Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches: Papers from the 2015 AAAI Spring Symposium

Learning Probabilistic Logic Models with Human Advice

Phillip Odom
Department of Computer Science
Indiana University Bloomington

Tushar Khot *
Department of Computer Science
University of Madison-Wisconsin

Sriraam Natarajan
School of Informatics and Computing
Indiana University Bloomington

Abstract

We consider the problem of interactive machine learning for rich, structured and noisy domains. We present a recently successful learning algorithm and provide several extensions for incorporating rich, high level, human feedback. We then discuss some open problems in this area.

Introduction

Research in Artificial Intelligence (AI) has always pursued two paths - symbolic AI where the focus is on rich representations and inferences over groups of objects and statistical AI where the focus has been on using data to develop robust models in the presence of uncertainty. Recently, there is a push towards combining the benefits of the two distinct approaches in developing a research field called Statistical Relational AI (StaRAI) that uses symbolic representations to model the underlying rich structure of the domain and employs statistical techniques to model the uncertainty and noise present in the data. At the core of StaRAI is Statistical Relational Learning (SRL) (Getoor and Taskar 2007) that learns rich representations (first-order logic (FOL)) using statistical techniques (primarily probabilistic methods).

While these models are expressive, learning (i.e., model selection) in complex, relational domains is a hard task. Consequently, there is a lot of focus on this problem, primarily in learning the qualitative structure (FOL rules) (Kok and Domingos 2009; 2010). This is due to the fact that as with propositional graphical models, every step of learning the structure involves repeated parameter learning which in turn involves probabilistic inference. To deal with this issue, recently a technique based on the successful gradient boosting technique called Relational Functional-Gradient Boosting (RFGB) (Natarajan et al. 2012; 2011; Khot et al. 2011) was adapted for probabilistic logic models (PLMs) that learns structure and parameters simultaneously.

In this work-in-progress paper, we review this body of work and present the overarching framework for learning PLMs efficiently. The key idea behind this work is to turn the problem of model learning into a series of relational regression models learned in a stage-wise manner. We show that while this work is indeed effective in learning large set of small (probabilistic) rules, there is an over-reliance on purely the data for inducing these models. One of the key attractive features of symbolic representations such as FOL is that these representations make it natural for humans to provide advice and interact with the system. However, unlike both purely statistical (Fung, Mangasarian, and Shavlik 2002; Towell and Shavlik 1994) and purely symbolic (Baffes and Mooney 1996) learning approaches, there is not much research on developing SRL approaches that can seamlessly interact with and learn from a human expert.

We present a series of methods that adapt RFGB to accept and exploit advice from a domain expert. Our first approach is to explicitly model the trade-off between false positives and false negatives by modifying the objective function. Note that class imbalance is an important and pertinent issue in many real tasks as most relations such as friends, co-authors, marriedto, and neighbors are false between most of the groundings of the variables. While this is effective in learning with such imbalanced data, the expert is restricted to provide only the cost of the trade-off. So we next extend the framework to handle richer advice such as label preferences (or action preferences in the case of sequential decision making tasks). We show that a similar objective function can be used to learn from such rich advice that are specified as FOL clauses to the learning system.

As a final contribution of the paper, we present several possible future extensions of this direction. Specifically, we note that while providing label preferences is indeed effective, the method may not be fully efficient. This is due to the fact that the expert might be required to provide all the advice upfront to the learning algorithm, potentially requiring significant effort. To mitigate this problem, we outline a high-level approach based on active learning so that the system solicits advice from the expert as necessary instead of requiring all the advice before learning. We then present another extension that allows for refinement of human advice. We also discuss the importance of understanding the trade-off between data and advice.

Relational Functional-Gradient Boosting

The standard gradient descent learning algorithm starts with initial parameters θ_0 and computes the gradient of the log-likelihood function ($\Delta_1 = \frac{\partial}{\partial\theta} \log P(\mathbf{X}; \theta_0)$). Friedman

*Currently at Allen AI

(2001) proposed an alternate approach to perform gradient descent where the log-likelihood function is represented using a regression function ψ over the examples $\mathbf{x}$ and the gradients are performed with respect to $\psi(x)$.

