Seminar

„Statistische Lerntheorie und ihre Anwendungen“

Support Vector Machines (SVM)

Jasmin Fischer

12. Juni 2007

Inhaltsverzeichnis

1 Grundlagen

In diesem Abschnitt werden kurz die zugrundeliegenden Voraussetzungen erläutert und auf die wichtigsten Bezeichnungen eingegangen, die im Folgenden verwendet werden sollen.

Voraussetzungen:
Im Rahmen der Klassifikation mit Support Vector Machines wird von N Trainingsdaten $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ mit $x_i \in \mathbb{R}^g$ und $y_i \in \{\pm 1\}$ (für i=1,2,...,N) ausgegangen. Dabei entsprechen die ersten Vektorkomponenten den Eingangsdaten und die zweiten Komponenten den beiden Klassen, in die eingeordnet werden soll.

Die grundlegende Idee der Support Vector Klassifikation besteht darin, die Menge von Objekten durch eine Hyperebene in zwei Klassen zu unterteilen.

Vorgehensweise:
Um eine solche Einordnung zu erreichen, sucht man eine Funktion $f : \mathbb{R}^g \to \{\pm 1\}$, die die Trainingsmenge korrekt klassifiziert, d.h. eine Funktion, die $f(x_i) = y_i$

- im Fall der Trennbarkeit $\quad \forall\, i = 1, ..., N$
- sonst für zumindest „viele“ i

erfüllt.
Neue Punkte x_{neu} können schließlich durch $f(x_{neu})$ einer Klasse zugeordnet werden.

2 Lineare Trennung

Ausgangslage der anschließenden Betrachtungen soll nun die lineare Trennbarkeit der vorliegenden Trainingsdaten sein. Anhand dieses leichtesten Falls der SVM-Klassifikation werden die Idee und die Vorgehensweise verdeutlicht, die auch in den späteren Kapiteln unter verschärften Bedingungen an die Trainingsdaten noch Gültigkeit besitzen.

2.1 Aufstellung der Hyperebenengleichung

In diesem Zusammenhang stellt sich hauptsächlich die Frage, wie die gewünschte Hyperebene zu wählen ist.

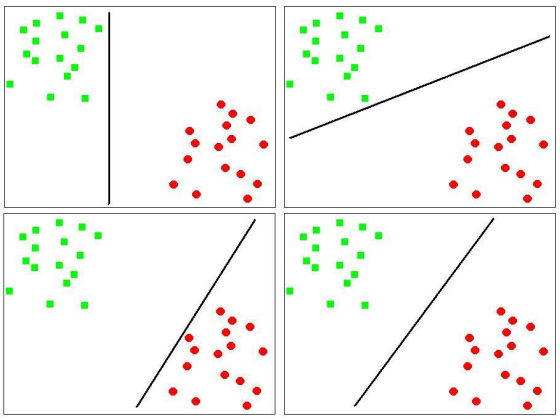

Abbildung 1: Verschiedene trennende Hyperebenen

Abbildung 1 verdeutlicht die auftretende Problematik. Wie zu erkennen ist, kann eine lineare Trennung auf vielfältige Weise erfolgen. Allerdings wird auch klar, dass es wünschenswert ist, die Hyperebene so zu legen, dass der Abstand zu beiden Klassen jeweils möglichst groß wird. Denn nur dadurch kann die Wahrscheinlichkeit einer korrekten Neueinordnung von Punkten maximal werden.
Aus diesem Grund wird die Methode der SVM auch oft als *„large margin classification“* bezeichnet.

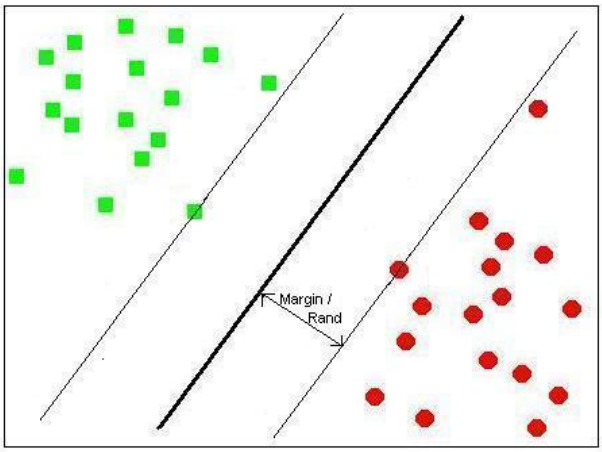

Abbildung 2: Erhaltung eines möglichst breiten Randes um die Klassengrenzen herum

In einem ersten Schritt wird die **trennende Hyperebene** $\mathcal{H}$ folgendermaßen definiert:

$$\mathcal{H} := \{x \in \mathbb{R}^g \mid \langle w, x \rangle + b = 0\}$$

mit den bestimmenden Elementen w und b. Dabei bezeichnet $w \in \mathbb{R}^g$ einen zu $\mathcal{H}$ orthogonalen Vektor und $b \in \mathbb{R}$ die Verschiebung.

