# Dynamic Markov Logic Networks as a knowledge base for cognitive technical systems

Thomas Geier

Universität Ulm
Institut für Künstliche Intelligenz

17. Oktober 2011

# Overview

Motivation

Markov Logic Networks

Inference in Markov Logic Networks

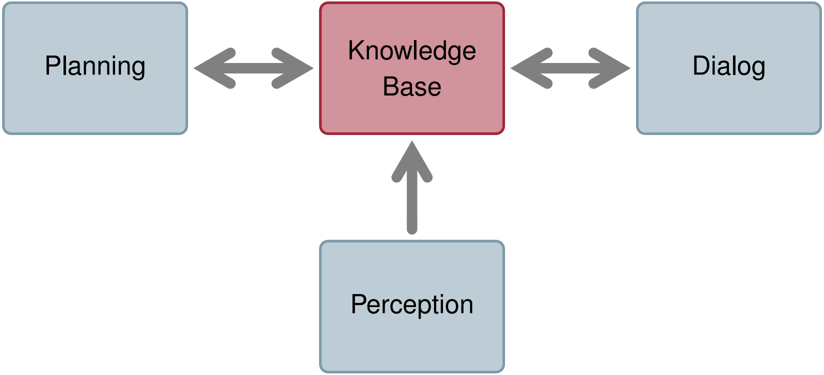
Dynamic Markov Logic Networks

# What is the purpose of a knowledge base?

Representing the knowledge of the system!

- the single purpose of knowledge (for a technical system) is to form a basis for selecting an action
- action is used in the most general form, representing anything that escapes the technical system by the means of an actuator
- knowledge that does not influence the behavior is redundant — it can be removed without any effect to the outside

# Motivation



# How must the knowledge of a cognitive technical system look like?

1. It is uncertain.

- the system is interacting with a natural environment
- parts of the environment are not observable
  (e.g. the emotional state of user)
- input can be imperfect
  (failing sensors, poor classifiers, wrong user input)

# How must the knowledge of a cognitive technical system look like?

1. It is uncertain.
2. It is symbolic.

- knowledge must be accessible / symbolic
- thus, its meaning can be expressed in natural language
- e.g., you cannot simply connect a neural network to the GUI — you must do this through a label giving a meaningful interpretation to the output

# How must the knowledge of a cognitive technical system look like?

1. It is uncertain.
2. It is symbolic.
3. It is dynamic.

- the state of the environment changes
- thus the knowledge has to be “updated”
- a future state strongly depends on a past state

## How must the knowledge of a cognitive technical system look like?

1. It is uncertain.
2. It is symbolic.
3. It is dynamic.
4. It is flexible.

- the model must be flexible and easy to extend
- it should not be fixed to a certain task

# How can we fulfill these criteria?

1. It is uncertain.
   employ a probabilistic model

# How can we fulfill these criteria?

1. It is uncertain.
   employ a probabilistic model
2. It is symbolic.
   the state is factored into symbolic variables

# How can we fulfill these criteria?

1. It is uncertain.
   employ a probabilistic model
2. It is symbolic.
   the state is factored into symbolic variables
3. It is dynamic.
   time is represented explicitly

# How can we fulfill these criteria?

1. It is uncertain.
   employ a probabilistic model
2. It is symbolic.
   the state is factored into symbolic variables
3. It is dynamic.
   time is represented explicitly
4. It is flexible.
   state variables are predicates over objects

# Probabilistic models

- the possible states of the world form a probability space $\Omega$
- a probability (mass) function $P : \Omega \to [0, 1]$ represents the belief of the system about what the true state might be
- possible world states assigned a higher probability are more likely the true state

# Learning new facts means making observations

- observations are used to condition the distribution

$$P'(x) = P(x \mid e) = \frac{P(x, e)}{P(e)}$$

- observations change the belief of the system
  (they hopefully improve it, making the true state more probable)

# How to handle state?

- instead of having a very large set of states $\Omega = \{s_1, s_2, \dots\}$
- we can express the state by multiple random variables.
  Informally $\Omega = X_1 \times X_2 \times X_3 \times \cdots \times X_n$
  (obtaining a multivariate distribution)
- and assign meaning (labels) to those variables:
  $X_1 = \mathit{true}$ means "the user is at her desk",
  $X_2 = \mathit{true}$ means "the user is tired"

