Universität Stuttgart
Institut für
Maschinelle Sprachverarbeitung

Probabilistische Graphische Modelle in einer Neuronalen Welt

Wie können zwei Paradigmen vereint werden?

Mündliche Habilitationsleistung

17. Juli 2020

Roman Klinger
roman.klinger@ims.uni-stuttgart.de

@roman_klinger romanklinger
http://www.romanklinger.de/

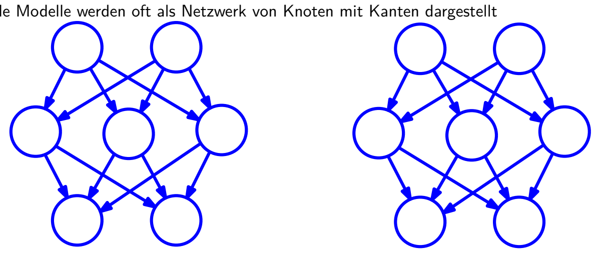
Folien und Literatur

Sie finden
- diese Folien
- das Literaturverzeichnis

unter:
- https://www.romanklinger.de/publications/habilitation/muendl/
- Kurzlink: https://tinyurl.com/RKHabil

Probabilistische Graphische Modelle und Neuronale Netze

Beide Modelle werden oft als Netzwerk von Knoten mit Kanten dargestellt



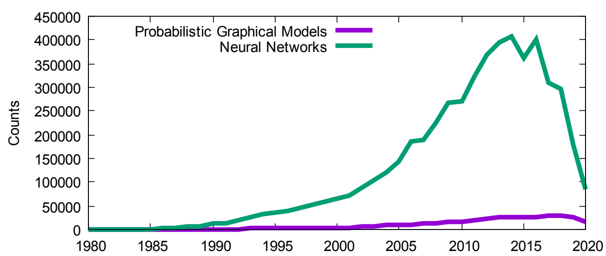
Directed Probabilistic Graphical Model

Feed-forward Neural Network

Probabilistische Graphische Modelle und Neuronale Netze

Beide Modellparadigmen…
- …werden als Graphen visualisiert.
- …sind Methoden des maschinellen Lernens.
- …können strukturierte Eingaben verarbeiten und strukturierte Ausgaben machen.
- …verwenden Algorithmen, welche Informationen durch das Netzwerk propagieren.
- …sind populär.

Probabilistische Graphische Modelle und Neuronale Netze



Ziele dieses Vortrags

- Was sind künstliche neuronale Netze (KNN) und was sind probabilistische graphische Modelle (PGM)?
- Neuronale Netze und Probabilistische Modelle werden beide als Graphen mit Knoten und Kanten dargestellt: Was sind Gemeinsamkeiten und was sind Unterschiede?
- Haben KNNs vielleicht PGMs obsolet gemacht?
- Haben sie unterschiedliche Verwendungszwecke? Bestehen jeweils Vorteile oder Nachteile?
- Wie können beide Paradigmen kombiniert werden?

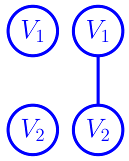
Vorgehen

- Einführung in PGMs und KNN (mit Schwerpunkt auf PGMs)
- Herausarbeiten von Unterschieden und Gemeinsamkeiten
- Verstehen von Herausforderungen und Chancen bei der Kombination
- Darstellung von möglichen zukünftigen Forschungsschritten

Outline

Probabilistische Graphische Modelle und Wahrscheinlichkeitsverteilungen

$p(x_1, x_2) = p(x_1)p(x_2)$



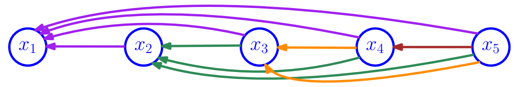
- PGM G ist Visualisierung von Wahrscheinlichkeitsverteilung p
- Zufallsvariablen x_i entsprechen Knoten V_i
- Kanten $E = (V_i, V_j)$ stellen Zusammenhänge zwischen x_i und x_j dar
- G ist ... von p, wenn ...
  - Dependency-Map (D-Map): $x_i \perp x_j \Rightarrow$es gibt keine Kante $E = (V_i, V_j)$ im Graph G
  - Independency-Map (I-Map): Es gibt keine Kante $E = (V_i, V_j) \Rightarrow x_i \perp x_j$
  - Perfect Map: G ist D-Map und I-Map

⇒ Wir sind in der Regel an I-Maps interessiert, wenn wir Modelle graphisch erstellen.

(Beierle/Kern-Isberner 2003, Koller/Friedman 2010)

Gerichtete Probabilistische Graphische Modelle: Faktorisierung und Visualisierung

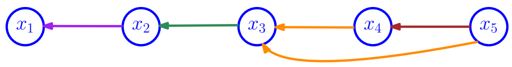
- Wie kommen wir nun von einer Verteilung zu einem Graph – und umgekehrt?
- Idee: Faktorisiere Verteilung mit der Kettenregel: $p(x_1, \ldots, x_n) = \prod_{i=1}^n p(x_i | x_{i+1}, \ldots, x_n)$
- $p(x_1, x_2, x_3, x_4, x_5) = p(x_1 | x_2, x_3, x_4, x_5) \cdot p(x_2 | x_3, x_4, x_5) \cdot p(x_3 | x_4, x_5) \cdot p(x_4 | x_5) \cdot p(x_5)$



- Diese Darstellung gilt für jede Verteilung und ist daher wenig hilfreich,...
- ...aber wir können mit dem graphischen Modell Unabhängigkeiten implizieren.

⇒ Weniger Parameter, mit Vorannahmen angepasstes Modell

Gerichtete Probabilistische Graphische Modelle: Unabhängigkeitsannahmen

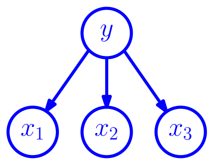


$p(x_1, x_2, x_3, x_4, x_5) = p(x_1 | x_2) \cdot p(x_2 | x_3) \cdot p(x_3 | x_4, x_5) \cdot p(x_4 | x_5) \cdot p(x_5)$

- Erinnerung: Wir nutzen den graphischen Formalismus um Aussagen über Annahmen in der Wahrscheinlichkeitsverteilung zu machen.

