Knowledge-Based Probabilistic Logic Learning

Phillip Odom
Dept. of Computer Science
Indiana University
Bloomington, IN

Tushar Khot
Dept. of Computer Science
University of Wisconsin
Madison, WI

Reid Porter
Intelligence and
Space Research Division
Los Alamos National Laboratory
Los Alamos, NM

Sriraam Natarajan
School of Informatics
and Computing
Indiana University
Bloomington, IN

Abstract

Advice giving has been long explored in artificial intelligence to build robust learning algorithms. We consider advice giving in relational domains where the noise is systematic. The advice is provided as logical statements that are then explicitly considered by the learning algorithm at every update. Our empirical evidence proves that human advice can effectively accelerate learning in noisy structured domains where so far humans have been merely used as labelers or as designers of initial structure of the model.

Introduction

Advice taking methods have a rich and long history in artificial intelligence - from the time of explanation based learning (DeJong and Mooney 1986) to theory refinement (Baffes and Mooney 1996). The key motivation underlying this research direction is that the experts possess a vast amount of knowledge in their respective fields of which many purely data-driven machine learning techniques (Mitchell 1997; Cristianini and Shawe-Taylor 2000; Schapire and Freund 2012) fail to take full advantage. Exploiting advice in various forms has yielded successful learning algorithms particularly in the context of reinforcement learning (Torrey et al. 2005; Wiewiora, Cottrell, and Elkan 2003; Ng, Harada, and Russell 1999) where the advice is essentially used as reward shaping. In supervised learning, typically, advice is provided as feature selection and/or as inductive bias on initial models. In graphical models, advice is typically used as an initial structure that is refined (Heckerman 1999).

A key assumption (particularly in supervised learning approaches) is that the data can be used to refine or fix human advice. In several tasks however, the noise in the data can be systematic. For instance, while driving it is quite common to run stop signs or in segmentation tasks, it is often possible to miss small minute segments. The key issue is that noise in such cases is not random. However, a human expert could provide robust advice in such cases - for instance, "stop at stop signs without fail" and "the small blurb on the image should be a segment in itself" etc. We address such systematic noise and targeted advice.

Incorporating prior knowledge in machine learning has a rich history. Knowledge-based Neural Networks (Towell and Shavlik 1994), Knowledge-based Support Vector Machines (Fung, Mangasarian, and Shavlik 2002) and their recent adaptations (Kunapuli et al. 2010; 2013) have explored the combination of knowledge and data to handle systematic noise. While specific adaptations differ, all these methods take as advice Horn clauses and convert them to their corresponding representation. We on the other hand, aim at better integrating the advice (as Horn clauses) with a probabilistic logic model (PLM). Our work can thus be viewed as preference elicitation (Boutilier 2002; Drummond and Boutilier 2014) for PLMs.

Specifically, we consider a recently successful PLM formalism called as Relational Functional Gradient Boosting (RFGB) (Natarajan et al. 2012) as the underlying base learner and provide human guided advice to this learning framework. Our advice consists of preferred and avoided target labels (labels that should have higher probability than other labels) defined over spaces of examples. The space of examples where the advice applies is given by horn clauses which can be thought of as if-then rules specified by experts. Since our underlying representation is also based on first-order logic, the system can more faithfully exploit the advice compared to converting it to a different representation internally by the learning algorithm.

We show that the proposed framework is not restricted to a particular type of PLM model. Specifically, we show that several formalisms and tasks such as Markov Logic Networks (MLNs) (Domingos and Lowd 2009), Relational Dependency Networks (RDNs) (Neville and Jensen 2007), relational imitation learning (Natarajan et al. 2011) and relational transfer can benefit from this framework. Our framework uses both the advice and the data throughout learning process and allows for sequential interaction where the expert could potentially add more advice as learning progresses.

We make the following contributions - (1) We present the first work on exploiting human advice while learning several PLMs (2) We outline a natural framework for the human to provide general advice about the domain as well as specific advice about particular examples (3) Our method naturally integrates human input with examples in a boosting formalism and (4) Finally, our experiments on a several data sets demonstrate that the framework efficiently exploits the human advice in learning effective models.

