Vorlesung Maschinelles Lernen

Strukturelle Modelle – SVMstruct

Katharina Morik

LS 8 Informatik
Technische Universität Dortmund

10.12.2013

Gliederung

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Jenseits des Bag of Words

- Bisher haben wir Texte als Anzahl und Häufigkeit von Wörtern repräsentiert.
- Damit haben wir die Struktur der Sprache ignoriert.
  - Grammatik
  - Koreferenz
  - Eigennamen
  - Semantische Relationen
- Es gibt eine ganze Reihe von Ansätzen des maschinellen Lernens, um (sprachliche) Strukturen zu behandeln.
- Wir besprechen hier nur die SVM bezogenen Ansätze.

Lernaufgabe Named Entity Recognition

- Wortfolgen, die sich auf ein individuelles Objekt beziehen, werden Named Entities (NE) genannt.
- Eigennamen, Ortsnamen, Firmennamen sind z.B. NEs.
- Gegeben Beispiele von Sätzen, in denen NEs annotiert sind, lerne die Entscheidungsfunktion, die für jedes Wort angibt, ob es zu einer bestimmten NE gehört, oder nicht.
- Beispiel:

Johann	Sebastian	Bach	publiziert	im	Henle	Verlag	München.
Per	Per	Per	0	0	Org	Org	Place

Lernaufgabe Named Entity Recognition

- Wortfolgen, die sich auf ein individuelles Objekt beziehen, werden Named Entities (NE) genannt.
- Eigennamen, Ortsnamen, Firmennamen sind z.B. NEs.
- Gegeben Beispiele von Sätzen, in denen NEs annotiert sind, lerne die Entscheidungsfunktion, die für jedes Wort angibt, ob es zu einer bestimmten NE gehört, oder nicht.
- Beispiel:

Johann	Sebastian	Bach	publiziert	im	Henle	Verlag	München.
Per	Per	Per	0	0	Org	Org	Place

Lernaufgabe Named Entity Recognition

- Wortfolgen, die sich auf ein individuelles Objekt beziehen, werden Named Entities (NE) genannt.
- Eigennamen, Ortsnamen, Firmennamen sind z.B. NEs.
- Gegeben Beispiele von Sätzen, in denen NEs annotiert sind, lerne die Entscheidungsfunktion, die für jedes Wort angibt, ob es zu einer bestimmten NE gehört, oder nicht.
- Beispiel:

Johann	Sebastian	Bach	publiziert	im	Henle	Verlag	München.
Per	Per	Per	0	0	Org	Org	Place

Lernaufgabe Named Entity Recognition

- Wortfolgen, die sich auf ein individuelles Objekt beziehen, werden Named Entities (NE) genannt.
- Eigennamen, Ortsnamen, Firmennamen sind z.B. NEs.
- Gegeben Beispiele von Sätzen, in denen NEs annotiert sind, lerne die Entscheidungsfunktion, die für jedes Wort angibt, ob es zu einer bestimmten NE gehört, oder nicht.
- Beispiel:

Johann	Sebastian	Bach	publiziert	im	Henle	Verlag	München.
Per	Per	Per	0	0	Org	Org	Place

Anwendungen

Wenn wir in Dokumenten die NEs automatisch annotieren,

- können wir sie im Text markieren, so dass die Benutzer schneller interessante Stellen auffinden;
- können wir alle Sätze zu einer Person, Firma, einem Ort herausschreiben und so eine Zusammenfassung für einen Text erstellen;
- eine weitere Lernaufgabe aufsetzen: Relationen zwischen NEs lernen, z.B. Fusion von Firmen $\mathit{fusion}(\mathit{Org}_1, \mathit{Org}_2)$, Mitglied im Aufsichtsrat $\mathit{aufsicht}(\mathit{Org}, \mathit{Per})$.

Letztlich erstellen wir eine Datenbank aus einer Dokumentensammlung. Auf diese Datenbank wenden wir dann unsere Lernverfahren wie gehabt an.

Anwendungen

Wenn wir in Dokumenten die NEs automatisch annotieren,

- können wir sie im Text markieren, so dass die Benutzer schneller interessante Stellen auffinden;
- können wir alle Sätze zu einer Person, Firma, einem Ort herausschreiben und so eine Zusammenfassung für einen Text erstellen;
- eine weitere Lernaufgabe aufsetzen: Relationen zwischen NEs lernen, z.B. Fusion von Firmen $\mathit{fusion}(\mathit{Org}_1, \mathit{Org}_2)$, Mitglied im Aufsichtsrat $\mathit{aufsicht}(\mathit{Org}, \mathit{Per})$.

Letztlich erstellen wir eine Datenbank aus einer Dokumentensammlung. Auf diese Datenbank wenden wir dann unsere Lernverfahren wie gehabt an.

Anwendungen

Wenn wir in Dokumenten die NEs automatisch annotieren,

- können wir sie im Text markieren, so dass die Benutzer schneller interessante Stellen auffinden;
- können wir alle Sätze zu einer Person, Firma, einem Ort herausschreiben und so eine Zusammenfassung für einen Text erstellen;
- eine weitere Lernaufgabe aufsetzen: Relationen zwischen NEs lernen, z.B. Fusion von Firmen $\mathit{fusion}(\mathit{Org}_1, \mathit{Org}_2)$, Mitglied im Aufsichtsrat $\mathit{aufsicht}(\mathit{Org}, \mathit{Per})$.

Letztlich erstellen wir eine Datenbank aus einer Dokumentensammlung. Auf diese Datenbank wenden wir dann unsere Lernverfahren wie gehabt an.

Part of Speech Tagging

- Unter Part-of-speech Tagging versteht man die Zuordnung von Wörtern eines Textes zu Wortarten (engl.: part of speech).
- Beispiel:

Die	Noten	erscheinen	bei	Henle.
Det	N	V	Prep	N

Part of Speech Tagging

- Unter Part-of-speech Tagging versteht man die Zuordnung von Wörtern eines Textes zu Wortarten (engl.: part of speech).
- Beispiel:

Die	Noten	erscheinen	bei	Henle.
Det	N	V	Prep	N

Full Parsing

- Syntaxregeln produzieren einen Syntaxbaum für einen Satz, dessen Wurzel das Startsymbol S ist und die Blätter sind die Wortarten (präterminalen Knoten), denen dann die Wörter zugeordnet sind. Full Parsing erstellt Syntaxbäume für Sätze.
- Beispiel:

  x: Die Noten erscheinen bei Henle.

  y:

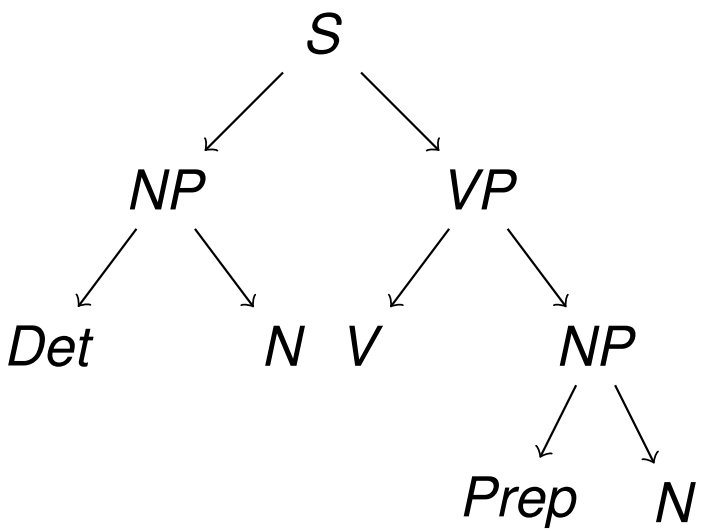
$S \rightarrow NP, VP$
$VP \rightarrow V, NP$
$NP \rightarrow Det, N$
$NP \rightarrow Prep, N$

