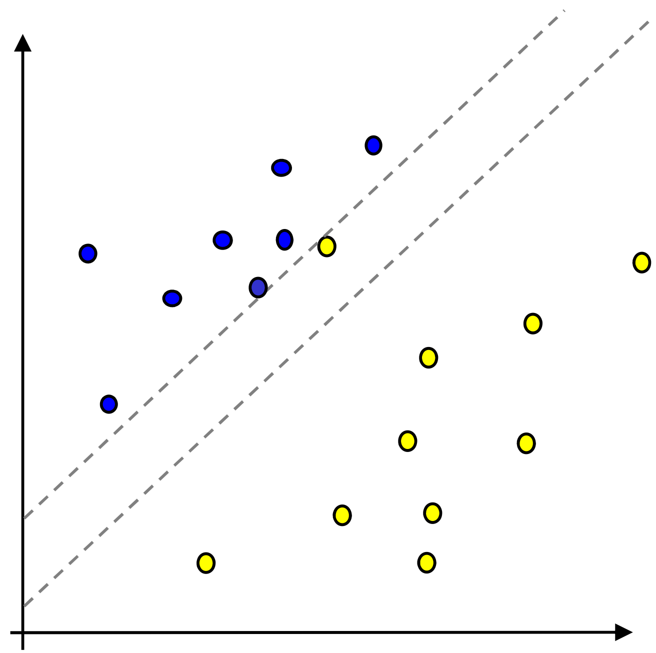
⇒ Lösung des Problems mit Algorithmen aus der Optimierungstheorie
⇒ bis jetzt nur linear separierbarer Fall: Soft Margin Optimierung
⇒ Einführung von Kernelfunktionen zur Steigerung der Kapazität

Soft Margin

Behandlung nicht linear trennbarer Daten: *Soft Margin Optimierung*



Daten nicht separierbar

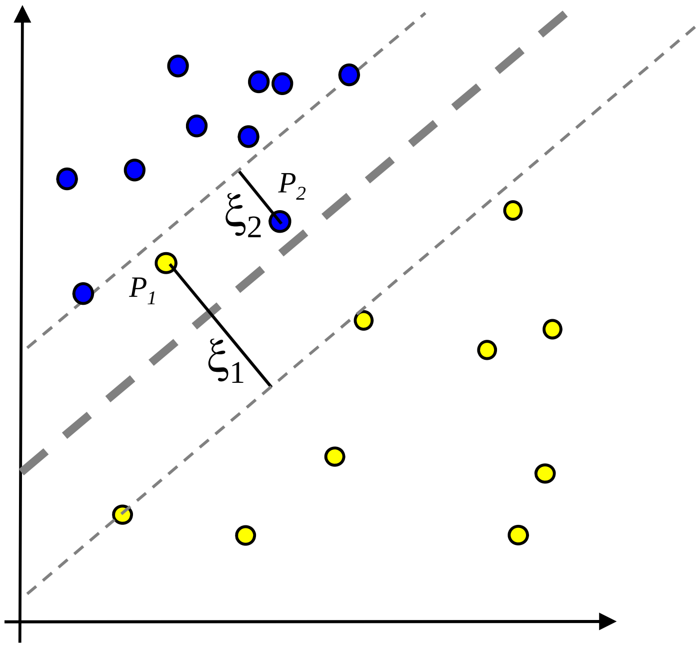


vollständige Separation ist nicht optimal

⇒ Trade-Off zwischen Trainingsfehler und Breite des Randes

Soft Margin

Betrachte beim Optimieren zusätzlich noch die Anzahl der Trainingsfehler.



- *ξ_i ist der Abstand von P_i zum Rand (wird auch Slack-Variable genannt)*
- *C reguliert den Einfluss eines einzelnen Trainingsvektors*

Primäres OP : minimiere $J(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\langle \vec{w}, \vec{w} \rangle + C \cdot \sum_{i=1}^{n} \xi_i$

unter Nebenbedingung für ∀i ∈ [1..n] sei $y_i\left(\langle \vec{w}, \vec{x_i} \rangle + b\right) \geq 1 - \xi_i$ und $\xi_i \geq 0$

$\Rightarrow$ Primäres Optimierungsproblem unter weichen Grenzen (Soft Margin)