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Background

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 (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 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 (RRTs)(Blockeel 1999). 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 RDNs 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 that the functional gradient of the likelihood for RDNs as

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

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 gradients for negative examples is negative and hence the ψ function is pushed closer to $-\infty$ and probability closer to 0.

Advice-based Learning: 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 PLMs have not been expressive and mainly consisted of hand-coding the structure and possibly even the parameters of the model. Such techniques have been used in many problems including natural language processing (Riedel et al. 2009; Yoshikawa et al. 2009; Poon and Vanderwende 2010). 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 related method by 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. But if the training data is sub-optimal, it is possible that the learning algorithm will refine the model away from the advice (which is precisely the goal in transfer learning as the target domain is different from the source domain). On the other hand, we seek to use the advice throughout the learning process and thereby handle noisy examples.

Advice-Based Learning in Relational Problems

As mentioned earlier, learning in PLMs, which is similar to the propositional graphical models, has two components - parameter learning and structure learning. Structure learning identifies the presence of a relationship between logical predicates while the parameters quantify the strength of these relationships. Most of the structure learning in literature use an initial model from the expert and uses the data to fix the mistakes in these models. As mentioned earlier, this essentially fixes the mistake in the advice and does not explicitly model the systematic errors in the training instances. We now present a model that effectively exploits user advice to fix the systematic noise. To this effect, we adapt the functional boosting approach with a modified objective function that trades off between the effect of advice and the examples. We first present our definition of the user advice before outlining the objective function and the modified algorithm.

Relational Advice

A natural question to ask is: what are the type and desiderata of good advice? We argue that this advice must be (1) generalized (as against targeting a specific example), (2) described using parameterized properties (as against example identifiers), (3) 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 (4) in the same representation as that of the learning algorithm. To this effect, we employ the use of first-order logic to capture the expert's knowledge as a set of preference rules on the space of labels/actions as a function of the attributes of the object or related objects. Formally, our advice is defined as,

Definition 1. *Relational advice set (RAS), R is specified as a set of relational advice rules (RAR), $r_1, r_2, \ldots, r_A$. Each RAR, r_a is defined using the triple $\langle F, l+, l-\rangle$ where F is the relational advice constraint (RAC) clause specifying the subset of examples, $l+$ is the preferred label and $l-$ is the avoided label.*

Definition 2. *A relational advice constraint (RAC), F is defined using a Horn clause $\wedge_i f_i(\mathbf{x_i}) \Rightarrow label(\mathbf{x_e})$, where $\wedge_i f_i(\mathbf{x_i})$ specifies the conjunction of conditions under which the advice applies on the example arguments $\mathbf{x_e}$.*

Example 1. *Consider the problem of predicting whether an actor has worked under a director on a movie. 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. Suppose, the advice set R contains one rule, $r_1 = \langle F, l+, l-\rangle$. Here $F = actor(a) \wedge$*

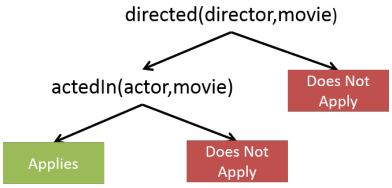

Figure 1: An advice model in the IMDB domain for predicting if an actor has worked under a director. Each node in the decision tree is a relational condition. The leaf nodes represent whether or not the advice will apply to examples that reach that leaf.

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

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. Consider $\mathbf{s}$ to be the set of training examples for which the advice is applicable, i.e., $\mathbf{s} = \{s_i \mid B, F \vdash label(s_i)\}$, where B is the background knowledge and F is the advice constraint. The learned model should have a higher probability of the preferred target than the probability of the avoided target in the training examples, i.e., $\forall s_i \in \mathbf{s}, P(l+(s_i)) \geq P(l-(s_i))$.

Example 2. *For the example presented earlier, any actor a and director d that appear in the same movie,* $P(workedUnder(a,d)) \geq P(\neg workedUnder(a,d))$.