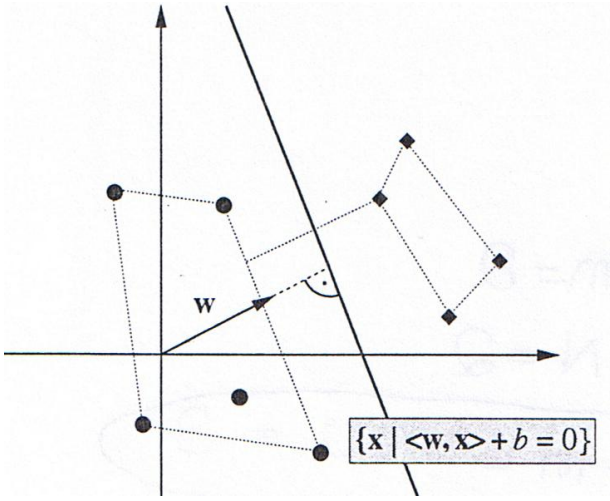

Abbildung 3: Zwei trennbare Klassen, die durch die Hyperebene (bestimmt durch w und b) geteilt werden

Allerdings ergibt sich bei obiger Festlegung das Problem, dass es sich um keine eindeutige Beschreibung der Hyperebene handelt, denn

$$\mathcal{H} = \{x \in \mathbb{R}^g \mid \langle aw, x\rangle + ab = 0\} \quad \forall\, a \in \mathbb{R} \setminus \{0\}.$$

Ein Ausweg wird durch folgenden Skalierung erreicht:
Man nennt das Paar $(w, b) \in \mathbb{R}^g \times \mathbb{R}$ **kanonische Form der Hyperebene** wenn es

$$\min_{i=1,..,N} |\langle w, x_i\rangle + b| = 1$$

erfüllt.
Über diese Festlegung kann in einem zweiten Schritt einer der wichtigsten Begriffe im Zusammenhang mit Support Vector Machines eingeführt werden - der sogenannte Rand (engl. *margin*).

Als **Rand** bezeichnet man den Abstand der kanonischen Hyperebene zu dem Punkt, der ihr am nächsten liegt.
Er lässt sich zu $\frac{1}{\|w\|}$ berechnen.

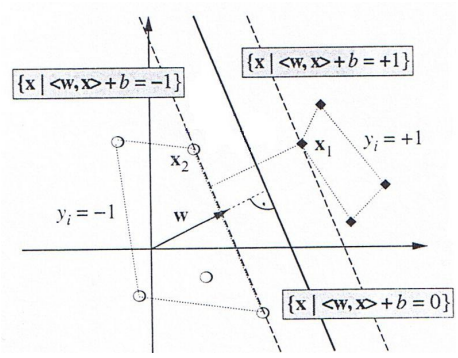

Abbildung 4: Durch Betrachtung zweier gegensätzlicher Punkte x_1 und x_2, die direkt auf dem Rand liegen, kann gezeigt werden, dass sich dieser zu exakt $\frac{1}{\|w\|}$ ergibt.

$$\begin{aligned} \langle w, x_1\rangle + b &= +1 \\ \langle w, x_2\rangle + b &= -1 \\ \Rightarrow \langle w, (x_1 - x_2) &= 2 \\ \Rightarrow \langle \frac{w}{\|w\|}, (x_1 - x_2)\rangle &= \frac{2}{\|w\|} \end{aligned}$$

An dieser Stelle soll kurz auf die **geometrische Bedeutung** eingegangen werden.
Wird der Vektor w normiert, so bezeichnet das Skalarprodukt von w und x gerade die Länge der Projektion des Vektors x in Richtung w. Durch Addieren von b erreicht man damit den Abstand des Punktes x zur Hyperebene.
Somit kann folglich die Entscheidungsfunktion durch Anwendung der Signum-Funktion auf $\langle w, x\rangle + b$ aufgestellt werden.

