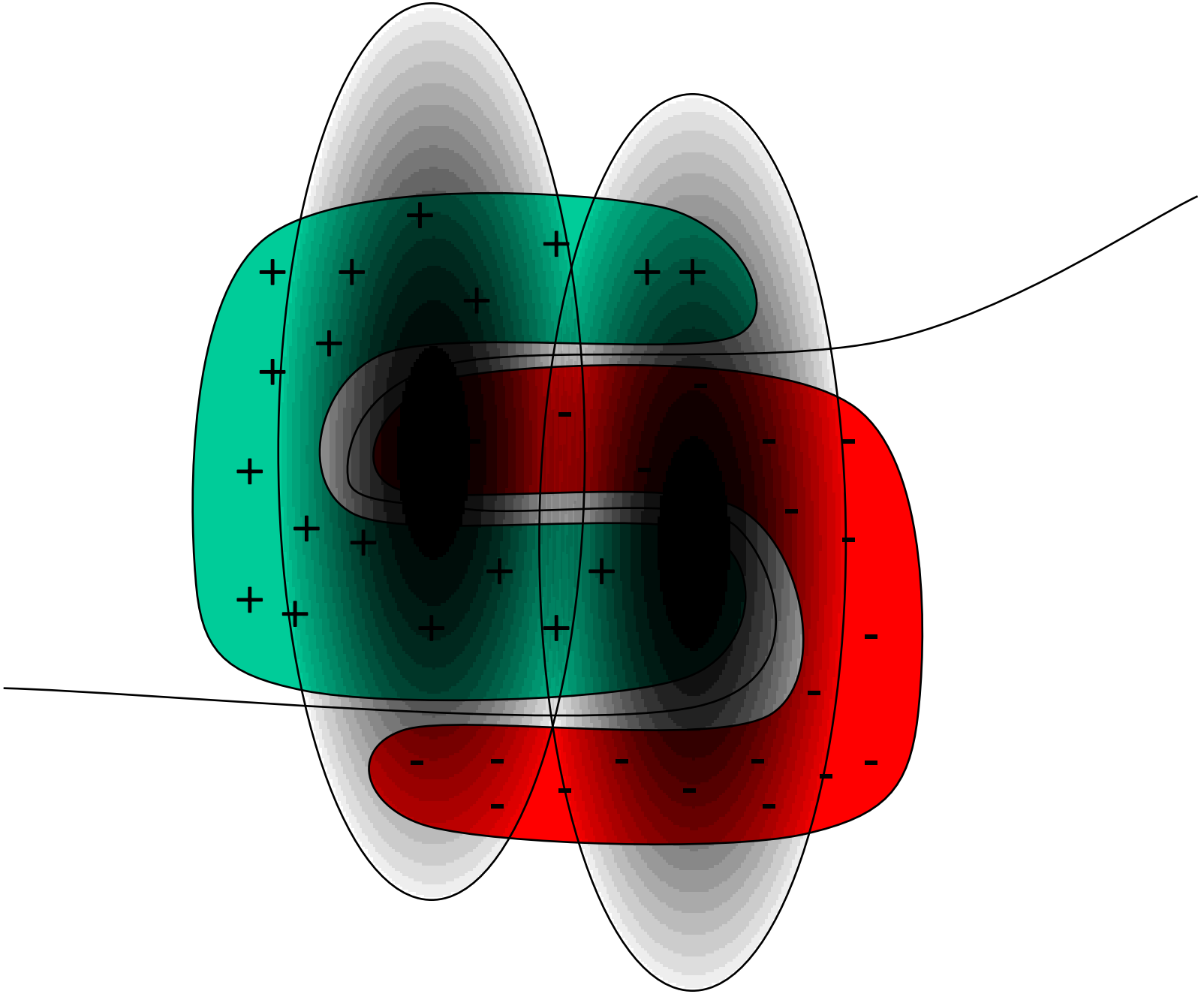
Kapitel 10
Die Support-Vektor-Maschine (SVM)

Ein statistischer Ansatz der Lerntheorie zum Entwurf eines optimalen Klassifikators

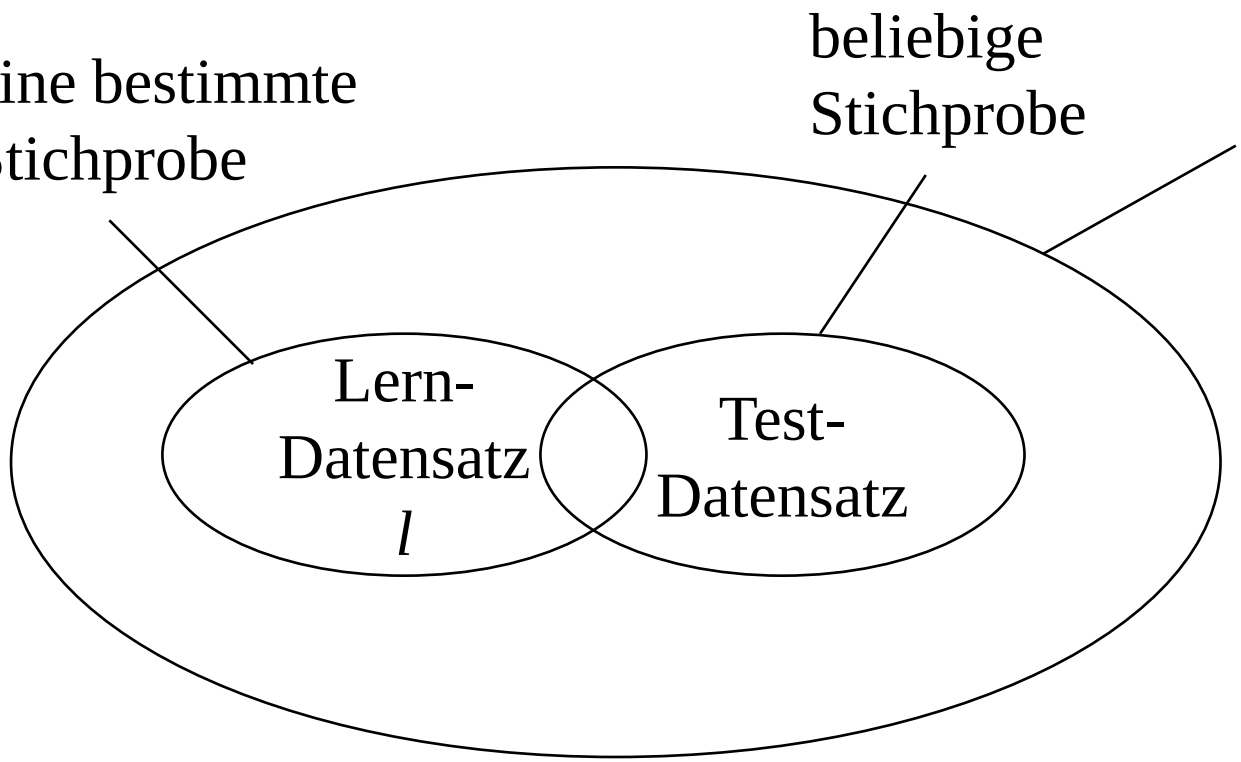
Inhalt:

1. Problemstellung
2. VC-Dimension und Gesamtfehlerminimierung
3. Lineare SVM
   - Separierbare Klassen (Beispiel)
   - Nichtseparierbare Klassen (Beispiel)
4. Nichtlineare SVM
   - Trick mit Kernfunktion (Beispiel)
   - Mercer`s Theorem
5. Eigenschaften und Berechnungskomplexität
6. Beispiele und Demos mit MATLAB

Durch Normalverteilungen schlecht darstellbare Klassen



Mustererkennungsexperiment



Stochastischer Prozess mit *unbekannter* Verbundverteilung: $P(\mathbf{x},y)$

Gesucht ist eine Zuordnungsfunktion $f(\mathbf{x},\boldsymbol{\alpha})$: $\mathbf{x}_i \rightarrow y_i$ welche durch die unbekannten Parameter des Vektors $\boldsymbol{\alpha}$ charakterisiert wird, welche den Erwartungswert des Zuordnungsfehlers minimiert:

$$R(\boldsymbol{\alpha})=E\{R_{test}(\boldsymbol{\alpha})\}$$

Empirischer Fehler

$R_{emp}(\boldsymbol{\alpha})$

Test-Fehler (Generalisierungs-fähigkeit)

$R_{test}(\boldsymbol{\alpha})$

Lerntheoretischer Ansatz

Überwachtes Lernen:

- Gegeben: *l* Beobachtungen (Lernstichprobe) aus dem Mustererkennungs-experiment mit der Verbundverteilung $P(\mathbf{x},y)$ mit den dazugehörigen Klassenzuordnungen (Labels) (Zunächst Beschränkung auf Zweiklassenproblem); ansonsten existiert *keinerlei Vorwissen*:

$$\left\{(\mathbf{x}_i, y_i) \in P(\mathbf{x}, y)\right\} \quad i = 1,\ldots,l \quad \text{mit: } \mathbf{x}_i \in \mathbb{R}^N, \quad y_i \in \left\{+1,-1\right\}$$

Gesucht:

- deterministische Zuordnungsfunktion, $f(\mathbf{x},\boldsymbol{\alpha})$: $\mathbf{x} \rightarrow y$ aufbauend auf eine Lernstichprobe, welche den *Erwartungswert des Zuordnungsfehlers beim Testdatensatz (expected risk)* minimiert:

$$R(\boldsymbol{\alpha}) = E\left\{R_{test}(\boldsymbol{\alpha})\right\} = E\left\{\tfrac{1}{2}\left|y - f(\mathbf{x},\boldsymbol{\alpha})\right|\right\} = \int \tfrac{1}{2}\left|y - f(\mathbf{x},\boldsymbol{\alpha})\right| dP(\mathbf{x},y)$$

$$= \int \tfrac{1}{2}\left|y - f(\mathbf{x},\boldsymbol{\alpha})\right| p(\mathbf{x},y)\, d\mathbf{x}dy \quad \text{falls Verbundverteilungsdichte bekannt}$$

Problem: dieser Ausdruck kann jedoch nicht ausgewertet werden, da $P(\mathbf{x},y)$ nicht zur Verfügung steht und ist somit wenig hilfreich!

Zentrales Problem der statistischen Lerntheorie: Wann führt ein niedriger Trainingsfehler zu einem niedrigen echten Fehler?

- Das *empirische Risiko*, nämlich die Fehlerrate für einen gegebenen Trainingsdatensatz lässt sich leicht berechnen gemäß:

$$R_{emp}(\boldsymbol{\alpha}) = \frac{1}{2l} \sum_{i=1}^{l} \left| y_i - f(\mathbf{x}_i, \boldsymbol{\alpha}) \right|$$

- Der Ansatz mit einem Neuronalen Netz und dem Backpropagation Learning z.Bsp. begnügt sich mit einer **E**mpirischen **R**isiko **M**inimierung (ERM)
- Hier nun die Frage: Wie nah ist man nach *l* Trainingsbeispielen am *echten* Fehler? Und: Wie gut kann aus dem empirischen Risiko das echte Risiko abgeschätzt werden (**S**tructural **R**isc **M**inimization (SRM) anstatt Empirical Risc Minimization (ERM)) ? Dies beinhaltet die Generalisierungsfähigkeit!
- Eine Antwort darauf versucht die Lerntheorie von Vapnik-Chervonenkis zu geben!

Eine Obergrenze für die Generalisierungsfähigkeit des Lernens mit der VC-Theorie

(Vapnik/Chervonenkis)

Mit der Wahrscheinlichkeit (1-η) gilt die folgende obere Abschätzung für den tatsächlichen Fehler (d.h. z.Bsp. mit η = 0,05 und damit mit einer Wahrscheinlichkeit von 95%):

$$R(\boldsymbol{\alpha}) \leq R_{emp}(\boldsymbol{\alpha}) + \underbrace{\Phi(h,l,\eta)}_{\text{VC-Konfidenz}}$$

mit: $$\Phi(h,l,\eta) = \sqrt{\frac{h\left(\log\frac{2l}{h}+1\right) - \log\left(\frac{\eta}{4}\right)}{l}}$$

Die Unsicherheit Φ wächst mit *h* und erniedrigt sich bei wachsender Lernstichprobe *l*

Entwurfsziel:
VC-Konfidenz möglichst klein und damit die Abschätzung möglichst scharf halten!

- *l* Anzahl der Trainingsbeispiele
- *h* VC-Dimension des verwendeten Hypothesenraums

Bemerkenswert ist, dass dieser Ausdruck unabhängig ist von der zugrundeliegenden Verbundverteilung P(**x**,y) ! (vorausgesetzt, die Trainings- und Testdatensätze werden statistisch unabhängig gezogen). D.h. falls man die Auswahl zwischen verschiedenen Lernmaschinen hat (einhergehend mit speziellen Familien von Abbildungsfunktionen f(**x**,**α**)), entscheidet man sich für die Maschine, welche bei gegebenem empir. Fehler den geringsten Wert für die Konfidenz Φ liefert.

Die VC-Dimension ist eine Eigenschaft für gegebene Funktionenklassen $\{f(\mathbf{x},\alpha)\}$

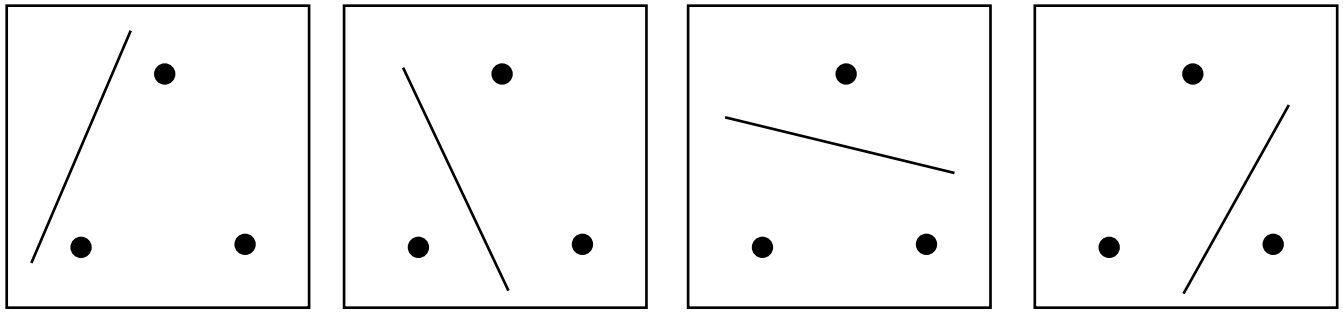
- Eine gegebene Menge von l Punkten kann für den Zweiklassen-Fall in 2^l mögliche Klassen aufgeteilt werden. Die VC-Dimension einer Menge von Funktionen $\{f(\mathbf{x},\alpha)\}$, ist definiert als die maximale Anzahl von Trainingspunkten, welche durch diese Klasse von Funktionen in allen möglichen Zuordnungen separiert werden können.

- Liefert Maße für „Kapazität“ oder Variabilität von Funktionenklassen
  - Funktionenklassen z.B.
    - Menge der linearen Trennfunktionen (Hyperebenen)
    - Menge künstlichen NN (MLP)
    - Menge der Polynomfunktionen
    - Menge der radialen Basisfunktionen

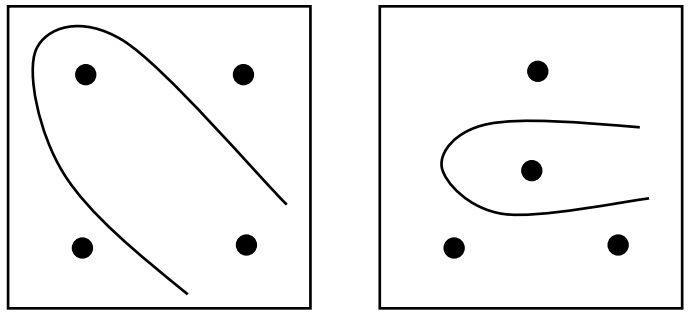
- Die VC-Konfidenz wächst monoton mit h: Demnach würde man bei einer Auswahl von Lernmaschinen, deren empirisches Risiko 0 ist, sich für diejenige entscheiden, deren assoziierte Menge von Abbildungsfunktionen minimale VC-Dimension besitzt (Modell so einfach wie möglich!).

VC-Dimension *h* von Hyperebenen, hier: $\mathbb{R}^2$

- Drei Punkte in $\mathbb{R}^2$ lassen sich mit *Hyperebenen* (Geraden) in allen Konstellationen trennen



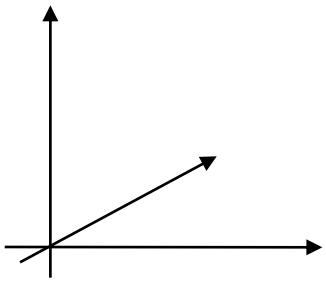
- Vier Punkte in $\mathbb{R}^2$ lassen sich nicht mehr mit *Hyperebenen* in allen Konstellationen trennen; jedoch z.B. gelingt es mit *Polynomen*!

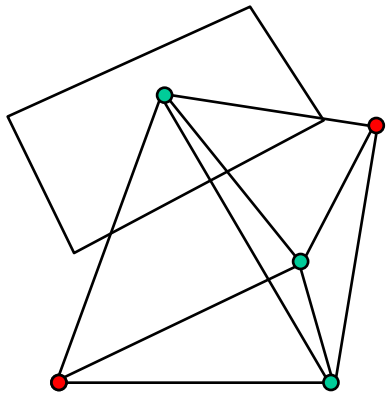


  - => Hyperebenen in $\mathbb{R}^2$: $h = 3$
  - die VC-Dimension bei Polynomen wächst mit ihrem Grad!

VC-Dimension *h* von Hyperebenen in $\mathbb{R}^3$

- Vier Punkte in $\mathbb{R}^3$ (Tetraeder) lassen sich mit *Hyperebenen* (Ebenen) in allen Konstellationen trennen
- Fünf Punkte in $\mathbb{R}^3$ lassen sich nicht mehr mit *Hyperebenen* in allen Konstellationen trennen (z.B. die roten Punkte in einer Klasse); jedoch z.B. gelingt es mit *Polynomen*!

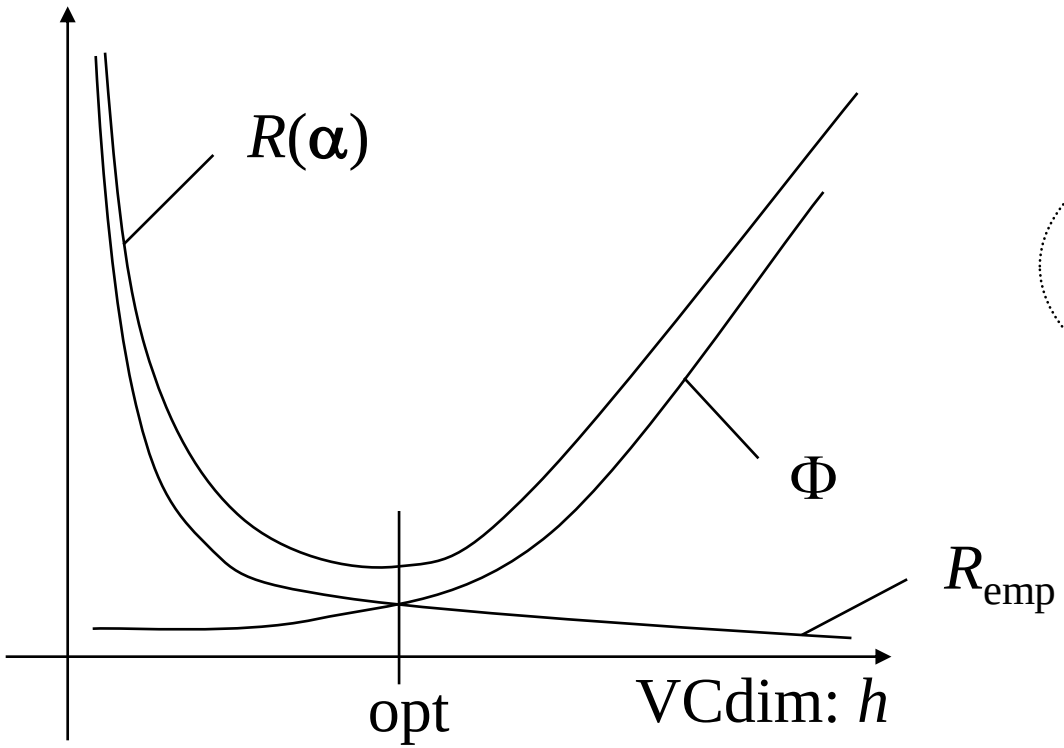


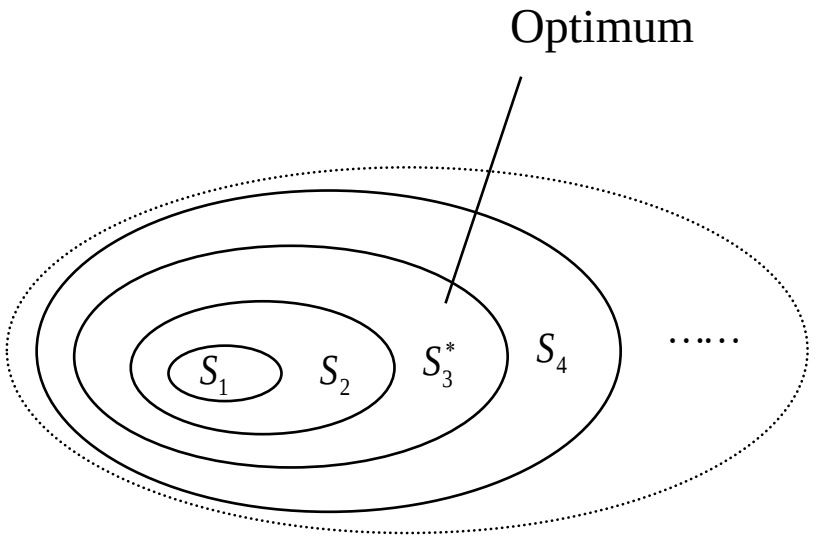


- => Hyperebenen in $\mathbb{R}^3$: $h = 4$
- Allgemein: Hyperebenen in $\mathbb{R}^N$: $h = N+1$

Structural Risk Minimization (SRM)

- SRM ist Minimieren der Abschätzung für $R(\alpha)$ über anwachsende Funktionenklassen S_i . Diese bilden Hypothesenräume mit anwachsender VC-Dimension VCdim=h_i ($h_1<h_2<h_3<...$).
- Kompromiss zwischen empirischem Risiko und Generalisierungsfähigkeit





- R_{emp} wird kleiner mit wachsendem h (overfitting)
- Φ wächst an mit h

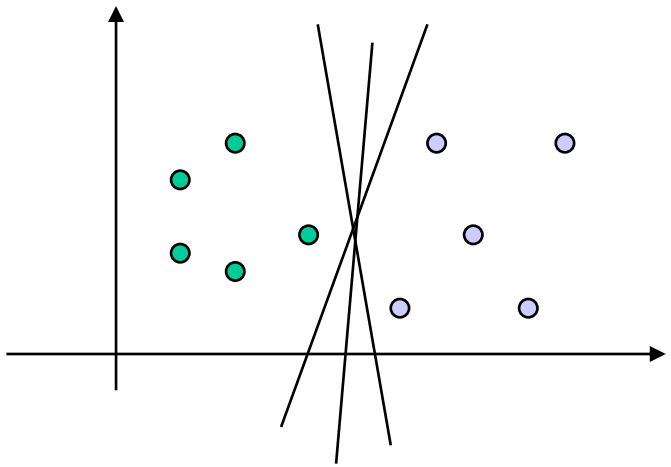
Entwurfshinweise

Mehrere Ausdrücke für das gleiche Phänomen:

- Bias/Varianz-Kompromiss
- Generalsierungs/Overfitting-Kompromiss
- Empirischer Fehler/VC-Dimensions-(Kapazitätskontrolle) -Kompromiss

Zunächst Überlegungen für linear separierbare Klassen

- Daten in $\mathbb{R}^2$



Es gibt verschiedene Lösungen!
Welche Qualität haben diese Lösungen?