Full Parsing

- Syntaxregeln produzieren einen Syntaxbaum für einen Satz, dessen Wurzel das Startsymbol S ist und die Blätter sind die Wortarten (präterminalen Knoten), denen dann die Wörter zugeordnet sind. Full Parsing erstellt Syntaxbäume für Sätze.
- Beispiel:

  x: Die Noten erscheinen bei Henle.

y:



$S \rightarrow NP, VP$
$VP \rightarrow V, NP$
$NP \rightarrow Det, N$
$NP \rightarrow Prep, N$

Lernaufgaben Part of Speech Tagging, Full Parsing

- Part of Speech Tagging: Gegeben eine Menge von Sätzen, bei denen zu jedem Wort die Wortart angegeben ist, lerne eine Entscheidungsfunktion, die bei beliebigen Sätzen jedem Wort eine Wortart zuordnet.
- Full Parsing Learning: Gegeben eine Menge von Sätzen, eine Menge von Syntaxregeln und die Anzahl von Regelanwendungen für jeden Satz, lerne eine Entscheidungsfunktion, die beliebigen Sätzen die Anzahl von Regelanwendungen zuordnet (discriminant model).
- Zur Abgrenzung: Es sind
  Gegeben eine Menge von syntaktisch korrekten Sätzen (positive Beispiele) und eine Menge von synaktisch falschen Sätzen (negative Sätze), bei denen jeweils die Wortarten annotiert sind, lerne Syntaxregeln, die gerade die syntaktisch korrekten Sätze produzieren (generative model).

Lernaufgaben Part of Speech Tagging, Full Parsing

- Part of Speech Tagging: Gegeben eine Menge von Sätzen, bei denen zu jedem Wort die Wortart angegeben ist, lerne eine Entscheidungsfunktion, die bei beliebigen Sätzen jedem Wort eine Wortart zuordnet.
- Full Parsing Learning: Gegeben eine Menge von Sätzen, eine Menge von Syntaxregeln und die Anzahl von Regelanwendungen für jeden Satz, lerne eine Entscheidungsfunktion, die beliebigen Sätzen die Anzahl von Regelanwendungen zuordnet (discriminant model).
- Zur Abgrenzung: Es sind
  Gegeben eine Menge von syntaktisch korrekten Sätzen (positive Beispiele) und eine Menge von synaktisch falschen Sätzen (negative Sätze), bei denen jeweils die Wortarten annotiert sind, lerne Syntaxregeln, die gerade die syntaktisch korrekten Sätze produzieren (generative model).

Lernaufgaben Part of Speech Tagging, Full Parsing

- Part of Speech Tagging: Gegeben eine Menge von Sätzen, bei denen zu jedem Wort die Wortart angegeben ist, lerne eine Entscheidungsfunktion, die bei beliebigen Sätzen jedem Wort eine Wortart zuordnet.
- Full Parsing Learning: Gegeben eine Menge von Sätzen, eine Menge von Syntaxregeln und die Anzahl von Regelanwendungen für jeden Satz, lerne eine Entscheidungsfunktion, die beliebigen Sätzen die Anzahl von Regelanwendungen zuordnet (discriminant model).
- Zur Abgrenzung: Es sind
  Gegeben eine Menge von syntaktisch korrekten Sätzen (positive Beispiele) und eine Menge von synaktisch falschen Sätzen (negative Sätze), bei denen jeweils die Wortarten annotiert sind, lerne Syntaxregeln, die gerade die syntaktisch korrekten Sätze produzieren (generative model).

Support Vector Machines für alles...

Bisher haben wir zwei Lernaufgaben für *Support Vector Machines* betrachtet:

- Binäre Klassifikation
- SVM zur Regression

Aktuelles Übungsblatt: *Mehrklassen-Problem*

Jetzt: Lernen von Funktionen für beliebige strukturelle Ausgaben (Bäume, Graphen) mit Hilfe der *SVMstruct*.

Support Vector Machines für alles...

Bisher haben wir zwei Lernaufgaben für *Support Vector Machines* betrachtet:

- Binäre Klassifikation
- SVM zur Regression

Aktuelles Übungsblatt: *Mehrklassen-Problem*

Jetzt: Lernen von Funktionen für beliebige strukturelle Ausgaben (Bäume, Graphen) mit Hilfe der *SVMstruct*.

SVMstruct

Strukturelle Modelle

Sei X die Menge der Beispiele. Ist die Ausgabe-Variable Y nicht ein Skalar, sondern eine Struktur (z.B. eine Folge, ein Baum, ein Graph), so heißt das Modell

$$f : X \to Y$$

strukturelles Modell.

- *Large Margin Methods for Structured and Interdependent Output Variables*, I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, J. of Machine Learning Research, Vol. 6, p. 1453 – 1484, 2005
- *Training Linear SVMs in Linear Time*, Thorsten Joachims, Proc. KDD 2006

SVMstruct

Strukturelle Modelle

Sei X die Menge der Beispiele. Ist die Ausgabe-Variable Y nicht ein Skalar, sondern eine Struktur (z.B. eine Folge, ein Baum, ein Graph), so heißt das Modell

$$f : X \to Y$$

strukturelles Modell.

- *Large Margin Methods for Structured and Interdependent Output Variables*, I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, J. of Machine Learning Research, Vol. 6, p. 1453 – 1484, 2005
- *Training Linear SVMs in Linear Time*, Thorsten Joachims, Proc. KDD 2006

SVMstruct

Strukturelle Modelle

Sei X die Menge der Beispiele. Ist die Ausgabe-Variable Y nicht ein Skalar, sondern eine Struktur (z.B. eine Folge, ein Baum, ein Graph), so heißt das Modell

$$f : X \to Y$$

strukturelles Modell.

- *Large Margin Methods for Structured and Interdependent Output Variables*, I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, J. of Machine Learning Research, Vol. 6, p. 1453 – 1484, 2005
- *Training Linear SVMs in Linear Time*, Thorsten Joachims, Proc. KDD 2006

Lernaufgabe der SVMstruct

Lernaufgabe SVMstruct

Sei $Y = Y_1 \times \ldots \times Y_q$ eine Menge und $\vec{y} \in Y$ eine *Konfiguration* in Y und

$$\mathcal{T} = \left\{(\vec{x}_1, \vec{y}_1), \ldots, (\vec{x}_n, \vec{y}_n)\right\} \subset X \times Y$$

eine Menge von Beobachtungen.

Ziel: Finde eine Funktion f mit

$$f : X \to Y$$

SVMstruct

Es sei eine $\vec{\beta}$-parametrisierte Schar von Funktionen

$$F_{\vec{\beta}} : X \times Y \to \mathbb{R}$$

gegeben, die die Ähnlichkeit zwischen $\vec{x}$ und $\vec{y}$ ausdrücken.

Gesucht ist nun ein $\vec{\beta}^*$ derart, daß

$$f(x, \vec{\beta}^*) = \arg\max_{\vec{y} \in Y} F_{\vec{\beta}^*}(\vec{x}, \vec{y})$$

jeweils zu einem $\vec{x}$, das am besten passende $\vec{y} \in Y$ liefert.

Hinweis: In der Literatur wird für $F_{\vec{\beta}}(\vec{x}, \vec{y})$ meist $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ geschrieben.