Our advice can also handle sets of preferred and avoided labels by converting them into multiple advice rules for every pair of preferred and avoided labels. This is especially relevant when dealing with more than two labels. E.g., an expert might say that if the director is "Michael Bay", then the preferred genre of the movie is "action", i.e., if director("Michael Bay", movie) then $\forall$ g $\neq$ "action", P(genre(movie,"action")) $\geq$ P(genre(movie,g)). Also we are not limited to Horn clauses for specifying the advice constraints. We can also use RRTs (Blockeel 1999) to specify the regions of the example space where advice is applicable. Note that implementation of this is not difficult, since every path from root to leaf can be viewed as a Horn clause and thus a tree is a decision-list of Horn clauses.

Example 3. *Figure 1 shows the relational advice constraints as a relational tree for our earlier example. The advice is applicable on examples which reach the green node and not applicable for examples reaching the red node.*

Knowledge-based Objective Function

One simple way to use this type of advice is to define the advice as the first tree of the boosted model and subsequent models can be learned based on data. The key issue is that this method in the worst case, can undo the advice to better fit the (possibly noisy) training instances. As mentioned earlier, if the training examples are noisy, we do not want to fit only to the training examples but the advice rules as well. Hence, there is a necessity to faithfully and seamlessly integrate the advice into the learning model. We achieve this by modifying the objective function and including a cost to account for explicitly considering the value 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,

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

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 $c(x_i,\psi)$. We define the cost function as

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

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.

Example 4. *In our movie domain, consider two advice rules* $r_1 = \langle\ movie(m, MichaelBay) \Rightarrow label(m), action,\ horror\ \rangle$ *and* $r_2 = \langle\ movie(m, WillArnett) \Rightarrow label(m), comedy, action \rangle$.

The Transformers movie has Michael Bay *but doesn't have* Will Arnett. *Hence for the example action(Transformers), only the first rule applies where action is the preferred label. So* $n_t = 1$ *and* $n_f = 0$ *for action(Transformers), while* $n_t = 0$ *and* $n_f = 1$ *for horror(Transformers). Whereas, the new Teenage Mutant Ninja Turtles (TMNT) movie has both* Michael Bay *and* Will Arnett. *While both the rules apply on action(TMNT), action is the preferred label in one rule and avoided label in the other. Hence* $n_t = 1$ *and* $n_f = 1$ *for action(TMNT).*

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.

Knowledge-based RFGB

We now use functional-gradient boosting to maximize our modified objective function (MLL). The cost function in Equation 2 can be replaced with Equation 3

$$MLL(\mathbf{x}) = \sum_i \log \exp(\psi(x_i;y_i)) - \log\left(\sum_{y'} \exp\left(\psi(x_i;y'\right)\right.$$

$$-\lambda \times \psi(x_i) \times [n_t(x_i) - n_f(x_i)])\Big)$$
$$= \sum_i \log \exp \psi(x_i; y_i) - \log \sum_{y'} \exp(\psi(x_i; y'))$$
$$- \log \exp(-\lambda \times \psi(x_i) \times [n_t(x_i) - n_f(x_i)])$$
$$= \sum_i \log P(y_i, x_i; \psi) + \lambda \cdot \psi(x_i) \cdot [n_t(x_i) - n_f(x_i)]$$

Obtaining functional gradients of MLL,

$$\begin{aligned} \frac{\partial MLL(\mathbf{x})}{\partial \psi(x_i; y_i)} &= \frac{\partial}{\partial \psi(x_i; y_i)} \log P(y_i, x_i; \psi) \\ &+ \lambda \cdot \psi(x_i) \cdot [n_t(x_i) - n_f(x_i)] \\ \Delta(x_i) &= I(y_i = 1) - P(y_i = 1; \psi) \\ &+ \lambda \cdot [n_t(x_i) - n_f(x_i)] \end{aligned}$$

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*. We rewrite our gradients by setting $\eta = \alpha$ and $\lambda = (1 - \alpha)/\eta$ to get

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

Our approach can now be understood as linearly trading-off between the data and the advice as controlled by α. The RRTs are learned stagewise as explained in the previous section and shown in figure 2. As shown in algorithm 1, 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. Using the advice just as the initial model will result in the learning algorithm undoing the advice if the examples are noisy. This can also be seen in our empirical evaluation.