  - Allgemeine Hyperebene $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$
  - Klassifikation via $f(x) = \mathrm{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$
  - z.B. Rosenblatt´s Perceptron (1956)

    Iteratives Lernen, Korrektur nach jeder Fehlklassifikation

    -> keine Eindeutigkeit der Lösung

Optimale Hyperebene: Maximierung des Randes

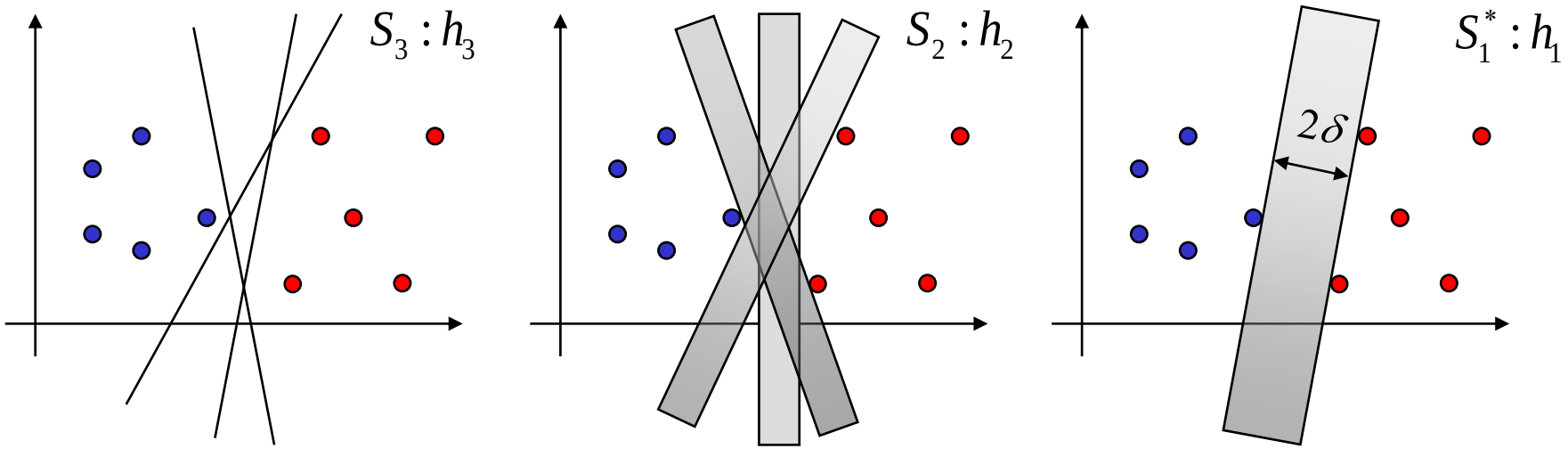
- Geht man von einem linear separierbarem Zweiklassenproblem aus, so erreichen alle Hyperebenen, welche beide Klassen trennen einen empirischen Fehler Null
- Die Konfidenz Φ wird minimiert durch ein Polynom minimaler VC-Dimension, nämlich eine Hyperebene
- Die VC-Dimension kann weiter abgesenkt werden durch „breite Hyperebenen“ (large margin hyperplanes)

• Trennende Hyperebene mit maximalem Rand		• Trennende Hyperebene mit minimaler VC-Dimension

- Dieses Ergebnis ist plausibel. Bei konstanter Intraklassenstreuung, wächst die Klassifikationssicherheit mit wachsendem Interklassenabstand.
- oder: bei konstant gehaltenem Interklassenabstand muss die Intraklassenstreuung maximal zulässig werden.

“Large-margin“-Klassifikator

– Hyperebene mit größtem Rand [Vap63]



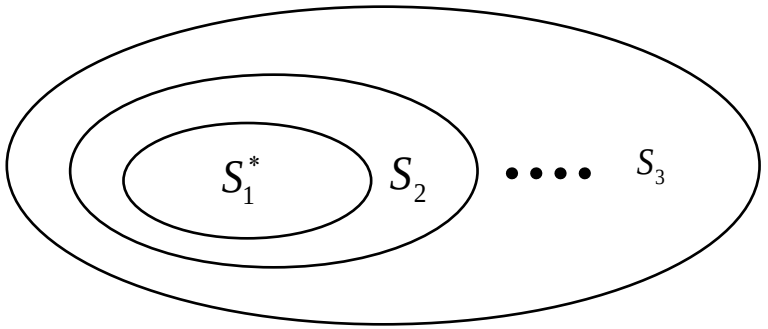
hohe VC-Dimension

in $\mathbb{R}^N$: $N+1$

mittlere VC-Dimension
Variabilität wird kleiner!

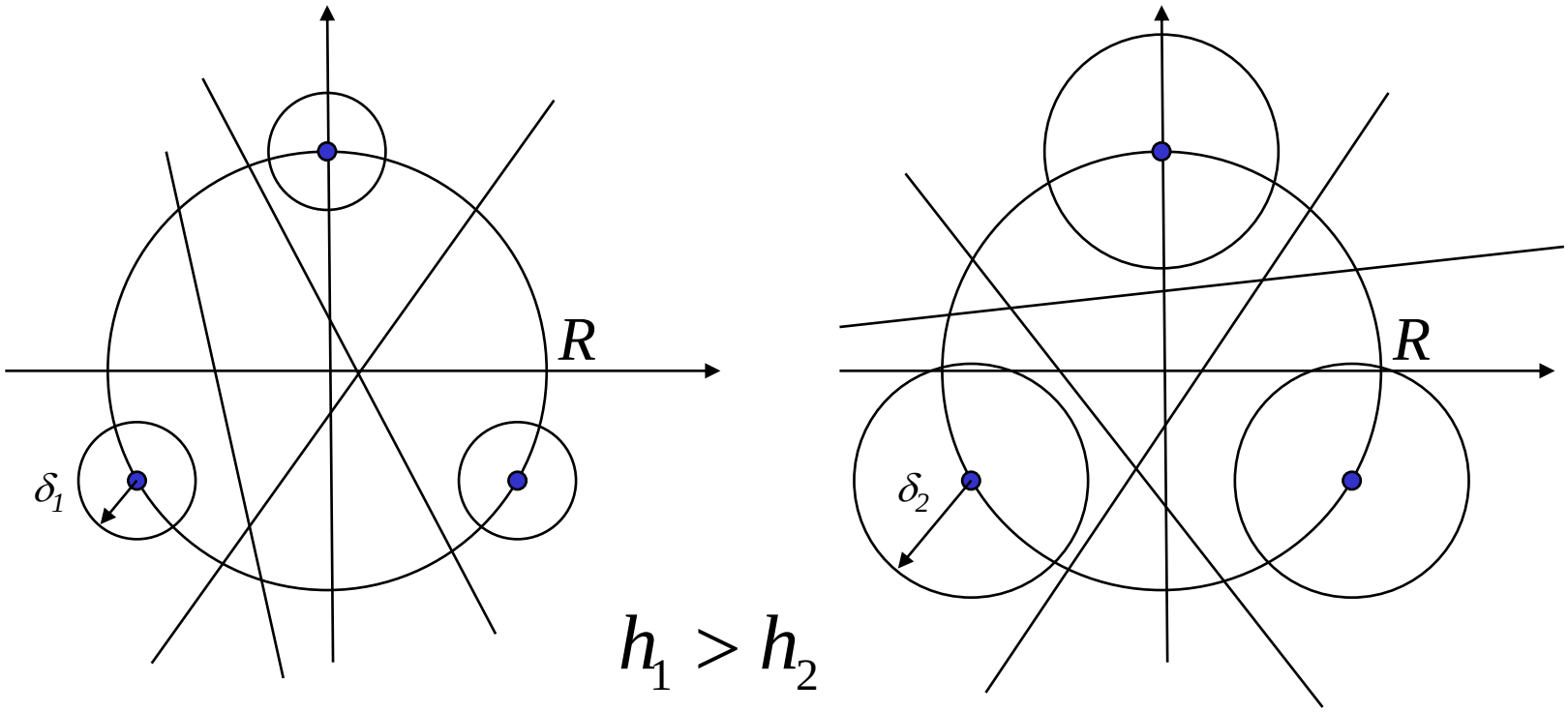
kleinste VC-Dimension
mit maximaler Breite
Variabilität gleich Null

- anschaulich sinnvoll
- theoretisch begründet
- Lösung abhängig von wenigen Daten:
  =>“Support-Vektoren“



$h_1 < h_2 < \ldots < h_3$

VC-Dimension von „breiten“ Hyperebenen

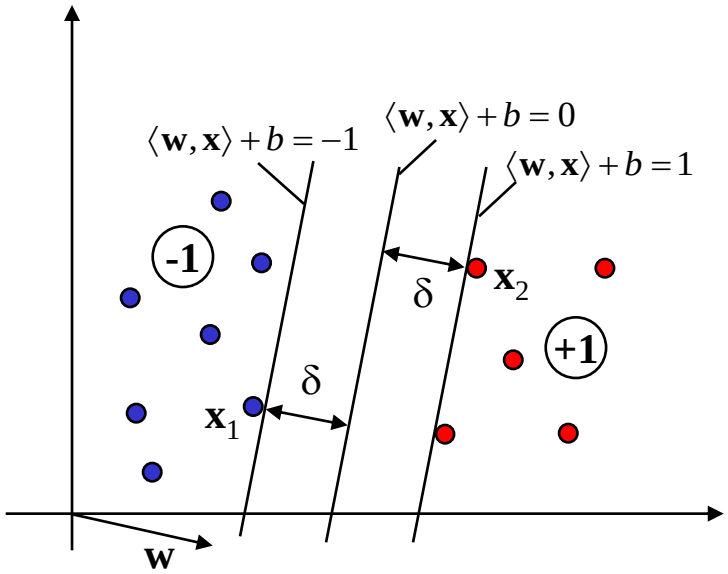


Die VC-Dimension h von Hyperebenen mit einem Mindestabstand δ von den zu trennenden Punkten ist begrenzt durch:

$$h \leq \frac{R^2}{\delta^2} + 1$$

=> grosser Margin δ, kleine VCdim h

Formalisierung: gesucht ist Trennebene mit maximalem Margin δ



Normalenvektor

Abstand eines Punktes von der Trennebene:

$$d(\mathbf{w}, b; \mathbf{x}) = \frac{|\langle \mathbf{w}, \mathbf{x} \rangle + b|}{\|\mathbf{w}\|}$$

Der Abstand zwischen den kanonischen Hyperebenen ergibt sich durch Projektion von $\mathbf{x}_1$-$\mathbf{x}_2$ auf den Normalenvektor $\mathbf{w}/\|\mathbf{w}\|$ unter Beachtung der Nebenbedingung (**NB**):

Die Daten werden korrekt klassifiziert, falls:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) > 0$$

NB: Einführung von *kanonischen Hyperebenen* durch die der Trennebene nächstgelegenen Punkte beider Klassen (***Supportvektoren*** $\mathbf{x}_1$,$\mathbf{x}_2$). Der Ausdruck ist invariant gegenüber einer positiven Reskalierung mit dem Faktor a :

$$y_i(\langle a\mathbf{w}, \mathbf{x}_i \rangle + ab) > 0$$

a wird so gewählt, dass gilt:

$\langle \mathbf{w}, \mathbf{x}_1 \rangle + b = -1$ Rand für blaue Klasse

$\langle \mathbf{w}, \mathbf{x}_2 \rangle + b = +1$ Rand für rote Klasse

$\{\mathbf{x}_1, \mathbf{x}_2\}$ Supportvektoren

$$2\delta = \left| \langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_2 - \mathbf{x}_1) \rangle \right| = \frac{1}{\|\mathbf{w}\|} \left| \underbrace{\langle \mathbf{w}, \mathbf{x}_2 \rangle}_{(1-b)} - \underbrace{\langle \mathbf{w}, \mathbf{x}_1 \rangle}_{-(1+b)} \right| = \frac{2}{\|\mathbf{w}\|}$$

$$\Rightarrow \quad \boxed{\delta = 1/\|\mathbf{w}\|}$$

Die Maximierung von δ ist gleichwertig mit einer Minierung von $\|\mathbf{w}\|^2$ =>

Quadratisches Optimierungsproblem für l Trainingsstichproben $(\mathbf{x}_i, y_i)$

Primales OP:

$$\text{minimiere} \quad J(\mathbf{w}, b) = \tfrac{1}{2}\|\mathbf{w}\|^2 = \tfrac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle$$

$$\text{unter der N.B.:} \quad \forall i \left[y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \right] \quad = 1 \text{ für SVn}$$

Einführung einer Lagrangefunktion mit den Lagrange-Faktoren α_i:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1]$$

mit: $\alpha_i \geq 0$

Die partiellen Ableitungen nach w_i, b und den α_i führen nach Einsetzen in das primale OP auf das äquivalente:

(wobei: $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$)

Wolf-duale OP:

$$\text{maximiere} \quad L'(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \boldsymbol{\alpha}^T \mathbf{1} - \tfrac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}$$

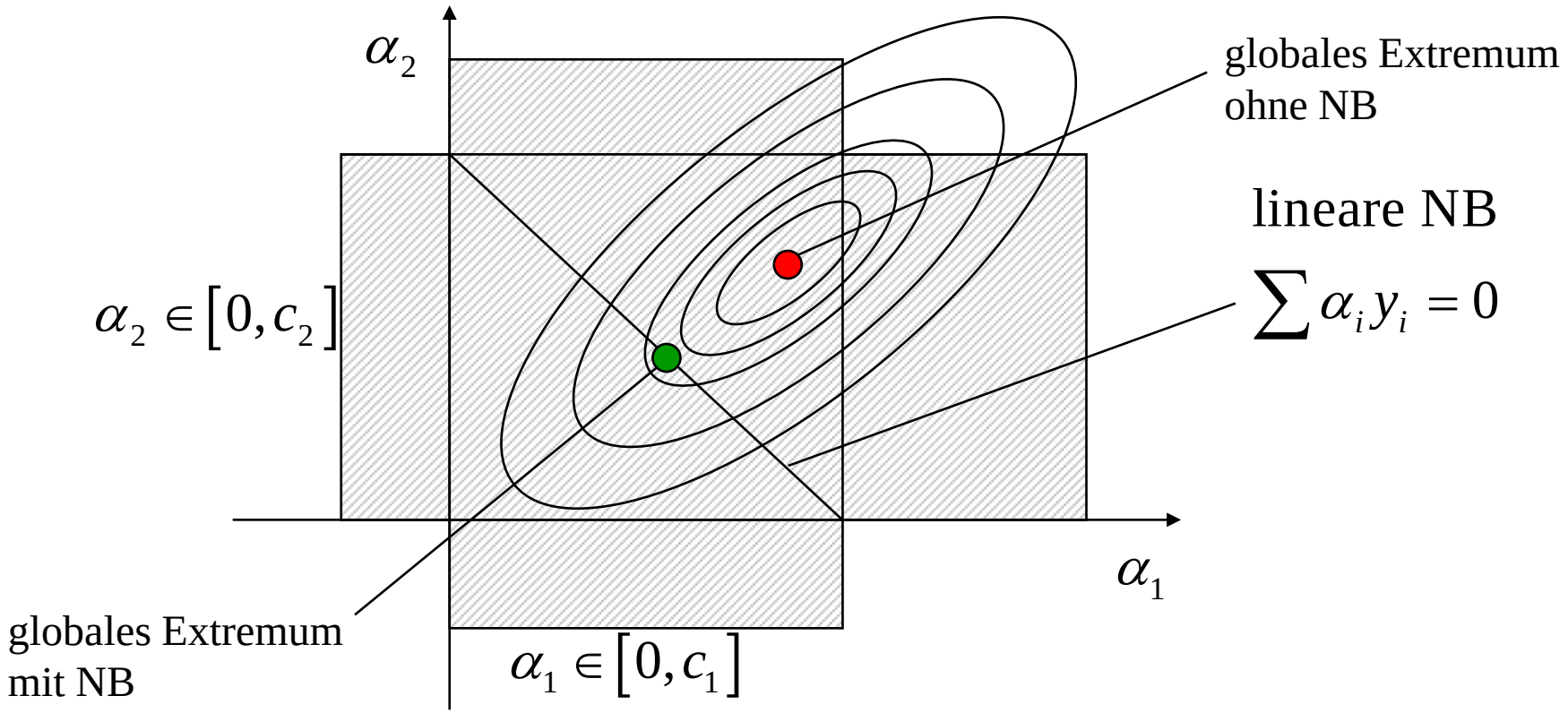
$$\text{unter den N.B.:} \quad 0 \leq \alpha_i \leq C \quad (\text{C}=\infty \text{ für den Fall ohne "slack"})$$

$$\text{und} \quad \sum_{i=1}^{l} y_i \alpha_i = \boldsymbol{\alpha}^T \mathbf{y} = 0 \qquad (\text{mit: } Q_{ij} = y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle)$$

Dies ist ein positiv semidefinites quadr. Problem für $\boldsymbol{\alpha}$ mit lin. NB, welches mit Hilfe der konvexen quadratischen Programmierung numerisch iterativ gelöst werden kann!

Globales, eindeutiges Optimum

- Optimierung eines quadratischen Problems in **w** bzw. **α** (konvex)
- unter linearen Nebenbedingungen ergibt sich ein globales Optimum
- beschränkt zugelassene Lösungsgebiete sind konvexe Mengen; mehrere lin. Beschränkungen ergeben als Schnitt wieder konvexe Mengen => diese geschnitten mit ursprünglichen Problem erhalten die Eigenschaften eines globalen Optimums, selbst wenn dieses jetzt auch am Rand liegen kann!



Lösung:

Lösung des dualen Problems für $\boldsymbol{\alpha^*}$ liefert eindeutig die gewünschte Hyperebene (Gln. gelten auch für den Fall mit Slack-Variablen!)

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{x}_i = \sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \mathbf{x}_i$$

$$b^* = -\frac{1}{2}\left(\max_{i, y_i = -1} (\langle \mathbf{w}^*, \mathbf{x}_i \rangle) + \min_{i, y_i = +1} (\langle \mathbf{w}^*, \mathbf{x}_i \rangle) \right) = -\frac{1}{2}\left(\max_{\mathbf{x}_i \in SV, y_i = -1} (\langle \mathbf{w}^*, \mathbf{x}_i \rangle) + \min_{\mathbf{x}_i \in SV, y_i = +1} (\langle \mathbf{w}^*, \mathbf{x}_i \rangle) \right)$$

Klassifikator:

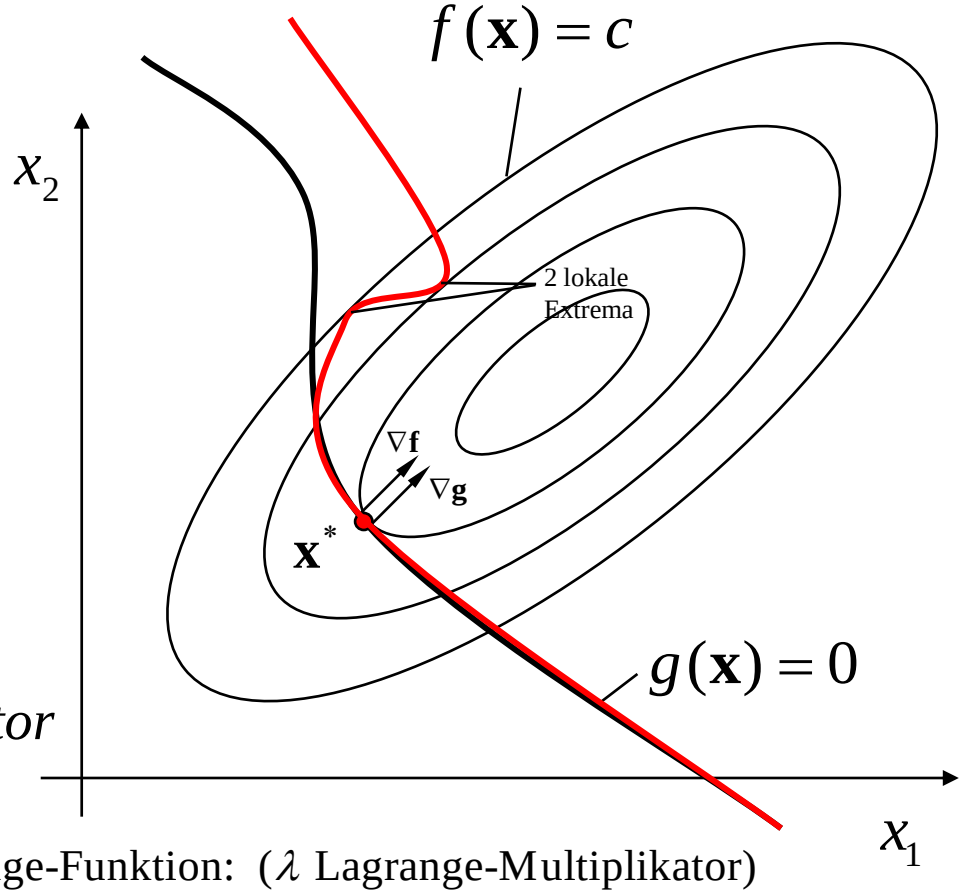
$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}^*, \mathbf{x} \rangle + b^*) = \operatorname{sgn}\left(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b^* \right)$$

Lösung nur abhängig von den Supportvektoren $(\alpha_i > 0)$ ***!!***

Beobachtungen:

- Für die Support-Vektoren gilt: $0 < \alpha_i < \infty$
- Für alle Nicht-Support-Vektoren gilt: $\alpha_i = 0$
  => Support-Vektoren, "sparse"-Darstellung der Lösung
- Eindeutigkeit der Trennebene, *globales Optimum*!!

Optimierung von $f(\mathbf{x})$ unter gegebenen Nebenbedingungen $g(\mathbf{x})=0$ mit Hilfe des Lagrange-Ansatzes



$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}$$

Gradient = *Normalenvektor*

$$f(\mathbf{x}) \overset{!}{=} \min_{\mathbf{x}}$$

NB: $g(\mathbf{x}) = 0$

Eine Lösung für das Optimierungsproblem zu finden ist gleichbedeutend mit der Aufgabe, stationäre Punkte $\mathbf{x}^*$ zu finden, in denen gilt:

$$\boxed{\nabla \mathbf{f} \parallel \nabla \mathbf{g}}$$

Aus der Lagrange-Funktion: (λ Lagrange-Multiplikator)

$$\boxed{L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})}$$

folgt mit der notwendigen Bedingung für ein Extremum:

$$\nabla \mathbf{L} = \frac{\partial L}{\partial \mathbf{x}} = \nabla \mathbf{f} + \lambda \nabla \mathbf{g} = \mathbf{0} \quad \Rightarrow \nabla \mathbf{f} = -\lambda \nabla \mathbf{g}$$

$\Rightarrow \boxed{\nabla \mathbf{f} \parallel \nabla \mathbf{g}}$ dies ist aber genau die Bedingung für einen stationären Punkt!

Dies ist nur eine notwendige Bedingung für ein lokales Optimum; es kann mehrere Lösungen geben!
Bei linearer NB gibt es nur eine Lösung!