Wir brauchen also eine Art Kostenfunktion, die uns die Ähnlichkeit zwischen $\vec{x}$ und $\vec{y}$ liefert. Sei dazu Ψ eine Abbildung

$$\Psi : X \times Y \to \mathcal{F}$$

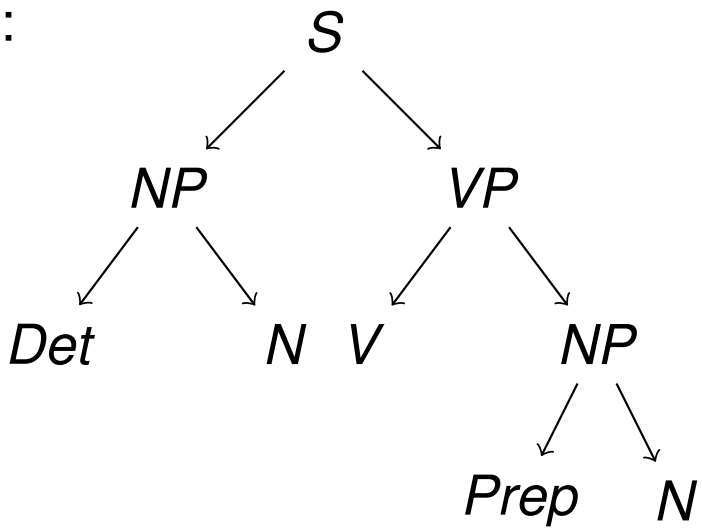
von $(\vec{x}, \vec{y})$ auf eine gemeinsame Repräsentation $\Psi(\vec{x}, \vec{y})$.

Die Abbildung Ψ hängt dabei vom konkreten Problem ab, eine Darstellung für Parse-Trees liefert z.B.

x: Die Noten erscheinen bei Henle.

$f : X \to Y$

y:



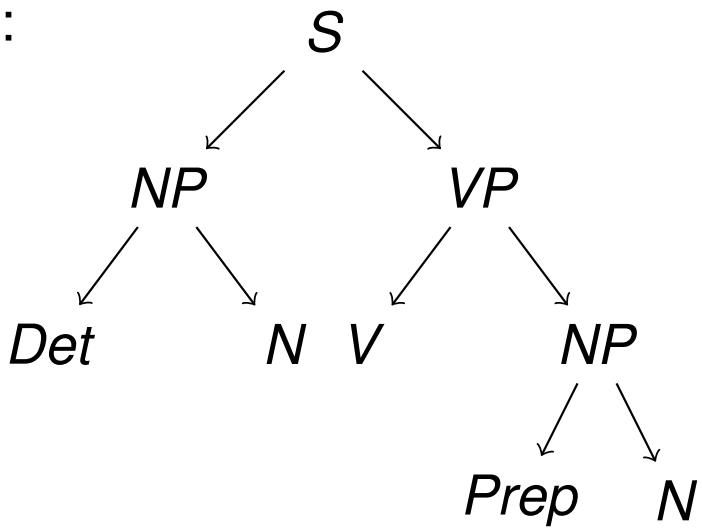
$$\Psi(\vec{x}, \vec{y}) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad \begin{matrix} S \to NP, VP \\ NP \to Det, N \\ NP \to Prep, N \\ NP \to Adj, N \\ \vdots \\ Prep \to in \\ Prep \to bei \end{matrix}$$

Merkmalsabbildung am Beispiel der ParseTrees

x: Die Noten erscheinen bei Henle.

$f : X \rightarrow Y$

y:



$$\Psi(\vec{x}, \vec{y}) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \begin{matrix} S \rightarrow NP, VP \\ NP \rightarrow Det, N \\ NP \rightarrow Prep, N \\ NP \rightarrow Adj, N \\ \vdots \\ Prep \rightarrow in \\ Prep \rightarrow bei \end{matrix}$$

Wir nehmen im Folgenden an, daß F linear in $\Psi(\vec{x}, \vec{y})$ ist, d.h.

$$F_{\vec{\beta}}(\vec{x}, \vec{y}) = \left\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \right\rangle$$

Wir lernen also über input-/output-Kombinationen die Ranking-Funktion

$$F : X \times Y \rightarrow \mathbb{R}.$$

Was hat das mit SVM zu tun?

Wo ist jetzt die SVM-Idee hier?

Mit der Funktion $F_{\vec{\beta}^*}$ haben wir eine Ordnung auf den Paaren $(\vec{x}, \vec{y})$, so daß wir jedem neuen $\vec{x}'$ ein $\vec{y}'$ zuordnen können:

$$f(\vec{x}', \vec{\beta}^*) = \vec{y}' = \arg\max_{\vec{y} \in Y} F_{\vec{\beta}^*}(\vec{x}', \vec{y})$$

Wir wählen also immer das am besten bzgl. $F_{\vec{\beta}^*}$ passende $\vec{y}$!

Lernen von $\vec{\beta}^*$ über Optimierung mit der SVM:
Maximiere den Abstand (Margin) zwischen der besten und der zweit-besten Lösung!

Primales Problem

Wir suchen also eine Hyperebene, die das Beste von allen anderen Beispielen trennt.

Dazu soll $f(\vec{x}_i, \vec{\beta})$ für $\vec{x}_i \in \mathcal{T}$ jeweils das "richtige $\vec{y}_i$" vorhersagen, d.h.

$$\forall i \; : \; \max_{\vec{y} \in Y \setminus \vec{y}_i} \left\{ \left\langle \vec{\beta}, \Psi(\vec{x}_i, \vec{y}) \right\rangle \right\} < \left\langle \vec{\beta}, \Psi(\vec{x}_i, \vec{y}_i) \right\rangle$$

$$\Rightarrow \quad \forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \quad : \quad \left\langle \vec{\beta}, \Psi(\vec{x}_i, \vec{y}) \right\rangle < \left\langle \vec{\beta}, \Psi(\vec{x}_i, \vec{y}_i) \right\rangle$$

$$\Rightarrow \quad \forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \quad : \quad \left\langle \vec{\beta}, \Psi(\vec{x}_i, \vec{y}_i) - \Psi(\vec{x}_i, \vec{y}) \right\rangle > 0$$

Primales Problem

Sind die Nebenbedingungen

$$\forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \quad : \quad \left\langle \vec{\beta}\,,\, \Psi(\vec{x}_i, \vec{y}_i) - \Psi(\vec{x}_i, \vec{y}) \right\rangle > 0$$

erfüllbar, führt dies typischerweise zu einer Menge optimaler Lösungen.

Eine eindeutige Lösung läßt sich jetzt finden, indem man das *Maximum Margin*-Prinzip der SVM nutzt und $\vec{\beta}$ mit $||\vec{\beta}|| \leq 1$ so wählt, dass der Abstand zum nächstbesten $\vec{\beta}'$ maximal wird, also

$$\hat{\vec{y}}_i = \arg\max_{\vec{y} \neq \vec{y}_i} \left\langle \vec{\beta}\,,\, \Psi(\vec{x}_i, \vec{y}) \right\rangle$$

Primales Problem

Schließlich führt das zum primalen Problem der *SVMstruct*:

Primales Problem

Minimiere

$$L_P(\vec{\beta}) = \frac{1}{2}||\vec{\beta}|| \tag{1}$$

unter den Nebenbedingungen

$$\forall i, \forall \vec{y} \in Y \setminus \vec{y}_i \quad : \quad \left\langle \vec{\beta}, \delta\Psi_i(\vec{y}_i) \right\rangle \geq 1 \tag{2}$$

mit $\delta\Psi_i(\vec{y}) = \Psi(\vec{x}_i, \vec{y}_i) - \Psi(\vec{x}_i, \vec{y})$

Die SVMstruct optimiert also unter Beachtung von $n|Y| - n$ Nebenbedingungen.

