Sequential minimal optimization: A fast Algorithm for Training Support Vector machines

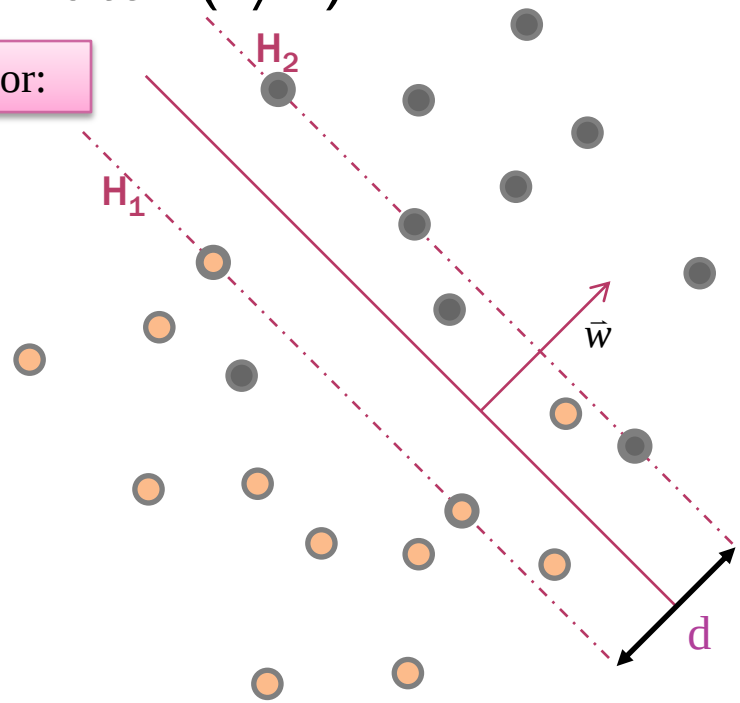
By John C. Platt (1998)

Referat von Joerg Nitschke

Fall der nicht-trennbaren Trainings-Daten (1/2)

In der Realität kommen linear nicht-trennbare Daten eher vor:

Wie können die erarbeiteten Konzepte dennoch weiterhin benutzt werden?



=> Relaxation der Bedingungen, d.h. Einführung einer Schlupf-Variable ξ_i - jedoch nur da, wo der Fehler auftritt:

Für $y_i = +1$:

$$x_i \cdot w + b \geq +1 - \xi_i$$

Für $y_i = -1$:

$$x_i \cdot w + b \leq -1 + \xi_i$$

Falls also ein Fehler auftritt, muß ξ_i 1 überschreiten;

Eine Obergrenze für die Anzahl der Trainings-Fehler ist: $\sum_{i=1}^{l} \xi_i$

mit:

$$\xi_i \geq 0 \quad \forall i$$

Fall der nicht-trennbaren Trainings-Daten (2/2)

Statt des ursprünglichen O.P.: $\frac{1}{2}\|w\|^2$
minimieren wir nun $\frac{1}{2}\|w\|^2 + C(\Sigma_i \xi_i)^k$

wobei C ein vom Benutzer wählbarer Parameter zur Gewichtung der Strafterme ist.