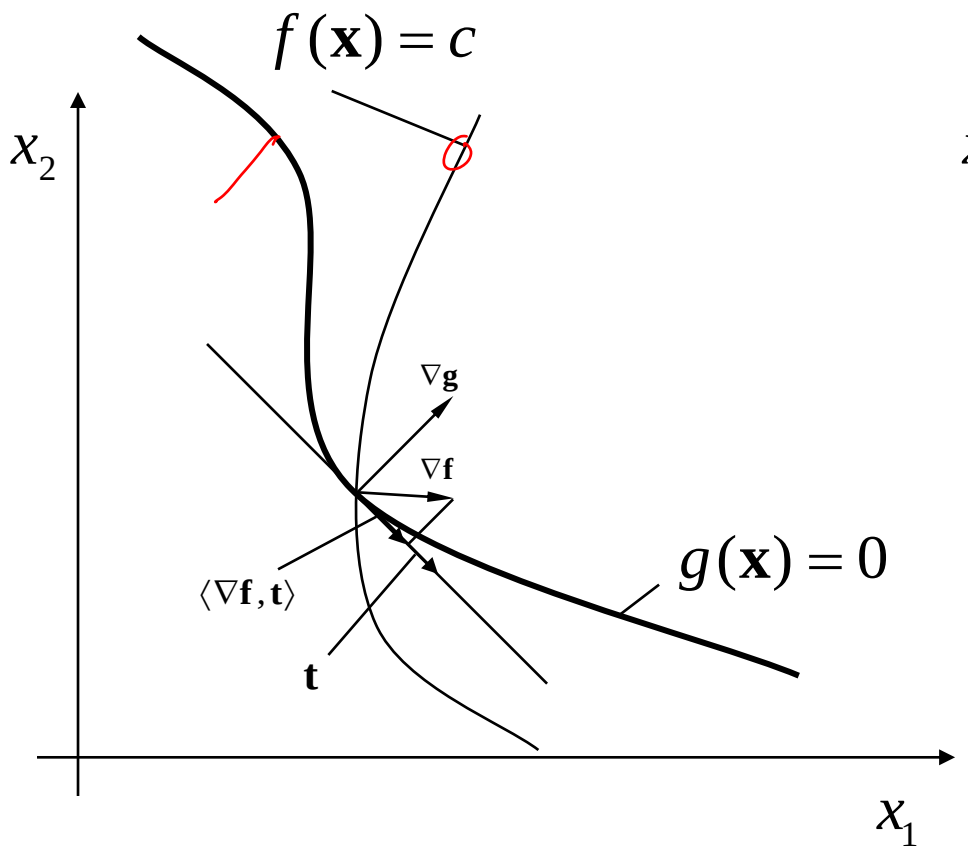
Der Gradient ist ein Normalenvektor an die Kurve $g(\mathbf{x})=0$

Betrachten wir die Taylorentwicklung von g bzgl. einer kleinen vektoriellen Störung $\mathbf{t}$ an einer Stelle $\mathbf{x}$ auf der Kurve $g(\mathbf{x})=0$, so erhält man:

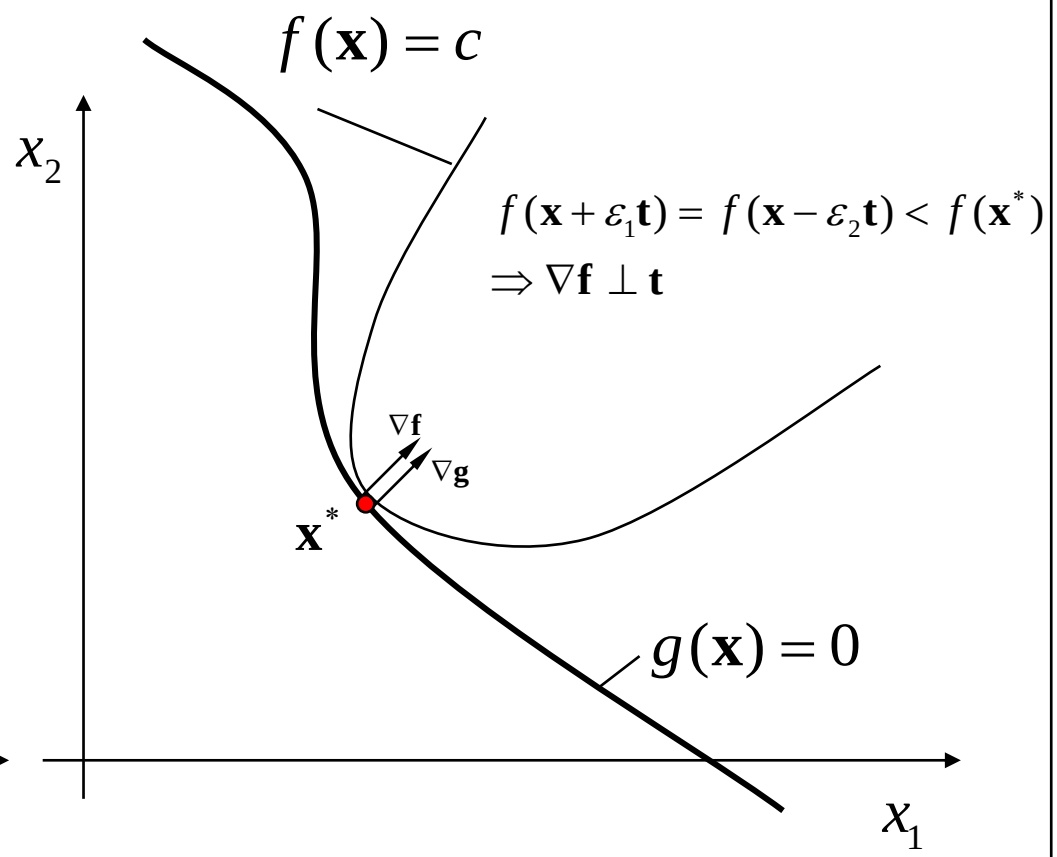
$$g(\mathbf{x}+\mathbf{t}) = g(\mathbf{x}) + \mathbf{t}^T \nabla \mathbf{g} + \ldots \quad \text{mit: } \nabla \mathbf{g} = \frac{\partial g}{\partial \mathbf{x}}$$

Bewegt sich eine kleine Störung $\mathbf{t}$ tangential entlang der Kurve, so gilt $g(\mathbf{x}+\mathbf{t}) \approx g(\mathbf{x})$ und somit auch $\mathbf{t}^T \nabla \mathbf{g}(\mathbf{x}) = 0$. Daraus erkennen wir, dass der Gradient senkrecht steht auf den Tangentenvektor entlang der Kurve (Oberfläche) $g(\mathbf{x})=0$ und damit per Definition der Normalenvektor ist.

Bedingung für einen stationären Punkt **x***



a) Verhältnisse an einem nichtstationären Punkt



b) Verhältnisse an einem stationären Punkt

a) Enthält die Projektion von **∇f** auf die Tangentenrichtung **t** einen von Null verschiedenen Beitrag, so kann das Optimierungskriterium durch Bewegung in diese Richtung entlang der Kurve der NB verbessert werden! => kein stationärer Punkt!

b) An einem stationären Punkt wird das Gütekriterium f(x) in *beiden* tangentialen Richtungen verschlechtert. **∇f** steht senkrecht auf **t**.

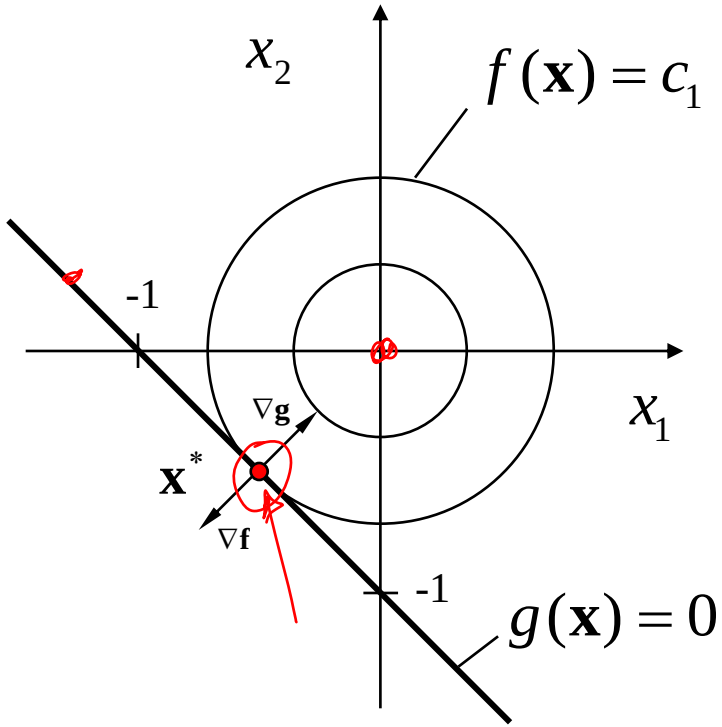
Beispiel:

1) OP: $f(\mathbf{x}) = x_1^2 + x_2^2$

2) NB: $g(\mathbf{x}) = x_1 + x_2 + 1 = 0$

$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = 2 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\nabla \mathbf{g} = \begin{bmatrix} \frac{\partial g}{\partial x_1} \\ \frac{\partial g}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$



$$\nabla \mathbf{f}^* \parallel \nabla \mathbf{g}^*$$

$L(\mathbf{x}, \alpha) = f(\mathbf{x}) + \alpha g(\mathbf{x})$ Langrange-Funktion

notw. Bed. für ein Optimum:

1) $\nabla \mathbf{L} = \frac{\partial L}{\partial \mathbf{x}} = \nabla \mathbf{f} + \alpha \nabla \mathbf{g} = \mathbf{0}$

2) $\frac{\partial L}{\partial \alpha} = g(\mathbf{x}) = 0$

$\Rightarrow$

1) $2x_1 + \alpha = 0$

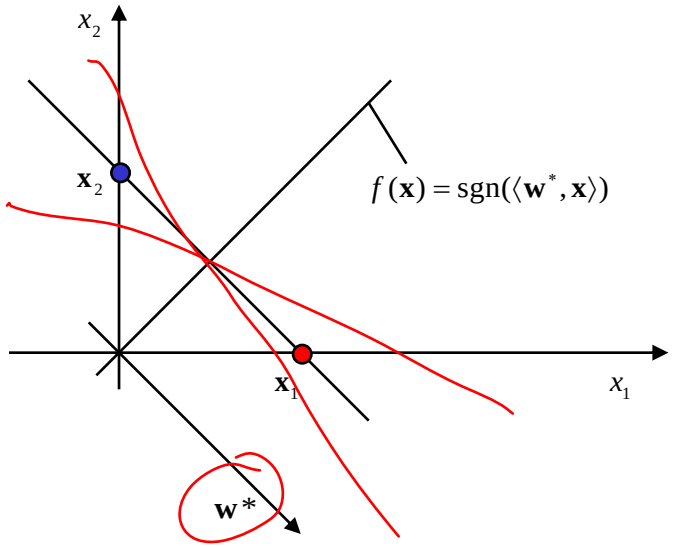
$2x_2 + \alpha = 0$

2) $x_1 + x_2 + 1 = 0$

$$\Rightarrow \alpha^* = 1 \text{ und } \mathbf{x}^* = -\begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

$$\Rightarrow \nabla \mathbf{f}(\mathbf{x}^*) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} = -\alpha^* \nabla \mathbf{g}(\mathbf{x}^*) = -1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Beispiel: Entwurf einer SVM für 2 Punkte

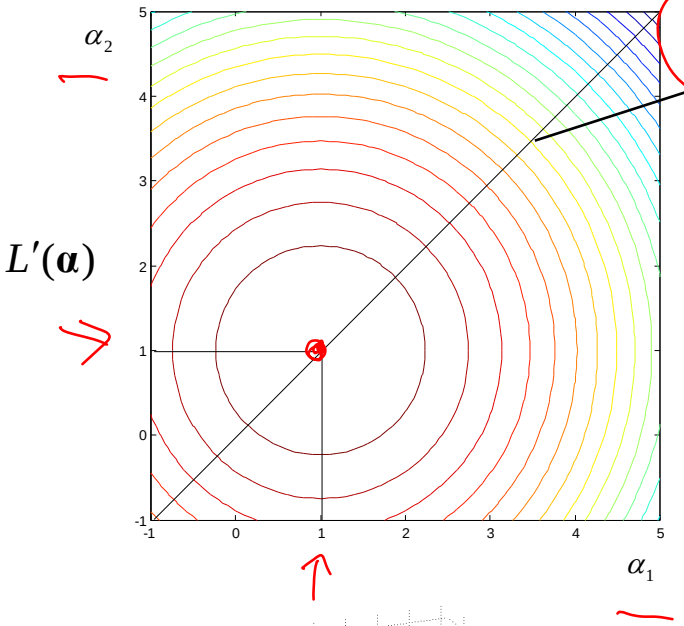


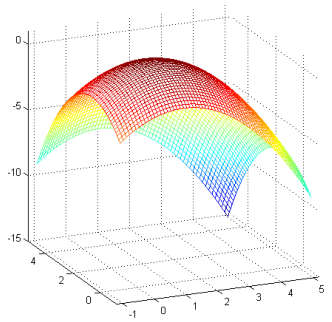
$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad y_1 = +1 \quad y_2 = -1$$

Beide Vektoren sind Supportvektoren!

NB:

$$\alpha_1 = \alpha_2$$





1) maximiere $L'(\boldsymbol{\alpha}) = \sum_{i=1}^{2} \alpha_i - \frac{1}{2} \sum_{i=1}^{2} \sum_{j=1}^{2} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle = (\alpha_1 + \alpha_2) - \frac{1}{2}(\alpha_1^2 + \alpha_2^2)$

2) unter den N.B.: $0 \le \alpha_i < \infty$

3) und $\sum_{i=1}^{2} y_i \alpha_i = (\alpha_1 - \alpha_2) = 0$

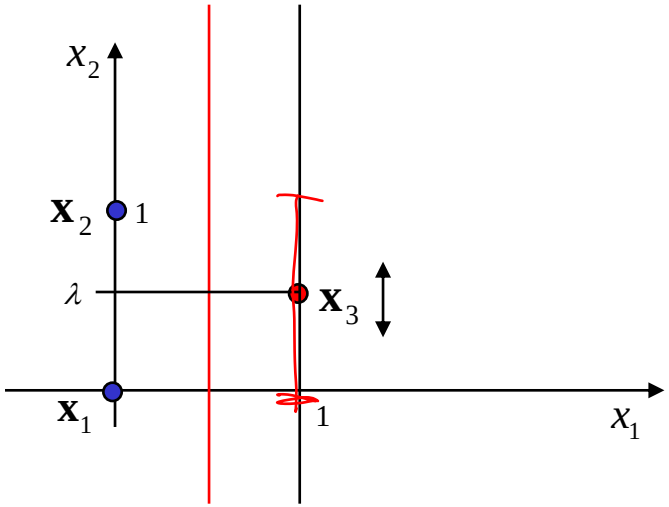
Lösung: $\alpha_1^* = \alpha_2^* = 1$

$$\mathbf{w}^* = \sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \mathbf{x}_i = \mathbf{x}_1 - \mathbf{x}_2 = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$

$$b^* = -\frac{1}{2}\left(\max_{\mathbf{x}_i \in SV, y_i = -1} (\langle \mathbf{w}^*, \mathbf{x}_i \rangle) + \min_{\mathbf{x}_i \in SV, y_i = +1} (\langle \mathbf{w}^*, \mathbf{x}_i \rangle) \right) = -\frac{1}{2}(1 - 1) = 0$$

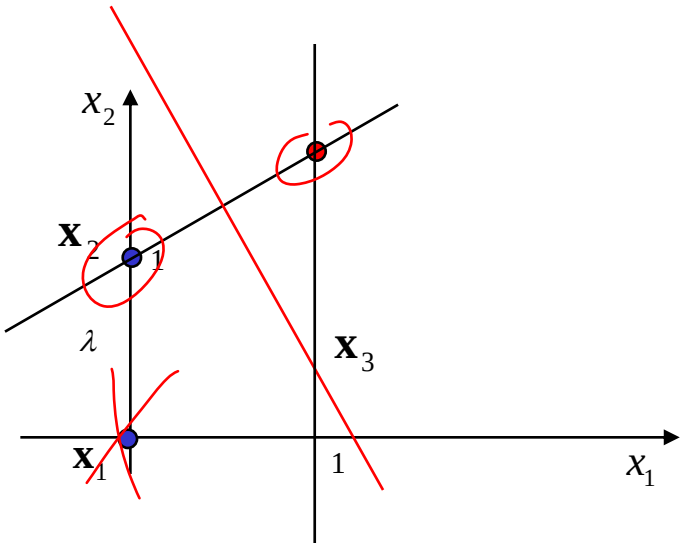
Klassifikation: $f(\mathbf{x}) = \mathrm{sgn}(\langle \mathbf{w}^*, \mathbf{x} \rangle)$

Beispiel: Entwurf einer SVM für 3 Punkte



$$\mathbf{x}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 1 \\ \lambda \end{bmatrix} \qquad y_1 = y_2 = +1 \qquad y_3 = -1$$

Fall I: für $0 \leq \lambda \leq 1$ immer drei Supportvektoren



Fall II: für sonst. Werte: zwei Supportvektoren

Beispiel: Entwurf einer SVM für 3 Punkte

1) maximiere $L'(\boldsymbol{\alpha}) = \sum_{i=1}^{3} \alpha_i - \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle = (\alpha_1 + \alpha_2 + \alpha_3) - \frac{1}{2}(\alpha_2^2 - 2\lambda\alpha_2\alpha_3 + (1+\lambda^2)\alpha_3^2)$

2) unter den N.B.: $0 \le \alpha_i < \infty$

3) und $\sum_{i=1}^{3} y_i \alpha_i = (\alpha_1 + \alpha_2 - \alpha_3) = 0 \quad \Rightarrow \quad \alpha_1 = \alpha_3 - \alpha_2$

3) Setzt man die lin. NB 3) in 1) ein so ergibt sich:

4) maximiere $L'(\alpha_2, \alpha_3) = 2\alpha_3 - \frac{1}{2}(\alpha_2^2 - 2\lambda\alpha_2\alpha_3 + (1+\lambda^2)\alpha_3^2)$

5) unter den N.B.: $0 \le \alpha_i < \infty$

Die Lösung von 4) ohne Beachtung der NB 2) (unbeschränkte Lösung) erhält man mit:

$$\frac{\partial L'(\alpha_2, \alpha_3)}{\partial \alpha_2} = -\alpha_2 + \lambda\alpha_3 \stackrel{!}{=} 0 \quad \Rightarrow \quad \alpha_2 = \lambda\alpha_3$$

$$\frac{\partial L'(\alpha_2, \alpha_3)}{\partial \alpha_3} = 2 + \lambda\alpha_2 - (1+\lambda^2)\alpha_3 \stackrel{!}{=} 0 \quad \Rightarrow \quad 2 + \lambda^2\alpha_3 - (1+\lambda^2)\alpha_3 = 0$$

und daraus schließlich:

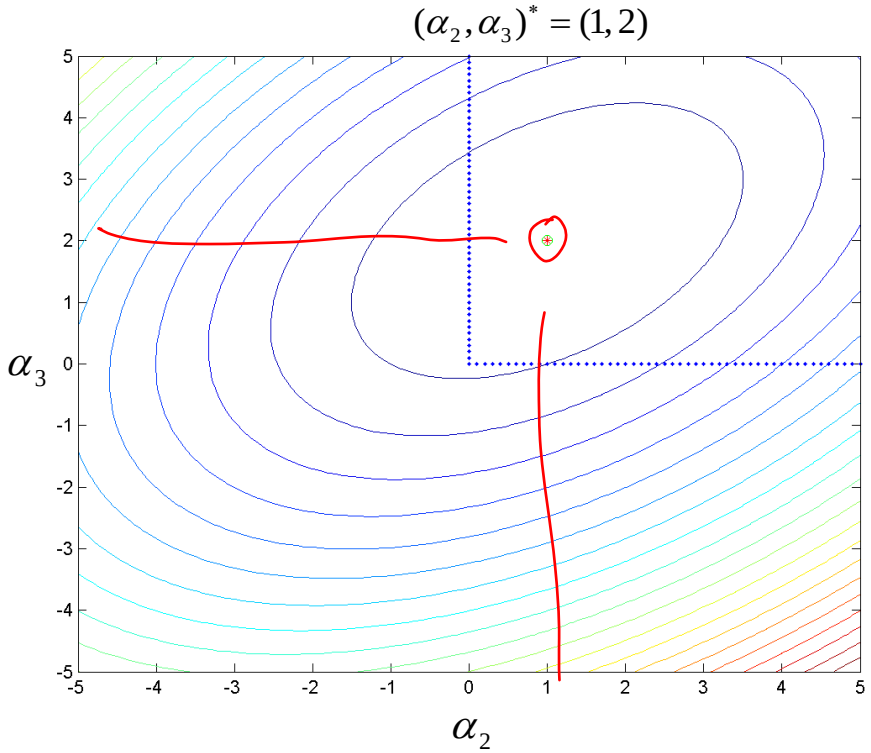
$\alpha_1 = 2(1-\lambda) \quad \alpha_2 = 2\lambda \quad \alpha_3 = 2$ für $0 \le \lambda \le 1$ sind alle $\alpha_i \ge 0$ (Fall I)

dafür gilt:

$$\mathbf{w}^* = \sum_{\mathbf{x}_i \in SV} \alpha_i y_i \mathbf{x}_i = 2\lambda \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 2 \begin{bmatrix} 1 \\ \lambda \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

$$b^* = -\frac{1}{2}\left(\langle \mathbf{w}^*, \mathbf{x}_3 \rangle + \min_{\mathbf{x}_i \in SV, y_i = +1} (0, \langle \mathbf{w}^*, \mathbf{x}_2 \rangle) \right) = -\frac{1}{2}(-2 + 0) = 1$$

Fall I: λ=0,5 absolutes Maximum im Bereich $\alpha_i \geq 0$



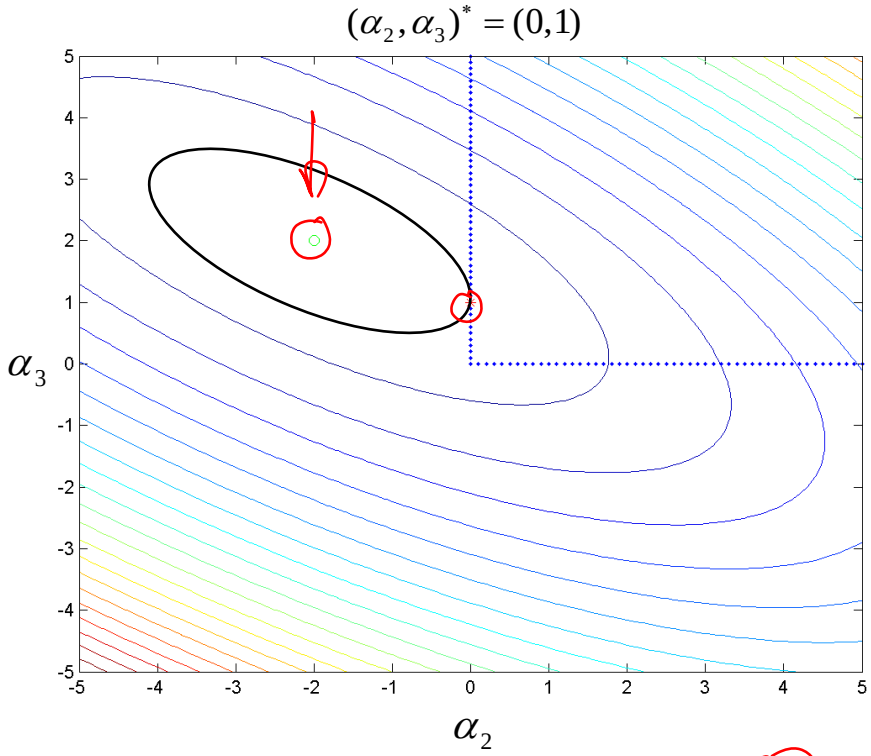
unbeschränkte Lösung mit $\lambda = 0,5$:

$\alpha_1 = 2(1-\lambda) = 1$

$\alpha_2 = 2\lambda = 1$

$\alpha_3 = 2$

Fall II: λ = -1 absolutes Maximum außerhalb des Bereichs $\alpha_i \geq 0$, richtige Lösung am Rand

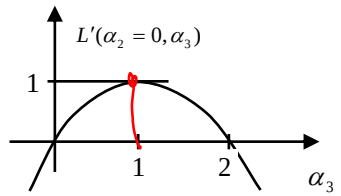


Hier liegt die Lösung offensichtlich auf dem Rand mit $\alpha_2 = 0$, d.h. $\mathbf{x}_2$ ist kein Support-Vektor! Das Maximum von

$$L'(\alpha_2, \alpha_3)\big|_{\alpha_2=0,\lambda=-1} = 2\alpha_3 - \tfrac{1}{2}((1+\lambda^2)\alpha_3^2) = \alpha_3(2-\alpha_3)$$

ergibt sich aus:

$$\frac{dL'(\alpha_3)}{d\alpha_3} = 2 - 2\alpha_3 = 0 \quad \Rightarrow \quad \alpha_3 = 1$$



und damit: $\boxed{(\alpha_1, \alpha_2, \alpha_3) = (1,0,1)}$

Das absolute Maximum liegt hingegen bei: $\boxed{(\alpha_1, \alpha_2, \alpha_3) = (4,-2,2)}$

Demos mit MATLAB

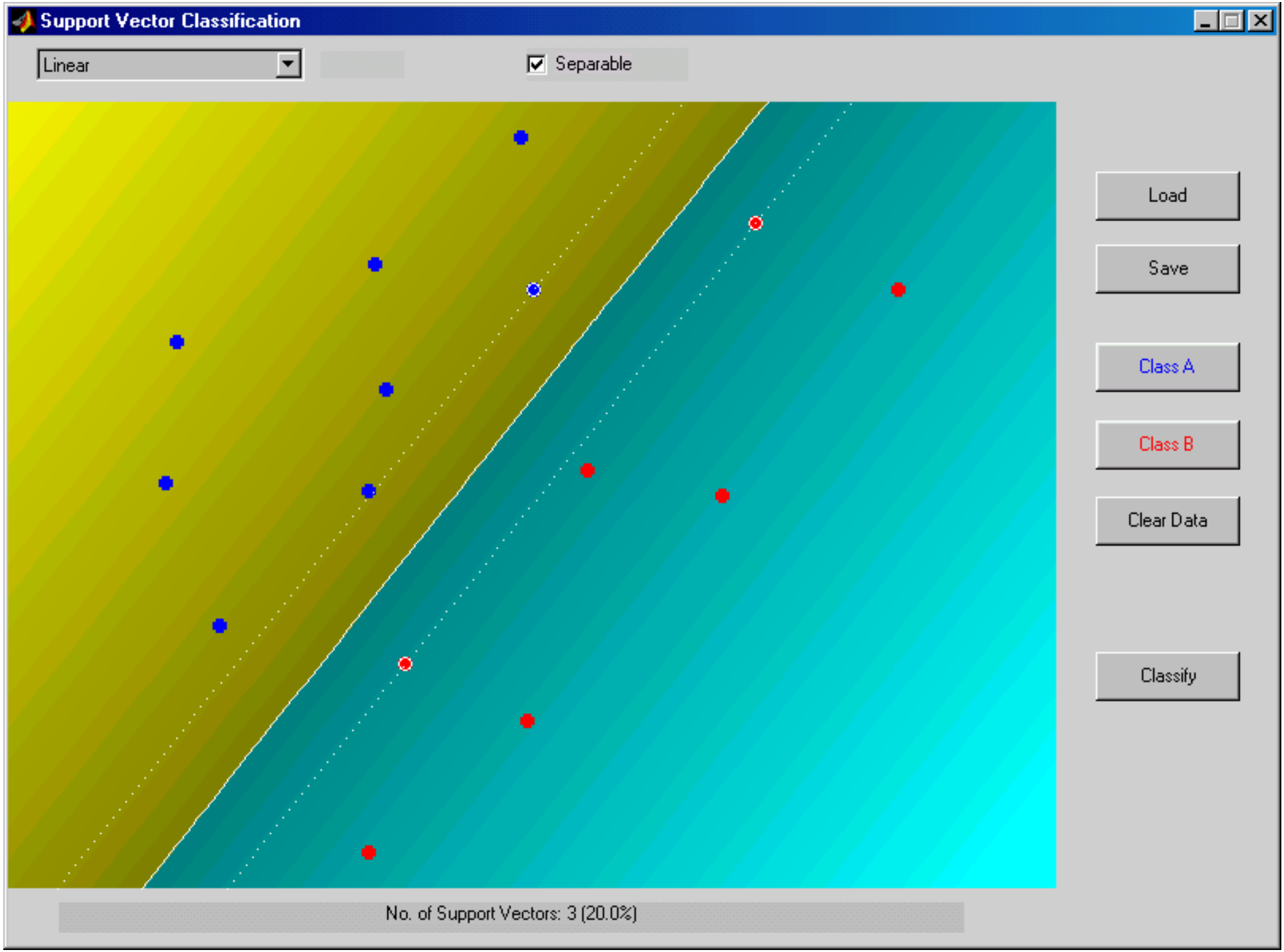
(svmmatlab, uiclass.m)

- Linearer Klassifikator
  - Problem linear separierbar
    - Harter Rand, Problem separierbar
    - Harter kleiner Rand wegen Ausreißer, Problem separierbar, aber schlechte Generalisierung => emp. Fehler vergrößern mit Gewinn bei Generalisierung => weichen Rand einführen
  - Problem nicht linear separierbar
    - Weichen Rand verwenden; Kontrolle mit *C*
- Nichtlinearer Klassifikator
  - Für nichtlineare Probleme muss VC-Dimension der trennenden Hyperflächen und damit ihre Kapazität vergrößert werden!
  - XOR mit Polynom 2. Grades
  - Lineare Separierung eines quadratischen Problems mit weichem Rand; nicht zufriedenstellend (\ME_2009\matlab\svmmatlab\parabel.mat laden)
  - Polynomiale Separierung (p=2), harter Rand
  - Polynomiale Separierung (p=4), harter Rand
  - Bananenshape mit Polynomkernen (banane.mat laden, Polynomgrad 3)
  - Bananenshape mit Gauss-Radialbasisfunktionen (Gauss-RBF, σ=3)
  - Beispiel Inseln.mat (Polynom Grad 2,3,4,10; Gauss-RBFs mit σ=3, σ=0.1 overfitting)

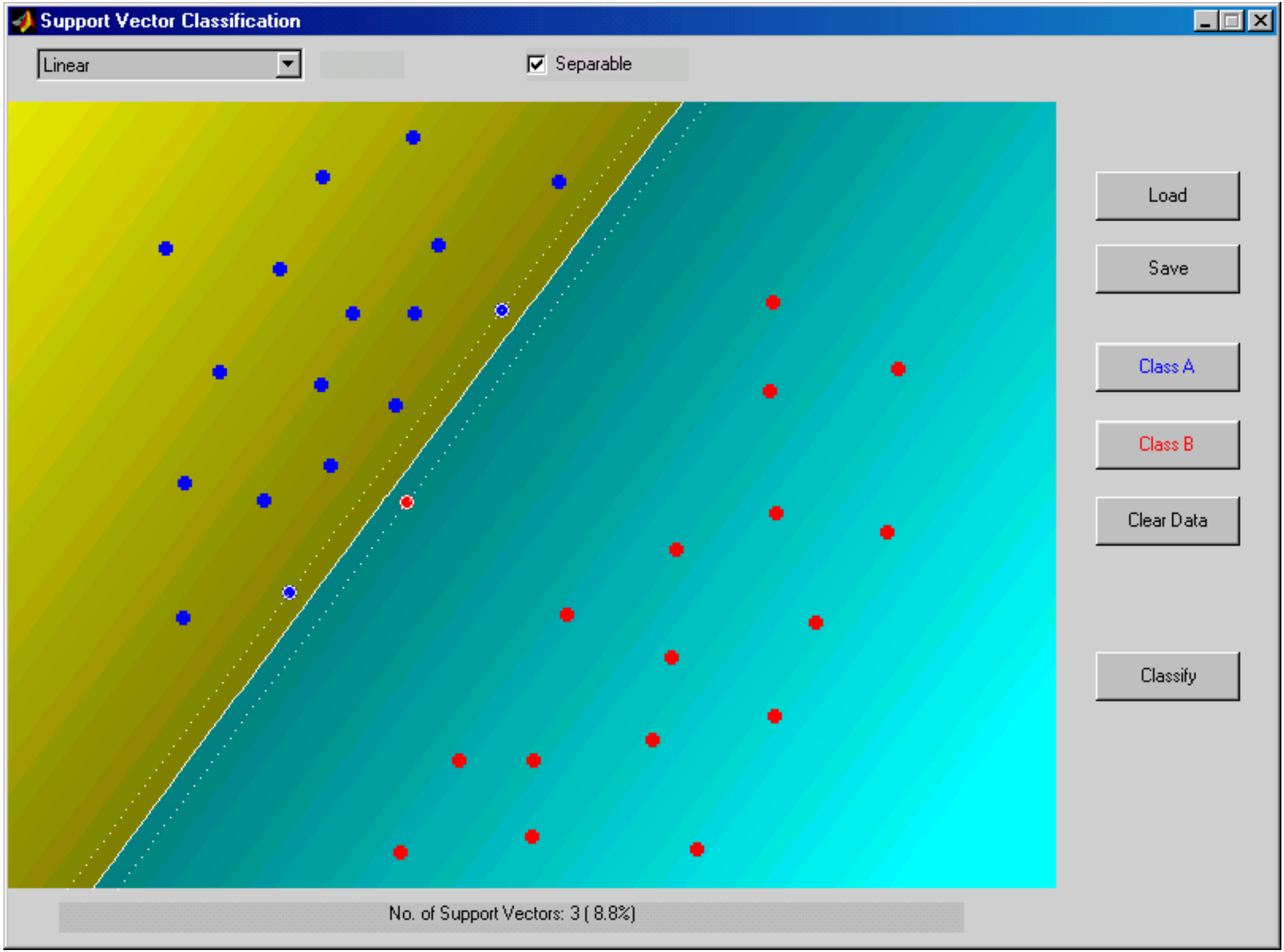
Start der Matlab-Demo

matlab-SVM.bat

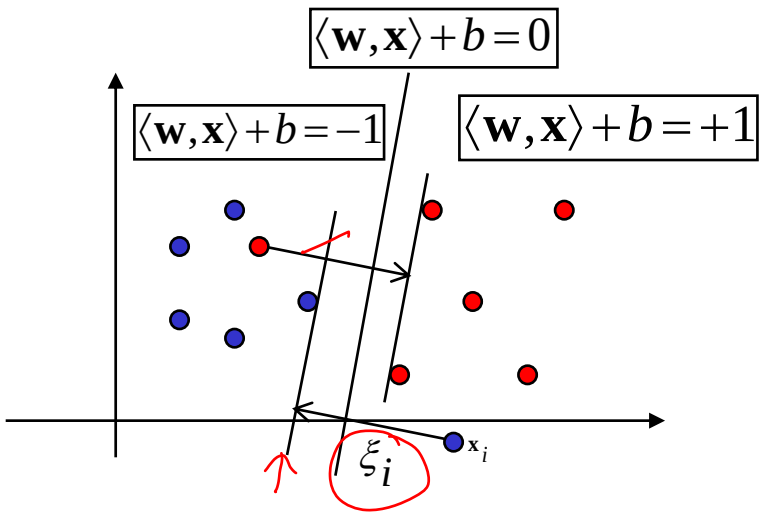
Linear separierbare Klassen; harter Rand (C= ∞)



Linear separiebare Klassen; harter, kleiner Rand, schlechte Generalisierung



Nichtseparabler Fall



Bestrafen von ***Randverletzungen*** mit "slack"(Schlupf)-Variablen [Smith68] ->"Soft-Margin" SVM

minimieren von: $\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i$

mit: $y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 - \xi_i$ und $\xi_i \geq 0$

- C kontrolliert Separationsweite u. Trainingsfehler; hohes C bedeutet hohe Bestrafung!
- $\Sigma\xi_i$ ist obere Schranke für Trainingsfehler

Gründe für diese Verallgemeinerung:

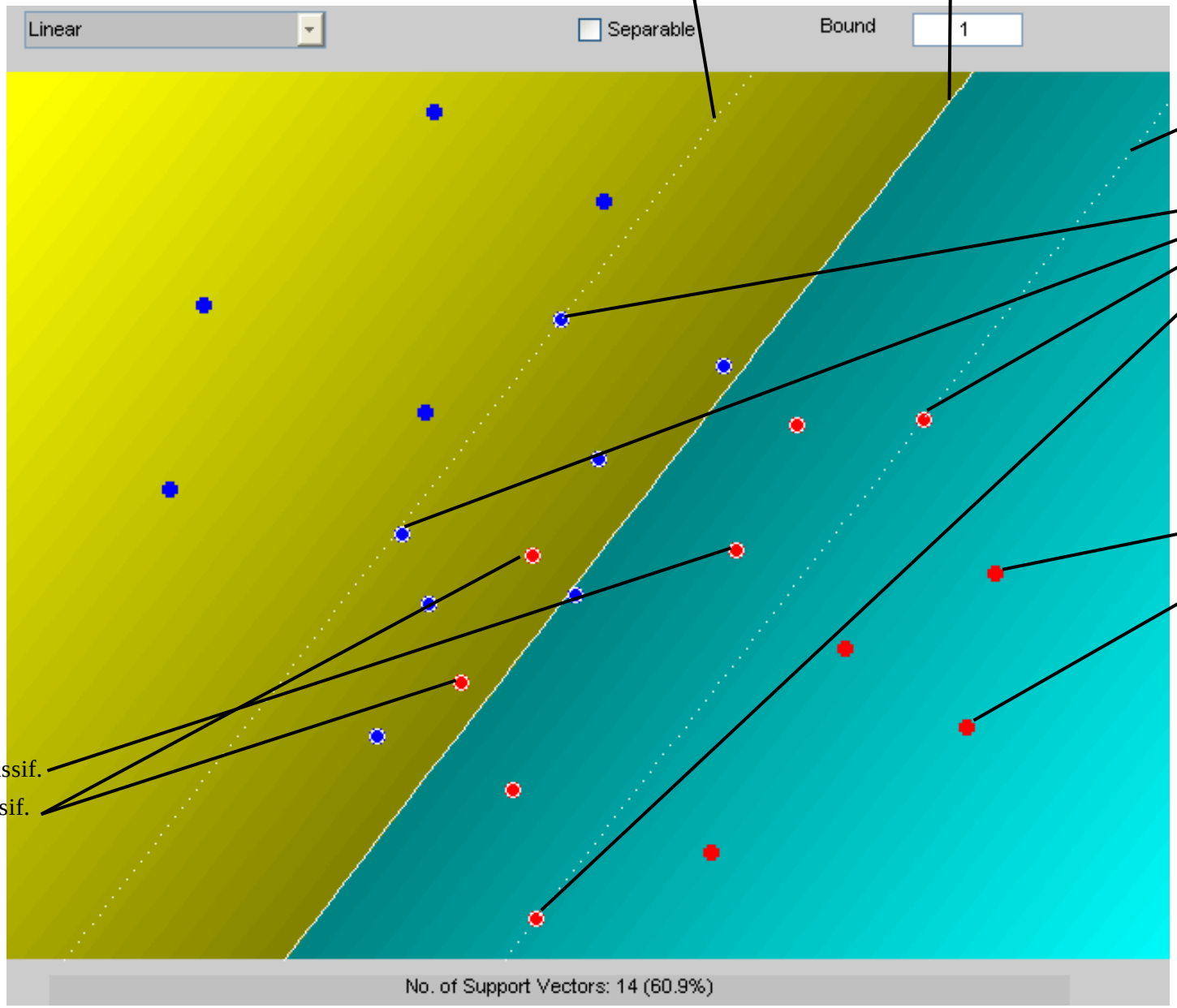
- Lösung nicht existent mit bisherigem Ansatz mit hartem Rand
- Verbesserung der Generalisierung durch größeren margin bei Ausreißern in der Randzone auf Kosten eines etwas erhöhten empirischen Fehlers

Es sind jetzt drei Fälle zu unterscheiden:

- $0<\alpha_i<C$ $\Leftrightarrow$ SV mit $\xi_i = 0$ (liegen auf kanon. Hyperebenen)
- $\alpha_i=C$ $\Leftrightarrow$ SV mit $\xi_i > 0$ ($\xi_i > 1$ => falsche Zuordnung)
- $\alpha_i=0$ $\Leftrightarrow$ für die restlichen Vektoren $\mathbf{x}_i$

$C\rightarrow\infty$ entspricht separierbarem Ansatz und hartem Rand!

“Soft-Margin“ SVM



$\langle \mathbf{w}, \mathbf{x} \rangle + b = -1$

$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$

$\langle \mathbf{w}, \mathbf{x} \rangle + b = +1$

$0 < \alpha_i < C$

$\xi_i = 0$

$\alpha_i = 0$

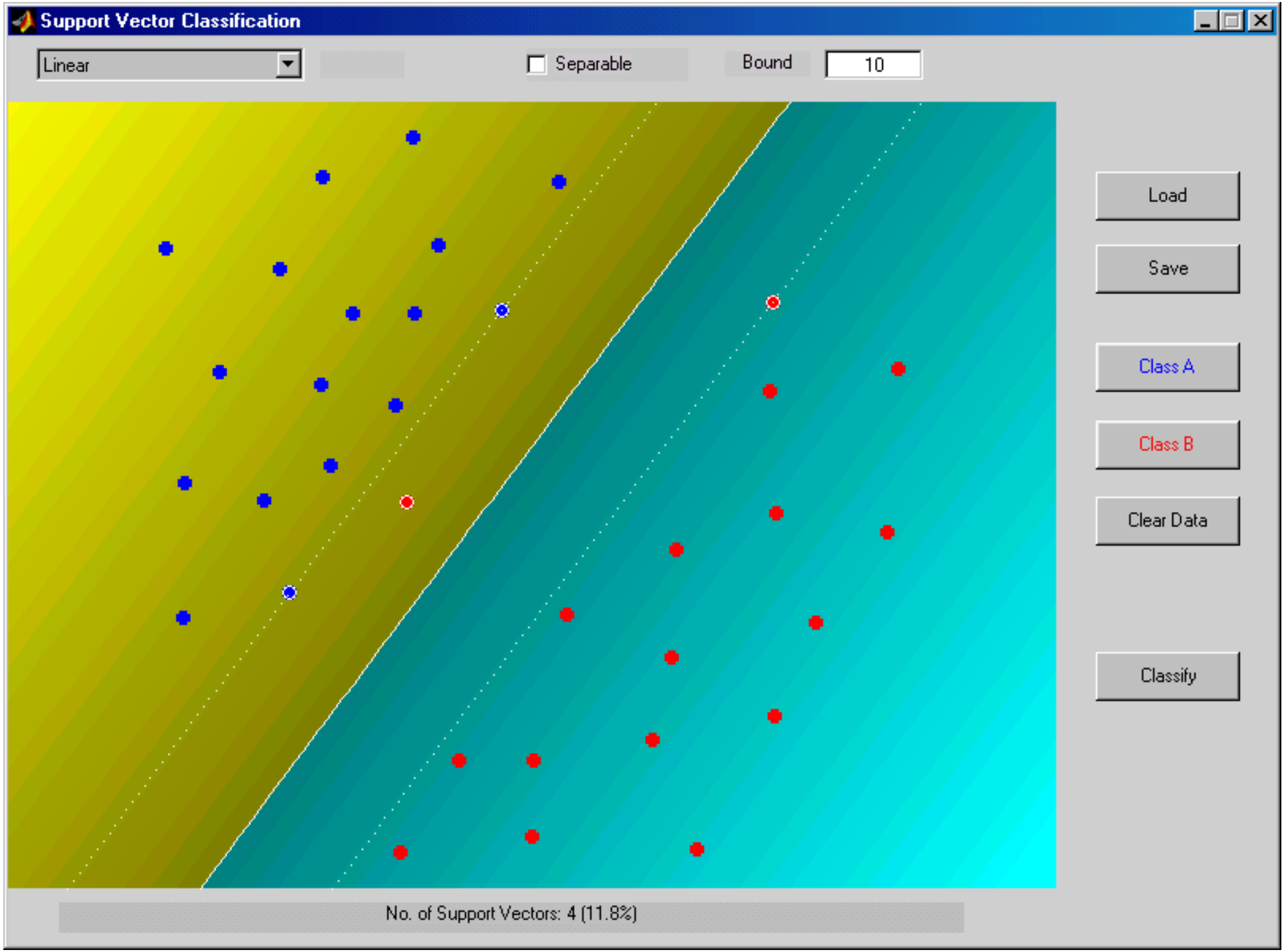
$\xi_i = 0$

$\alpha_i = C$

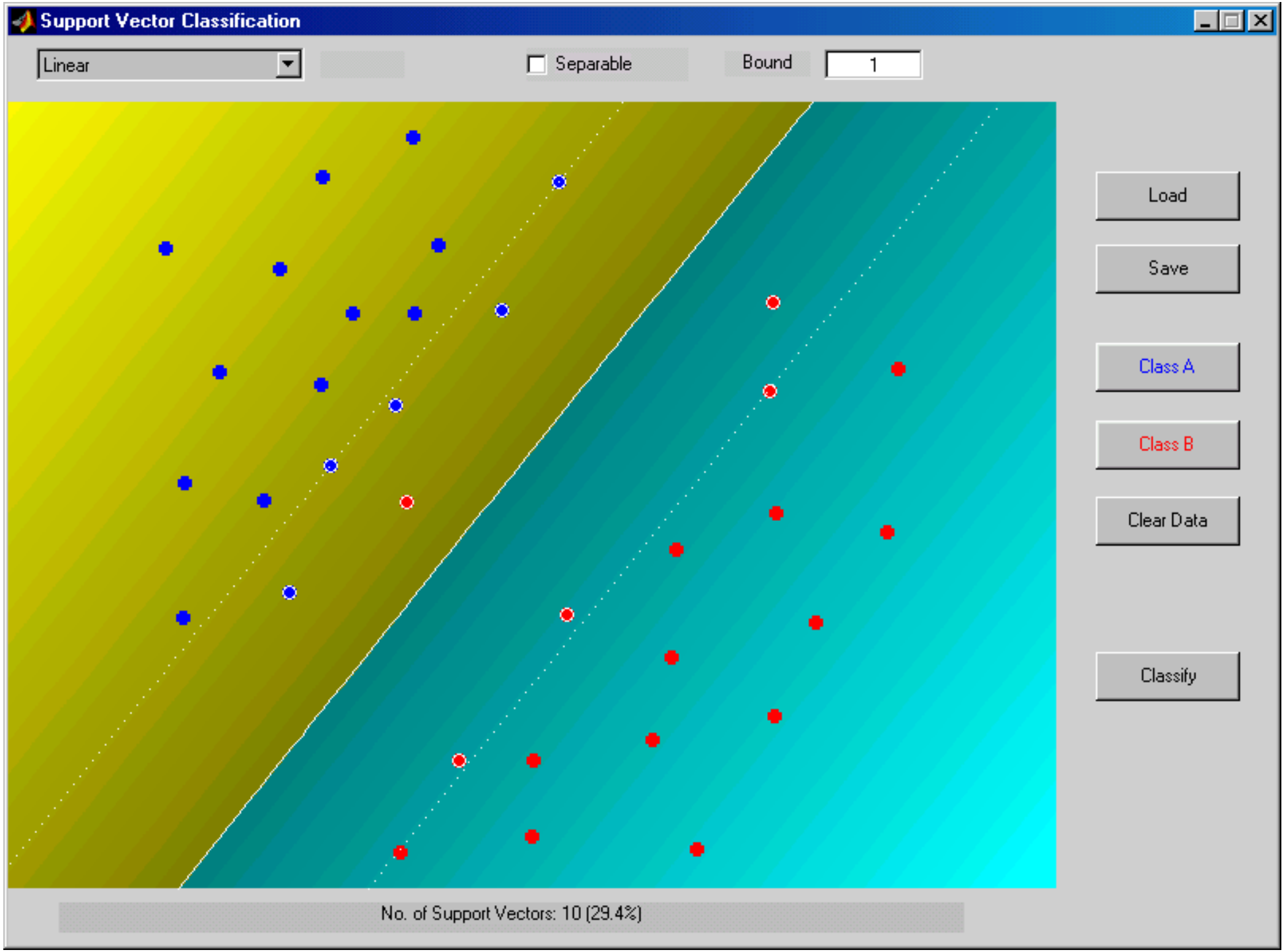
$\xi_i < 1$ korrekt klassif.