Populäre Beispiel-PGMs

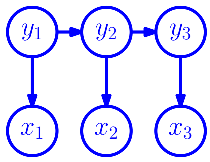
Naïve Bayes



$p(y, \vec{x}) = p(y) \prod_{i=1}^n p(x_i | y)$

(Domingos, 1997)

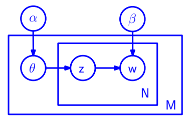
Hidden Markov Model



$p(\vec{y}, \vec{x}) = p(y_1) \prod_{i=1}^n p(y_i | y_{i-1}) p(x_i | y_i)$

(Rabiner, 1989)

Latent Dirichlet Allocation



$p(\vec{w}, \vec{z}, \vec{\theta}, \alpha, \beta) = p(\beta) \cdot \prod_{j=1}^M (p(\theta_j \mid \alpha) \cdot \prod_{t=1}^N (p(z_{j,t} \mid \theta_j) p(w_{j,t})))$

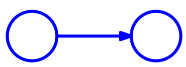
(Blei, 2003)

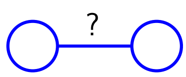
Definition: Gerichtetes Graphisches Modell

Ein gerichtetes graphisches Modell, oder Bayesisches Netzwerk:

- ist ein gerichteter azyklischer Graph
  - Knoten entsprechen Variablen
- Jeder Knoten hat eine bedingte Wahrscheinlichkeitsverteilung: $p(x_i \mid \mathrm{Eltern}(x_i))$
- Das Netzwerk stellt dann eine gemeinsame Wahrscheinlichkeitsverteilung dar:

$$p(x_1, \ldots, x_n) = \prod_i p(x_i \mid \mathrm{Eltern}(x_i))$$

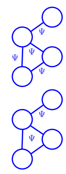




Definition: Ungerichtete Probabilistische Graphische Modelle

- Ein paarweises Markov-Netzwerk ist ein ungerichteter Graph mit Knoten, welche Variablen $x_1, \ldots, x_n$ darstellen
  - Kante $E = (x_i, x_j)$ ist mit Potentialfunktion $\Psi(x_i, x_j)$ verknüpft
- Ein Markov-Netzwerk G mit Cliquen-Faktorisierung hat Potenzialfunktionen welche (maximalen) Cliquen entsprechen

$$p(\vec{x}) = \frac{1}{Z} \prod_{C \in \mathrm{cliques}(G)} \Psi_C(\vec{x}_C)$$



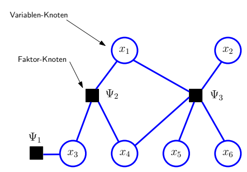
- Typische Formulierung von Potentialfunktionen: $\Psi_i(\vec{x}) = \exp(\sum_j \lambda_j f_j(\vec{x}))$

(Koller/Friedman 2010)

Faktor-Graph

Ein *Faktor-Graph* ist ein bipartiter Graph über Faktoren und Variablen

- Faktor Ψ_i berechnet ein Skalar über alle Variablen
- Seien $\vec{x}$ Zufallsvariablen
- $\Psi_i(\vec{x}_i) = \exp\left(\sum_k \lambda_{ki} f_{ki}(\vec{x}_i)\right)$
- Wahrscheinlichkeitsverteilung: $p(\vec{x}) = \frac{1}{Z} \prod_i \Psi_i(\vec{x}_i)$

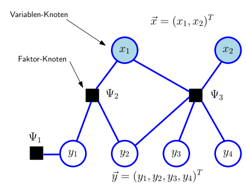


(Bishop, 2006; Kschischang, 2001)

Conditional Random Field als Faktorgraph

Conditional Random Field: Markovnetzwerk mit zusätzlichen, immer bekannten Parametern.

- Faktor Ψ_i berechnet ein Skalar
- Seien $\vec{x}$ Eingabe- und $\vec{y}$ Ausgabevariablen
- $\Psi_i(\vec{x}_i, \vec{y}_i) = \exp\left(\sum_k \lambda_{ki} f_{ki}(\vec{x}_i, \vec{y}_i)\right)$
- Wahrscheinlichkeitsverteilung: $p(\vec{y} | \vec{x}) = \frac{1}{Z(\vec{x})} \prod_i \Psi_i(\vec{x}_i, \vec{y}_i)$



Lernen und Inferenz

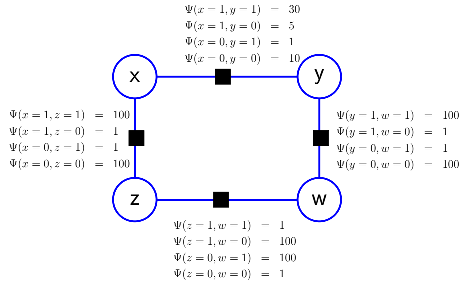
Lernen

- Gerichtete Modelle: Bestimmung der bedingten Wahrscheinlichkeiten durch Abzählen in Trainingsdaten $\mathcal{D}$
- Ungerichtete Modelle: Iterative Maximierung von $\log \sum_{\vec{x} \in \mathcal{D}} p_{\vec{\lambda}}(\vec{x})$

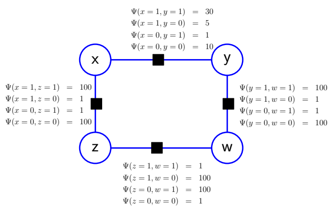
Inferenzaufgaben

- Berechnen von Wahrscheinlichkeitsverteilungen einzelner Teilmengen von Knoten: Sum-Product-Algorithmus (später, Spezialfall: Forward-Backward)
- Berechnen der Belegung, welche die Gesamtwahrscheinlichkeit maximiert: Max-Product-Algorithmus (Spezialfall: Viterbi, kleine Variation zu Sum-Product)
- $\Rightarrow$ Belief-Propagation