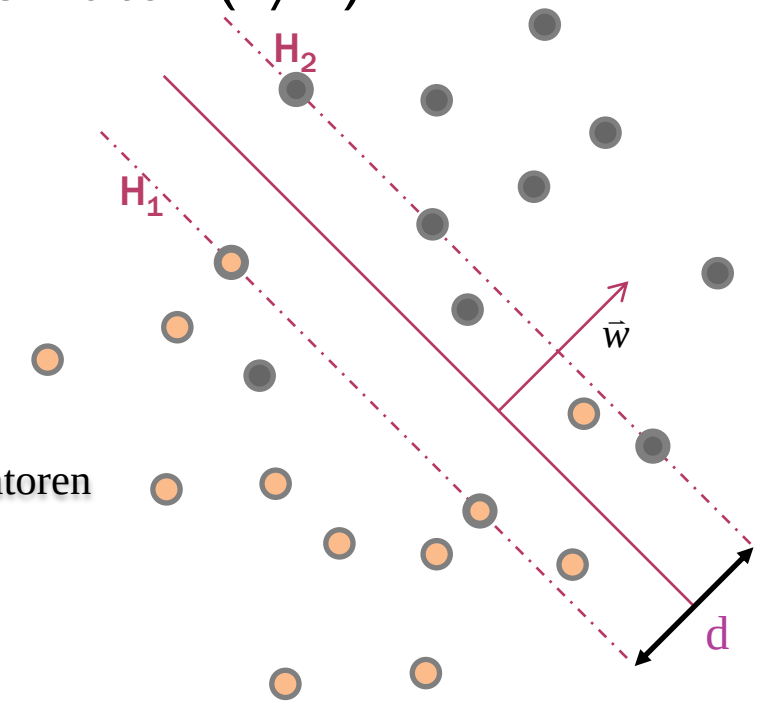
Für beliebige k: Weiterhin konvexes Problem
Für k={1,2}: QP-Problem
Für k=1: Weder ξ_i noch deren Lagrange-Multiplikatoren tauchen im dualen O.P. auf:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j$$

N.B.: $0 \le \alpha_i \le C$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

Die Lösung ist also wie bereits gegeben:

$$\vec{w} = \sum_{i=1}^{|SV|} \alpha_i y_i x_i$$



Für $y_i = +1$:
$x_i \cdot w + b \ge +1 - \xi_i$
Für $y_i = -1$:
$x_i \cdot w + b \le -1 + \xi_i$
mit:
$\xi_i \ge 0 \quad \forall i$

KKT-Bedingungen mit Schlupf-Variablen ξ_i

Das primale O.P. ändert sich $\forall$i zu:

$$L_p \equiv \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i - \sum_{i=1}^{l}\alpha_i\{y_i(x_i \cdot w + b) - 1 + \xi_i\} - \sum_{i=1}^{l}\mu_i\xi_i$$

Geänderte KKT-Bedingungen:

(1) $\frac{\partial}{\partial w_\nu} L_p = w_\nu - \sum_{i=1}^{l}\alpha_i y_i x_i = 0$ mit $\nu = 1,\ldots,d$ (Dimension)

(2) $\frac{\partial}{\partial b} L_p = \sum_{i=1}^{l}\alpha_i y_i = 0$

(3) $\frac{\partial}{\partial \xi_i} L_p = C - \alpha_i - \mu_i = 0$

(4) $y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0$ mit i = 1,…,l

(5) $\mu_i \geq 0$ $\alpha_i \geq 0$ $\forall$i

(6) $\alpha_i\{y_i(x_i \cdot w + b) - 1 + \xi_i\} = 0$ $\forall$i

(7) $\mu_i\xi_i = 0$ $\forall$i

Hinzu kommt ein neuer Lagrange-Multiplikator μ_i für die Schlupfvariable ξ_i

Super!: In den Gradienten nach w_ν und b tauchen weder μ_i noch ξ_i .

Anhand des Gradienten nach ξ_i und letzter Bedingung sieht man, dass

$\xi_i = 0$, wenn $\alpha_i < C$

Daher kann beliebiges $\alpha_i < C$ (mit $\xi_i = 0$) gewählt werden, um b zu berechnen.

