Robert Hennings 16.09.2020

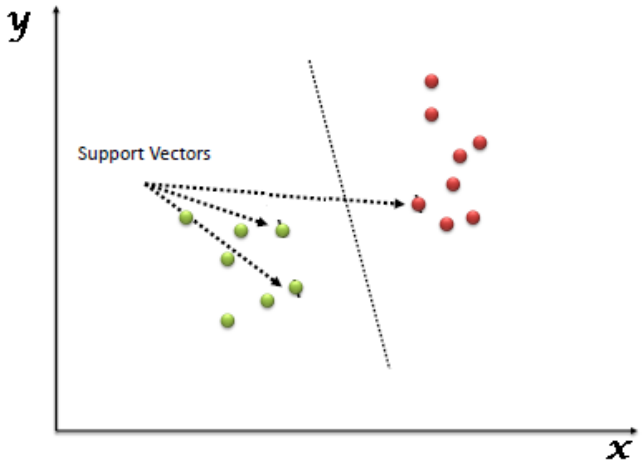
Handout Support Vector Machine Algorithmus (SVM)

Zielsetzung und Idee

Ziel des SVM Algorithmus (**Supervised** Machine Learning): Klassifikation von Daten in verschiedene Gruppen, nicht primär wie anderweitig die Voraussage von (Daten) Entwicklungen jedoch kann der SVM auch dafür benutzt werden.

Dies wird über das Finden einer sogenannten „Hyperebene“ erreicht, was ein allgemeiner Ausdruck für eine Ebene im Raum ist. Wichtig dabei ist das dieses Konzept auf Räume beliebiger Dimensionen anwendbar ist.
Im SVM soll diese Ebene im Raum für N Dimensionen gefunden werden, wobei N für die Anzahl der „Features“ bzw. Attribute des Datensatzes steht.

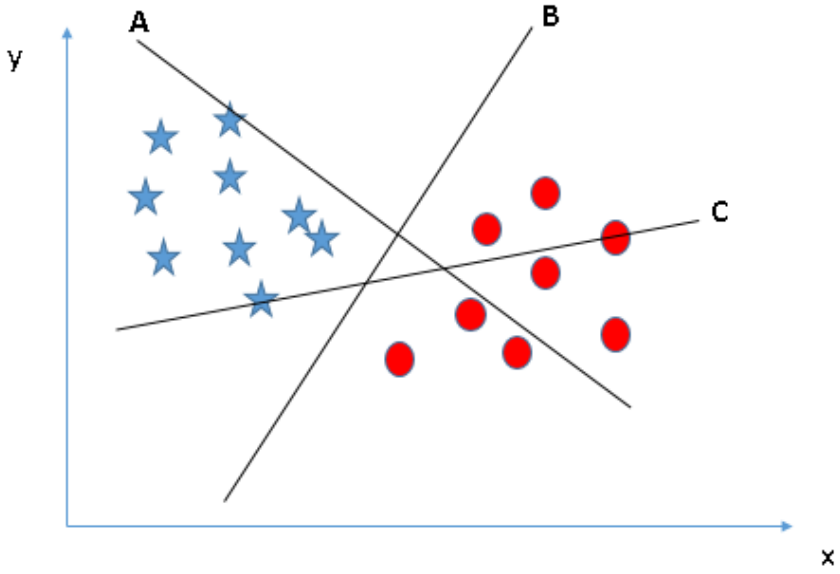
Im Folgenden könnte sich auf verschiedene Dimensionen bezogen werden, es wird sich zunächst auf den zweidimensionalen Fall bezogen der in einem normalen XY Diagramm abgetragen werden kann durch Einzeichnung der Datenpunkte. Jeder einzelne eingetragene Datenpunkt kann durch seinen einzelnen eigenen Stützvektor ganz genau bestimmt werden. Dies kann dann wie folgt aussehen:



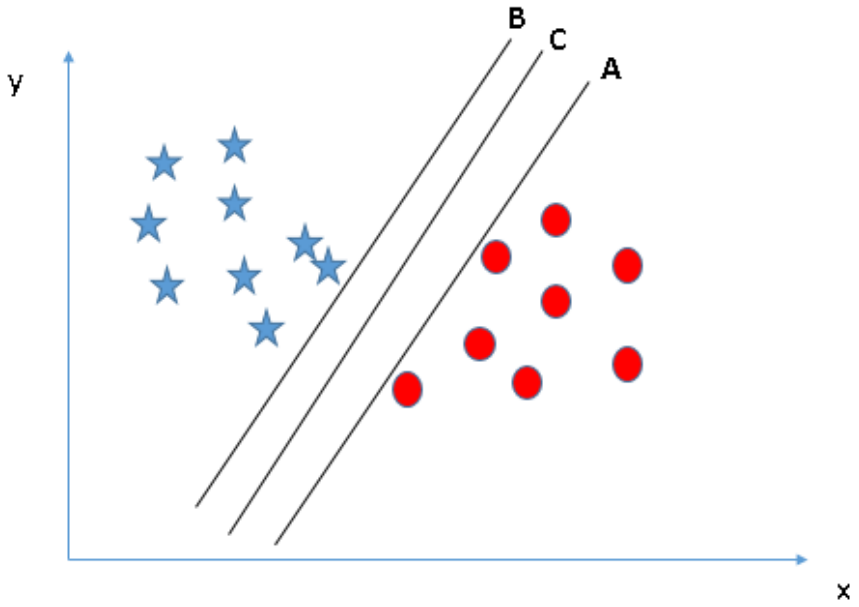
Jeder einzelne Datenpunkt besitzt zwei ihn charakterisierende Merkmale (Y und X), sind beide genau bekannt so kann die Lage des Datenpunkts genau im Diagramm bestimmt werden, genauso dann entsprechend weiter für weitere Attribute die hinzukommen.
Dort soll nun die optimale Ebene im zweidimensionalen Raum gefunden werden, diese wird erzielt durch den maximalen Abstand von der Ebene zu beiden Arten der Datenpunkte (in der Mitte soll ein möglichst weiter freier Bereich entstehen), sie müsste demnach genau „in der Mitte“ liegen um den maximalen Abstand zu beiden Datenpunkten zu gewährleisten.