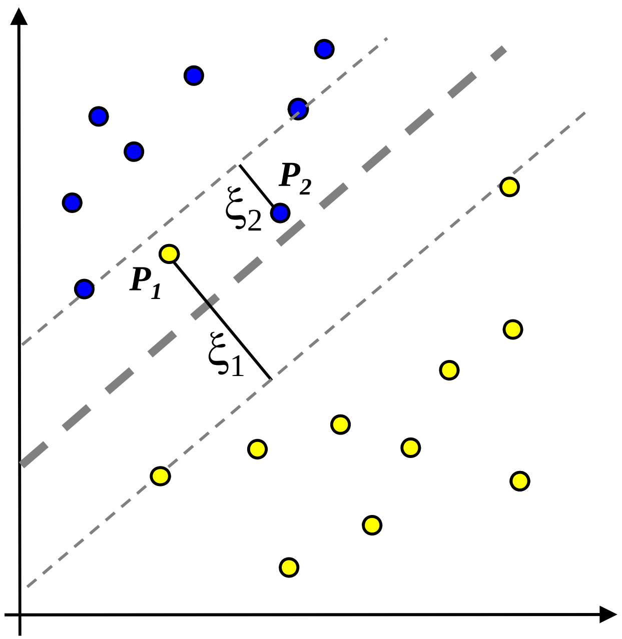
Soft Margin

Das duale OP mit Lagrange Multiplikatoren verändert sich wie folgt:

Duales OP: maximiere $L(\vec{\alpha}) = \left(\sum_{i=1}^{n} \alpha_i \right) - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \left\langle \vec{x}_i, \vec{x}_j \right\rangle$

mit Bedingung $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$ und $0 \leq \alpha_i \leq \boldsymbol{C}$

- $0 < \alpha_i < C \Leftrightarrow$ Support Vektor mit $\xi_i = 0$
- $\alpha_i = C \Leftrightarrow$ Support Vektor mit $\xi_i > 0$
- $\alpha_i = 0$ *sonst*



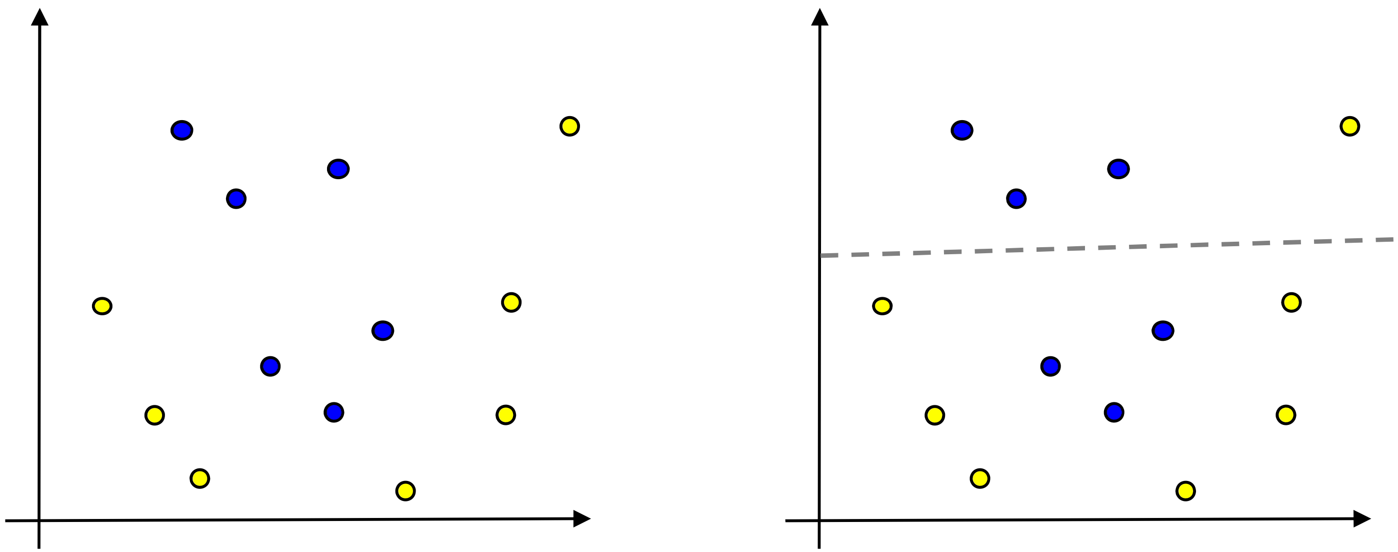
Entscheidungsregel:

$$h(\vec{x}) = sign\left(\sum_{x_i \in SV} \alpha_i \cdot y_i \left\langle \vec{x}_i, \vec{x} \right\rangle + b \right)$$