3. Lineare SVMs

Spätere Bedeutung für unsere Trainingsbeispiele in SMO:

Unser QP-Problem ist gelöst wenn für alle i gilt:

$$\alpha_i = 0 \Leftrightarrow y_i u_i \geq 1$$
$$0 < \alpha_i < C \Leftrightarrow y_i u_i = 1$$
$$\alpha_i = C \Leftrightarrow y_i u_i \leq 1$$

Hierbei ist u_i dieAusgabe der SVM für das i-te Trainings-Beispiel
Mit

. $$u = \sum_{j=1}^{N} y_j \alpha_j K(\vec{x}_j, \vec{x}) - b$$

Probleme bei Lösung mittels QP:

- Matrix mit n^2 Datenpunkten nötig

→Enormes Aufblähen der Matrix schon bei wenigen benutzten Trainingsbeispielen

→Sehr schnell ansteigender RAM-Bedarf

Lösungsansätze:

- **Chunking**
  Vapnik, 1982
- **Methode nach Osuna**
  Osuna et al., 1997
- **SMO**
  Platt et al, 1998

Chunking:

Vorüberlegung:
Nur Trainingsbeispiele mit $\alpha_i > 0$ interessieren uns
(da nur sie als Stützvektoren in Frage kommen)

Folgerung:
Trainingsbeispiele mit $\alpha_i = 0$ können aus der Matrix ausgelassen werden
→ Verkleinerung der Matrix

Anwendung:
Schrittweises Aufbauen der Matrix
Jeder Schritt beinhaltet alle Trainingsbeispiele mit $\alpha_i > 0$ aus dem vorherigen Trainingsschritt, sowie jene Trainingsbeispiele aus der Restmenge, welche die KKT-Bedingung am extremsten verletzen.

Chunking:

Was haben wir gewonnen?
Verkleinerung der Matrix auf l^2
(Mit l = Zahl der Trainingsbeispiele mit $\alpha_i > 0$)

Problem
Chunking verschiebt die das Problem nur,
Matrixgrösse steigt weiterhin stark an, so dass wir mit hinreichend grossen Trainingsmengen (bzw. einer grossen Zahl an Stützvektoren) wieder leicht die Grenze der RAM-Kapazität erreichen

Methode nach Osuna:

Osunas Theorem

•Grosse QP-Probleme können in eine Zahl kleinerer QP-Subprobleme gesplittet und schrittweise bearbeitet werden.

•Verletzt in jedem Trainings-Schritt mindestens eines der Trainings-Beispiele die KKT-Bedingung, so führt jeder dieser Schritte zu einer weiteren Optimierung und schlussendlich zu einem Konvergieren der Funktion.

Anwendung:

Es wird eine feste Grösse für die Traingsmatrix vorgegeben
In jedem Trainings-Schritt werden i Beispiele aus der Trainingsmatrix gelöscht, und dafür k Beispiele aus der Restmenge hinzugefügt, welche die KKT-Bedingung verletzen.
Osunas Ansatz sah noch k = 1 vor,
Was zu einer grossen Anzahl von Trainingschritten führte
Daher arbeiten Forscher heutzutage mit grösseren k

Allgemeines Problem mit Osuna und Chunking:

Beide Methoden erfordern einen numerischen QP-Solver

Dieser ist jedoch u.a. aufgrund vielfältiger Präzisionsprobleme schwierig zu handhaben

Vorteil von SMO:

Auf numerische QP-Optimierung wird verzichtet

Statt dessen wird das Gesamt-Problem in die kleinstmöglichen Sub-Probleme aufgeteilt (=Vergleich zweier Trainings-Beispiele) und diese dann analytisch gelöst.

Speicherplatz für eine Matrix wird nicht benötigt,
da in jedem Schritt immer nur 2 Trainingsbeispiele betrachtet werden

Programmschritte von SMO:

1. **Anwendung heuristischer Methoden, um die beiden Trainingsbeispiele zu ermitteln, die im nächsten Schritt analysiert werden**

2. **Lösung des Optimierunsproblems für die beiden gewählten Trainings-Beispiele um deren α_1 und α_2 zu bestimmen**

2. b) Update der SVM mit den neu ermittelten α-Werten

Auswählen neuer Trainingsdaten für den nächsten Trainings-Schritt:

Erinnerung:

Osunas Theorem

•Grosse QP-Probleme können in eine Zahl kleinerer QP-Subprobleme gesplittet und schrittweise bearbeitet werden.

