SVMstruct

Regina Fritsch
PG 520: Intelligence Service

9. Oktober 2007

Inhaltsverzeichnis

Einleitung

- SVMstruct soll komplexe Ausgaben miteinbeziehen
- es soll eine Zuordnung von einer Eingabe $x \in \mathcal{X}$ zu einer diskreten Ausgabe $y \in \mathcal{Y}$ gefunden werden, basierend auf einer Trainingsmenge von Eingabe-Ausgabe Paaren (x_1, y_1),...,$(x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$
- $\mathcal{Y}$ ist ein strukturierter Ausgaberaum
  - $y \in \mathcal{Y}$ sind z.B. Sequenzen, Strings, Bäume, Gitter, Graphen...
  - derartige Probleme gibt es bei einer Vielzahl von Anwendungen
- wir verallgemeinern die Large Margin Methode (Standard Multiclass - SVM) zu dem Problem, strukturierte Antworten zu finden

Diskriminanten und Verlustfunktionen

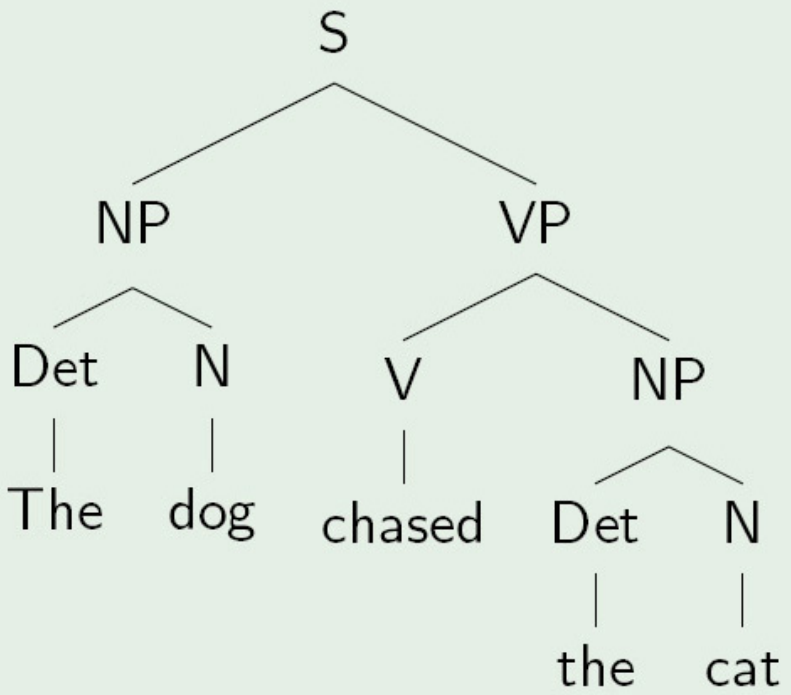
- Funktionenlernen $f : \mathcal{X} \rightarrow \mathcal{Y}$, basiert auf einer Trainingsmenge von Eingabe-Ausgabe Paaren

Bsp.: natürlichsprachliche Syntaxanalyse

x: The dog chased the cat

$\downarrow f : \mathcal{X} \rightarrow \mathcal{Y}$

y:



$$\Psi(x,y) = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ . \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{matrix} S \rightarrow \text{NP VP} \\ S \rightarrow \text{NP} \\ NP \rightarrow \text{Det N} \\ VP \rightarrow \text{V NP} \\ \\ Det \rightarrow \text{dog} \\ Det \rightarrow \text{the} \\ N \rightarrow \text{dog} \\ V \rightarrow \text{chased} \\ N \rightarrow \text{cat} \end{matrix}$$

Diskriminanten und Verlustfunktionen

- es soll eine Diskriminanzfunktion $F : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$ über die Eingabe/Ausgabe Paare gelernt werden, von der Vorhersagen abgeleitet werden können, indem F über die Rückgabevariable für die Eingabe x maximiert wird
  - f(x;w)= $\text{argmax}_{y \in \mathcal{Y}}$ F(x,y;w) mit w: Parameter-Vektor
- F soll linear in seiner kombinierten Merkmalsrepräsentation von Ein- und Ausgaben $\Psi(x, y)$ sein:
  F(x,y;w) = $\langle w, \Psi(x, y) \rangle$
- die spezifische Form von Ψ hängt von der Art des Problems ab

