Maschinelles Lernen 1 Markov Logik Netze (Topic 10)

Grundidee: Kombinieren von Prädikatenlogik erster Ordnung und probabilistisch grafischen Modellen (PGM)
-> „MLN sind gewichtete prädikatenlogische Formen"

Warum diese Kombination?
- -> Prädikatenlogik erster Ordnung gut für strukturierte Information
- -> Probabilistische Modelle gut für verrauschte Information (ggf. unstrukturiert)

Wozu verwendet: - Inferenz (für Auswertung von Anfragen)
- Lernen von Modellen

Prädikatenlogik erster Ordnung

- Symbole: Konstanten (z.B. Anna), Variablen (z.B. x), Funktionen, Prädikate (z.B. MotherOf(x), Friends(x,y)), Operatoren, Quantoren
- Wissensbasis: Formeln
- „Grounding": Ersetzen der Variablen durch Konstanten, z.B. Friends(Anna, Bob)
- „Interpretation": Zuordnung von Wahrheitswerten zu belegten Prädikaten („potentielle Welten")

Beispiel: aus der Wissensbasis bekannt: $\forall x: Smoking(x) \Rightarrow Cancer(x)$
und der Menge der Konstanten {A}

$$\forall x: Smoking(x) \Rightarrow Cancer(x) \equiv$$
$$\forall x: \neg Smoking(x) \vee Cancer(x)$$

Es gibt eine Belegung und vier potentielle Welten:

{S(A), C(A)}: {false, false}, {true, false}, {false, true}, {true, true}

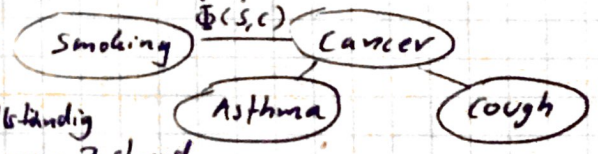
Davon eine unmögliche (die die Formel nicht erfüllt):
{S(A), C(A)}: {true, false}

Markov Netz - Verbundwahrscheinlichkeit

Allgemein: PGM beschreiben Verbundwahrscheinlichkeiten über die Mengen von Zufallsvariablen
- Knoten repr. Zufallsvariablen
- Graph kodiert Abhängigkeiten

Einschub:
- Bayes'sche Netze -> gerichtete azyklische Graphen (Kausalität)
- Markov'sche Netze -> ungerichtete Graphen (Korrelationen)

Ungerichteter Graph mit Clique:



Potentialfunktion für Cliquen (vollständig verbundener Teilgraph) abhängig vom Zustand der Zufallsvariablen definieren Verbundwahrsch.:

$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

Norm. →

Umformung der Verbundwahrsch. für binäre Zufallsvariablen:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x)\right)$$

f_i - Zustand der i-ten Clique c
w_i - $\log(\Phi_c)$

Gewicht Feature i ← Feature i (Indikatorfkt.)

z.B. $w_1 = 1.5 = \log(4,5)$

i	Smoking	Cancer	$\Phi(S,C)$
1	false	false	4.5

(2)

$f_i(\text{Smoking}, \text{Cancer}) = \begin{cases} 1 & \text{if } \neg\text{Smoking} \vee \text{Cancer} \\ 0 & \text{else} \end{cases}$

Grundidee: Markov Logik
- Logische Wissensbasis (harte Entscheidungen)
- harte Entscheidungen aufweichen => jede Formel wird gewichtet (je höher G. desto stärker der Einfluss)
(Idee: wenn eine Welt eine Regel (Formel) verletzt, wird sie weniger wahrscheinlich)

Gesamtwahrscheinlichkeit einer Welt: $P(\text{Welt}) \propto \exp\left(\sum_{\substack{\text{Formel} \\ \text{an Welt}}} \text{Gewicht der Formel die erfüllt ist}\right)$

Markov Logik Netz Definition
- Markov Logik Netz (MLN) ist Menge von Tupeln (F_i, w_i), wobei: (Syntax)
  - F_i ist eine Formel in Prädikatenlogik erster Ordnung
  - w_i ist eine reelle Zahl

