The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

Relational Boosted Bandits

Ashutosh Kakadiya,[1] **Sriraam Natarajan,** [2] **Balaraman Ravindran** [1]
[1] Robert Bosch Centre for Data Science and Artificial Intelligence, Indian Institute of Technology Madras
[2] The University of Texas at Dallas
kashutosh@cse.iitm.ac.in, Sriraam.Natarajan@utdallas.edu, ravi@cse.iitm.ac.in

Abstract

Contextual bandits algorithms have become essential in real-world user interaction problems in recent years. However, these algorithms represent context as attribute value representation, which makes them infeasible for real-world domains like social networks, which are inherently relational. We propose *Relational Boosted Bandits* (*RB2*), a contextual bandits algorithm for relational domains based on (relational) boosted trees. *RB2* enables us to learn interpretable and explainable models due to the more descriptive nature of the relational representation. We empirically demonstrate the effectiveness and interpretability of *RB2* on tasks such as link prediction, relational classification, and recommendation.

Introduction

The contextual bandit framework has gained a lot of attention in several real-world personalization applications from news recommendation to online advertising (Li et al. 2010), comments recommendation (Mahajan et al. 2012), clinical trials (Durand et al. 2018), A/B testing, and dialogue systems (Liu et al. 2018). Contextual bandit is an extension of *multi-armed bandits* with a context vector for each user. This context about the individual user allows the personalization of the actions than calculating a simple argmax over all actions. The general framework of contextual bandits (Langford and Zhang 2008) can be formalized as follows: at each time instance t, a user arrives with a vector of information (or features) referred to as a context vector. The goal is to choose one action for the user among K actions given the context (user and actions). The reward r is observed for only the chosen action and the objective is to maximize the total reward over time.

Most of the contextual bandit algorithms (Zhou 2015) have focused on propositional domains, where the context is described using a flat feature-vector representation. Typically however, many real-world domains are naturally structured and are described by interacting objects and relations between them. This representation allows for learning richer models. Additional domain knowledge in the form of inductive/search bias is typically employed for learning in such domains. Inspired by this direction, we explore the combination of contextual bandit and Statistical Relational Learning (SRL).

SRL (Getoor and Taskar 2007) combines the power of relational/symbolic representations with the ability of probabilistic/machine learning models to handle uncertainty. Consequently, it is well suited for several real-world tasks such as social network analysis, recommendation and biomedical applications. Initial research focused on the development of several models – directed models (Friedman et al. 1999; Kersting and De Raedt 2007), undirected models (Richardson and Domingos 2006) and bi-directed models (Neville and Jensen 2007). More recently, focus has rightly turned to learning SRL models (Getoor and Taskar 2007; De Raedt, Kimmig, and Toivonen 2007; Natarajan et al. 2012b). Arguably, learning in these models is computationally intensive as it requires exploring multiple levels of abstractions (at the object level, partial instance level or fully ground level). One of the recent successful methods is to learn a set of relational regression trees using gradient-boosting (RRTGB) (Natarajan et al. 2012b). The key advantage of this method is that it learns a set of weak classifiers and can avoid searching for a single model. This method has been successfully applied for many applications such as recommendation (Yang et al. 2017), cardiovascular conditions (Natarajan et al. 2012a), PPMI (Dhami et al. 2017) and rare diseases (MacLeod et al. 2016) to name a few. While successful, these models are restricted to only batch mode learning. While a previous online algorithm exists (Huynh and Mooney 2011) for learning an undirected model, this algorithm does not have the distinct advantage of bandit approaches - the ability to explore or exploit.

Motivated by the success of the boosting method in batch settings, we propose a new *relational contextual bandit framework* based on RRTGB. The resulting framework, called *Relational Boosted Bandits* (*RB2*), combines the power of a powerful learning algorithm with the exploration-exploitation abilities of bandits. One of the key motivations of this combination is that the structure and parameters of the underlying model can be learned simultaneously, thus allowing for effective learning in online settings. Our **key insight** is to represent the policy as a set of relational regression trees (RRT) (Blockeel and De Raedt 1998). Consequently, the online relational learning algorithm *RB2* has the ability to learn using relational data while employing

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

an effective sampling mechanism to handle the exploration-exploitation trade-off. In addition, since these trees are essentially combined using a simple addition operator, we construct a final tree based on these different trees to build an interpretable model.

We make the following key contributions: (1) To the best of our knowledge, we are proposing the first contextual bandit algorithm based on SRL for relational domains. (2) We propose a parameter-free sampling algorithm for the online learning of probabilistic relational models. (3) Finally, we perform comprehensive experiments on several tasks and demonstrate the efficiency and effectiveness of the proposed approach.

The rest of the paper is organized as follows. After introducing the necessary background and related work about contextual bandit and boosted relational regression trees, we propose our *RB2* algorithm. Then we demonstrate an empirical assessment of our algorithm on real-world data sets before concluding by outlining the areas of future research.