Diskriminanten und Verlustfunktionen

Bsp.: natürlichsprachliche Syntaxanalyse

- F soll so gewählt werden, dass ein Modell entsteht, dass einer kontextfreien Grammatik entspricht
- jeder Knoten, im Syntaxbaum y für einen Satz x, entspricht einer Grammatikregel g_j mit den Kosten w_j
- alle gültigen Syntaxbäume y für den Satz x bekommen eine Punktzahl zugewiesen, die sich aus der Summe der w_j seiner Knoten berechnet
  $\Rightarrow$ F(x,y;w) = $\langle w, \Psi(x, y)\rangle$
- $\Psi(x, y)$ ist ein Histogrammvektor:
  er zählt die Vorkommen jeder Grammatikregel g_j im Baum y
- f(x;w) kann effizient berechnet werden, indem die Struktur $y \in \mathcal{Y}$ gefunden wird, die F(x,y;w) maximiert

Diskriminanten und Verlustfunktionen

- das Lernen über strukturiereten Ausgaberäumen $\mathcal{Y}$, bezieht andere Verlustfunktionen mit ein als die Standard $0/1$ - Klassifikationsverluste

Bsp.: natürlichsprachliche Syntaxanalyse

Ein Syntaxbaum, der sich vom korrekten Syntaxbaum in nur wenigen Knoten unterscheidet, soll anders behandelt werden, als ein Syntaxbaum der sich von diesem vollkommen unterscheidet.

- deshalb gehen wir davon aus, dass eine beschränkte Verlustfunktion vorhanden ist:
  $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$
- wenn y der wahre Ausgabewert ist, mist $\Delta(y, \hat{y})$ den Verlust der mit der Vorhersage $\hat{y}$ verbunden ist

Margins und Margin-Maximierung

- separierbarer Fall: Trainingsfehler ist null
- Annahme: $\Delta(y, y') > 0$ mit $y \neq y'$ und $\Delta(y, y) = 0$
- die Bedingung, dass der Trainingsfehler null ist, soll kompakt dargestellt werden, als die Menge der nicht linearen Bedingungen:
  $\forall i : \max_{y \in \mathcal{Y} \setminus y_i} \{\langle w, \Psi(x_i, y)\rangle\} < \langle w, \Psi(x_i, y_i)\rangle$
- jede dieser nicht linearen Ungleichheiten, kann durch $|\mathcal{Y}| - 1$ lineare Ungleichheiten ersetzt werden:
  $\rightarrow$ das macht insg.: $n\,|\mathcal{Y}| - n$ lineare Bedingungen:
  $\forall i, \forall y \in \mathcal{Y} \setminus y_i$: $\langle w, \delta\Psi_i(y)\rangle > 0$
  mit $\delta\Psi_i(y) \equiv \Psi(x_i, y_i) - \Psi(x_i, y)$
- wenn die Menge dieser Ungleichheiten zulässig ist, gibt es typischerweise mehr als eine Lösung w^*

Margins und Margin-Maximierung

- um eine einzige Lösung zu erhalten:
  das w wählen, für das sich die Punktzahl des korrekten Labels y_i konstant am meisten von dem nahesten zweitplatzierten unterscheidet
  $\hat{y}_i(w) = \text{argmax}_{y \neq y_i} \langle w, \Psi(x_i, y) \rangle$
  $\rightarrow$ Generalisierung des Maximum-Margin-Prinzips angewandt in SVM
- das resultierende Hard-Margin-Optimierungsproblem:
  $SVM^{linear} : \min_w \frac{1}{2}\|w\|^2 ; \qquad \forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle w, \delta\Psi_i(y) \rangle \geq 1$
- um einen Fehler in der Trainingsmenge zu erlauben: Schlupfvariablen einführen
  für jede nicht lineare Bed. eine $\rightarrow$ obere Grenze des Trainingsfehlers
- das Einfügen einer Strafbedingung zu den Zielergebnissen, die linear in den Schlupfvariablen ist:
  $SVM^{nonlinear} : \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$, so dass $\forall i, \xi_i \geq 0$
  $\forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle w, \delta\Psi_i(y) \rangle \geq 1 - \xi_i$