Beispielfaktorgraph



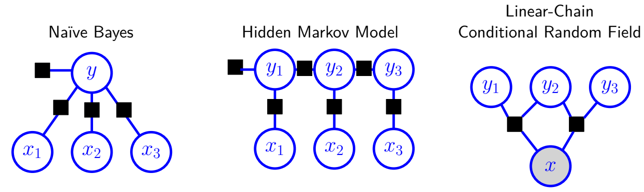
Beispielfaktorgraph



Werte

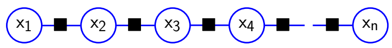
Variablenbelegung				Wert	Wahrscheinlichkeit
x	y	z	w		
0	0	0	0	100000	0.0159445
0	0	0	1	100000	0.0159445
0	0	1	0	100000	0.0159445
0	0	1	1	10	0.0000016
0	1	0	0	10000	0.0015945
0	1	0	1	10000	0.0015945
0	1	1	0	100	0.0000159
0	1	1	1	100	0.0000159
1	0	0	0	500	0.0000797
1	0	0	0	500	0.0000797
1	0	0	1	500	0.0000797
1	0	1	0	5000000	0.7972269
1	0	1	1	50000	0.0079723
1	1	0	0	30	0.0000048
1	1	0	1	300000	0.0478336
1	1	1	0	300000	0.0478336
1	1	1	1	300000	0.0478336

Beispiel-Faktorgraphen



(Laffery, 2001)

Inferenz auf einer Kette (I)



- $p(\vec{x}) = \frac{1}{Z} \Psi_{1,2}(x_1, x_2) \Psi_{2,3}(x_2, x_3) \cdots \Psi_{n-1,n}(x_{n-1}, x_n)$
- Annahme: Variablen können k diskrete Werte annehmen
- Jede Potentialfunktion hat also k^2 Parameter
- Anzahl aller Parameter für die gemeinsame Verteilung entsprechend der Faktoren: $(n-1)k^2$
- Inferenz der Randverteilung durch Aussummieren von x_i:

$$p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} p(\vec{x})$$

- k^n Werte für $\vec{x}$ – schade.

Inferenz auf einer Kette (II)

- $p(\vec{x}) = \frac{1}{Z} \Psi_{1,2}(x_1, x_2) \Psi_{2,3}(x_2, x_3) \cdots \Psi_{n-1,n}(x_{n-1}, x_n)$
- $p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} p(\vec{x})$

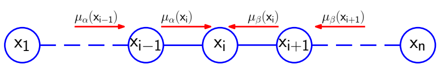
Idee: Umsortieren dieser Berechnung:

$$p(x_i) = \frac{1}{Z} \cdot \left[\sum_{x_{i-1}} \Psi_{i-1,i}(x_{i-1}, x_i) \cdots \left[\sum_{x_2} \Psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \Psi_{1,2}(x_1, x_2) \right] \right] \cdots \right] \cdot \left[\sum_{x_{i+1}} \Psi_{i,i+1}(x_i, x_{i+1}) \cdots \left[\sum_{x_n} \Psi_{n-1,n}(x_{n-1}, x_n) \right] \cdots \right]$$

$\Rightarrow O(nk^2)$ Berechnungen, wie wir gleich sehen werden.

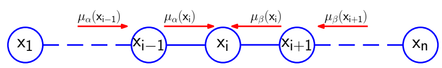
Inferenz auf einer Kette (III)

$$p(x_i) = \frac{1}{Z} \cdot \underbrace{\left[\sum_{x_{i-1}} \Psi_{i-1,i}(x_{i-1}, x_i) \cdots \left[\sum_{x_2} \Psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \Psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_i)} \cdot \underbrace{\left[\sum_{x_{i+1}} \Psi_{i,i+1}(x_i, x_{i+1}) \cdots \left[\sum_{x_n} \Psi_{n-1,n}(x_{n-1}, x_n) \right] \cdots \right]}_{\mu_\beta(x_i)}$$



- Jede Nachricht ist eine Menge von k Werten
- Eine Wahrscheinlichkeit für jeden Wert

Inferenz auf einer Kette (IV)



Rekursive Evaluation von Nachrichten:

- $\mu_\alpha(x_i) = \sum_{x_{i-1}} \Psi_{i-1,i}(x_{i-1}, x_i) \left[\sum_{x_{i-2}} \cdots \right] = \sum_{x_{i-1}} \Psi_{i-1,i}(x_{i-1}, x_i) \mu_\alpha(x_{i-1})$
- Erste Evaluation: $\mu_\alpha(x_2) = \sum_{x_1} \Psi_{1,2}(x_1, x_2)$
- $\mu_\beta(x_i) = \sum_{x_{i+1}} \Psi_{i+1,i}(x_{i+1}, x_i) \left[\sum_{x_{i+2}} \cdots \right] = \sum_{x_{i+1}} \Psi_{i+1,i}(x_{i+1}, x_i) \mu_\beta(x_{i+1})$

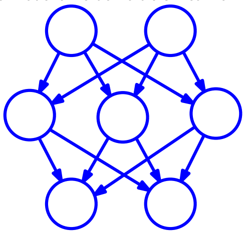
Stand der Dinge

- Probabilistische Graphische Modelle stellen Unabhängigkeitsannahmen über Wahrscheinlichkeitsverteilungen dar
- Inferenz ist effizient, wenn Graph ein Baum ist
- Bei beliebigen Graphen ist Inferenz NP schwer: Approximationsalgorithmen und Samplingverfahren existieren

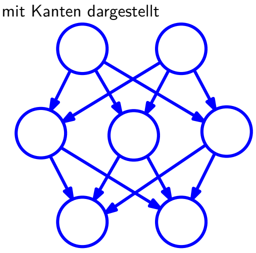
Outline

Probabilistische Graphische Modelle und Neuronale Netze

Beide Modelle werden oft als Netzwerk von Knoten mit Kanten dargestellt

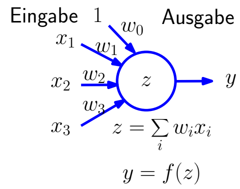


Directed Probabilistic Graphical Model



Feed-forward Neural Network

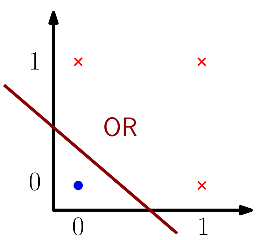
Ein sehr einfaches künstliches neuronales “Netzwerk”

