Combining logic with probability

Motivation

- First-order logic is a powerful language to represent complex relational information
- Probability is the standard way to represent uncertainty in knowledge
- Combining the two would allow to model complex probabilistic relationships in the domain of interest

Combining logic with probability

logic graphical models

- Graphical models are a mean to represent joint probabilities highlighting the relational structure among variables
- A compressed representation of such models can be obtained using templates, cliques in the graphs sharing common parameters (e.g. as in HMM for BN or CRF for MN)
- Logic can be seen as a language to build templates for graphical models
- Logic based versions of HMM, BN and MN have been defined

First-order logic (in a nutshell)

Symbols

- Constant symbols representing objects in the domain (e.g. Nick, Polly)
- Variable symbols which take objects in the domain as values (e.g. x, y)
- Function symbols which mapping tuples of objects to objects (e.g. BandOf). Each function symbol has an arity (i.e. number of arguments)
- Predicate symbols representing relations among objects or object attributes (e.g. Singer, SangTogether). Each predicate symbol has an arity.

First-order logic

Terms

- A term is an expression representing an object in the domain. It can be:
  - A constant (e.g. Niel)
  - A variable (e.g. x)
  - A function applied to a tuple of objects. E.g.: BandOf(Niel), SonOf(f,m), Age(MotherOf(John))

First-order logic

Formulas

- A (well formed) atomic formula (or atom) is a predicate applied to a tuple of objects. E.g.: Singer(Nick), SangTogether(Nick, Polly), Friends(X, BrotherOf(Emy))
- Composite formulas are constructed from atomic formulas using logical connectives and quantifiers
First-order logic

Connectives

negation $\neg F$: true iff formula $F$ is false

conjunction $F_1 \land F_2$: true iff both formulas $F_1, F_2$ are true

disjunction $F_1 \lor F_2$: true iff at least one of the two formulas $F_1, F_2$ is true

implication $F_1 \Rightarrow F_2$: true iff $F_1$ is false or $F_2$ is true (same as $F_2 \lor \neg F_1$)

equivalence $F_1 \Leftrightarrow F_2$: true iff $F_1$ and $F_2$ are both true or both false (same as $(F_1 \Rightarrow F_2) \land (F_2 \Rightarrow F_1)$)

Literals

• A positive literal is an atomic formula

• A negative literal is a negated atomic formula

First-order logic

Quantifiers

existential quantifier $\exists x F_1$: true iff $F_1$ is true for at least one object $x$ in the domain. E.g.: $\exists x \text{ Friends}(x, \text{BrotherOf}(\text{Emy}))$

universal quantifier $\forall x F_1$: true iff $F_1$ is true for all objects $x$ in the domain. E.g.: $\forall x \text{ Friends}(x, \text{BrotherOf}(\text{Emy}))$

Scope

• The scope of a quantifier in a certain formula is the (sub)formula to which the quantifiers applies

First-order logic

Precedence

• Quantifiers have the highest precedence

• Negation has higher precedence than other connectives

• Conjunction has higher precedence than disjunction

• Disjunction have higher precedence than implication and equivalence

• Precedence rules can as usual be overruled using parentheses

Examples

• Emy and her brother have no common friends:
  $$\neg \exists x (\text{Friends}(x, \text{Emy}) \land \text{Friends}(x, \text{BrotherOf}(\text{Emy})))$$

• All birds fly:
  $$\forall x (\text{Bird}(x) \Rightarrow \text{Flies}(x))$$
First-order logic

Closed formulas
• A variable-occurrence within the scope of a quantifier is called *bound*. E.g. $x$ in:
  \[ \forall x (\text{Bird}(x) \Rightarrow \text{Flies}(x)) \]
• A variable-occurrence outside the scope of any quantifier is called *free*. E.g. $y$ in:
  \[ \neg \exists x (\text{Friends}(x, \text{Emy}) \land \text{Friends}(x, y)) \]
• A *closed* formula is a formula which contains no free occurrence of variables

*Note*
• We will be interested in closed formulas only

First-order logic

Ground terms and formulas
• A *ground* term is a term containing no variables
• A *ground* formula is a formula made of only ground terms