Verschieden Beispiele der Hyperebene

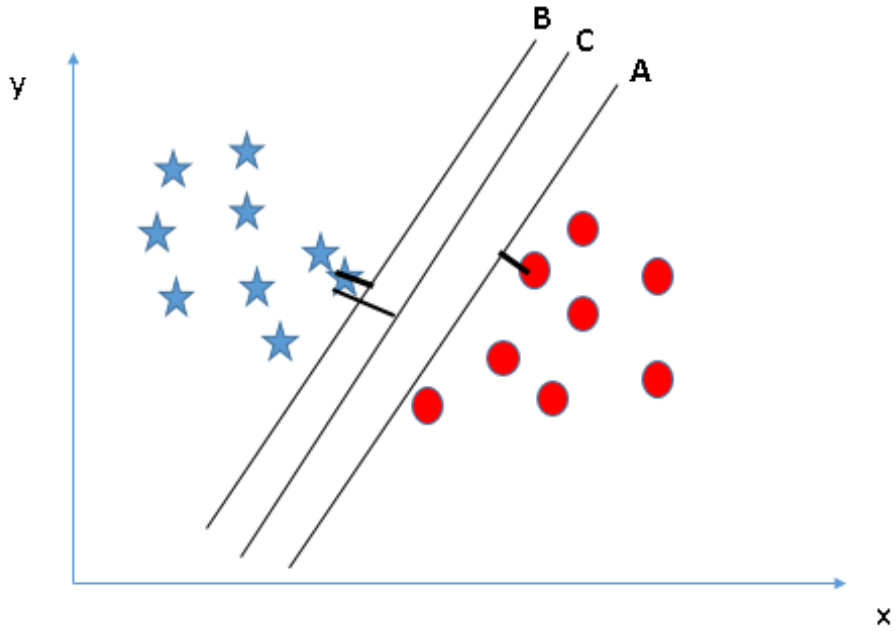
Dies wird nochmal anhand eines einfachen Beispiels veranschaulicht:
Welche Hyperebene trennt hier optimal die beiden unterschiedlichen Klassen:

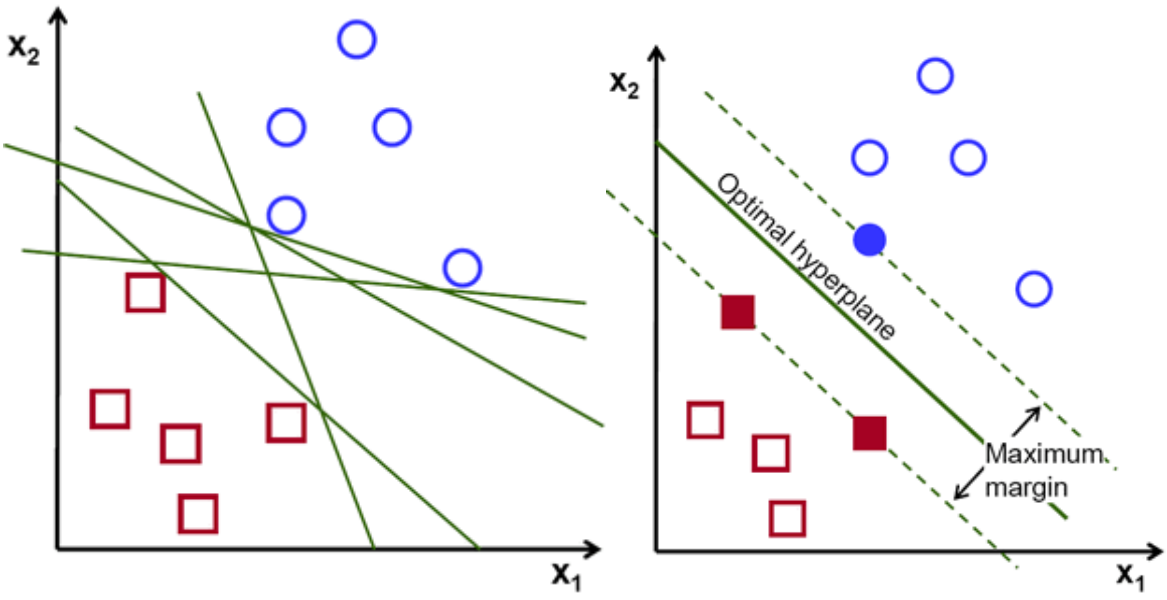


Weiterführend soll folgendes Beispiel betrachtet werden, bei dem wieder die optimale Hyperebene zu unterscheiden ist:

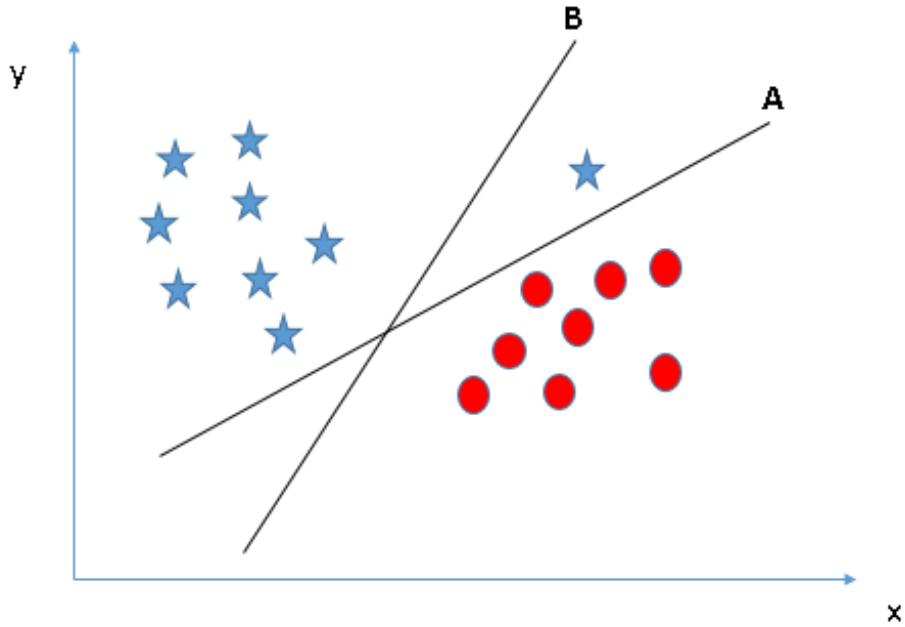


Dieser freigelassene Rand (maximaler Abstand zu den Datenpunkten) erleichtert es später Datenpunkte zu klassifizieren die nicht genau den Trainingsobjekten entsprechen.
Daher ist die richtige Hyperebene hier C:



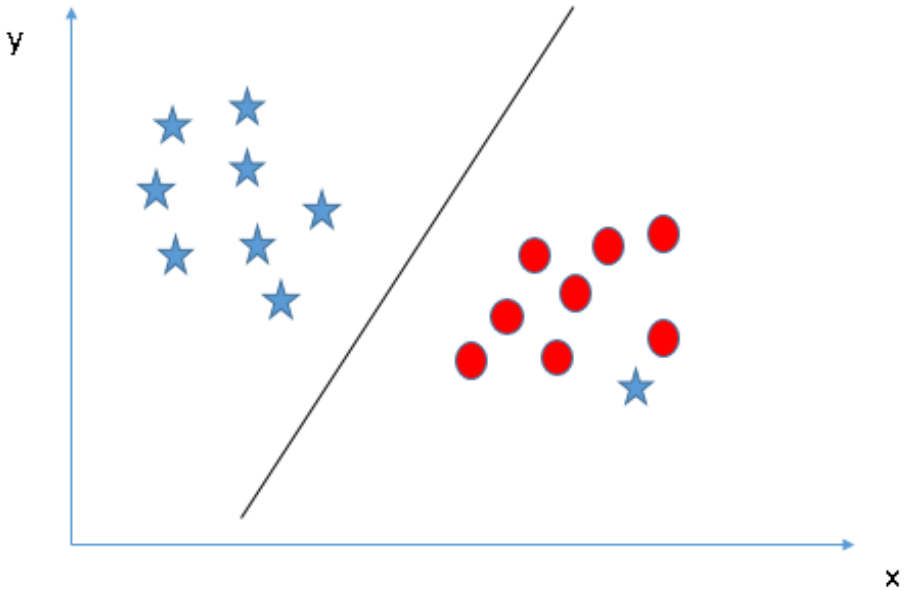


Ist dann erst diese „Trennungsebene“ generiert worden ist es später einfacher Daten entweder in die eine oder die andere Kategorie einzuordnen allein aufgrund ihres Abstandes.
Aber Achtung dies ist vielleicht nicht immer so offensichtlich wie es auf den ersten Blick scheint:



Hier wäre A die richtige Antwort, B würde einen Klassifizierungsfehler produzieren (siehe der einsame Stern oben rechts auf der falschen Seite, dies darf nicht passieren daher gilt hier der Grundsatz: Richtige Klassifizierung geht vor maximalem Abstand (“Margin“).

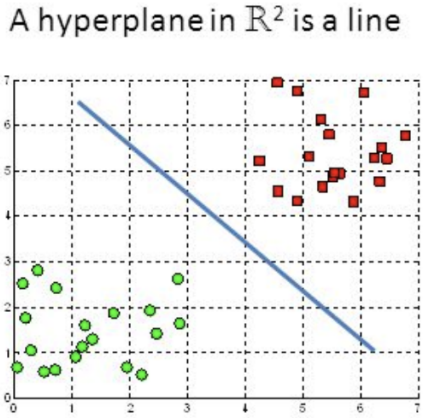
Hinzu könnte man sich auch den folgenden Fall denken:

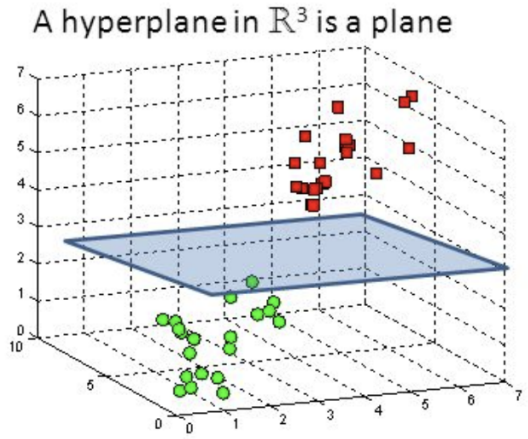


Der SVM ist gegen solche Ausreißer jedoch robust und kann sie ignorieren um trotzdem ein valides Ergebnis zu liefern.

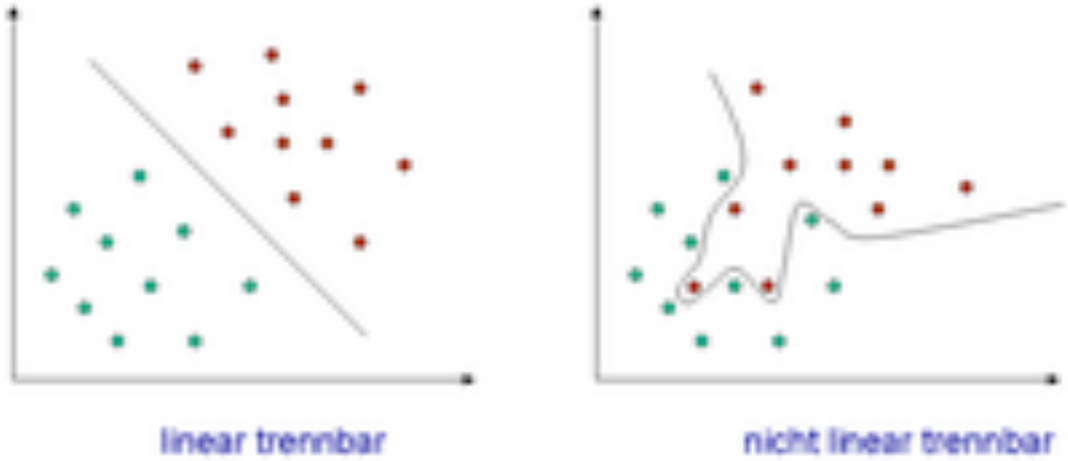
Hyperebene im N dimensionalen Raum
Diese Ebene im Raum, im zwei dimensionalen Raum, ist eine Gerade, wies sieht das Ganze dann aber bei mehreren Kategorien aus in die der Datensatz eingeteilt werden, also >= 2?

Dementsprechend verändert sich auch der Raum und somit die Konstruktion der Ebene:





Ebenso muss nicht immer zwingend ein linearer (einfacher) Zusammenhang wie hier im Beispiel vorliegen, dieser kann auch genauso gut nicht linear sein, was dann so aussehen würde:

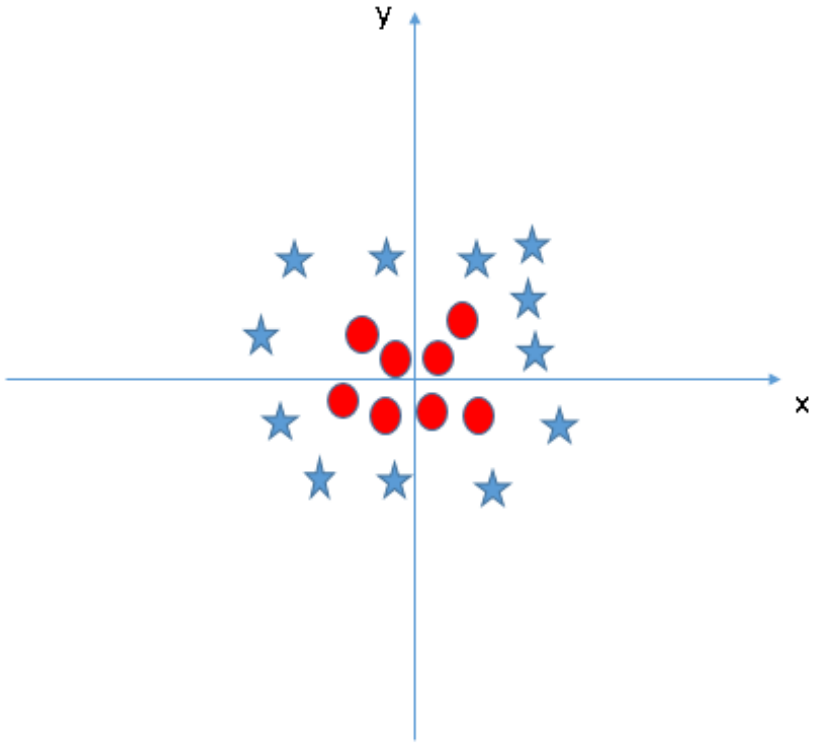


Um dennoch eine Trennlinie einzeichnen zu könne wird der Kernel Trick verwendet.