2.2 Optimierungsproblem und Lösung

Wie oben bewiesen, ergibt sich der Rand zu $\frac{1}{\|w\|}$.
Um ihn zu maximieren muss also die Länge des Vektors w minimiert werden.
Zudem soll die zugehörige Entscheidungsfunktion $f(x) = sgn(\langle w, x\rangle + b)$ die Bedingung $f(x_i) = y_i$ für jeden Trainingspunkt (d.h. $\forall\ i = 1, ..., N$) erfüllen. Multiplikation mit der jeweiligen Klasse y_i stellt sicher, dass der Term $y_i(\langle w, x\rangle + b)$ im Falle der korrekten Klassifikation immer positiv ist. Daraus ergibt sich eine äquivalente Umformung der Nebenbedingungen zu

$$y_i(\langle x_i, w\rangle + b) \geq 1 \quad \forall\ i = 1, ..., N.$$

Zusammenfassend kann man das folgende konvexe Optimierungsproblem (mit quadratischer Zielfunktion und linearen Nebenbedingungen) aufstellen:

$$\underset{w\in\mathbb{R}^g,\ b\in\mathbb{R}}{\text{minimiere}} \quad \frac{1}{2}\|w\|^2$$
$$\text{unter den Bedingungen:} \quad y_i(\langle x_i, w\rangle + b) \geq 1 \quad \forall\ i = 1, ..., N.$$

Das führt zu der zugehörigen Lagrange-Funktion

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i(y_i(\langle x_i, w\rangle + b) - 1)$$

mit $\alpha = (\alpha_1, ..., \alpha_N)$ und $\alpha_i \geq 0$ (Lagrange Multiplikatoren).
Diese ist bezüglich α zu maximieren und bezüglich w und b zu minimieren,
d.h.

$$\frac{\partial}{\partial b}L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial w}L(w, b, \alpha) = 0.$$

Damit folgt

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{und} \quad w = \sum_{i=1}^{N} \alpha_i y_i x_i.$$

Laut den Kuhn-Tucker-Bedingungen gilt für den Sattelpunkt:

$$\alpha_i [y_i(\langle x_i, w \rangle + b) - 1] = 0 \quad \forall\ i = 1, ..., N.$$

Folglich stellt man fest, dass im Optimum entweder $\alpha_i = 0$ oder $y_i(\langle x_i, w \rangle + b) = 1$ gilt. Unter zusätzlicher Betrachtung der obigen Darstellung von $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ lässt sich erkennen, dass nur jene Trainingspunkte mit $\alpha_i > 0$ Einfluss auf die optimale Lösung haben. Diese Punkte erfüllen letztere Sattelpunkt-Gleichung und liegen somit auf dem Rand. Sie werden *Support Vectors („Stützvektoren“)* genannt.

⇒ Nur die Support Vectors bestimmen die eindeutige Lösung:

$$w = \sum_{\{i \in \{1,...,N\} : x_i \ Support \ vector\}} \alpha_i y_i x_i$$

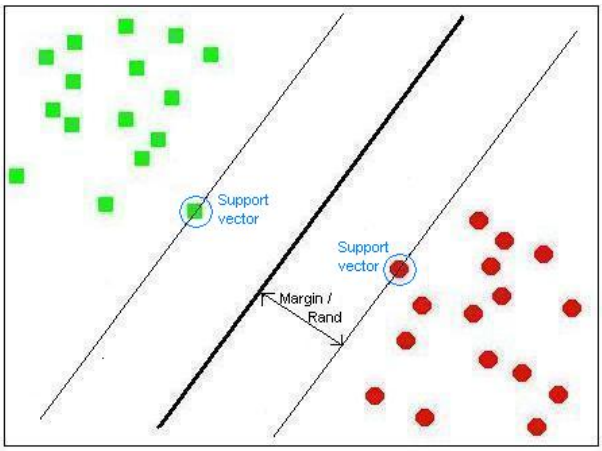

Abbildung 5: Support Vectors

Nun kann das duale Programm aufgestellt werden.
Dazu werden die Ergebnisse der primalen Optimierung in die Lagrange-Funktion eingesetzt. Man erhält:

$$\underset{\alpha\in\mathbb{R}^N}{\text{maximiere}} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle$$

unter den Bedingungen

$$\begin{aligned} \alpha_i &\geq 0 \quad \forall\, i = 1, ..., N \\ \sum_{i=1}^{N}\alpha_i y_i &= 0 \quad . \end{aligned}$$