  => daraus Markov Netz erzeugen

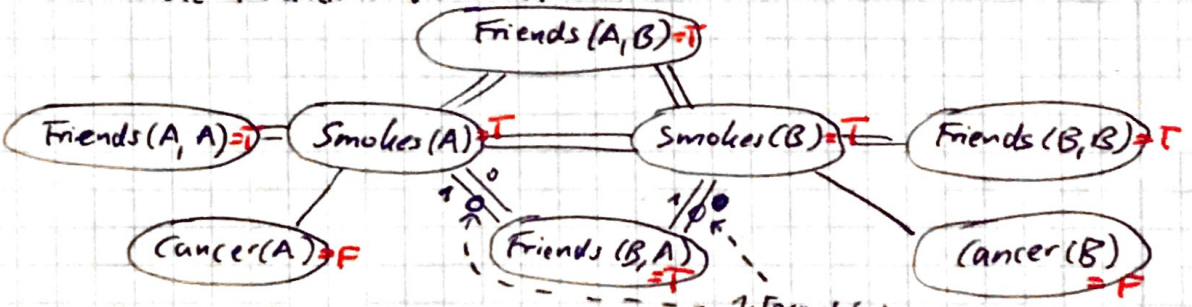
- Mit je einer Menge von Konstanten wird daraus ein Markov Netz, definiert mit: (Semantik)
  - je einem Knoten für jede mögliche Belegung jedes Prädikats des MLN
  - je einem Feature für jede Belegung (grounding) jeder Formel F_i des MLN mit dem entsprechendem Gewicht w_i

Beispiel: ① Prädikatenlogische Formeln: $\forall_x \text{Smokes}(x) \Rightarrow \text{Cancer}(x)$
$\forall_{x,y} \text{Friends}(x,y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

② Umformen und gewichten
1.5 $\forall x \neg\text{Smokes}(x) \vee \text{Cancer}(x)$
1.1 $\forall x,y \neg\text{Friends}(x,y) \vee (\neg\text{Smokes}(x) \vee \text{Smokes}(y))$ (a)
" $\forall x,y \neg\text{Friends}(x,y) \vee (\text{Smokes}(x) \vee \neg\text{Smokes}(y))$ (b)

③ Zwei Konstanten: Anna (A) und Bob (B)
(aus MLN ein MN generieren)

④ Markov Netz instanziieren & aufbauen:



in rot: ein Bsp für eine Welt, aus • ergibt sich:
- $P(S(A)) = T$ - $P(S(B)) = T$ - $P(F(B,A)) = T$ - $P(C(A)) = F$ etc.

=> $P(x) = (1.5 \cdot 0) + (1.1 \cdot 4) + (1.1 \cdot 4)$ •

- MLN ist eine Schablone (Template) für den Aufbau eines Markov Netzes
- Wahrsch. einer Welt:

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) \qquad Z = \sum_x \sum_i w_i n_i(x)$$

$n_i(x)$: Anzahl der wahren Belegungen der Formel (Klausel) i in x
w_i: Gewicht der Formel i
Z: Normierung

-> typisierte Variablen & Konstanten reduzieren sehr stark die Größe des aufgebauten Markov Netzes
-> Möglichkeit der Nutzung Funktionen, Quantoren, endliche / unendliche Domänen

Atom = Friends(x, mother(x))
von Formel (cancer(x) ∧ Friends(...)

(3)

Vergleich zu Prädikatenlogik erster Ordnung

- Wenn erfüllbare Wissensbasis, d.h. Welten bei der alle Formeln wahr sind:
  → erfüllende Interpretationen (Welten) sind die Modalwerte (in der Verteilung häufigst/vorkommende Welt) der Verteilung
  → Vorteil: auch bei verrauschten Daten sind die Welten enthalten, die durch PL ausgedrückt werden
  ⇒ Markov Logik erlaubt Widersprüche unter den Formeln
  -> wichtig bei verrauschten/unvollständigen Daten

Inferenz über MAP/MPE (most probable estimate)

Problem: Welcher Zustand der Welt (y) ist am wahrscheinlichsten, gegeben Evidenzen x -> z.B. durch Festhalten einer Zuordnung (Friends(A,A) = T) und Kombinieren der anderen Zuordnungen

Evidenz: fixe/bekannte Teil-Variablenbelegung aus Beobachtung bekannt

Formal:

$$\arg\max_y P(y \mid x)$$

- y: „Query" / Variablenbelegung => Output
- x: „Evidence" / beobachtete Variablenbelegung => Input (oder ohne Beobachtung: $P(y)$)