Background and Related Work

Contextual Bandit

Contextual Bandit is a variant of the classical multi-armed bandit problem (Langford and Zhang 2008) where the agent has access to side information (context) for better decision making. The agent has to make context-based sequential decisions from time $t = 1, 2, 3, ..., T$. At each time-step, the agent has to decide which arm to select from the given K arms. After selecting the arm, the agent receives the payoff from the environment corresponding only to the selected arm while the other payoffs are unknown. The goal is to learn the policy that maximizes the expected payoff. The arm with the highest expected payoff may be different for different contexts. The context includes static and dynamic information about both the agent and the arms. The typical evaluation metric used is the *regret* which is given by the cumulative sum of the difference between optimal payoff and the actual payoff received over the horizon T. Many real-world problems can be formulated in a contextual bandit setting (Bouneffouf and Rish 2019). Few examples are recommendation systems, financial portfolio management, ad placement on websites, and healthcare. While successful, they are not generally applied to multi-relational settings, a key direction in our work.

Boosted Relational Regression Trees

Gradient-boosted relational regression trees (RRTGB) (Natarajan et al. 2012b, 2011) adapts gradient-boosting (GB) (Friedman 2000) to relational domains. For classification, typically GB, calculates the functional-gradient of the examples in the form (x_i, y_i), $i = 1, 2, 3, .., M$ and $y_i \in \{1, 2, 3, .., K\}$. This gradient is the difference between the true label (represented by an indicator function) and the predicted probability of the true label. RRTGB uses a relational regression tree learner (TILDE) (Blockeel and De Raedt 1998) to represent the potential function ψ. In a relational regression tree, each node represents the conjunction of literals. In RRTGB, the functional gradient ascent starts with the initial potential function ψ_0, iteratively computes gradients and adds to the existing model. Formally at the end of n^{th} iteration, the potential function is given as,

$$\psi_n = \psi_0 + \Delta_1 + \Delta_2 + ... + \Delta_n \tag{1}$$

And, the functional gradient Δ_n at iteration n is given as,

$$\Delta_n = \eta_n \times E_{x,y}[\partial/\partial\psi_{n-1} logP(y|x; \psi_{n-1})] \tag{2}$$

This procedure is repeated until a fixed number of iterations is reached or till convergence. Our algorithm uses RRTGB as a base learner to model the relation between context and the probabilities of getting a reward of 1.

Problem Formulation

For every time-step, $t = 1, 2, 3, ..., T$, a user arrives with context as x_t. The features consist of the defined predicates of both users and the arms. At a fairly high level, *RB2* will choose the action $a_t \in \{1, 2, 3, .., K\}$, and obtains the reward $r_{t,a_t} \in \{0, 1\}$ associated with the chosen arm a_t and context x_t. The goal is to maximize the (expected) cumulative sum of reward.

Let $p(x, a)$ denote the probability of observing a reward $r = 1$. The model predicts $p(x, a)$ $\forall a$ and selects the arm $a_t = \arg\max_a p(x, a)$. The regret is simply the difference between reward received by the algorithm r_{t,a_t} and reward r_{t,a_t*} associated with the optimal action $a*$. Formally, at time period T, the cumulative regret $R(T)$ is defined as,

$$R(T) = \sum_{t=1}^{T} (r_{t,a_t^*} - r_{t,a_t}) \tag{3}$$

We use cumulative regret as the evaluation metric.

Relational Boosted Bandits

A key aspect of our setting is that the context is inherently relational, i.e., the context cannot be specified with a flat feature vector based representation. Instead, each instance's attributes/relations could be of differing size (papers published, movies acted, hospital visits, lab tests, etc.) thus requiring a representation that is more general than a simple feature vector. To this effect, we employ first-order logic based notations for representation and learning. A (logical) **predicate** is of the form $\mathcal{R}(t_1, \ldots, t_k)$ where $\mathcal{R}$ is a predicate and t_i are **arguments** or logical variables. We refer to the totality of observed contexts as *background knowledge*. A **substitution** is of the form $\theta = \{\langle v_1, \ldots, v_k \rangle / \langle t_1, \ldots, t_k \rangle\}$ where v_is are logical variables and t_is are terms. A **grounding** of a predicate with variables $v_1, \ldots, v_k$ is a substitution $\{\langle v_1, \ldots, v_k \rangle / \langle V_1, \ldots, V_k \rangle\}$[1] mapping each of its variables to a constant in the domain of that variable.

Given: The context x_t at time t and the accumulated background knowledge $\mathcal{B}$.
To Do: Predict $a_t = \mathcal{F}(P(x_t, a_t|\mathcal{B}))$

[1] We use uppercase for predicates/groundings and lowercase for variables.

The goal of our system is to pick an action based on the learned action model P and the exploration-exploitation strategy $\mathcal{F}$. To learn P, we employ gradient-boosting while for $\mathcal{F}$, we consider two different types of strategies – epsilon-greedy and informed sampling. We now present the details of the formulation and the learning methodology.

RB2 Framework

The key aspect of our framework is that it is an **online** algorithm. The only related prior work that considered online learning of MLNs is by Huynh and Mooney (yearhuynh11). Our approach can be seen as a more efficient approach that explicitly learns to the trade-off between exploration and exploitation. More precisely, *RB2* is an **online** algorithm where the learning happens in mini-batches. The exploration strategy is employed in the choice of batches while training and the action selection during evaluation. During training, the action choices are aggregated into mini batches and the parameters are updated once every mini batch. During deployment/testing, there is no aggregation since there is no learning. We now proceed to explain each of the components in greater detail. We focus on learning the distribution before discussing the action selection.