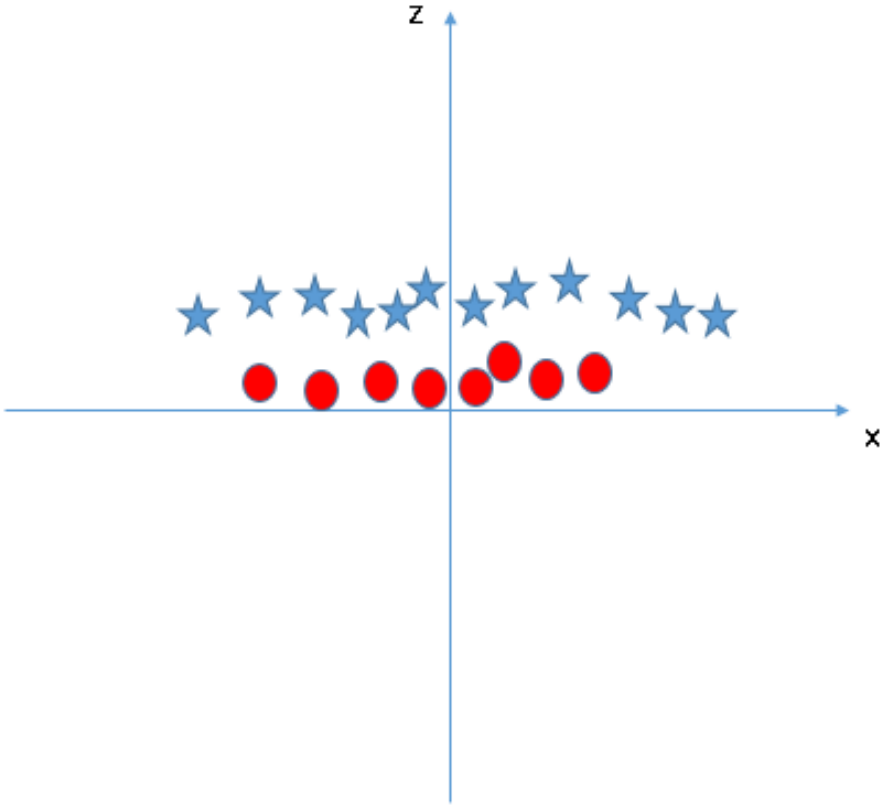
Exkurs Kernel Trick

Die Idee besteht darin den Vektorraum und damit auch die darin befindlichen Trainingsvektoren in einen höherdimensionalen Raum zu überführen. In einem Raum mit genügend hoher Dimensionsanzahl (im Zweifelsfall sogar unendlich) wird auch die verschachtelste Vektormenge linear trennbar. Dies muss als Voraussetzung gelten damit die Trennlinie bestimmt werden kann, dann wird dies, wenn diese besteht, rücktransformiert. In diesem höherdimensionalen Raum wird nun die trennende Hyperebene bestimmt. Dann wird das Ganze wieder rücktransformiert, in den niedriger dimensionalen Raum, wodurch die lineare Hyperebne zu einer nichtlinearen sogar nicht zusammenhängenden Hyperfläche die die Trainingsdatenpunkte dennoch in zwei abgrenzbare Klassen einteilt.

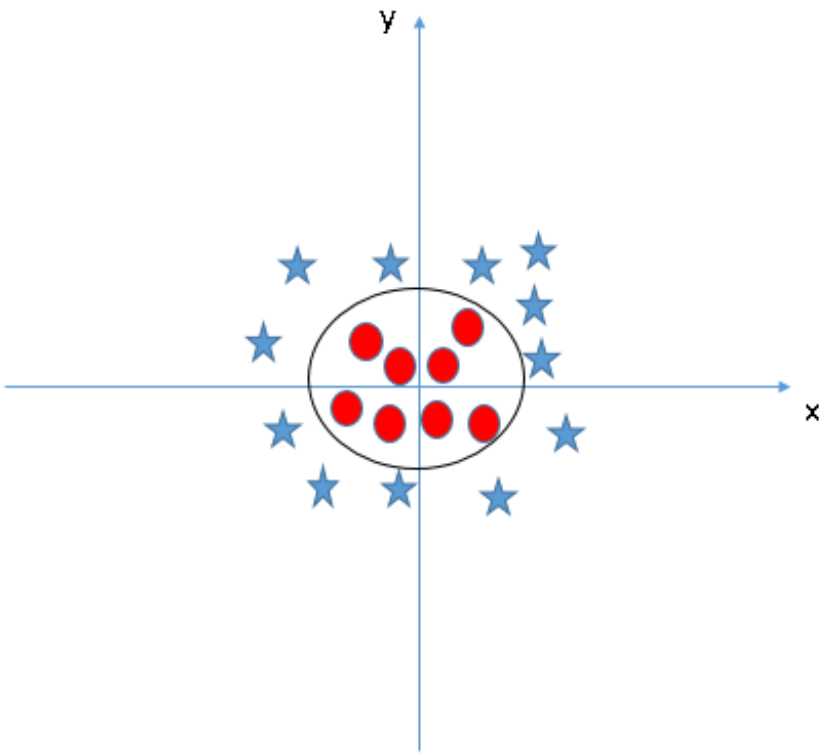
Um dies nochmal zu veranschaulichen, betrachten wir folgende Situation:



Wie kann nun dort die optimale trennende Hyperebene konstruiert werden?
Die Datenpunkte bzw. deren Dimension muss verändert werden mit: Z = x^2+y^2
Dann werden die veränderten Datenpunkte erneut, diesmal an der X-Z Achse geplottet:



Nun lässt sich das Problem durch die Hochskalierung deutlich leichter lösen.
Im Originalzustand würde die optimale Hyperebene wie folgt aussehen:



Probleme des Kernel Tricks

1) Hochtransformation in höhere Dimensionen ist sehr rechenaufwendig
2) Darstellung der Trennflächen im niedrigdimensionalen Raum sehr komplex damit praktisch unbrauchbar

Verwendet man hier aber den Kernel Trick und nutzt zur Beschreibung der Trennfläche geeignete Kernel Funktionen, die im hochdimensionalen die Hyperebene beschreiben und trotzdem im niedrigdimensionalen „guatartig“ bleiben, so ist die Hin- und Rücktransformation möglich, ohne sie tatsächlich rechnerisch ausführen zu müssen. Dafür genügt der Teil der Vektoren, die die Ebene aufspannen.