$\xi_i > 1$ falsch klassif.

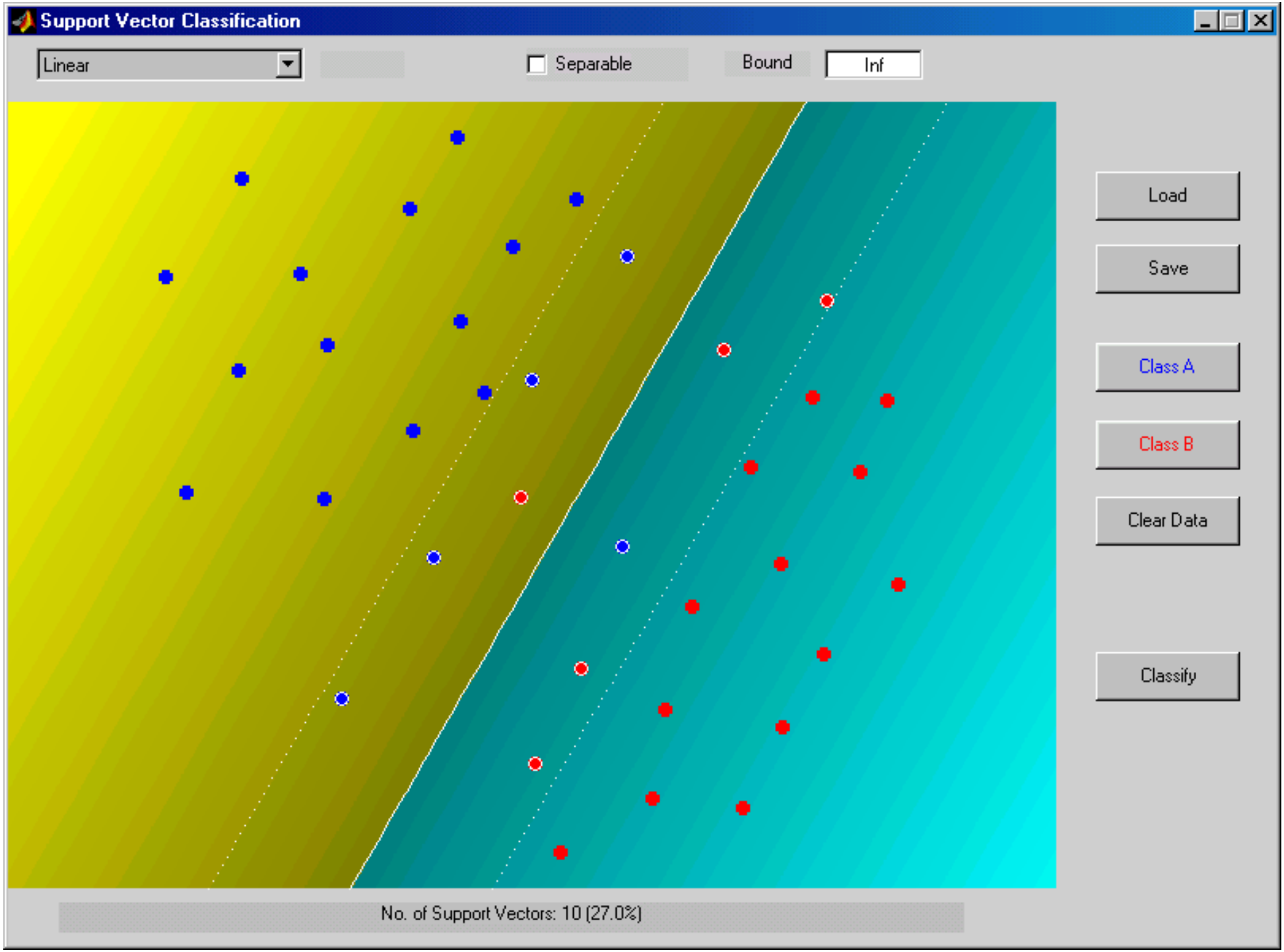
Linear sep. Klassen; weicher, breiter Rand => R_{emp} größer, aber gute Generalisierung



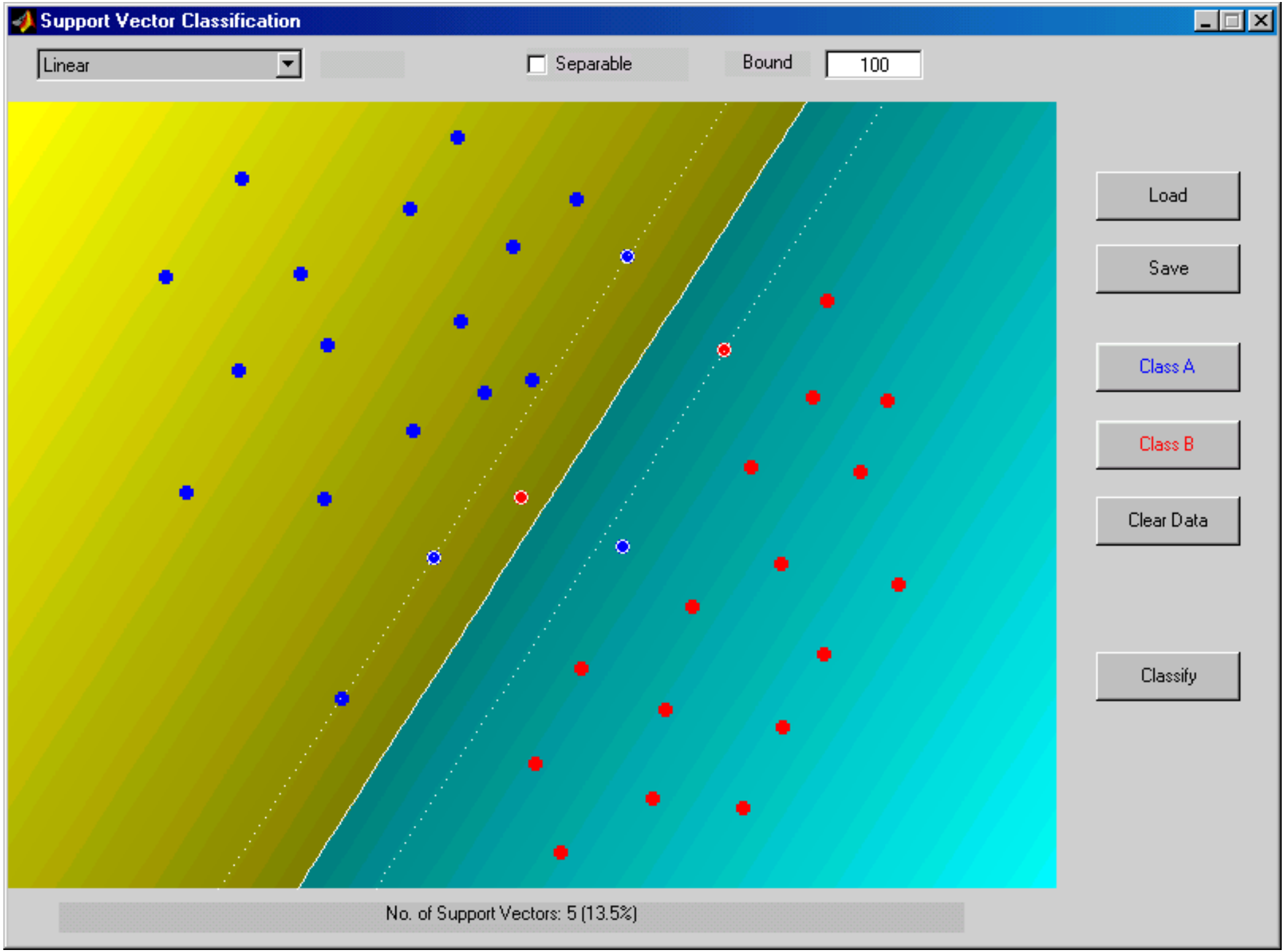
Linear sep. Klassen; weicher, breiter Rand => R_{emp} größer, gute Generalisierung



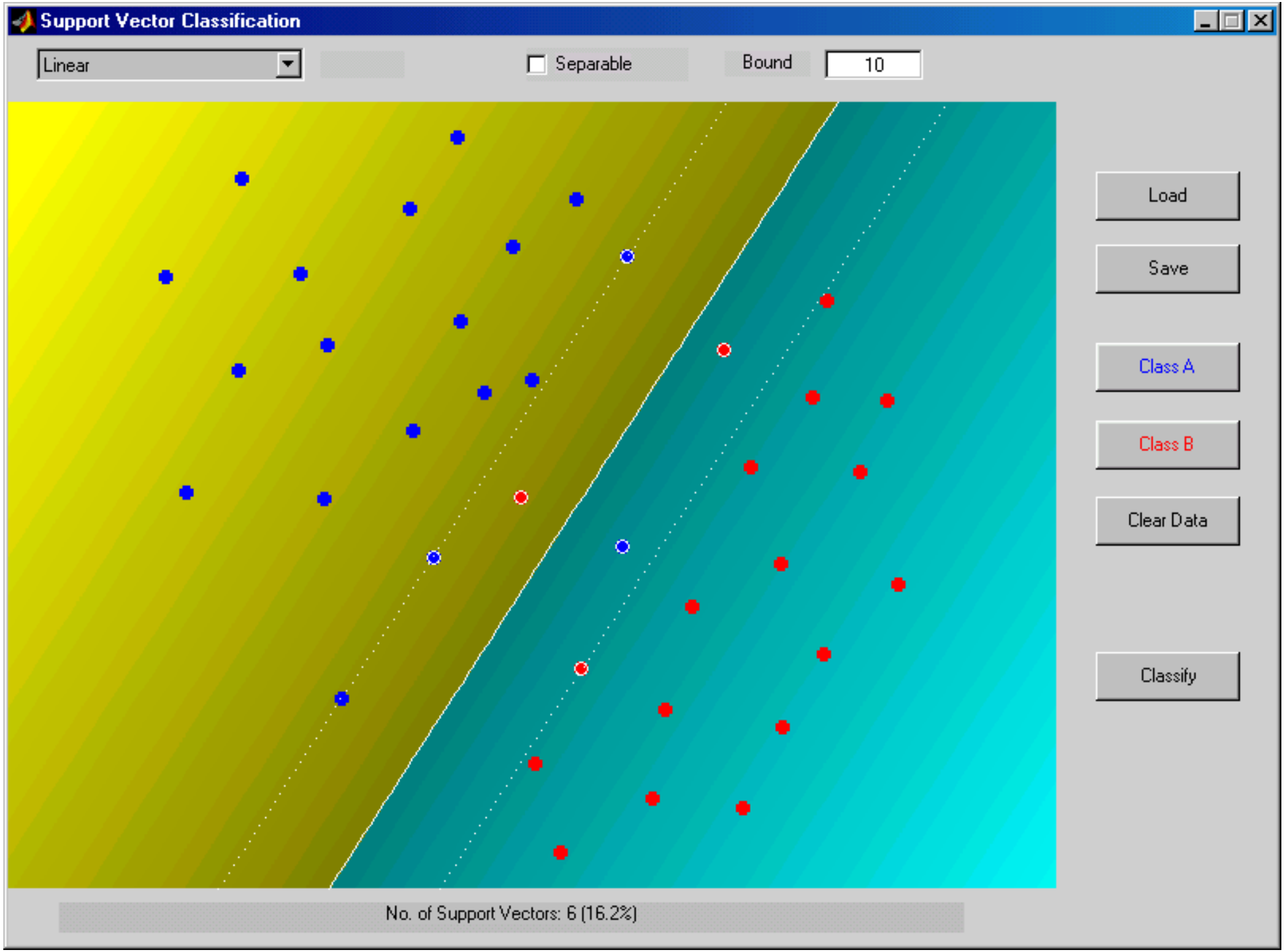
Nicht linear separierbare Klassen; harter Rand



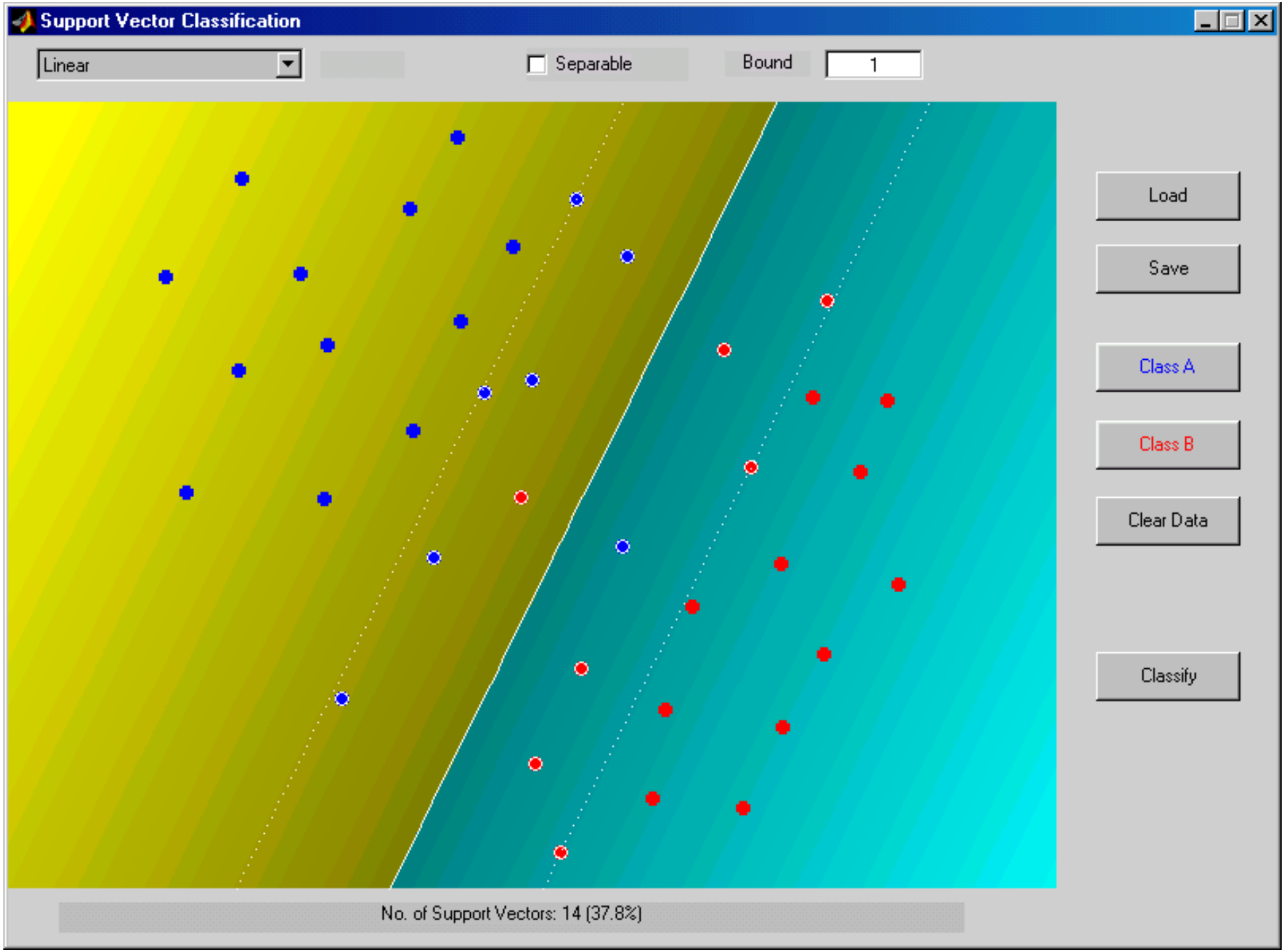
Nicht linear separierbare Klassen; breiter Rand



Linear separiebare Klassen; breiter Rand

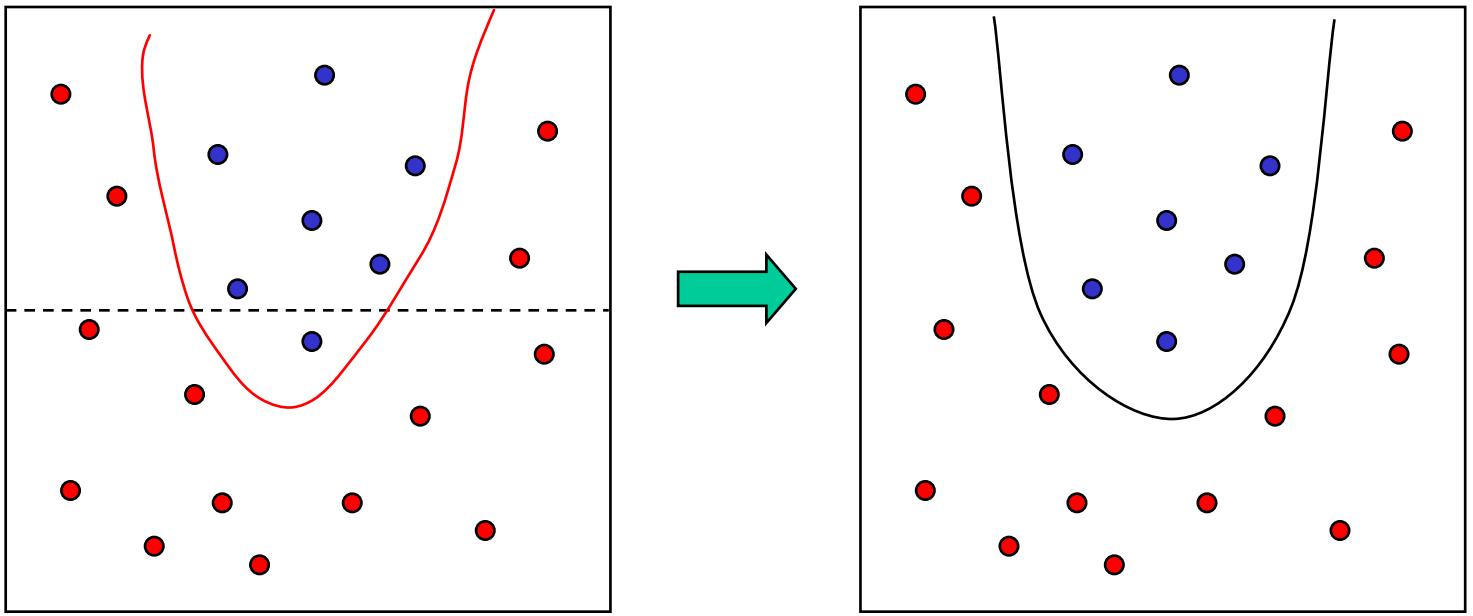


Nicht linear separiebare Klassen; breiter Rand



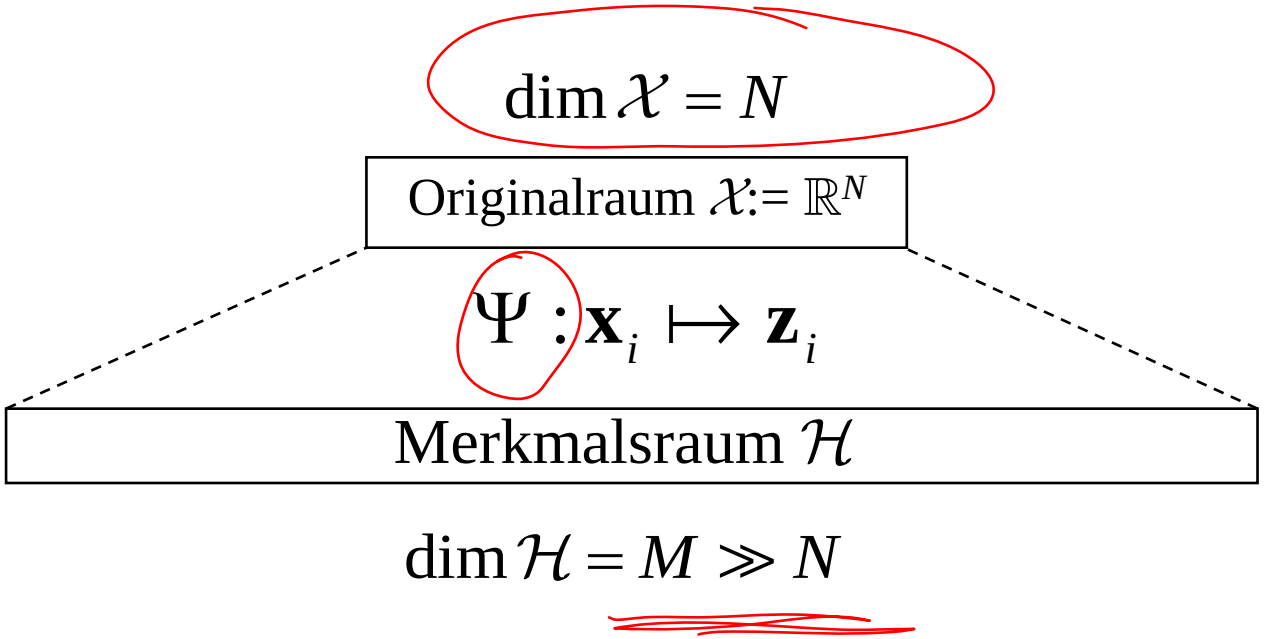
Nichtlineare Probleme

- manche Probleme haben nichtlineare Klassengrenzen
- Hyperebenen erreichen keine zufriedenstellende Genauigkeit (VC-Dimension zu klein)