# How to handle time?

- conditioning a probabilistic model on a fact
  with probability zero is not possible
- contradictory observations can be made when time progresses
  - the user is at her desk
  - the user is away
- unless time is represented explicitly
  - the user is at her desk at 4pm
  - the user is away at 5pm

# Relational models

- a relational model contains objects
  `14:32`, `14:45`, `u#1`, `iPhone`, `iPod`
- and relations between them
  `touches(Time,User,Device)`
- inserting objects into relations forms atoms (or propositions)
  `touches(14:32,u#1,iPhone)`, `touches(14:45,u#1,iPod)`
- dependencies can abstract over objects using variables
  `touches(t,user,device) => near(t,user,device)`
- thus relational models can scale with the number of objects
- time has not to be built-in but can be expressed freely using time-dependent predicates

# Markov Logic Networks

- Markov Logic Networks [1] (MLNs) are a probabilistic, relational model
- dependencies are expressed through first-order logical formulas
- as such they fulfill all presented requirements

[1] Matthew Richardson and Pedro Domingos. "Markov logic networks". In: *Machine Learning* 62.1-2 (2006), pp. 107–136. ISSN: 0885-6125. DOI: 10.1007/s10994-006-5833-1

# An example — users and devices

An informal description of a model about interface devices and users.

- if a user touches a device, she is near the device
- if a user is near a device, she can probably see it
- if a user is near a device, she will probably still be near the device later

# An example — Markov Logic Networks

The same model formalized as a Markov Logic Network.

```
touches(t,user,device) => near(t,user,device).
1.0 near(t,user,device) => sees(t,user,device)
4.0 near(t,user,device) => near(t+1,user,device)
```

---

# Definition: Syntax of Markov Logic Networks

- a Markov Logic Network $L$ is a set of weighted first-order logical formulas

$$L = \{(f_i, w_i) \mid i \leqslant n, f_i \in \text{FOL}, w_i \in \mathbb{R}\}$$

- the formulas can use predicates from sorted logical language and constants from a given constant domain

# Sytnax of MLNs — logical language example

- the logical language defines which propositions can be expressed

```
Predicate near(Time,Person,Device)
Predicate touches(Time,Person,Device)
Predicate sees(Time,Person,Device)
```

---

- objects can be categorized into sorts or types
  (Time, Person, Device)

# MLNs require a finite set of constants

- in addition to the weighted formulas one need to give a finite set of constants for each sort
- using as constants
  `Time = {0,1}, Person = {u#1}, Device = {iPod,iPad}`
  one obtains the atoms for `near(Time,Person,Device)`:
  `near(0,u#1,iPod), near(0,u#1,iPad),`
  `near(1,u#1,iPod), near(1,u#1,iPad)`
- putting the constants in all valid combinations into the predicates, one obtains the set of all ground atoms (the Herbrand base)

# Interpretations are possible worlds

- in logics, an *interpretation* maps atoms to truth values
- an interpretation can be seen as a possible world state that defines the truth of every proposition
- MLNs express the uncertainty over which possible world state is the true world state as a probability distribution over interpretations

# A MLN Defines a Distribution Over Interpretations

- given a formula with free variables, a grounding assigns a constant to each free variable
- given an interpretation $x$, let $n_i(x)$ be the number of true groundings of formula $f_i$ under $x$
- then a MLN defines the following probability distribution over interpretations

$$P(x) = \frac{1}{Z} \prod_i \exp(w_i n_i(x))$$

# A closer look at the example

```
touches(t,user,device) => near(t,user,device).
1.0 near(t,user,device) => sees(t,user,device)
4.0 near(t,user,device) => near(t+1,user,device)
```

---

- line 1: a full-stop marks a deterministic formula
- line 2: the odds of seeing a device when being near it are $\frac{1}{\exp(-1.0)+1} \approx 0.73$
- line 3: the odds of remaining in the vicinity of a device in one time step are $\frac{1}{\exp(-4.0)+1} \approx 0.98$

The equivalence of formula weights and log-odds of the formula being true is only given when formulas do not share atoms.