Praktische Ausgestaltung eines Kernels

```
Lineare Gestaltung:
K(x, xi) = sum(x * xi)
```

Polynom Gestaltung:

```
K(x,xi) = 1 + sum(x * xi)^d
```

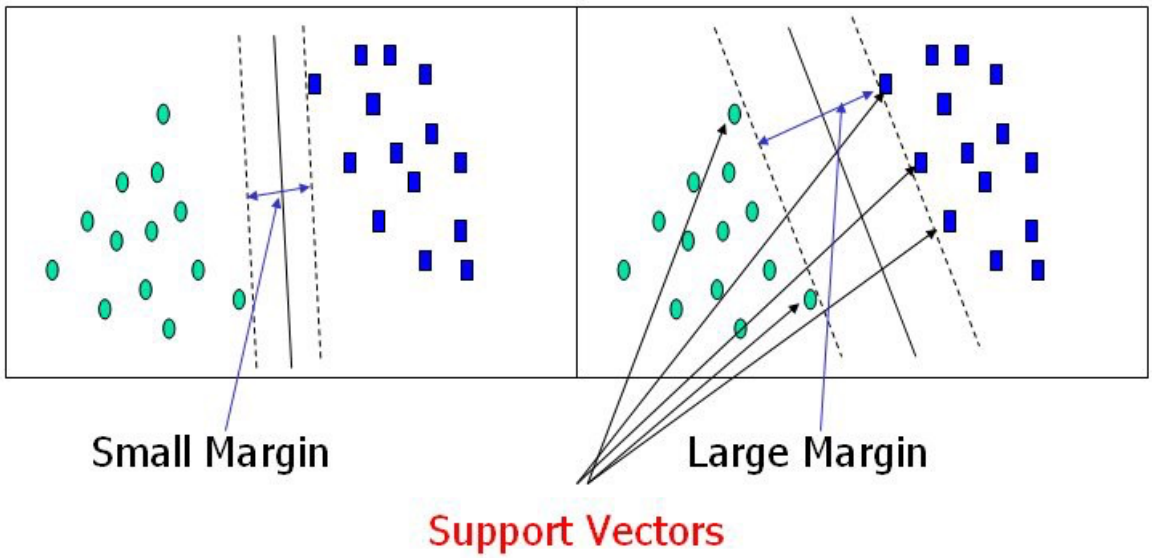
Radial Basis Function:

```
K(x,xi) = exp(-gamma * sum((x - xi^2))
```

Die konstruierten Ebenen stellen also eine Entscheidungshilfe dar, die es erleichtern Daten in bestimmte Klassen voneinander abgegrenzt einzuteilen.
Wie zu sehen war, die Anzahl der „Features“ gibt die Dimension vor. Bei N=2 also im zweidimensionalen Raum ist die Ebene lediglich eine Gerade, bei N=3 jedoch im dreidimensionalen Raum ist die Ebene eine zweidimensionale Ebene, keine Gerade mehr.

Nun ist die Frage, z.B. anhand eines N=2 Raumes wie diese Ebene konstruiert werden kann. Ganz allgemein wird eine Ebene mithilfe von Stützvektoren aufgespannt die die Eckpunkte

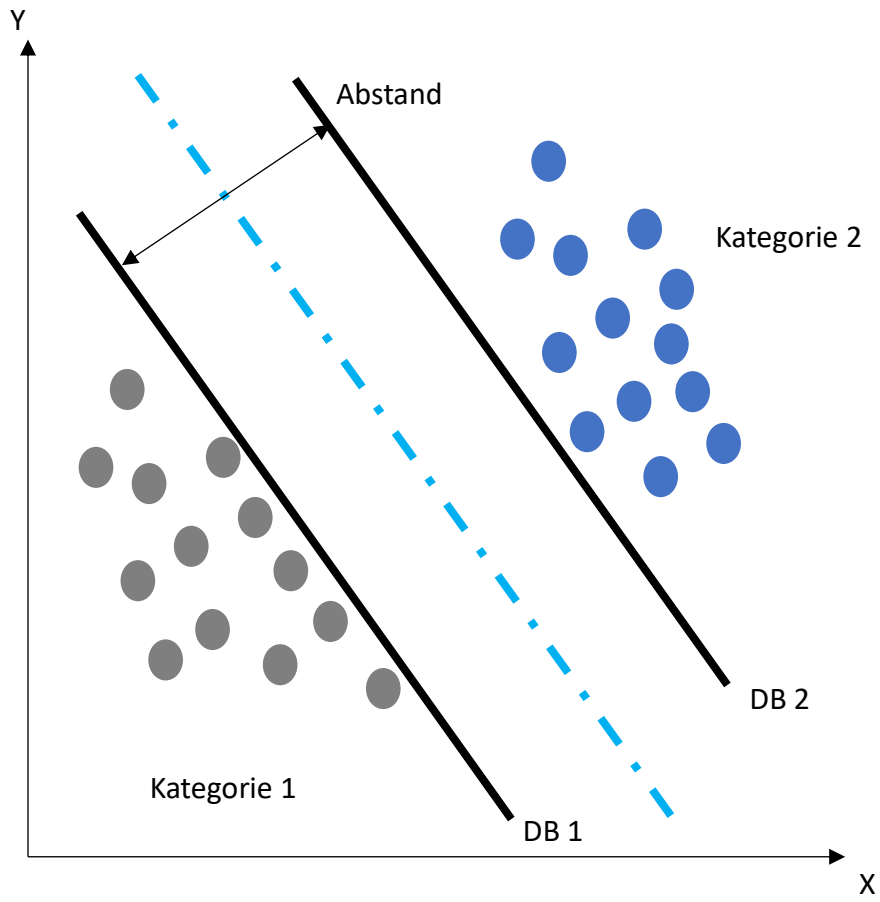
der Ebene als Ziel haben. Sie bestimmen also die einzelnen Ecken aus denen dann die Ebene gezeichnet bzw. konstruiert werden kann.



