

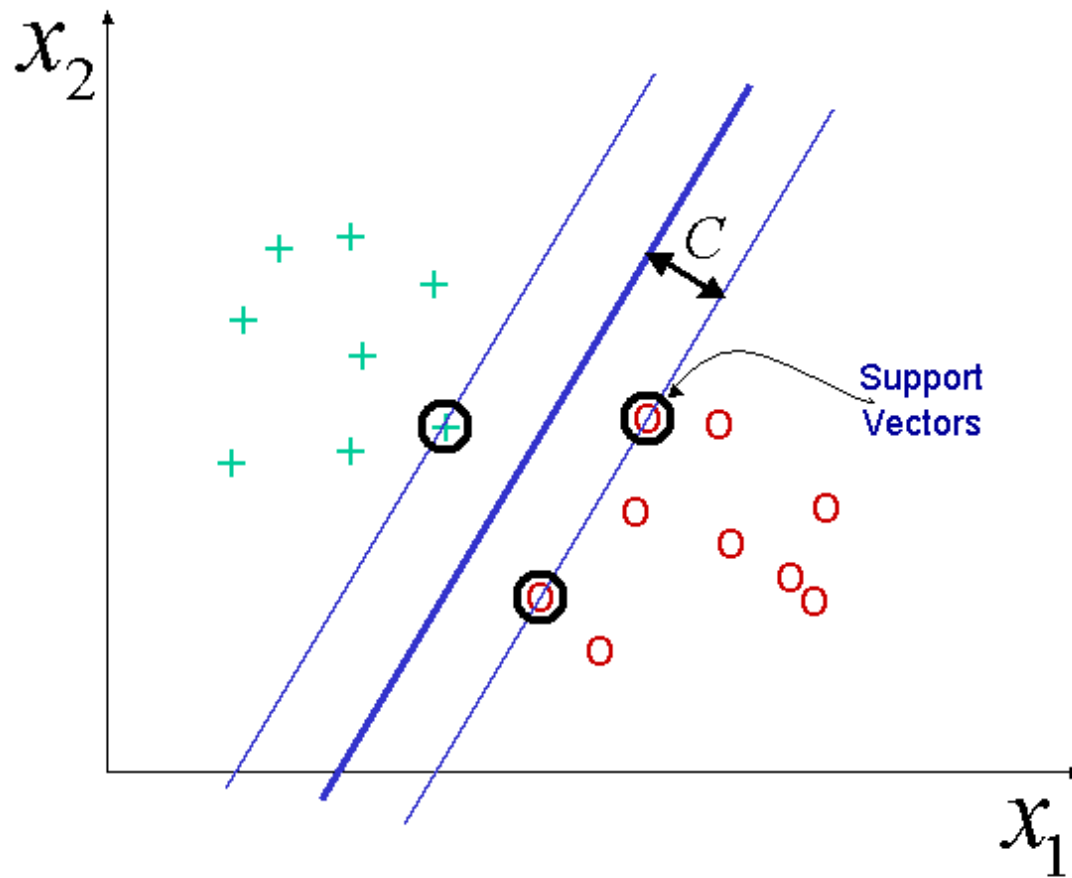
Optimal-trennende Hyperebenen und die Support Vector Machine

Volker Tresp

(Vapnik's) Optimal-trennende Hyperebenen (Optimal Separating Hyperplanes)

- Wir betrachten wieder einen linearen Klassifikator mit $y_i \in \{-1, 1\}$
- Mit Ausnahme der Perzeptrons trennen die soweit vorgestellten Algorithmen nicht notwendigerweise zwei Klassen, auch wenn diese separabel sind
- Dies ist auch unter Umständen richtig, da Klassen sich überlappen können
- Dennoch ist es wünschenswert, auch Algorithmen im Repertoire zu haben, die trennbare Klassen auch trennen
- Vapnik stellte einen Algorithmus vor, der trennbare Klassen trennt; falls Klassen sich nicht trennen lassen, wird die Anzahl der Missklassifikationen klein gehalten
- Ziel ist es, die Klassen zu trennen und dabei den Margin \mathcal{C} zu maximieren (falls Klassen separabel sind)

Optimal-trennende Hyperebenen (2D)



Kostenfunktion mit Nebenbedingungen

- Man verlangt Erfüllung der Nebenbedingungen

$$y_i(\mathbf{x}_i^T \mathbf{w}) = y_i \sum_{j=0}^{M-1} w_j x_{i,j} \geq 1 \quad i = 1, \dots, N$$

erfüllt sind

- Von allen Gewichtsvektoren, die zu einer Lösung führen, die die Nebenbedingungen erfüllen wählt man denjenigen, für den gilt

$$\mathbf{w}_{opt} = \arg \min_{\mathbf{w}} \tilde{\mathbf{w}}^T \tilde{\mathbf{w}} = \arg \min_{\mathbf{w}} \sum_{j=1}^{M-1} w_j^2$$

wobei $\tilde{\mathbf{w}} = (w_1, \dots, w_{M-1})$. (D.h. bei $\tilde{\mathbf{w}}$ fehlt der Offset w_0); $y_i \in \{-1, 1\}$.