# Some properties of MLNs

- degeneration to first-order logic when using infinite weights
- no uncertainty over the numbers of objects
- formula weights can be either specified or trained from data-sets

# Inference over MLNs

There are two common inference tasks for MLNs

1. find the most probable interpretation given some evidence $e$ (MAP or MPE)

$$\arg\max_x (P(X = x \mid e))$$

2. compute the marginal probabilities for some $z$ event (assigning to variables $Z \subseteq X$) given some evidence $e$ (MAR)

$$P(Z = z \mid e) = \sum_{x \in \sim\{Z\}} P(X = x \mid e)$$

During the rest of the talk we focus on computing marginal probabilities.

# Incorporating uncertain evidence

- a multivariate probability distribution can only be conditioned on a certain/sure assignment to a variable
- this means an observation can be “`near(15,u#1,iPad)` is true”
- observations obtained from classifiers can be uncertain: “`near(15,u#1,iPad)` is true with a certainty of $p$”
- we can express this by adding this formula with $w = \ln(\frac{p}{1-p})$ to the MLN:
  `w near(15,u#1,iPad)`

# Two ways to infer: MCMC and belief propagation

- inference on MLNs is usually done on an undirected graphical model obtained through grounding all formulas and predicates[2]
- then every algorithm for inference on graphical models can be used
- these algorithms usually fall into two categories
  - message passing / belief propagation
  - Markov Chain Monte Carlo methods
- won't give further details on these algorithms

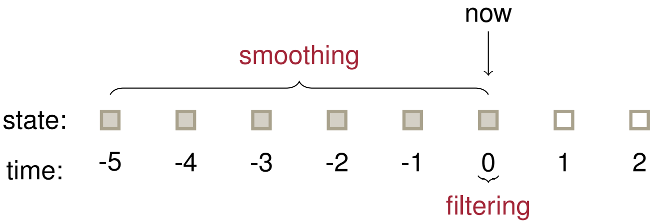
[2] with recent trends towards lifted inference

# Expressing dynamics with MLNs

- predicates can be made dynamic by adding a time index
- DMLNs are general enough to be used to express Dynamic Bayesian Networks with discrete variables
- in this way dynamic Markov Logic Networks (DMLNs) have already been applied in classification tasks
  - Adam Sadilek and Henry Kautz. "Recognizing Multi-Agent Activities from GPS Data". In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. 2010, pp. 1134–1139
  - S. Tran and L. Davis. "Event modeling and recognition using markov logic networks". In: *Computer Vision–ECCV 2008* (2008), pp. 610–623

# Which inference task for dynamic, probabilistic models is needed?

- filtering means predicting the *current* state of the world based on past observations
- smoothing means predicting past states in addition (new observations can make past predictions more accurate)




- for an application in a cognitive technical system
  we focus on filtering

# The offline approach to filtering/smoothing in DMLNs

- ground the DMLN from the start up to the current time step and infer on this graph
- this is the way DMLNs have been applied in the literature for dynamic classification tasks
- this includes the smoothing task and can only applied offline

# Online filtering in DMLNs

- we need a method to reuse information gained from inference for the previous time step
- one existing work describes an approximate online inference algorithm for message passing[3]
- how to make online inference for MCMC methods is less clear
- we propose temporal slice inference as an approximative online inference method for DMLNs, working with any inference algorithm for MLNs
- Thomas Geier and Susanne Biundo. "Approximate Online Inference for Dynamic Markov Logic Networks". In: *to appear at ICTAI*. 2011

[3] A. Nath and Domingos. "Efficient Belief Propagation for Utility Maximization and Repeated Inference". In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. 2010, pp. 1187–1192

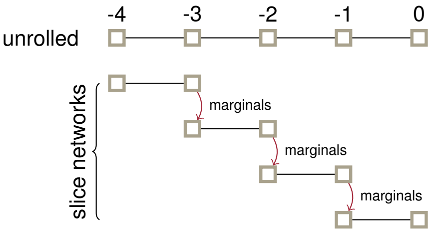
# Temporal Slice inference for DMLNs

- idea is similar to the factored frontier algorithm for DBNs[4]
- it approximates the distribution over the last time step by the marginal probabilities of the variables
- performs only a forward pass over the network, since no smoothing is required