SVM-Algorithmus:
Zusammenfassend lässt sich die Vorgehensweise einer Support Vector Machine folgendermaßen angeben:

1. Durch das duale Programm könnnen die Lagrange-Mulitiplikatoren α_i der Support Vektoren berechnet und...
2. mit diesen der Vektor $w = \sum_{i=1}^{N}\alpha_i y_i x_i$ der kanonischen Hyperebene bestimmt werden.
3. Die Verschiebung ergibt sich zu $b = y_j - \sum_{i=1}^{N} y_i\alpha_i\langle x_j, x_i\rangle$.
4. Anschließend stellt man die gesuchte Entscheidungsfunktion $f(x) = sgn(\langle w, x\rangle + b)$ in folgender Form auf:

$$f(x) = sgn\Big(\sum_{i=1}^{N}\alpha_i y_i\langle x, x_i\rangle + b\Big).$$

An dieser Stelle soll nochmal darauf hingewiesen werden, dass es sich bei den Support Vector Machines um statistische Methoden zur Klassifikation handelt, die in Computerprogrammen umgesetzt werden und nicht um greifbare Maschinen (bestehend aus Bauteilen). Der Namensteil Machine deutet vielmehr auf des maschinelle Lernen hin.

3 Nichtlineare Klassifikation

Es stellt sich nun die Frage, wie mit nicht linear trennbaren Trainingsdaten umgegangen werden soll. Im Folgenden wird gezeigt, dass es auch unter diesen Voraussetzungen möglich ist, Support Vector Machines zu verwenden. Dazu bedient man sich einiger Tricks und Vorgehensweisen, die nachfolgend erläutert werden sollen.

3.1 Grundlegende Idee

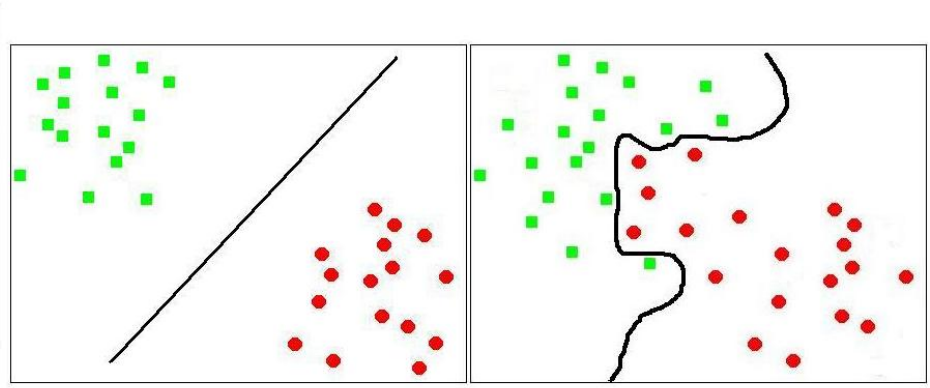

Abbildung 6: Links der linear trennbare Fall, rechts die überlappenden Daten

Um die bisher angewandte Methodik der Hyperebenen-Trennung weiter anwenden zu können, besteht der Grundgedanke darin, den Vektorraum und die zugehörigen Trainingsdaten durch eine Funktion Φ in einen Raum mit so hoher Dimension zu überführen, dass sich die Trainingsdaten dort linear trennen lassen. Wenn dies geschehen ist, kann die kanonisch trennende Hyperebene in diesem höherdimensionalen Raum wie bisher diskutiert bestimmt werden. Bei der Rücktransformation in den ursprünglichen Raum wird die Hyperebene zu einer nicht-linearen Trennfläche.