Adaptations for Multiple Frameworks

Note that the objective function that we modified is a general one used in prior literature for relational domains. Thus we can use our knowledge-based RFGB for any model that uses RFGB. We show the adaptation of our work in two directions: (1) what can be learned and (2) what problems can be solved. To this effect, we show that we can learn the structure of both RDNs and MLNs. We also show that we can apply these models for relational policies and transfer learning. This can also be employed in other similar formalisms.

What can be learned?

RDN: Based on prior work for RDNs (Natarajan et al. 2012), we learn the structure of RDNs by learning a sequence of relational regression trees (RRT) for every predicate independently. We use the modified gradients to calculate the regression values for each ground example and use the modified regression examples to learn the RRTs.

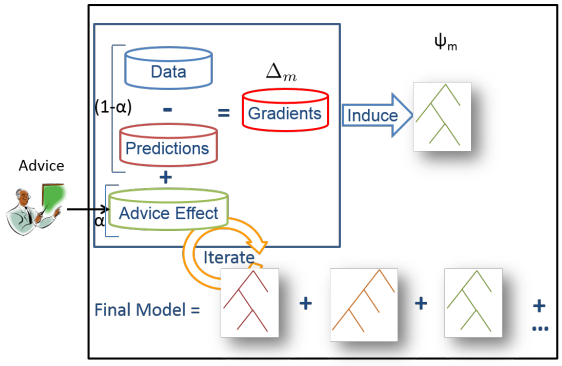

Figure 2: Standard RFGB is shown inside the black square (advice penalty equal to 0 for all examples) where relation regression trees are learned in a stage-wise fashion. When provided expert advice, the gradients for each example for which the advice applies are pushed in the direction of the advice (positive if the advice corresponds to the probability of the target being higher and vice versa).

MLN: Since MLNs make the conditional independence assumption only during learning, prior work (Khot et al. 2011) iteratively learned one tree for each predicate and used the previous trees for all the predicates for calculating the gradients. We also used the same approach with the modified gradients presented here.

What problems can be solved?

Relational policies: Imitation learning uses expert trajectories to directly learn a policy that best fits the expert trajectory. For relational domains, RFGB has been used to learn the relational policy by learning RRTs for every action (Natarajan et al. 2011). Since our trajectories can be noisy, we use advice to compute the modified gradients for every action to learn knowledge-based relational policy.

Transfer learning: Transfer learning can be viewed as learning from a source task that has systematic differences with the target task. Previous work used RFGB to learn a model for a target domain by using the model from the source domain as an initial model (Natarajan et al. 2013) and refining the model with target examples. We are able to learn only by using source examples. Expert advice can make source examples more applicable to target domain by focusing on key differences between source and target tasks.

Experiments

We present empirical analysis to study different questions: 1) how effective is advice for relational classification, 2) can we employ advice in sequential problems (specifically, imitation learning), 3) does our method leverage knowledge across domains (transfer learning), and 4) how does the advice help in standard domains for learning MLNs?.

Algorithm 1 ARFGB: Advice for Relational Function Gradient Boosting

function ADVICEBOOST($Data$,$Advice$)
  ▷ Iterate through the predicates
  for $1 \leq k \leq K$ **do**
    ▷ Iteration through the gradient steps
    for $1 \leq m \leq M$ **do**
      S_k = GENEXAMPLES(k,$Data$,F^k_{m-1},$Advice$)
      $\Delta_m(k)$ = FITRELREGRESSTREE(S_k,L)
      $F^k_m = F^k_{m-1} + \Delta_m(k)$
    end for
    ▷ ψ^k is obtained by grounding F^k_M
    $P(Y_k = y_k|\mathbf{Pa}(X_k)) \propto \psi^k$
  end for
end function
function GENEXAMPLES(k,$Data$,$Advice$)
  $S = \emptyset$
  for $1 \leq i \leq N_k$ **do** ▷ Iterate over all examples
    ▷ Probability of the predicate being true
    Compute $P(y^i_k|x^i_k, Pa(X^i_k))$
    Compute $C^i = n_t(x_i) - n_f(x_i)$
    $\Delta(y^i_k, x^i_k, Advice) = (1 - \alpha)$
    $\left[I(y_i = 1) - P(y^i_k = 1|Pa(x^i_k))\right] + \alpha C^i$
    $S = S \bigcup [(y^i_k, x^i_k, \Delta(y^i_k, x^i_k, Advice)]$
  end for
  return S
end function