Beachte: man minimiert dennoch in Bezug auf (ganz) \mathbf{w}

Margin und Support-Vektoren

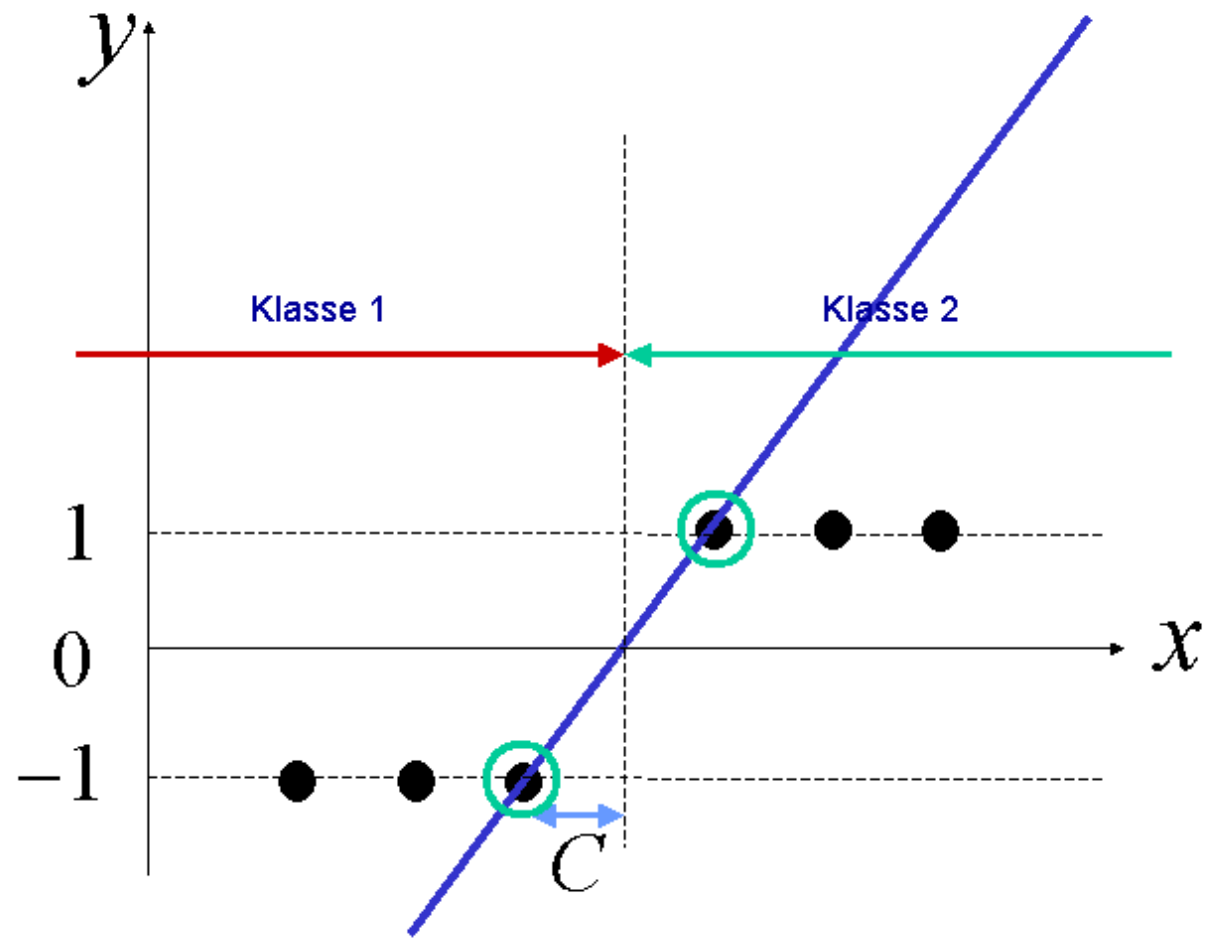
- Der Margin wird dann

$$c = \frac{1}{\|\tilde{\mathbf{w}}_{opt}\|}$$

- Für die *Support-Vektoren* gilt,

$$y_i(\mathbf{x}_i^T \mathbf{w}_{opt}) = 1$$

Optimal-rennende Hyperebenen (1D)



Optimierung: Optimal Separating Hyperplane

- Zur Optimierung mit Randbedingungen (Ungleichheiten) definiert man die Lagrange Funktion

$$L_P = \frac{1}{2} \tilde{\mathbf{w}}^T \tilde{\mathbf{w}} - \sum_{i=1}^N \alpha_i [y_i (\mathbf{x}_i^T \mathbf{w}) - 1]$$

- Die Lagrange Funktion wird in Bezug auf (ganz) \mathbf{w} minimiert und in Bezug auf die Lagrange Multiplikatoren $\alpha_i \geq 0$ maximiert (Sattelpunktlösung).
- Intuition:
 - Wenn eine Nebenbedingung nicht erfüllt ist, so ist $[y_i (\mathbf{x}_i^T \mathbf{w}) - 1] < 0$ und α_i wird anwachsen (beachte negatives Vorzeichen des 2ten Terms)
 - Die Adaption der Gewichte \mathbf{w} wird gleichzeitig versuchen, den Term in den eckigen Klammern zu maximieren bis dieser gleich Null wird
 - Wenn eine Nebenbedingung erfüllt ist, so ist $[y_i (\mathbf{x}_i^T \mathbf{w}) - 1] \geq 0$ und α_i wird Null werden und die Nebenbedingung wird inaktiv

Karush-Kuhn-Tucker (KKT)-Bedingungen

- Optima können oft über das Nullsetzen der Ableitungen gefunden werden. Wegen der Ungleichheitsnebenbedingung werden die Bedingungen etwas komplizierter (siehe KKT (1))
- Im Optimum gelten die (Karush-Kuhn-Tucker (KKT)-Bedingungen. KKT (1) ist

$$\alpha_i [y_i (\mathbf{x}_i^T \mathbf{w} - 1)] = 0 \quad \forall i$$

Entweder α_i ist Null oder der Term in den eckigen Klammern ist Null (wir lassen jetzt den Subskript *opt*) weg) (KKT (1))

- Durch Null-Setzen der Ableitungen von L_P nach $\tilde{\mathbf{w}}$ erhält man KKT (2)

$$\tilde{\mathbf{w}} = \sum_{i=1}^N \alpha_i y_i \tilde{\mathbf{x}}_i$$

und nach w_0 KKT (3)

$$0 = \sum_{i=1}^N \alpha_i y_i$$

- Beachte, dass KKT (2) bedeutet, dass man die optimalen Parameter als lineare gewichtete Summe der Eingangsvektoren schreiben kann (Kern-Trick)!