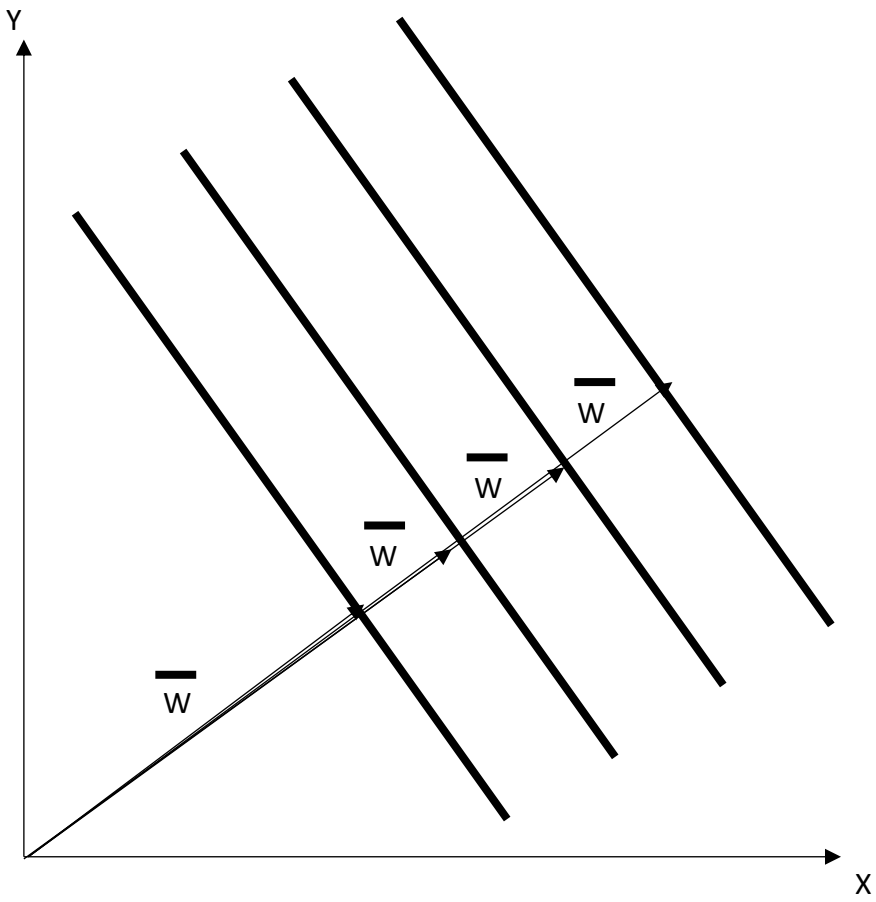
Diese unterstützenden Punkte die man also zunächst für die Konstruktion der Ebene benötigt können aus dem Datensatz, genauer den einzelnen Datenpunkten abgelesen werden, da wenn wir uns erinnern der Abstand ja zu jeder „Seite“ maximal werden soll, es werden also Datenpunkte mit den Kanten der Ebene „angeschnitten“
Die letztendlichen Support Vektors sind die einzelnen Datenpunkte, die am nächsten zur Ebene liegen, diese aufspannen und deren Form beeinflussen.

Konzept der „Large Margin“
Das Outputergebnis einer linearen Funktion wie sie z.B. hier im zweidimensionalen Fall benutzt werden würde oder auch im einfachen Fall einer linearen Regression die aus einer ähnlichen einfachen linearen Funktion besteht, wird auf einen bestimmten Bereich normiert der die Einteilung erleichtert:
Im SVM ist dies der normierte Bereich [-1:1], liegt also ein Outputergebnis jenseits einer bestimmten Schwelle lässt es sich bereits in eine der Klassen einordnen, die Schwell-/bzw. Grenzwerte sind hier die bereits angesprochene Wertgrenze.

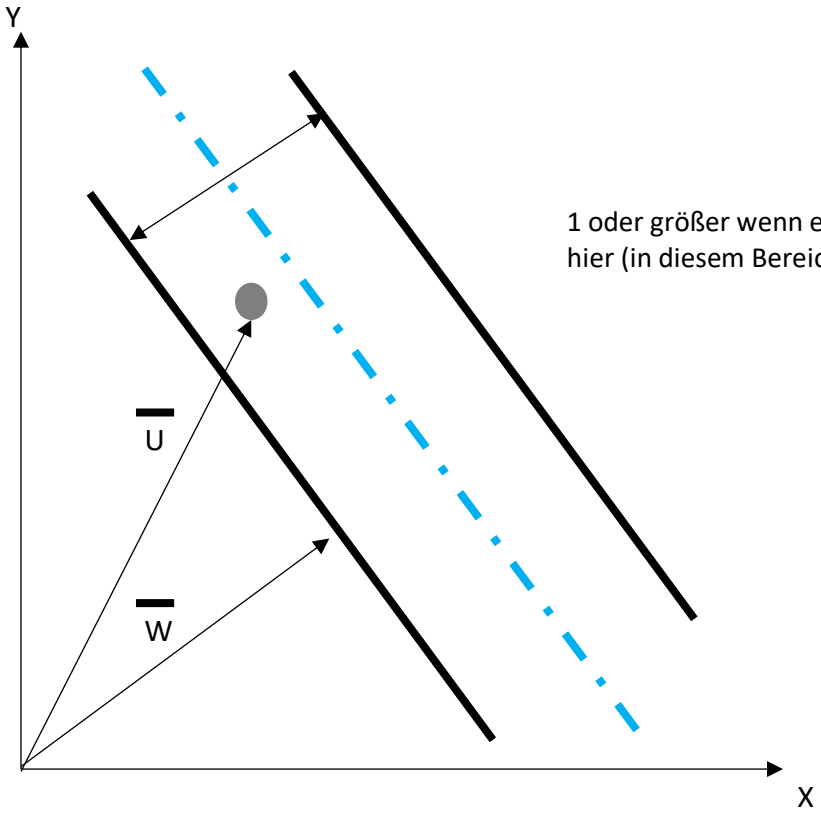
Mathematisches Konzept
Ziel ist es die „Margin“, der Abstand zwischen den Datenpunkten zu maximieren, um eine einfachere Klassifizierung zu ermöglichen.
Die Datenpunkte liegen in der Ausgangssituation erstmal frei im zweidimensionalen Raum vor und sollen durch die Hyperebene (erstmal linear angenommen) getrennt werden.
Es braucht also wie schon vorher besprochen eine „Decision boundary“ um dies zu entscheiden:



Die DBs sollen die Decision Boundaries darstellen. Diese, als Geraden, werden durch einen Stützvektor und einen weiteren Richtungsvektor konstruiert. Dieser soll als W bezeichnet werden, dessen genaue Länge jedoch unbekannt ist und bestimmt werden muss um die Gerade zu konstruieren und in dem Schritt dies mit dem maximalen Abstand für die Kategorisierung.
Es muss dafür jedoch erstmal die Länge des Normalenvektors gefunden werden, da es an sich unendlich viele gibt. Die genaue Länge ist ja nicht weiter spezifiziert:



Es muss also eine bestimmte Annahme getroffen werden, um zunächst das Chaos an Datenpunkten auf einen bestimmten Bereich zu normieren.



