

# A Learning Algorithm for Relational Logistic Regression: Preliminary Results\*

**Bahare Fatemi, Seyed Mehran Kazemi and David Poole**

The University of British Columbia  
Vancouver, BC, V6T 1Z4  
{bfatemi, smkazemi, poole}@cs.ubc.ca

## Abstract

Relational logistic regression (RLR) is a representation of conditional probability in terms of weighted formulae for modelling multi-relational data. In this paper, we develop a learning algorithm for RLR models. Learning an RLR model from data consists of two steps: 1- learning the set of formulae to be used in the model (a.k.a. structure learning) and learning the weight of each formula (a.k.a. parameter learning). For structure learning, we deploy Schmidt and Murphy's hierarchical assumption: first we learn a model with simple formulae, then more complex formulae are added iteratively only if all their sub-formulae have proven effective in previous learned models. For parameter learning, we convert the problem into a non-relational learning problem and use an off-the-shelf logistic regression learning algorithm from Weka, an open-source machine learning tool, to learn the weights. We also indicate how hidden features about the individuals can be incorporated into RLR to boost the learning performance. We compare our learning algorithm to other structure and parameter learning algorithms in the literature, and compare the performance of RLR models to standard logistic regression and RDN-Boost on a modified version of the MovieLens data-set.

Statistical relational learning (SRL) (De Raedt et al., 2016) aims at unifying logic and probability to provide models that can learn from complex multi-relational data. Relational probability models (RPMs) (Getoor and Taskar, 2007) (also called template-based models (Koller and Friedman, 2009)) are the core of the SRL systems. They extend Bayesian networks and Markov networks (Pearl, 1988) by adding the concepts of objects, properties and relations, and by allowing for probabilistic dependencies among relations of individuals.

Unlike Bayesian networks, in RPMs a random variable may depend on an unbounded number of parents in the grounding. In these cases, the conditional probability cannot be represented as a table. To address this issue, many of the existing relational models (e.g., (De Raedt, Kimmig, and Toivonen, 2007; Natarajan et al., 2012)) use simple aggregation models such as existential quantifiers, or noisy-or models. These models are much more compact than tabular

representations. There also exists other aggregators with different properties (e.g., see (Horsch and Poole, 1990; Friedman et al., 1999; Neville et al., 2005; Perlish and Provost, 2006; Kisynski and Poole, 2009; Natarajan et al., 2010)).

Relational logistic regression (RLR) (Kazemi et al., 2014) has been recently proposed as an aggregation model which can represent much more complex functions than the previous aggregators. RLR uses weighted formulae to define a conditional probability. Learning an RLR model from data consists of a structure learning and a parameter learning phase. The former corresponds to learning the features (weighted formulae) to be included in the model, and the latter corresponds to learning the weight of each feature. When all of the parents are observed (e.g., for classification), Poole et al. (2014) observed that an RLR model has similar semantics as a Markov logic network (MLN) (Richardson and Domingos, 2006). Therefore, one can use a discriminative learning algorithm for MLNs to learn an RLR model.

Huynh and Mooney (2008) proposed a bottom-up algorithm for discriminative learning of MLNs. They use a logic program learner (ALEPH (Srinivasan, 2001)) to learn the structure, and then use L1-regularized logistic regression to learn the weights and enable automatic feature selection. The problem with this approach is that the ALEPH (or any other logic program learner) and the MLN have different semantics: features marked as useful by ALEPH are not necessarily useful features for an MLN, and features marked as useless by ALEPH are not necessarily useless features for MLN. The former happens because ALEPH generates features with logical accuracy which is only a rough estimate of their value in a probabilistic model. The latter happens because the representational power of an MLN is more than that of ALEPH: MLNs can leverage counts but ALEPH is based on the existential quantifier. While the former issue may be resolved using L1-regularization, it is not straightforward to resolve the latter issue.

