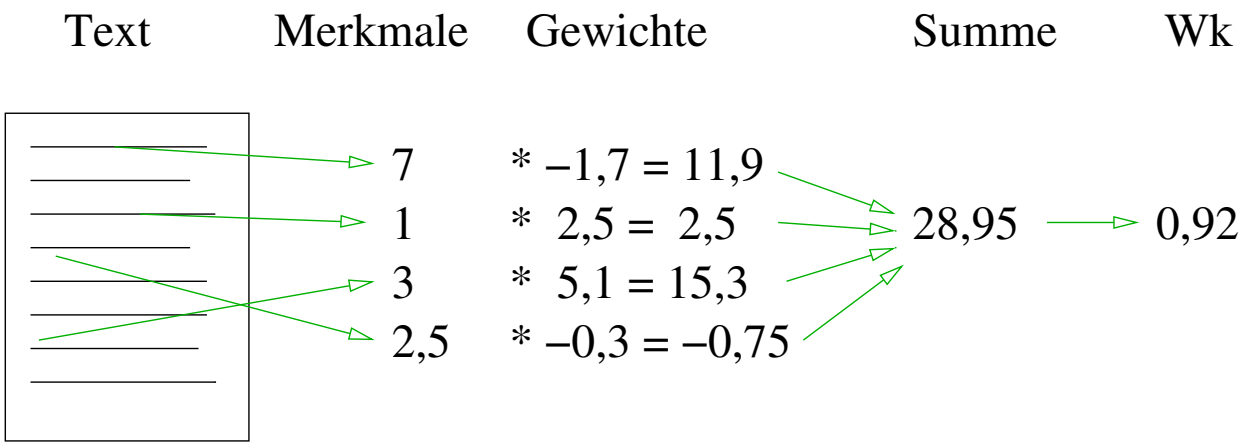
Log-lineare Modelle

Grundidee:

- Manuell erstellte Merkmalsfunktionen extrahieren relevante Information aus dem Dokument.
- Jeder Merkmalswert wird mit einem Gewicht multipliziert, das ausdrückt, ob das Merkmal auf eine bestimmte Klasse hindeutet.
- Die gewichteten Merkmalswerte werden aufsummiert und in Wahrscheinlichkeiten transformiert.



Merkmale

Log-lineare Modelle definieren eine Menge von Merkmalsfunktionen $f_1(c, d), ..., f_M(c, d)$

- binäre Merkmale
$$f_1(c, d) = \begin{cases} 1 & \text{falls Textlänge} \in [20, ..., 40) \text{ und } c = \text{negativ} \\ 0 & \textit{sonst} \end{cases}$$
- numerische Merkmale
$$f_2(c, d) = \begin{cases} n & \text{falls ,,toll'' n-mal auftaucht und } c = \text{positiv} \\ 0 & \text{falls } c \neq \text{positiv} \end{cases}$$

Merkmale

Beispiel: positive und negative Filmbewertungen erkennen

Das ist ein spannender Film aber die Handlung des Filmes ist sehr klischeehaft.

extrahierte Merkmale (nach Stoppwortentfernung und Lemmatisierung)

Klasse: positiv	
positiv, spannend	1
positiv, Film	2
positiv, Handlung	1
positiv, klischeehaft	1

Klasse: negativ	
negativ, spannend	1
negativ, Film	2
negativ, Handlung	1
negativ, klischeehaft	1

Gewichte

Jedem Merkmal $f_i(c, d)$ wird ein Gewicht θ_i zugeordnet

- $\theta_i > 0$: das Merkmal $f_i(c, d)$ deutet auf die Klasse c hin
- $\theta_i < 0$: das Merkmal $f_i(c, d)$ deutet auf eine andere Klasse als c hin

Die Gewichte sind beliebige reelle Zahlen.

Lineare Klassifikatoren

Die Merkmale $f_i(c, d)$ und Gewichte θ_i werden multipliziert und aufsummiert:

$$score(c, d) = \sum_i \theta_i f_i(c, d)$$

Die Werte können direkt zur Klassifikation verwendet werden (linearer Klassifikator)

$$\hat{c} = \arg\max_c score(c, d)$$

Wir wollen aber eine Wahrscheinlichkeitsverteilung definieren (also nicht-negative Werte, die zu 1 summieren)

Log-Lineare Modelle

Die "Scores" sind beliebige reelle Zahlen.