Margins und Margin-Maximierung

- diese Formulierung für beliebige Verlust Funktionen Δ generalisieren
- zunächst den Maßstab der Schlupfvariablen ändern, und zwar gemäß des Verlustes der in jeder linearen Bedingung angefallen ist
- das Verletzen einer Margin-Bedingung die einen Ausgabewert mit hohem Verlust $\Delta(y_i, y)$ zur Folge hat, soll strenger bestraft werden, als ein Verstoß aus dem ein kleiner Verlust folgt
- dies wird erreicht, indem die Schlupfvariablen mit dem inversen Verlust skaliert werden:

$$SVM^{struct} : \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i, \text{ so dass } \forall i, \xi_i \geq 0$$
$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle w, \delta\Psi_i(y)\rangle \geq 1 - \frac{\xi_i}{\Delta(y_i, y)}$$

SVMstruct Algorithmus

- die Hauptaufgabe beim Lösen dieser verallgemeinerten SVM - Formulierung, liegt in der großen Anzahl der Margin-Bedingungen
  $\longrightarrow$ die Gesamtanzahl der Bedingungen beträgt $n\,|\mathcal{Y}|$
- $|\mathcal{Y}|$ ist in vielen Fällen extrem groß (z.B. Grammatik-Lernen)
  $\longrightarrow$ die Standardlösungen sind ungeeignet
- der Algorithmus soll die Struktur des Maximum-Margin-Problems ausnutzen
  $\longrightarrow$ nur noch eine kleine Teilmenge von Bedingungen detailiert betrachten

SVMstruct Algorithmus

- der Algorithmus soll eine kleine Menge von aktivierten Bedingungen finden, die eine ausreichend fehlerfreie Lösung garantieren
- dies gibt eine gute und gültige Lösung, da es immer eine polynomiell große Teilmenge von Bedingungen gibt, so dass die zugehörige Lösung alle Bedingungen mit einer Genauigkeit von mind. ϵ erfüllt
  $\longrightarrow$ die anderen Bedingungen werden garantiert durch nicht mehr als ϵ verletzt
- der folgende Algorithmus kann auf alle SVM^{struct}-Formulierungen angewandt werden
  $\longrightarrow$ nur anpassen der Kostenfunktion (Schritt 5)

SVMstruct Algorithmus

Algorithm 1: SVM-Struct

Input: (x_1, y_1), ..., (x_n, y_n), C, ϵ
$S_i \leftarrow \varnothing$ for all i = 1, ..., n
repeat
  for i = 1, ..., n **do**
    set up cost function [z.B: $SVM_1^{\Delta s}$: H(y) $\equiv (1 - \langle \delta\Psi_i(y), w \rangle)\Delta(y_i, y)$]
    compute $\hat{y} = \text{argmax}_{y \in \mathcal{Y}} H(y)$
    compute $\xi_i = \max\{0, \max_{y \in S_i} H(y)\}$
    if $H(\hat{y}) > \xi_i + \epsilon$ **then**
    $S_i \leftarrow S_i \cup \{\hat{y}\}$
    $\alpha_S \leftarrow$ optimize dual over S, $S = \cup_i S_i$
    end if
  end for
until no S_i has changed during iteration

SVMstruct Algorithmus

- es gibt eine Arbeitsmenge S_i für jedes Trainingsbeispiel (x_i, y_i) (Schritt 2) $\rightarrow$ festhalten der aktuell ausgewählten Bedingungen
- beim Abarbeiten der Trainingsbeispiele wird die Bedingung gesucht, die den größten Verlust zur Folge hat (Schritt 6)
- wenn dessen Margin-Überschreitung den aktuell errechneten Wert von ξ_i (Schritt 7) mit mehr als ϵ überschreitet (Schritt 8)
  $\rightarrow$ dann soll die Bedingung $\hat{y}$ zur Arbeitsmenge hinzufügt werden (Schritt 9)
  - das entspricht einer sukzessiven Verstärkung des grundlegenden Problem, durch eine Schnittebene, die die aktuelle Lösung von der zulässigen Menge entfernt
  - die gewählte Schnittebene entspricht der Bedingung, die den den kleinsten zulässigen Wert für ξ_i bestimmt

SVMstruct Algorithmus

- nachdem eine Bedingung zur Arbeitsmenge hinzugefügt wurde: $\longrightarrow$ die Lösung (in Bezug auf S) neu errechnen (Schritt 10)
- der Algorithmus stoppt, wenn keine Bedingung durch mehr als ϵ verletzt wird und sich somit kein S_i geändert hat (Schritt 13)
- es ist leicht zu zeigen, dass der Algorithmus eine Lösung nahe am Optimum findet
- für eine große Klasse von Problemen konvergiert der Algorithmus in polynomieller Zeit (trotz evtl. exponentiellen oder unendlichen $|\mathcal{Y}|$), denn die Anzahl der Bedingungen in S ist nicht abhängig von $|\mathcal{Y}|$
- wenn Schritt 6 polynomiell ist $\longrightarrow$ ist der Algorithmus polynomiell

