Advanced Machine Learning Techniques for Temporal, Multimedia, and Relational Data

Vibhav Gogate
The University of Texas at Dallas

Many slides courtesy of Pedro Domingos
Statistical Relational Learning: Motivation

• Most learners assume i.i.d. data (independent and identically distributed)
  – One type of object
  – Objects have no relation to each other

• Real applications: dependent, variously distributed data
  – Multiple types of objects
  – Relations between objects
Examples

- Web search
- Information extraction
- Natural language processing
- Perception
- Medical diagnosis
- Computational biology
- Social networks
- Ubiquitous computing
- Etc.
Costs and Benefits of SRL

• **Benefits**
  – Better predictive accuracy
  – Better understanding of domains
  – Growth path for machine learning

• **Costs**
  – Learning is much harder
  – Inference becomes a crucial issue
  – Greater complexity for user
Goal and Progress

• **Goal:**
  Learn from non-i.i.d. data as easily as from i.i.d. data

• Progress to date
  – Burgeoning research area
  – We’re “close enough” to goal
  – Easy-to-use open-source software available

• Lots of research questions (old and new)
Plan

• We have the elements:
  – **Probability** for handling uncertainty
  – **Logic** for representing types, relations, and complex dependencies between them
  – **Learning** and **inference** algorithms for each

• Figure out how to put them together

• Tremendous leverage on a wide range of applications
Disclaimers

• Not a complete survey of statistical relational learning
• Or of foundational areas
• Focus is practical, not theoretical
• Assumes basic background in logic, probability and statistics, etc.
• Please ask questions
• Tutorial and examples available at alchemy.cs.washington.edu
• New version of alchemy available on my website – http://www.hlt.utdallas.edu/~vgogate/software.html
Markov Logic

• An approach for statistical relational learning
• Most developed approach to date
• Many other approaches can be viewed as special cases
• Main focus of rest of this tutorial
Markov Logic: Intuition

• A logical KB is a set of hard constraints on the set of possible worlds

• Let’s make them soft constraints: When a world violates a formula, it becomes less probable, not impossible

• Give each formula a weight
  (Higher weight \( \Rightarrow \) Stronger constraint)

\[
P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)
\]
Markov Logic: Definition

• A **Markov Logic Network (MLN)** is a set of pairs \((F, w)\) where
  – \(F\) is a formula in first-order logic
  – \(w\) is a real number

• Together with a set of constants, it defines a Markov network with
  – One node for each grounding of each predicate in the MLN
  – One feature for each grounding of each formula \(F\) in the MLN, with the corresponding weight \(w\)
Example: Friends & Smokers

- Smoking causes cancer.
- Friends have similar smoking habits.
Example: Friends & Smokers

\[ \forall x \ Smokes(x) \Rightarrow Cancer(x) \]
\[ \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \]
Example: Friends & Smokers

| 1.5 | $\forall x \ Smokes(x) \Rightarrow Cancer(x)$ |
| 1.1 | $\forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \iff Smokes(y))$ |
Example: Friends & Smokers

<table>
<thead>
<tr>
<th></th>
<th>( \forall x , \text{Smokes}(x) \Rightarrow \text{Cancer}(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>( \forall x, y , \text{Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y)) )</td>
</tr>
</tbody>
</table>

Two constants: **Anna** (A) and **Bob** (B)
Example: Friends & Smokers

| 1.5 | \( \forall x \; \text{Smokes}(x) \Rightarrow \text{Cancer}(x) \) |
| 1.1 | \( \forall x, y \; \text{Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y)) \) |

Two constants: **Anna** (A) and **Bob** (B)
Example: Friends & Smokers

Two constants: Anna (A) and Bob (B)
Example: Friends & Smokers

1.5 $\forall x \ Smokes(x) \Rightarrow Cancer(x)$
1.1 $\forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

Two constants: **Anna** (A) and **Bob** (B)
Example: Friends & Smokers

<table>
<thead>
<tr>
<th>1.5</th>
<th>$\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>$\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$</td>
</tr>
</tbody>
</table>