At each training mini-batch b, a certain set of training examples have been collected. From these examples, a certain set of examples (say D_b) will be selected based on the exploration strategy that we will explain next. For each example, given its context x_t, the currently accumulated background knowledge $\mathcal{B}$, the goal is to learn $P(a_t, x_t, \mathcal{B})$ which gives the probability distribution over the actions given the context. In our bandit setting, this corresponds to the choice of multiple arms. In a classification setting, this could correspond to the distribution over the classes, in a recommendation setting over the items to be recommended etc.

As mentioned earlier, we model the distribution $P(a_t, x_t, \mathcal{B})$ using RRTGB. Hence the distribution $P(a_t, x_t, \mathcal{B})$ is represented as,

$$P(a_t, x_t, \mathcal{B}) = \frac{e^{\psi(a_t, x_t, \mathcal{B})}}{1 + e^{\psi(a_t, x_t, \mathcal{B})}}$$

RRTGB now learns the gradient of the loglikelihood over the mini-batch $\sum_t [log(P(a_t, x_t, \mathcal{B}))]$ w.r.t ψ. Following the standard gradient-boosting, point-wise gradients are computed for each example. This is simply of the form $I(a = a_t) - P(a_t, x_t, \mathcal{B})$ which is the difference between whether the action was chosen in the given data and the current predicted probability of the action given the context and background knowledge. These point-wise gradients are chosen for all the examples in D_b. Next a TILDE tree is fit over these gradients, where the goal of the tree learner is to minimize the weighted variance of the regression value. Once the trees are fitted, boosting proceeds to the next iteration where newer gradients are computed and the new tree is fit. The process is repeated for the preset number of boosting iteration in each minibatch. Typically, in our experiments, the number of trees (K) is preset to $4 \leq K \leq 10$.

We update the background knowledge $\mathcal{B}$ periodically as new relational information arrives. Typically, this is in the form of newer attributes of either known objects/entities or new objects themselves. For example, in a university domain, this could be newer courses offered by the department or in a movie domain, this could be newer movies directed/acted by a particular person. This new information could also be a modified background knowledge, for instance, new merger/acquisitions of the concerned firms. One of the advantages of our approach is that we can **adapt to the changing background knowledge** more seamlessly because the later iterations of the boosted model can appropriately model the target distribution to better reflect this updated knowledge.

Given that we have explained how $P(a_t, x_t, \mathcal{B})$ is learned, we now turn our focus on effective sampling strategy for online learning. We note that this sampling algorithm is specifically useful in relational domains. The goal of this sampling algorithm is to assign a high probability of getting sampled to important samples. If we use uniform sampling then important data samples have very less chance to getting selected. We use stochastic prioritization to assign priority(weights) to data samples. To achieve this, we divide each mini-batch data set $\mathcal{D}$ into two data sets $\mathcal{D}_p$ and $\mathcal{D}_n$. $\mathcal{D}_p$ consists of all data samples with $r = 1$ and $\mathcal{D}_n$ with $r = 0$. We can easily obtain the priority probability distribution $P_s(i)$ for data sample $i \in \mathcal{D}_p$ by Equation 4

$$P_s(i) = \frac{e^{1-p_i}}{\sum_{j \in \mathcal{D}_p} e^{1-p_j}} \tag{4}$$

and for data sample $i \in \mathcal{D}_n$ by equation 5.

$$P_s(i) = \frac{e^{p_i}}{\sum_{j \in \mathcal{D}_n} e^{p_j}} \tag{5}$$

Recall that RRTGB predicts the probability p of choosing the correct action, i.e., $p(a_t) = 1 \forall t$. This corresponds to getting a reward 1. As with classification, the goal is to iteratively make $p \rightarrow 1$ for positive samples. To improve the prediction, our model should predict with higher p for $i \in \mathcal{D}_p$ and lower p for $i \in \mathcal{D}_n$ after each batch training. This necessitates assignment of higher prirority to samples $i \in \mathcal{D}_p$ with lower p and samples $i \in \mathcal{D}_n$ with higher p. Intuitively, lower confidence indicates that a model has not learned about these samples. This is achieved by employing the sampling probabilities given by equation 4 for data sample $i \in \mathcal{D}_p$ and equation 5 for data sample $i \in \mathcal{D}_n$. During each batch update, the goal is to obtain a batch of samples with this distribution and train the model incrementally on it. This stochastic prioritization will also better help to avoid overfitting the model than simply sampling greedily. While we demonstrate this aspect empirically, it can be easily understood by observing that our model simply does not pick up the top k most uncertain samples in the batch but samples based on a priority distribution. We now formalize the algorithm.

Algorithm

Algorithm 1 outlines the Relational Boosted Bandits procedure. Let us denote the data set buffer as $\mathcal{D}$. Data set $\mathcal{D}_l$ is used to train the boosting classifier and consists of the set of