Erweiterung des Hypothesenraumes

Idee: Finde Hyperebene im höherdimensionalen Merkmalsraum

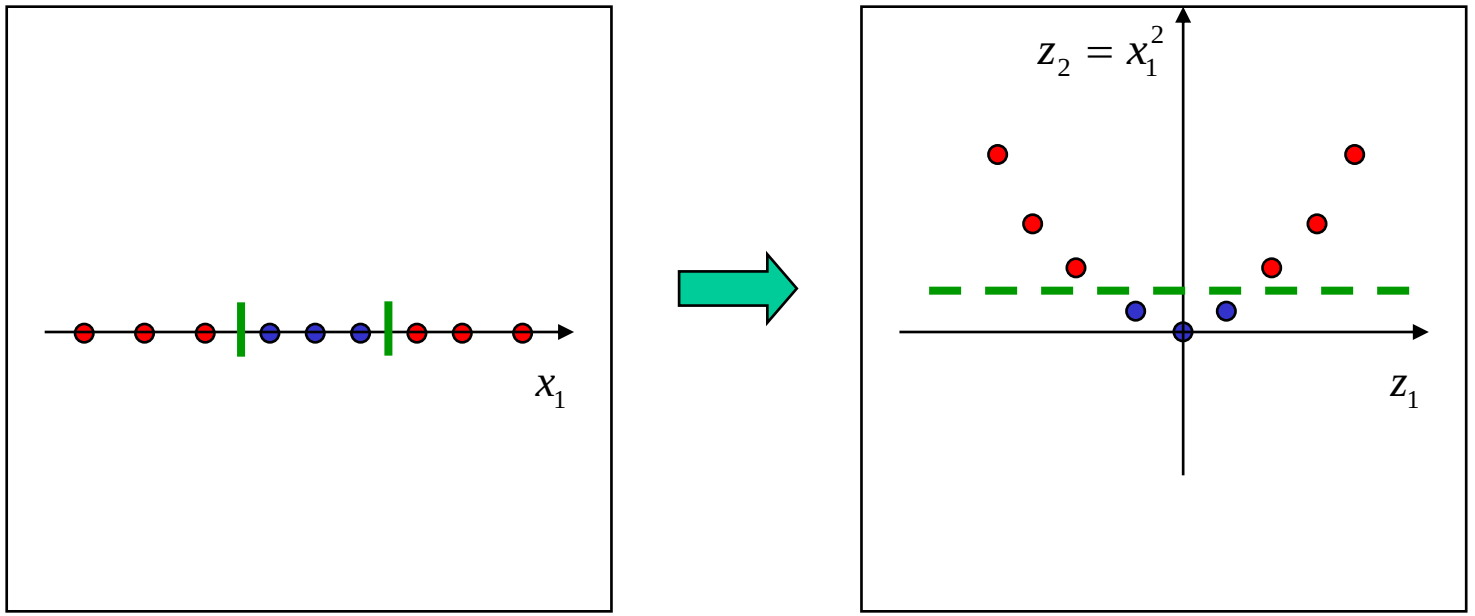


Die trennende Hyperebene im Merkmalsraum $\mathcal{H}$ ist eine nichtlineare Trennfläche im Originalraum (siehe XOR-Problem mit Polynom-klassifikator)

Nichtlineare Probleme

- Eindimensionaler Originalraum: x_1
- Zweidimensionaler Merkmalsraum:

$$\Psi(x_1) = \mathbf{z}^T = \left[z_1 = x_1, z_2 = x_1^2 \right]^T$$



linear nicht separierbar

lineare Separation

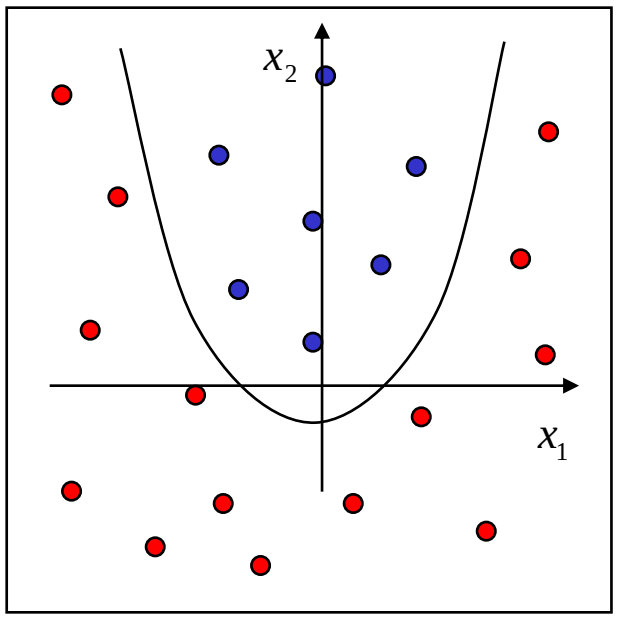
Nichtlineare Probleme

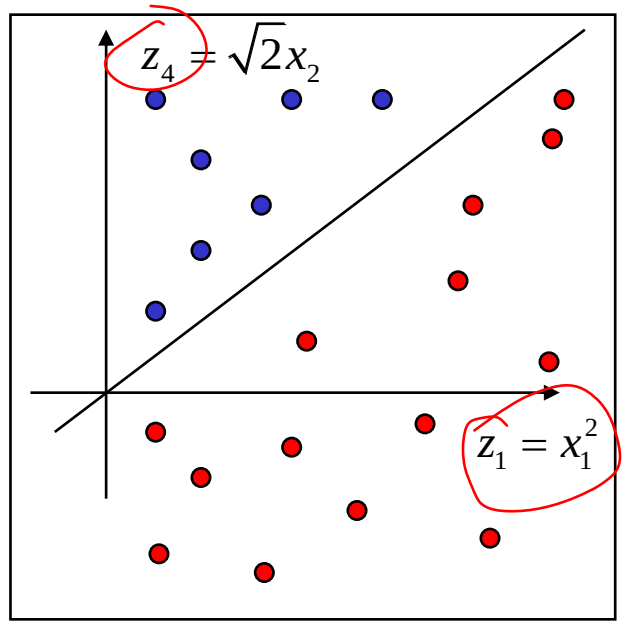
- Originalraum:

  $\mathbf{x}=(x_1,x_2)$ (zweidimensional)

- Merkmalsraum:

$$\Psi(\mathbf{x}) = \mathbf{z}^T = \left[z_1 = x_1^2, z_2 = x_2^2, z_3 = \sqrt{2}x_1, z_4 = \sqrt{2}x_2, z_5 = \sqrt{2}x_1x_2, z_6 = 1 \right]^T$$





- $x_2 > x_1^2$
- $x_2 < x_1^2$

nichtlineare Separation

- $z_4 > \sqrt{2}z_1$
- $z_4 < \sqrt{2}z_1$

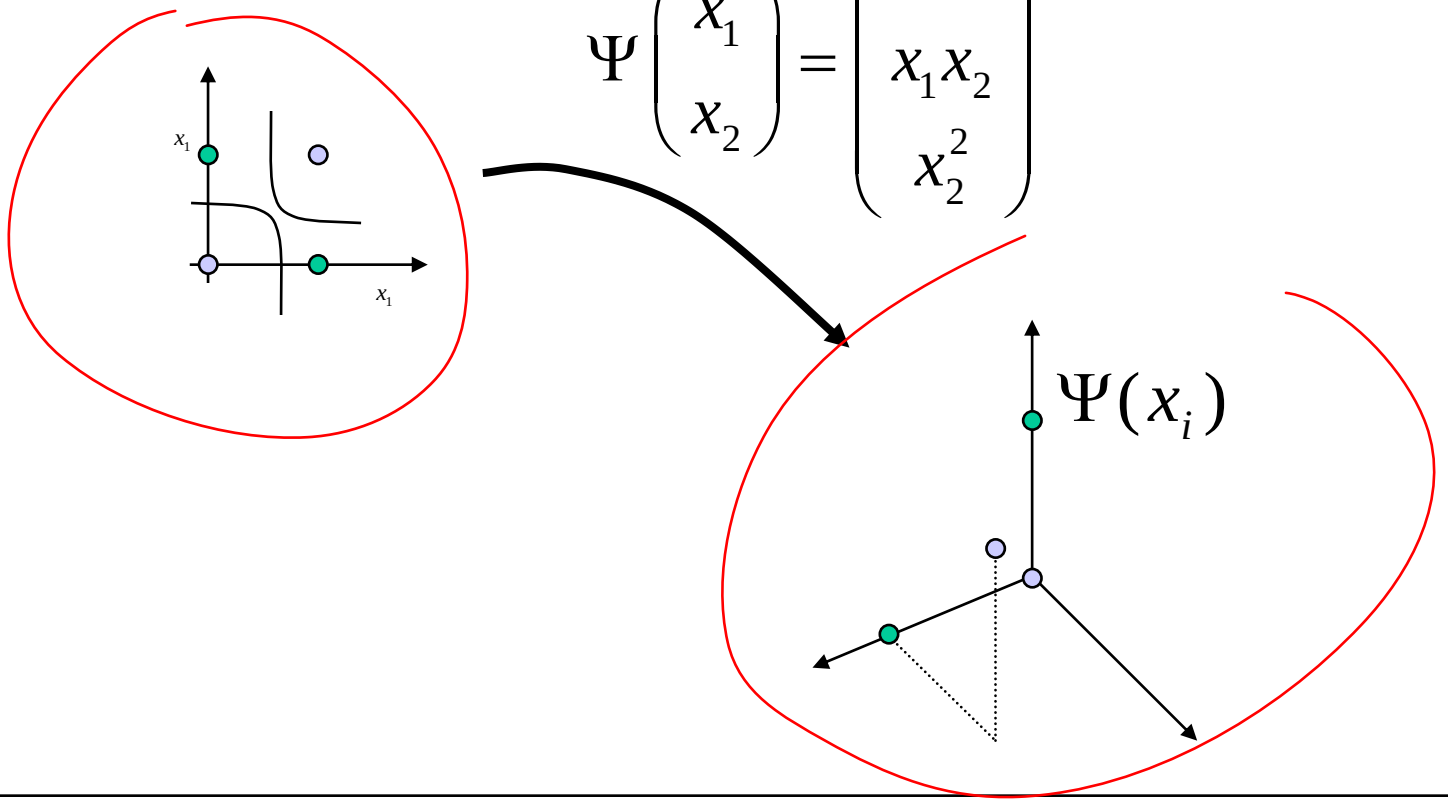
lineare Separation

Nichtlineare Erweiterung

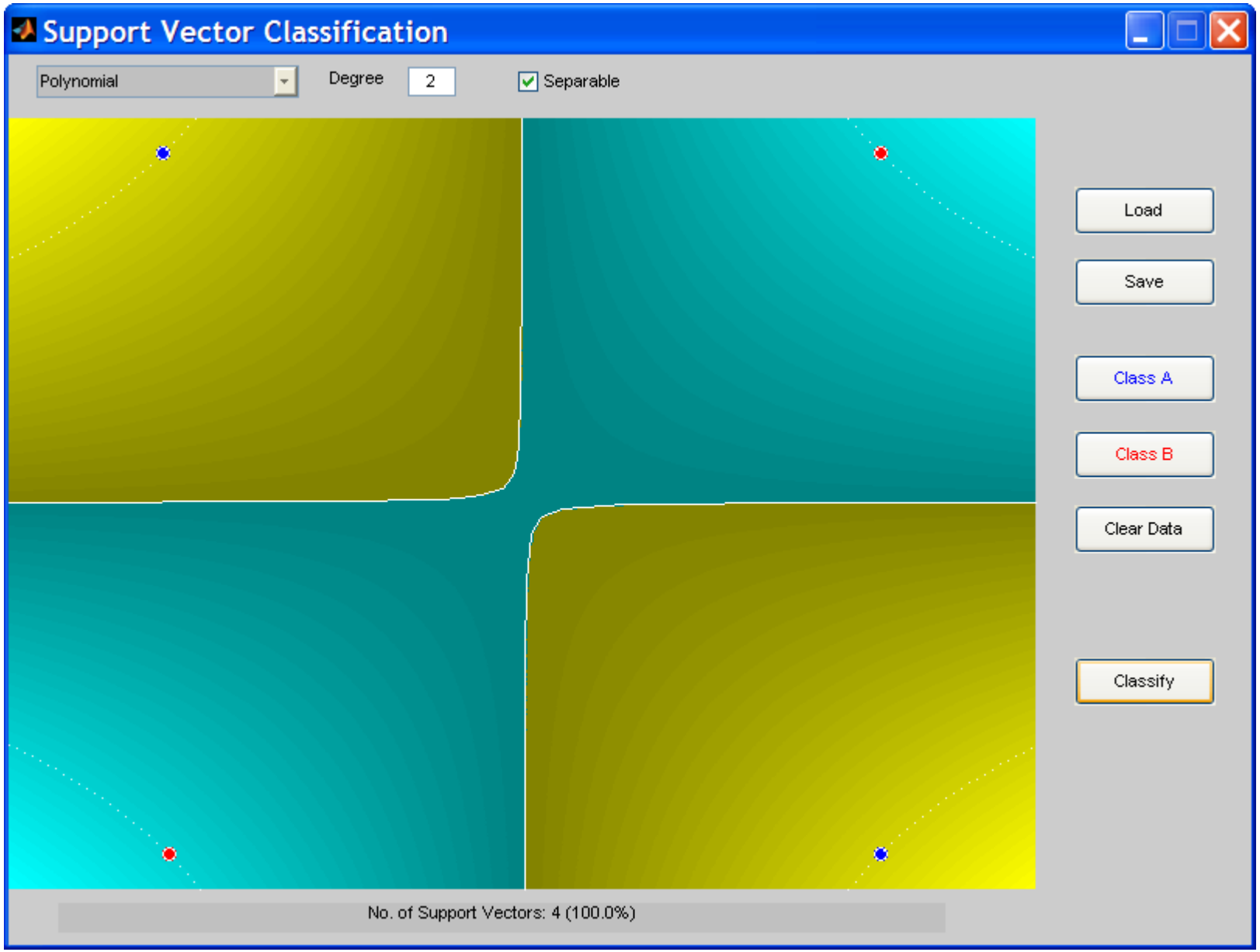
Nichtlineare Abbildung vorschalten $\Psi(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathcal{H}$

- Beispiel XOR-Problem:

$$\Psi \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix}$$



Lösung des XOR-Problems



Konsequenzen

- Effekt:

  Steigerung der Separabilität

  Trennfläche im Ursprungsraum nichtlinear

- Fragen:

  1. Optimalität der Hyperebene?

  2. Hoher Rechenaufwand in hochdimensionalen Räumen?

- Zu 1: Optimalität bleibt erhalten, erneut positiv semidefinite Form, da in der zu optimierenden Funktion die gleichen Skalarprodukte auftauchen, nur in einem neuen Raum $\mathcal{H}$
- Zu 2: der hohe Aufwand in den hochdimensionalen Merkmalsraum $\mathcal{H}$ wird durch den Trick mit Kernfunktionen reduziert. Das Innenprodukt in H hat einen äquivalente Formulierung mit Kernfunktionen im Originalraum $\mathcal{X}$

Der Trick mit Kernfunktionen

Problem: Sehr hohe Dimension des Merkmalraumes! Polynome p-ten Grades über der Dimension N=dim($\mathbf{x}$) des Originalraums führen zu O(M=N^p) Dimensionen im Merkmalsraum!

Lösung: Im dualen OP tauchen nur Skalarprodukte $<\mathbf{x}_i,\mathbf{x}_j>$ auf. Im korrespondierenden Problem im Merkmalsraum tauchen dann ebenfalls nur Skalarprodukte in $< \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j) >$ auf. Diese müssen nicht explizit ausgerechnet werden, sondern können mit reduzierter Komplexität mit Kernfunktionen im Originalraum $\mathcal{X}$ ausgedrückt werden:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j) \rangle = \langle \mathbf{z}_i, \mathbf{z}_j \rangle$$

Beispiel für $N = 2$:

Für $\Psi(\mathbf{x}) = \mathbf{z}^T = \left[z_1 = x_1^2, z_2 = x_2^2, z_3 = \sqrt{2}x_1, z_4 = \sqrt{2}x_2, z_5 = \sqrt{2}x_1x_2, z_6 = 1 \right]^T$ $M = 6$

berechnet $K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^2 = \langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j) \rangle$ $M = N = 2$

das Skalarprodukt im Merkmalsraum.

Häufig verwendete Kernfunktionen

Polynom-Kerne	$K(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^2$
Gauss-Kerne	$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / (2\sigma^2)\right)$
Sigmoid-Kerne	$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta)$

Die resultierenden Klassifikatoren sind vergleichbar mit Polynomklassifikatoren, radialen Basisfunkten und mit Neuronalen Netzen (sie werden allerdings anders motiviert).

Allgemeine Anforderung: Mercer´s Bedingung. Sie garantiert, dass eine bestimmte Kernfunktion tatsächlich auch ein Skalarprodukt in irgendeinem Raum ist, aber sie erklärt nicht wie das dazugehörige Abbildung Ψ aussieht und wie der Raum $\mathcal{H}$ beschaffen ist.

Ausserdem: Linearkombinationen von gültigen Kernen liefern neue Kerne (die Summe 2er pos. def. Fkten ist wieder pos. def.)

Das Theorem von Mercer

Es existiert eine Abbildung Ψ und eine Entwicklung .

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}_j) \rangle$$

genau dann, wenn für ein *beliebiges* quadratisch integrierb. $g(\mathbf{x})$ mit

$$\int g(\mathbf{x})^2 \, d\mathbf{x} < \infty$$

gilt, dass K eine symmetrische, positiv semidefinite Funktion in $\mathbf{x}$ ist:

$$\int K(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_i) g(\mathbf{x}_j) d\mathbf{x}_i d\mathbf{x}_j \geq 0$$

Es gibt allerdings auch Fälle, wo Kernfunktionen die Mercer-Bedingung nicht erfüllen, aber für einen *bestimmten Trainingsdatensatz* zu einer positiv semidefiniten Hesse-Matrix führen und damit zu einem globalen Optimum konvergieren.

Endformulierung

– Trainieren:

maximiere: $L(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$

mit: $\sum_{i=1}^{l} y_i \alpha_i = 0$ und $0 \le \alpha_i \le C$ (harter Rand: $C = \infty$)

C wägt ab zwischen emp. Fehler und Generalisierung

– Klassifizieren eines unbekannten Objekts $\mathbf{x}$ ($\alpha_i \neq 0$ für alle *SV*):

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \Psi(\mathbf{x}_i) = \sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \Psi(\mathbf{x}_i)$$

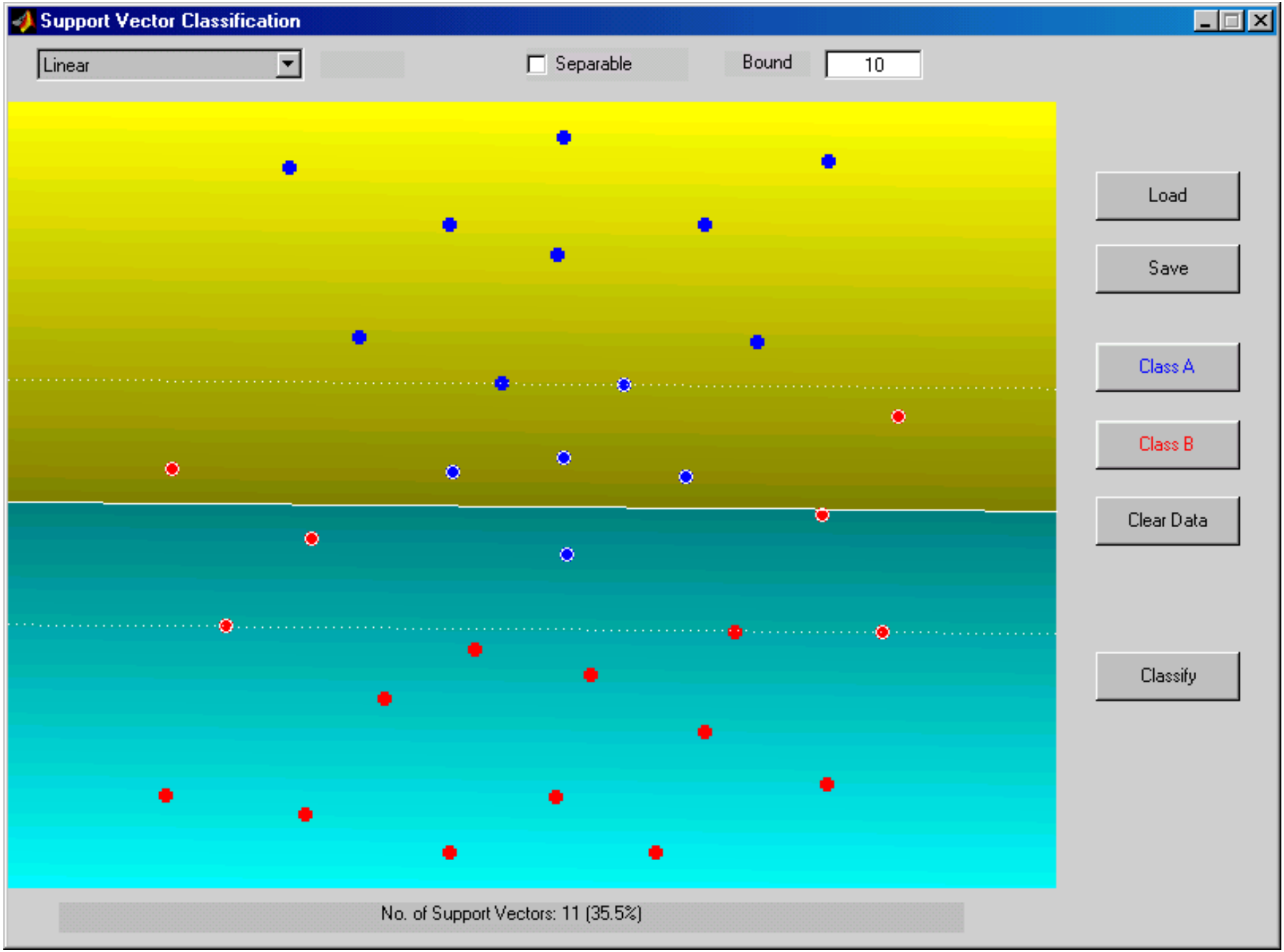
$$b^* = -\frac{1}{2} \left(\max_{i, y_i = -1} \left(\sum_{j=1}^{l} y_j \alpha_j^* K(\mathbf{x}_i, \mathbf{x}_j) \right) + \min_{i, y_i = +1} \left(\sum_{j=1}^{l} y_j \alpha_j^* K(\mathbf{x}_i, \mathbf{x}_j) \right) \right)$$

$$= -\frac{1}{2} \left(\max_{\mathbf{x}_i \in SV, y_i = -1} \left(\sum_{\mathbf{x}_j \in SV} y_j \alpha_j^* K(\mathbf{x}_i, \mathbf{x}_j) \right) + \min_{\mathbf{x}_i \in SV, y_i = +1} \left(\sum_{\mathbf{x}_j \in SV} y_j \alpha_j^* K(\mathbf{x}_i, \mathbf{x}_j) \right) \right)$$