Exponentiation liefert nicht-negative Werte:

$$e^{score(c,d)} = e^{\sum_i \theta_i f_i(c,d)}$$

Normalisierung liefert Werte, die zu 1 summieren:

$$p(c|d) = \frac{1}{Z} e^{\sum_i \theta_i f_i(c,d)} \quad \text{mit } Z = \sum_{c'} e^{\sum_i \theta_i f_i(c',d)}$$

Ein solches Modell heißt log-lineares Modell.

Vektorschreibweise: $p(c|d) = \frac{1}{Z} e^{\theta \cdot \mathbf{f}(c,d)}$

Die Gewichte werden durch Training auf Dokumenten mit bekannter Klasse gelernt.

Log-Lineare Modelle: Training

Ziel des Trainings

Die Modell-Wahrscheinlichkeit der Trainingsdaten soll möglichst groß werden.

Log-Lineare Modelle: Training

Trainingsdaten: $D = \{(c_1, d_1), (c_2, d_2), ..., (c_N, d_N)\}$

Wir wollen den Gewichtsvektor $\hat{\theta}$ finden, der die bedingte Wahrscheinlichkeit (Likelihood) der korrekten Klassen maximiert:

$$\hat{\theta} = \arg\max_{\theta} \prod_{(c,d) \in D} p_\theta(c|d)$$

Statt der Wahrscheinlichkeit der Daten maximieren wir ihren Logarithmus (Log-Likelihood):

$$LL_\theta(D) = \sum_{(c,d) \in D} \log p_\theta(c|d)$$

Log-Lineare Modelle: Training

Trainingsdaten: $D = \{(c_1, d_1), (c_2, d_2), ..., (c_N, d_N)\}$

Wir wollen den Gewichtsvektor $\hat{\theta}$ finden, der die bedingte Wahrscheinlichkeit (Likelihood) der korrekten Klassen maximiert:

$$\hat{\theta} = \arg\max_{\theta} \prod_{(c,d) \in D} p_\theta(c|d)$$

Statt der Wahrscheinlichkeit der Daten maximieren wir ihren Logarithmus (Log-Likelihood):

$$LL_\theta(D) = \sum_{(c,d) \in D} \log p_\theta(c|d)$$

Log-Likelihood

Umformung des Ausdruckes:

$$
\begin{aligned}
LL_\theta(D) &= \sum_{(c,d)\in D} log\ p_\theta(c|d) \\
&= \sum_{(c,d)\in D} log \left(\frac{1}{Z_\theta(d)} e^{\sum_{i=1}^{m} \theta_i f_i(c,d)} \right) \\
&= \sum_{(c,d)\in D} \left(log \frac{1}{Z_\theta(d)} + log\ e^{\sum_{i=1}^{m} \theta_i f_i(c,d)} \right) \\
&= \left(\sum_{(c,d)\in D} \sum_{i=1}^{m} \theta_i f_i(c,d) \right) - \sum_{(c,d)\in D} log\ Z_\theta(d)
\end{aligned}
$$

Um das Maximum der Log-Likelihood in Abhängigkeit von den Parametern θ_j zu finden, setzen wir die partiellen Ableitungen nach θ_j gleich 0.

Log-Likelihood

Umformung des Ausdruckes:

$$
\begin{aligned}
LL_\theta(D) &= \sum_{(c,d)\in D} log\ p_\theta(c|d) \\
&= \sum_{(c,d)\in D} log \left(\frac{1}{Z_\theta(d)} e^{\sum_{i=1}^{m} \theta_i f_i(c,d)} \right) \\
&= \sum_{(c,d)\in D} \left(log \frac{1}{Z_\theta(d)} + log\ e^{\sum_{i=1}^{m} \theta_i f_i(c,d)} \right) \\
&= \left(\sum_{(c,d)\in D} \sum_{i=1}^{m} \theta_i f_i(c,d) \right) - \sum_{(c,d)\in D} log\ Z_\theta(d)
\end{aligned}
$$

Um das Maximum der Log-Likelihood in Abhängigkeit von den Parametern θ_j zu finden, setzen wir die partiellen Ableitungen nach θ_j gleich 0.