Algorithm 1 Boosted Relational Bandits: Softmax Exploration with Informed Sampling

```
 1: Define $\mathcal{D} = \mathcal{D}_l$    ▷ Logged data, gathered by arbitrary policy
 2: $\mathcal{F}_0 := \text{RRTGB}(\mathcal{D}_l)$    ▷ Cold start training
 3: for batch i = 1,2,...,N do
 4:     for t = 1,2,...,T do
 5:         Observe context $x_t$
 6:         Sample $a_t \sim \mathcal{F}(p(a/x_t, \mathcal{B}))$    ▷ Softmax Action selection under learned distribution.
 7:         Receive reward $r_t \in \{0, 1\}$    ▷ Obtain 0/1 reward
 8:         $\mathcal{D}_i = \mathcal{D}_i \cup \{(x_t, a_t, r_t, p_t)\}$    ▷ Update $\mathcal{D}_i$
 9:         Update the background knowledge $\mathcal{B}$
10:     end for
11:     $\mathcal{D'}_i \sim$ Informed Sampling$(\mathcal{D}_i)$    ▷ Sample a batch of data using Algorithm2
12:     $\mathcal{F}_i := \mathcal{F}_{i-1} + \text{RRTGB}(\mathcal{D'}_i)$    ▷ Update model by adding new $K$ trees learned on $\mathcal{D'}_i$
13: end for
```

tuples represented as (x, a, r). Initially, data set $\mathcal{D}_l$ is constructed using any random policy. This is due to the lack of an informed prior on the policies. If there are informed priors as inductive biases, domain knowledge and/or constraints on the samples, they could be incorporated easily.

We learn the first set of K relational boosted trees and this is denoted by $\mathcal{F}_0$. At every time stamp t, a user arrives with context x_t and we sample action a_t(line **6**) according to softmax distribution,

$$P(a_i) = \frac{e^{(p(a_i/x_t, \mathcal{B}))/\tau}}{\sum_{j=1}^{K} e^{(p(a_j/x_t, \mathcal{B}))/\tau}}$$

In the softmax, τ controls the degree of exploration, i.e., when $\tau = 0$, the arm is chosen purely greedy. When $\tau \to \infty$, the algorithm selects randomly. Reward $r_t \in \{0, 1\}$ is then obtained for selected arm a_t. Line **8** shows that data set $\mathcal{D}_i$ is updated with new data sample (x_t, a_t, r_t, p_t). For the periodical batch-update, we sample the data set $\mathcal{D}'_{\rangle}$ according to Algorithm 2. In line **12** we incrementally update the model $\mathcal{F}_i$ by adding a new set of K relational trees to the already fitted model $\mathcal{F}_{i-1}$. These new trees are trained on sampled data set $\mathcal{D}'$. In general, at i^{th} batch update, new K trees will be added to previous $(i - 1) * K$ trees. This process is repeated as each batch of the data arrives. The final set of trees is then returned.

Algorithm 2 presents the sampling algorithm for online learning of gradient boosted RRTs. Note that this is another *important contribution of our work.* As far as we are aware, apart from the work of Huynh and Mooney (2011), this is one of the few online algorithms for SRL models and the first online algorithm for gradient boosting RRTs.

Lines **2-3** refer to the division of data set into $\mathcal{D}_p$ and $\mathcal{D}_n$. We sample the data using stochastic prioritization using Equation 4 for $i \in \mathcal{D}_p$ and Equation 5 for $i \in \mathcal{D}_n$. Lines 7 and 10 demonstrate the removal of the sampled data sample to avoid re-sampling. If a batch contains multiple occurrences of the same ground atom, the algorithm will consider these several groundings as a single data sample at training time. The data sample is removed after getting sampled from the data set to mitigate this problem. At the end, both data sets are merged into a new data set $\tilde{\mathcal{D}}$ where incremental training is performed.

Algorithm 2 Informed Sampling: Stochastic Prioritization,

```
 1: function INFORMEDSAMPLING($\mathcal{D}$)
 2:     $\mathcal{D}_p := \mathcal{D}[r = 1]$    ▷ Examples with reward 1
 3:     $\mathcal{D}_n := \mathcal{D}[r = 0]$    ▷ Examples with reward 0
 4:     for k=1 to K/2 do    ▷ K is batchsize
 5:         Sample transition $i$ from $\mathcal{D}_p \sim P_s(i) = \frac{e^{1-p_i}}{\sum_j e^{1-p_j}}$    ▷ Sample transition from positive examples
 6:         $\tilde{\mathcal{D}}_p := \mathcal{D}_p \cup i$
 7:         $\mathcal{D}_p := \mathcal{D}_p \setminus \{i\}$    ▷ Remove for avoiding re-sampling of grounding examples
 8:         Sample transition $k$ from $\mathcal{D}_n \sim P_s(k) = \frac{e^{p_k}}{\sum_j e^{p_j}}$    ▷ Sample transition from negative examples.
 9:         $\tilde{\mathcal{D}}_n := \tilde{\mathcal{D}}_n \cup k$
10:         $\mathcal{D}_n := \mathcal{D}_n \setminus \{k\}$    ▷ Remove for avoiding re-sampling of grounding examples
11:     end for
12:     $\tilde{\mathcal{D}} := \tilde{\mathcal{D}}_p \cup \tilde{\mathcal{D}}_n$    ▷ Merge sampled data sets
13:     return $\tilde{\mathcal{D}}$
14: end function
```

Explainability of the Model

A natural question arises about the interpretability and explainability of the learned model. This is particularly true because our underlying model is based on boosting. A key advantage of our model is that this underlying combination functions of these trees is a **sum**. Hence, these trees could be combined analytically similar to how First-order decision diagrams could be added (Joshi, Kersting, and Khardon 2009). The key difference is that unlike the more general relational structures, our trees have single path semantics, which makes the combination more cumbersome. For instance, multiple instances of the same object in different trees require unification and this needs to be performed repeatedly in domains with a small number of predicates. Finally, one could also approximate these sums by removing branches where the differences in regression values are under a predefined threshold δ.