SVMstruct mit Ausnahmen – slack rescaling

Ein Strafterm C für alle Beispiele $\vec{x}$, bei denen die Nebenbedingungen verletzt sind, und die Relaxierung durch ξ führt zu dem Minimierungsproblem:

Slack Rescaling SVM

$$SVM_1 : \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \tag{3}$$

unter den Bedingungen

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y}_i : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \xi_i, \xi_i \geq 0 \tag{4}$$

C ist linear in den ξ_i.

SVMstruct mit Ausnahmen – margin rescaling

Verletzungen der Nebenbedingungen können auch durch einen quadratischen Term bestraft werden.

Margin rescaling SVM

$$SVM_2: \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{2N}\sum_{i=1}^{N}\xi_i^2 \tag{5}$$

unter den Bedingungen

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y}_i : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \xi_i, \xi_i \geq 0 \tag{6}$$

Empirisches Risiko und Slack Rescaling

Auch bei strukturellen Modellen geht es darum, den Fehler zu minimieren. Der erwartete Fehler bei irgend einer Verlustfunktion Δ ist für eine Menge von Beispielen $\mathcal{T}$

$$R_{\mathcal{T}}(f) = \frac{1}{2} \sum_{i=1}^{N} \Delta(\vec{y}_i, f(\vec{x}_i)) \tag{7}$$

Um die Verletzung der Nebenbedingung für ein $\vec{y} \neq \vec{y}_i$ bei großem Verlust $\Delta(\vec{y}_i, \vec{y})$ stärker zu bestrafen als bei geringerem, passen wir die Nebenbedingungen an:

$$SVM_1: \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i$$

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y}_i : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \frac{\xi_i}{\Delta(\vec{y}_i, \vec{y})} \tag{8}$$

$$SVM_2: \quad \min_{\vec{\beta},\xi} \ \frac{1}{2}\|\vec{\beta}\|^2 + \frac{C}{2N}\sum_{i=1}^{N}\xi_i^2$$

$$\forall i, \forall \vec{y} \in Y \backslash \vec{y}_i : \langle \vec{\beta}, \delta\Psi_i(\vec{y}) \rangle \geq 1 - \frac{\xi_i}{\sqrt{\Delta(\vec{y}_i, \vec{y})}} \tag{9}$$

Obere Schranke des empirischen Fehlers

Satz

Seien $\xi_i^*(\vec{\beta})$ die optimalen Schlupfvariablen für ein gegebenes $\vec{\beta}$, dann ist

$$R_T(\vec{\beta}) \leq \frac{1}{N} \sum_{i=1}^{N} \xi_i^*$$

Obere Schranke des empirischen Fehlers

Beweis:
Wir wissen:

$$\xi_i^* = \max\left\{0, \max_{\vec{y}\neq\vec{y}_i}\left\{\Delta(\vec{y}_i,\vec{y})\left[1-\langle\vec{\beta},\delta\Psi_i(\vec{y})\rangle\right]\right\}\right\}$$

Sei $\vec{y}^* = f(\vec{x}_i,\vec{\beta})$, es ergeben sich zwei Fälle

$$\begin{array}{lll} \text{1. Fall: } \vec{y}^* = \vec{y} & : & \text{Es ist } \Delta(\vec{y}_i, f(\vec{x}_i,\vec{\beta})) = 0 \leq \xi_i^* \\ \text{2. Fall: } \vec{y}^* \neq \vec{y} & : & \Rightarrow \left\langle \vec{y}_i\,,\, \delta\Psi_i(\vec{y}^*)\right\rangle \leq 0 \\ & & \Rightarrow \frac{\xi_i^*}{\Delta(\vec{y}_i,\vec{y})} \geq 1 \Leftrightarrow \Delta(\vec{y}_i,\vec{y}) \leq \xi_i^* \end{array}$$

Da die Schranke für jedes Beispiel gilt, gilt sie auch für den Durchschnitt. □

Hinführung zum dualen Problem

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

- Jetzt sind $\vec{y}_i, \vec{y}_j$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der N Beispiele $\vec{x}_i$ mit jedem der $\mid Y \mid$ möglichen $\vec{y}_i$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x}_i, \vec{y})$ ist.
- Wir haben also nicht mehr ein α je Beispiel in X, sondern ein α für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die α.

Hinführung zum dualen Problem

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

- Jetzt sind $\vec{y}_i, \vec{y}_j$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der N Beispiele $\vec{x}_i$ mit jedem der $\mid Y \mid$ möglichen $\vec{y}_i$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x}_i, \vec{y})$ ist.
- Wir haben also nicht mehr ein α je Beispiel in X, sondern ein α für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die α.

Hinführung zum dualen Problem

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

- Jetzt sind $\vec{y}_i, \vec{y}_j$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der N Beispiele $\vec{x}_i$ mit jedem der $\mid Y \mid$ möglichen $\vec{y}_i$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x}_i, \vec{y})$ ist.
- Wir haben also nicht mehr ein α je Beispiel in X, sondern ein α für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die α.

Hinführung zum dualen Problem

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

- Jetzt sind $\vec{y}_i, \vec{y}_j$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der N Beispiele $\vec{x}_i$ mit jedem der $\mid Y \mid$ möglichen $\vec{y}_i$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x}_i, \vec{y})$ ist.
- Wir haben also nicht mehr ein α je Beispiel in X, sondern ein α für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die α.

Hinführung zum dualen Problem

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

- Jetzt sind $\vec{y}_i, \vec{y}_j$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der N Beispiele $\vec{x}_i$ mit jedem der $\mid Y \mid$ möglichen $\vec{y}_i$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x}_i, \vec{y})$ ist.
- Wir haben also nicht mehr ein α je Beispiel in X, sondern ein α für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die α.

Hinführung zum dualen Problem

- Wir wollen auch hier das duale Problem formulieren.
- Bisher war es:

$$L_D(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \left\langle \vec{x}_i \, , \, \vec{x}_j \right\rangle$$

- Jetzt sind $\vec{y}_i, \vec{y}_j$ nicht mehr einfach Werte, sondern Strukturen.
- Für jedes der N Beispiele $\vec{x}_i$ mit jedem der $\mid Y \mid$ möglichen $\vec{y}_i$ müssen wir feststellen, wie groß der Abstand zu allen anderen $\Psi(\vec{x}_i, \vec{y})$ ist.
- Wir haben also nicht mehr ein α je Beispiel in X, sondern ein α für jedes Paar in $X \times Y$.
- Erst sehen wir uns den Raum an, in dem optimiert wird, dann die α.

Raum, in dem SVMstruct optimiert

- Bei der klassischen SVM haben wir beim dualen Problem für $i, j = 1, ..., N$ das Skalarprodukt $\langle \vec{x_i}, \vec{x_j} \rangle$ gerechnet.
- Jetzt müssen wir für $i, j = 1, ..., N$ rechnen $J_{(i\vec{y})(j\vec{y'})} \equiv \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.

	$\vec{x_1}$	...	$\vec{x_j}$	...	$\vec{x_N}$
$\vec{x_1}$	—	...	...	...	...
...	...	...	...	...	...
$\vec{x_i}$	...	...	$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$	...	...
...	...	...	...	...	...
$\vec{x_N}$	...	...	...	...	—

- Dabei ist $\langle \delta\Psi_j(\vec{y}), \delta\Psi_i(\vec{y'}) \rangle$ wieder eine Matrix!

Raum, in dem SVMstruct optimiert

- Bei der klassischen SVM haben wir beim dualen Problem für $i, j = 1, ..., N$ das Skalarprodukt $\langle \vec{x_i}, \vec{x_j} \rangle$ gerechnet.
- Jetzt müssen wir für $i, j = 1, ..., N$ rechnen $J_{(i\vec{y})(j\vec{y'})} \equiv \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.