•Verletzt in jedem Trainings-Schritt mindestens eines der Trainings-Beispiele die KKT-Bedingung, so führt jeder dieser Schritte zu einer weiteren Optimierung und schlussendlich zu einem Konvergieren der Funktion.

Folgerung

Gezielt nach Trainingsdaten suchen, welche die KKT-Bedingungen verletzen

Auswählen neuer Trainingsdaten für den nächsten Trainings-Schritt:

Definition: Ungebundene Untermenge (unbound subset)
Teilmenge der Trainingsdaten, bei denen α != 0 und α != C gilt
Diese Teilmenge hat die höchste Wahrscheinlichkeit, Trainingsdaten zu beinhalten, welche die KKT-Bedingung verletzen

Vorgehensweise von SMO
Unterschiedliche Vorgehensweisen für jeden Kandidaten
1. Wahl des Kandidaten für α_1 (=äussere Schleife)
2. Wahl des Kandidaten für α_2 (=innere Schleife)

Preprocessing
Für jedes Trainingsbeispiel wird der Fehlerwert E ermittelt und ständig aktuell gehalten

Wahl des Kandidaten für α_1:
Die Schleife zur Auswahl von α_1 sucht gezielt nach Daten, welche die KKT-Bedingung verletzen. Sie alterniert dabei zwischen Durchgängen in denen der gesamte Trainings-Datensatz durchsucht wird und Durchgängen, in denen nur die ungebundene Untermenge durchsucht wird

Wahl des Kandidaten für α_2:
Ziel:
Maximierung des im jeweiligen Trainings-Schritt erzielten Erfolges
Methodik:
Vergleich des Fehlerwertes des ersten Trainingsbeispiels (E_1) mit dem Fehlerwert von möglichen Kandidaten für das Zweite (E_2).
•Falls E_1 positiv, wähle Kandidaten mit negativem E.
•Falls E1 negativ, wähle Kandidaten mit positivem E

Sonderfälle:
Wahl des Kandidaten für α_2 resultierte nicht in Fortschritten bei der Optimierung

Behandlung von Sonderfällen:
Hierarchie an alternativen Auswahlmöglichkeiten für α_2 :
•Iteration durch die anderen ungebundenen Trainings-Beispiele
•Einbeziehung des ganzen Trainings-Datensatzes
•Falls immer noch kein α_2 gefunden wurde mit dem Fortschritte erzielt werden konnten:
•Wahl eines neuen Trainingsdatensatzes für α_1

SMO, Optimierungsfunktion:

1. Lösen des Optimierungsproblems

Eingabe in Prozedur: Trainingsbeispiele x_1 und x_2

Zu ermitteln: Optimale Lagrange-Multiplikatoren α_1 und α_2

Wichtige Nebenbedingungen für unsere α_i :

a) $0 < \alpha_i < C$

b)