Log-Likelihood

Umformung des Ausdruckes:

$$
\begin{aligned}
LL_\theta(D) &= \sum_{(c,d)\in D} log\ p_\theta(c|d) \\
&= \sum_{(c,d)\in D} log \left(\frac{1}{Z_\theta(d)} e^{\sum_{i=1}^{m} \theta_i f_i(c,d)} \right) \\
&= \sum_{(c,d)\in D} \left(log \frac{1}{Z_\theta(d)} + log\ e^{\sum_{i=1}^{m} \theta_i f_i(c,d)} \right) \\
&= \left(\sum_{(c,d)\in D} \sum_{i=1}^{m} \theta_i f_i(c,d) \right) - \sum_{(c,d)\in D} log\ Z_\theta(d)
\end{aligned}
$$

Um das Maximum der Log-Likelihood in Abhängigkeit von den Parametern θ_j zu finden, setzen wir die partiellen Ableitungen nach θ_j gleich 0.

Ableitung der Log-Likelihood nach θ_j

$$\frac{\partial}{\partial \theta_j} \sum_{(c,d) \in D} \sum_{i=1}^{m} \theta_i f_i(c,d) - \sum_{(c,d) \in D} log\ Z_\theta(d) = 0$$

Ableitung des linken Ausdrucks:

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} \theta_i f_i(c,d) = f_j(c,d)$$

Ableitung des rechten Ausdrucks:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} log\ Z_\theta(d) &= \frac{\partial}{\partial \theta_j} log \sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} \\
&= \frac{1}{\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)}} \left(\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} f_j(c',d) \right) \\
&= \sum_{c'} p_\theta(c'|d)\ f_j(c',d) \\
&= E_{p_\theta(c'|d)}\ f_j(c',d)
\end{aligned}$$

Ableitung der Log-Likelihood nach θ_j

$$\frac{\partial}{\partial \theta_j} \sum_{(c,d) \in D} \sum_{i=1}^{m} \theta_i f_i(c,d) - \sum_{(c,d) \in D} log\ Z_\theta(d) = 0$$

Ableitung des linken Ausdrucks:

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} \theta_i f_i(c,d) = f_j(c,d)$$

Ableitung des rechten Ausdrucks:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} log\ Z_\theta(d) &= \frac{\partial}{\partial \theta_j} log \sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} \\
&= \frac{1}{\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)}} \left(\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} f_j(c',d) \right) \\
&= \sum_{c'} p_\theta(c'|d)\ f_j(c',d) \\
&= E_{p_\theta(c'|d)}\ f_j(c',d)
\end{aligned}$$

Ableitung der Log-Likelihood nach θ_j

$$\frac{\partial}{\partial \theta_j} \sum_{(c,d) \in D} \sum_{i=1}^{m} \theta_i f_i(c,d) - \sum_{(c,d) \in D} log\ Z_\theta(d) = 0$$

Ableitung des linken Ausdrucks:

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} \theta_i f_i(c,d) = f_j(c,d)$$

Ableitung des rechten Ausdrucks:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} log\ Z_\theta(d) &= \frac{\partial}{\partial \theta_j} log \sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} \\
&= \frac{1}{\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)}} \left(\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} f_j(c',d) \right) \\
&= \sum_{c'} p_\theta(c'|d)\ f_j(c',d) \\
&= E_{p_\theta(c'|d)}\ f_j(c',d)
\end{aligned}$$

Ableitung der Log-Likelihood nach θ_j

$$\frac{\partial}{\partial \theta_j} \sum_{(c,d) \in D} \sum_{i=1}^{m} \theta_i f_i(c,d) - \sum_{(c,d) \in D} log\ Z_\theta(d) = 0$$

Ableitung des linken Ausdrucks:

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} \theta_i f_i(c,d) = f_j(c,d)$$

Ableitung des rechten Ausdrucks:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} log\ Z_\theta(d) &= \frac{\partial}{\partial \theta_j} log \sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} \\
&= \frac{1}{\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)}} \left(\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} f_j(c',d) \right) \\
&= \sum_{c'} p_\theta(c'|d)\ f_j(c',d) \\
&= E_{p_\theta(c'|d)}\ f_j(c',d)
\end{aligned}$$