Two constants: **Anna** (A) and **Bob** (B)
Markov Logic Networks

• **MLN is template** for ground Markov nets
• Probability of a world \( x \):

\[
P(x) = \frac{1}{Z} \exp \left( \sum_i w_i n_i(x) \right)
\]

  - Weight of formula \( i \)
  - No. of true groundings of formula \( i \) in \( x \)

• **Typed** variables and constants greatly reduce size of ground Markov net
• Functions, existential quantifiers, etc.
• Infinite and continuous domains
**Markov logic**

<table>
<thead>
<tr>
<th>Formula</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$</td>
<td>1.5</td>
</tr>
<tr>
<td>$\forall x, y \text{ Smokes}(x) \land \text{Friends}(x, y) \Rightarrow \text{Smokes}(y)$</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Two constants: **Anna** (A) and **Bob** (B)

- $\text{Smokes}(A) \Rightarrow \text{Cancer}(A), \exp(1.5)$
- $\text{Smokes}(B) \Rightarrow \text{Cancer}(B), \exp(1.5)$
- $\text{Smokes}(A) \land \text{Friends}(A, A) \Rightarrow \text{Smokes}(A), \exp(1.1)$
- $\text{Smokes}(A) \land \text{Friends}(A, B) \Rightarrow \text{Smokes}(B), \exp(1.1)$
- $\text{Smokes}(B) \land \text{Friends}(B, A) \Rightarrow \text{Smokes}(A), \exp(1.1)$
- $\text{Smokes}(B) \land \text{Friends}(B, B) \Rightarrow \text{Smokes}(B), \exp(1.1)$

Probability of $\omega$ is proportional to the product of exponentiated weights of satisfied ground formulas

World $\omega$: $S(A), \neg C(A), F(A, A), \neg F(A, B), F(B, A), F(B, B), \neg S(B), \neg C(B)$

$n_1 = 1$

$n_2 = 4$
Relation to Statistical Models

• Special cases:
  – Markov networks
  – Markov random fields
  – Bayesian networks
  – Log-linear models
  – Exponential models
  – Max. entropy models
  – Gibbs distributions
  – Boltzmann machines
  – Logistic regression
  – Hidden Markov models
  – Conditional random fields
Relation to First-Order Logic

• Infinite weights $\Rightarrow$ First-order logic
• Satisfiable KB, positive weights $\Rightarrow$
  Satisfying assignments = Modes of distribution
• Markov logic allows contradictions between formulas
Marginal/Counting Inference

**Probabilistic Theorem Proving problem**

**Given** Probabilistic knowledge base $K$
Query formula $Q$

**Output** $P(Q|K)$

*Compare to:*

**Logical Theorem proving**

**Given** Knowledge base $K$
Query formula $Q$

**Output:** Does $K$ entail $Q$
Lifted Weighted Model Counting

• ModelCount(CNF) = # worlds that satisfy CNF
• Assign a weight to each literal
• Weight(world) = product of literals that are true in the world

• Weighted model counting:
  – Sum of weights of all world that satisfy CNF

• Lifted Weighted model counting:
  – Each literal is first-order literal
Inference Problems

PTP is reducible to LWMC
Weighted Model Counting

\[ \text{WMC}(CNF, \text{weights}) \]

if all clauses in \( CNF \) are satisfied

\[
\text{return } \prod_{A \in A(CNF)} (w_A + w_{\neg A})
\]

if \( CNF \) has empty unsatisfied clause \text{return} 0

Base Case
Weighted Model Counting

**WMC**(*CNF, weights*)

if all clauses in CNF are satisfied

return \( \prod_{A \in A(CNF)} (w_A + w_{\neg A}) \)

if CNF has empty unsatisfied clause return 0

if CNF can be partitioned into CNFs \( C_1, \ldots, C_k \) sharing no atoms

return \( \prod_{i=1}^{k} WMC(C_i, weights) \)
Weighted Model Counting

\[ \text{WMC}(\text{CNF}, \text{weights}) \]

- If all clauses in \( \text{CNF} \) are satisfied
  \[ \text{return } \prod_{A \in A(\text{CNF})} (w_A + w_{\neg A}) \]