1 oder größer wenn ein Datenpunkt z.B. hier (in diesem Bereich) liegt

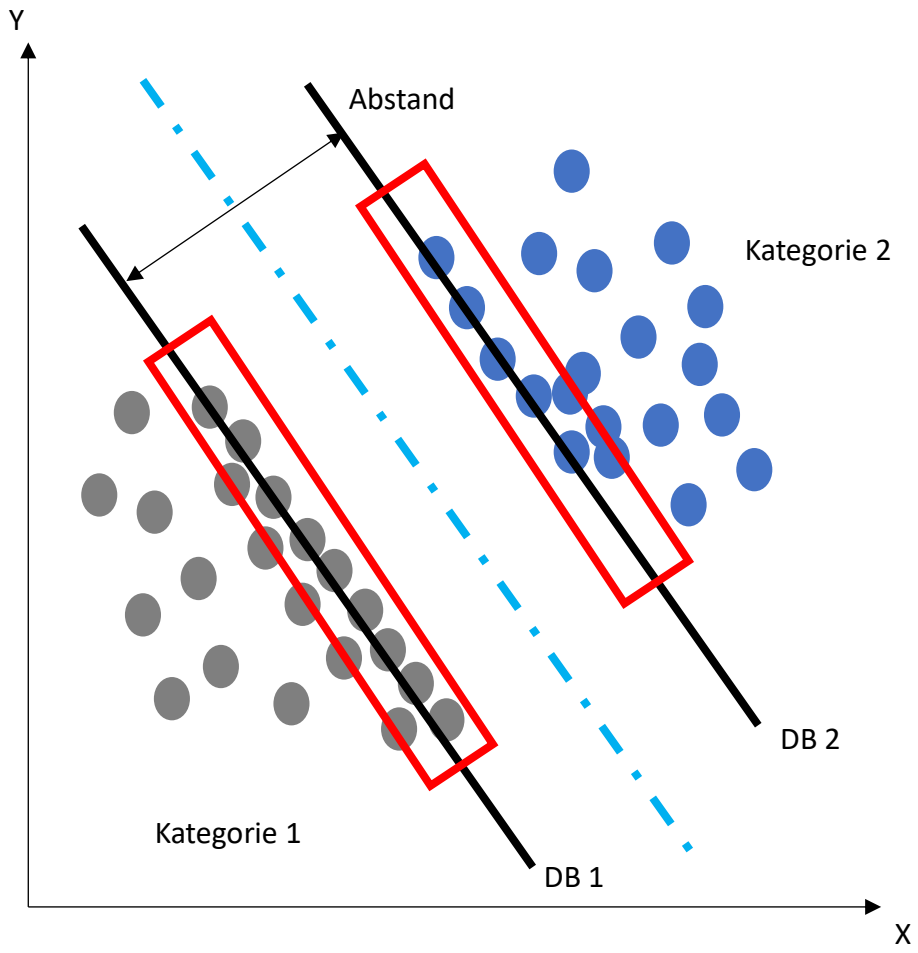
-1 oder kleiner wenn ein Datenpunkt z.B. hier (in diesem Bereich) liegt

Die bereits aufgestellte Normierung wird vereinfacht mit Hilfe der Variablen Yi die eine dichotome Variable darstellt: Yi = 1 für alle positiven Samples, Yi = -1 für alle negativen Samples. Dies soll als Funktion dargestellt werden, um diese zunächst besser auseinander halten zu können:

$$1)\ w \cdot \overrightarrow{X_+} + b \geqq 1$$
$$2)\ w \cdot X_+ + b \leqq 1$$

$$1)\ y_i \cdot (x_i \cdot \vec{w} + b) \geqq 1$$
$$2)\ y_i \cdot (x_i \cdot \vec{w} + b) \leqq 1$$

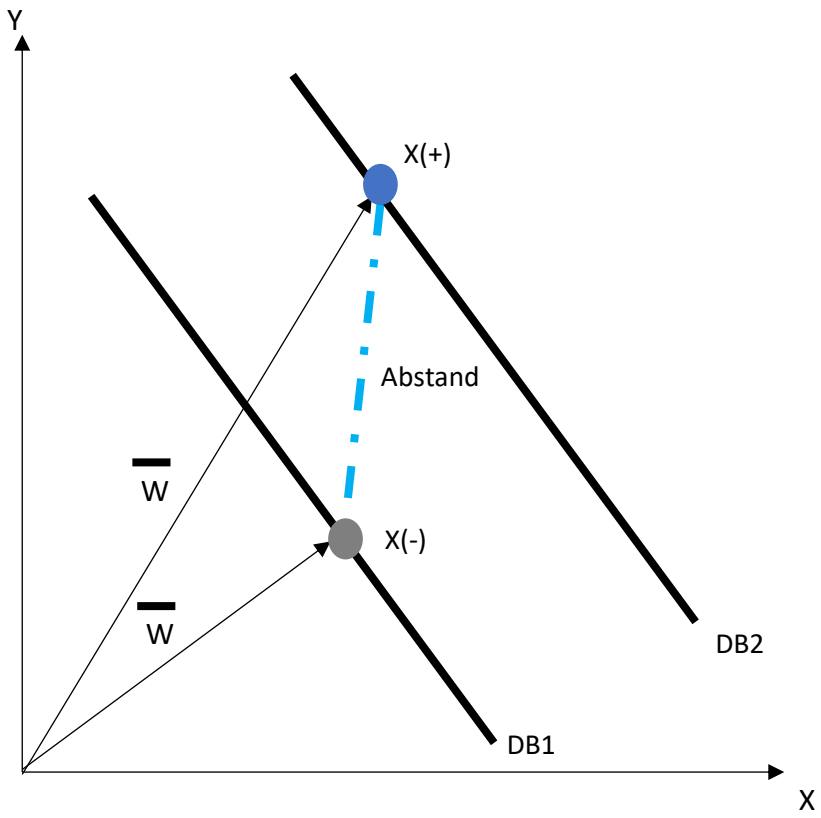
$$1)\ y_i \cdot (x_i \cdot \vec{w} + b) - 1 = 0$$
$$2)\ y_i \cdot (x_i \cdot \vec{w} + b) - 1 = 0$$



Bereich ist also normiert von -1 bis +1.

Zu sehen ist also, je nachdem ob ein Datenpunkt oberhalb oder unterhalb der Trennebene liegt, so hat er auch ein positives oder negatives Vorzeichen, das ihn zuteilt in die eine oder die andere Klasse. Die Datenpunkte der einen Klasse sollen also alle ein positives Vorzeichen erhalten, die der anderen Klasse alle ein negatives Vorzeichen.

Die Definition der Nebenbedingung ist bis hierhin nun abgeschlossen, weiter geht es mit der Formulierung der Hauptbedingung.
Der Abstand kann also auch alternativ gesehen werden als:



Abstand angegeben in Einheiten des Normalenvektors: $(X_+ - X_-) \cdot \frac{\vec{w}}{\| w \|}$

Dieser Bruch ist der Abstand angegeben in Einheiten des Einheitsvektors, bzw. der Abstand wurde so normiert um die Hyperebene zu konstruieren, denn dem Normalenvektor w muss durch Multiplikation ja seine spezifische Länge gegeben werden.