Ableitung der Log-Likelihood nach θ_j

$$\frac{\partial}{\partial \theta_j} \sum_{(c,d) \in D} \sum_{i=1}^{m} \theta_i f_i(c,d) - \sum_{(c,d) \in D} log\ Z_\theta(d) = 0$$

Ableitung des linken Ausdrucks:

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{m} \theta_i f_i(c,d) = f_j(c,d)$$

Ableitung des rechten Ausdrucks:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} log\ Z_\theta(d) &= \frac{\partial}{\partial \theta_j} log \sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} \\
&= \frac{1}{\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)}} \left(\sum_{c'} e^{\sum_{i=1}^{m} \theta_i f_i(c',d)} f_j(c',d) \right) \\
&= \sum_{c'} p_\theta(c'|d)\ f_j(c',d) \\
&= E_{p_\theta(c'|d)}\ f_j(c',d)
\end{aligned}$$

Log-Lineare Modelle

Zusammensetzen der Teilergebnisse:

$$\frac{\partial LL_\theta(D)}{\partial \theta_j} = \sum_{(c,d)\in D} f_j(c,d) - \sum_{(c,d)\in D} E_{p_\theta(c'|d)} f_j(c',d) = 0$$

$$\sum_{(c,d)\in D} f_j(c,d) = \sum_{(c,d)\in D} E_{p_\theta(c'|d)} f_j(c',d)$$

$\Rightarrow$ Die erwartete Häufigkeit jedes Merkmales muss gleich der beobachteten Häufigkeit werden.

$\Rightarrow$ Die partielle Ableitung ist gleich der Differenz der beiden Häufigkeiten.

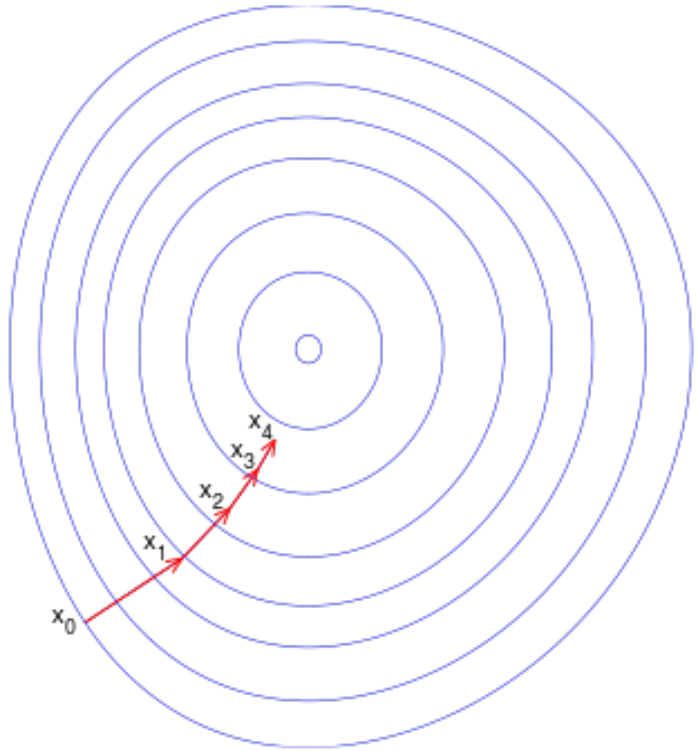
$\Rightarrow$ Der Gradient $\nabla LL_\theta(D)$ fasst alle partiellen Ableitungen zu einem Vektor zusammen.

Training mit Gradientenanstieg

Um die Likelihood zu erhöhen, müssen wir die Parameter in Richtung steigender Likelihood verändern (Gradientenanstieg).

Algorithmus

initialize θ
for N iterations do
$\theta \leftarrow \theta + \eta \nabla LL_\theta(D)$



aus Wikipedia

Die Lernrate η bestimmt, wie groß die einzelnen Schritte sind.

Wenn eine Funktion minimiert werden muss, spricht man von Gradientenabstieg.