- If \( \text{CNF} \) has empty unsatisfied clause \text{return 0}

- If \( \text{CNF} \) can be partitioned into \( \text{CNFs} \) \( C_1, \ldots, C_k \) sharing no atoms
  \[ \text{return } \prod_{i=1}^{k} \text{WMC}(C_i, \text{weights}) \]

- Choose an atom \( A \)

  \[ \text{return } w_A \text{WMC}(\text{CNF} | A, \text{weights}) + w_{\neg A} \text{WMC}(\text{CNF} | \neg A, \text{weights}) \]
First-Order Case

• PTP schema remains the same
• Conversion of PKB to hard CNF and weights:
  New atom in $F_i \iff A_i$ is now $\text{Predicate}_i(\text{variables in } F_i, \text{constants in } F_i)$
• New argument in WMC:
  Set of substitution constraints of the form $x = A, x \neq A, x = y, x \neq y$
• Lift each step of WMC
Logical/First-order Structure

• Exploit Symmetry in the first-order representation

\[
Z = \prod_{X \in \{A,B,C,D\}} Z[x\backslash X]
\]

Independent

\[
R(x) \lor S(x), v \\
R(A) \lor S(A), v \\
R(B) \lor S(B), v \\
R(C) \lor S(C), v \\
R(D) \lor S(D), v
\]

Linear time

Independent And Identical

\[
R(x) \lor S(x), v \\
R(A) \lor S(A), v \\
R(B) \lor S(B), v \\
R(C) \lor S(C), v \\
R(D) \lor S(D), v
\]

\[
Z = (Z[x\backslash X])^4
\]

Constant time
Lifted/First-order Structure: POWER RULE

• Of course, you cannot always take powers and solve it efficiently

• Following conditions must be satisfied for a variable x:
  – “x” must appear in every predicate symbol in the formula
  – If there is another unifiable variable “y”, then “x” and “y” must appear in the same position in every predicate in every formula

• MLN: R(x,y) v S(x,z) and R(y,z) v T(y,u)
  – Z=Z[x/A, y/A]^n

• MLN: R(x,y) v S(x,z) and R(z,y) v T(y,u)
  – cannot apply.
**Lifted/First-order Structure: BINOMIAL RULE**

- Applies to singleton atoms
  - Condition on singleton atoms in a special way
- MLN: \((f=R(x) \lor S(x,y) \lor T(y), v)\)
  - If domain-size of \(x\) is “\(n\)”\), naïve conditioning on \(R(x)\) yields \(2^n\) truth-assignments
- BINOMIAL RULE: Condition on \((n+1)\)-truth assignments

\[
Z(f, v) = \sum_{i=0}^{n} \binom{n}{i} Z(f_{R,i}, v)
\]

\(f_{R,i}\) is obtained from \(f\) by setting exactly "\(i\)" groundings of \(R\) to True
Lifted Weighted Model Counting

**LWMC**(*CNF, substs, weights*)

if all clauses in *CNF* are satisfied
return \( \prod_{A \in A(CNF)} (w_A + w_{-A})^{n_A(substs)} \)

if *CNF* has empty unsatisfied clause return 0

Base Case
Lifted Weighted Model Counting

\[ \text{LWMC}(\text{CNF}, \text{substs}, \text{weights}) \]

\begin{align*}
&\text{if all clauses in } \text{CNF} \text{ are satisfied} \\
&\quad \text{return } \prod_{A \in A(\text{CNF})} \left( w_A + w_{\neg A} \right)^{n_A(\text{substs})} \\
&\text{if } \text{CNF} \text{ has empty unsatisfied clause return } 0 \\
&\text{if there exists a lifted decomposition of } \text{CNF} \\
&\quad \text{return } \prod_{i=1}^{k} \left[ \text{LWMC}(\text{CNF}_{i,1}, \text{substs}, \text{weights}) \right]^{m_i}
\end{align*}
Lifted Weighted Model Counting

\[ \text{LWMC}(\text{CNF}, \text{substs}, \text{weights}) \]