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

Mit N = Anzahl der Trainingsbeispiele = 2

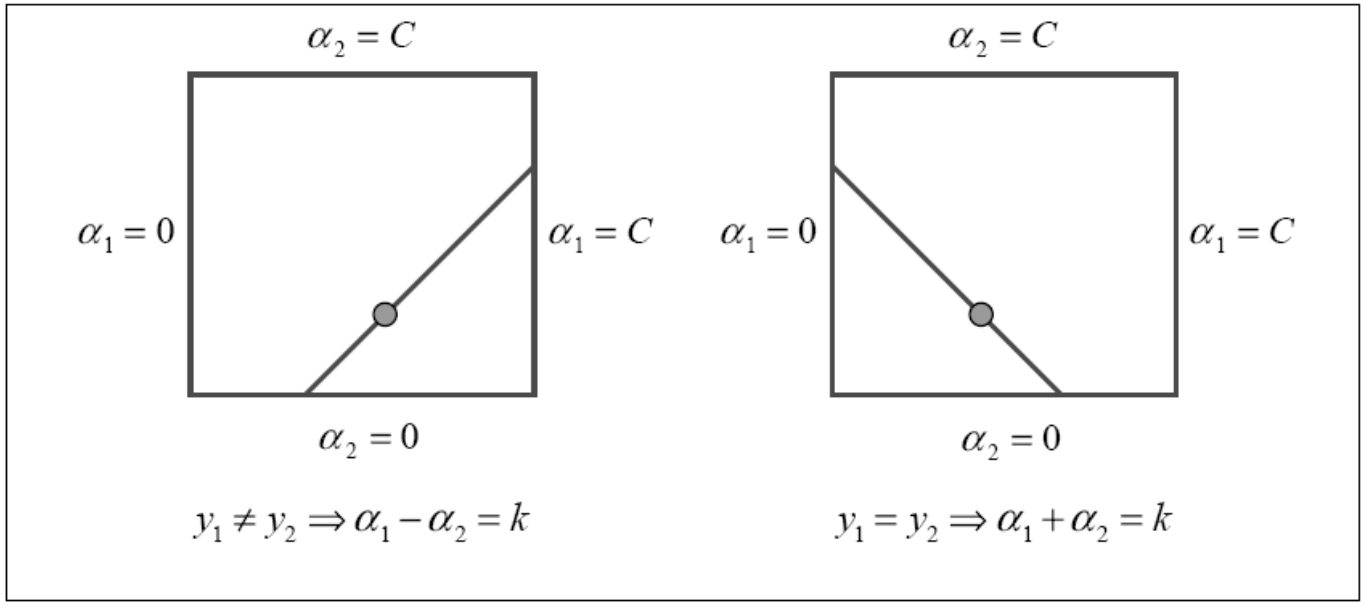
SMO, Optimierungsfunktion:

Mit diesen Nebenbedingungen lassen sich wichtige Aussagen über die gesuchten Kombination der gesuchten $\alpha_1$und α_2 herleiten

Aus $0 < \alpha_i < C$:
Die α-Werte liegen in einer Box, begrenzt von 0 und C

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

Die α-Werte liegen auf einer Linie innerhalb der Box



Um die Zweite Bedingung hinein zu bringen benötigen wir auch mindestens 2 Trainings-Beispiele

SMO, Optimierungsfunktion:

Bestimmung von α_2:

Obere/Untere Schranke für α_2:
Unterscheidung nach Vergleich von y_1 und y_2:

Falls $y_1 == y_2$ (beide Trainingsbeispiele in selber Klasse):
L = max (0, Diff) **H**= min (C, C + Diff)

Falls $y_1 != y_2$ (beide Trainingsbeispiele in unterschiedlicher Klasse):
L = max (0, C – Sum) **H** = min (C, Sum)

Mit Diff = $\alpha_2 - \alpha_1$ und Sum = $\alpha_2 + \alpha_1$

Bestimmung von Fehlerwert E_1 und E_2:
$E_i = u_i - y_i$

Aus der 2. Ableitung der Zielfunktion lässt sich η berechnen:

$$\eta = K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) + K(\vec{x}_1, \vec{x}_2)$$

Daraus lässt sich nun ein neues α_2 berechnen:

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta}$$

Aufgrund der erwähnten Constraints für L und H müssen wir den so berechneten Wert ggf. noch clippen um unser neues α_2 zu erhalten:

$$\alpha_2^{new,clipped} = \begin{cases} H & falls \quad \alpha_2^{new} \geq H \\ \alpha_2^{new} & falls \quad L < \alpha_2^{new} < H \\ L & falls \quad \alpha_2^{new} \leq L \end{cases}$$