Functional gradient descent starts with an initial function ψ_0 and iteratively adds gradients Δ_m. Each gradient term (Δ_m) is a regression function over the training examples and the gradients at the m^{th} iteration can be represented as $\langle x_i, \Delta_m(x_i)\rangle$ where $x_i \in$ training examples. Also rather than directly using $\langle x_i, \Delta_m(x_i)\rangle$ as the gradient function, functional gradient boosting *generalizes* by fitting a regression function $\hat{\psi}_m$ (generally regression trees) to the gradients Δ_m. The final model $\psi_m = \psi_0 + \hat{\psi}_1 + \cdots + \hat{\psi}_m$ is a sum over these regression trees.

This method has been extended to various relational models for learning the structure (Natarajan et al. 2012; Karwath, Kersting, and Landwehr 2008; Kersting and Driessens 2008; Natarajan et al. 2011). The examples are ground atoms of the target predicate such as `workedUnder`(x, y). The ψ function is represented using relational regression trees (RRT) (Blockeel 1999). But since these are relational models, the ψ function depends on all the ground atoms and not just the grounding of the target predicate. For example, the probability function used by Natarajan et al. (2012) to learn the structure of Relational Dependency Networks was: $P(x_i)$= $sigmoid(\psi(x_i; Pa(x_i)))$ where $Pa(x_i)$ are all the relational/first-order logic facts that are used in the RRTs learned for x_i. They showed the functional gradient of likelihood for RDNs as

$$\frac{\partial log P(\mathbf{X} = \mathbf{x})}{\partial \psi(x_i)} = I(y_i = 1) - P(y_i = 1; x_i, Pa(x_i)) \quad (1)$$

which is the difference between the true distribution (I is the indicator function) and the current predicted distribution. For positive examples, the gradient is always positive and pushes the ψ function value ($\psi_0+\Delta_1+\cdots+\Delta_m$) closer to ∞ and the probability value closer to 1. Similarly the gradient for negative examples is negative and hence the ψ function is pushed closer to $-\infty$ and probability closer to 0.

Relational Advice Framework

Incorporating advice in propositional domains to improve the learning process has been explored in several directions (Kunapuli et al. 2013; 2010; Towell and Shavlik 1994). Commonly, in all these methods, a single piece of advice is defined over some set of the ground feature or example space. Traditionally advice in SRL methods have not been expressive and mainly consisted of hand-coding the structure and possibly even the parameters of the model. While such techniques have been successful, the learning algorithm does not modify the structure of the model. As a result, they do not introduce potentially novel interactions based on the training data.

A recent approach to handle expert advice in SRL models (Yang et al. 2014) used expert-specified costs for false negatives and false positives to learn relational models using RFGB. To this effect, inspired by regularization approaches in log-linear models (Gimpel and Smith 2010) this approach added a penalty term in the objective function that allowed trading-off between the false positives and false negatives. Gradients were derived based on this objective function that looked similar to the ones presented above except for allowing an extra term [1] to explicitly model the trade-off. Yang et al (2014) showed that this term reduced the need for subsampling negatives and instead learned from the full imbalanced data set.

While effective, this approach is limited in the scope of the advice it can handle. Another relational method (Natarajan et al. 2013) used learned models from a source domain as initial models in the target domain and boosted the gradients based on examples from target domain. The same approach can be used to handle expert advice, but if the training data is sub-optimal, it is possible that the learning algorithm will refine the model away from the advice. We instead seek to use the advice throughout the learning process and thereby handle noisy examples.

Following earlier work, we consider human interaction through expert-specified horn clauses. As opposed to treating advice as an initial model, our framework aims to use the advice throughout the learning process leading to a more accurate, robust model. We first introduce our first-order logic advice and then discuss our advice-learning framework.