We use two baseline approaches to compare against our method (called *Gradient* in the results). To evaluate the importance of advice for RFGB, we compare against RFGB without using any advice rules called *RFGB* in the results. Also, we compare our approach against using the advice as initial model for RFGB (called *Initial*) to demonstrate the effectiveness of our approach for incorporating advice. As the advice itself rarely specifies enough information to build a complete model, we show the relative influence of the advice in table 1, where we show the fraction of training examples to which the advice applies. Although our advice only covers a part of the example space, the key intuition is that there are several small yet important regions where good advice might be crucial, which we also show empirically. To compare the approaches, we use test-set accuracy averaged over multiple folds. We evaluate the results across three data sets. Figure 3 presents two of these experimental domains.

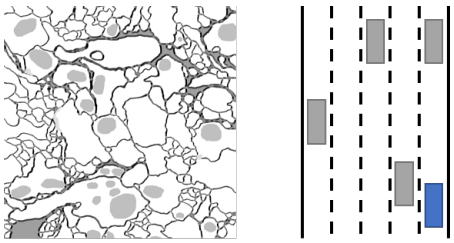

Figure 3: Domains: Drosophila (**LEFT**) - different grayscale values represent different segment classes, and Driving (**RIGHT**) - the blue car is driving to avoid other cars on the highway.

Table 1: Domain and Advice Descriptions

Domain	RC	IL	TL	IMDB
Advice Coverage	27.2	55.3	38.0	85.3

Relational Classification

We evaluate our approach on Relational Classification in an image labeling domain where the goal is to label the segmented regions of an image. We use the Drosophila dataset (Cardona et al. 2010), which has 20 stacks of microscopic images of fruit flies ventral nerve cord. The possible labels include extra-cellular region, intra-cellular region, mitochondria, and membrane. This domain is naturally relational as the different regions have spatial relationships, e.g. membrane surrounds the cell (intra and mitochondria) and the number of segments can vary across images. We assume that we are given a perfectly segmented image (each region corresponds to one and only one object). Our features include region properties including color, color variance, area, perimeter, circularity, number of neighbors as well as edge features (representing boundary between two regions) such as length, shape, area and color difference between regions.

We divide the dataset into 4 train and test sets, where each training set consists of 3 layers, while each test set consist of 2 layers that are 10 layers away from the training set. We introduced targeted noise in the membrane labels to emulate the natural mistakes in segmentation. Thus it is difficult to classify membrane regions in the noisy space. Our advice is then given on the noisy space to mimic a human expert.

In addition to relational models, we also compare against propositional models learned with a limited set of features (*Prop*) which do not have the spatial information or relational neighborhood information, while the relational (*Rel*) models have the full feature set. We compare models learned without advice to ones with advice with our method (*Gradient*) and our baseline method (*Initial*). All the boosting approaches learn five trees in this domain.

In table 2.**LEFT**, we show the overall accuracy over all the labels where the label distribution is heavily skewed towards membrane.

All advice methods are able to outperform those that did not have access to the advice. When comparing the advice models, our *Gradient* method is able to outperform the baseline *Initial* method. Even the *Gradient (Prop)* model which was learned from propositional feature space outperforms the *Initial (Rel)* method which had access to the neighborhood information too. Recall that this is due to the *Initial* method starting with the advice and then refining the model away from the advice, while our *Gradient* method enables refining the models towards the advice at each step. Our method *Gradient* is able to combine information from both the training data and the advice model; thus achieve performance gains from both.