Lernen bei nicht linear trennbaren Datenmengen

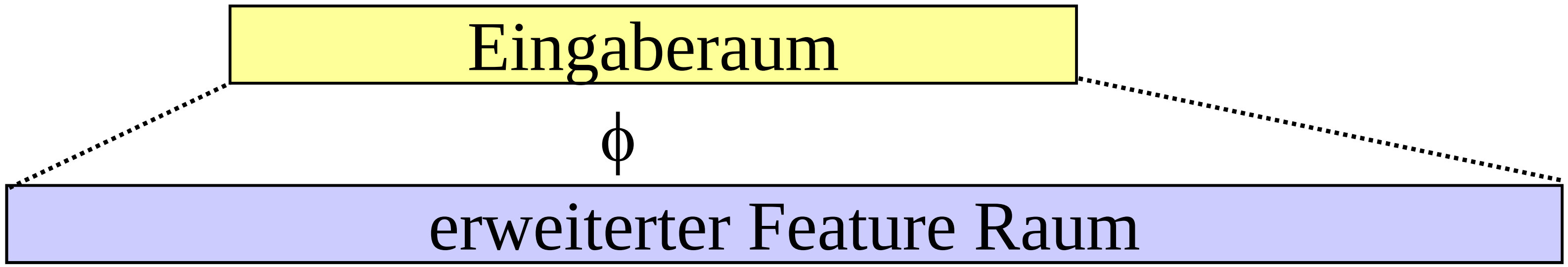
Problem: Bei realen Problemen ist häufig keine lineare Separation mit hoher Klassifikationsgenauigkeit mehr möglich.

Idee: Transformiere Daten in einen nicht linearen Feature-Raum und versuche sie im neuen Raum linear zu separieren.
(Erweiterung des Hypothesenraumes)



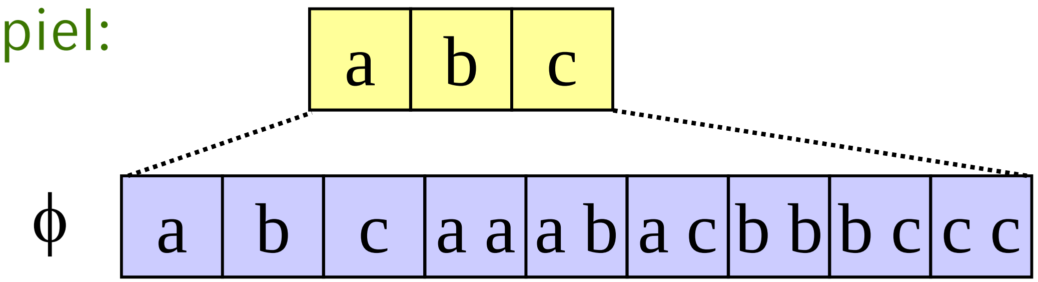
Beispiel: quadratische Transformation

Erweiterung der Hypothesenraumes



⇒ Versuche jetzt in erweitertem Feature Raum linear zu separieren

Beispiel:

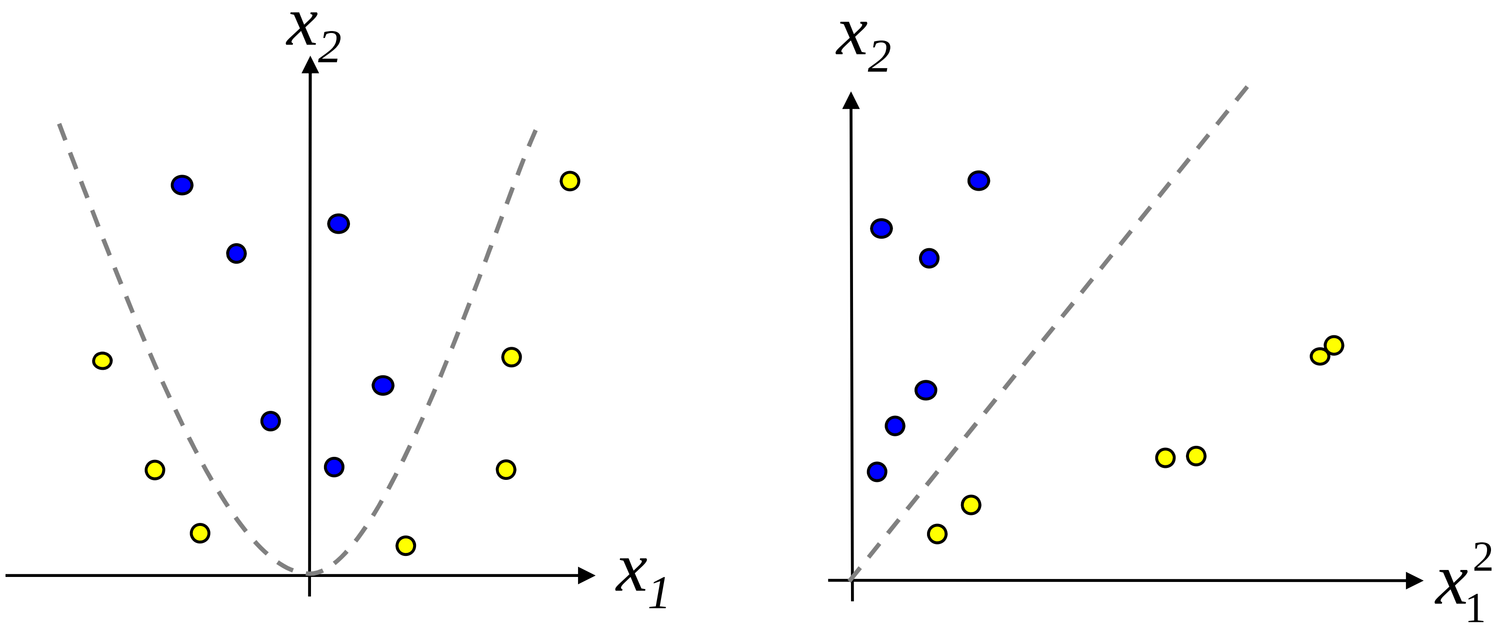


hier: Eine Hyperebene im erweiterten Feature Raum ist ein Polynom 2. Grades im Eingaberaum.

Kernel Machines

Eingaberaum: $\vec{x} = (x_1, x_2)$ (2 Attribute)

im erweiterten Raum: $\phi(\vec{x}) = \left(x_1^2, x_2^2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2, \sqrt{2} \cdot x_1 \cdot x_2, 1\right)$
(6 Attribute)



Einführung eines Kernels ⇔ (Implizite) Featuretransformation mittels $\phi(\vec{x}): FS_{alt} \longrightarrow FS_{neu}$

Duales OP: maximiere

$$L(\vec{\alpha}) = \left(\sum_{i=1}^{n} \alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \left\langle \phi(\vec{x}_i), \phi(\vec{x}_j) \right\rangle$$

mit Bedingung $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$ und $0 \le \alpha_i \le C$

Zusätzliche Featuretransformation wirkt sich nur auf das Skalarprodukt der Trainingsvektoren aus. ⇒ Kernel K ist eine Funktion mit:

$$K_\phi\left(\vec{x}_i, \vec{x}_j\right) = \left\langle \phi(\vec{x}_i), \phi(\vec{x}_j) \right\rangle$$

Wann ist eine Funktion K(x,y) ein Kernel ?