We take a more empirical approach suggested by Craven and Shavlik (1995) who proposed constructing tree structured representations of neural nets. The high-level idea is to train a neural network and then make predictions on the training data. Given these predictions, one could simply learn a tree structured model on these relabeled data (based on the original model). We take a similar approach. After the boosted model is trained, we make predictions on the entire data set and then use the predictions to train a single relational regression tree (TILDE tree). This tree has the distinct advantage of being explainable. We demonstrate such trees learned in our domains. We now discuss our experiments.

data set	Number of Facts	Target
Simulated Movie	85565	willclick
IMDB	938	Workunder
Movie Lense	166486	like
ICML Co-Author	1395	CoAuthor
Drug Interaction	1774	interaction
Sports NELL	7824	teamplayssport

Table 1: Description of data sets that used in experiments.

data set	Batch Size	Trees per batch (K)
Simulated Movie Lense	256	8
IMDB	128	6
Movie Lense	256	5
ICML Co-Author	128	8
Drug Interaction	128	5
Sports NELL	128	6

Table 2: Hyperperameters used in experiments

Empirical Evaluation

Our experimental evaluation explicitly aims to answer the following specific questions:

1. **Effectiveness:** How does *RB2* perform against other baselines on real-world data?
2. **Necessity of a complex model:** How does *RB2* perform against linear approximation model?
3. **Interpretability:** Can the resulting model of *RB2* be interpretable?

Data Sets

We assessed the empirical performance of our model on a synthetic Movie Recommendation data set and five real-world relational data sets (Table1). For the simulated movie data set, we created predicates and relations such as *user information, good movie, friends, similar movies* etc. The goal in this domain is to predict on the user clicking a suggested movie. For this work, we created about 80k facts to allow for testing the scale of the learning model. We now explain the real data sets. Movie Lense data set (Motl and Schulte 2015) contains predicates that cover the relations such as *user age, movietype, movie rating* etc. The goal is to predict genres of the movies. Drug-Drug Interaction(DDI) (Dhami et al. 2018) has information like *Enzyme, Transporter, EnzymeInducer* etc. The goal is to predict the interaction of two drugs. ICML Co-author (Dhami et al. 2020) includes *affiliation, research interests,location* etc. The goal is to predict if two persons worked together in a paper. IMDB (Mihalkova and Mooney 2007) contains predicates and relations such as *Gender, Genre, Movie, Director* etc. We predict the target *WorkUnder*, i.e., predict if an actor works under a director. We also employ the sports data of Never Ending Language Learner (NELL) data set (Mitchell et al. 2018) that includes *information of players, sports, league information*. The goal is to predict which specific sport does a particular team plays.

Benchmark Algorithms

We compared our *RB2* algorithm with following algorithms:

1. Batch RRTGB (No exploration): We consider an online SRL learning variant of RRTGB RRTGB as a baseline. The model learns a set of trees incrementally on a batch of data, as explained in the algorithm section without any exploration. We used random sampling to sample a batch of data for batch training. This will allow to establish if **exploration indeed helps** in learning a better model.
2. Epsilon Greedy RRTGB: $\epsilon-$greedy is the standard baseline for multi-armed bandits. We use batch RRTGB as a base learner with $\epsilon-$greedy exploration. Action selection will be made as described in equation 6. With probability ϵ, we explore uniformly selected random action among all K actions. Else we exploit the best action among all K actions. This is a **relational epsilon greedy baseline**.

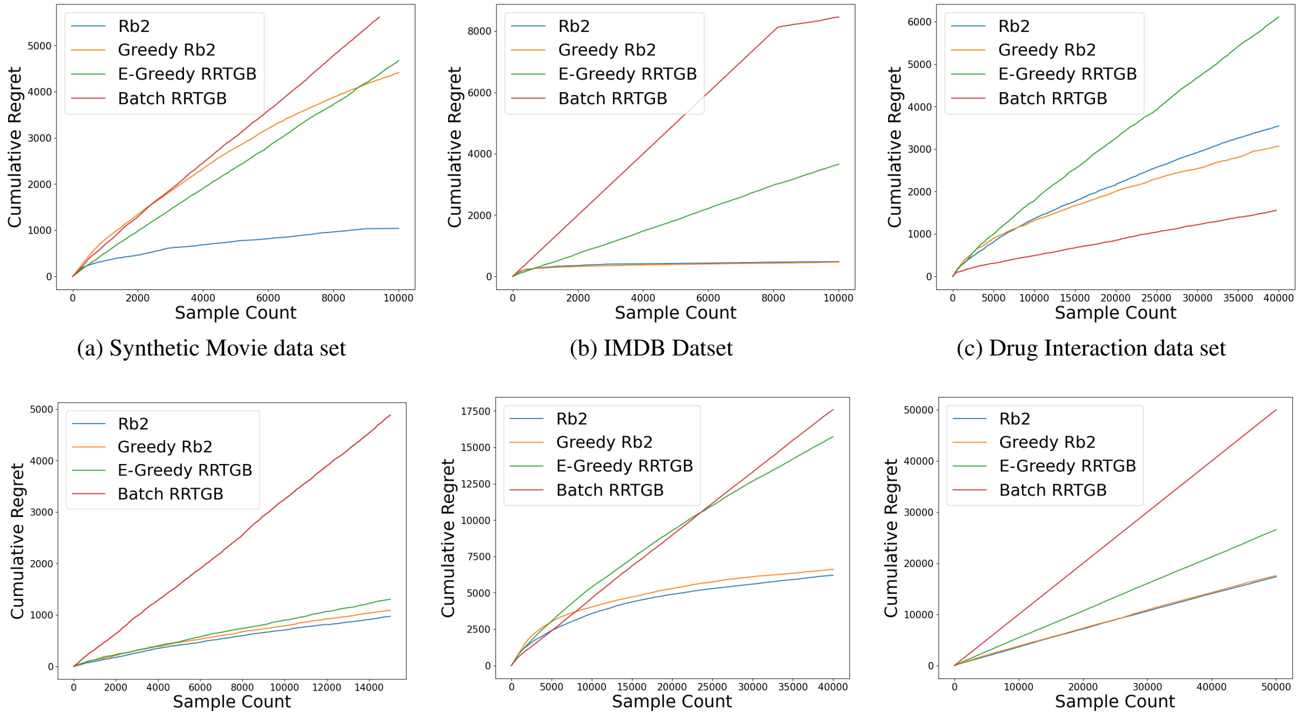
$$a := \begin{cases} \arg\max_a \mathcal{F}_{t-1}(\text{target}(x_t, a)) & \text{with } 1-\epsilon \\ \text{a uniformly random action} & \text{with } \epsilon \end{cases} \quad (6)$$