Sei

$s = y_1 y_2$

dann bekommen wir so direkt auch ein neues α_1:

$$\alpha_1^{new} = \alpha_1 + s(\alpha_2 - \alpha_2^{new,clipped})$$

Sollten wir einen negativen Wert für η bekommen haben, müssen wir allerdings ggf. noch die Zielfunktion Ψ neu auswerten:

$$
\begin{aligned}
f_1 &= y_1(E_1 + b) - \alpha_1 K(\vec{x}_1, \vec{x}_1) - s\alpha_2 K(\vec{x}_1, \vec{x}_2), \\
f_2 &= y_2(E_2 + b) - s\alpha_1 K(\vec{x}_1, \vec{x}_2) - \alpha_2 K(\vec{x}_2, \vec{x}_2), \\
L_1 &= \alpha_1 + s(\alpha_2 - L), \\
H_1 &= \alpha_1 + s(\alpha_2 - H), \\
\Psi_L &= L_1 f_1 + L f_2 + \tfrac{1}{2} L_1^2 K(\vec{x}_1, \vec{x}_1) + \tfrac{1}{2} L^2 K(\vec{x}_2, \vec{x}_2) + sLL_1 K(\vec{x}_1, \vec{x}_2), \\
\Psi_H &= H_1 f_1 + H f_2 + \tfrac{1}{2} H_1^2 K(\vec{x}_1, \vec{x}_1) + \tfrac{1}{2} H^2 K(\vec{x}_2, \vec{x}_2) + sHH_1 K(\vec{x}_1, \vec{x}_2).
\end{aligned}
$$

Berechnung des Bias b:
Erfolgt nach jedem Trainings-Schritt, falls α_1 und α_2 den KKT-Bedingungen gehorchen. Wir berechnen ein Bias für jedes der beiden Trainingsbeispiele, und leiten daraus unser Gesamtbias her.
Berechnung:

$$b_1 = E_1 + y_1(\alpha_1^{new} - \alpha_1)K(\vec{x}_1, \vec{x}_1) + y_2(\alpha_2^{new,clipped} - \alpha_2)K(\vec{x}_1, \vec{x}_2) + b$$

$$b_2 = E_2 + y_1(\alpha_1^{new} - \alpha_1)K(\vec{x}_1, \vec{x}_2) + y_2(\alpha_2^{new,clipped} - \alpha_2)K(\vec{x}_2, \vec{x}_2) + b$$

Für beide b_i gilt:
Sie sind gültig. Falls das dazugehörige α_i nicht auf den Grenzen (also 0 oder C) liegt

Für das Gesamtbias b gilt:
Falls $b_1 = b_2 \rightarrow b = b_1$
Falls $b_1 != b_2 \rightarrow b = (b_1 + b_2)/2$

Berechnung von w:

$$\vec{w}^{new} = \vec{w} + y_1(\alpha_1^{new} - \alpha_1)\vec{x}_1 + y_2(\alpha_2^{new,clipped} - \alpha_2)\vec{x}_2$$

Dieser Wert muss nur für die gesamte SVM abgespeichert/aktualisiert werden

Vergleich von SMO und Chunking

Exemplarisch für Kategorisierung von Webseiten:
Gegeben: Datensatz mit 49749 Webseiten
Gesucht: Klassifikation

Ergebnisse:

Training Set Size	SMO time	Chunking time	Number of Non-Bound Support Vectors	Number of Bound Support Vectors
2477	2.2	13.1	123	47
3470	4.9	16.1	147	72
4912	8.1	40.6	169	107
7366	12.7	140.7	194	166
9888	24.7	239.3	214	245
17188	65.4	1633.3	252	480
24692	104.9	3369.7	273	698
49749	268.3	17164.7	315	1408

Zusammenfassung:

Vorteile von SMO:

- Kommt ohne numerischen QP-Solver aus → keine Präzisionsprobleme
- Kein Speicherplatz für eine Matrix nötig → kein zur Zahl der Trainingsbeispiele exponentiell ansteigender Speicherplatzbedarf
- Erheblich bessere Rechenzeit, zumindest im Vergleich mit Chunking (allerdings keine Vergleichsdaten zu Osuna vorhanden)
- Einfach zu implementieren (Pseudocode-Version im Paper kommt mit 90 LOC aus)

Vielen Dank für Ihre Aufmerksamkeit