Varianten von Gradientenanstieg

- (Batch-)Gradientenanstieg
  - berechnet die Ableitung für alle Daten zusammen
  - Anpassung der Gewichte nachdem alle Daten verarbeitet wurden
  - konvergiert langsam
- Stochastischer Gradientenanstieg
  - berechnet die Ableitung einzeln für jeden Datensatz
  - passt die Gewichte nach jedem Datensatz an
  - lernt schneller, aber konvergiert nicht
- Minibatch-Gradientenanstieg
  - berechnet die Ableitung für eine bestimmte Zahl von Datensätzen
  - passt die Gewichte nach der Verarbeitung jedes solchen Minibatches an
  - Eigenschaften zwischen Batch GD und SGD je nach Minibatchgröße

Overfitting

Bei begrenzten Trainingsdaten besteht die Gefahr des Overfittings.

Beipiel: (fälschlich als negativ klassifizierte Filmbewertung in Trainingsdaten)

Dieser Film ist unglaublich spannend. Die Hauptfigur Hugh Glass wird von Leonardo DiCaprio gespielt.

Wenn das Wort *Glass* nur in dieser Rezension auftaucht, dann wird das Gewicht des Merkmals *(Glass, negativ)* im Training so lange erhöht, bis die Rezension als negativ klassifiziert wird.

Das Problem taucht nicht nur bei Annotationsfehlern in den Trainingsdaten auf, sondern auch dann, wenn die vorhandenen Merkmale nicht alle notwendigen Informationen repräsentieren.

Regularisierung

Gegenmaßnahme:

Ziel-Funktion so modifizieren, dass große Gewichte ,,bestraft" werden:

$$LL_\theta(D) - \frac{\mu}{2} \sum_i \theta_i^2 \quad (L_2\text{-Regularisierung})$$

$$LL_\theta(D) - \mu \sum_i |\theta_i| \quad (L_1\text{-Regularisierung})$$

$\Rightarrow$ Große Gewichte sind nur erlaubt, wenn sie eine große Verbesserung der Likelihood bewirken.

Regularisierung

neuer Trainingsalgorithmus

initialize θ
for N iterations do

$$\delta \leftarrow \begin{cases} \mu\theta & \text{falls } L_2\text{-Regularisierung} \\ \mu sign(\theta) & \text{falls } L_1\text{-Regularisierung} \end{cases}$$

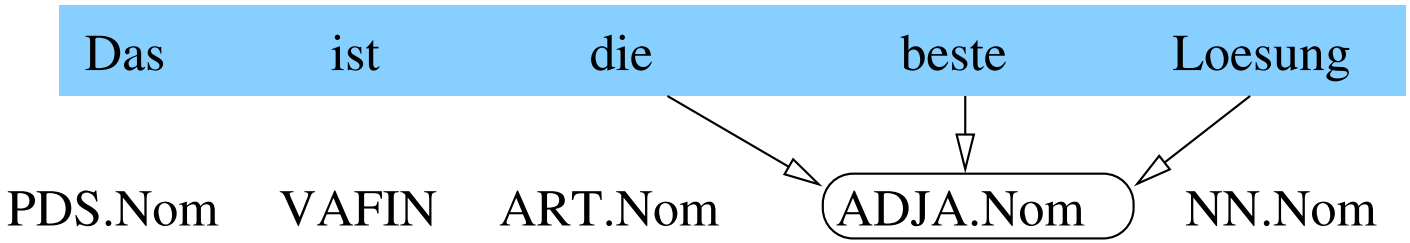
$$\theta \leftarrow \theta + \eta(\nabla LL_\theta(D) - \delta)$$

Taggen mit log-linearen Modellen

Log-lineare Modelle können auch zum Wortart-Taggen verwendet werden.