Imitation Learning

We next consider Imitation Learning setting using a driving simulator extended from Judah et al (2014). The goal in this domain is to navigate on a 5-lane highway, changing lanes in

Model	ACC
FGB (Prop)	68.6 $\pm$0.4
RFGB (Rel)	69.3 $\pm$0.5
Adv-Initial (Prop)	68.8 $\pm$0.6
Adv-Initial (Rel)	91.6 $\pm$0.4
Adv-Gradient (Prop)	97.0 $\pm$0.5
Adv-Gradient (Rel)	**99.1** $\pm$0.4

Model	ACC
RFGB	52.5 $\pm$5.0
Adv-Initial	52.2 $\pm$5.9
Adv-Gradient	**96.0** $\pm$0.4

Model	mit	other	total
FGB (Prop)	73.0	92.1	92.0 $\pm$0.9
RFGB (Rel)	79.7	91.7	91.6 $\pm$0.4
Adv-Initial (Prop)	84.3	90.1	90.0 $\pm$0.7
Adv-Initial (Rel)	86.3	90.3	90.3 $\pm$0.6
Adv-Gradient (Prop)	83.6	99.8	**99.7** $\pm$0.2
Adv-Gradient (Rel)	75.2	99.5	**99.4** $\pm$0.1

Table 2: Relational Classification - Drosophila (**LEFT**), Imitation Learning - Driving (**CENTER**), Transfer Learning - Drosophila (**RIGHT**)

order to avoid other cars that are traveling at various speeds. The driver may either stay in the current lane, or move to the right or left lane. This domain is naturally relational as the spatial information is crucial to making driving decisions and the number of cars changes across different scenarios. For example, if there is a car in front of an agent and a car to the right of the car in front, then moving into right lane might not be the correct action.

We learn from 10000 training examples (100 trajectories) of an expert acting in the domain but choosing a suboptimal action in certain states (like always driving in the left lane) and test on another 10000 (100 trajectories) examples of another expert driving correctly. This is averaged over 5 runs. In this domain, sample advice could be to drive in the rightmost lane when possible, even though human drivers often prefer to ride in the left lanes. In our experiments, the advice was to stay in the current lane unless there are cars ahead.

The results, table 2.**CENTER**, show that our method with advice is able to significantly outperform Natarajan et al (2011) which learns without advice. In this domain too, we successfully leverage data and advice to make a more accurate prediction.

Transfer Learning

We next verify how our advice framework can assist in transfer learning in a relational domain. This could be especially useful if there is a related domain whose data points could be useful but the label space either has incorrect distribution or the space is simply different. Our advice can make the data in this form more applicable to the target domain, thus increasing the value of these examples for the target problem.

We use a version of the Drosophila dataset from the previous section. However, we assume that the source data has a limited label space. The source data is over whether a region is inside a cell or outside a cell. For our training data, we assume that intra-cellular and mitochondria regions are inside the cell, while extra-cellular and membrane regions are outside of the cell. However, the target problem is determining whether or not a region belongs to the mitochondria class. Notice that while the source data has a relationship with the target problem, it alone will not be sufficient to build a model for mitochondria detection as there are a significant number of other object inside a cell. Advice is used to identify interesting objects (mitochondria) in the cell.

The results (shown in table 2.**RIGHT**) show that the advice is able to filter out many of objects that are non-interesting (not mitochondria) and thus significantly improve the model. This is shown in the *Gradient* models improved accuracy on the non-interesting classes (other).

Table 3: Relational Classification - IMDB

Model	5% Noise	10% Noise	25% Noise
RFGB	55.5 $\pm$9.3	55.9 $\pm$7.6	57.6 $\pm$ 9.0
Adv-Gradient	**84.5** $\pm$6.0	**81.0** $\pm$2.3	**75.9** $\pm$4.4

IMDB

Finally, we also evaluate our advice framework in learning MLNs on a standard relational dataset, IMDB.

The IMDB dataset (Mihalkova and Mooney 2007) has information about movies, actors, and directors represented with five predicates: `actor`, `director`, `genre`, `gender` and `workedUnder`. We predict the workedUnder relationship between the people in this dataset. We performed five-fold cross-validation on this domain. We apply uniform noise (5%, 10%, 25%) to this dataset repeated five times for each fold and average the results. Unlike other domains, the IMDB dataset has a relatively large number of negative examples and hence even with 25% noisy positive examples, the impact on the negative examples is marginal. Hence, we also reduce the number of negative examples in all our approaches in this domain.