	$\vec{x_1}$	...	$\vec{x_j}$	...	$\vec{x_N}$
$\vec{x_1}$	—	...	...	...	...
...	...	...	...	...	...
$\vec{x_i}$	...	...	$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$	...	...
...	...	...	...	...	...
$\vec{x_N}$	...	...	...	...	—

- Dabei ist $\langle \delta\Psi_j(\vec{y}), \delta\Psi_i(\vec{y'}) \rangle$ wieder eine Matrix!

Raum, in dem SVMstruct optimiert

- Bei der klassischen SVM haben wir beim dualen Problem für $i, j = 1, ..., N$ das Skalarprodukt $\langle \vec{x_i}, \vec{x_j} \rangle$ gerechnet.
- Jetzt müssen wir für $i, j = 1, ..., N$ rechnen $J_{(i\vec{y})(j\vec{y'})} \equiv \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.

	$\vec{x_1}$	...	$\vec{x_j}$	...	$\vec{x_N}$
$\vec{x_1}$	—	...	...	...	...
...	...	...	...	...	...
$\vec{x_i}$	...	...	$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$	...	...
...	...	...	...	...	...
$\vec{x_N}$	...	...	...	...	—

- Dabei ist $\langle \delta\Psi_j(\vec{y}), \delta\Psi_i(\vec{y'}) \rangle$ wieder eine Matrix!

Matrix **J**

- In der $N \times N$ Matrix sind die Einträge $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.
- Wir erinnern uns: $\delta\Psi_i(\vec{y}) \equiv \Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})$
- Statt einer einfachen Matrix haben wir einen Tensor, d.h. der Eintrag in die $N \times N$ Matrix ist eine $|\ Y\ | \times |\ Y\ |$ Matrix **J**:

	$\vec{y_1}$	...	$\vec{y_{\|Y\|}}$
$\vec{y_1}$	$-$	...	$\Psi(\vec{x}, \vec{y_1}) - \Psi(\vec{x}, \vec{y_{\|Y\|}})$
...	...	...	...
$\vec{y_{\|Y\|}}$	$\Psi(\vec{x}, \vec{y_{\|Y\|}}) - \Psi(\vec{x}, \vec{y_1})$	...	$-$

- **J** ist eine Kernfunktion über $X \times Y$: $J_{(i\vec{y})(j\vec{y'})} = \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$

Matrix **J**

- In der $N \times N$ Matrix sind die Einträge $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.
- Wir erinnern uns: $\delta\Psi_i(\vec{y}) \equiv \Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})$
- Statt einer einfachen Matrix haben wir einen Tensor, d.h. der Eintrag in die $N \times N$ Matrix ist eine $|\ Y\ | \times |\ Y\ |$ Matrix **J**:

	$\vec{y_1}$	...	$\vec{y_{\lvert Y \rvert}}$
$\vec{y_1}$	$-$	...	$\Psi(\vec{x}, \vec{y_1}) - \Psi(\vec{x}, \vec{y_{\lvert Y \rvert}})$
...	...	...	...
$\vec{y_{\lvert Y \rvert}}$	$\Psi(\vec{x}, \vec{y_{\lvert Y \rvert}}) - \Psi(\vec{x}, \vec{y_1})$	...	$-$

- **J** ist eine Kernfunktion über $X \times Y$: $J_{(i\vec{y})(j\vec{y'})} = \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$

Matrix **J**

- In der $N \times N$ Matrix sind die Einträge $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.
- Wir erinnern uns: $\delta\Psi_i(\vec{y}) \equiv \Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})$
- Statt einer einfachen Matrix haben wir einen Tensor, d.h. der Eintrag in die $N \times N$ Matrix ist eine $| Y | \times | Y |$ Matrix **J**:

	$\vec{y_1}$	...	$\vec{y_{\|Y\|}}$
$\vec{y_1}$	$-$	...	$\Psi(\vec{x}, \vec{y_1}) - \Psi(\vec{x}, \vec{y_{\|Y\|}})$
...	...	...	...
$\vec{y_{\|Y\|}}$	$\Psi(\vec{x}, \vec{y_{\|Y\|}}) - \Psi(\vec{x}, \vec{y_1})$	...	$-$

- **J** ist eine Kernfunktion über $X \times Y$: $J_{(i\vec{y})(j\vec{y'})} = \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$

Matrix **J**

- In der $N \times N$ Matrix sind die Einträge $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$.
- Wir erinnern uns: $\delta\Psi_i(\vec{y}) \equiv \Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})$
- Statt einer einfachen Matrix haben wir einen Tensor, d.h. der Eintrag in die $N \times N$ Matrix ist eine $\mid Y \mid \times \mid Y \mid$ Matrix **J**:

	$\vec{y_1}$	...	$\vec{y_{\mid Y \mid}}$
$\vec{y_1}$	$-$	...	$\Psi(\vec{x}, \vec{y_1}) - \Psi(\vec{x}, \vec{y_{\mid Y \mid}})$
...	...	...	...
$\vec{y_{\mid Y \mid}}$	$\Psi(\vec{x}, \vec{y_{\mid Y \mid}}) - \Psi(\vec{x}, \vec{y_1})$	...	$-$

- **J** ist eine Kernfunktion über $X \times Y$: $J_{(i\vec{y})(j\vec{y'})} = \langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$

$\alpha_{i\vec{y}}$ und optimales β

- Statt α_i für $\vec{x_i}$, haben wir α_{ij} mit $j = 1, ..., |\ Y\ |$

$$\begin{pmatrix} \alpha_{i1} \\ ... \\ \alpha_{im} \end{pmatrix}$$

- Das optimale $\vec{\beta}$ ist

$$\begin{aligned} \hat{\vec{\beta}} &= \sum_{i=1}^{N} \sum_{\vec{y} \neq \vec{y_i}}^{|Y|} \alpha_{(i\vec{y})} (\Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})) \\ &= \sum_{i=1}^{N} \sum_{\vec{y} \neq \vec{y_i}}^{|Y|} \alpha_{(i\vec{y})} \delta\Psi_i(\vec{y}) \end{aligned} \tag{10}$$

$\alpha_{i\vec{y}}$ und optimales β

- Statt α_i für $\vec{x_i}$, haben wir α_{ij} mit $j = 1, ..., |\ Y\ |$

$$\begin{pmatrix} \alpha_{i1} \\ ... \\ \alpha_{im} \end{pmatrix}$$

- Das optimale $\vec{\beta}$ ist

$$\begin{aligned} \hat{\vec{\beta}} &= \sum_{i=1}^{N} \sum_{\vec{y} \neq \vec{y_i}}^{|Y|} \alpha_{(i\vec{y})} (\Psi(\vec{x_i}, \vec{y_i}) - \Psi(\vec{x_i}, \vec{y})) \\ &= \sum_{i=1}^{N} \sum_{\vec{y} \neq \vec{y_i}}^{|Y|} \alpha_{(i\vec{y})} \delta\Psi_i(\vec{y}) \end{aligned} \tag{10}$$

Duales Problem der SVMstruct

- SVMstruct bei linear separierbaren Beispielen:

$$L_D(\alpha) = -\frac{1}{2}\sum_{i,\vec{y}\neq\vec{y}_i}^{N}\sum_{j,\vec{y'}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}}\alpha_{j,\vec{y'}}J_{(i\vec{y})(j\vec{y'})} + \sum_{i,\vec{y}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}} \tag{11}$$

- Für die Slack Rescaling SVM_1 mit Ausnahmen muss zusätzlich gelten:

$$\sum_{\vec{y}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}} \leq \frac{C}{N}, \forall i = 1, ..., N$$

- Für die Margin Rescaling SVM_2 mit Ausnahmen wird $J_{(i\vec{y})(j\vec{y'})}$ unter Verwendung der Indikatorfunktion $I(a, b) = 1$ falls $a = b$, sonst 0 zu:

$$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'})\rangle + I(i,j)\frac{N}{C}$$

- Immer soll $\vec{\alpha}$ maximiert werden.