KKT (4) ist

$$\alpha_i \geq 0 \quad \forall i$$

(1), (2), (3),(4) und bilden die Karush-Kuhn-Tucker Bedingungen.

Wolfe-Dual

Durch einsetzen von KKT (2) erhält man (Wolfe-Dual) das duale Optimierungsproblem (beachte: $\mathbf{w} = (w_0, \tilde{\mathbf{w}}^T)^T$, $\mathbf{x}_i = (1, \tilde{\mathbf{x}}_i^T)^T$,

$$\begin{aligned} L_D &= \frac{1}{2} \sum_{i=1}^N \alpha_i y_i \tilde{\mathbf{x}}_i^T \sum_{i=1}^N \alpha_i y_i \tilde{\mathbf{x}}_i - \sum_{i=1}^N \alpha_i [y_i (\mathbf{x}_i^T \mathbf{w}) - 1] \\ &= \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_k - \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_k \\ &\quad - w_0 \sum_{i=1}^N \alpha_i y_i + \sum_{i=1}^N \alpha_i \end{aligned}$$

Die ersten beiden Terme sind gleich bis auf die Konstante. Der dritte Term ist im Optimum gleich Null (KKT (3)).

Optimierung: Zusammenfassung

- Man löst schließlich: Maximiere in Bezug auf die α_i

$$L_D = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \alpha_i \alpha_k y_i y_k \tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_k$$

mit den Nebenbedingungen (KKT (4))

$$\alpha_i \geq 0$$

und (KKT (3))

$$0 = \sum_{i=1}^N \alpha_i y_i$$

Primale Lösung

- Nachdem die optimalen α_i gefunden sind, setzt man diese in KKT (2) und erhält die optimalen Gewichte.
- Für einen neuen Eingang \mathbf{z} erhält man die Vorhersage

$$\hat{t} = \text{sign}(\tilde{\mathbf{z}}^T \tilde{\mathbf{w}} + w_0)$$

Duale Lösung

- Alternativ lässt sich die Lösung schreiben (mit KKT (2)) als gewichtete Summe über die Support-Vektoren,

$$\hat{t} = \text{sign} \left(\sum_{i \in SV} y_i \alpha_i \tilde{\mathbf{x}}_i^T \tilde{\mathbf{z}} + w_0 \right) = \text{sign} \left(\sum_{i \in SV} y_i \alpha_i k(\tilde{\mathbf{z}}, \tilde{\mathbf{x}}_i) + w_0 \right)$$

mit Kern

$$k(\tilde{\mathbf{z}}, \tilde{\mathbf{x}}_i) = \tilde{\mathbf{z}}^T \tilde{\mathbf{x}}_i$$

- Diese Schreibweise begründet den Begriff *Support Vector Machine*
- Beachte, dass natürlich ebenso mit Basisfunktionen gearbeitet werden kann, wo dann der Kern wird

$$k(\mathbf{z}, \mathbf{x}_i) = \phi(\mathbf{z})^T \phi(\mathbf{x}_i)$$

Optimal-trennende Hyperebenen: Nicht-trennbare Klassen

- Bei sich überlappenden Klassen führt man *slack* Variablen ξ_i ein:
- Die optimale Trennebene kann gefunden werden als

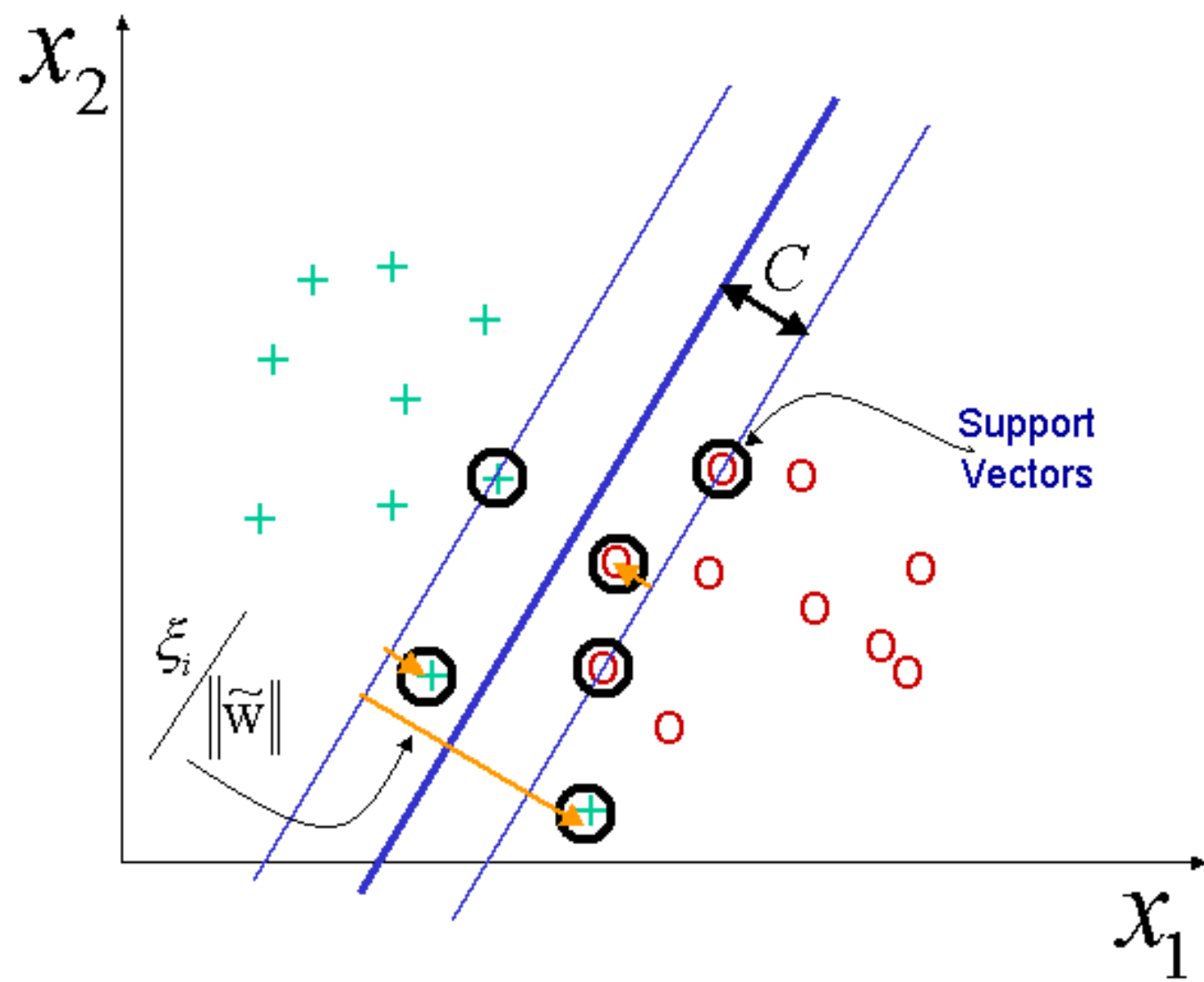
$$\mathbf{w}_{opt} = \arg \min_{\mathbf{w}} \tilde{\mathbf{w}}^T \tilde{\mathbf{w}}$$