Wenn die **Kernel-Matrix** (Gram Matrix) KM

$$KM(K) = \begin{pmatrix} K(\vec{x}_1, \vec{x}_1) & .. & K(\vec{x}_1, \vec{x}_n) \\ .. & .. & .. \\ K(\vec{x}_n, \vec{x}_1) & .. & K(\vec{x}_n, \vec{x}_n) \end{pmatrix}$$

positiv (semi) definit ist, also keine negativen
Eigenwerte besitzt, dann ist *K(x,y)* ein Kernel (siehe Mercer's Theorem)

Notwendige Bedingungen:

- $K_\phi\left(\vec{x}, \vec{y}\right) = \left\langle \phi(\vec{x}), \phi(\vec{y}) \right\rangle = \left\langle \phi(\vec{y}), \phi(\vec{x}) \right\rangle = K_\phi\left(\vec{y}, \vec{x}\right)$ (Symmetrie)
- $K_\phi\left(\vec{x}, \vec{y}\right)^2 \leq K_\phi\left(\vec{x}, \vec{x}\right) \cdot K_\phi\left(\vec{y}, \vec{y}\right)$ (Cauchy-Schwarz)

Symmetrie und Cauchy-Schwarz sind keine hinreichenden Bedingungen!

Kernel Machines

einige Regeln zur Kombination vom Kerneln:

$$K(\vec{x},\vec{y}) = K_1(\vec{x},\vec{y}) \cdot K_2(\vec{x},\vec{y})$$

$$K(\vec{x},\vec{y}) = K_1(\vec{x},\vec{y}) + K_2(\vec{x},\vec{y})$$

$$K(\vec{x},\vec{y}) = a \cdot K_1(\vec{x},\vec{y})$$

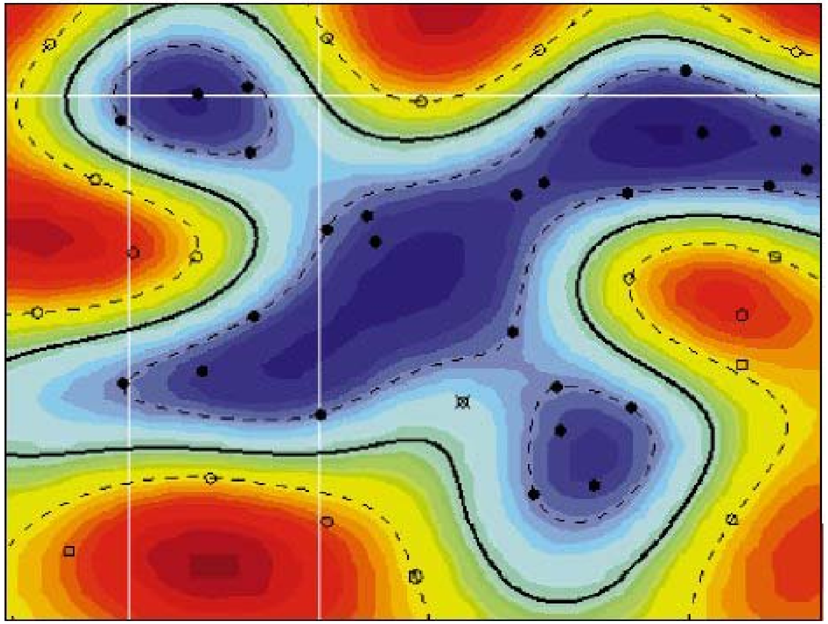
$$K(\vec{x},\vec{y}) = x^T \cdot B \cdot y$$

für K_1, K_2 Kernelfunktionen, a eine positive Konstante und B eine symmetrische positiv semi-definite Matrix.

Beispiele für verwendete Kernel-Funktionen:

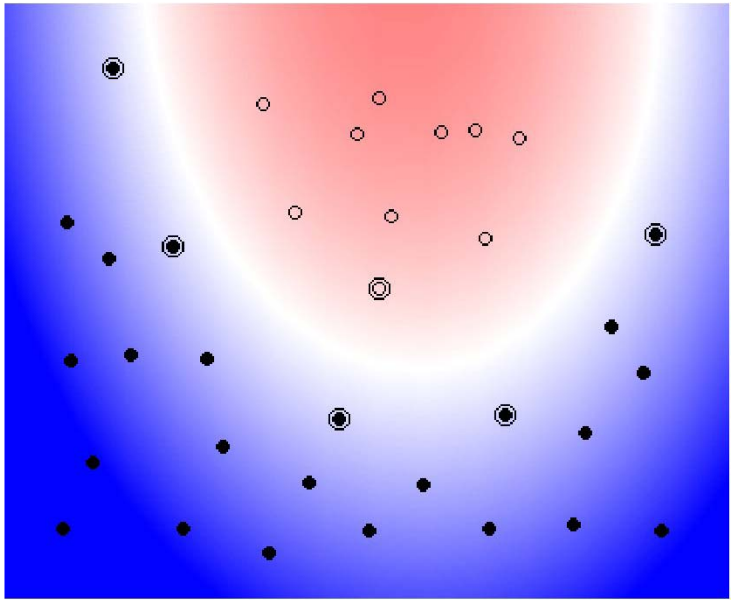
- linear: $K(\vec{x},\vec{y}) = \left\langle \vec{x},\vec{y} \right\rangle$
- polynomiell: $K(\vec{x},\vec{y}) = \left(\left\langle \vec{x},\vec{y} \right\rangle + c \right)^d$
- Radiale Basisfunktionen: $K(\vec{x},\vec{y}) = \exp\left(-\gamma \cdot \left| \vec{x} - \vec{y} \right|^2 \right)$
- Gauss Kernel: $K(\vec{x},\vec{y}) = \exp\left(-\frac{\left\| \vec{x} - \vec{y} \right\|^2}{2\sigma^2} \right)$
- Sigmoid: $K(\vec{x},\vec{y}) = \tanh\left(\gamma \cdot \left\langle \vec{x},\vec{y} \right\rangle + c \right)$

Kernel Machines



Radial Basis Kernel

Polynomieller Kernel (Grad 2)



Training einer SVM

zu lösen ist folgendes Problem:

Duales OP: maximiere

$$L(\vec{\alpha}) = \left(\sum_{i=1}^{n} \alpha_i\right) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\vec{x}_i, \vec{x}_j)$$

mit Bedingung $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$ und $0 \le \alpha_i \le C$

oder

$$\max \begin{bmatrix} 1 \\ .. \\ 1 \end{bmatrix}^T \cdot \begin{bmatrix} \alpha_1 \\ .. \\ \alpha_n \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ .. \\ \alpha_n \end{bmatrix}^T \cdot \begin{bmatrix} y_1 y_1 K(\vec{x}_1, \vec{x}_1) & .. & y_1 y_n K(\vec{x}_1, \vec{x}_n) \\ .. & .. & .. \\ y_n y_1 K(\vec{x}_n, \vec{x}_1) & .. & y_n y_n K(\vec{x}_n, \vec{x}_n) \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ .. \\ \alpha_n \end{bmatrix}$$

mit Bedingung $\sum_{i=1}^{n} \alpha_i \cdot y_i = 0$ und $0 \le \alpha_i \le C$

Training einer SVM

zur Lösung:

- Standardalgorithmen aus der Optimierungstheorie für *konvexe quadratische Probleme*

- für große Trainingsmengen numerische Algorithmen notwendig

- es existieren einige Spezialalgorithmen für *SVM*-Training:
    - Chunking / Decomposition
    - Sequential Minimal Optimisation (*SMO*)

Multi-Class SVMs

Bisher SVMs nur anwendbar auf 2 Klassen Probleme !!

Idee: Kombination mehrere 2-Klassen SVMs zu Klassifikatoren, die beliebig viele Klassen unterscheiden können.