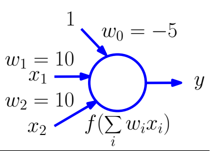


$f(\sum_i w_i x_i) = y$

- $z = \sum_i w_i x_i$: interner Zustand
- $f(\cdot)$ Aktivierungsfunktion (z.B. linear, tanh, sigmoid, ReLU)

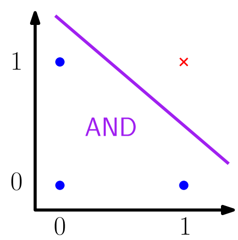
ODER mit einem Neuron

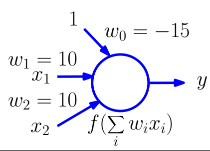




x_1	x_2	$\sum w_i x_i$	Sigmoid
0	0	−5	0.007
0	1	5	0.993
1	0	5	0.993
1	1	15	0.999

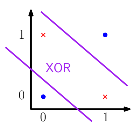
UND mit einem Neuron

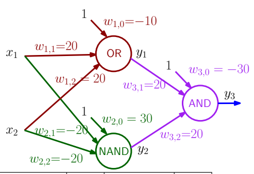




x_1	x_2	$\sum w_i x_i$	Sigmoid
0	0	−15	0.000
0	1	−5	0.007
1	0	−5	0.007
1	1	5	0.993

ENTWEDER-ODER als Netzwerk von UND, NUND, und ODER

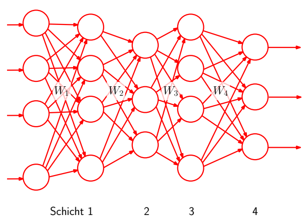




x_1	x_2	$\sum x_i w_{1,i}$	y_1	$\sum x_i w_{2,i}$	y_2	$\sum x_i w_{3,i}$	y_3
0	0	−10	≈ 0	30	≈ 1	≈ -10	≈ 0
0	1	10	≈ 1	10	≈ 1	≈ 10	≈ 1
1	0	10	≈ 1	10	≈ 1	≈ 10	≈ 1
1	1	30	≈ 1	−10	≈ 0	≈ -10	≈ 0

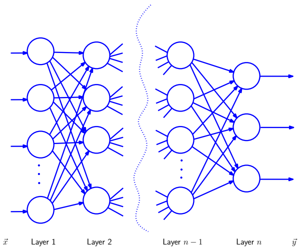
- $\mathrm{XOR} = \mathrm{AND}(\mathrm{OR}(x_1, x_2), \mathrm{NAND}(x_1, x_2))$

Semantik von vorwärtsgerichteten Neuronalen Netzen



- Feed-forward Neuronal Network stellt eine Funktionskomposition dar
- $\vec{y} = \vec{f}_4(\vec{f}_3(\vec{f}_2(\vec{f}_1(\vec{x}, W_1), W_2), W_3), W_4)$
- Vorwärtspropagierung durch das Netz: Ebenenweise Berechnung der jeweiligen Ausgabewerte

Inferenz und Lernen

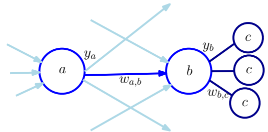


- Inferenz: Forward Propagation, berechne Ergebnisse Ebene für Ebene
- Lernen: Gewichtsanpassung durch Minimierung eines Fehlers der Vorhersage auf Trainingsdaten:
  $$E(\mathbf{W}, \mathcal{D}) = \sum_{\vec{x}, \vec{y}^* \in \mathcal{D}} \frac{1}{2} \|\vec{y}(\vec{x}, \mathbf{W}) - \vec{y}^*\|^2$$
- Minimierung durch Gradientabstieg, Berechnung des Gradienten: Backpropagation

(Bishop 2006)

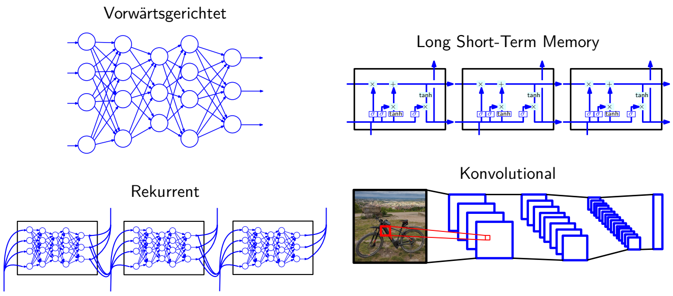
Backpropagation

- $w^t_{a,b} = w^{t-1}_{a,b} + \alpha \cdot \delta_b \cdot y_a$
  - $w_{a,b}$: Gewicht von Neuron a zu Neuron b
  - a: Neuron in Schicht i
  - b: Neuron in Schicht $i+1$
  - y_a: Ausgabe von Neuron a



- Lokale Gradienten δ_b:
  - $\delta_b = \begin{cases} (y^*_b - y_b) \cdot f'(z_b) & \text{wenn } b \text{ in Ebene } n \\ \left(\sum_{c \in \mathrm{Succ}(b)} (\delta_c \cdot w_{b,c})\right) \cdot f'(z_b) & \text{sonst.} \end{cases}$
  - z_b: interner Zustand von b
  - Rekursive Berechnung

Beispiele für Architekturen Neuronaler Netze



Stand der Dinge

- KNN stellen Funktionskompositionen dar
- Berechnung der Ausgabe: Vorwärtspropagierung
- Lernen: Gradientenabstieg, Backpropagation
- Komplexe Zusammenhänge durch latente Variablen lernbar

Outline

Vergleich der Propagierungsalgorithmen

	Modell-Paradigma	
Aufgabe	PGM	FF-KNN
Inferenz	Belief Propagation (Max-Product, Max-Sum)	Vorwärtspropagierung
Lernen	Zählen, Gradientenaufstieg (Normalisierungsberechnung: Max-Product)	Gradientenabstieg (Gradientenbestimmung: Backpropagation)

Ist Belief Propagation das Gleiche wie Backpropagation?

