International Journal of Approximate Reasoning 93 (2018) 561–577

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Learning to rank in PRISM ☆

Ryosuke Kojima [a,*], Taisuke Sato [b]

[a] Department of Biomedical Data Intelligence, Graduate School of Medicine, Kyoto University, Kyoto, Japan
[b] AI research center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

ARTICLE INFO

Article history:
Received 1 June 2017
Received in revised form 5 October 2017
Accepted 20 November 2017
Available online 2 December 2017

Keywords:
PRISM
Probabilistic logic programming language
Learning to rank
Ranking
Preference learning

ABSTRACT

Learning parameters associated with propositions is one of the main tasks of probabilistic logic programming (PLP), and learning algorithms for PLP have been primarily developed based on maximum likelihood estimation or the optimization of discriminative criteria. This paper explores yet another innovative approach to parameter learning, learning to rank or rank learning, that has been studied mainly in the field of preference learning. We combine learning to rank with techniques developed in PLP to make the latter applicable to a variety of ranking problems such as information retrieval. We implement our approach in PRISM, a PLP system based on the distribution semantics. It supports many parameter learning algorithms such as the expectation maximization algorithm, the variational Bayes algorithm and an algorithm for Viterbi training efficiently by mapping them onto a single data structure called explanation graph. To ensure the same efficiency for parameter learning by learning to rank as in the current PRISM, we introduce a gradient-based learning method that takes advantage of dynamic programming on the explanation graph. This paper also presents three experimental results. The first one is with synthetic data to check the learning behaviors of the proposed approach. The second one uses a knowledge base (knowledge graph) and apply rank learning to a DistMult model for the task of deciding whether relations over entities exist or not. The last one tackles the problem of parsing by a probabilistic context free grammar whose parameters are learned from a tree corpus by rank learning. These experiments successfully demonstrated the potential and effectiveness of learning to rank in PLP. We plan to release a new version of PRISM augmented with the ability of learning to rank in the near future.

1. Introduction

One of the main roles of *probabilistic logic programming (PLP)* is to provide a high-level interface for statistical modeling based on the flexible and expressive framework of logic programming. Two core elements in statistical modeling are learning parameters of models and probabilistic inference using learned models. Existing PLP systems such as Alchemy [15] (an implementation of Markov logic networks [18]), ProbLog [11], ProPPR [31], and PRISM [22] all provide these elements and achieve flexible modeling supported by efficient computation for the parameter learning and probabilistic inference. Traditional parameter learning in PLP has been realized by optimizing data fitting criteria be they discriminative or generative, and discriminative models which usually show better performance than generative counterparts require positive and

☆ This paper is part of the Virtual special issue on Probabilistic Logic Programming 2016, edited by James Cussens and Arjen Hommersom.

* Corresponding author.
E-mail address: kojima.ryosuke.8e@kyoto-u.ac.jp (R. Kojima).

https://doi.org/10.1016/j.ijar.2017.11.011
0888-613X/

negative examples to learn parameters. Learned parameters are used to judge whether given data is positive or negative. However, in real world applications such as document retrieval, judging dichotomically whether a retrieved document is positive (correct) or negative (wrong) is difficult or even harmful. In such situations, giving preference by a *preference pair*, representing which one is preferred in the pair, is an effective alternative. An approach to learning parameters from a set of preference pairs is called (pair-wise) *"learning to rank"* in the field of preference modeling. Learning to rank is frequently used together with *"ranking"* entities, inference of the ranks of entities, in a plethora of information retrieval applications including collaborative filtering, question answering, multimedia retrieval, text summarization, and online advertising [16]. To support these broad applications, a variety of approaches for preference modeling have been developed [7,6,2,35,16] and we can choose a model most suitable for applications and data to use. We contribute to PLP by extending the scope of PLP to preference modeling with rank learning. Also, at the same time, we contribute to the development of ranking applications by providing an expressive logic-based framework for preference modeling. In the traditional preference learning, relationships such as is-a, part-of and transitive closure between properties and those between objects are not necessarily taken into account. In the logic based approach, they are naturally handled by logical expressions built up from predicates and variables.

