Using Probabilistic Logic Programming to Find Patterns

Theofrastos Mantadelis
theo.mantadelis@dcc.fc.up.pt

Ricardo Rocha
ricroc@dcc.fc.up.pt

Jorge Oliveira
oliveira_jorge@dcc.fc.up.pt

Miguel Tavares Coimbra
mcoimbra@dcc.fc.up.pt

CRACS & INESC TEC
Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre 1021/1055, 4169-007 Porto, Portugal

Instituto de Telecomunicações
Faculdade de Ciências da Universidade do Porto
Rua do Campo Alegre 1021/1055, 4169-007 Porto, Portugal

Abstract

This short paper, briefly presents the probabilistic logic programming language ProbLog and the system MetaProbLog. We present an example Hidden Markov Model to illustrate the three main tasks of the system. Furthermore, we mention some of the existing ProbLog applications which are used to find connections/patterns in relational databases. Finally, we present an application that uses MetaProbLog for phonocardiogram classification which is used in order to diagnose heart diseases.

1 Introduction

Probabilistic Logic Programming (PLP) combines technologies from logic programming, knowledge representation and reasoning and machine learning. Probabilistic models in our days are principled and a widely used approach to deal with uncertainty. First order logic can elegantly represent complex situations involving a variety of objects as well as relations among the objects.

MetaProbLog[6] is a framework of the ProbLog[3][5] probabilistic logic programming language. ProbLog extend Prolog programs by annotating facts with probabilities. In that way it defines a probability distribution over all Prolog programs. ProbLog follows the distribution semantics presented by Sato[2]. MetaProbLog extends the semantics of ProbLog by defining a "ProbLog engine" which permits the definitions of probabilistic meta calls[6]. MetaProbLog inference, currently allows the computation of marginal probabilities with or without evidence. Furthermore, it allows the computation of marginal probabilities for the answers of non-ground queries.

MetaProbLog has three primary inference methods: exact inference, program sampling and most probable explanation. The exact inference method, uses state of the art knowledge compilation methods[2]; program sampling, is a rejection sampling approach; and finally, the most probable explanation inference uses a dynamic algorithm to find the most probable explanation of a query.

2 Semantics

A ProbLog program T consists of a set of facts annotated with probabilities \( p_f \) – called probabilistic facts – together with a set of standard definite clauses \( h : = b_1, \ldots, b_n \), that can have positive and negative probabilistic literals in their body. A probabilistic fact \( p_f \) is true with probability \( p_f \). These facts correspond to random variables, which are assumed to be mutually independent. Together, they thus define a distribution over subsets of \( L_T = \{p_f_1, \ldots, p_f_n\} \). The definite clauses add arbitrary background knowledge (BK) to those sets of logical facts. To keep a natural interpretation of a ProbLog program we assume that probabilistic facts cannot unify with other probabilistic facts or with the background knowledge rule heads. Formally, a ProbLog program is of the form \( T = \{p_f_1, \ldots, p_f_n\} \cup BK \).

Given the one-to-one mapping between ground definite clause programs and Herbrand interpretations, a ProbLog program defines a distribution over its Herbrand interpretations.

The distribution semantics are defined by generalising the least Herbrand models of the clauses by including subsets of the probabilistic facts. If fact \( p_f \) is annotated with \( p_f \), \( p_f \) is included in a generalised least Herbrand model with probability \( p_f \) and left out with probability \( 1 - p_f \).

The different facts are assumed to be probabilistically independent, however, negative probabilistic facts in clause bodies allow the user to enforce a choice between two clauses.

As such, a ProbLog program specifies a probability distribution over all its possible non-probabilistic subprograms. The success probability of a query is defined as the probability that the query succeeds in such all its possible non-probabilistic subprograms. The success probability of a random subprogram. ProbLog follows the distribution semantics[9] proposed by Sato.

3 Example Program & Queries

The syntax of MetaProbLog uses logic programming, specifically Prolog, in order to be very expressive as a language and be able to describe complex models. Next we present a small MetaProbLog program that defines the Hidden Markov Model illustrated at Figure 1. The different colors indicate the possible transitions from each state which are modeled by annotated disjunctions[6] in the program.