3. Greedy *RB2*: This is a variant of the proposed *RB2* algorithm. To demonstrate the **effectiveness of informed sampling** described in algorithm 2, we compared it with greedy sampling described in algorithm 3. In greedy sampling, we act greedily by picking the best K data points by sorting them on probability p_i.

In relational domains, typically, there could be multiple groundings of the same predicate. For instance, *ActedIn(P,M)* could yield multiple movies M for the same actor P. Therefore, we constructed a contextual bandit problem for the relational classification data in the following way: a regret is counted as 0 (else 1) if and only if the algorithm predicts one of the correct label (i.e., similar to the multi-label classification) of the given data instance correctly. A similar framework in propositional domains is widely adapted in the literature (Bietti, Agarwal, and Langford 2018; Agarwal et al. 2014; Elmachtoub et al. 2017).

Algorithm 3 Greedy Sampling

```
function GreedySampling($\mathcal{D}$)
    $\mathcal{D}_p := \mathcal{D}[r = 1]$          ▷ Set of examples with reward 1
    $\mathcal{D}_n := \mathcal{D}[r = 0]$          ▷ Set of examples with reward 0
    Sort the data set $\mathcal{D}_n$ according to $1 - {p_i}'s$
    Sort the data set $\mathcal{D}_p$ according to $p_i' s$
    $\tilde{\mathcal{D}}_p := \mathcal{D}_p[0 : K/2]$          ▷ Pick first K/2 data points
    $\tilde{\mathcal{D}}_n := \mathcal{D}_n[0 : K/2]$          ▷ Pick first K/2 data points
    $\tilde{\mathcal{D}} := \tilde{\mathcal{D}}_p \cup \tilde{\mathcal{D}}_n$          ▷ Merge both sampled data set
    return $\tilde{\mathcal{D}}$
end function
```



(a) Synthetic Movie data set

(b) IMDB Datset

(c) Drug Interaction data set

(d) Movie Lense data set

(e) ICML Co-Author data set

(f) NELL-Sports data set

Figure 1: Comparison of cumulative sum of regret on various relational data sets. Regret of 0 iff algorithm predicts correct labels for a given data point, otherwise 1.

Experimental Results

[Q1] How does *RB2* perform against benchmark algorithm on real-world data sets?

Figure 1 presents the cumulative regret of *RB2* with other algorithms on synthetic and real-world data sets. The hyperparameter description that has been employed in our experiments are present in Table 2. In all cases, except the Drug interaction data set, *RB2* achieves an equal performance or better than all other baselines. Note that in drug interaction, even batch RRTGB without any exploration outperforms all others. One of the caveats is that the data set is relatively easy to learn with only a few initial batches of data forming a representative sample of the whole data set. Therefore, very little exploration is required for an initial set of weak learners to find an optimal solution. Greedy *RB2* also shows similar performance on NELL and IMDB data sets. Because after enough training data, *RB2* eventually learned the data distribution. Thus it does not require the stochasticity needed for exploration in samples. Except for the drug interaction data set, batch RRTGB without exploration has converged to a lower asymptote. Thus the batch learning method failed to learn the accurate context-reward distribution. Our analysis showed that this happens due to overfitting on a given batch sample. The lack of exploration to find the truth labels could be the key reason for overfitting.

[Q2] How does *RB2* perform against linear approximation model?

Next, we compare our *RB2* framework against the classic propositional contextual bandit benchmark *LinUCB* (Li et al. 2010) on IMDB data set. Note that this is just a representative data set and chosen to highlight why relational models are necessary. This experiment's underlying motivation is to evaluate the RRTGB framework's performance against the linear approximation model in the relational domain. *LinUCB* fits the linear regression model on context-reward relationship for each action. We choose the action with the highest upper confidence bound among all actions with respect to given new context's calculated probability of getting $r = 1$.