Anwendungen und Experimente

- um die Effektivität und Vielseitigkeit dieses Ansatzes zu zeigen, folgen einige Ergebnisse zu einer Menge von verschiedenen Aufgaben
  - Multiclass Klassifikation
  - Klassifikation mit Taxonomien
  - **Label Sequence Learning**
  - Sequence Alignment (Sequenz Abgleich)
  - **Natural Language Parsing (natürlichsprachliche Syntaxanalyse)**
- eine Anpassung des Algorithmus an das neue Problem erfolgt durch die Implementierung von:
  - Merkmalsabbildung $\Psi(x, y)$
  - Verlustfunktion $\Delta(y_i, y)$
  - Maximierung des Verlusts (Schritt 6)

Anwendungen und Experimente: Label Sequence Learning

- eine Sequenz von Labeln $y = (y^1, ..., y^n), y^k \in \Sigma$ soll vorhergesagt werden, gegeben einer Sequenz von Eingaben $x = (x^1, ..., x^m)$
- Anwendung:
  - optische Zeichenerkennung, natürlichsprachliche Syntaxanalyse, Informationsgewinnung, Bioinformatik

Bsp.: NER

- eine Sammlung von 300 Sätzen
- die Zeichenmenge besteht aus Bezeichnungslosen und zusätzlich aus dem Anfang und der Weiterführung von
  - Personennamen
  - Organisationen
  - Orten
  - diversen Namen
- insgesamt 9 verschiedene Labels

Anwendungen und Experimente: Label Sequenz Lernen

Bsp.: NER

- $\Psi(x, y)$ ist ein Histogramm der Zustandsübergänge + eine Menge von Merkmalen die die Ausgaben beschreiben

Methode	HMM	CRF	Perceptron	SVM
Fehler	9.36	5.17	5.94	5.08

- die Standard HMM-Methode wird von allen Lernmethoden übertroffen

Methode	Train Fehler	Test Fehler	Const	Verlust
$SVM^{nonlinear}$	0.2 ± 0.1	5.1 ± 0.6	2824 ± 106	1.02 ± 0.01
SVM^{struct}	0.4 ± 0.4	5.1 ± 0.8	2626 ± 225	1.10 ± 0.08

- die Ergebnisse sind vergleichbar

Anwendungen und Experimente: natürlichsprachliche Syntaxanalyse

Bsp.:

- es soll eine gerichtete kontextfreie Grammatik erlernt werden
- **Trainingsmenge**: Sektionen F2-F21 $\rightarrow$ 4098 Sätze mit einer Wortanzahl von max. 10 Wörtern
- **Testmenge**: Sektion F22 $\rightarrow$ 163 Sätze (max. 10 Wörter)

Methode	Train		Test		Trainings Effizienz	
	Acc	F_1	Acc	F_1	Const	CPU(%QP)
PCFG	61.4	90.4	55.2	86.0	N/A	0
$SVM^{nonlinear}$	66.3	92.0	58.9	86.2	7494	1.2(81.6%)
SVM^{struct_1}	62.2	92.1	58.9	88.5	8043	3.4(10.5%)
SVM^{struct_2}	63.5	92.3	58.3	88.4	7117	3.5(18.0%)

Ergebnisse

- die Entwicklung einer SVM für überwachtes Lernen mit strukturierten und voneinander abhängigen Ausgaben
- dies basiert auf einer Merkmalsabbildung über Eingabe/Ausgabe - Paare und deckt damit eine breite Klasse von Modellen ab:
  - gewichtete kontextfreie Grammatiken
  - Hidden Markov Models
  - Sequence Alignment ...
- der Ansatz ist flexibel im Umgang mit methodenspezifischen Verlustfunktionen
- **ein** allgemeingültiger Algorithmus
- der Ansatz ist, für einen weiten Bereich, in der Genauigkeit mindestens vergleichbar mit den üblichen Ansätzen (oft besser)
- eine garantierte Eigenschaft: nutzbar um komplexe Modelle zu trainieren, die nur schwer im üblichen Rahmen behandelbar wären

Ende

Vielen Dank für Eure Aufmerksamkeit.