Es gibt 2 Möglichkeiten, einen Wort-Tagger zu implementieren:

1) Jedes Tag wird unabhängig von allen anderen Tags zugewiesen



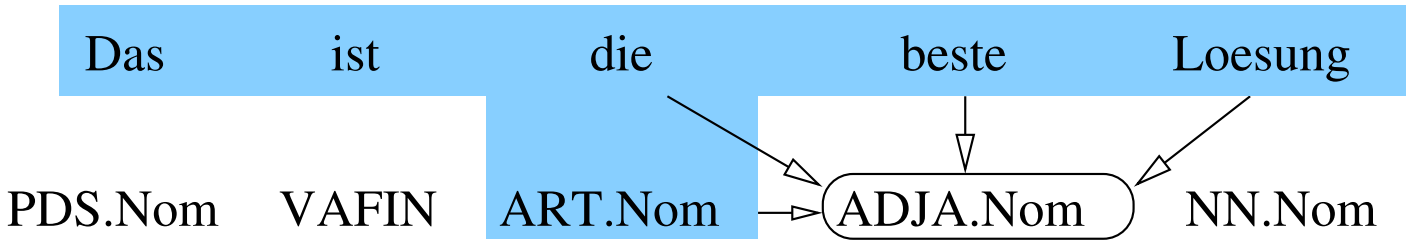
Problem: inkonsistente Ausgaben

Beispiel: Das ist die/Nom beste/Akk Lösung/Akk ./$.

Taggen mit log-linearen Modellen

2) Alle Wörter werden gemeinsam desambiguiert (analog HMMs)

Dabei hängt jedes Tag vom vorherigen Tag ab.



→ (Linear Chain) Conditional Random Field

Conditional Random Field

Für jede Wortposition wird eine Menge von Merkmalen extrahiert

1. lexikalische Merkmale (ohne Nachbartag-Information)
   Wort+Tag, Wortsuffix+Tag, Wort"shape"+Tag, voriges Wort+Tag, nächstes Wort+Wort+Tag, Wort+Tag im Lexikon
2. Kontext-Merkmale (mit Nachbartag-Information)
   voriges Tag+Tag, letzte2Tags+Tag, voriges Tag+Wort+Tag

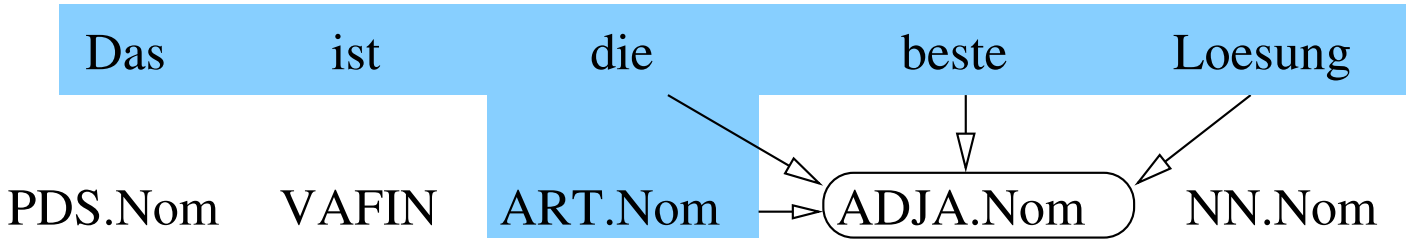
Die Merkmale f_k werden mit Gewichten θ_k multipliziert und über alle Positionen aufsummiert.

$$p(t_1^n|w_1^n) = \frac{1}{Z(w_1^n)} e^{\sum_{i=1}^{n+1} \sum_k \theta_k f_k(t_{i-m}^i, w_1^n, i)}$$

m: Zahl der Vorgängertags, von denen das nächste Tag abhängt

i iteriert bis $n+1$, weil ein Satzende-Tag $\langle/s\rangle$ hinzugefügt wird.

Conditional Random Fields



Jedes Tag darf abhängen von

- beliebigen Wörtern (einfaches log-lineares Modell)
- zusätzlich vom letzten Tag (LC-CRF 1. Ordnung)
- zusätzlich vom vorletzten Tag (LC-CRF 2. Ordnung)

wichtig: Bei einem LC-CRF sind Algorithmen der dynamischen Programmierung anwendbar (Viterbi, Forward-Backward)

Viterbi-Algorithmus (Bigramm-Tagger)

berechnet die beste Tagfolge (hier mit logarithmierten Viterbiwahrsch.)

lokaler Score: $s(t', t, w_1^n, i) := \sum_k \theta_k f_k(t', t, w_1^n, i)$

1. Initialisierung: $\delta_t(0) = \begin{cases} 0 & \text{falls } t = \langle s \rangle \\ -inf & \text{sonst} \end{cases}$

2. Berechnung:
$$\delta_t(i) = \max_{t'} \delta_{t'}(i-1) + s(t', t, w_1^n, i)$$
$$\psi_t(i) = \arg\max_{t'} \delta_{t'}(i-1) + s(t', t, w_1^n, i)$$

3. Ausgabe
$$t_{n+1} = \langle /s \rangle$$
$$t_{i-1} = \psi_{t_i}(i)$$

t und t' sind Tags, i ist eine Wortposition, $\delta_t(i)$ ein Viterbiwert und $\psi_t(i)$ das beste Vorgängertag.