Every PLP language requires a semantics to connect probability with a logic programming language. The distribution semantics [20] is one of the most prominent semantics in PLP. For example, PRISM [22], ICL [17], ProbLog [4] and LPAD [30] are based on this semantics. This paper focuses on PRISM which realizes a variety of probabilistic modeling techniques based on a single data structure called explanation graph. It compactly encodes all possible proof trees for a query. In PRISM, using dynamic programming on explanation graphs, a broad range of machine learning algorithms from generative/discriminative parameter learning to Viterbi inference to Markov chain Monte Carlo (MCMC) is efficiently implemented under the distribution semantics. In this paper we introduce a mechanism of learning to rank or rank learning to PRISM, thereby merging preference modeling with logic-based probabilistic modeling. We also demonstrate two applications with real data. One is link prediction in a knowledge base (knowledge graph) and the other probabilistic parsing for a tree corpus of Japanese sentences. In traditional approaches, these tasks are handled with different notations and different data structure. For example, a knowledge graph is often represented by as a set of triples encoding relations between subjects and objects while a tree corpus of sentences is usually represented as a set of parse trees. Our logic-based approach uniformly represent them in terms of logical expressions.

The most closely related work to our work is ProPPR [31], a PLP system for question answering. Given a query `q(a,X)`, it returns an answer substitution for the variable `X` using a probabilistic logic programming framework based on features. Roughly speaking, learning data in ProPPR is a set of triplets `(q(a,X),q(a,right_answer),q(a,wrong_answer))` specifying that the `right_answer` is preferred to the `wrong_answer` as a substitution for `X` in a query `q(a,X)`. Parameters (weights) associated with features that determine right/wrong are learned by a sophisticated learning algorithm applied to SLD proof trees for `q(a,X)` obtained by sampling. The target of ProPPR is to solve such question answering and preference learning is specialized to this purpose. Since ProPPR adopts a specific parameterization different from parameterization suitable for the distribution semantics, it is hard to directly compare with PRISM at semantic level or implementation level. We may say however that unlike ProPPR, PRISM is a general programming language supporting multiple machine learning methods from conventional ones such as the expectation maximization (EM) algorithm [21], the variational Bayes (VB) algorithm [23], an algorithm for Viterbi training (VT) [24] and MCMC [19] to a non-conventional one, i.e., parameter learning by learning to rank. Our contributions thus include the introduction of a natural extension of conventional parameterization suitable for learning to rank and the integration of learning to rank with the other conventional parameter learning frameworks into one system.

Now we explain what is targeted in this paper. Ranking entities can be achieved by sorting entities in the order of their scores computed by a scoring function. The scoring function has parameters and in learning to rank, those parameters are trained by ordered data, typically given as a set of preference pairs. Our fundamental idea is to equate an entity with a Prolog's goal, i.e., a ground atom, and a scoring function with a (log) probability of the goal. For example, if `goal(a1)` is considered better than `goal(a2)`, a higher probability would be assigned to `goal(a1)` than the probability assigned to `goal(a2)`. We prefer to only impose constraints on the order of ground atoms rather than forcibly bring the probability of positive examples closer to one while negative ones closer to zero. We embody this idea in PRISM.

The first thing to do is to tackle a problem of learning parameters by learning to rank from training data consisting of a set of lists of ranked goals. For example, given `[[goal(a1),goal(a2)], [goal(a2),goal(a3)]]`, using a model written as a PRISM program, we have to learn parameters so that a set of conditions `{P(goal(a1))>P(goal(a2)), P(goal(a2))>P(goal(a3))}` are satisfied.

Note that this setting is different from the ProPPR's learning setting mentioned above. ProPPR is designed to learn the preference of substitutions for a query from triplets of the form `(q(a,X),q(a,right_answer),q(a,wrong_answer))`. In contrast, PRISM learns parameters from pairs of propositions (ground atoms) and they can have different predicates; thus, a list `[q1(a,right_answer), q2(a,wrong_answer)]` means that the `q1(a,right_answer)` is preferred to the `q2(a,wrong_answer)`. Since this parameter learning setting is compatible with the usual learning setting of PRISM (Section 4), our rank learning is applicable to any generative model written in PRISM. Later we will show examples of generative modeling with learning to rank using a hidden Markov model (HMM) and a probabilistic context-free grammar (PCFG) in Section 4 and Section 5.