Nein.

Unterschiede:

- Belief Propagation:
  Inferenz entsprechend Struktur des graphischen Modells
- Backpropagation:
  Berechnung von Gradienten entsprechend eines Fehlers in der Ausgabe

Aber:

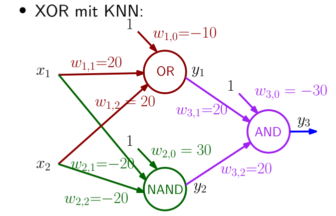
- Es existieren Arbeiten, die die Verfahren aufeinander abbilden.
  (Eisner, 2016; Dauwels, 2005; Dauwels, 2006)
- Diese Arbeiten führen nicht zu einer Gleichstellung von PGMs und KNNs.

Outline

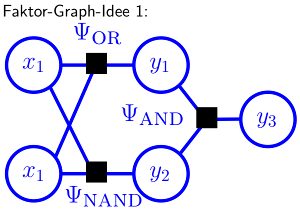
1. Einführung und Motivation
2. Methodischer Überblick
   - Probabilistische Graphische Modelle (PGM)
   - Künstliche Neuronale Netze (KNN)
3. Verhältnis von PGM und KNN
   - Propagierungsalgorithmen
   - Verwandtschaft der Formulierungen
   - Hidden CRF
4. Integration von KNN in PGM
   - Fallstudien
   - Werkzeuge
5. Zusammenfassung und Ausblick

Sind KNN und PGM verwandt? Beispiel: XOR

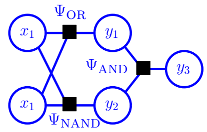
- XOR mit KNN:



- Faktor-Graph-Idee 1:



Sind KNN und PGM verwandt? Beispiel: XOR

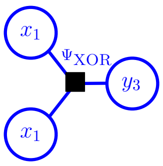


- $\Psi_i(\cdot) = \exp(\sum_j \lambda_{ij} f_{ij}(\cdot))$
  - $f_{i1}(x_a, x_b) = \{1$, wenn $x_a = 1 \wedge x_b = 1 \wedge y = 1; 0$ sonst$\}$
  - $f_{i2}(x_a, x_b) = \{1$, wenn $x_a = 1 \wedge x_b = 0 \wedge y = 1; 0$ sonst$\}$
  - $f_{i3}(x_a, x_b) = \{1$, wenn $x_a = 0 \wedge x_b = 1 \wedge y = 1; 0$ sonst$\}$
  - $f_{i4}(x_a, x_b) = \{1$, wenn $x_a = 0 \wedge x_b = 0 \wedge y = 1; 0$ sonst$\}$
- Ψ_{OR}: $\lambda_{i1} = 1,\ \lambda_{i2} = 1,\ \lambda_{i3} = 1,\ \lambda_{i4} = 0$
- Ψ_{NAND}: $\lambda_{i1} = 0,\ \lambda_{i2} = 1,\ \lambda_{i3} = 1,\ \lambda_{i4} = 0$
- Ψ_{AND}: $\lambda_{i1} = 1,\ \lambda_{i2} = 0,\ \lambda_{i3} = 0,\ \lambda_{i4} = 0$

- Vergleichbare Darstellung möglich, aber Inferenz und Lernen dennoch vollkommen unterschiedlich.

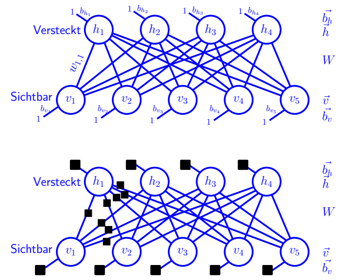
Sind KNN und PGM verwandt? Beispiel: XOR

Idee 2:



- $\Psi_{\text{XOR}}(\cdot) = \exp(\sum_j \lambda_j f_j(\cdot))$
  - $f_{i1}(x_a, x_b) = \{1$, wenn $x_a = 1 \wedge x_b = 1 \wedge y = 1; 0$ sonst$\}$
  - $f_{i2}(x_a, x_b) = \{1$, wenn $x_a = 1 \wedge x_b = 0 \wedge y = 1; 0$ sonst$\}$
  - $f_{i3}(x_a, x_b) = \{1$, wenn $x_a = 0 \wedge x_b = 1 \wedge y = 1; 0$ sonst$\}$
  - $f_{i4}(x_a, x_b) = \{1$, wenn $x_a = 0 \wedge x_b = 0 \wedge y = 1; 0$ sonst$\}$
- $\lambda_1 = 0,\ \lambda_2 = 1,\ \lambda_3 = 1,\ \lambda_4 = 0$

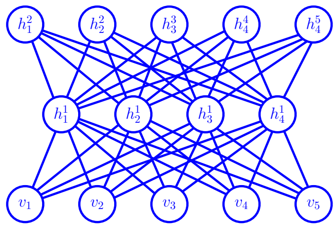
Restricted Boltzmann-Machine



- Lernt Zusammenhänge zwischen sichtbaren Variablen $\vec{v}$ mit Hilfe der latenten Variablen $\vec{h}$
- $p(\vec{v}, \vec{h}) = \frac{1}{Z} \cdot \exp\left(-E(\vec{v}, \vec{h})\right)$
- $E(v, \vec{h}) = -\vec{b}_v^\intercal \vec{v} - \vec{b}_h^\intercal \vec{h} - \vec{v}^\intercal W \vec{h}$
- $p(\vec{h} \mid \vec{v}) = \prod_i p(h_i \mid \vec{v})$ und $p(\vec{v} \mid \vec{h}) = \prod_i p(v_i \mid \vec{h})$
- Inferenz/Training mit Block-Gibbs-Sampling
- Baustein zur Erstellung tiefer Netzwerke