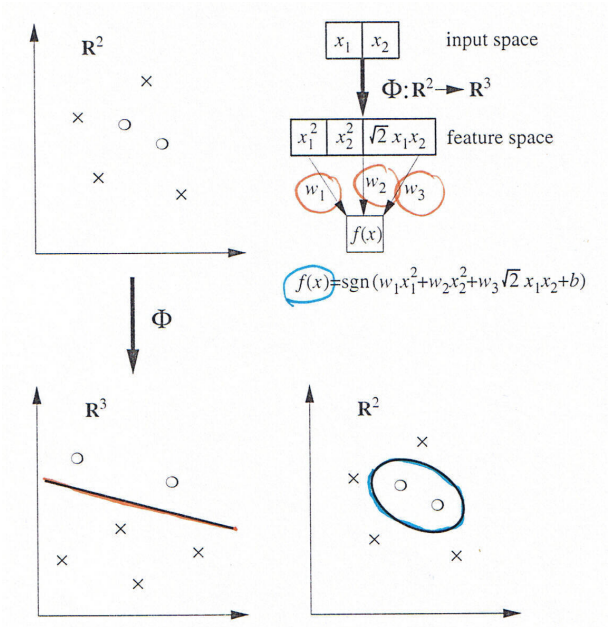

Abbildung 7: Die ursprünglichen Daten *(oben links)* werden durch Φ nichtlinear in einen höher dimensionalen Raum $\mathcal{M}$ *(hier:* $\mathcal{M} = \mathbb{R}^3$*)* abgebildet und dort eine trennende Hyperebene *(unten links)* konstruiert. Die SVM *(oben rechts)* entspricht einer nichtlinearen Entscheidungsfläche im ursprünglichen Raum *(unten rechts)*.

3.2 Der Kern-Trick

Allerdings taucht bei diesen Überlegungen die Problematik auf, dass sowohl im dualen Optimierungsproblem, als auch in der Entscheidungsfunktion die Trainingsdaten x_i nur in den Skalarprodukten erscheinen. Im höherdimensionalen Raum sind folglich Berechnungen der Form $\langle \Phi(x_i), \Phi(x_j) \rangle$ nötig. Dies ist sehr komplex und rechenlastig.
Ein Ausweg wird durch Anwendung einer sogenannten *Kern-Funktion* k erreicht, die aus dem $\mathbb{R}^g \times \mathbb{R}^g$ abbildet und sich wie ein Skalarprodukt in $\mathcal{M}$ verhält:

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle.$$

Kern-Funktionen zeichnen sich im Wesentlichen durch die Eigenschaften

- $k : \mathbb{R}^g \times \mathbb{R}^g \to \mathcal{M}$ (wobei $\mathcal{M}$ mit einem Skalarprodukt versehen sein soll)
- k symmetrisch und positiv definit

aus.

Im Zusammenhang mit SVM werden typischerweise die Funktionen

- POLYNOMIELL VOM GRAD d: $k(x_i, x_j) = (c + \langle x_i, x_j \rangle)^d$ für c konstant
- RADIAL BASIS: $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{c})$ für $c > 0$
- NEURONALES NETZWERK: $k(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle + \theta)$ mit $\kappa > 0$ und $\theta \in \mathbb{R}$

angewendet.

Um den Nutzen dieses Tricks zu verdeutlichen, folgt ein **Beispiel:**
Betrachte

- zwei Trainingsdaten (x_1, y_1) und (x_2, y_2), wobei $y_1, y_2 \in \{\pm 1\}$ und $x_1, x_2 \in \mathbb{R}^2$, d.h. $x_1 = (x_{11}, x_{12})$ und $x_2 = (x_{21}, x_{22})$
- polynomieller Kern zweiten Grades mit $c = 1$

Dann gilt:

$$\begin{aligned} k(x_1, x_2) &= (1 + \langle x_1, x_2 \rangle)^2 \\ &= (1 + x_{11}x_{21} + x_{12}x_{22})^2 \\ &= 1 + 2x_{11}x_{21} + 2x_{12}x_{22} + (x_{11}x_{21})^2 + (x_{12}x_{22})^2 + 2x_{11}x_{21}x_{12}x_{22} \end{aligned}$$

Mit $\Phi(x_1) = \Phi((x_{11}, x_{12})) \mapsto (1, \sqrt{2}x_{11}, \sqrt{2}x_{12}, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}x_{12})$ folgert:

$$\langle \Phi(x_1), \Phi(x_2) \rangle = k(x_1, x_2).$$

Damit ergeben sich einige bemerkenswerte **Beobachtungen**:

1. Der Raum $\mathcal{M}$ muss nicht bekannt sein. Die Kern-Funktion als Maß der Ähnlichkeit ist für alle Berechnungen ausreichend.
2. Die Lösung des optimalen Programms ergibt sich also durch Ersetzen des ursprünglichen Skalarproduktes durch die Kern-Funktion.
3. Die Entscheidungsfunktion hat die folgende Form:

$$f(x) = sgn\Big(\sum_{i=1}^{N} \alpha_i y_i k(x, x_i) + b\Big).$$

4 Soft Margin Hyperebene

Wie in den vorhergehenden Abschnitten zu sehen war, gibt es die Möglichkeit die SVM-Methode sowohl im Falle der linearen, als auch im Fall der nicht-linearen Trennbarkeit anzuwenden. Allerdings ist die Überführung in einen höherdimensionalen Raum bei nicht linear trennbaren Daten mit einem erhöhten Rechenaufwand und der Kenntnis einer entsprechenden Kern-Funktion verbunden.
Es stellt sich nun die Frage, ob ein solcher Aufwand im Falle nur einiger weniger Ausreißer nicht umgangen werden kann. Die Theorie der Soft Margin Hyperebene stellt eine entsprechende Möglichkeit dar und hält somit eine Art Zwischenstellung inne.