Relational Advice

Our relational advice has the advantage of being generalized (as against targeting a specific example), described using parameterized properties (as against example identifiers), preferential (i.e., must avoid quantifying the importance of some of these properties as these numbers are hard to be elucidated by the expert) and in the same representation as that of the learning algorithm.

To achieve this, the advice we consider consists of two parts:

- horn clauses in the form $\wedge_i f_i(\mathbf{x_i}) \Rightarrow label(\mathbf{x_e})$, where $\wedge_i f_i(\mathbf{x_i})$ specifies the conjunction of conditions under which the advice applies on the example arguments $\mathbf{x_e}$.
- Sets of preferred (l_+) and non-preferred (l_-) labels

Given the label preferences, the goal is to learn a model that has higher probabilities for the preferred labels as compared to the avoided labels , ie $\forall s_i \in \mathbf{s}, P(l_+(s_i)) \geq P(l_-(s_i))$ where $\mathbf{s}$ is the set of training examples to which the advice applies. Thus our advice corresponds to IF-THEN rules where IF consist of some relational features and THEN consist of a preference over the target labels. Similar rules have been found to be more easily understood by human experts (Kulhmann et al. 2004).

Consider the problem of predicting whether an actor has worked under a director. A potential advice would be that if an actor, a, has acted in a movie which was directed by director, d, then it is likely that a has worked under d. This advice could be represented by the following rule:

Example 1. $actor(a) \wedge director(d) \wedge movie(m, a) \wedge movie(m, d) \Rightarrow label(a, d)$, $l_+ = workedUnder$ *and* $l_- = \neg workedUnder$.

[1] This term was $I - \lambda P$ instead of $I - P$ in the original RFGB

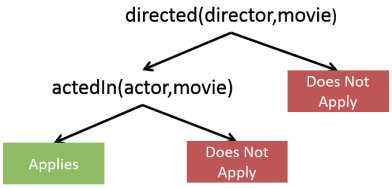

Figure 1: An advice model in the IMDB domain for predicting if an actor has worked under a director. Each node is a relational condition and leaf nodes identify whether or not advice will apply.

Resulting in, for any actor a and director d that appear in the same movie, $P(workedunder(a,d)) \geq P(\neg workedunder(a,d))$.

Figure 1 shows the relational advice constraints as a relational tree for a similar example. Advice applies to examples which reach the green node (not the red node).

Objective Function

As mentioned earlier, the goal of this advice-based framework is to faithfully and seamlessly integrate the advice into the learning model (learning from both training data and advice). We achieve this by modifying the objective function and including a cost to account for explicitly considering the expert advice. Inspired by prior work of Gimpel and Smith (2010), we introduce a cost function in the denominator of the log-likelihood function. While they employ the use of a regularization term for a log-linear model, we employ this as a penalty term for violating the advice provided by the expert. This modified log-likelihood (MLL) function using the functional representation is given as,

$$MLL(\mathbf{x},\mathbf{y}) = \sum_{x_i \in \mathbf{x}} \log \frac{\exp\left(\psi(x_i; y_i)\right)}{\sum_{y'} \exp\left(\psi(x_i; y') + cost(y_i, y', \psi)\right)} \tag{2}$$

Our cost function is used to penalize the model that does not fit to the advice. Since the cost function penalty depends only on the advice and the current model, and not on the example labels y and y'; we can redefine it as $cost(x_i, \psi)$. We define the cost function as

$$cost(x_i, \psi) = -\lambda \times \psi(x_i) \times [n_t(x_i) - n_f(x_i)] \tag{3}$$

We use n_t to indicate the number of advice rules that prefer the example to be true and n_f to be the number of rules that prefer it to be false. We use λ to scale the cost function and $\psi(x_i)$ is the current value of the ψ function for the x_i.