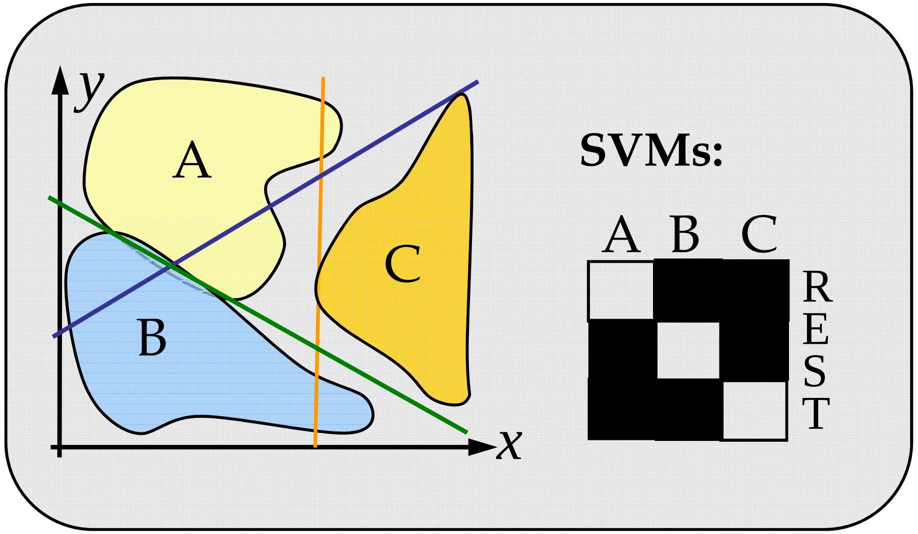
Multi-Class SVMs

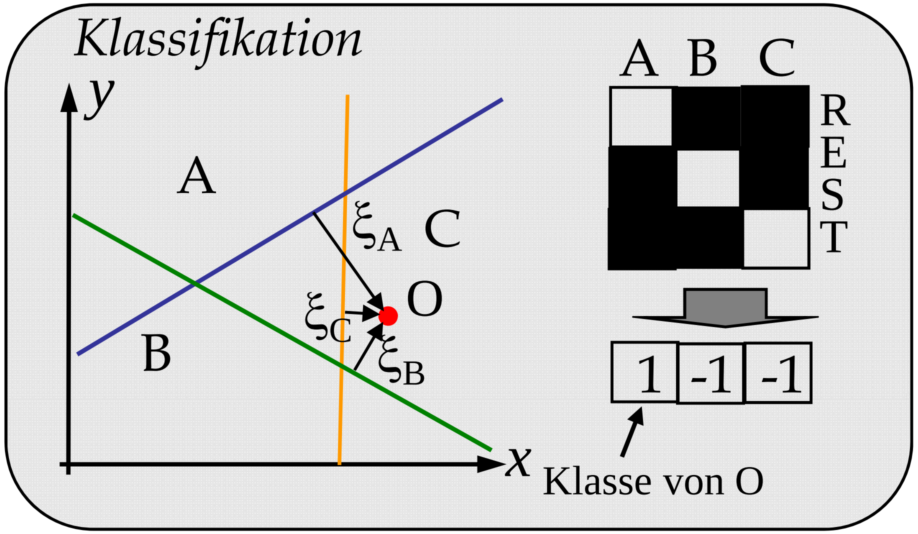
2 klassische Ansätze:

$\Rightarrow$ Unterscheide jede Klasse von allen anderen *(1-versus-Rest)*

$\Rightarrow$ Unterscheide je 2 Klassen *(1-versus-1)*

1-versus-Rest Ansatz





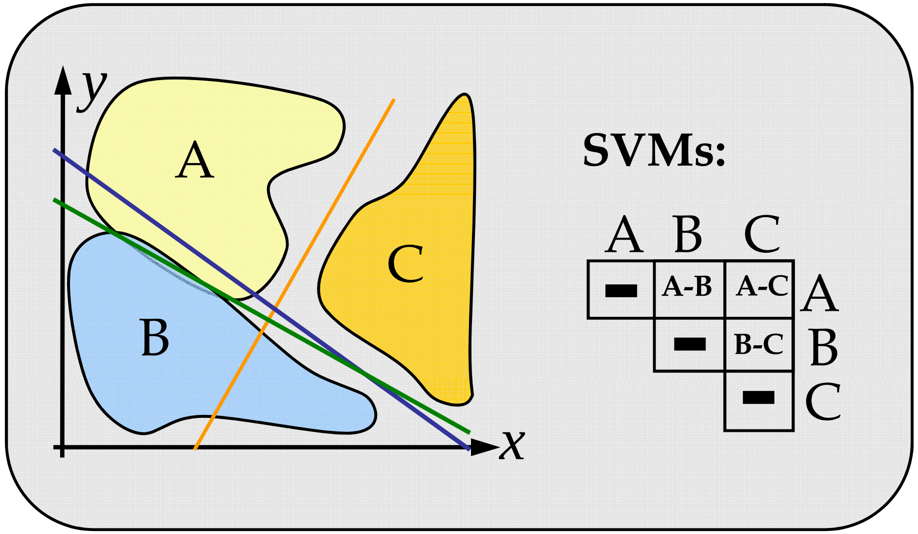
- 1-versus-Rest Ansatz : 1 SVM für jede Klasse.
- SVM trennt jedes Klasse von Vereinigung aller anderen Klassen ab
- Klassifiziere O mit allen Basis-SVMs.