4.1 Grundlagen

Wie bereits klargemacht wurde ist es in der Praxis sinnvoller, einen Algorithmus zu entwickeln, der eine bestimmte Anzahl an Ausreißern zulässt.
Die hauptsächliche Idee dahinter besteht darin, derartige Fehleinordnungen zu erlauben, aber auch zu *bestrafen*.

Vorgehen:
Im speziellen Fall der SVM wird dies durch eine Abschwächung der Randbedingungen erreicht, d.h. es werden die sogenannten *Schlupfvariablen* $\xi_i \geq 0$ eingeführt, die die Gleichung

$$y_i(\langle x_i, w\rangle + b) \geq 1 - \xi_i \quad \forall i = 1, ..., N$$

erfüllen sollen.
Zudem wird eine Strafe in Form des Kostenterms $\gamma\xi_i$ festgelegt.
γ kann als *Fehlergewicht* interpretiert werden.

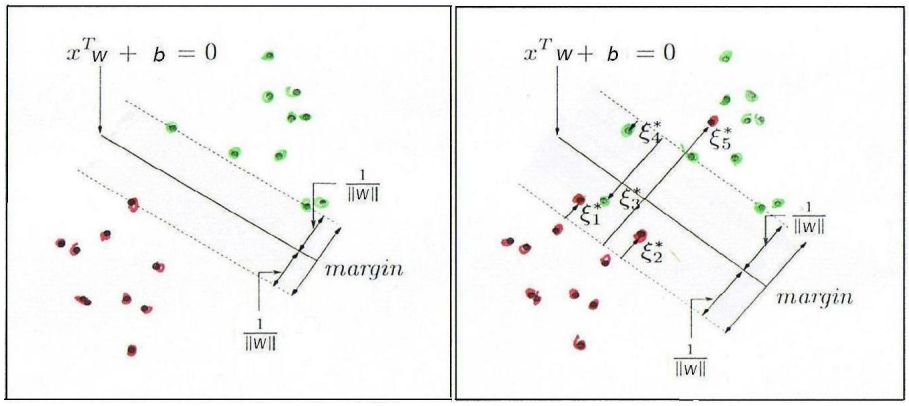

Abbildung 8: Links trennbare Daten und rechts Daten mit Ausreißern

Abbildung 8 verdeutlicht die Rolle der Schlupfvariablen: Korrekt klassifizierte Trainingsdaten besitzen ein $\xi_i = 0$. Daten innerhalb des Randes werden durch $0 < \xi_i \leq 1$ charakterisiert und für alle Punkte auf der falschen Seite von $\mathcal{H}$ gilt $\xi_i > 1$.

Die Schlupfvariablen haben somit die folgende **Interpretation:**
Ein Rand, der die Trainingsdaten korrekt klassifiziert, ist zu bevorzugen. Ist dies nicht möglich (im nicht linear trennbaren Fall), so werden die Nebenbedingung derart abgeschwächt, dass die Strafe proportional zum Ausmaß der Misklassifikation ist.
γ kontrolliert also die Gewichtung zwischen dem konkurrierenden Zielen „breiter Rand mit großen Fehlern“ und „kleine Fehler aber schmaler Rand“.

4.2 Mathematische Ausformulierung

Um bisher benutzte Methoden und Vorgehensweisen anwenden zu können muss der Strafterm nur noch in das Minimierungsproblem aufgenommen werden, d. h.

$$\underset{w\in\mathbb{R}^g,\ b\in\mathbb{R},\ \xi\in\mathbb{R}^N}{\text{minimiere}} \quad \frac{1}{2}\|w\|^2 + \gamma \sum_{i=1}^{N} \xi_i$$

unter den Bedingungen

$$\begin{aligned} \xi_i &\geq 0 \\ y_i(\langle x_i, w\rangle + b) &\geq 1 - \xi_i \quad \forall i = 1, ..., N. \end{aligned}$$