Duales Problem der SVMstruct

- SVMstruct bei linear separierbaren Beispielen:

$$L_D(\alpha) = -\frac{1}{2}\sum_{i,\vec{y}\neq\vec{y}_i}^{N}\sum_{j,\vec{y'}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}}\alpha_{j,\vec{y'}}J_{(i\vec{y})(j\vec{y'})} + \sum_{i,\vec{y}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}} \tag{11}$$

- Für die Slack Rescaling SVM_1 mit Ausnahmen muss zusätzlich gelten:

$$\sum_{\vec{y}\neq\vec{y}_i}^{N}\alpha_{i\vec{y}} \leq \frac{C}{N}, \forall i = 1, ..., N$$

- Für die Margin Rescaling SVM_2 mit Ausnahmen wird $J_{(i\vec{y})(j\vec{y'})}$ unter Verwendung der Indikatorfunktion $I(a, b) = 1$ falls $a = b$, sonst 0 zu:

$$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'})\rangle + I(i,j)\frac{N}{C}$$

- Immer soll $\vec{\alpha}$ maximiert werden.

Duales Problem der SVMstruct

- SVMstruct bei linear separierbaren Beispielen:

$$L_D(\alpha) = -\frac{1}{2} \sum_{i,\vec{y} \neq \vec{y}_i}^{N} \sum_{j,\vec{y'} \neq \vec{y}_i}^{N} \alpha_{i\vec{y}} \alpha_{j,\vec{y'}} J_{(i\vec{y})(j\vec{y'})} + \sum_{i,\vec{y} \neq \vec{y}_i}^{N} \alpha_{i\vec{y}} \qquad (11)$$

- Für die Slack Rescaling SVM_1 mit Ausnahmen muss zusätzlich gelten:

$$\sum_{\vec{y} \neq \vec{y}_i}^{N} \alpha_{i\vec{y}} \leq \frac{C}{N}, \forall i = 1, ..., N$$

- Für die Margin Rescaling SVM_2 mit Ausnahmen wird $J_{(i\vec{y})(j\vec{y'})}$ unter Verwendung der Indikatorfunktion $I(a, b) = 1$ falls $a = b$, sonst 0 zu:

$$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle + I(i, j)\frac{N}{C}$$

- Immer soll $\vec{\alpha}$ maximiert werden.

Duales Problem der SVMstruct

- SVMstruct bei linear separierbaren Beispielen:

$$L_D(\alpha) = -\frac{1}{2} \sum_{i,\vec{y} \neq \vec{y}_i}^{N} \sum_{j,\vec{y'} \neq \vec{y}_i}^{N} \alpha_{i\vec{y}} \alpha_{j,\vec{y'}} J_{(i\vec{y})(j\vec{y'})} + \sum_{i,\vec{y} \neq \vec{y}_i}^{N} \alpha_{i\vec{y}} \tag{11}$$

- Für die Slack Rescaling SVM_1 mit Ausnahmen muss zusätzlich gelten:

$$\sum_{\vec{y} \neq \vec{y}_i}^{N} \alpha_{i\vec{y}} \leq \frac{C}{N}, \forall i = 1, ..., N$$

- Für die Margin Rescaling SVM_2 mit Ausnahmen wird $J_{(i\vec{y})(j\vec{y'})}$ unter Verwendung der Indikatorfunktion $I(a, b) = 1$ falls $a = b$, sonst 0 zu:

$$\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle + I(i, j)\frac{N}{C}$$

- Immer soll $\vec{\alpha}$ maximiert werden.

Die SVMstruct stellt ein schwieriges Optimierungsproblem!

- Bei $N \mid Y \mid -N$ Nebenbedingungen und vermutlich sehr großem $\mid Y \mid$ ist normale Optimierung durch quadratische Programmierung nicht möglich.
- Es sollen nun deutlich weniger Nebenbedingungen wirklich bearbeitet werden.
- Beobachtung: Es gibt immer eine **Teilmenge** von Nebenbedingungen, so dass die damit errechnete Lösung auch **alle** Nebenbedingungen erfüllt mit einer Ungenauigkeit von nur ϵ.

Die SVMstruct stellt ein schwieriges Optimierungsproblem!

- Bei $N \mid Y \mid -N$ Nebenbedingungen und vermutlich sehr großem $\mid Y \mid$ ist normale Optimierung durch quadratische Programmierung nicht möglich.
- Es sollen nun deutlich weniger Nebenbedingungen wirklich bearbeitet werden.
- Beobachtung: Es gibt immer eine **Teilmenge** von Nebenbedingungen, so dass die damit errechnete Lösung auch **alle** Nebenbedingungen erfüllt mit einer Ungenauigkeit von nur ϵ.

Die SVMstruct stellt ein schwieriges Optimierungsproblem!

- Bei $N \mid Y \mid -N$ Nebenbedingungen und vermutlich sehr großem $\mid Y \mid$ ist normale Optimierung durch quadratische Programmierung nicht möglich.
- Es sollen nun deutlich weniger Nebenbedingungen wirklich bearbeitet werden.
- Beobachtung: Es gibt immer eine **Teilmenge** von Nebenbedingungen, so dass die damit errechnete Lösung auch **alle** Nebenbedingungen erfüllt mit einer Ungenauigkeit von nur ϵ.

SVMstruct: Algorithmus zum Optimieren – Idee

- Für jedes Beispiel $\vec{x_i}$ gibt es einen working set S_i, in dem die verletzten Nebenbedingungen gespeichert sind.
  Zunächst sind S_i leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x_i}$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y_i^*}$ festgestellt und S_i hinzugefügt.
  Das Problem wird zunehmend stärker beschränkt.
- Optimiere α
  - bezüglich aller working sets gemeinsam oder
  - nur für ein S_i, wobei die α_{jy} mit $j \neq i$ eingefroren werden.
- Wenn kein S_i mehr verändert wurde, STOP.

SVMstruct: Algorithmus zum Optimieren – Idee

- Für jedes Beispiel $\vec{x_i}$ gibt es einen working set S_i, in dem die verletzten Nebenbedingungen gespeichert sind.
  Zunächst sind S_i leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x_i}$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y_i^*}$ festgestellt und S_i hinzugefügt.
  Das Problem wird zunehmend stärker beschränkt.
- Optimiere α
  - bezüglich aller working sets gemeinsam oder
  - nur für ein S_i, wobei die α_{jy} mit $j \neq i$ eingefroren werden.
- Wenn kein S_i mehr verändert wurde, STOP.

SVMstruct: Algorithmus zum Optimieren – Idee

- Für jedes Beispiel $\vec{x_i}$ gibt es einen working set S_i, in dem die verletzten Nebenbedingungen gespeichert sind.
  Zunächst sind S_i leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x_i}$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y_i^*}$ festgestellt und S_i hinzugefügt.
  Das Problem wird zunehmend stärker beschränkt.
- Optimiere α
  - bezüglich aller working sets gemeinsam oder
  - nur für ein S_i, wobei die $\alpha_{j\vec{y}}$ mit $j \neq i$ eingefroren werden.
- Wenn kein S_i mehr verändert wurde, STOP.