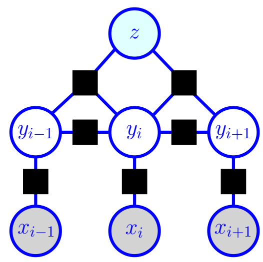
(Goodfellow, 2016)

Beispiele eines Deep Belief Networks



⇒ PGM
⇒ Struktur entsprechend tiefem neuronalen Netz
⇒ Semantik versteckter Variablen nicht a priori definiert

Probabilistisches Modell mit latenten Variablen: Hidden CRF



- Eingabevariablen x (immer beobachtbar)
- Ausgabevariablen z (zum Trainingszeitpunkt beobachtbar)
- Ausgabevariablen y (niemals beobachtbar)

⇒ Vergleichbare Situation zu Feed-Forward Neural Networks
  - Semantik der latenten Variablen ist aber vordefiniert.
- Training: Gradientaufstieg (aber nicht konvex) oder Expectation Maximization

(Quattoni, 2007; Tackstrom, 2011)

Stand der Dinge

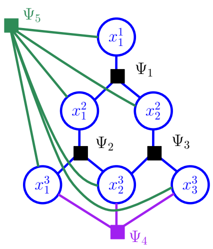
- PGMs und KNNs sind nicht das Gleiche, haben aber ähnliche Komponenten
- KNNs nutzen latente Variablen um komplexe Zusammenhänge zu lernen
  - PGMs können das ebenfalls: Hidden CRF
- PGMs sind Formalismen um Annahmen über Variablenzusammenhänge zu modellieren
  - Eher unüblich in KNNs
- Gewichte in (manchen) neuronalen Netzen können als Faktoren in PGMs gesehen werden: Repräsentationslernen mit Hilfe von RBM

Outline

Einbindung von KNN-Faktoren in PGMs?

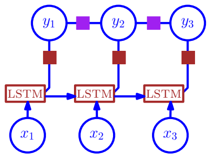
Idee mit Beispiel:

- Modellierung der Zusammenhänge entsprechend Expertenwissen für Variablen mit bekannter Semantik
- Einbindung von neuronalen Faktoren für automatisches Lernen komplexer Zusammenhänge
- $\Psi_1(x_1^1, x_1^2, x_2^2) = \exp(\sum_i(\lambda_i f(x_1^1, x_1^2, x_2^2)))$
- $\Psi_4(x_1^3, x_2^3, x_3^3) = \mathrm{RNN}(\cdot)$
- $\Psi_5(\vec{x}) = \mathrm{FFNN}(\cdot)$
- $p(\vec{x}) = \frac{1}{Z} \cdot \prod_i \Psi_i(\cdot)$
  - Welche Realisierungen gibt es von solchen Modellen?
  - Ich stelle nun einige Fallstudien vor.



LSTM-CRF

Idee: Modelliere Eigenschaften der Ausgabevariablen mit einem PGM, nutze flexibles RNN um Dateneigenschaften zu lernen.



- Standard linear-chain CRF:
  $p(\vec{y} \mid \vec{x}) = \frac{1}{Z(\vec{x})} \prod_i \exp(\sum_j \lambda_j f_j(y_{i-1}, y_i, \vec{x}, i))$
- Andere Formulierung:
  $p(\vec{y} \mid \vec{x}) =$
  $\frac{1}{Z(\vec{x})} \cdot \prod_i \exp(\sum_j \lambda_j f(y_{i-1}, y_i)) \cdot \prod_i \exp(\sum_j \lambda_j f(\vec{x}, y_i, i))$
- Ersetze datenbezogenes Log-lineares Modell durch LSTM:
  $p(\vec{y} \mid \vec{x}) = \frac{1}{Z(\vec{x})} \cdot \prod_i \exp(\sum_j \lambda_j f(y_{i-1}, y_i)) \cdot \prod_i \mathrm{LSTM}(\vec{x}_i, y_i)$

Dies ist ein etabliertes Modell.

LSTM-CRF Anwendungsbeispiele (1)

Labeling von Wörtern in Text mit einer IOBE-Sequenz (Huang et al., 2015)

$\vec{x}$ =	the	Severe	acute	respiratory	syndrome	coronavirus	2	...
$\vec{y}$ =	O	B	I	I	I	I	E	...

CRF-Schicht zeigt Fehlerreduktion von etwa 10 %

LSTM-CRF Anwendungsbeispiele (2)

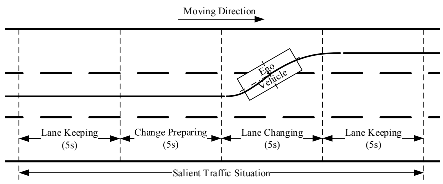
2D-Variante: Bildsegmentierung (Zheng et al., 2015)

Original image (hover to highlight segmented parts) — Semantic segmentation

Verbesserungen insbesondere bei filigranen Strukturen
(http://www.robots.ox.ac.uk/~szheng/crfasrnndemo/)

LSTM-CRF Anwendungsbeispiele (3)

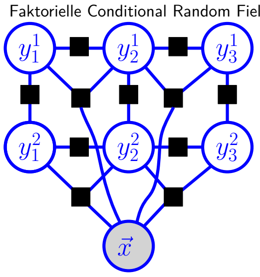
Manoeverentscheidungen bei autonomem Fahren (Wang et al., 2018)



	Lane Keeping		Change prep		Left change		Right change		
	P	R	P	R	P	R	P	R	F_1
LSTM	.99	.80	.39	.97	.67	.97	.61	.97	.75
LSTM+CRF	.99	.78	.42	.99	.80	.98	.79	.99	.81

Mehrere in Relation stehende Sequenzen

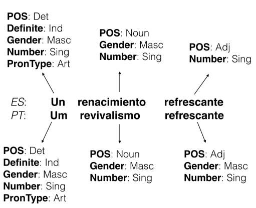
Faktorielle Conditional Random Fields (Sutton, 2007)

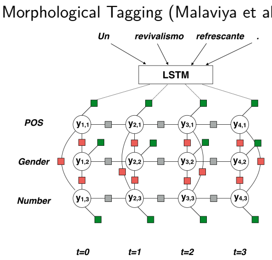


- Entspricht Kombination zweier linearer Ketten, welche mit einer Eingabe konditioniert werden
- Aber: Vorhersage ist nicht unabhängig, daher messen Faktoren Kompatibilität zwischen Ausgabesequenzen
- Loopy Graph!