Die Minimierung der Lagrange-Funktion

$$L(w,b,\alpha,\mu) = \frac{1}{2}\|w\|^2 + \gamma\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\langle x_i,w\rangle + b) - (1-\xi_i)) - \sum_{i=1}^{N}\mu_i\xi_i$$

bezüglich w, b und ξ_i ergibt (analog zu oben) die Lösung:

$$\begin{aligned} w &= \sum_{i=1}^{N}\alpha_i x_i y_i \\ 0 &= \sum_{i=1}^{N}\alpha_i y_i \\ \alpha_i &= \gamma - \mu_i \qquad \forall\ i = 1,...,N \end{aligned}$$

wobei die α_i durch Lösen des quadratischen Programmes

$$\underset{\alpha\in\mathbb{R}^N}{\text{maximiere}} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j\langle x_i,x_j\rangle$$

$$0 \le \alpha_i \le \gamma \quad \forall\ i = 1,...,N$$

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

bestimmt werden können.

Betrachtet man die Kuhn-Tucker-Bedingungen

$$\begin{aligned} \alpha_i[y_i(\langle x_i,w\rangle + b) - (1-\xi_i)] &= 0 \\ \mu_i\xi_i &= 0 \\ y_i(\langle x_i,w\rangle + b) - (1-\xi_i) &\ge 0 \qquad \forall i = 1,..,N \end{aligned}$$

so ergeben sich *zwei* mögliche Arten von **Support Vektoren**:

- Punkte direkt auf dem Rand
  (mit $\xi_i = 0$ und daraus folgend $0 < \alpha_i < \gamma$)
- Punkte jenseits ihres Randes
  (mit $\xi_i > 0$ und $\alpha_i = \gamma > 0$).

Bemerkungen:

Besonders hervorzuheben sind in diesem Zusammenhang die folgenden Beobachtungen:
Die Schlupfvariablen verschwinden aus dem dualen Programm und die Konstante γ taucht dort nur noch als zusätzliche Beschränkung der Lagrange-Multiplikatoren α_i auf.
Zudem kann auch im Fall der Soft Margin Klassifikation der Kern-Trick angewendet werden, indem in obigen Termen das Skalarprodukt durch einen geeigneten Kern k ersetzt wird.
Die Entscheidungsfunktion und Verschiebung b bestimmen sich analog zu oben und ermöglichen somit weiterhin die Anwendung des bereits besprochenen SVM-Algorithmus.

Ein entscheidende **Schwachstelle** hat die Soft Margin Klassifikation jedoch.
Bisher wurde keine Aussage über die Wahl von γ gemacht, die jedoch keinesfalls trivial ist. Wird γ groß gewählt, so ist dies gleichbedeutend mit einem hohen Fehlergewicht, welches zu einem kleinen Rand und einer Fokusierung auf Punkte nahe $\mathcal{H}$ führt. Auf der anderen Seite bedeutet ein kleines γ ein schwaches Fehlergewicht, aber einen breiten Rand und die Einbeziehung ferner Punkte. Welche Wahl die Richtige ist lässt sich für den Anwender nicht unbedingt immer erkennen.
Üblicherweise wird das Verfahren der *Kreuzvalidierung* [1] genutzt, um γ zu schätzen.

[1] Bei der Kreuzvalidierung werden in einem ersten Schritt die zugrunde liegenden Daten in p Teile gleicher Größe geteilt. Anschließend werden p Trainingsläufe durchgeführt, wobei jedes mal einer der p Teile weggelassen und zur unabhängigen Beurteilung des auftretenden Fehlers verwendet wird, um den Parameter γ zu optimieren. Im einfachsten Fall wird derjenige Parameter gewählt, der im Durchschnitt über die p Läufe den kleinsten Fehler ergab.

5 Abschließende Betrachtungen

5.1 Multi-Klassen-Einteilung

In der Praxis ist es häufig nötig, Probleme zu lösen, die eine Einteilung in M Klassen (mit $M > 2$) erfordern.
Im Folgenden werden drei Möglichkeiten betrachtet, dies mit SVM umzusetzen.

5.1.1 One Versus the Rest

Bei dieser Methode wird eine Klassifikatoren-Menge $f^1, ..., f^M$ durch jeweiliges Trennen einer Klasse von den Restlichen gebildet. Die Einteilung eines neuen Punktes x erfolgt durch Anwendung der Funktion

$$f(x) := \arg\max_{j=1,...,M} g^j(x), \text{ wobei } g^j(x) = \sum_{i=1}^{N} y_i \alpha_i^j k(x, x_i) + b^j.$$

Hierbei hat es sich als nachteilig erwiesen, dass eine größere „Grauzone“ entstehen kann, d.h. eine Fläche im Raum, in die Punkte eingeordnet werden und die nicht eindeutig einer der Klassen zuzuordnen ist. In einem solchen Fall ist die Klassenauswahl wieder rein zufällig.