First-order logic

First order language
• The set of symbols (constants, variables, functions, predicates, connectives, quantifiers) constitute a first-order *alphabet*
• A first order *language* given by the alphabet is the set of formulas which can be constructed from symbols in the alphabet

Knowledge base (KB)
• A first-order knowledge base is a set of formulas
• Formulas in the KB are implicitly conjoined
• A KB can thus be seen as a single large formula

First-order logic

Interpretation
• An *interpretation* provides *semantics* to a first order language by:
  1. defining a domain containing all possible objects
  2. mapping each ground term to an object in the domain
  3. assigning a truth value to each ground atomic formula (*a possible world*)
• The truth value of complex formulas can be obtained combining interpretation assignments with connective and quantifier rules
First-order logic: example

The formula is true under the interpretation as the following atomic formulas are true:

\[ \neg \exists x \left( \text{Friends}(x, \text{Emy}) \land \neg \text{Friends}(x, \text{BrotherOf}(\text{Emy})) \right) \]

- Friends(Ann, Emy)
- Friends(Ann, BrotherOf(Emy))
- Friends(Matt, Emy)
- Friends(Matt, BrotherOf(Emy))
- Friends(John, Emy)
- Friends(John, BrotherOf(Emy))
First-order logic

Types

- Objects can be typed (e.g. people, cities, animals)
- A typed variable can only range over objects of the corresponding type
- A typed term can only take arguments from the corresponding type. E.g. MotherOf(John), MotherOf(Amy)

First-order logic

Inference in first-order logic

- A formula $F$ is satisfiable iff there exists an interpretation under which the formula is true
- A formula $F$ is entailed by a KB iff $F$ is true for all interpretations for which the KB is true. We write it:

$$KB \models F$$
the formula is a logical consequence of KB, not depending on the particular interpretation
• Logical entailment is usually done by refutation: proving that $KB \land \neg F$ is unsatisfiable

*Note*
• Logical entailment allows to *extend* a KB inferring new formulas which are true for the same interpretations for which the KB is true

**First-order logic**

**Clausal form**
• The *clausal* form or *conjunctive normal form* (CNF) is a regular form to represent formulas which is convenient for automated inference:
  – A *clause* is a disjunction of literals.
  – A KB in CNF is a conjunction of clauses.
• Variables in KB in CNF are always implicitly assumed to be universally quantified.
• Any KB can be converted in CNF by a mechanical sequence of steps
• Existential quantifiers are replaced by Skolem constants or functions

**Conversion to clausal form: example**

<table>
<thead>
<tr>
<th>First Order Logic</th>
<th>Clausal Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Every bird flies” $\forall x \ (Bird(x) \Rightarrow Flies(x))$</td>
<td>$Flies(x) \lor \neg Bird(x)$</td>
</tr>
<tr>
<td>“Every predator of a bird is a bird” $\forall x, y \ (Predates(x,y) \land Bird(y) \Rightarrow Bird(x))$</td>
<td>$Bird(x) \lor \neg Bird(y) \lor \neg Predates(x,y)$</td>
</tr>
<tr>
<td>“Every prey has a predator” $\forall y \ (Prey(y) \Rightarrow \exists x \ Predates(x,y))$</td>
<td>$Predates(PredatorOf(y),y) \lor \neg Prey(y)$</td>
</tr>
</tbody>
</table>

**First-order logic**

**Problem of uncertainty**
• In most real world scenarios, logic formulas are *typically* but not *always* true
• For instance:
  – “Every bird flies” : what about an ostrich (or Charlie Parker) ?
  – “Every predator of a bird is a bird”: what about lions with ostriches (or heroin with Parker) ?
  – “Every prey has a predator”: predators can be extinct
• A world failing to satisfy even a single formula would not be possible
• there could be no possible world satisfying all formulas
First-order logic

Handling uncertainty

- We can relax the hard constraint assumption on satisfying all formulas
- A possible world not satisfying a certain formula will simply be less likely
- The more formula a possible world satisfies, the more likely it is
- Each formula can have a weight indicating how strong a constraint it should be for possible worlds
- Higher weight indicates higher probability of a world satisfying the formula wrt one not satisfying it