SVMstruct: Algorithmus zum Optimieren – Idee

- Für jedes Beispiel $\vec{x_i}$ gibt es einen working set S_i, in dem die verletzten Nebenbedingungen gespeichert sind.
  Zunächst sind S_i leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x_i}$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y_i^*}$ festgestellt und S_i hinzugefügt.
  Das Problem wird zunehmend stärker beschränkt.
- Optimiere α
  - bezüglich aller working sets gemeinsam oder
  - nur für ein S_i, wobei die $\alpha_{j\vec{y}}$ mit $j \neq i$ eingefroren werden.
- Wenn kein S_i mehr verändert wurde, STOP.

SVMstruct: Algorithmus zum Optimieren – Idee

- Für jedes Beispiel $\vec{x}_i$ gibt es einen working set S_i, in dem die verletzten Nebenbedingungen gespeichert sind.
  Zunächst sind S_i leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x}_i$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y}_i^*$ festgestellt und S_i hinzugefügt.
  Das Problem wird zunehmend stärker beschränkt.
- Optimiere α
  - bezüglich aller working sets gemeinsam oder
  - nur für ein S_i, wobei die $\alpha_{j\vec{y}}$ mit $j \neq i$ eingefroren werden.
- Wenn kein S_i mehr verändert wurde, STOP.

SVMstruct: Algorithmus zum Optimieren – Idee

- Für jedes Beispiel $\vec{x}_i$ gibt es einen working set S_i, in dem die verletzten Nebenbedingungen gespeichert sind.
  Zunächst sind S_i leer, das Problem unbeschränkt.
- Für jedes Beispiel $\vec{x}_i$ wird die am schlimmsten verletzte Nebenbedingung bzgl. $\vec{y}_i^*$ festgestellt und S_i hinzugefügt.
  Das Problem wird zunehmend stärker beschränkt.
- Optimiere α
  - bezüglich aller working sets gemeinsam oder
  - nur für ein S_i, wobei die $\alpha_{j\vec{y}}$ mit $j \neq i$ eingefroren werden.
- Wenn kein S_i mehr verändert wurde, STOP.

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x}_1, \vec{y}_1), ..., (\vec{x}_N, \vec{y}_N)\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y}_i, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y}_i, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y}^* := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y}^*) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y}^*\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

SVMstruct: Algorithmus zum Optimieren

1. Input: $\mathcal{T} = \{(\vec{x_1}, \vec{y_1}), ..., (\vec{x_N}, \vec{y_N})\}, C, \epsilon$
2. $S_i := \{\}$ für alle $i = 1, ..., N$
3. Solange ein S_i sich in der Iteration ändert:
4. **for** $i = 1, ..., N$ **do**
5. Kosten: $H(\vec{y}) \begin{cases} 1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle & SVM_0 \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_1 \ (s.8) \\ (1 - \langle \delta\Psi_i(\vec{y}), \vec{\beta} \rangle)\Delta(\vec{y_i}, \vec{y}) & SVM_2 \ (s.9) \end{cases}$ wobei

   $\vec{\beta} \equiv \sum_j \sum_{\vec{y'} \in S_j} \alpha_{j\vec{y'}} \delta\Psi_j(\vec{y'})$
6. $\vec{y^*} := \arg\max_{\vec{y} \in Y} H(\vec{y})$ – schwieriger Schritt!
7. $\xi_i := max\{0, max_{\vec{y} \in S_i} H(\vec{y})\}$
8. **if** $H(\vec{y^*}) > \xi_i + \epsilon$ **then**
9. $S_i := S_i \cup \{\vec{y^*}\}$
10. $\alpha_S :=$optimiere duales Problem für $S = \cup S_i$

Full Parsing Learning mit SVMstruct

- Probabilistische kontextfreie Grammatik: Regeln $n_l[C_i \rightarrow C_j, C_k], \beta_l$. Dabei gibt β_l die logarithmierte Wahrscheinlichkeit dafür an, dass ein Knoten C_i mit Regel n_l expandiert wird.
- Lernaufgabe: Gegeben Paare $(\vec{x}, \vec{y})$, wobei $\vec{x} = x_1, ..., x_p$ ein Satz (Kette von Wortarten) ist und $\vec{y}$ ein Syntaxbaum, lerne $X \rightarrow Y$, wobei nur in den Beispielen vorkommende Regeln verwendet werden.
- Formuliert als Maximierungsproblem:

$$h(\vec{x}) = \arg\max_{\vec{y} \in Y} P(\vec{y}|\vec{x}) = argmax_{\vec{y} \in Y} \left\{ \sum_{n_l \in rules(\vec{y})} \beta_l \right\}$$

$rules(\vec{y})$ ist die Menge der Regeln, die in $\vec{y}$ verwendet sind.

Full Parsing Learning mit SVMstruct

- Probabilistische kontextfreie Grammatik: Regeln $n_l[C_i \rightarrow C_j, C_k], \beta_l$. Dabei gibt β_l die logarithmierte Wahrscheinlichkeit dafür an, dass ein Knoten C_i mit Regel n_l expandiert wird.
- Lernaufgabe: Gegeben Paare $(\vec{x}, \vec{y})$, wobei $\vec{x} = x_1, ..., x_p$ ein Satz (Kette von Wortarten) ist und $\vec{y}$ ein Syntaxbaum, lerne $X \rightarrow Y$, wobei nur in den Beispielen vorkommende Regeln verwendet werden.
- Formuliert als Maximierungsproblem:

$$h(\vec{x}) = \arg\max_{\vec{y} \in Y} P(\vec{y}|\vec{x}) = argmax_{\vec{y} \in Y} \left\{ \sum_{n_l \in rules(\vec{y})} \beta_l \right\}$$

$rules(\vec{y})$ ist die Menge der Regeln, die in $\vec{y}$ verwendet sind.

Full Parsing Learning mit SVMstruct

- Probabilistische kontextfreie Grammatik: Regeln $n_l[C_i \rightarrow C_j, C_k], \beta_l$. Dabei gibt β_l die logarithmierte Wahrscheinlichkeit dafür an, dass ein Knoten C_i mit Regel n_l expandiert wird.
- Lernaufgabe: Gegeben Paare $(\vec{x}, \vec{y})$, wobei $\vec{x} = x_1, ..., x_p$ ein Satz (Kette von Wortarten) ist und $\vec{y}$ ein Syntaxbaum, lerne $X \rightarrow Y$, wobei nur in den Beispielen vorkommende Regeln verwendet werden.
- Formuliert als Maximierungsproblem:

$$h(\vec{x}) = \arg\max_{\vec{y} \in Y} P(\vec{y}|\vec{x}) = argmax_{\vec{y} \in Y} \left\{ \sum_{n_l \in rules(\vec{y})} \beta_l \right\}$$

$rules(\vec{y})$ ist die Menge der Regeln, die in $\vec{y}$ verwendet sind.

Full Parsing Learning mit SVMstruct

- Es ergibt sich: $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle = \sum_{n_l \in rules(\vec{y})} \beta_l$
- Den schwierigen Schritt $\vec{y}^* := argmax_{\vec{y} \in Y} \langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ löst nun ein Parser, der sowohl den besten als auch den zweitbesten Syntaxbaum für $\vec{x}$ liefert. Somit können die Beispiele bei der Optimierung (Schritt 6) effizient bearbeitet werden.
- Das Lernergebnis ordnet bisher nicht gesehenen Sätzen $\vec{x}$ die richtigen Syntaxbäume zu. Dabei erweitert es die Fähigkeit der Grammatik – nicht die Menge der Regeln.

