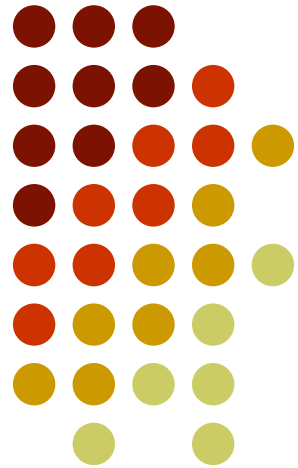
Markov Logik

Matthias Balwierz

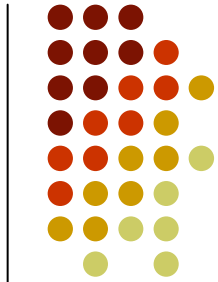
Seminar: Maschinelles Lernen

WS 2009/2010

Prof. Fürnkranz



Überblick



- Markov Netze
- Prädikatenlogik erster Stufe
- Markov Logik
- Inferenz
- Lernen
- Anwendungen
- Software

Markov Netze

- Statistisches Model zur Beschreibung von ungerichteten Zusammenhängen eines Netzwerkes
- Anwendungen:
  - Segmentierung von klassifizierten Flächen
  - Optisches Tracking und Matching
  - Magnetismus in Festkörpern

Markov Netze II

- Markov Netz besteht aus:
  - Menge von Variablen $X = (x_1, x_2, ..., x_n) \in \aleph$
  - Ein ungerichteter Graph
  - Eine Menge potenzieller Funktionen ϕ_k

- Graph hat einen Knoten pro Variable
- Ein Modell hat eine potenzielle Funktion pro Clique
- Eine potenzielle Funktion ist reell und nicht negativ

Markov Netze III

- Multivariate Verteilung:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

- Mit der Zustandssumme

$$Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$$

- Mit Zustand $x_{\{k\}}$ der *k*-ten Clique

Markov Netze IV

- Lässt sich als log-lineares Modell darstellen

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x) \right)$$

- Mit durch w_j gewichteten Features $f_j(x) \in \{0,1\}$
- Features können Funktionen sein die den Zustand einer Clique beschreiben

Prädikatenlogik erster Stufe

- Eine Wissensbasis besteht einer Menge von logischen Formeln
- Formeln bestehen aus
  - Konstanten aus dem Definitionsbereich (z.B. Anna, Bob, Chris)
  - Variablen
  - Funktionen (Abbildung von Tupel von Objekten auf Objekte z.B. MutterVon)
  - Prädikate (Relationen zwischen zwei Objekten oder Attribute z.B. Freunde oder Raucht)

Prädikatenlogik erster Stufe II

- Variablen und Konstanten können typisiert sein
- Ein Term ist eine Konstante, Variable oder eine Funktion angewendet auf Unterterme
- Eine atomare Formel ist ein Prädikat angewendet auf Unterterme
- Eine Formel ist rekursiv konstruiert aus atomaren Formeln verknüpft mit logischen Symbolen $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ und Quantoren $\exists x, \forall x$
- Atomare Formeln sind wahr, negierte sind unwahr

Prädikatenlogik erster Stufe III

- Beispiel
  - Fr(x, y): x und y sind Freunde
  - Sm(x): x ist Raucher
  - Ca(x): x hat Krebs

English	First-Order Logic	Clausal Form	Weight
Friends of friends are friends.	$\forall x \forall y \forall z\ \mathtt{Fr}(x,y) \wedge \mathtt{Fr}(y,z) \Rightarrow \mathtt{Fr}(x,z)$	$\neg\mathtt{Fr}(x,y) \vee \neg\mathtt{Fr}(y,z) \vee \mathtt{Fr}(x,z)$	0.7
Friendless people smoke.	$\forall x\ (\neg(\exists y\ \mathtt{Fr}(x,y)) \Rightarrow \mathtt{Sm}(x))$	$\mathtt{Fr}(x,g(x)) \vee \mathtt{Sm}(x)$	2.3
Smoking causes cancer.	$\forall x\ \mathtt{Sm}(x) \Rightarrow \mathtt{Ca}(x)$	$\neg\mathtt{Sm}(x) \vee \mathtt{Ca}(x)$	1.5
If two people are friends, either	$\forall x \forall y\ \mathtt{Fr}(x,y) \Rightarrow (\mathtt{Sm}(x) \Leftrightarrow \mathtt{Sm}(y))$	$\neg\mathtt{Fr}(x,y) \vee \mathtt{Sm}(x) \vee \neg\mathtt{Sm}(y)$,	1.1
both smoke or neither does.		$\neg\mathtt{Fr}(x,y) \vee \neg\mathtt{Sm}(x) \vee \mathtt{Sm}(y)$	1.1