Markov Logic networks

Definition

- A Markov Logic Network (MLN) $L$ is a set of pairs $(F_i, w_i)$ where:
  - $F_i$ is a formula in first-order logic
  - $w_i$ is a real number (the weight of the formula)
- Applied to a finite set of constants $C = \{c_1, \ldots, c_{|C|}\}$ it defines a Markov network $M_{L,C}$:
  - $M_{L,C}$ has one binary node for each possible grounding of each atom in $L$. The value of the node is 1 if the ground atom is true, 0 otherwise.
  - $M_{L,C}$ has one feature for each possible grounding of each formula $F_i$ in $L$. The value of the feature is 1 if the ground formula is true, 0 otherwise. The weight of the feature is the weight $w_i$ of the corresponding formula

Markov Logic networks

Intuition

- A MLN is a template for Markov Networks, based on logical descriptions
- Single atoms in the template will generate nodes in the network
- Formulas in the template will be generate cliques in the network
- There is an edge between two nodes iff the corresponding ground atoms appear together in at least one grounding of a formula in $L$

Markov Logic networks: example

![Graph Example]
Ground network

- A MLN with two (weighted) formulas:

  \[ w_1 \forall x (\text{Bird}(x) \Rightarrow \text{Flies}(x)) \]
  \[ w_2 \forall x, y (\text{Predates}(x,y) \land \text{Bird}(y) \Rightarrow \text{Bird}(x)) \]

- applied to a set of two constants \{\text{Sparrow}, \text{Eagle}\}
- generates the Markov Network shown in figure

Markov Logic networks

Joint probability

- A ground MLN specifies a joint probability distribution over possible worlds (i.e. truth value assignments to all ground atoms)
- The probability of a possible world \( x \) is:

  \[ p(x) = \frac{1}{Z} \exp \left( \sum_{i=1}^{F} w_i n_i(x) \right) \]

  where:
  - the sum ranges over formulas in the MLN (i.e. clique templates in the Markov Network)
  - \( n_i(x) \) is the number of true groundings of formula \( F_i \) in \( x \)

Joint probability: example 1

Computing joint probability

\[ p(x) = \frac{1}{Z} \exp(2w_1 + w_2) \]

- The partition function \( Z \) sums over all possible worlds (i.e. all possible combination of truth assignments to ground atoms)
Joint probability: example 2

Computing joint probability

\[ p(x) = \frac{1}{Z} \exp(w_1) \]

- This possible world is less likely than the previous, as it violates two more (ground) formulas:
  \[
  \begin{align*}
  &\text{Bird(Sparrow)} \Rightarrow \text{Flies(Sparrow)} \\
  &\text{Predates(Eagle, Sparrow)} \land \text{Bird(Sparrow)} \Rightarrow \text{Bird(Eagle)}
  \end{align*}
  \]

Joint probability: example 3

Computing joint probability

\[ p(x) = \frac{1}{Z} \exp(2w_1 + 4w_2) \]

- This possible world is most likely among all possible worlds
- The problem is that we did not encode constraints saying that:
  - A bird is not likely to be predator of itself
  - A prey is not likely to be predator of its predator
Hard constraints
Impossible worlds

• It is always possible to make certain worlds impossible by adding constraints with infinite weight
• Infinite weight constraints behave like pure logic formulas: any possible world has to satisfy them, otherwise it receives zero probability

Example

• Let’s add the infinite weight constraint:

  “Nobody can be a self-predator”

  \[ w_3 \quad \forall x \neg \text{Predates}(X, X) \]

  to the previous example

Hard constraint: example 3

\[
\begin{array}{c}
\text{Predates(Eagle, Eagle)}\quad \text{Predates(Eagle, Sparrow)} \quad \text{Predates(Sparrow, Eagle)} \\
\text{Bird(Eagle)} \quad \text{Predates(Sparrow, Sparrow)} \quad \text{Bird(Sparrow)} \\
\text{Flies(Eagle)} \quad \text{Flies(Sparrow)} \quad \text{Flies(Sparrow)}
\end{array}
\]

= true

Computing joint probability

\[
p(x) = \frac{1}{Z} \exp(2w_1 + 4w_2) = 0
\]

• The numerator does not contain \( w_3 \), as the no-self-predator constraint is never satisfied

• However the partition function \( Z \) sums over all possible worlds, including those in which the constraint is satisfied.