Full Parsing Learning mit SVMstruct

- Es ergibt sich: $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle = \sum_{n_l \in rules(\vec{y})} \beta_l$
- Den schwierigen Schritt $\vec{y}^* := argmax_{\vec{y} \in Y} \langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ löst nun ein Parser, der sowohl den besten als auch den zweitbesten Syntaxbaum für $\vec{x}$ liefert. Somit können die Beispiele bei der Optimierung (Schritt 6) effizient bearbeitet werden.
- Das Lernergebnis ordnet bisher nicht gesehenen Sätzen $\vec{x}$ die richtigen Syntaxbäume zu. Dabei erweitert es die Fähigkeit der Grammatik – nicht die Menge der Regeln.

Full Parsing Learning mit SVMstruct

- Es ergibt sich: $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle = \sum_{n_l \in rules(\vec{y})} \beta_l$
- Den schwierigen Schritt $\vec{y}^* := argmax_{\vec{y} \in Y} \langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ löst nun ein Parser, der sowohl den besten als auch den zweitbesten Syntaxbaum für $\vec{x}$ liefert. Somit können die Beispiele bei der Optimierung (Schritt 6) effizient bearbeitet werden.
- Das Lernergebnis ordnet bisher nicht gesehenen Sätzen $\vec{x}$ die richtigen Syntaxbäume zu. Dabei erweitert es die Fähigkeit der Grammatik – nicht die Menge der Regeln.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Experiment

- Trainingsmenge: 4098 Sätze mit maximal $p = 10$ Wörtern (dargestellt durch ihre Wortart)
- Testmenge: 163 Sätze mit maximal $p = 10$
- Maximum likelihood zum Lernen ergibt: $86,8\%$ precision, $85,2\%$ recall, $86\% F_1$ measure
- SVM_2 mit slack rescaling ergibt: $88,9\%$ precision, $88,1\%$ recall, $88,5\% F_1$ measure
- Der Unterschied des F-measures ist signifikant.
- SVM_2 hat in 12 Iterationen insgesamt 8043 Nebenbedingungen behandelt.
- Das Lernen dauerte insgesamt 3,4 Stunden, wovon die SVM_2 $10,5\%$ verwendete.

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y}_i, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y}_i, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y}_i, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y_i}, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y}_i, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y_i}, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y_i}, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Andere Anwendungen der SVMstruct

- Wenn man die SVMstruct anwenden will, muss man
  - die Merkmalsabbildung $\Psi(\vec{x}, \vec{y})$ definieren und ggf. implementieren
  - die Verlustfunktion implementieren $\Delta(\vec{y_i}, \vec{y})$
  - die Selektion verletzter Bedingungen (Schritt 6 des Algorithmus') implementieren.
- Klassifikation mit Taxonomien
- Named Entity Recognition
- Mehrklassen-Klassifikation
- ...

Was wissen Sie jetzt?

- Sie wissen, was strukturelle Modelle sind: Y kann mehr sein als nur ein Wert.
- $\Psi(\vec{x}, \vec{y})$ erweitert die üblichen Beispiele so, dass nun wieder ein Skalarprodukt $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ gerechnet werden kann.
- Das Problem sind die $N \times \mid Y \mid -N$ Nebenbedingungen, weil wir jedes Beispiel mit jedem anderen nicht nur bezüglich eines y, sondern bezüglich der $\mid Y \mid$ möglichen $\vec{y}$ vergleichen müssen.
- Dabei wird dieser Vergleich als *joint kernel* aufgefasst: $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$. Es gibt noch viele andere Arbeiten zu *string kernels, tree kernels*, die Sie hier nicht kennen gelernt haben.

Was wissen Sie jetzt?

- Sie wissen, was strukturelle Modelle sind: Y kann mehr sein als nur ein Wert.
- $\Psi(\vec{x}, \vec{y})$ erweitert die üblichen Beispiele so, dass nun wieder ein Skalarprodukt $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ gerechnet werden kann.
- Das Problem sind die $N \times \mid Y \mid -N$ Nebenbedingungen, weil wir jedes Beispiel mit jedem anderen nicht nur bezüglich eines y, sondern bezüglich der $\mid Y \mid$ möglichen $\vec{y}$ vergleichen müssen.
- Dabei wird dieser Vergleich als *joint kernel* aufgefasst: $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$. Es gibt noch viele andere Arbeiten zu *string kernels, tree kernels*, die Sie hier nicht kennen gelernt haben.

Was wissen Sie jetzt?

- Sie wissen, was strukturelle Modelle sind: Y kann mehr sein als nur ein Wert.
- $\Psi(\vec{x}, \vec{y})$ erweitert die üblichen Beispiele so, dass nun wieder ein Skalarprodukt $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ gerechnet werden kann.
- Das Problem sind die $N \times \mid Y \mid -N$ Nebenbedingungen, weil wir jedes Beispiel mit jedem anderen nicht nur bezüglich eines y, sondern bezüglich der $\mid Y \mid$ möglichen $\vec{y}$ vergleichen müssen.
- Dabei wird dieser Vergleich als *joint kernel* aufgefasst: $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$. Es gibt noch viele andere Arbeiten zu *string kernels, tree kernels*, die Sie hier nicht kennen gelernt haben.

Was wissen Sie jetzt?

- Sie wissen, was strukturelle Modelle sind: Y kann mehr sein als nur ein Wert.
- $\Psi(\vec{x}, \vec{y})$ erweitert die üblichen Beispiele so, dass nun wieder ein Skalarprodukt $\langle \vec{\beta}, \Psi(\vec{x}, \vec{y}) \rangle$ gerechnet werden kann.
- Das Problem sind die $N \times \mid Y \mid -N$ Nebenbedingungen, weil wir jedes Beispiel mit jedem anderen nicht nur bezüglich eines y, sondern bezüglich der $\mid Y \mid$ möglichen $\vec{y}$ vergleichen müssen.
- Dabei wird dieser Vergleich als *joint kernel* aufgefasst: $\langle \delta\Psi_i(\vec{y}), \delta\Psi_j(\vec{y'}) \rangle$. Es gibt noch viele andere Arbeiten zu *string kernels, tree kernels*, die Sie hier nicht kennen gelernt haben.

Sie wissen noch mehr!

- Der Ansatz von Joachims besteht darin,
  - dass als margin der Abstand zwischen der besten und der zweitbesten Lösung maximiert wird,
  - dass nur wenige der Nebenbedingungen wirklich behandelt werden müssen,
  - dass beliebige Verlustfunktionen Δ in den Nebenbedingungen und in der Auswahl der am stärksten verletzten Nebenbedingung verwendet werden können.

Sie wissen noch mehr!

- Der Ansatz von Joachims besteht darin,
  - dass als margin der Abstand zwischen der besten und der zweitbesten Lösung maximiert wird,
  - dass nur wenige der Nebenbedingungen wirklich behandelt werden müssen,
  - dass beliebige Verlustfunktionen Δ in den Nebenbedingungen und in der Auswahl der am stärksten verletzten Nebenbedingung verwendet werden können.

Sie wissen noch mehr!

- Der Ansatz von Joachims besteht darin,
  - dass als margin der Abstand zwischen der besten und der zweitbesten Lösung maximiert wird,
  - dass nur wenige der Nebenbedingungen wirklich behandelt werden müssen,
  - dass beliebige Verlustfunktionen Δ in den Nebenbedingungen und in der Auswahl der am stärksten verletzten Nebenbedingung verwendet werden können.

Sie wissen noch mehr!

- Der Ansatz von Joachims besteht darin,
  - dass als margin der Abstand zwischen der besten und der zweitbesten Lösung maximiert wird,
  - dass nur wenige der Nebenbedingungen wirklich behandelt werden müssen,
  - dass beliebige Verlustfunktionen Δ in den Nebenbedingungen und in der Auswahl der am stärksten verletzten Nebenbedingung verwendet werden können.

Literatur I