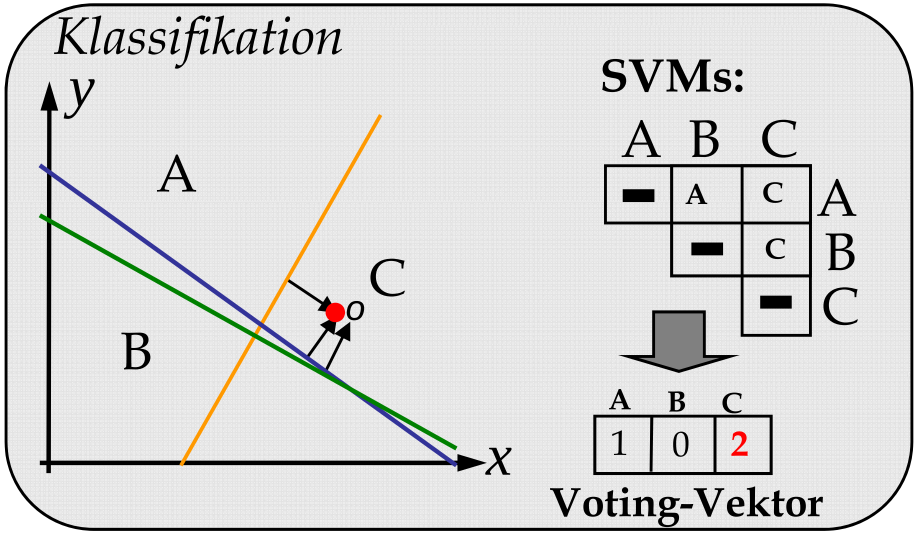
  $\Rightarrow$ **Multiple Klassenzugehörigkeit möglich (Multi-Classification)**

  oder

  Entscheidung für die Klasse,
  bei der Abstand ξ_i am größten ist.

1-versus-1 Ansatz





- 1-versus-1 Ansatz : 1 SVM für jedes Paar von Klassen.
- Klassifikation von Objekt O:
  1. Klassifiziere O mit allen Basis-SVMs.
  2. Zähle "Stimmen"(Votes) für jede Klasse.
- Maximale Anzahl an Votes => Ergebnis.

Vergleich 1-versus-rest und 1-versus-1

Kriterium	1-versus-rest	1-versus-1
Aufwand Training	linear zur Anzahl der Klassen ($O(\|K\|)$)	Quadratisch zur anzahl der Klassen ($O(\|K\|^2)$)
Aufwand Klassifikation	linear zur Anzahl der Klassen ($O(\|K\|)$)	Quadratisch zur Anzahl der Klassen ($O(\|K\|^2)$) **Verbesserung:** „Decision Directed Acyclic Graphs“ Klassifikation in $O(\|K\|)$ [Platt,Christianini 1999]
Genauigkeit	tendenziell schlechter	tendenziell höher, (nutzt wissen über unterschiedliche Klassen besser aus)

Diskussion

+ erzeugt Klassifikatoren mit hoher Genauigkeit
+ verhältnismäßig schwache Tendenz zu Overfitting (Begründung durch Generalisierungtheorie)
+ effiziente Klassifikation neuer Objekte
+ kompakte Modelle

- unter Umständen lange Trainingszeiten
- aufwendige Implementierung
- gefundene Modelle schwer zu deuten

Literatur:

- C. Cortes, V. Vapnik: Support-vector networks.
  Machine Learning, 20:273-297, November 1995.

- C.J.C. Burges: A tutorial on support vector machines for pattern recognition.
  Data Mining and Knowledge Discovery, 2(2):121-167,1998.

- T. Joachims: Text categorisation with Support Vector Machines.
  in Proceedings of European Conference on Machine Learning (ECML), 1998.

- N. Cristianini, J Shawne-Taylor: An Introduction to Support Vector Machines and other kernel-based learning methods.
  Cambridge University Press 2000.