Below follows the MetaProbLog program that models the Hidden Markov Model of Figure 1.

```
0.80::trans(s1,s2,T1,T2);
0.20::trans(s1,noise,T1,T2) <- next(T1, T2).

0.80::trans(s2,s1,T1,T2));
0.20::trans(s2,noise,T1,T2) <- next(T1, T2).

0.40::trans(noise,s1,T1,T2);
0.40::trans(noise,s2,T1,T2);
0.20::trans(noise,noise,T1,T2) <- next(T1, T2).

0.20::start(noise,0);
0.40::start(s1,0);
0.40::start(s2,0) <- true.

signal(State, 0) :- start(State, 0).

signal(State2, T2) :-
  trans(State1,State2,T1,T2),
  signal(State1, T1).

next(T1, T2) :- integer(T1), !, T2 is T1 + 1.
next(T1, T2) :- integer(T2), T1 is T2 - 1.
```

For our example we use the term `trans/4` to describe a transition of the model from one state to another (first and second argument of the

![Figure 1: The graphical representation of a 3 state Hidden Markov Model](image-url)

1MetaProbLog’s website [www.dcc.fc.up.pt/metaproblog](http://www.dcc.fc.up.pt/metaproblog)

2ProbLog’s syntax: Pl::ChoiceC :- ; Pl::ChoiceC <- Body, is used to model exclusive choices with \( \sum_{P_1} \cdot P_1 = 1.0 \). This construct is called annotated disjunction and two choices of an annotated disjunction can never be true at the same time.
ProbLog systems have found applications in many fields with most common examples to include:

- Link discovery in Bioinformatic Alzheimer database [11]. Biomime Alzheimer database is a real-world biological dataset of Alzheimer genes which corresponds to a directed probabilistic graph of 11530 edges and 5220 nodes. ProbLog was used to discover relations among genes and other biological properties [3].
- WebKB (http://www.cs.cmu.edu/~webkb) is a dataset from a collective classification domain in which university webpages are classified according to their textual content. ProbLog has been used to learn the probabilities that two webpages are related and to query the WebKB [4].
- The probabilistic Dictionary [12] is used to discover the probability that two words have the same meaning. It includes around 250 different words from the English language and meanings for about 30 of them. Some words are related together according to their semantic relatedness. This relation is marked with the probability that the two words have the same meaning.
- ProbLog has also been used for robotic affordance model learning by [8]. In this application ProbLog was successfully used to learn a robotic task with multiple objects and complex spatial relations.
- Finally, ProbLog has been used to model Mobile Ad hoc Networks and analyse Fadip [7], a Publish/Subscribe protocol for Mobile Ad hoc Networks. ProbLog efficiently calculates the probability that a message would be transmitted from one device to another in the network, analysing statistics for the traffic and reachability of the protocol.

Lately, the classification of phonocardiogram (PCG) signals has got significant attention in the academic community [1]. Classifying PCGs is both a challenging and an important task. Heart sounds are non-trivial signals, since they might contain non-stationary noise, have artifacts and murmurs sounds. Heart sound auscultation techniques is one of the most reliable and successful tools in early diagnosis used for potentially deadly heart diseases, such as natural and prosthetic heart valve dysfunction or even in heart failure. Therefore a computer-aided auscultation may allow detection of diseases that are hardly recognized through the traditional methods, for instance ischemic heart disease.

Recently, HMMs have been used for modeling and characterizing real-world signals such as heart sound signals [10]. For future work, we aim to model PCG signals as an HMM and use MetaProbLog to find the most likely sequence of events (S1, S2, S3, S4, noise, murmur, etc.) and finally, use our model in order to characterize real life segmented signals.

Acknowledgements

This work is funded by the Instituto de Telecomunicações in the scope of Project Rheumus (Projeto QREN no: 38505) and the Fundação para a Ciência e a Tecnologia (FCT) (Portuguese Foundation for Science and Technology) within the project UID/EEA/50014/2013.

References