\begin{align*}
\text{if all clauses in } \text{CNF} \text{ are satisfied} & \quad \text{return } \prod_{A \in A(\text{CNF})} (w_A + w_{\neg A})^{n_A(\text{substs})} \\
\text{if } \text{CNF} \text{ has empty unsatisfied clause return } 0 & \\
\text{if there exists a lifted decomposition of } \text{CNF} & \\
\text{return } \prod_{i=1}^{k} [\text{LWMC}(\text{CNF}_{i,1}, \text{substs}, \text{weights})]^{m_i} \\
\text{choose an atom } A & \\
\text{return } \sum_{i=1}^{l} n_i w_A^{f_i} w_{\neg A}^{f_i} \text{LWMC}(\text{CNF} \mid \sigma_j, \text{substs}_j, \text{weights})
\end{align*}
Approximate Inference

\[ \text{WMC}(\text{CNF}, \text{weights}) \]

\textbf{if} all clauses in \text{CNF} are satisfied
\textbf{return} \( \prod_{A \in \text{A(CNF)}} (w_A + w_{\neg A}) \)

\textbf{if} \text{CNF} has empty unsatisfied clause \textbf{return} 0

\textbf{if} \text{CNF} can be partitioned into \text{CNFs} \( C_1, \ldots, C_k \) sharing no atoms
\textbf{return} \( \prod_{i=1}^{k} \text{WMC}(C_i, \text{weights}) \)

choose an atom \( A \)
\textbf{return} \( \frac{w_A}{Q(A | \text{CNF}, \text{weights})} \text{WMC}(\text{CNF} | A, \text{weights}) \)

with probability \( Q(A | \text{CNF}, \text{weights}) \), etc.
Link Prediction

Time in seconds vs. Number of objects

PTP
FOVE

UT DALLAS
Coreference (Cora)

![Graph showing coreference results over time]

- **Negative log likelihood**
- **Time in minutes**

Log-log scale graph with lines for:
- MC-LWMC
- Lifted-BP
- MC-WMC
- MC-SAT

Vibhav Gogate
MAP/MPE Inference

- **Problem**: Find most likely state of world given evidence

\[
\max_y P(y \mid x)
\]

Query  Evidence
MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

\[
\max_y \frac{1}{Z_x} \exp \left( \sum_i w_i n_i (x, y) \right)
\]
MAP/MPE Inference

• **Problem:** Find most likely state of world given evidence

\[
\max_y \sum_i w_i n_i (x, y)
\]
MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence
  \[
  \max_y \sum_i w_i n_i(x, y)
  \]

- This is just the weighted MaxSAT problem
- Use weighted SAT solver (e.g., MaxWalkSAT [Kautz et al., 1997])
- Potentially faster than logical inference (!)
The MaxWalkSAT Algorithm

for $i \leftarrow 1$ to $\text{max-tries}$ do
    \textit{solution} = random truth assignment
    for $j \leftarrow 1$ to $\text{max-flips}$ do
        if $\sum \text{weights(sat. clauses)} > \text{threshold}$ then
            return \textit{solution}
        \textit{c} $\leftarrow$ random unsatisfied clause
        with probability $p$
            flip a random variable in \textit{c}
        else
            flip variable in \textit{c} that maximizes $\sum \text{weights(sat. clauses)}$
    return failure, best \textit{solution} found
But ... Memory Explosion

• **Problem:**
  If there are $n$ constants and $k$ distinct logical variables in each formula, we get $O(n^k)$ ground formulas

• **Solution:**
  Exploit sparseness; ground clauses lazily
  – LazySAT algorithm [Singla & Domingos, 2006]
  – Fast WALKSAT by grounding to monadic first-order logic (In progress)
  – Lifted MPE (in progress)
Learning

• Data is a relational database
• Closed world assumption (if not: EM)
• Learning parameters (weights)
• Learning structure (formulas)
Weight Learning

• Parameter tying: Groundings of same clause

\[
\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)]
\]

No. of times clause \(i\) is true in data

Expected no. times clause \(i\) is true according to MLN

• Generative learning: Pseudo-likelihood
• Discriminative learning: Cond. likelihood, use Lifted sampling or MaxWalkSAT for inference
Structure Learning