Mehrere in Relation stehende Sequenzen: Morphologisches Tagging

Neural Factor Graph Models for Cross-lingual Morphological Tagging (Malaviya et al., 2018)





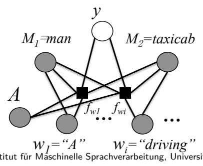
Mehrere in Relation stehende Sequenzen: Morphologisches Tagging, Ergebnisse

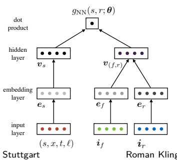
Language	Model	*tgt_size* = 100 Accuracy	F1-Micro	F1-Macro	*tgt_size*=1000 Accuracy	F1-Macro	F1-Micro
SV	Baseline	15.11	8.36	10.37	68.64	76.36	76.50
	Ours	**29.47**	**54.09**	**54.36**	**71.32**	**84.42**	**84.46**
BG	Baseline	**29.05**	14.32	29.62	**59.20**	**67.22**	**67.12**
	Ours	27.81	**40.97**	**42.43**	39.25	60.23	60.84
HU	Baseline	21.97	13.30	16.67	**50.75**	58.68	62.79
	Ours	**33.32**	**54.88**	**54.69**	45.90	**74.05**	**73.38**
PT	Baseline	18.91	7.10	10.33	74.22	81.62	81.87
	Ours	**58.82**	**73.67**	**74.07**	**76.26**	**87.13**	**87.22**

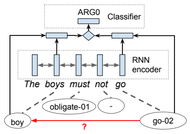
Table 3: Token-wise accuracy and F1 scores on mono-lingual experiments

Relationserkennung in Text

- Relationserkennung: Integration von neuronalen Netzen mit Merkmalen (Gormley, 2015)
- SRL: Prädikate und Rollen werden durch neuronale Faktoren bewertet (FitzGerald, 2015)
- AMR Parsing: Integration von Konzepterkennung und Relationserkennung in gemeinsamem Model (Lyu, 2018)







Outline

Werkzeuge

- Lineare CRF-Schicht, z.B. PyTorch oder AllenNLP
- Torch Struct (Rush, 2020; best demo-paper award vor einer Woche auf der ACL 2020)
  - Vorimplementierte Faktorgraphen, u.a. Bäume, Sequenzen, Semi-Markov-Modelle, Alignierung
- LP-SparseMAP (Nicolae, 2020; publiziert diese Woche auf ICML 2020)
  - Beliebige Faktorgraphen möglich, Beispiele gezeigt für Multilabelklassifikation, Ketten und Bäume sowie weitere Nebenbedingungen zwischen Variablen
- SPEN (Belanger/McCallum, 2016)
  - Beliebige Faktorgraphen möglich, insbesondere wird die Struktur hier aber gemeinsam mit den Parametern in dem neuronalen Netz gelernt

Outline

Sind KNNs und PGMs also wirklich unterschiedlich?

Deep Learning

Deep learning models typically have more latent variables than observed variables. Complicated nonlinear interactions between variables are accomplished via indirect connections that flow through multiple latent variables.

Graphische Modelle

[...] traditional graphical models usually contain mostly variables that are at least occasionally observed [...] [and] use higher-order terms and structure learning to capture complicated nonlinear interactions between variables. If there are latent variables, they are usually few in number.

Variablen

The deep learning practitioner typically does not intend for the latent variables to take on any specific semantics ahead of time—the training algorithm is free to invent the concepts it needs to model a particular dataset.

(Goodfellow/Bengio/Courville: Deep Learning, p. 575)

Mitnachhausenehmennachricht

- PGMs und KNN sind stark verwandt, haben aber in der praktischen Verwendung Unterschiede
- PGM:
  - Vorteile: Gut geeignet um Weltwissen zu modellieren
  - Nachteile: Mit latenten Variablen nicht effizient (exakt) zu trainieren
- KNN:
  - Vorteile: Lernverfahren entscheidet datengetrieben über Bedeutung von Variablen, Parameterschätzung durch Backpropagation effizient
  - Nachteile: Unklar zu interpretieren
- Was fehlt?
  - Kombinationsstrategien, um Modellierung von Weltwissen mit neuronalem Ende-zu-Ende-Lernen zu kombinieren
  - Aber: Einzelne Beispiele existieren

Diskussion

- Software zur Nutzung bestimmer PGM-Strukturen existiert
- PGMs mit flexiblen Strukturen können mit Hilfe von verschiedenen Sprachen definiert werden:
  - Deklarative Faktorgraphen: FACTORIE (McCallum, 2010)
  - Markov-Logik: Alchemy (Richardson, Domingos, 2006) und Markov thebeast (Riedel, 2008)
  - Infer.Net/CSoft von Microsoft

  ⇒ Keiner dieser Ansätze hat auch nur annähernd den Erfolg von KNN
- Software zur Nutzung bestimmer PGM-Strukturen mit KNN existiert
  - Torch-Struct, LP-SparseMAP, CRF-Layer in DL-Bibliotheken...

Stand der Dinge

Software zur Kombination von beliebigen PGMs mit KNN-Faktoren ist limitiert.

Wieso gibt es das nicht? (Hypothesen)

- Definition von Faktorgraphen bedarf komplexer Sprachen, welche schwierig zu erlernen sind (insbesondere bei repetitiven Strukturen und Parameter-Binding).
- Die manuelle Modellierung von Weltwissen ist nicht notwendig.
- PGMs mögen das Problem akkurater abbilden als einfache Klassifikationsprobleme zu nutzen. Aber die Modellierung ist komplizierter, als das Problem zu vereinfachen.
- ...