Training von Conditional Random Fields

- Für das Training mit Gradientenanstieg werden die erwarteten Merkmalshäufigkeiten benötigt.
  - Wie bei den HMMs werden diese mit dem Forward-Backward-Algorithmus berechnet.
  - Hier wird er aber für überwachtes Training eingesetzt.
- Die erwarteten Häufigkeiten werden von den beobachteten Häufigkeiten in den Trainingsdaten subtrahiert, um den Gradienten zu erhalten.
- Der Gradient wird mit der Lernrate multipliziert und zum Gewichtsvektor addiert. (Gradientenanstieg)
- Diese Schritte werden N-mal wiederholt.

Training von Conditional Random Fields

$$p(t_1^n|w_1^n) = \frac{1}{Z(w_1^n)} \prod_{i=1}^{n+1} e^{\sum_k \theta_k f_k(t_{i-m}^i, w_1^n, i)}$$

Zur Berechnung der erwarteten Merkmalshäufigkeiten eines lex. Merkmals muss berechnet werden:

$$\frac{\sum \text{ Wk. aller Tagfolgen mit Tag } t \text{ an Position } i}{\sum \text{Wk. aller Tagfolgen insgesamt}}$$

- Die Normalisierungskonstanten $Z(w_1^n)$ können im Zähler und Nenner aus der Summe ausgeklammert und gekürzt werden.
- Die Summe im Zähler wird als Produkt von Forward- und Backward-Wahrscheinlichkeiten berechnet.
- Die Summe im Nenner (ohne Normalisierungskonstante) ist gleich dem Wert der Forward-Funktion für das Satzende-Tag.

Forward-Backward-Algorithmus (Modell 1. Ordnung)

Wenn T die Menge der Tags ist und $w_1^n = w_1 ... w_n$ die Wortfolge des Eingabesatzes der Länge n und θ_k das Gewicht des Merkmales k und $f_k(...)$ der Wert des Merkmales k, dann werden die Forward-Wahrscheinlichkeiten $\alpha_t(i)$, die Backward-Wahrscheinlichkeiten $\beta_t(i)$ und die Aposteriori-Wahrscheinlichkeiten $\gamma_t(i)$ des Tags $t \in T$ an der Satzposition i wie folgt berechnet:

$$z(t', t, w_1^n, i) := e^{s(t', t, w_1^n, i)} = e^{\sum_k \theta_k f_k(t', t, w_1^n, i)}$$

$$\begin{aligned}
\alpha_t(0) &= 1 \text{ falls } t = \langle s \rangle \text{ sonst } 0 \\
\alpha_t(i) &= \sum_{t' \in T} \alpha_{t'}(i-1)\, z(t', t, w_1^n, i) \quad \text{für alle } t \in T \text{ und } 0 < i \leq n+1 \\
\beta_t(n+1) &= 1 \text{ falls } t = \langle /s \rangle \text{ sonst } 0 \\
\beta_t(i-1) &= \sum_{t' \in T} \beta_{t'}(i)\, z(t, t', w_1^n, i) \quad \text{für alle } t \in T \text{ und } 0 < i \leq n+1 \\
\gamma_t(i) &= \frac{\alpha_t(i)\, \beta_t(i)}{\alpha_{\langle /s \rangle}(n+1)} \quad \text{für alle } t \in T \text{ und } 0 < i \leq n+1 \\
\gamma_{tt'}(i) &= \frac{\alpha_t(i-1)\, z(t, t', w_1^n, i)\, \beta_{t'}(i)}{\alpha_{\langle /s \rangle}(n+1)} \text{ für alle } t, t' \in T \text{ und } 0 < i \leq n+1
\end{aligned}$$