• Generalizes feature induction in Markov nets
• Any inductive logic programming approach can be used, but . . .
• Goal is to induce any clauses, not just Horn
• Evaluation function should be likelihood
• Requires learning weights for each candidate
• Turns out not to be bottleneck
• Bottleneck is counting clause groundings
• Solution: Subsampling

Vibhav Gogate  UT DALLAS
Structure Learning

- **Initial state:** Unit clauses or hand-coded KB
- **Operators:** Add/remove literal, flip sign
- **Evaluation function:**
  Pseudo-likelihood + Structure prior
- **Search:** Beam search, shortest-first search
Alchemy

Open-source software including:

• Full first-order logic syntax
• Generative & discriminative weight learning
• Structure learning
• Weighted satisfiability and MCMC
• Programming language features

• alchemy.cs.washington.edu
• http://www.hlt.utdallas.edu/~vgogate/software
<table>
<thead>
<tr>
<th></th>
<th>Alchemy</th>
<th>Prolog</th>
<th>BUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Representation</strong></td>
<td>F.O. Logic + Markov nets</td>
<td>Horn clauses</td>
<td>Bayes nets</td>
</tr>
<tr>
<td><strong>Inference</strong></td>
<td>Probabilistic Theorem proving</td>
<td>Theorem proving</td>
<td>Gibbs sampling</td>
</tr>
<tr>
<td><strong>Learning</strong></td>
<td>Parameters &amp; structure</td>
<td>No</td>
<td>Params.</td>
</tr>
<tr>
<td><strong>Uncertainty</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Relational</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Applications

- Statistical parsing
- Semantic processing
- Bayesian networks
- Relational models
- Robot mapping
- Planning and MDPs
- Practical tips

- Basics
- Logistic regression
- Hypertext classification
- Information retrieval
- Entity resolution
- Hidden Markov models
- Information extraction
Running Alchemy

• Programs
  – Infer
  – Learnwts
  – Learnstruct
  – LiftedInfer

• Options

• MLN file
  – Types (optional)
  – Predicates
  – Formulas

• Database files
Uniform Distribn.: Empty MLN

Example: Unbiased coin flips

Type: \( \text{flip} = \{ 1, \ldots, 20 \} \)

Predicate: \( \text{Heads}(\text{flip}) \)

\[
P(\text{Heads}(f)) = \frac{1}{Z} e^0 \left( \frac{1}{Z} e^0 + \frac{1}{Z} e^0 \right) = \frac{1}{2}
\]
Binomial Distribn.: Unit Clause

Example: Biased coin flips

Type: \( \text{flip} = \{ 1, \ldots, 20 \} \)

Predicate: \( \text{Heads(\text{flip})} \)

Formula: \( \text{Heads(f)} \)

Weight: Log odds of heads: \( w = \log \left( \frac{p}{1-p} \right) \)

\[
P(\text{Heads(f)}) = \frac{\frac{1}{Z} e^w}{\frac{1}{Z} e^w + \frac{1}{Z} e^0} = \frac{1}{1 + e^{-w}} = p
\]

By default, MLN includes unit clauses for all predicates (captures marginal distributions, etc.)
Multinomial Distribution

Example: Throwing die

Types: \( \text{throw} = \{ 1, \ldots, 20 \} \)

\( \text{face} = \{ 1, \ldots, 6 \} \)

Predicate: \( \text{Outcome}(\text{throw,face}) \)

Formulas: \( \text{Outcome}(t,f) \land f \neq f' \Rightarrow \neg \text{Outcome}(t,f') \).

\[ \exists f \text{ Outcome}(t,f). \]

Too cumbersome!
Example: Throwing die

Types:  \( \text{throw} = \{ 1, \ldots, 20 \} \)
      \( \text{face} = \{ 1, \ldots, 6 \} \)

Predicate: \( \text{Outcome(throw,face!)} \)

Formulas:

Semantics: Arguments without “!” determine arguments with “!”.
Also makes inference more efficient (triggers blocking).
Example: Throwing biased die

Types: \( \text{throw} = \{ 1, \ldots, 20 \} \)
\(\text{face} = \{ 1, \ldots, 6 \} \)

Predicate: \(\text{Outcome(throw,face!)}\)
Formulas: \(\text{Outcome}(t,+f)\)