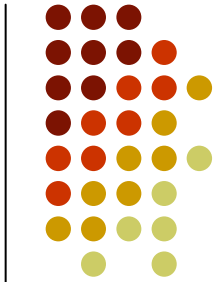
(Richardson et al 2004)

- Problem: Nicht jeder Raucher hat Krebs

Markov Logik

- Prädikatenlogik hat harte Randbedingungen
- Eine Welt welche eine Formel verletzt hat die Wahrscheinlichkeit 0
- Markov Logik weicht diese Randbedingung auf
- Eine Welt die eine Formel verletzt ist weniger wahrscheinlich aber nicht unmöglich
- Die „Härte“ einer Formel wird durch Gewichte ausgedrückt

Markov Logik II



- Definition:
- Ein MLN L ist eine Menge von Paaren (F_i, w_i)
  - F_i ist eine Prädikatenlogische Formel
  - w_i ist ein Gewicht
- Mit einer endlichen Menge von Konstanten $C = \{c_1, c_2, ..., c_{|C|}\}$ ergibt das ein Markov Netz $M_{L,C}$

Markov Logik III

Beispiel:

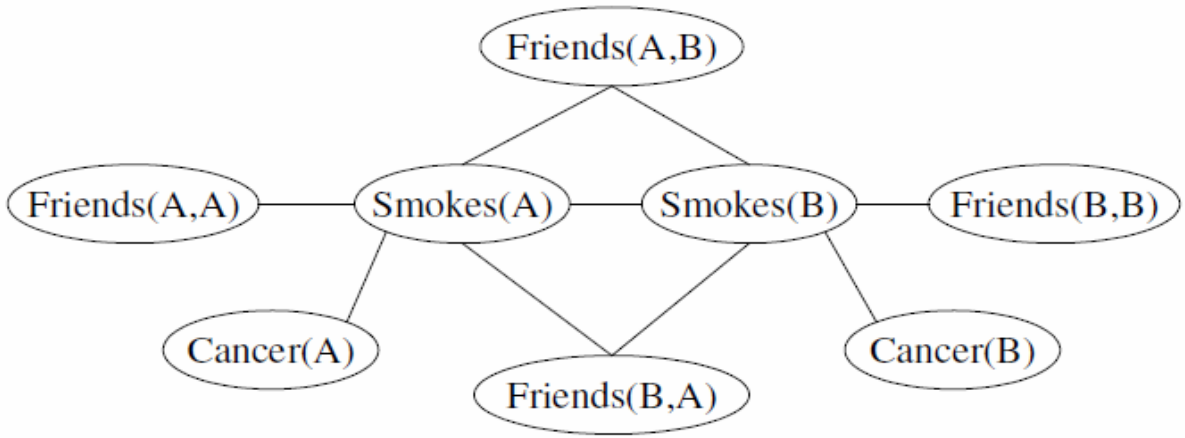
Freunde haben gleiches Rauchverhalten

$$\forall x \forall y Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$$

und Rauchen verursacht Krebs

$$\forall x Smokes(x) \Rightarrow Cancer(x)$$

mit Konstanten Anna und Bob



(De Raedt et al 2008)

Markov Logik IV

- MLN ist eine Schablone zum Konstruieren von Markov Netzen mit der Wahrscheinlichkeitsverteilung:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i}^{F} w_i n_i(x) \right)$$

- *F* ist die Anzahl der Formeln und n_i ist die Anzahl der wahren Groundings von F_i

Markov Logik V

- MLN mit der Formel $\forall x Smokes(x) \Rightarrow Cancer(x)$ und $C = \{A\}$

  ergeben sich vier mögliche Welten

$$\{\neg R(A), \neg C(A)\}, \{R(A), \neg C(A)\}, \{\neg R(A), C(A)\}, \{R(A), C(A)\}$$

  mit $P(\{R(A), \neg C(A)\}) = 1/(3e^w + 1)$ und $e^w/(3e^w + 1)$ für die anderen

- Markov Logik generalisiert die Prädikatenlogik mit $w \to \infty$

Inferenz

- Aufgabe: Finde den wahrscheinlichsten Zustand der Welt mit gegebenen Beweisen
  - Maximierungsproblem welches mit typischen Solvern gelöst werden kann (P-vollständig)
  - MaxWalkSAT
  - LazySAT noch besser
- Aufgabe: Wahrscheinlichkeit für eine Formel
  - MC-SAT

Inferenz II

Algorithm 1. MaxWalkSAT(*weighted_clauses*, *max_flips*, *max_tries*, *target*, *p*)

```
vars ← variables in weighted_clauses
for i ← 1 to max_tries do
   soln ← a random truth assignment to vars
   cost ← sum of weights of unsatisfied clauses in soln
   for i ← 1 to max_flips do
      if cost ≤ target then
         return "Success, solution is", soln
      end if
      c ← a randomly chosen unsatisfied clause
      if Uniform(0,1) < p then
         v_f ← a randomly chosen variable from c
      else
         for each variable v in c do
            compute DeltaCost(v)
         end for
         v_f ← v with lowest DeltaCost(v)
      end if
      soln ← soln with v_f flipped
      cost ← cost + DeltaCost(v_f)
   end for
end for
return "Failure, best assignment is", best soln found
```