unter den Nebenbedingungen, dass

$$y_i(\mathbf{x}_i^T \mathbf{w}) \geq 1 - \xi_i \quad i = 1, \dots, N$$

$$\xi_i \geq 0 \quad \sum_{i=1}^N \xi_i \leq 1/\gamma$$

- $\gamma > 0$ bestimmt den Kompromiss zwischen Trennbarkeit von Klassen und Überlappungsgrad. Für $\gamma \rightarrow \infty$ erhält man den separierbaren Fall



Optimal-trennende Hyperebenen: Kommentare

- Die optimale Trennebene wird gefunden über eine trickreiche Optimierung des resultierenden quadratischen Optimierungsproblems mit linearen Nebenbedingungen
- γ ist ein zu optimierender Hyperparameter (Kreuzvalidierung)

Optimierung über Penalty/Barrier Method

- Ziel: Transformation eines Optimierungsproblems mit Nebenbedingungen in ein Problem ohne Nebenbedingungen
- In der Penalty-Methode wird ein Strafterm (penalty) zur Zielfunktion addiert für Punkte, die die Nebenbedingungen verletzen
- Die Barriere-Methoden (interior point method) sind ähnlich, nur dass der Strafterm unendlich ist für Punkte, die die Nebenbedingungen verletzen und endlich für Punkte, die "fast" die Nebenbedingungen verletzen
- Für unser Optimierungsproblem wählt man (Penalty Methode):

$$\arg \min_{\mathbf{w}} 2\gamma \sum |1 - y_i(\mathbf{x}_i^T \mathbf{w})|_+ + \tilde{\mathbf{w}}^T \tilde{\mathbf{w}}$$

wobei $|arg|_+ = \max(arg, 0)$. Man beginnt in der Optimierung mit kleinem γ und lässt es langsam anwachsen. Für $\gamma \rightarrow \infty$ verlangt man, dass alle Nebenbedingungen in der Lösung erfüllt sind. Lässt man γ endlich, dann erlaubt man "Slack"

Vergleich: Musterbasiertes Lernen

- Perzeptron

$$w_j \longleftarrow w_j + \eta \left(y(t) - \text{sign} \left(x(t)^T w \right) \right) x_j(t)$$

$y(t) \in \{-1, 1\}$. Beachte, dass die Klammer Null ist, wenn richtig klassifiziert. Ansonsten ist der Term entweder gleich 2 oder gleich -2. Auch

$$w_j \longleftarrow w_j + \eta y(t) x_j(t)$$

für falsch klassifizierte Muster

- Logistic regression: Die “natürliche” kontinuierliche Verallgemeinerung

$$w_j \longleftarrow w_j + \eta \left(y(t) - \text{sig} \left(x(t)^T w \right) \right) x_j(t)$$

- Neuronale Netze

$$w_j \longleftarrow w_j + \eta \left(y(t) - \text{sig} \left(x(t)^T w \right) \right) \text{sig}' \left(x(t)^T w \right) x_j(t)$$

Wird ein Muster mit hoher Sicherheit falsch klassifiziert, ist der Gradient nahezu Null!

- Regression (ADALINE)

$$w_j \longleftarrow w_j + \eta \left(y(t) - \left(x(t)^T w \right) \right) x_j(t)$$

- Vapniks optimal-trennende Hyperebenen

$$w_j \longleftarrow w_j + \eta y(t) x_j(t) \quad \text{wenn } y(t) \left(x(t)^T w \right) - 1 > 0$$