[4] K. Murphy and Y. Weiss. “The factored frontier algorithm for approximate inference in DBNs”. In: *Proceedings of the 17th Conference on Uncertainty in AI*. 2001, pp. 378–385

# Temporal Slice inference for DMLNs (2)

- for each successive pair of time steps, have a temporal slice MLN spanning only these (like a two-step DBN)
- carry over information from the last slice networks by marginal probabilities over single random variables



# Temporal Slice inference for DLMNs (3)

- example slice for $t = 43$

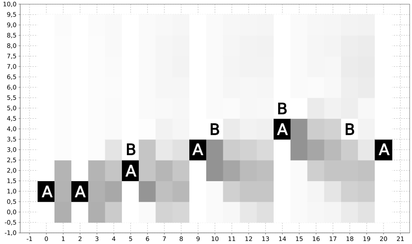
```
touches(43,user,device) => near(43,user,device).
1.0 near(43,user,device) => sees(43,user,device)
4.0 near(42,user,device) => near(43,user,device)

w1   touches(42,u#1,iPod)
w2   touches(42,u#1,iPad)
...
```

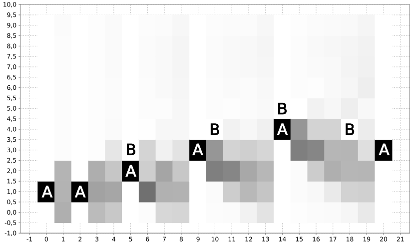
---

- `w1` $= \ln \frac{p_1}{1-p_1}$ with $p_1$ the marginal probability of `touches(42,u#1,iPod)` during the slice for $t = 42$
- the marginals of the slice MLN can then be inferred using any algorithm calculating marginals for MLNs

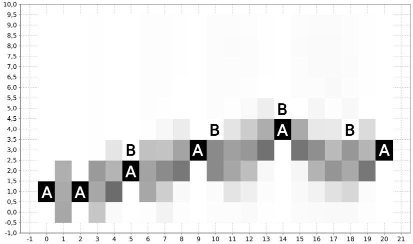
# Approximation error of temporal slice networks




Exact filtering



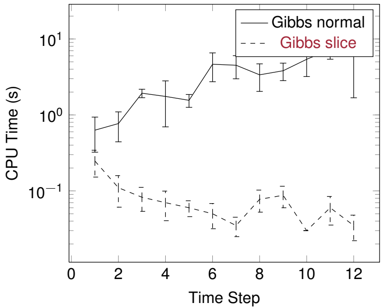

Temporal slice network filtering

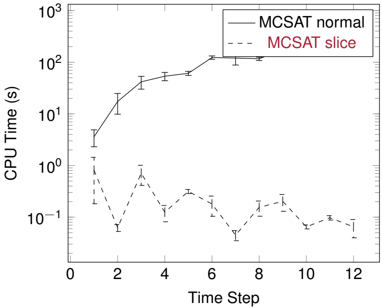



Smoothing

# CPU time for temporal slice inference algorithm

Plots show the CPU time for approximate filtering using temporal slice sampling compared to exact filtering, unrolling the complete network; running on an artificial MLN with random evidence; against number of time steps.



# Online DMLN knowledge base for a companion system

- a module containing the online inference engine has been fitted with a Semaine connection
- it can handle exact observations (with uncertain observations planned)
- it answers queries for marginal probabilities of single atoms

# Future work

- add continuous variables
- further improve inference methods for DMLNs (lifted inference, error-bounded approximation)
- extending and evaluating the model to support output channel selection (together with B3)
- incorporating uncertain observations from classifiers (together with C5) to recognize complex actions
  - a MLN constructed via “expert knowledge” can give hints as to when certain actions are applicable and thus improve classification
  - it allows to incorporate knowledge gained from other sources independently of the trained classifier

# Das Ende

Thank you for your attention.