MLN einsetzen:

$$\arg\max_y \sum_i w_i n_i(x, y)$$

(Allgemein:) man sucht eine Variablenbelegung, sodass möglichst viele Formeln erfüllt sind => „weighted MaxSat" Problem)

- SAT = satisfiabel belief/satisfiability-solver -> alle Formeln erfüllen
- MaxSAT -> so viele Formeln wie möglich erfüllen
- Weighted MaxSat -> max. gewichtete Anzahl von Formeln

=> SAT-Solver potentiell schneller für weighted als für logischen (unweighted) Inferenz

(Exkurs) Möglichkeit: Weighted MaxWalkSAT Algorithmus (speicher-)
-> hohe Speicherkomplexität, exp. Aufwand, n Konstanten
c Variablen -> n^c

Praxis: dünnes Netz aufbauen (nur wichtigste Klauseln)

Anfragen und Wahrscheinlichkeiten

Häufige Fragestellung: wie wahrscheinlich ist die Erfüllung einer Formel, gegeben ein MLN und Konstanten C (z.B. Raucher)

$$P(\text{Formel} \mid \text{MLN}, C)$$

=> entspricht der Summe der Wahrscheinlichkeiten der Welten in denen die Formel erfüllt ist

=> Problem: alle Zuordnungskombinationen berechnen -> exp. Anzahl von Welten

(Exkurs) => Lösung: - MCMC -> Stochastisches Abtasten der Welten (wiederholtes Sampeln von Belegungen für Variablen und dabei MAP betrachten)
(- Testen ob Formel erfüllt ist)

Lernen von MLN

Gegeben: - Daten (beobachtete Welten) (mit Werten belegte Prädikate)
- geschlossene Welt
  ↳ jedes Atom (z.B. Anna und Bob sind Freunde), das nicht in den Daten vorkommt ist falsch
- Formeln gegeben

Gesucht: a) Parameter / Gewichte
- Generatives oder Diskriminatives Lernen

b) Struktur (Formeln und Gewichte unbekannt) (Strukturlernen)

Generatives Lernen von Gewichten (lernt Datenverteilung) => lernt $P(x,y)$ $\overset{\text{Bayes}}{=}$ $P(x|y)\,P(y)$

Ziel: Gewichte so wählen, dass sie am wahrscheinlichsten die Daten generieren (iterativ)
=> Maximum Likelihood $\max P_w(x)$ [eig. $P_w(x,y)$] ($P(X)$) „Wahrsch. einer Welt" ableiten

z.B. via Gradientenabstiegsverfahren

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

- $n_i(x)$: Anzahl der wahren Belegungen der Formel i in den Daten
- $E_w[n_i(x)]$: erwartete Anzahl der wahren Belegungen (= Inferenz!) entsprechend dem aktuellen Modell
- $P_w(X=x')$: Normierungsterm, hier a priori
- $n_i(x')$: Anzahl gew. wahre Belegungen

=> benötigt Inferenz in jedem Schritt (Verbundwahrsch. „a priori" $P_w(x')$) -> langsam
=> konvex, keine lokalen Minima

Diskriminatives Lernen von Gewichten (lernt Klassengrenzen / Cubes) => lernt $P(y|x)$

Maximieren der bedingten Wahrsch. von y (der Variablenbelegung) gegeben x (Evidenzen) <=> Teilbelegungen

$$\frac{\partial}{\partial w_i} \log P_w(y|x) = n_i(x,y) - \sum_{y'} P_w(y'|x)\, n_i(x,y') = n_i(x,y) - E_w[n_i(x,y)]$$

$E_w[n_i(x,y)]$ ← inferieren; Evidenzen müssen nicht inferiert werden

=> Erwartungswerte berechnen weniger teuer, weil weniger Inferenzschritte (Anzahl wahre Belegungen der Evidenzformeln bekannt, teilweise unabhängig der y-Variablenbelegung, nur y-Belegungen müssen auf Wahrheit geprüft werden

Strukturlernen (zufällig)
- Atomare Klauseln oder vorgegebene Wissensbasis werden initialisiert
- Ändern von Operatoren (Hinzufügen, Löschen, Negieren...)
- Lernen der Gewichte
=> sehr heuristisch!