• As \( w_3 = \infty \), the partition function takes infinite value and the possible worlds gets zero probability.

Hard constraint: example 1

\[
\begin{array}{c}
\text{Predates(Eagle, Eagle)}\quad \text{Predates(Eagle, Sparrow)} \quad \text{Predates(Sparrow, Eagle)} \\
\text{Bird(Eagle)} \quad \text{Predates(Sparrow, Sparrow)} \quad \text{Bird(Sparrow)} \\
\text{Flies(Eagle)} \quad \text{Flies(Sparrow)} \quad \text{Flies(Sparrow)}
\end{array}
\]

= true
Computing joint probability

\[ p(x) = \frac{1}{Z} \exp(2w_1 + w_2 + 2w_3) \neq 0 \]

- The only non-zero probability possible worlds are those always satisfying hard constraints
- Infinite weight features cancel out between numerator and possible worlds at denominator which also satisfy the constraints, while those which do not become zero

Inference

Assumptions

- For simplicity of presentation, we will consider MLN in form:
  - function-free (only predicates)
  - clausal
- However the methods can be applied to other forms as well
- We will use general first-order logic form when describing applications

Inference

MPE inference

- One of the basic tasks consists of predicting the most probable state of the world given some evidence (the most probable explanation)
- The problem is a special case of MAP inference (maximum a posteriori inference), in which we are interested in the state of a subset of variables which do not necessarily include all those without evidence.

Inference

MPE inference in MLN

- MPE inference in MLN reduces to finding the truth assignment for variables (i.e. nodes) without evidence maximizing the weighted sum of satisfied clauses (i.e. features)
- The problem can be addressed with any weighted satisfiability solver
- MaxWalkSAT has been successfully used for MPE inference in MLN.
MaxWalkSAT

Description

• Weighted version of WalkSAT

• Stochastic local search algorithm:

  1. Pick an unsatisfied clause at random
  2. Flip the truth value of an atom in the clause

• The atom to flip is chosen in one of two possible ways with a certain probability:
  – randomly
  – in order to maximize the weighted sum of the clauses satisfied with the flip

• The stochastic behaviour (hopefully) allows to escape local minima

MaxWalkSAT pseudocode

1: procedure MaxWalkSAT(weighted_clauses, max_flips, max_tries, target, p)
2:    vars ← variables in weighted_clauses
3:    for i ← 1 to max_tries do
4:        soln ← a random truth assignment to vars
5:        cost ← sum of weights of unsatisfied clauses in soln
6:        for j ← 1 to max_flips do
7:            if cost ≤ target then
8:                return "Success, solution is", soln
9:            end if
10:            c ← a randomly chosen unsatisfied clause
11:            if Uniform(0,1) < p then
12:                v_f ← a randomly chosen variable from c
13:            else
14:                for all variable v in c do
15:                    compute DeltaCost(v)
16:                end for
17:                v_f ← v with lowest DeltaCost(v)
18:            end if
19:            soln ← soln with v_f flipped
20:            cost ← cost + DeltaCost(v_f)
21:        end for
22:    end for
23:    return "Failure, best assignment is", best soln found
24: end procedure

MaxWalkSAT

Ingredients

• target is the maximum cost considered acceptable for a solution

• max_tries is the number of walk restarts

• max_flips is the number of flips in a single walk

• p is the probability of flipping a random variable

• Uniform(0,1) picks a number uniformly at random from [0,1]

• DeltaCost(v) computes the change in cost obtained by flipping variable v in the current solution
Inference

Marginal and conditional probabilities

• Another basic inference task is that of computing the marginal probability that a formula holds, possibly given some evidence on the truth value of other formulas

• Exact inference in generic MLN is intractable (as it is for the generic MN obtained by the grounding)

• MCMC sampling techniques have been used as an approximate alternative

Inference

Constructing the ground MN

• In order to perform a specific inference task, it is not necessary in general to ground the whole network, as parts of it could have no influence on the computation of the desired probability

• Grounding only the needed part of the network can allow significant savings both in memory and in time to run the inference

Inference

Partial grounding: intuition

• A standard inference task is that of computing the probability that \( F_1 \) holds given that \( F_2 \) does.