5.1.2 Paarweise Klassifikation

Diese Vorgehensweise wird häufig auch „One Versus One“ genannt, denn die Klassifikatoren werden für jedes mögliche Paar von Klassen ($\frac{M(M-1)}{2}$ Stück) gebildet. Die Einordnung eines neuen Datenpunktes erfolgt in diejenige Klasse, die die höchste Anzahl an *Stimmen* (d.h. Klassifikatoren, die den Datenpunkt in diese Klasse einordnen) aufweisen kann.
Obwohl die „Grauzonen-Problematik“ gegenüber One Versus the Rest weniger stark ausgeprägt ist , hat auch dieses Verfahren einen entscheidenden Nachteil: Die Anzahl der benötigten Klassifikatoren (wie erwähnt $\frac{M(M-1)}{2}$) ist sehr viel größer, wodurch sich auch die Trainingszeiten verlängern. Allerdings muss auch gesehen werden, dass die Trainingsprobleme selbst signifikant kleiner sind und somit diesen Nachteil ausgleichen können.

5.1.3 Error-Correcting Output Coding

Durch Aufteilung der ursprünglichen Trainingsdaten in jeweils zwei disjunkte Klassen werden L binäre Klassifikatoren $f^1, ..., f^L$ ermittelt. Die Auswertung eines Datenpunktes anhand aller L Funktionen bestimmt seine Klasse somit eindeutig (d.h. jede Klasse entspricht einem eindeutig bestimmten Vektor in $\{\pm 1\}^L$). Für M Klassen ergibt sich damit die sogenannte *decoding matrix* $\mathcal{D} \in \{\pm 1\}^{M \times L}$ und ein neuer Datenpunkt wird durch Vergleich von dessen L-dimensionalem Vektor mit den Zeilen der Matrix $\mathcal{D}$ einer Klasse zugeteilt.
Auch hier muss abgewägt werden, ob die Methodik auf die vorhandenen Trainingsdaten anzuwenden ist, denn bei überlappenden oder zu kleinen Datensätzen kann keine zuverlässige Schätzung der Klassifikatoren stattfinden und es besteht die Gefahr für einen neuen Datenpunkt x_{neu}, einen Vektor $(f^1(x_{neu}), ..., f^L(x_{neu}))$ zu kreieren, der nicht in der Matrix $\mathcal{D}$ vorhanden ist.

5.2 Vor- und Nachteile der SVM

Abschließend werden an dieser Stelle nochmal alle Stärken und Schwächen der Klassifikation mit Support Vector Machines zusammengefasst.

Vorteile:

1. Eine Klassifikation ist durch Anwendung der SVM sehr schnell möglich, denn die benötigten Parameter basieren nur auf (wenigen) Support Vektoren und nicht auf dem kompletten Trainingsdatensatz.

2. Zudem besitzt die SMV eine hohe Generalisierungsfähigkeit und kann gut auf reale Probleme angewendet werden.

3. Das Arbeiten in hohen Dimensionen wird ebenfalls ermöglicht.

Nachteile:

1. Für neu hinzukommende (verschiedene) Eingabedaten ist jedes mal ein neues Training erforderlich. Es besteht keine Möglichkeit, lediglich die vorhandenen Ergebnisse zu ergänzen.

2. Überdies ist der Umgang mit nicht linear separierbaren Problemen trickreich, denn die Überführung der Daten in einen höherdimensionalen Raum setzt die Kenntnis der Größe der benötigten Dimension voraus.

3. Zudem stellt sich in diesem Fall die Frage nach der Wahl des Kernels, der empirisch gesucht werden muss.

Zusammenfassend lässt sich sagen, dass die Support Vector Machines zwar durchaus einige entscheidende Nachteile aufweisen, dennoch aber ein gutes Mittel darstellen, um im Rahmen des statistischen Lernens verschiedenste Problematiken lösen zu können.

Literatur

[1] T. Hastie, R. Tibshirani, J. Friedman. *The elements of statistical learning*. Springer, 2001. Kapitel 12.

[2] B. Schölkopf, A. Smola. *Learning with kernels*. MIT Press, 2002. Kapitel 7.

[3] http://www.wikipedia.org/wiki/Support-Vector-Machine