First, we convert the relational data into the flat-vector representations. We use binary vector representations to encode relations into a feature vector. The resulting feature vector will be very sparse due to the polynomial numbers of possibility of relations. For example, in *Friends(A,B)* relation with a total of n people, $\binom{n}{2}$ combinations are possible, which is approx. $O(n^2)$. For all the entities and relationships in the domain, we encode into binary vector and perform *LinUCB* on it.

Figure 2 shows the cumulative regret of *RB2* and *LinUCB* on IMDB data set. The α parameter governs the exploration-exploitation trade-off in *LinUCB*. *RB2* clearly outperforms the *LinUCB*. This also shows the importance of using tree learners for binary response data. While the necessity of relational learning methods is well established in litera-

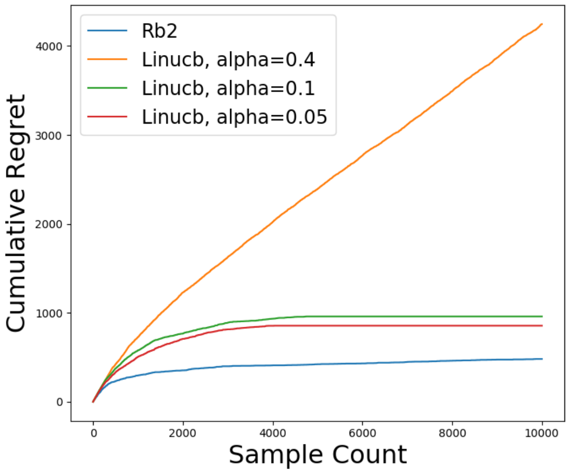


Figure 2: Comparison of *RB2* with *LinUCB* on IMDB data set. The performance of *LinUCB* is measured on different values of hyper parameter $\alpha = 0.05, 0.1, 0.4$

ture (Getoor and Taskar 2007), it is necessary to demonstrate this effectiveness in the context of bandits.

[Q3] How effective is *RB2* in terms of interpretability?

Figure 3 represents the estimated context-reward distribution on the ICML dataset, learned as a single relational regression tree. This tree represents the sum of all the RRTs. The nodes except at the leaf represent the predicates and conjunctions of literals. The leaf nodes represent the probability values of receiving reward $r = 1$ for predicting whether author A and B have worked together. If the predicate in an interior node is true, then we traverse the left path, otherwise the right path.

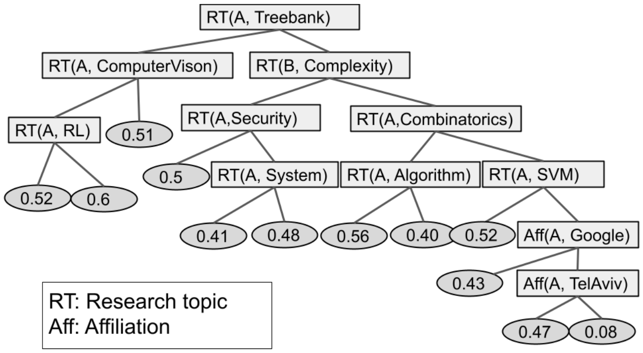


Figure 3: Example of a reward prediction model learnt by *RB2* on ICML dataset. The target here is Co-Author(A,B)

Conclusion

We presented *RB2* , a novel contextual bandit algorithm for online learning in relational domains. We use gradient-boosted relational trees as a base learner and softmax exploration for the exploration-exploitation trade-off. We also proposed a parametric-free sampling algorithm that is suitable for online relational learning. We empirically showed the performance of *RB2* with other benchmark algorithms on the cumulative regret evaluation metric and presented an interpretable tree for evaluation. Considering other exploration techniques for efficient exploration can result in efficient learning could be an interesting direction. Combining active learning strategies with efficient exploration strategies can result in a powerful human-in-the-loop system for online learning. To achieve true human-in-the-loop learning, it is essential that the learned models are explained. Going beyond the empirical combination and constructing an analytical additive combination is an interesting direction for future research.

Acknowledgements

AK acknowledges the support of Verisk AI Faculty Research Award to BR. SN gratefully acknowledges the support of AFOSR award FA9550-18-1-0462 and ARO award W911NF2010224. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the AFOSR, ARO or the US government.

Ethics Statement

Our goal is to develop a new online learning algorithm for learning from noisy, structured data. The major impact that our work aims to create is developing a bandit learner for multi-relational data. Our work has the potential to be employed in large-scale relational classification tasks such as entity resolution and relation extraction. Specifically, we anticipate the use of such algorithms in the context of recommendation systems. Rigorous evaluation on larger data can enable this adaptation.

As far as we are aware, there is no major ethical impact of this work. The only minor negative impacts could be the misinterpretation of the rules by experts but this is a standard issue with any rule learning method. Beyond this aspect, we do not anticipate any major negative societal impact.

References

Agarwal, A.; Hsu, D.; Kale, S.; Langford, J.; Li, L.; and Schapire, R. 2014. Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits. In *Proceedings of Machine Learning Research*, volume 32, 1638–1646. PMLR.

Bietti, A.; Agarwal, A.; and Langford, J. 2018. A contextual bandit bake-off. *arXiv preprint arXiv:1802.04064* .

Blockeel, H.; and De Raedt, L. 1998. Top-down induction of first-order logical decision trees. *Artificial Intelligence* 101(1): 285 – 297.