Zur Wiederholung:

Formulierung der Hauptbedingungen

Abstand als: $(X_+ - X_-) \cdot \frac{\vec{w}}{\| w \|}$

$y_i = 1$ für + Datensätze

Nebenbedingung: 1) $y_i \cdot (x_i \cdot \vec{w} + b) - 1 = 0$

(...) auflösen:

$(x_i \cdot \vec{w}) = 1 - b$ ⟷ $\overrightarrow{w_+} \cdot \frac{\vec{w}}{\| w \|}$

Vergleich zeigt: $\overrightarrow{x_+} = 1 - b$

Gleiches gilt für Gleichung 2):

$\overrightarrow{x_-} = 1 + b$

Es lässt sich einsetzen: $(1 - b - 1 + b) \cdot \frac{\vec{w}}{\| w \|}$

Vereinfachung:

Es bleibt: $\frac{2}{\| w \|}$

w^2 heißt hier **minimale quadratische Norm**

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Wie zu sehen ist, ist dies eine leichte Abwandlung der obigen Funktion nur dass f(x) nun durch den Part mit xi und dem zugeteilten w ausgedrückt wird die die zentralen Inputparameter der Funktion bilden. Lambda ist einfach der Parameter im Sinne des Lagrange Verfahrens.

Diese Funktion wird nun nach dem Normalenvektor (w) abgeleitet (partiell).

Genauer ausformuliert:

Formulierung der Hauptbedingungen

Optimierungsproblem unter Nebenbedingungen: Lagrange Verfahren

$$L = \frac{1}{2} \cdot (\parallel w \parallel)^2 - \sum \alpha_i \cdot [y_i \cdot \vec{w} \cdot x_i + b - 1]$$

Einzelne partielle Ableitungen nach den Parametern bilden

$$\frac{\delta L}{\delta w} = \vec{w} - \sum \alpha_i \cdot y_i \cdot x_i$$

$$\vec{w} = \sum \alpha_i \cdot y_i \cdot x_i$$

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 = 2\lambda w_k$$

Einsetzen in Lagrangefunktion anstatt w

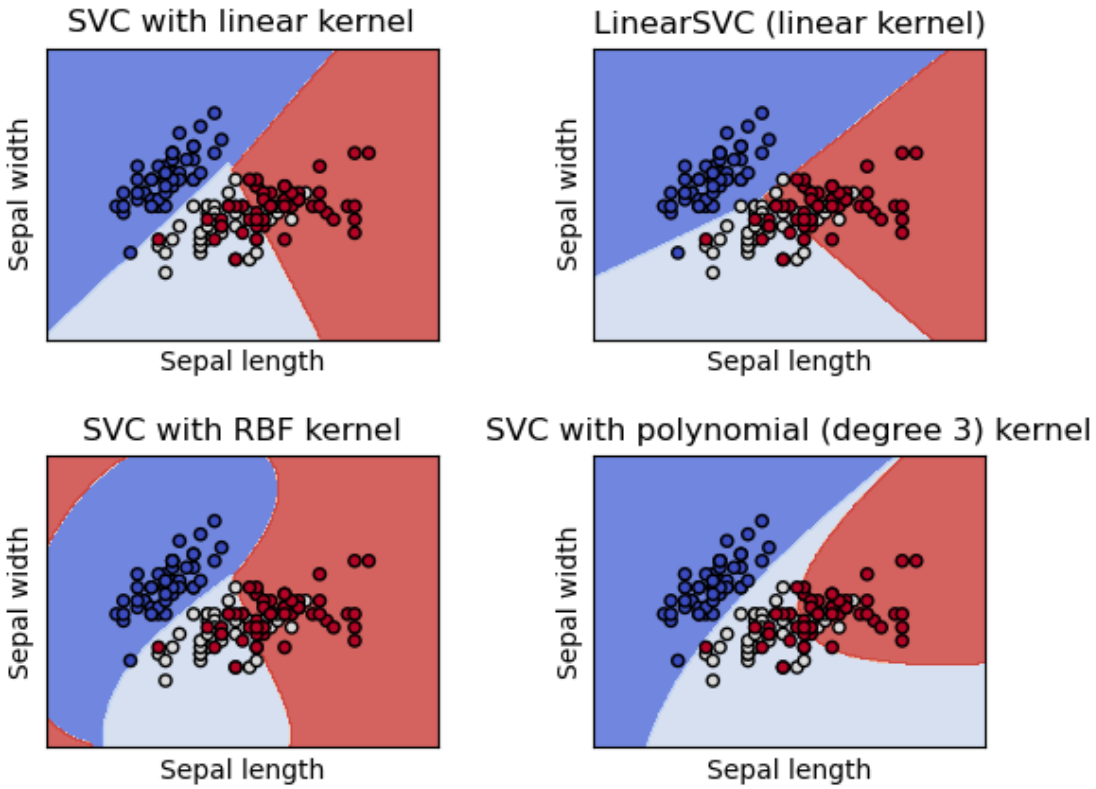
$$L = \frac{1}{2} \cdot (\sum \alpha_i \cdot y_i \cdot \vec{x_i}) - (\alpha_i \cdot y_i \cdot \vec{x_i}) - (\sum \alpha_i \cdot y_i \cdot x_i) \cdot (\sum \alpha_j \cdot y_j \cdot x_j) - (\sum \alpha_i \cdot y_i + b) + \sum \alpha_i$$

Nach Vereinfachung bleibt: $L = \sum \alpha_i - \frac{1}{2} \cdot \sum_i \sum_j \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot x_i \cdot x_j$ Dies nun maximieren!

Y ist immer vorgegeben da entweder 1 oder -1, x ebenso als die einzelnen Datensatzinhalte, jedoch muss der Parameter Alpha durch den Algorithmus optimal bestimmt werden.

Diese Funktion wird jetzt vom Algorithmus optimiert.

Implementierung als Code in Python
Verschiedene **Kernelfunktionen** für die Darstellung unterschiedlicher Dimensionen:



Das verwendete Basisdatenset stellt wieder das Iris Set dar, welches in seiner Grundform drei Klassen beinhält. Eine dieser Klassen lassen wir weg damit wir ein anschaulicheres und einfacheres zweidimensionales Problem haben. Es müssen also zwei Klassen voneinander unterschieden werden.
Die genaue Art der Trennung (ob Linie, geschwungen oder sonst wie) hängt von der Transformationsvorschrift der definierten Kernel Funktion ab.

Anpassung von Hyperparametern in der umgesetzten Codeform in Python

1) Kerneltransformation
   Dort gibt es sehr verschiedene Varianten und Formen die je nach Problemstellung angewendet werden können, dementsprechend andere Parameter enthalten. Linear, Polynome, RBF uvw.
2) Regularization Parameter
   Dies ist der Parameter C bzw. Konstante b, die die Decision Boundary bildet als Entscheidungsgrenze in welche Kategorie der Datensatz fällt. Ein kleineres C hat eine kleinere Margin (Abstand) zur Folge et vice versa.
3) Gamma
   Ein kleinerer Gammawert fittet die Traningsdaten nur sehr wage, ein höherer Gammawert fittet die Trainingsdaten sehr genau (Achtung: Overfitting).

Quellen

1) https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

2) https://de.wikipedia.org/wiki/Support_Vector_Machine

3) https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
4) https://de.wikipedia.org/wiki/Gradient_(Mathematik)
5) https://de.wikipedia.org/wiki/Kernel_(Maschinelles_Lernen)