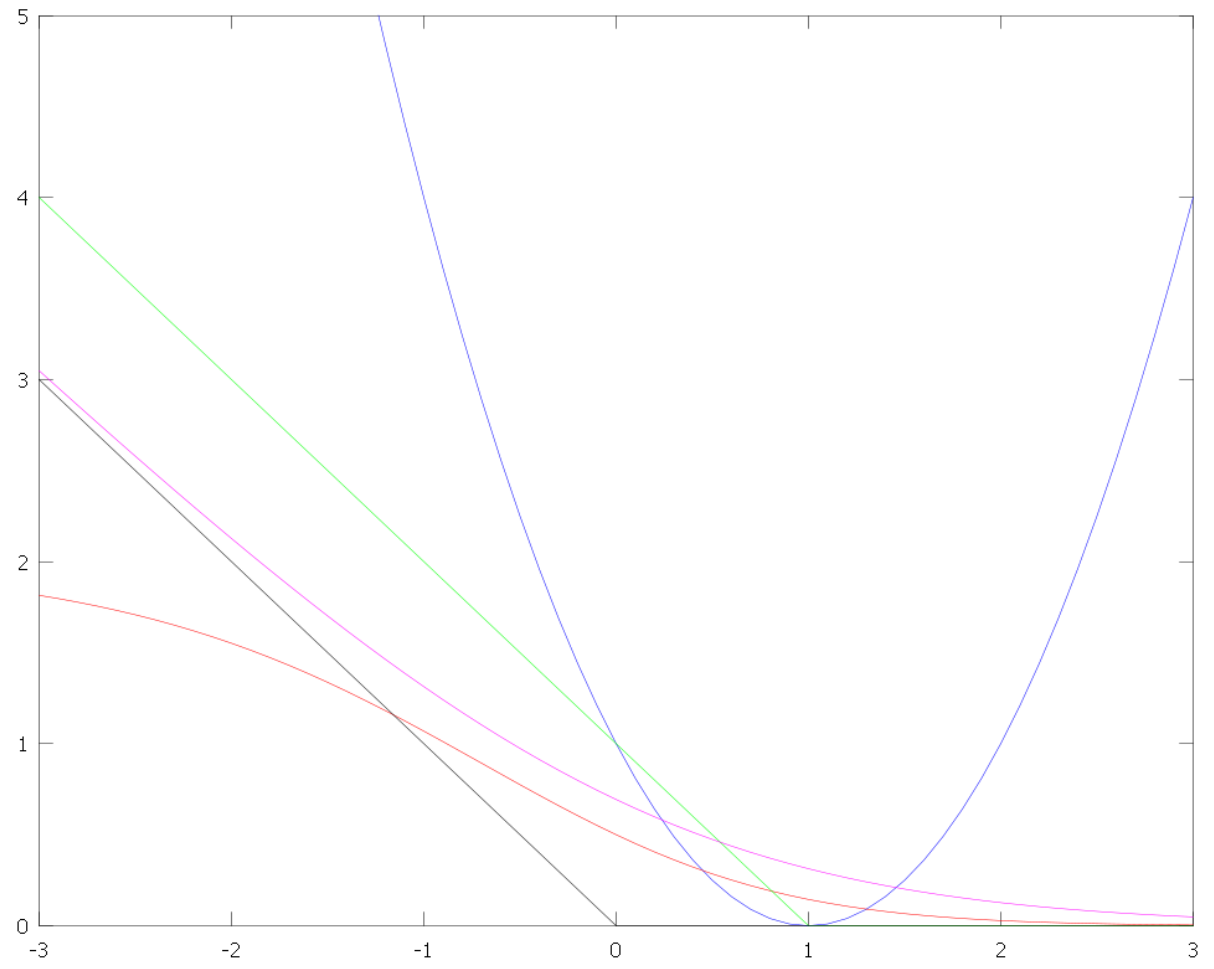
Beachte die große Ähnlichkeit zur Perzeptron Lernregel! Beachte, dass hier die Lösung durch die Regularisierung eindeutig ist!

- Bei allen Iterationen kann man einen regularisierenden *weight-decay* Term mit hinzunehmen, das heißt den Adaptionsschritt