Bouneffouf, D.; and Rish, I. 2019. A Survey on Practical Applications of Multi-Armed and Contextual Bandits. *CoRR* abs/1904.10040.

Craven, M. W.; and Shavlik, J. W. 1995. Extracting Tree-Structured Representations of Trained Networks. In *NIPS*, 24–30.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI*, 2468–2473.

Dhami, D. S.; Kunapuli, G.; Das, M.; Page, D.; and Natarajan, S. 2018. Drug-Drug Interaction Discovery: Kernel Learning from Heterogeneous Similarities. *Smart Health* 9-10: 88 – 100.

Dhami, D. S.; Soni, A.; Page, D.; and Natarajan, S. 2017. Identifying Parkinson's Patients: A Functional Gradient Boosting Approach. In *AIME*, 332–337.

Dhami, D. S.; Yan, S.; Kunapuli, G.; and Natarajan, S. 2020. Non-Parametric Learning of Gaifman Models. *CoRR* abs/2001.00528.

Durand, A.; Achilleos, C.; Iacovides, D.; Strati, K.; Mitsis, G. D.; and Pineau, J. 2018. Contextual Bandits for Adapting Treatment in a Mouse Model of de Novo Carcinogenesis. In *Proceedings of Machine Learning Research*, volume 85, 67–82.

Elmachtoub, A. N.; McNellis, R.; Oh, S.; and Petrik, M. 2017. A Practical Method for Solving Contextual Bandit Problems Using Decision Trees. *CoRR* abs/1706.04687.

Friedman, J. H. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29: 1189–1232.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning Probabilistic Relational Models. In *IJCAI*, 1300–1307.

Getoor, L.; and Taskar, B. 2007. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. ISBN 0262072882.

Huynh, T. N.; and Mooney, R. J. 2011. Online Structure Learning for Markov Logic Networks. In Gunopulos, D.; Hofmann, T.; Malerba, D.; and Vazirgiannis, M., eds., *Machine Learning and Knowledge Discovery in Databases*, 81–96.

Joshi, S.; Kersting, K.; and Khardon, R. 2009. Generalized First Order Decision Diagrams for First Order Markov Decision Processes 1916–1921.

Kersting, K.; and De Raedt, L. 2007. Bayesian logic programming: theory and tool. *Statistical Relational Learning* 270–321.

Langford, J.; and Zhang, T. 2008. The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information. In *NIPS*, 817–824.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *WWW*, 661–670.

Liu, B.; Yu, T.; Lane, I.; and Mengshoel, O. 2018. Customized Nonlinear Bandits for Online Response Selection in Neural Conversation Models. In *AAAI*, 5245–5252.

MacLeod, H.; Yang, S.; Oakes, K.; Connelly, K.; and Natarajan, S. 2016. Identifying Rare Diseases from Behavioural Data: A Machine Learning Approach. In *IEEE CHASE*, 130–139.

Mahajan, D. K.; Rastogi, R.; Tiwari, C.; and Mitra, A. 2012. LogUCB: An Explore-Exploit Algorithm for Comments Recommendation. In *CIKM*, 6–15.

Mihalkova, L.; and Mooney, R. J. 2007. Bottom-up Learning of Markov Logic Network Structure. In *ICML*, 625–632.

Mitchell, T.; Cohen, W.; Hruschka, E.; Talukdar, P.; Yang, B.; Betteridge, J.; Carlson, A.; Dalvi, B.; Gardner, M.; Kisiel, B.; Krishnamurthy, J.; Lao, N.; Mazaitis, K.; Mohamed, T.; Nakashole, N.; Platanios, E.; Ritter, A.; Samadi, M.; Settles, B.; Wang, R.; Wijaya, D.; Gupta, A.; Chen, X.; Saparov, A.; Greaves, M.; and Welling, J. 2018. Never-Ending Learning. *Commun. ACM* 61: 103–115.

Motl, J.; and Schulte, O. 2015. The CTU Prague Relational Learning Repository. *CoRR* abs/1511.03086.

Natarajan, S.; Joshi, S.; Saha, B. N.; Edwards, A.; Khot, T.; Moody, E.; Kersting, K.; Whitlow, C. T.; and Maldjian, J. A. 2012a. A Machine Learning Pipeline for Three-Way Classification of Alzheimer Patients from Structural Magnetic Resonance Images of the Brain. In *ICMLA*, 659–669.

Natarajan, S.; Joshi, S.; Tadepalli, P.; Kersting, K.; and Shavlik, J. W. 2011. Imitation Learning in Relational Domains: A Functional-Gradient Boosting Approach. In *IJCAI*, volume 8812, 64–75.

Natarajan, S.; Khot, T.; Kersting, K.; Gutmann, B.; and Shavlik, J. W. 2012b. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Mach. Learn.* 86(1): 25–56.

Neville, J.; and Jensen, D. 2007. Relational Dependency Networks 653–692.

Richardson, M.; and Domingos, P. 2006. Markov Logic Networks. *Mach. Learn.* 62: 107–136.

Yang, S.; Korayem, M.; AlJadda, K.; Grainger, T.; and Natarajan, S. 2017. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational Learning approach. *Knowl. Based Syst.* 37–45.

Zhou, L. 2015. A Survey on Contextual Multi-armed Bandits. *CoRR* abs/1508.03326.