Semantics: Learn weight for each grounding of args with “+”. 
Logistic Regression

Logistic regression:
\[
\log \left( \frac{P(C = 1 | F = f)}{P(C = 0 | F = f)} \right) = a + \sum b_i f_i
\]

Type:
\[
\text{obj} = \{ 1, \ldots, n \}
\]

Query predicate:
\[
C(\text{obj})
\]

Evidence predicates:
\[
F_i(\text{obj})
\]

Formulas:
\[
a = C(x) \\
b_i = F_i(x) \land C(x)
\]

Resulting distribution:
\[
P(C = c, F = f) = \frac{1}{Z} \exp \left( ac + \sum_i b_i f_i c \right)
\]

Therefore:
\[
\log \left( \frac{P(C = 1 | F = f)}{P(C = 0 | F = f)} \right) = \log \left( \frac{\exp(a + \sum b_i f_i)}{\exp(0)} \right) = a + \sum b_i f_i
\]
Text Classification

\[ \text{page} = \{ 1, \ldots, n \} \]
\[ \text{word} = \{ \ldots \} \]
\[ \text{topic} = \{ \ldots \} \]

\[ \text{Topic}(\text{page,topic!}) \]
\[ \text{HasWord}(\text{page,word}) \]

\[ \neg \text{Topic}(p,t) \]
\[ \text{HasWord}(p,+w) \Rightarrow \text{Topic}(p,+t) \]

For all \( w, t \) pairs we will learn a weight
Which denotes how indicative of a topic a particular word is
Hypertext Classification

Topic(page,topic!)
HasWord(page,word)
Links(page,page)

HasWord(p,+w) => Topic(p,+t)
Topic(p,t) ^ Links(p,p') => Topic(p',t)

Use hyperlinks to help classify text

Information Retrieval

\[ \text{InQuery}(\text{word}) \quad \text{// Suppose word is in our search query} \]
\[ \text{HasWord}(\text{page}, \text{word}) \]
\[ \text{Relevant}(\text{page}) \]

\[ \text{InQuery}(+w) \land \text{HasWord}(p, +w) \implies \text{Relevant}(p) \]
\[ \text{Relevant}(p) \land \text{Links}(p, p') \implies \text{Relevant}(p') \]

Problem: Given database, find duplicate records

HasToken(token, field, record)
SameField(field, record, record)
SameRecord(record, record)

HasToken(+t,+f,r) \land HasToken(+t,+f,r')
    \Rightarrow \quad \text{SameField}(f,r,r')
SameField(+f,r,r') \Rightarrow \text{SameRecord}(r,r')
SameRecord(r,r') \land SameRecord(r',r'')
    \Rightarrow \quad \text{SameRecord}(r,r'')

Can also resolve fields:

\[
\text{HasToken}(t,f,r) \land \text{HasToken}(t,f',r') \Rightarrow \text{SameField}(f,r,r')
\]
\[
\text{SameField}(f,r,r') \iff \text{SameRecord}(r,r')
\]
\[
\text{SameRecord}(r,r') \land \text{SameRecord}(r',r'') \Rightarrow \text{SameRecord}(r,r'')
\]
\[
\text{SameField}(f,r,r') \land \text{SameField}(f,r',r'') \Rightarrow \text{SameField}(f,r,r'')
\]

Hidden Markov Models

\[ \text{obs} = \{ \text{Obs1, \ldots , ObsN} \} \]
\[ \text{state} = \{ \text{St1, \ldots , StM} \} \]
\[ \text{time} = \{ 0, \ldots , T \} \]

State(state!, time)
Obs(obs!, time)

State(+s,0)
State(+s,t) \Rightarrow State(+s',t+1)
Obs(+o,t) \Rightarrow State(+s,t)
Practical Tips

• Add all unit clauses (the default)
• Implications vs. conjunctions
• Open/closed world assumptions
• How to handle uncertain data: $R(x,y) \Rightarrow R'(x,y)$ (the “HMM trick”)
• Controlling complexity
  – Low clause arities
  – Low numbers of constants
  – Short inference chains
• Use the simplest MLN that works
• Cycle: Add/delete formulas, learn and test