Diskussion

Jason Eisner
@adveisner

A neural component in your model is like a draw from a Gaussian. It marks where you stopped modeling your specific problem ... #ML #nlproc

Tweet übersetzen

8:17 nachm. · 6. Aug. 2017 aus Baltimore, MD · Twitter Web Client

27 Retweets **86** „Gefällt mir"-Angaben

https://twitter.com/adveisner/status/894261230285803520

Danke für die Aufmerksamkeit.
Gibt es Fragen?

?

Danke,

Agnieszka Faleńska, **Albert Gatt**, Alex Klenner, Alexander Bernhardt, Alexandra Balahur, Alin Roman, Amelie Heindl, Amelie Wührl, **Anders Björkelund**, Anton Bakalov, Andreas Stöckel, Andreas Vlachos, **Andrew McCallum**, Angelika Weihermüller, Anna Moskvina, Antje Rossdeutscher, Antje Schweitzer, **Antje Wolf**, Aravind Tallam, Arif Ijaz, Axel Pichler, Aysoltan Gravina, Benjamin Paassen, Bernd Müller, Burak Tekin, **Camilo Thorne**, Carina Haupt, **Christian Ebeling**, **Christian Scheible**, Christian Witte, Christina Unger, **Christoph M. Friedrich**, Constantin Seibold, Cord Wiljes, **Corinna Klein geb. Kolarik**, David Helbig, **David Mimno**, Deniz Cevher, Edgar Hoch, Ekta Sood, Elias Zaied, Elnaz Shafaei Bajestan, **Engelbert Thun**, **Enrica Troiano**, Erenay Dayanik, Erfan Younesi, **Evgeny Kim**, Felix Casel, Florian Strohm, Fotis Aisopos, Frank Grimm, Franziska Schmidtke, Frederike Strunz, Gabriel Viehhauser, **Gabriella Lapesa**, Gerhard Kremer, Gianluigi Marra, **Greg Druck**, **Günter Rudolph**, Hai Dang Nguyen, Hanna Kicherer, Hanno Ehrlicher, Hans Kamp, **Hans Werner Müller**, Hanna Wallach, Harsha Gurulingappa, Heike Adel, Helen Vernon, Hendrik Schuff, Hendrik ter Horst, **Hildegard Klinger**, Horst Schwichtenberg, Iman Zeinali Nia, **Ingo Wegener**, Jan Hofmann, Jan Philipp Göpfert, Jan Wessling, **Janik Jaskolski**, Jasmin Zohren, Jennifer Ling, Jennifer Majunke, **Jeremy Barnes**, Johannes Butscher, Johannes Erwerle, Johannes Schäfer, John McCrae, John Wilbur, **Jonas Kuhn**, Jonathan Sonntag, Josef Ruppenhofer, Judith Gaspers, Julia Christoph, Julia Maria Struß, Julian Liedtke, **Juliane Fluck**, Kai Kumpf, **Kai Sassenberg**, Karl Kirschner, **Katharina Morik**, Katja Temnow, **Katrin Tomanek**, **Kenneth Kahl**, Kiril Atanassov, Konstantin Buschmeier, Larry Smith, Lars Buttgereit, Lars Hildebrand, Lars Vogel, **Laura Bostan**, Laura I. Furlong, Laura Dietz, Lawrence Hunter, **Lonneke van der Plas**, Luci Fillinger, Lujan Miquel, Lukas Grebe, Marc Jacobs geb. Zimmermann, Marcel Dittrich, **Marco Hülsmann**, Marietta Hamberger, Mariia Kashpur, Mario Sänger, Martin Hofmann-Apitius, Matthias Hartung, Maximilian Köper, Maximilian Panzner, Meike Knieps, Michael Bittner, Michael Griebel, Michael Krapp, Michael Wiegand, **Michael Wick**, **Michal Jacovi**, Min Xu, Mohamad Alshaer, Naveen Kumar, **Nicole Brazda**, **Nils Reiter**, Nyamsuren Davaasambuu, Oliver Beyer, Oliver Wäldrich, Omar Abada, Orphee De Clercq, Patrizia Paggio, Peter Menke, **Philipp Cimiano**, **Philipp Senger**, **Philippe Thomas**, Raphael Dickfelder, **Robert McHardy**, Robert Pesch, Robin Schiewer, Roozbeh Bandpey, Ruth Sander, **Sabine Dieterle**, Sabine Mohr, **Sabine Schulte im Walde**, **Saif Mohammad**, Sandra Murr, Sameer Singh, Sarah Schulz, **Sean Papay**, **Sebastian Pado**, **Sebastian Riedel**, Sebastian Walter, Sebastian Zepf, Serbay Ekinoglu, Sherzod Hakimov, Shu Xing, Shweta Bagewadi, **Simone Spicks**, **Simone Wessler**, Sina Brandstetter, Soufian Jebbara, **Stefan Edelkamp**, Stefanie Anstein, Steffen Eger, **Sumit Madan**, Surayya Samat Suyliya, **Tamara Bobic**, Tarek Kirchhoffer, **Theo Mevissen**, Thomas Haider, **Thomas Klinger**, Thomas Tibroni, **Tim Rocktäschel**, Timo Reuter, **Torsten Zesch**, **Ulf Leser**, Ulrich Heid, **Ulrich Trottenberg**, **Ursula Thun**, Uwe Reyle, Valentino Sabbatino, Verena Meyer, Veronica Estrada, Vivi Nastase, **Wiebke Klinger**, Wiltrud Kessler, Winfried Menninghaus, Wolfgang Ziegler, Xiang Yu, Yan Yang, Zhanruo Qu, Özlem Cetinoglu.

Universität Stuttgart
Institut für
Maschinelle Sprachverarbeitung

Probabilistische Graphische Modelle in einer Neuronalen Welt

Wie können zwei Paradigmen vereint werden?

Mündliche Habilitationsleistung

17. Juli 2020

Roman Klinger
roman.klinger@ims.uni-stuttgart.de

@roman_klinger romanklinger
http://www.romanklinger.de/