(Domingos et al 2008)

Lernen

- Generative Gewichte:
  - Beispielwelt(en)
  - Geschlossen: Klauseln die nicht enthalten sind, sind falsch
  - Maximum-Likelihood-Methode

$$\frac{\partial}{\partial w_i} \log P_w(X = x) = n_i(x) - \sum_{x'} P_w(X = x') n_i(x')$$

  - Erfordert Inferenz, kann durch Pseudo-Likelihood optimiert werden (Markov Blanket)

Lernen II

- Diskriminative Gewichte
  - Partitionierung der Grund Atome in Beweise und „Queries“
  - Conditional Likelyhood
  - Schneller lösbar als Maximum-Likelihood

Lernen III

- Strukturen lernen:
  - Neue Klauseln finden
  - Alle atomaren Klauseln hinzufügen
  - Längere Klauseln bilden
    - Um (negiertes) Prädikat erweitern mit min einer gemeinsamen Konstante
  - Neue Kandidaten bewerten und beste(n) zu Klauseln hinzufügen

Lernen IV

- Bewertung über Likelihood
  - WPLL mit L-BFGS
  - Overfitting wird bestraft
- Weitere Regeln für bestehende Klauseln
  - Vorzeichenwechsel
  - Literale löschen
  - Variablen matchen (maximale Anzahl an Variablen)
  - Verfeinerte Klauseln löschen

Lernen V

function *FindBestClauses*(R, MLN, $Score$, $Clauses_0$, DB)
  inputs: R, a set of predicates
    MLN, a clausal Markov logic network
    $Score$, WPLL of MLN
    $Clauses_0$, a set of clauses
    DB, a relational database
  output: $BestClause$, a clause to be added to MLN
  $BestClause \leftarrow \emptyset$
  $BestGain \leftarrow 0$
  $Beam \leftarrow Clauses_0$
  Save the weights of the clauses in MLN
  repeat
    $Candidates \leftarrow$ *CreateCandidateClauses*($Beam$, R)
    for each clause $c \in Candidates$
      Add c to MLN
      LearnWeights(MLN, DB)
      $Gain(c) \leftarrow$ *WPLL*(MLN, DB) $- Score$
      Remove c from MLN
      Restore the weights of the clauses in MLN
    $Beam \leftarrow$ {The b clauses $c \in Candidates$ with highest
      $Gain(c) > 0$ and with $Weight(c) > \epsilon > 0$ }
    if $Gain$(Best clause c^* in $Beam$) $> BestGain$
      $BestClause \leftarrow c^*$
      $BestGain \leftarrow Gain(c^*)$
  until $Beam = \emptyset$ or $BestGain$ has not changed in two iterations
  return {$BestClause$}

(Kok et al 2005)

Anwendungen

- Web Mining
- Natural Language Processing
- Computational Biology
- Robotik
- Spiele
- Soziale Netzwerke

Software

- Alchemy
  - Open Source KI Framework
  - *alchemy.cs.washington.edu*

```
Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f’ => (!InField(i,+f,c) v !InField(i,+f’,c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i’,c’)
   ^ InField(i’,+f,c’) => SameField(+f,c,c’)
SameField(+f,c,c’) <=> SameCit(c,c’)
SameField(f,c,c’) ^ SameField(f,c’,c”)
   => SameField(f,c,c”)
SameCit(c,c’) ^ SameCit(c’,c”) => SameCit(c,c”)
```

(Domingos 2008)

Zusammenfassung

- Markov Logik
  - Vereinigt Wahrscheinlichkeitsnetze und Logik
  - Widersprüche machen eine Welt weniger wahrscheinlich aber nicht unmöglich
  - Wahrscheinlichste Zustände fehlender Information einer Welt können berechnet werden
  - Wahrscheinlichkeiten und Strukturen können aus Beispielen gelernt werden

Literatur

Luc De Raedt, Paolo Frasconi, Kristian Kersting, Stephen Muggleton (eds.) Probabilistic Inductive Logic Programming. Springer-Verlag, 2009.

Pedro Domingos. Markov Logic: A Unifying Language for Information and Knowledge Management. CIKM. (2008)

Lise Getoor and Ben Taskar, Introduction to Statistical Relational Learning, MIT Press. 2007.

Stanley Kok , Pedro Domingos, Learning the structure of Markov logic networks, Proceedings of the 22nd international conference on Machine learning, p.441-448, August 07-11, 2005, Bonn, Germany

M. Richardson, P. Domingos – Machine Learning, 2006 - Springer