Intuitively when the example label is the preferred target in more advice rules than the avoided target, $n_t - n_f$ is positive. Higher (positive) regression value, in this case, will result in a lower (negative) cost function. Since a high positive regression value corresponds to higher probability of example being true, the cost function is lower when the regression function aligns with the advice. On the other hand, if the regression value is negative, the cost is positive since the regression function doesn't align with the advice. Similarly, when $n_t - n_f$ is negative, negative regression value will have a lower negative cost and positive regression value will have a positive cost.

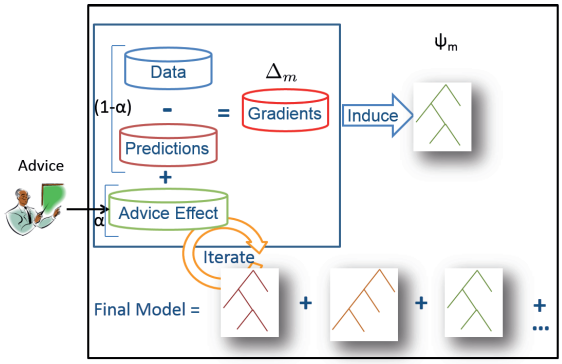

Figure 2: Standard RFGB is shown inside the black square ($\alpha = 0$) where relation regression trees are learned in a stage-wise fashion. When provided advice, the gradients for each example for which the advice applies are pushed in the direction of the advice.

Knowledge-based RFGB

We will use functional-gradient boosting to maximize our modified objective function (MLL). Omitting the derivation, the gradients of MLL will be represented as

$$\eta \cdot \Delta(x_i) = \alpha \cdot (I(y_i = 1) - P(y_i = 1; \psi)) + (1 - \alpha) \cdot [n_t(x_i) - n_f(x_i)]$$

Notice that the gradient can be decomposed into two parts: $(I - P)$ which gives the gradient from the data and $(n_t - n_f)$ which gives the gradient with respect to the advice. This is similar to other advice-based frameworks (Towell and Shavlik 1994; Fung, Mangasarian, and Shavlik 2002).

Intuitively when the example label is the preferred target in more advice models than the avoided target, $n_t(x_i) - n_f(x_i)$ is set to be positive. This will result in pushing the gradient of these examples in the positive direction (towards $+\infty$). Conversely when the example label should be avoided in more advice models, $n_t(x_i) - n_f(x_i)$ is set to be negative which will result in pushing the gradient of this example in the negative direction (towards $-\infty$). Examples where the advice does not apply or has equally contradictory advice, $n_t(x_i) - n_f(x_i)$ is 0. Hence, our approach can also handle *conflicting advice for the same example*.

The RRTs are learned stagewise as explained in the previous section and shown in Figure 2. The strength of our approach not only comes from the concise way that our method is able to handle advice, but also that this advice is used to improve the model throughout the learning process (at each stage). Furthermore, our framework allows for the advice to be changed at each iteration (for each tree learned).

Extensions

We now discuss several of the potential extensions to advice-based learning through the framework presented previously.

Advice Seeking: While advice-based learning can be successful in robust learning from noisy data, there is a substantial reliance on the advice-providing expert. The expert must take the time to not only understand the domain, but also understand where the data is noisy and thus where advice can

be most useful. An ideal learning algorithm should make an effort to understand where advice could be most useful.

Significant work has been done in active learning (Settles 2012) which seeks to find the most useful example to be labeled. While it seems that many active learning techniques could be applied to solicit advice, there are some issues. First, as advice can be general, we need advice over a set of states unlike active learning which only considers a label for a single example. This means that the potential possible advice is exponential with respect to the size of the state space. Second, we need to identify a set of states that are similar in the sense that a single advice will apply (same ordering over the labels). Finally, even if similar sets of states are identified, the data can have systematic noise that the advice must correct. This means that the algorithm may not be uncertain in the noisy areas of the state space where the advice is vital. These issues increase the difficulty in creating active techniques for advice-based learning.

Advice Refinement: Another important direction is to relax the implicit assumption that expert-advice is optimal. Even a benevolent expert may give over-generalized advice or may forget about key exceptions to a given rule. Exploring the ideas from theory refinement (Richards and Mooney 1995), such sub-optimal advice could potentially be modified by the algorithm in order to suggest improvements to the advice. This could not only improve the accuracy of the model but also be an essential tool for allowing experts to understand how to improve their own expertise.