Table 3 shows the accuracy of our approaches on this domain for learning MLNs. Our approach (Gradient) outperforms learning without advice (RFGB), showing that our approch can be useful even when dealing with noisy examples and not just systematic noise.

Trade-off Between Advice and Data

From our experiments across the domains for different α values, we noticed that the learning method is robust to reasonable α (0.25-0.75) values. While the optimal α value may require parameter tuning, our experimental evidence suggests that as long as it is reasonably set, the system can benefit from good advice. Large α pushes the model to trust the advice too much while low α makes it ignore the advice.

Conclusion and Future Work

We propose a novel method for allowing rich interaction between experts and PLMs. The main contribution of our approach is that it continuously leverages advice provided by domain experts while learning as against traditional methods that consider it as an initial model. We demonstrate empirically that our approach is effective when learning from noisy training examples and can still perform well when dealing

with few examples. Our framework can control the relative impacts of advice and training data on the learned model.

Going forward, we will work on providing a theoretical understanding of our update to the gradients. Currently our approach considers all advice to be equal, but this may not always be the case. We will work on associating weights with the expert advice which can either be learned or provided by the expert. While we discussed some initial results suggesting that the trade-off between advice and data is robust to reasonable values of α, we propose to study the impact of this trade-off with noisy advice.

Acknowledgements

SN and PO thank Army Research Office (ARO) grant number W911NF-13-1-0432 under the Young Investigator Program. SN and TK gratefully acknowledge the support of the DARPA DEFT Program under the Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0039. 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 DARPA, ARO, AFRL, or the US government.

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).

Boutilier, C. 2002. A pomdp formulation of preference elicitation problems. In *IAAI*.

Cardona, A.; Saalfeld, S.; Preibisch, S.; Schmid, B.; Cheng, A.; Pulokas, J.; Tomancak, P.; and Hartenstein, V. 2010. An integrated micro- and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. In *PLoS Biol*.

Cristianini, N., and Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

DeJong, G., and Mooney, R. 1986. Explanation-based learning: An alternative view. *Machine Learning* 1(2):145–176.

Domingos, P., and Lowd, D. 2009. *Markov Logic:An Interface Layer for Artificial Intelligence*. Morgan & Claypool.

Drummond, J., and Boutilier, C. 2014. Preference elicitation and interview minimization in stable matchings. In *AAAI*.

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*.

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

Heckerman, D. 1999. *A Tutorial on Learning with Bayesian Networks*. MIT Press.

Judah, K.; Fern, A.; Tadepalli, P.; and Goetschalckx, R. 2014. Imitation learning with demonstrations and shaping rewards. In *AAAI*.

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*.

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*.

Mihalkova, L., and Mooney, R. 2007. Buttom-up learning of markov logic networks. In *ICML*.

Mitchell, T. 1997. *Machine Learning*. Mcgraw Hill.

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*.

Neville, J., and Jensen, D. 2007. Relational dependency networks. *JMLR* 8.

Ng, A.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*.

Poon, H., and Vanderwende, L. 2010. Joint inference for knowledge extraction from biomedical literature. In *NAACL*.

Riedel, S.; Chun, H.; Takagi, T.; and Tsujii, J. 2009. A Markov logic approach to bio-molecular event extraction. In *BioNLP*.

Schapire, R., and Freund, Y. 2012. *Boosting: Foundations and Algorithms*. MIT Press.

Torrey, L.; Walker, T.; Shavlik, J.; and Maclin, R. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *ECML*.

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

Wiewiora, E.; Cottrell, G.; and Elkan, C. 2003. Principled methods for advising reinforcement learning agents. In *ICML*.

Yoshikawa, K.; Riedel, S.; Asahara, M.; and Matsumoto, Y. 2009. Jointly identifying temporal relations with Markov logic. In *ACL*.