$$w_j \longleftarrow w_j - \eta w_j$$

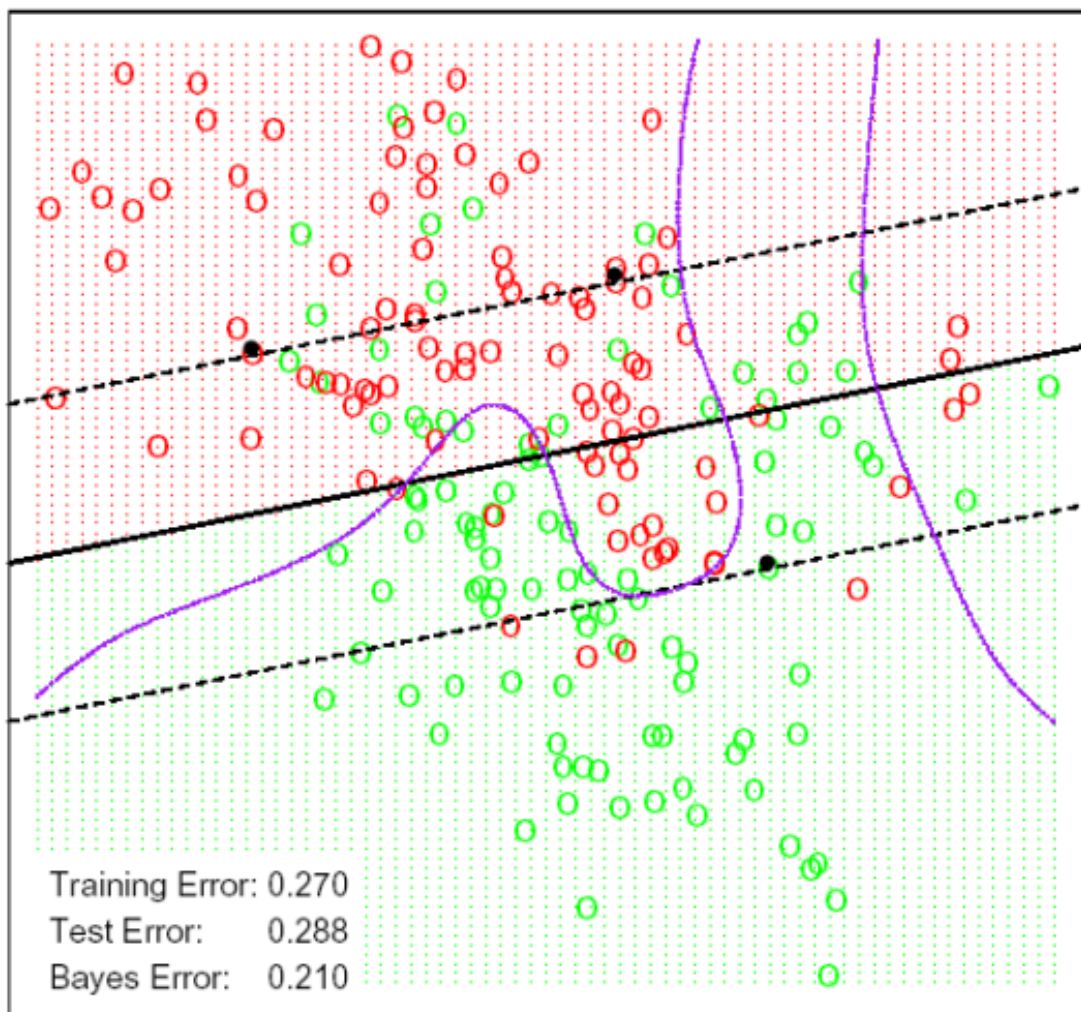
Vergleich der Kostenfunktionen

- Betrachten wir der Einfachheit einen Datenpunkt der Klasse 1
- Der Beitrag zur Kostenfunktion ist:
 - Quadratische Kostenfunktion (blau) : $(1 - (\mathbf{x}_i^T \mathbf{w}))^2$
 - Perzeptron (schwarz) : $|\mathbf{x}_i^T \mathbf{w}|_+$
 - Vapniks Hyperebenen (grün): $|1 - (\mathbf{x}_i^T \mathbf{w})|_+$
 - Logistische Regression (magenta): $\log(1 + \exp(-\mathbf{x}_i^T \mathbf{w}))$
 - Neuronales Netz (rot): $(1 - \text{sig}(\mathbf{x}_i^T \mathbf{w}))^2$



Künstliches Beispiel

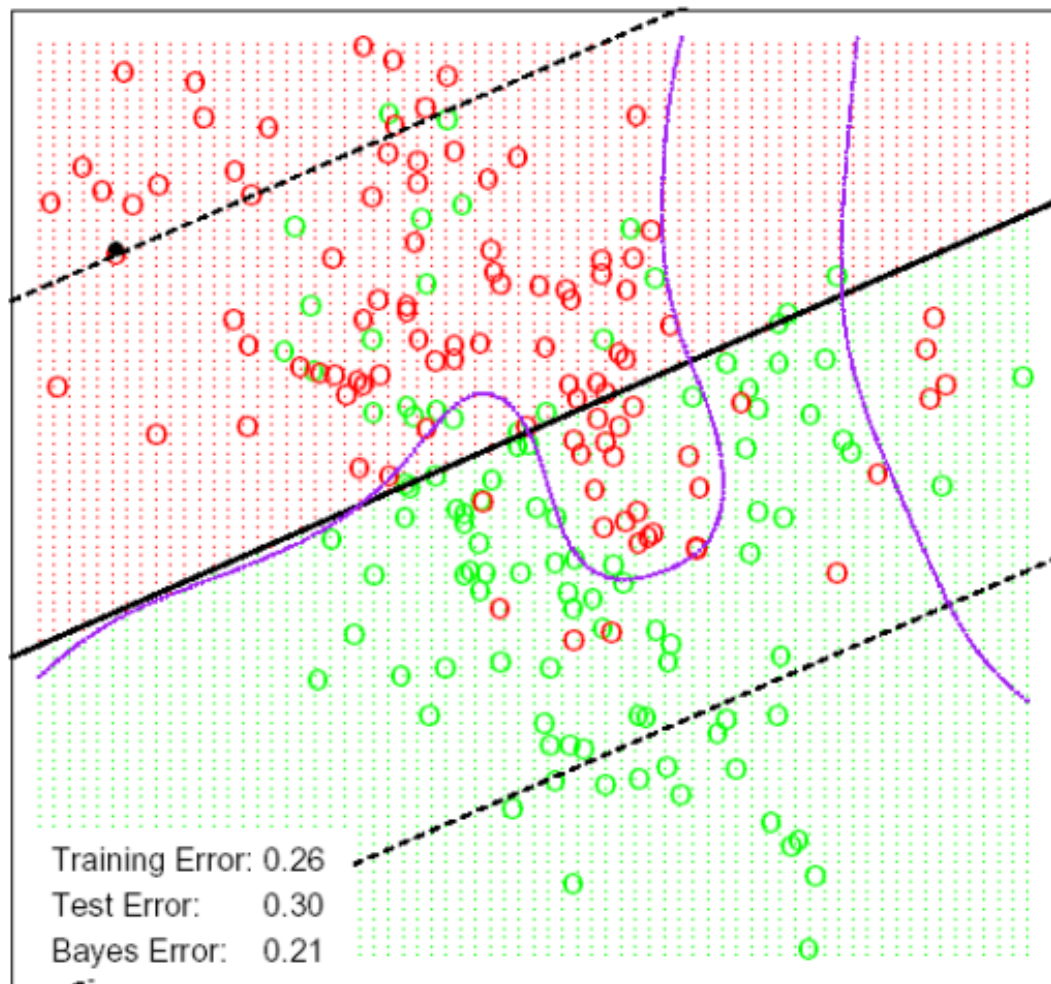
- Daten von zwei Klassen (rote, grüne Kringel) werden generiert
- Die Klassen überlappen
- Die optimale Klassifikationsgrenze ist gestrichelt gezeigt
- Gezeigt ist die Trennebene der linearen SVM mit großem γ