Advice-Data Trade-off: The advice-based learning framework described previously allows for a parameter (α) that trades-off between the influences of the advice and the data. Each individual advice can have a different value for α. These parameters can have a significant impact on the learned model and determining the best way to set these parameters is not straightforward as it depends on the quality of the data and the advice as well as any other advice.

Conclusion

We discuss a framework that fosters a rich interaction between experts and a probabilistic logic learning algorithm that allows learning of more accurate, robust models from noisy data. We are interested in how this algorithm might understand the areas of the state space where advice might be most useful, ensuring the most efficient use of the experts time. We are also interested in understanding how the framework might be able to cope with noisy advice.

Acknowledgments

SN and PO thank Army Research Office (ARO) grant number W911NF-13-1-0432 under the Young Investigator Program. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the ARO.

References

Baffes, P., and Mooney, R. 1996. A novel application of theory refinement to student modeling. In *AAAI*.

Blockeel, H. 1999. Top-down induction of first order logical decision trees. *AI Communications* 12(1-2).

Friedman, J. 2001. Greedy function approximation: A gradient boosting machine. In *Annals of Statistics*.

Fung, G.; Mangasarian, O.; and Shavlik, J. 2002. Knowledge-based support vector machine classifiers. In *NIPS*.

Getoor, L., and Taskar, B. 2007. *Introduction to statistical relational learning*. Cambridge: MIT Press.

Gimpel, K., and Smith, N. A. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *HLT-NAACL*.

Karwath, A.; Kersting, K.; and Landwehr, N. 2008. Boosting relational sequence alignments. In *ICDM*.

Kersting, K., and Driessens, K. 2008. Non-parametric policy gradients: A unified treatment of propositional and relational domains. In *ICML*.

Khot, T.; Natarajan, S.; Kersting, K.; and Shavlik, J. 2011. Learning markov logic networks via functional gradient boosting. In *ICDM*.

Kok, S., and Domingos, P. 2009. Learning Markov logic network structure via hypergraph lifting. In *ICML*.

Kok, S., and Domingos, P. 2010. Learning Markov logic networks using structural motifs. In *ICML*.

Kulhmann, G.; Stone, P.; Mooney, R.; and Shavlik, J. 2004. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *Proceedings of the AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*.

Kunapuli, G.; Bennett, K. P.; Shabbeer, A.; Maclin, R.; and Shavlik, J. W. 2010. Online knowledge-based support vector machines. In *ECML*, 145–161.

Kunapuli, G.; Odom, P.; Shavlik, J.; and Natarajan, S. 2013. Guiding autonomous agents to better behaviors through human advice. In *ICDM*.

Natarajan, S.; Joshi, S.; Tadepalli, P.; Kersting, K.; and Shavlik, J. 2011. Imitation learning in relational domains: A functional-gradient boosting approach. In *IJCAI*.

Natarajan, S.; Khot, T.; Kersting, K.; Gutmann, B.; and Shavlik, J. 2012,. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning* 86(1).

Natarajan, S.; Odom, P.; Joshi, S.; Khot, T.; Kersting, K.; and Tadepalli, P. 2013. Accelerating imitation learning in relational domains via transfer by initialization. In *ILP*.

Richards, B., and Mooney, R. 1995. Automated refinement of first-order horn-clause domain theories. *Machine Learning* 19(2):95–131.

Settles, B. 2012. *Active Learning*. Morgan & Claypool.

Towell, G., and Shavlik, J. 1994. Knowledge-based artificial neural networks. *Artificial Intelligence* 69:119–165.

Yang, S.; Khot, T.; Kersting, K.; Kunapuli, G.; Hauser, K.; and Natarajan, S. 2014. Learning from imbalanced data in relational domains: A soft margin approach. In *ICDM*.