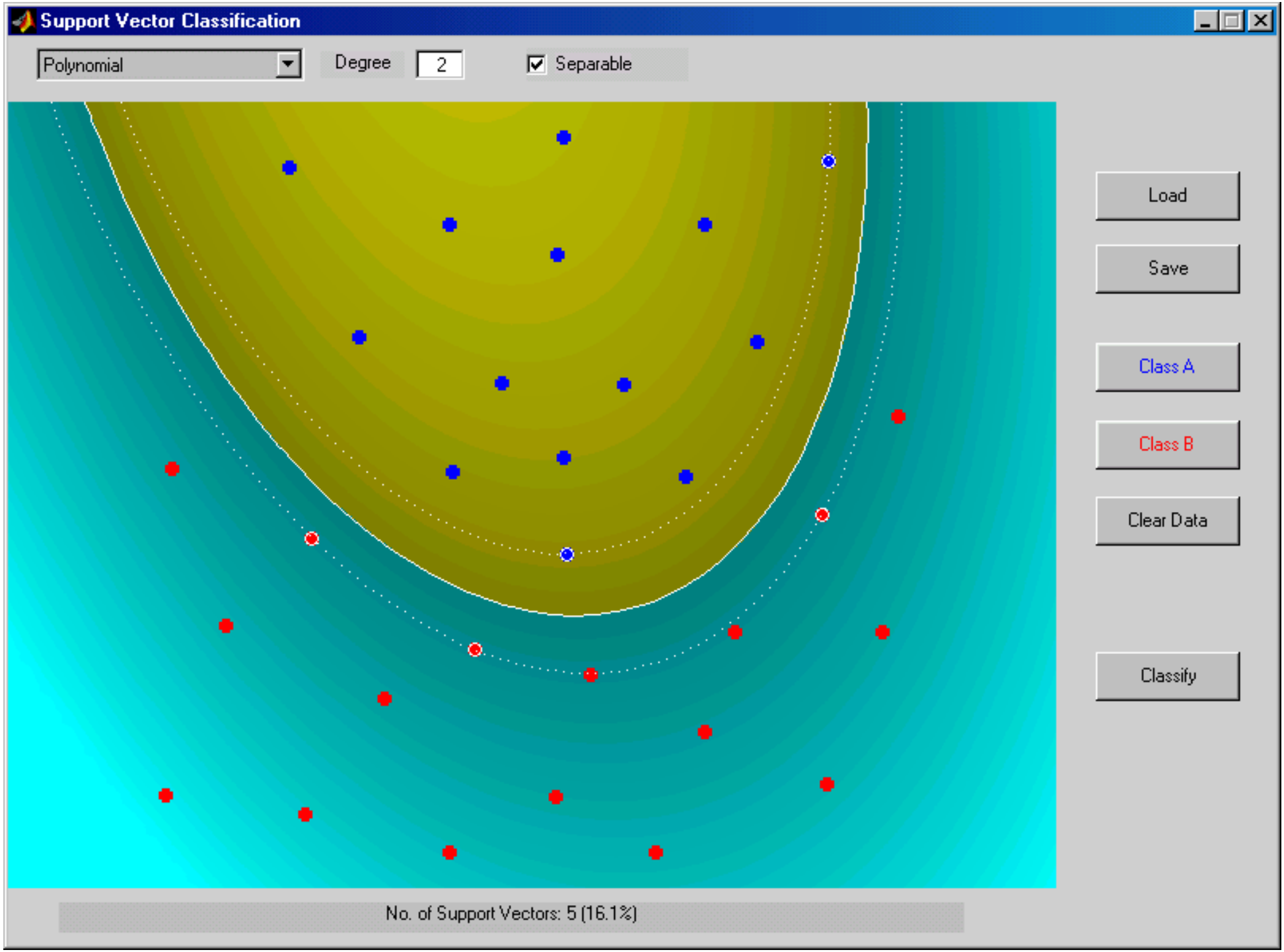
$$f(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}^*, \Psi(\mathbf{x}) \rangle + b^*) = \operatorname{sgn}(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i \langle \Psi(\mathbf{x}_i), \Psi(\mathbf{x}) \rangle + b^*)$$ (<= wird so nicht berechnet!)

$$= \operatorname{sgn} \left(\sum_{i=1}^{l} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^* \right) = \operatorname{sgn} \left(\sum_{\mathbf{x}_i \in SV} \alpha_i^* y_i K(\mathbf{x}_i, \mathbf{x}) + b^* \right)$$

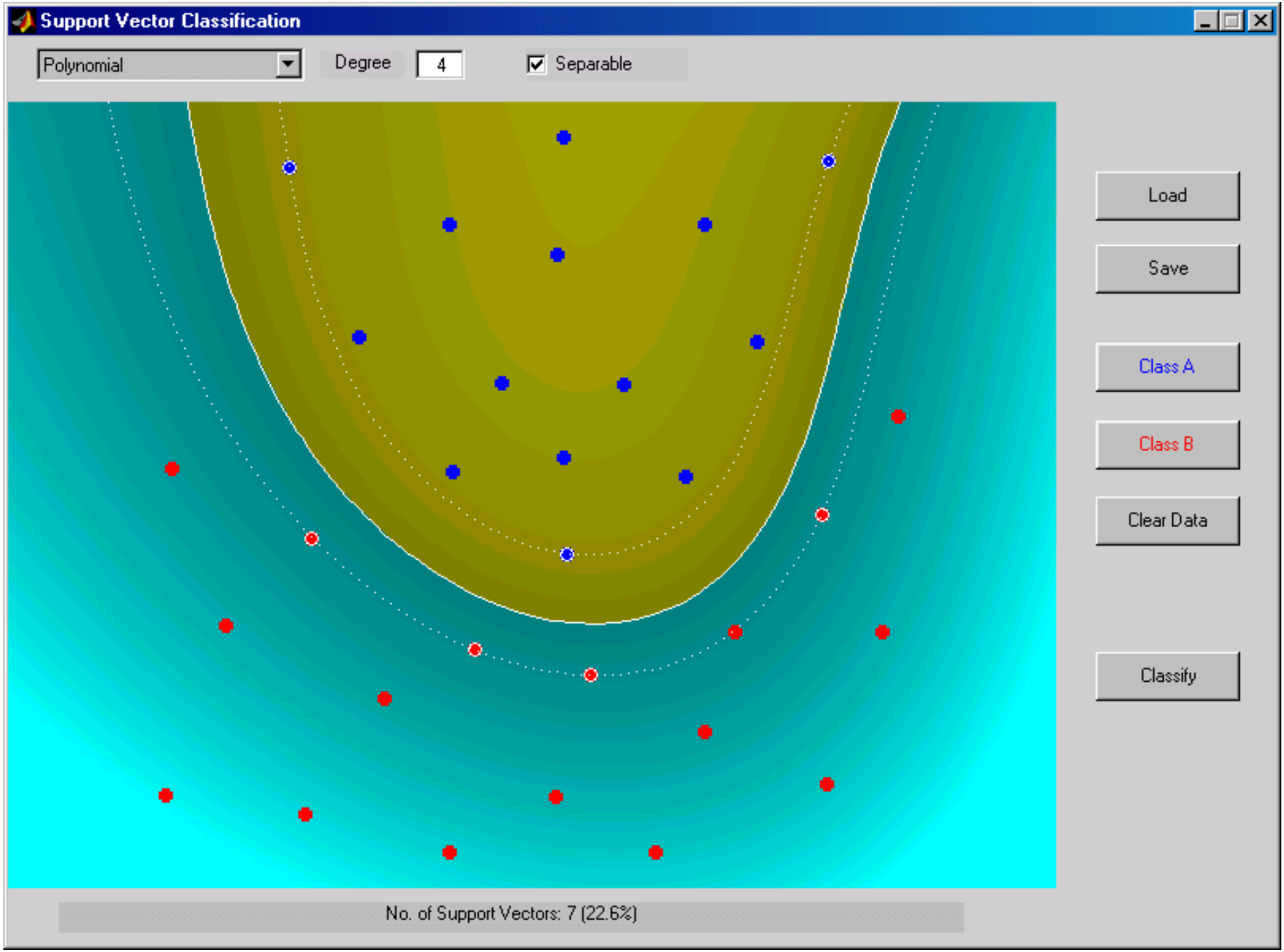
Lineare Separierung eines quadratischen Problems; weicher Rand



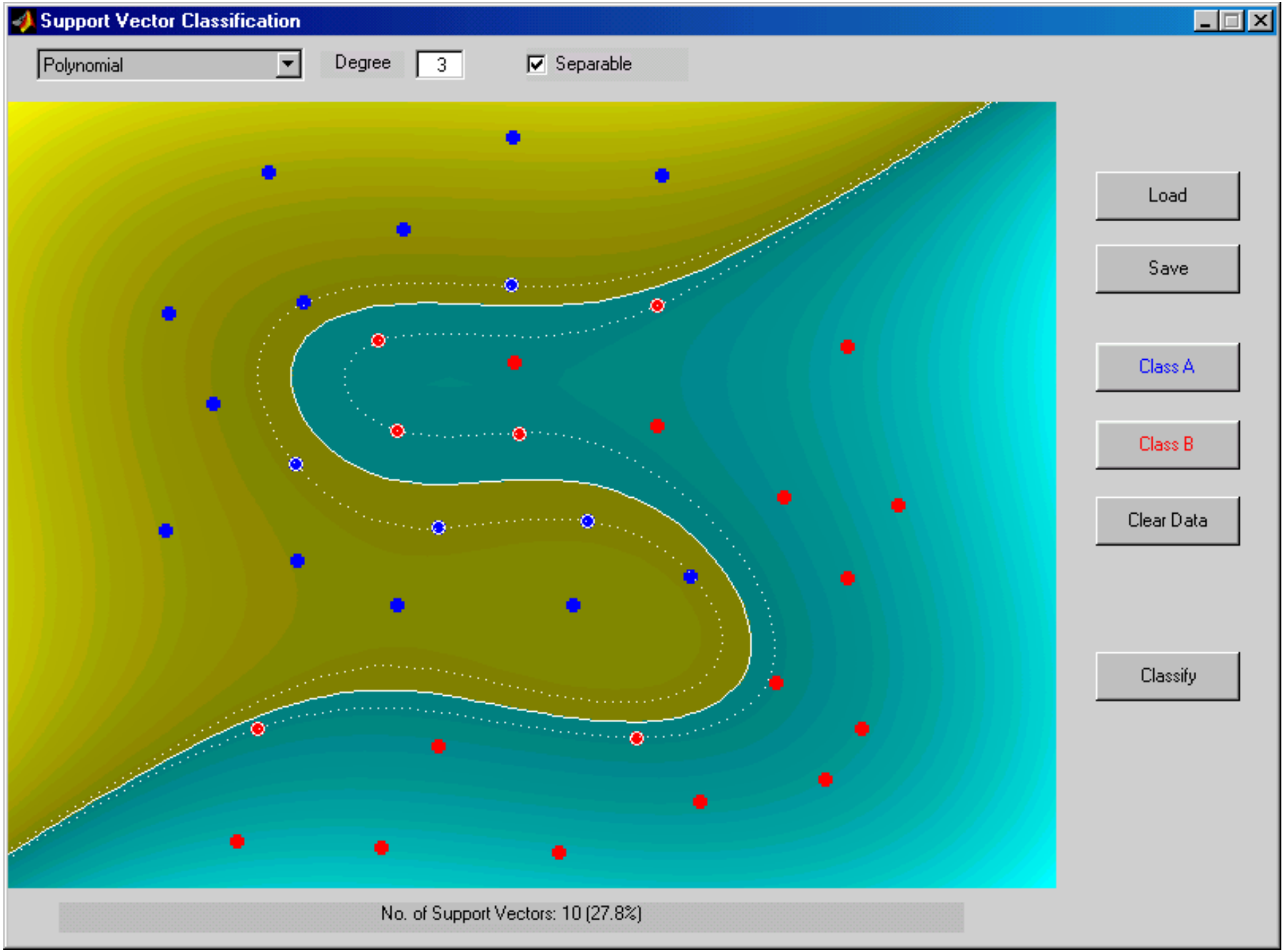
Polynomiale Separierung eines quadratischen Problems; harter Rand



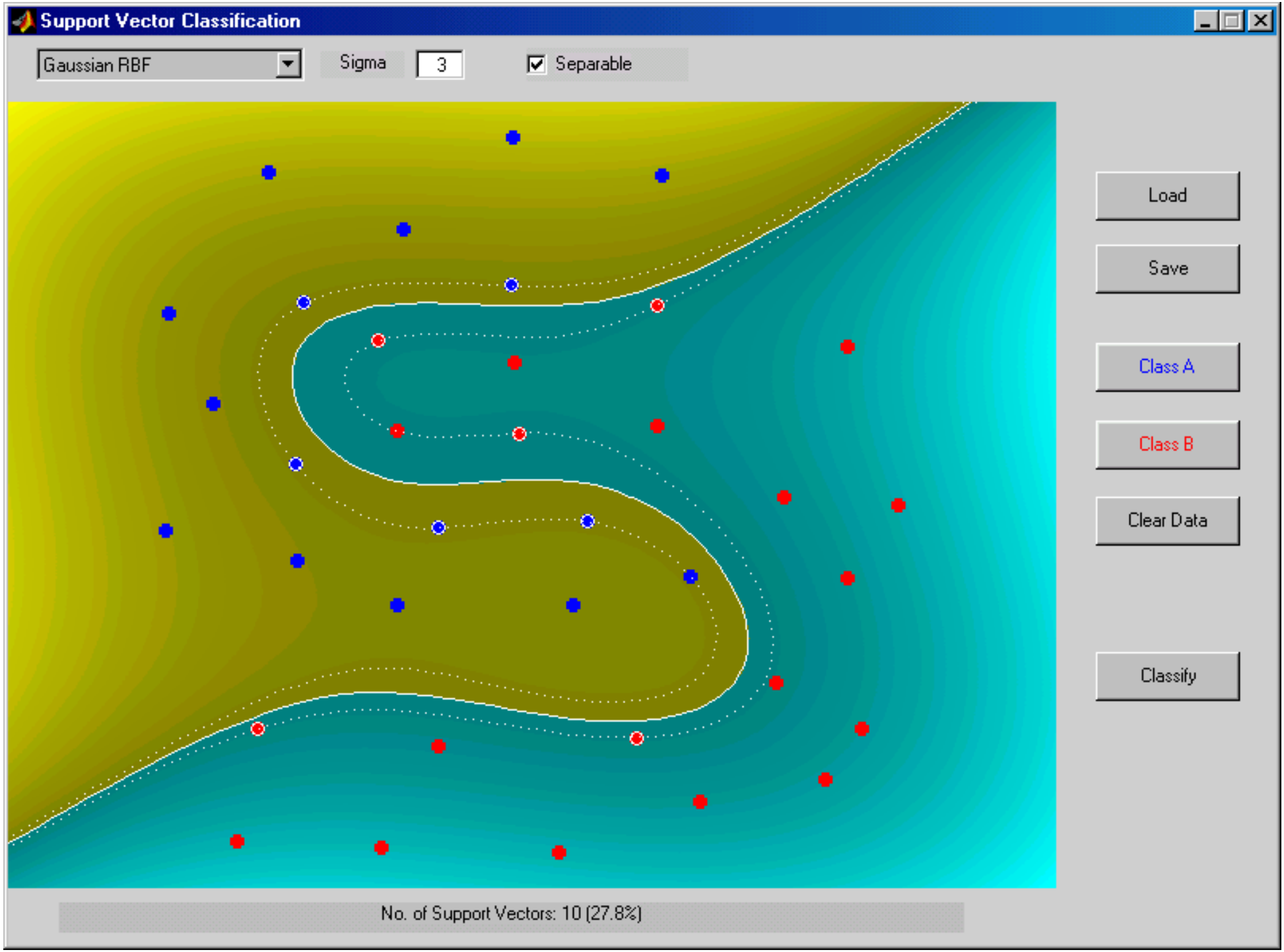
Polynomiale Separierung eines quadratischen Problems; harter Rand



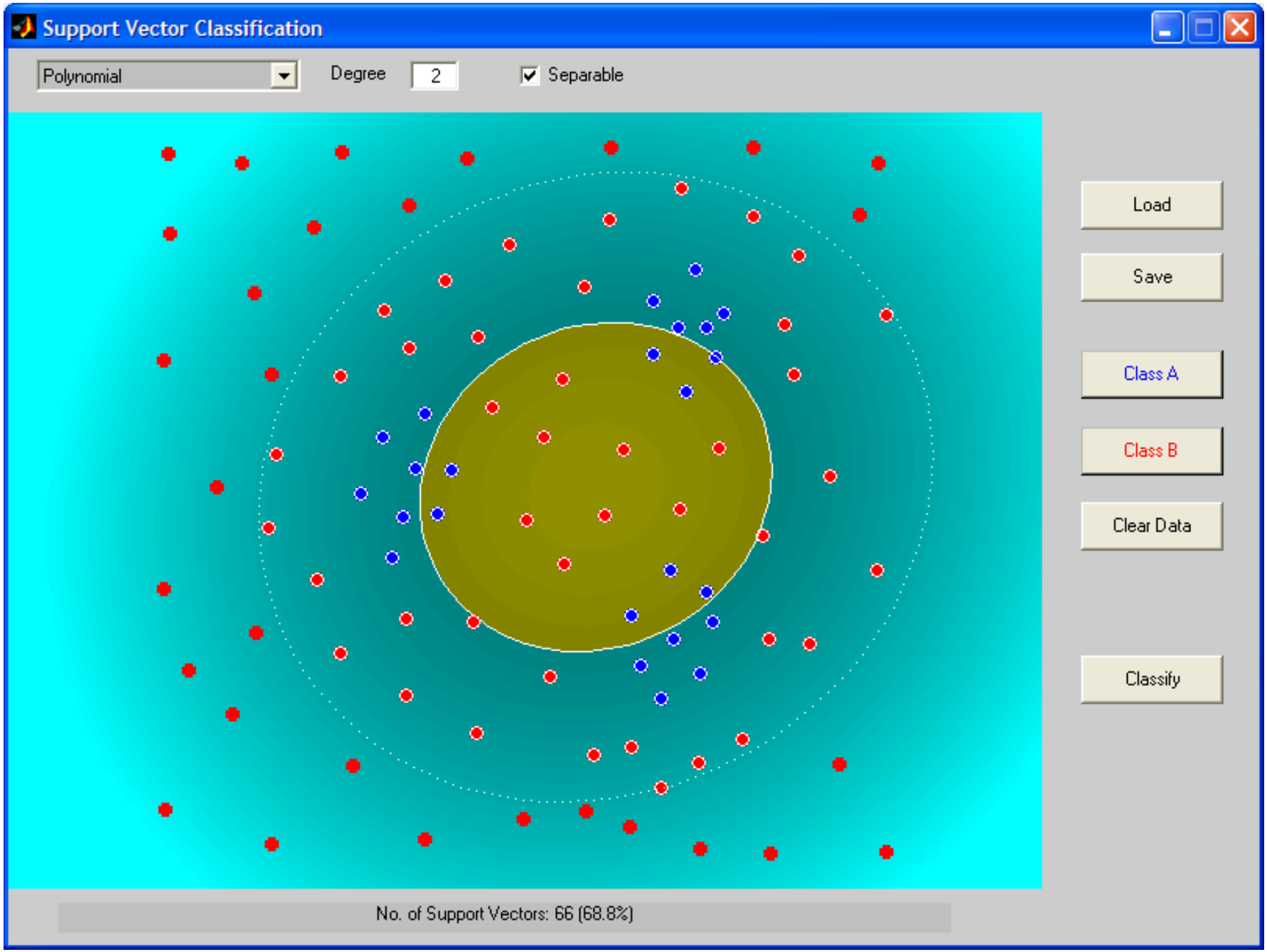
Nichtlinear separierbare Klassen; harter Rand



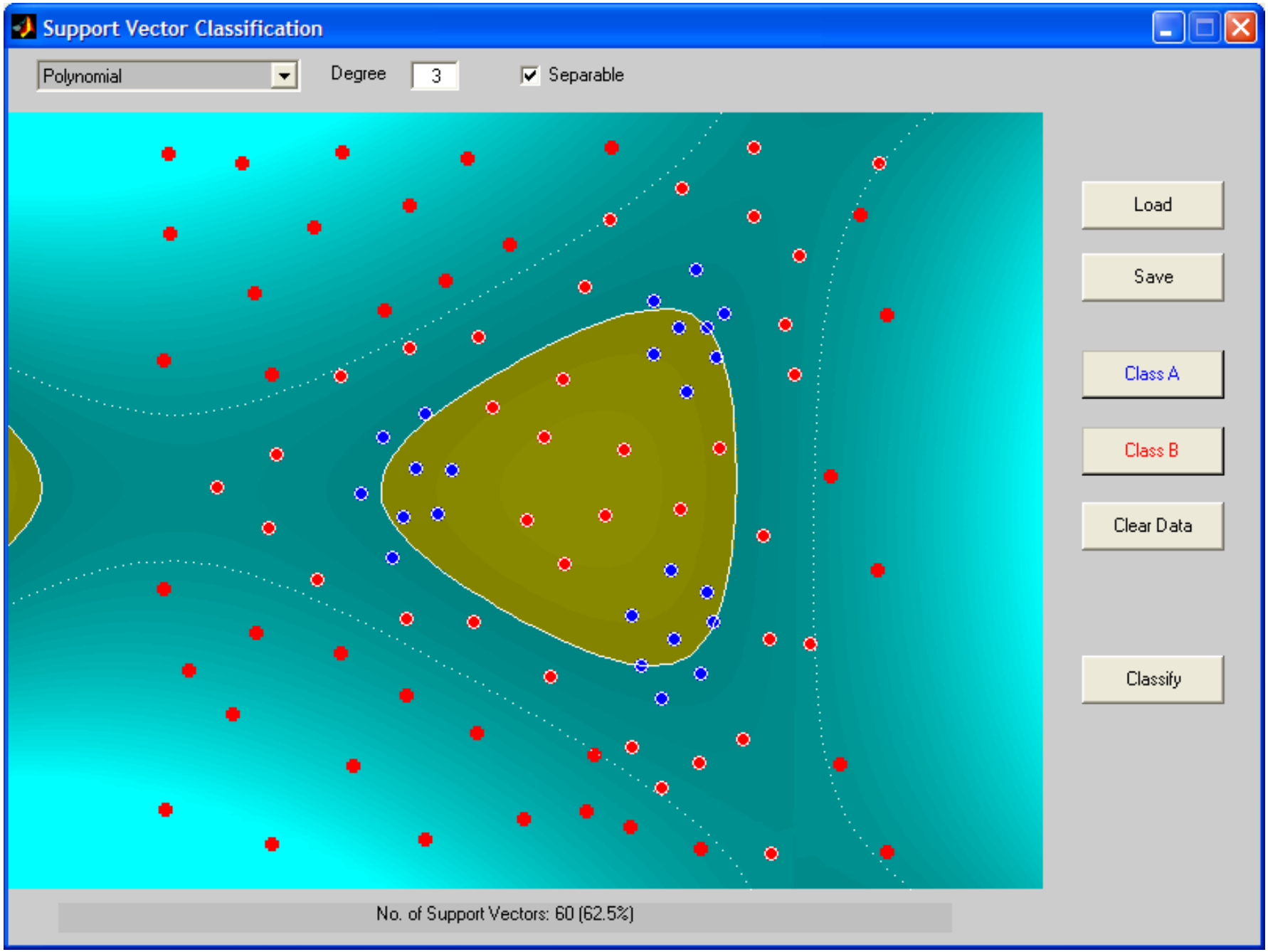
Nichtlinear separierbare Klassen; harter Rand