• We will focus on the common simple case in which \( F_1, F_2 \) are conjunctions of ground literals:

1. All atoms in \( F_1 \) are added to the network one after the other

2. If an atom is also in \( F_2 \) (has evidence), nothing more is needed for it

3. Otherwise, its Markov blanket is added, and each atom in the blanket is checked in the same way

Partial grounding: pseudocode

1: \texttt{procedure CONSTRUCTNETWORK}(\( F_1, F_2, L, C \))
2: \texttt{inputs:}
3: \( F_1 \) a set of query ground atoms
4: \( F_2 \) a set of evidence ground atoms
5: \( L \) a Markov Logic Network
6: \( C \) a set of constants
7: \texttt{output:} \( M \) a ground Markov Network
8: \texttt{calls:} \( MB(q) \) the Markov blanket of \( q \) in \( ML,C \)
9: \begin{align*}
10: &G \leftarrow F_1 \\
11: \text{while } &F_1 \neq \emptyset \\
12: &\text{for all } q \in F_1 \text{ do} \\
13: &\quad \text{if } q \notin F_2 \text{ then} \\
14: &\quad \quad F_1 \leftarrow F_1 \cup (MB(q) \setminus G) \\
15: &\quad \quad G \leftarrow G \cup MB(q) \\
16: &\quad \text{end if} \\
17: &\quad F_1 \leftarrow F_1 \setminus \{q\} \\
18: &\text{end for} \\
19: &\text{end while} \\
20: &\text{return } M \text{ the ground MN composed of all nodes in } G \text{ and all arcs between them in } ML,C, \text{ with features and weights of the corresponding cliques}
21: \end{align*}
22: \texttt{end procedure}
**Inference**

**Gibbs sampling**

- Inference in the partial ground network is done by Gibbs sampling.
- The basic step consists of sampling a ground atom given its Markov blanket.
- The probability of $X_l$ given that its Markov blanket has state $B_l = b_l$ is:
  
  $$p(X_l = x_l | B_l = b_l) = \frac{\exp \sum_{f_i \in F_l} w_i f_i(X_l = x_l, B_l = b_l)}{\exp \sum_{f_i \in F_l} w_i f_i(X_l = 0, B_l = b_l) + \exp \sum_{f_i \in F_l} w_i f_i(X_l = 1, B_l = b_l)}$$

  where:
  
  - $F_l$ is the set of ground formulas containing $X_l$
  - $f_i(X_l = x_l, B_l = b_l)$ is the truth value of the $i$th formula when $X_l = x_l$ and $B_l = b_l$

- The probability of the conjunction of literals is the fraction of samples (at chain convergence) in which all literals are true.

**Inference**

**Multimodal distributions**

- As the distribution is likely to have many modes, multiple independently initialized chains are run.
- Efficiency in modeling the multimodal distribution can be obtained starting each chain from a mode reached using MaxWalkSAT.

**Inference**

**Handling hard constraints**

- Hard constraints break the space of possible worlds into separate regions.
- This violate the MCMC assumption of reachability.
- Very strong constraints create areas of very low probability difficult to traverse.
- The problem can be addressed by *slice sampling* MCMC, a technique aimed at sampling from slices of the distribution with a frequency proportional to the probability of the slice.

**Learning**

**Maximum likelihood parameter estimation**

- Parameter estimation amounts at learning weights of formulas.
- We can learn weights from training examples as possible worlds.
- Let’s consider a single possible world as training example, made of:
  
  - a set of constants $C$ defining a specific MN from the MLN
  - a truth value for each ground atom in the resulting MN
- We usually make a *closed world* assumption, where we only specify the true ground atoms, while all others are assumed to be false.
- As all groundings of the same formula will share the same weight, learning can be also done on a single possible world.
Learning

Maximum likelihood parameter estimation

• Weights of formulas can be learned maximizing the likelihood of the possible world:

\[ w^\text{max} = \text{argmax}_w p_w(x) = \text{argmax}_w \frac{1}{Z} \exp \left( \sum_{i=1}^{F} w_i n_i(x) \right) \]