$$\gamma = 10000$$

Künstliches Beispiel

- Gezeigt ist die Trennebene der linearen SVM mit kleinem γ

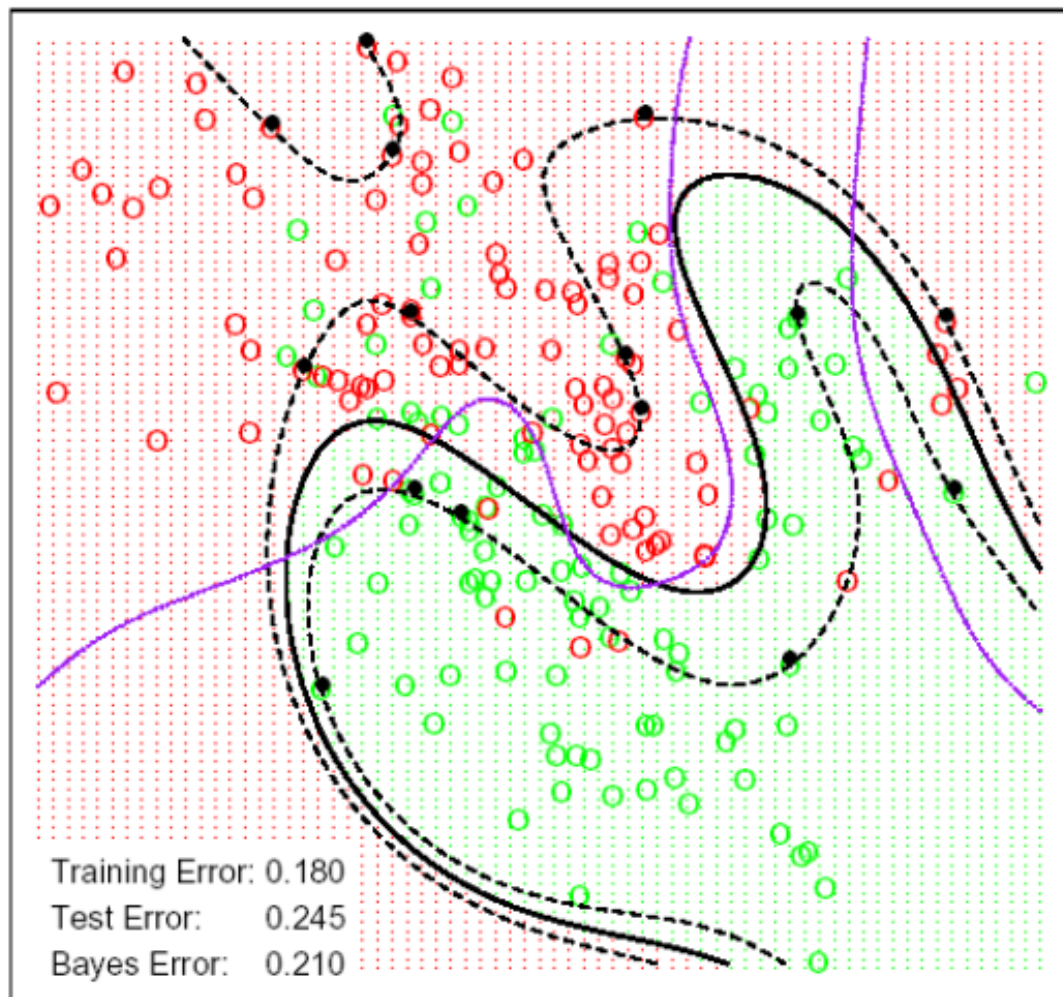


$$\gamma = 0.01$$

Künstliches Beispiel

- Mit polynomialen Basisfunktionen

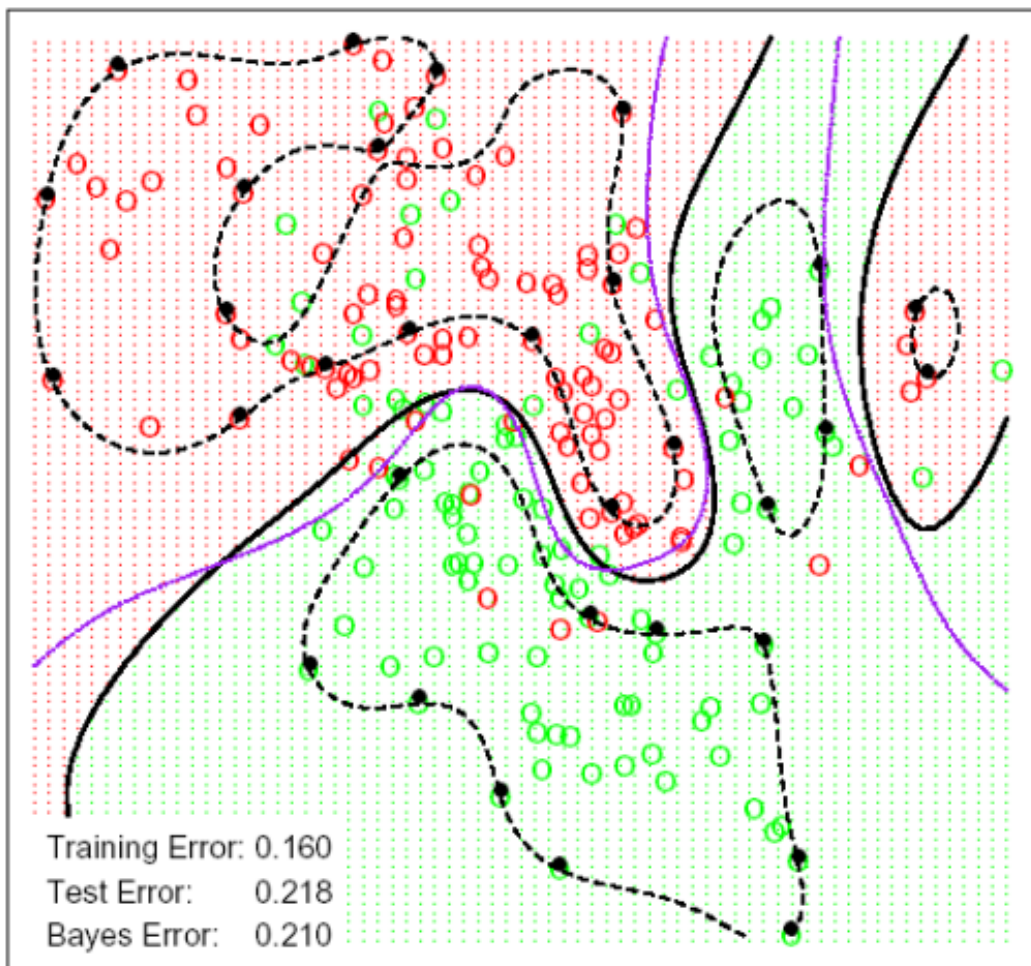
SVM - Degree-4 Polynomial in Feature Space