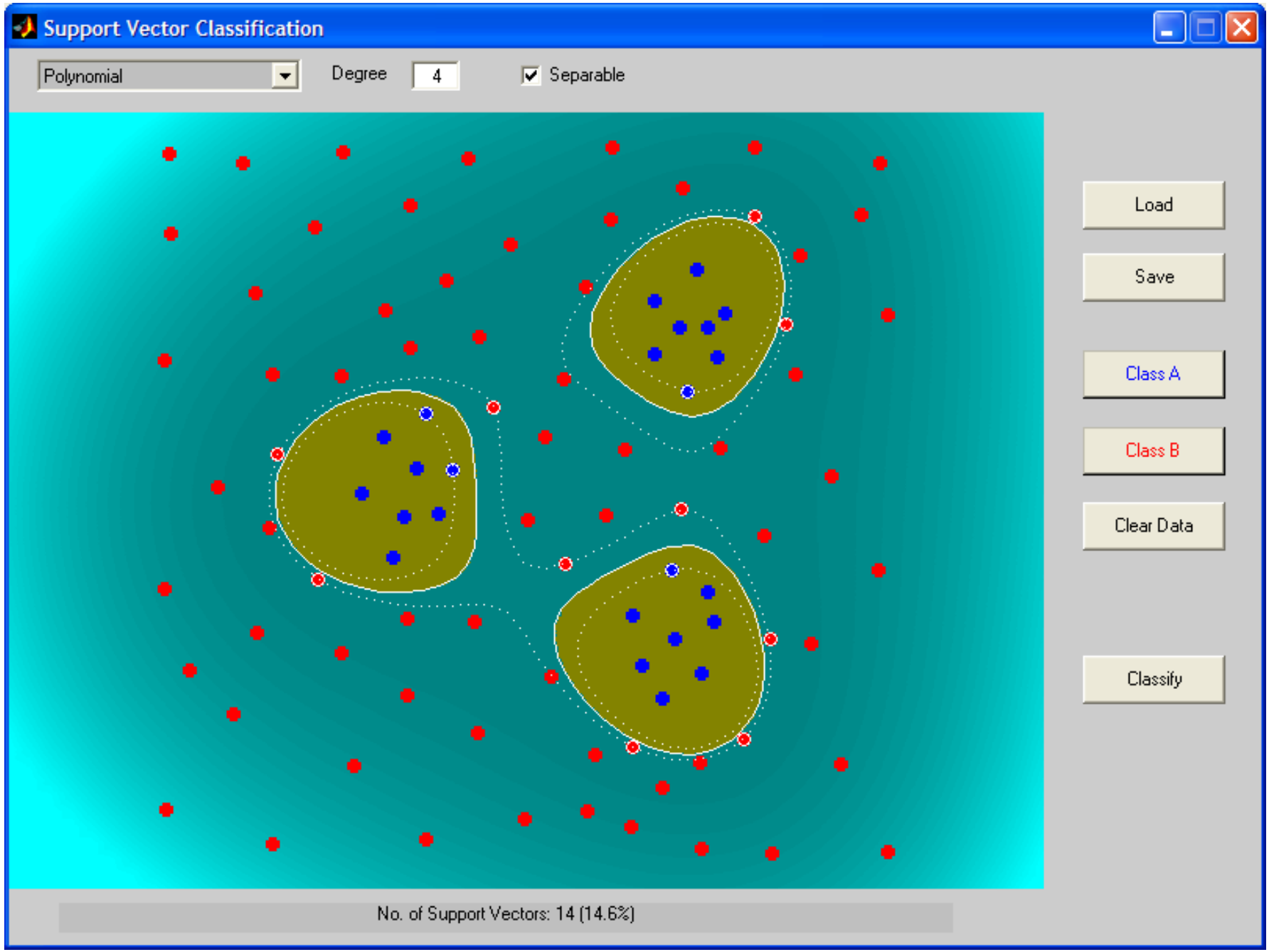
Beispiel: „Inseln“, Polynom mit $p = 2$



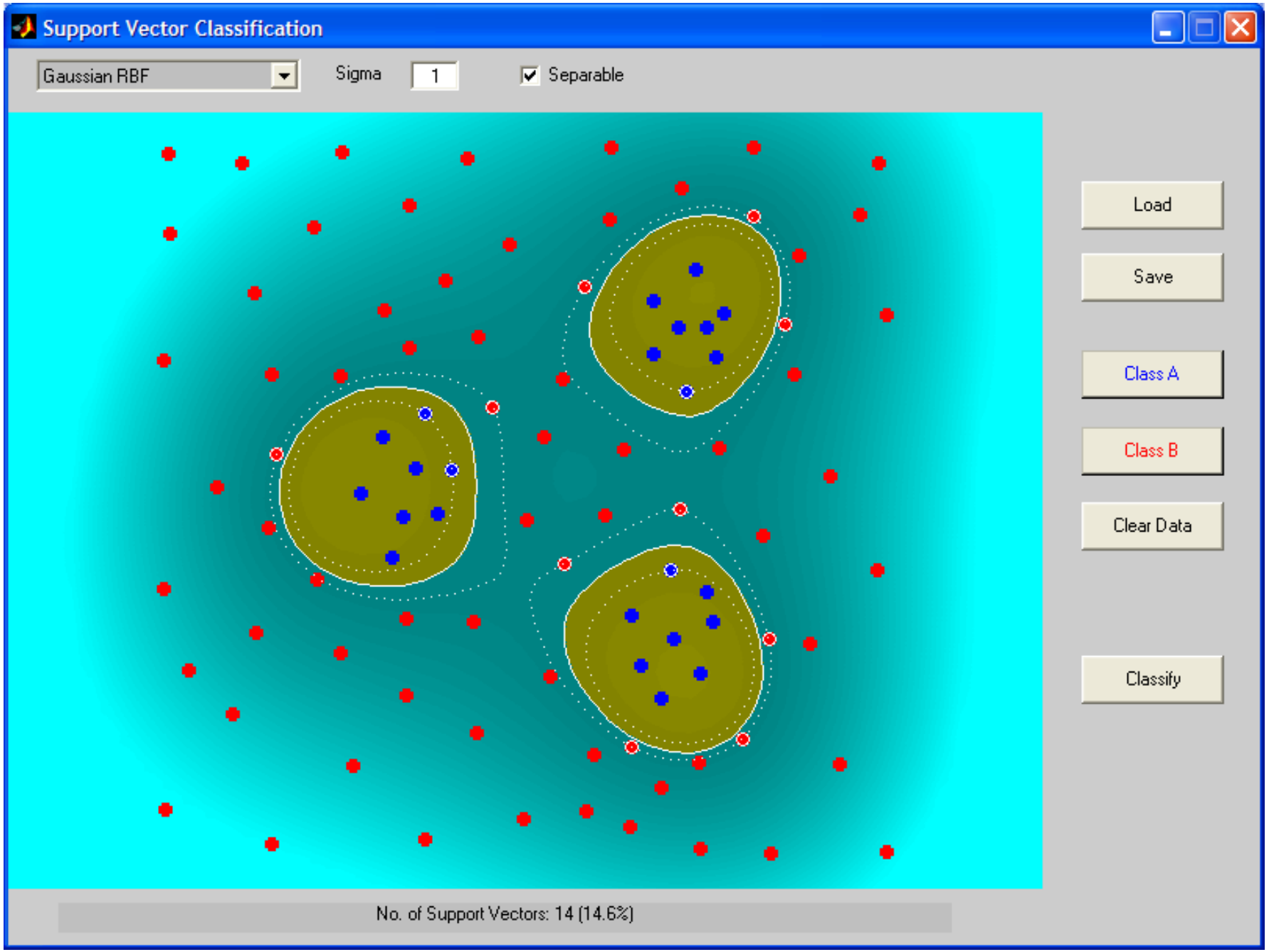
Beispiel: „Inseln“, Polynom mit $p = 3$



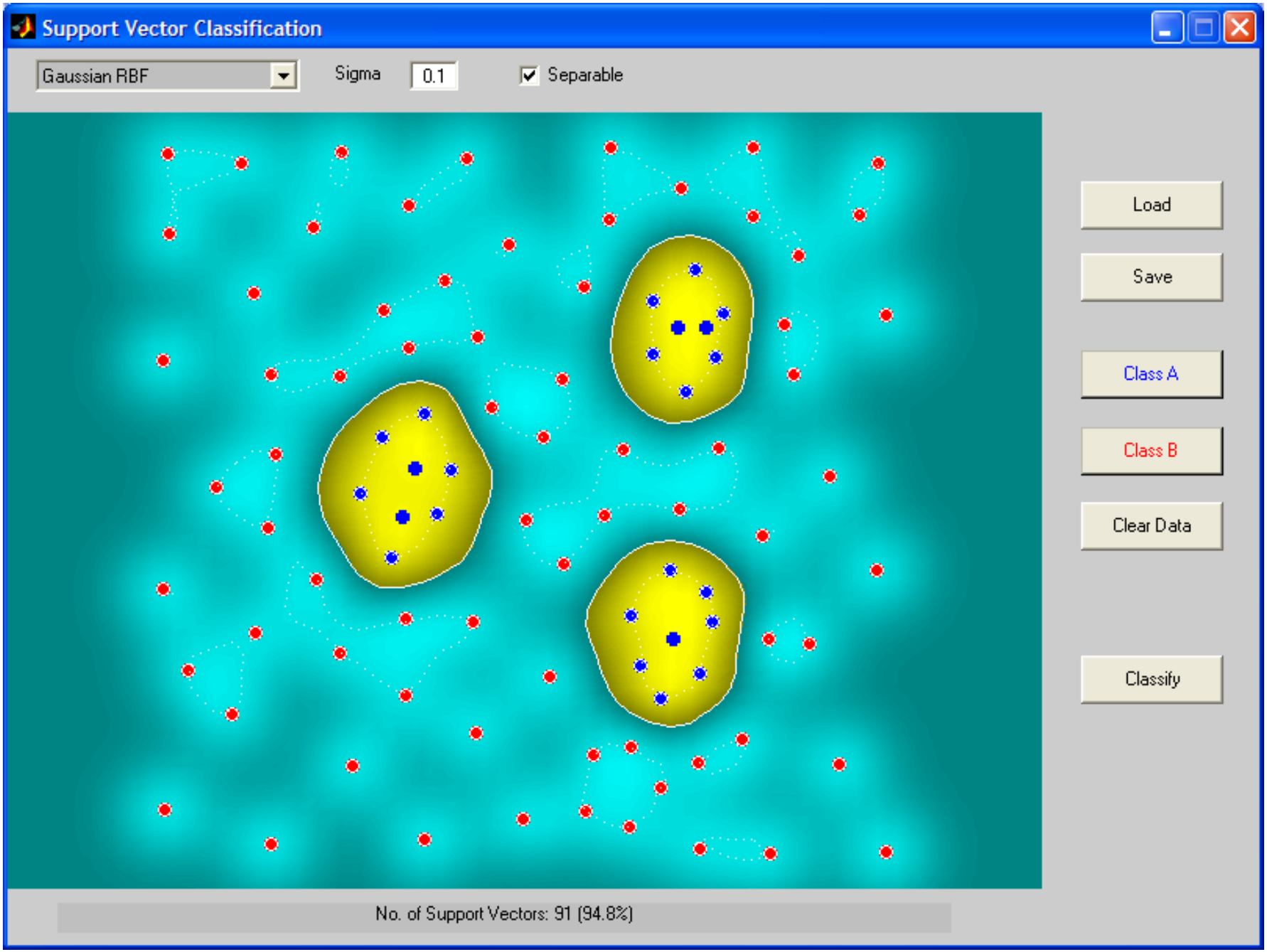
Beispiel: „Inseln“, Polynom mit $p = 4$



Beispiel: „Inseln“, RBFs mit σ=1



Bsp: „Inseln“, RBFs mit σ=0,1; overfitting! Zu viele SVen, nämlich alle $\mathbf{x}_i$!



Visualisierung von Support-Vektoren

(aus Diss. Claus Bahlmann, S. 123)

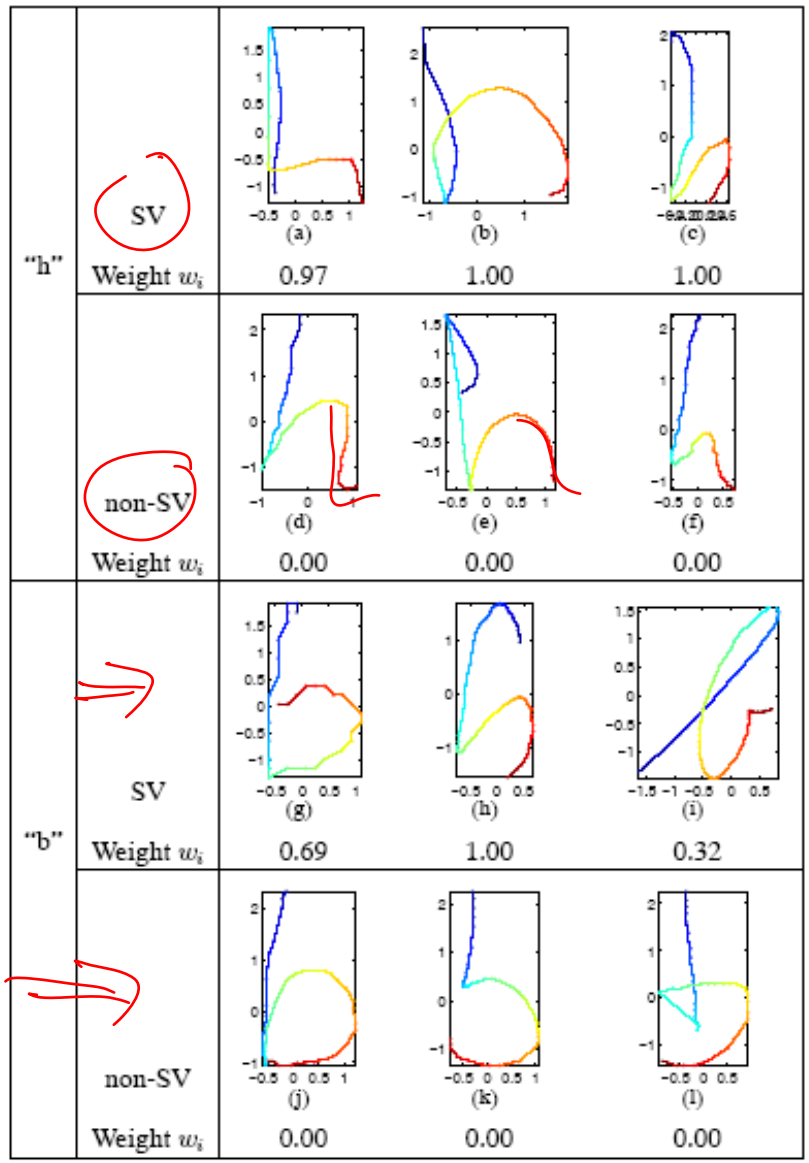


FIGURE 7.5: Support vectors (SV) and non-support vectors (non-SV) of a two-class SVM "h" $\leftrightarrow$ "b". The patterns are represented through the feature subspace according to the two features $\tilde{x}$ and $\tilde{y}$. Note that the tangent slope angle θ is also used in the SVM but not illustrated here.

Anwendungsbeispiel: Zeichenerkennung

Beispiel: US Postal Service Digits [Schö95/96]

16x16 Grauwertbilder

7291 Training, 2007 Test-Samples

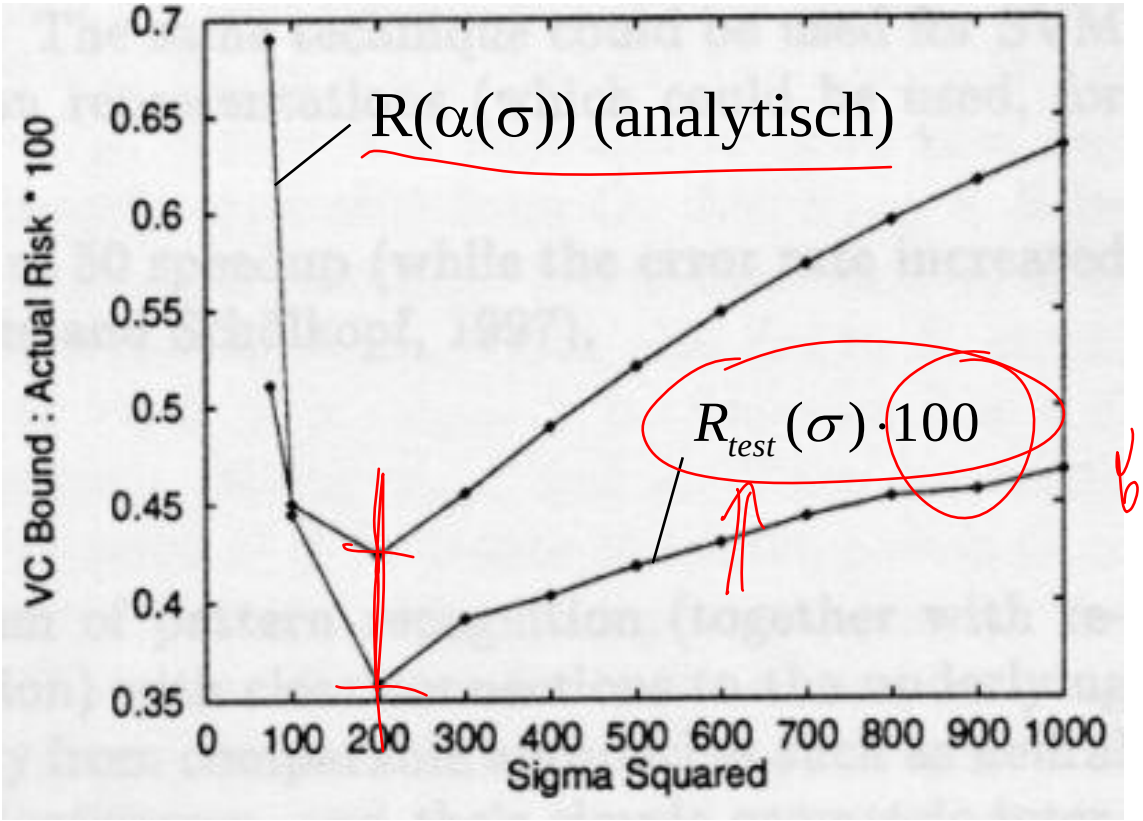
Ergebnisse:

Klassifikator	**Fehlerrate**
Mensch	2.5%
2-Schicht NN	5.9%
5-Schicht NN	5.1%
SVM (Polynom Grad 3)	4.0%
SVM + Invarianz	3.2%

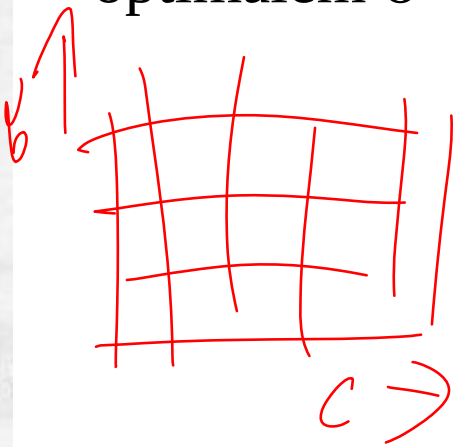
SVM+Invarianz: Ursprüngliche Datenmenge verfünffachen durch Shift der Grauwertmuster in 4 Hauptrichtungen (N,S,O,W) um einen Pixel.

SVM mit RBF, Risikoabschätzung im Vergleich zur tatsächlichen Testfehlerrate

(Nur ein Parameter: σ)



Suche nach optimalem σ



- Theorie: leider nur sehr grobe Abschätzungen, aber qualitativ aussagekräftig (Minimum an der gleichen Stelle). VC-Abschätzung nur sehr ungenau (Faktor 100) und praktisch wenig verwertbar.
- **Alternative:** Suche nach σ und *C* (soft-margin) durch Testfehlerminimierung mit ***Kreuzvalidierung***. Bildet man z.B. den Erwartungswert über alle leave-one-out Experimente, so erhält man eine Abschätzung des echten Risikos.
- Der Aufwand beim leave-one-out Experiment kann reduziert werden auf die Überprüfung der SV, da nur diese einen Einfluss auf die gemessene Fehlerrate haben.

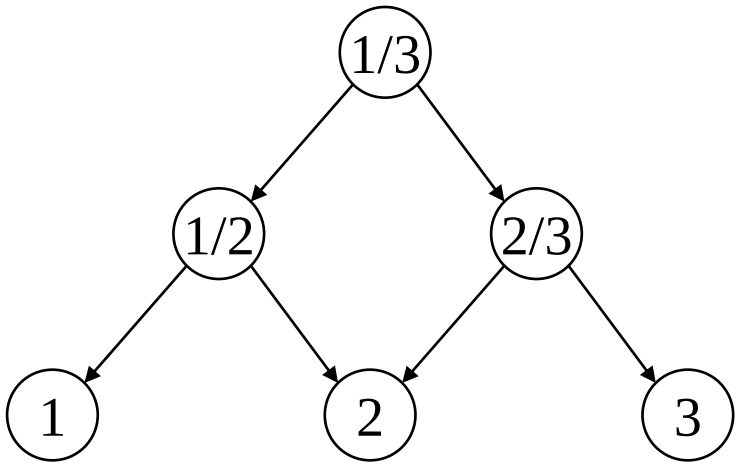
SVM: Eigenschaften und Berechnungskomplexität

Stärken:

- Die SVM liefert nach den derzeitigen Erkenntnissen sehr gute Ergebnisse und findet (unter gewissen Voraussetzungen) ein globales Minimum (Bei NN oder RBFN erhält man in der Regel nur suboptimale Lösungen)
- Sparse-Darstellung der Lösung über N_s Support-Vektoren
- Leicht anwendbar (wenig Parameter (C,σ), es wird kein a-priori-Wissen benötigt, kein Design); die genaue Wahl von (C,σ) ist jedoch zugleich eine der größten Schwächen im Entwurf (grid-search)
- Geometrisch anschauliche Funktionsweise
- Theoretische Aussagen über Ergebnis: globales Optimum, Generalisierungsfähigkeit
- Structural-Risc-Minim. möglich, wenn auch schwer kontrollierbar
- auch Probleme in großen Merkmalsräumen lösbar (100.000 Trainings- und Testmuster)
- Bei Semidefinitheit des Problems: das duale Problem hat dann viele Lösungen, aber alle liefern die gleiche Hyperebene!

Schwächen:

- Multiklassenansatz noch Gegenstand der Forschung. Verschiedene Möglichkeiten zur Reduktion auf Zweiklassenprobleme:
  - Ansatz: eine SVM pro Klasse (one versus rest), d.h. Aufwand und Speicherbedarf steigt mit der Anzahl der Klassen (häufig verwendet).
  - Reduktion eines Multiklassenproblems auf eine Reihe von binären Entscheidungsproblemen (Zweiklassenproblemen)



Schwächen:

- Keine quantitative Qualitätsaussage über Klassifikation
- Langsames, speicherintensives Lernen:

  Laufzeit: $O(N_s^3)$ (Inversion der Hesse-Matrix/Newton-Alg. zur Lösung des quadr. Optim.-Problems)

  (in der Praxis jedoch häufig weniger als quadratisch)

  Speicher: $O(N_s^2)$
- Ansatz zur Verbesserung: Zerlegen in Teilprobleme (Klassifikatorhierarchien)
- Langsames Klassifizieren mit $O(MN_s)$, wobei $M = \dim(\mathcal{H})$ (d.h. im Fall ohne Kernfunktionen $M=N=\dim(\mathbf{x}_i)$), aber bei geeignet gewählten Kernfunktionen gilt auch hier: $M=N$.

Hinweis auf LIBSVMTL:
http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/index.en.html
Objektorientierte C++ Support Vector Machine library (aufbauend auf die LIBSVM von Chih-Jen Lin).

Literatur:

(1) C.J.C. Burges, „A tutorial on support vector machines for pattern recognition“, Knowledge Discovery and Data Mining, 2(2), 1998. (http://www.kernel-machines.org/tutorial.html)

(2) V. Vapnik, „Statistical Learning Theory“, Wiley, New York, 1998.

(3) N. Cristianini, J. Shawe-Taylor, „An introduction to support vector machines and other kernel-based learning methods“, Cambridge Univ. Press, Cambridge 2000.

(4) B. Schölkopf, A. J. Smola, „Learning with kernels“, MIT Press, Boston, 2002.

(5) S.R. Gunn, „Support Vector Machines for Classification and Regression“, Technical Report, Department of Electronics and Computer Science, University of Southampton, 1998.