Human-in-the-loop Learning for Probabilistic Programming

Srirama Natarajan
University of Texas-Dallas

Phillip Odom
Georgia Institute of Technology

Tushar Khot
Allen Institute for AI

Kristian Kersting
TU Dortmund

Jude Shavlik
University of Wisconsin-Madison

Abstract
We present our BoostSRL system, a Java-based learning system that inductively learns probabilistic logic clauses from data. Our system is capable of learning different types of models, handling modeling of hidden data, learning with preferences from humans, scaling with large amounts of data by approximate counting and modeling temporal data. We review these capabilities briefly in this short paper.

1 BoostSRL system
Most learning methods inside Probabilistic Programming and the related Statistical Relational AI communities have focused on learning the parameters (probabilities/weights) given the program and the data. Our gradient boosted approach, presented in Figure 1 [9, 10], instead relies on the intuition that learning a set of weak partial programs (similar to ensemble methods in classical machine learning), and learns both the rules/programs and the parameters of the rules simultaneously. As shown in the figure, at a fairly high-level, the algorithm proceeds as follows: during every iteration, the gradients are computed for every example and these gradients become the regression values on the examples, a new logical (i.e., parameterized) regression tree is learned that fit these examples, the tree is added to the model and the process is repeated. Table 1 shows several rules from a probabilistic program learned for predicting the winner of a football game based on yards earned by each team, the number of turnovers as well as the season the games were played.

1.1 Multiple models:
Our framework and the system can learn different types of probabilistic programming models. It can learn Markov Logic Networks [2], undirected graphical models that employs the use of weighted logic clauses. It can learn Relational Dependency networks [10], cyclic lifted graphical models and can learn with hidden data [3]. In addition, this was employed in the context of sequential decision-making, imitation learning, and has been used to learn relational policies [7]. Finally, we extended our algorithm to learn over continuous time [17], the first of its kind in logical models.
1.2 Multiple distributions:

Our system can learn different types of distributions - multinomials, exponential, Gaussian and Dirichlet distributions. We are currently extending the system to learn fully hybrid relational models.

1.3 Learning with human advice:

Most research inside probabilistic logic models either employed the full model and performed inference or learned only the parameters given the rules themselves. Our previous work on boosting instead learned the rules from the data. In essence, in our prior work, the humans were restricted to be "mere experts". Consequently, we have extended our system to include human advice in the form of preferences [12], precision vs recall tradeoff [16], qualitative constraints [14] and more recently privileged information [14].

1.4 Closing the loop:

While BoostSRL system can take human advice as input, the input is typically taken before learning occurs. Inspired by active learning, we have extended the system to solicit advice as needed. It explicitly computes the uncertainty in its model and solicits advice from the expert as needed [13]. Our BoostSRL system is capable of active advice seeking for the different types of logical models specified above.

1.5 Scaling:

Modern challenges require systems capable of handling large-scale data. The BoostSRL system has has been implemented on top of a relational database [4]. Malec et al. [4] demonstrate the efficiency gains of combining databases with effective background knowledge in inductive logic programming. Das et al. [1] introduce an efficient approximate counting technique which can be used as a part of our BoostSRL system when learning MLNs by converting the partial programs into a graph. We have recently generalized this work to now learn hypergraphs that allows for effectively learning multi-predicates in an efficient manner.

1.6 Applications:

As part of the BoostSRL system, we have developed a NLP pipeline that has been employed for relation extraction previously [5, 11, 15]. The key aspect of this pipeline is that it reads text documents as inputs, runs it through Stanford NLP pipeline, creates corresponding logic facts and rule base thus making it ready for learning logical models. This pipeline has been used in combination with BoostSRL to identify adverse drug effects from PubMed abstracts [5] and knowledge base relations [15].

Medical applications: We used the BoostSRL system to classify patients into normal, mild cognitive impairment (onset of Alzheimer’s that is hard to detect) and Alzheimer’s using relational features from MRI scans [6]. We show part of the program in Table 2, where the system has identified special regions and features of the brain associated with Alzheimer’s. We have also used our system to predict coronary artery calcification levels based on historical data [8].

2 Conclusion

BoostSRL is a versatile system that is capable of learning different types of models of multiple distributions. Its ability to learn with different types of human knowledge and effectively scale up to large data enable it to be applicable to many challenging tasks.

3 Acknowledgements

The authors acknowledge the contributions of the other students and collaborators in the lab who have developed this system. In no particular order, we thank, Gautam Kunapuli, Ameet Soni, Shuo Yang, Mayukh Das, Devendra Dhami, Nandini Ramanan, Srijita Das, Navdeep Kaur, Alex Hayes, Kaushik Roy, Dileep Viswanathan, Anurag Wazalwar, Vishal Bangera, and Rakesh Kumaraswamy. We also acknowledge the support of DARPA, AFOSR, NSF, AFRL and ARO for their support. Kristian Kersting acknowledges the support by the German Science Foundation project AIJCAML: Argumentative Machine Learning (FRI1686/3-1) as part of the SPP 1999 (RATIO). Finally, we acknowledge the contributions of all the co-authors of the papers that have significantly improved the applicability of the system.

References


<table>
<thead>
<tr>
<th>RULE</th>
<th>WEIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>alz(B) ← CSF(B, Left parahippocampal gyrus) &gt; 0.19, CSF(B, Lobule IX of vermis) &gt; 0.13</td>
<td>0.57</td>
</tr>
<tr>
<td>alz(B) ← CSF(B, Left parahippocampal gyrus) &gt; 0.19, CSF(B, Lobule IX of vermis) ≤ 0.13, CSF(B, Left hippocampus) &gt; 0.37</td>
<td>0.70</td>
</tr>
<tr>
<td>alz(B) ← CSF(B, Left parahippocampal gyrus) ≤ 0.19, CSF(B, Right superior frontal gyrus) ≤ 0.25, GM(B, Left hippocampus) ≤ 0.56</td>
<td>0.68</td>
</tr>
</tbody>
</table>
Human-in-the-loop Learning for Probabilistic Programming