# The example as a factorgraph

- the distribution defined by a MLN can be represented by a factor graph
- atoms are variable nodes
- ground formulas are factor nodes; connected to the atoms they contain

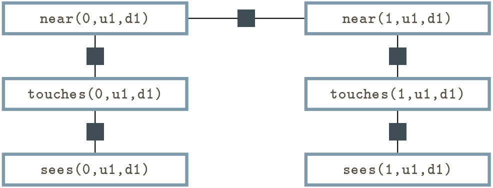



Figure: The factor graph for the example MLN.

# Why Gibbs sampling cannot handle deterministic dependencies

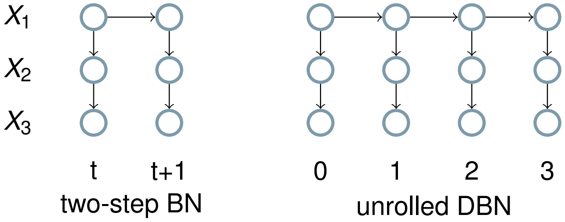
Assume a distribution over two binary random variables with the following probabilities.

| | $a$ | $\neg a$ |
|---|---|---|
| $b$ | 0.5 | 0 |
| $\neg b$ | 0 | 0.5 |

Starting in state $a$, $b$ the Markov chain produced by Gibbs sampling will never escape because it would have to temporarily enter a state with probability 0.

# Excursion: Dynamic Bayesian Networks

DBNs are probabilistic, multivariate, dynamic models.



# How to handle state? (2)

- DBNs “scale in time”
- but like BNs, DBNs are a propositional model, i.e., one has to specify each state variable
- a DBN about a person and her iPhone *is difficult to extend* to a DBN about a person and her iPhone and her iPod
- this is even though iPhones and iPods are very similar
- one needs to create new variables and new dependencies between them

# The Markov Chain Monte Carlo method

Given a MLN $L$ and its corresponding distribution $P_L$.

- we want to compute the marginal probability for some atom $a$ being true
- if we can obtain enough independent samples from $P_L$ then we can estimate $P_L(a = \mathit{true})$ by the frequency over the samples
- MCMC aims to obtain samples from $P_L$ by constructing a Markov chain whose stationary distribution equals $P_L$

# Gibbs sampling as a simple MCMC algorithm

Gibbs sampling works the following.

1. choose a starting interpretation randomly
2. choose an atom randomly
3. resample its truth value given the rest of the interpretation
4. repeat with 2

Gibbs sampling cannot handle deterministic dependencies arising from hard formulas.

# Handling deterministic dependencies with MCSAT

MCSAT improves over Gibbs sampling in handling deterministic dependencies.

- MCSAT is a different MCMC algorithm
- MCSAT re-samples every atom in every step by using slice sampling
- as a subroutine it uses the SampleSAT algorithm to sample uniformly from the satisfying interpretations of a CNF formula
- in contrast to Gibbs sampling MCSAT is able to jump between the modes of the distribution easily

# Obtain all groundings of a formula

- the formula `touches(t,user,device) => near(t,user,device)` contains the free variables `t,user,device`
- a grounding of a formula assigns a constant to each free variable
- a possible grounding of the presented formula is `touches(0,u1,d2) => near(0,u1,d2)`

# How to make a choice — The principle of maximizing expected utility

How to make decisions based on uncertain knowledge?

1. have a set of possible actions $A$
2. assign a utility to each state/action pair

$$u : A \times S \to \mathbb{R}$$

3. the utility $u_a$ for action $a$ is then a random variable
4. choose the action $a$ with the largest expected utility

$$a = \arg\max_{a' \in A} E[u_{a'}]$$

# Evaluation Domains

- Dynamic Smokers

```
1.0 smokes(t,x) => cancer(t,x)
1.0 friends(t,x,y) =>
    (smokes(t,x) => smokes(t,y))
3.0 friends(t,x,y) => friends(t+1,x,y)
3.0 smokes(t,x) => smokes(t+1,x)
```

- Simple Social Force

```
2 at(t+1,a,x) =>
  at(t,a,x-1) v at(t,a,x) v at(t,a,x+1)
1.5 !(at(t,a1,x) AND at(t,a2,x))
```

# Grounded Slice Network