Künstliches Beispiel

- Mit radialen Basisfunktionen den besten Testfehler

SVM - Radial Kernel in Feature Space



Bemerkungen

- Der “Kern-Trick” erlaubt es, in unendlich hohen Dimensionen zu arbeiten
- Dennoch können durch die Regularisierung sehr gute Ergebnisse erzielt werden; z.B. hängt die Generalisierung einer SVM vom Margin ab und nicht von der Dimensionalität des Problems

APPENDIX: Regression über Lagrange Parameter (ohne Rauschen)

- Ziel: Minimierung von $\sum_i w_i^2$ unter der Nebenbedingung, dass $y_i = \mathbf{x}_i^T \mathbf{w}$. Mit weniger Daten als Parametern gibt es eine Lösung (die Optimierungsaufgabe gehört zum *quadratic programming*). Die Lagrange Funktion ist

$$\begin{aligned} L &= \sum_{i=0}^{M-1} w_i^2 + \sum_{i=1}^N \mu_i (y_i - \mathbf{x}_i^T \mathbf{w}) \\ &= \mathbf{w}^T \mathbf{w} + \mu^T (\mathbf{y} - X \mathbf{w}) \end{aligned}$$

- μ ist ein Vektor von N Lagrange Parametern. L wird minimiert in Bezug auf die Parameter und maximiert in Bezug auf die Lagrange Parameter. Die Lösung kann explizit berechnet werden durch Nullsetzen der Ableitung nach den Gewichten \mathbf{w} :
 $\mathbf{w}_{opt} = 1/2 X^T \mu$
- Dies setzen wir in die Kostenfunktion ein. Nullsetzen der Ableitung nach den Lagrange Parametern μ ergibt: $\mu_{opt} = 2(X X^T)^{-1} \mathbf{y}$

- Somit

$$\mathbf{w}_{opt} = X^T (X X^T)^{-1} \mathbf{y}$$

APPENDIX: Regression über Lagrange Parameter mit Slack Variablen

- Minimiere $\sum_i w_i^2$ mit Gleichheitsnebenbedingung $y_i = \mathbf{x}_i^T \mathbf{w} + \epsilon_i$ wobei ϵ_i die Slack Variablen sind und Ungleichheitsnebenbedingung $\sum_i \epsilon_i^2 \leq A$
- Lagrange Funktion mit Lagrange Parametern und Slack Variablen

$$\begin{aligned} L &= \sum_{i=0}^M w_i^2 + \beta \left(\sum_{i=1}^N \epsilon_i^2 - A \right) + \sum_{i=1}^N \mu_i (y_i - \mathbf{x}_i^T \mathbf{w} + \epsilon_i) \\ &= \mathbf{w}^T \mathbf{w} + \beta (\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} - A) + \boldsymbol{\mu}^T (\mathbf{y} - \mathbf{X} \mathbf{w} + \boldsymbol{\epsilon}) \end{aligned}$$

- $\boldsymbol{\mu}$ ist ein Vektor als N Lagrange Parametern und $\beta \geq 0$ ist ein zusätzlicher Lagrange Parameter und es gilt $A > 0$.
- Nullsetzen der Ableitung nach den Gewichten \mathbf{w} ergibt: $\mathbf{w}_{opt} = 1/2 \mathbf{X}^T \boldsymbol{\mu}$
- Nullsetzen der Ableitung nach den Slack-Variablen $\boldsymbol{\epsilon}$ ergibt: $\boldsymbol{\epsilon}_{opt} = -\frac{1}{2\beta} \boldsymbol{\mu}$

- Beides setzen wir in die Kostenfunktion ein. Nullsetzen der Ableitung nach den Lagrange Parametern μ ergibt: $\mu_{opt} = 2(XX^T + 1/\beta I)^{-1}\mathbf{y}$.
- Somit (unter Verwendung eines Matrix Inversions Lemmas)

$$w_{opt} = X^T (XX^T + 1/\beta I)^{-1}\mathbf{y} = (X^T X + 1/\beta I)^{-1} X^T \mathbf{y}$$

Somit kann man den Regularisierungsparameter λ in Bezug bringen zu einem Lagrange Parameter!

- Schließlich gilt die KKT Bedingung $\beta \left(\sum_{i=1}^N \epsilon_i^2 - A \right) = 0$. Wenn A sehr groß gewählt wird, kann die Lösung tatsächlich sein $\beta_{opt} = 0$ und damit $\mathbf{w}_{opt} = 0$. Sonst ist $\left(\sum_{i=1}^N \epsilon_i^2 - A \right) = 0$ und ich kann β_{opt} durch A ausdrücken.