• As usual we will equivalently maximize the log-likelihood:

\[ \log(p_w(x)) = \sum_{i=1}^{F} w_i n_i(x) - \log(Z) \]

Priors

• In order to combat overfitting Gaussian priors can be added to the weights as usual (see CRF)

Learning

Maximum likelihood parameter estimation

• The gradient of the log-likelihood wrt weights becomes:

\[ \frac{\partial}{\partial w_i} \log p_w(x) = n_i(x) - \sum_{x'} p_w(x') n_i(x') \]

where the sum is over all possible worlds \( x' \), i.e. all possible truth assignments for ground atoms in the MN

• Note that \( p_w(x') \) is computed using the current parameter values \( w \)

• The \( i \)-th component of the gradient is the difference between number of true grounding of the \( i \)-th formula, and its expectation according to the current model

Applications

Entity resolution

• Determine which observations (e.g. noun phrases in texts) correspond to the same real-world object

• Typically addressed creating feature vectors for pairs of occurrences, and training a classifier to predict whether they match

• The pairwise approach doesn’t model information on multiple related objects (e.g. if two bibliographic entries correspond to the same paper, the authors are also the same)

• Some implications hold only with a certain probability (e.g. if two authors in two bibliographic entries are the same, the entries are more likely to refer to the same paper)
Applications

MLN for entity resolution

- MLN can be used to address entity resolution tasks by:
  - not assuming that distinct names correspond to distinct objects
  - adding an equality predicate and its axioms: reflexivity, symmetry, transitivity

- Implications related to the equality predicate can be:
  - grounding of a predicate with equal constants have same truth value
  - constants appearing in a ground predicate with equal constants are equal (i.e. the “same paper → same author” implication, which holds only probabilistically in general)

Applications

MLN for entity resolution

- Weights for different instances of such axioms can be learned from data
- Inference is performed adding evidence on entity properties and relations, and querying for equality atoms
- The network performs collective entity resolution, as the most probable resolution for all entities is jointly produced

Entity resolution

Entity resolution in citation databases

- Each citation has: author, title, venue fields.
- Citation to field relations:
  
  \[
  \text{Author(bib,author)} \quad \text{Title(bib,title)} \\
  \text{Venue(bib,venue)}
  \]

- field content relations:
  
  \[
  \text{HasWord(author,word)} \quad \text{HasWord(title,word)} \\
  \text{HasWord(venue,word)}
  \]

- equivalence relations:
  
  \[
  \text{SameAuthor(author1,author2)} \\
  \text{SameTitle(title1,title2)} \\
  \text{SameVenue(venue1,venue2)} \\
  \text{SameBib(bib1,bib2)}
  \]
Entity resolution

Same words imply same entity

• E.g.:

\[ \text{Title}(b_1,t_1) \land \text{Title}(b_2,t_2) \land \text{HasWord}(t_1, +w) \land \text{HasWord}(t_2, +w) \Rightarrow \text{SameBib}(b_1,b_2) \]

• here the ‘+’ operator is a template: a rule is generated for each constant of the appropriate type (i.e. words)

• a separate weight is learned for separate words (e.g. stopwords like articles or prepositions are probably less informative than other words)

Transitivity

• E.g.:

\[ \text{SameBib}(b_1,b_2) \land \text{SameBib}(b_2,b_3) \Rightarrow \text{SameBib}(b_1,b_3) \]

Entity resolution

Transitivity across entities

• E.g.:

\[ \text{Author}(b_1,a_1) \land \text{Author}(b_2,a_2) \land \text{SameBib}(b_1,b_2) \Rightarrow \text{SameAuthor}(a_1,a_2) \land \text{Author}(b_1,a_1) \]

\[ \land \text{Author}(b_2,a_2) \land \text{SameAuthor}(a_1,a_2) \Rightarrow \text{SameBib}(b_1,b_2) \]

• The second rule is not a valid logic rule, but holds probabilistically (citations with same authors are more likely to be the same)

Resources

References


Software

• The open source Alchemy system provides an implementation of MLN, with example networks for a number of tasks:

  – http://alchemy.cs.washington.edu/