In this paper, we develop and test an algorithm for learning RLR models from relational data which addresses the aforementioned issues with current learning algorithms. Our learning algorithm follows the hierarchical assumption of (Schmidt and Murphy, 2010): a formula  $a \wedge b$  may be a useful feature only if  $a$  and  $b$  have each proven useful. Our learning algorithm is, in spirit, similar to the refinement graphs of Popescul and Ungar (2004). In our algorithm,

\*In IJCAI-16 Statistical Relational AI Workshop.  
Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

however, feature generation is done using hierarchical assumption instead of query refinement, and feature selection is done for each level of hierarchy using hierarchical (or L1) regularization instead of sequential feature selection, thus reducing the number of times a model is learned from data and enabling previously selected features to be removed as the search proceeds.

We also incorporate hidden features about each individual in our RLR model and observe how they affect the performance. We test our learning algorithm on a modified version of the MovieLens data-set taken from Schulte and Khosravi (2012) for predicting users' gender and age. We compare our model with standard logistic regression models that do not use relational features, as well as the RDN-Boost (Natarajan et al., 2012) which is one of the state-of-the-art relational learning algorithms. The obtained results show that RLR can learn more accurate models compared to the standard logistic regression and RDN-Boost. The results also show that adding hidden features to RLR may increase the accuracy of the predictions, but may make the model over-confident about its predictions. Regularizing the RLR predictions towards the mean of the data helps avoid over-confidence.

## Background and Notations

In this section, we introduce the notation used throughout the paper, and provide necessary background information for readers to follow the rest of the paper.

### Logistic Regression

Logistic regression (LR) (Allison, 1999) is a popular classification method within machine learning community. We describe how it can be used for classification following Cessie and van Houwelingen (1992) and Mitchell (1997).

Suppose we have a set of labeled examples  $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ , where each  $x_i$  is composed of  $n$  features  $x_{i1}, x_{i2}, \dots, x_{in}$  and each  $y_i$  is a binary variable whose value is to be predicted. The  $x_{ij}$ s may be binary, multi-valued or continuous. Throughout the paper, we assume binary variables take their values from  $\{0, 1\}$ . Logistic regression learns a set  $w = \{w_0, w_1, \dots, w_n\}$  of weights, where  $w_0$  is the intercept and  $w_j$  is the weight of the feature  $x_{ij}$ . For simplicity, we assume a new dimension  $x_{i0} = 1$  has been added to the data to avoid treating  $w_0$  differently than other  $w_j$ s. LR defines the probability of  $y_i$  being 1 given  $x_i$  as follows:

$$P(y_i = 1 | x_i, w) = \sigma\left(\sum_{j=0}^n x_{ij}w_j\right) \quad (1)$$

where  $\sigma(x) = \frac{1}{1+\exp(-x)}$  is the Sigmoid function.

Logistic regression learns the weights by maximizing the log-likelihood of the data (or equivalently, minimizing the logistic loss function) as follows:

$$w_{LR} = \operatorname{argmax}_w \sum_{i=0}^m \log(P(y_i = 1 | x_i)) \quad (2)$$

An L1-regularization can be added to the loss function to encourage sparsity and do an automatic feature selection.

## Conjoined Features and the Hierarchical Assumption

Given  $n$  input random variables, logistic regression considers  $n + 1$  features: one bias (intercept) and one feature for each random variable. One can generate more features by conjoining the input random variables. For instance, if  $a$  and  $b$  are two continuous random variables, one can generate a new feature  $a * b$  (which is  $a \wedge b$  for Boolean variables). Given  $n$  random variables, conjoining (or multiplying) variables allows for generating  $2^n$  features. These  $2^n$  weights can represent arbitrary conditional probabilities<sup>1</sup>, in what is known as the canonical representation - refer to Buchman et al. (2012) or Koller and Friedman (2009). For a large  $n$ , however, generating  $2^n$  features may not be practically possible, and it also makes the model overfit to the training data.

In order to avoid generating all  $2^n$  features, Schmidt and Murphy (2010) make a hierarchical assumption: if either  $a$  or  $b$  are not useful features, neither is  $a \wedge b$ . Having this assumption, Schmidt and Murphy (2010) first learn an LR model considering only features with no conjunctions. They regularize their loss function with a hierarchical regularization function, so that the weights of the features not contributing to the prediction go to zero. Once the learning stops, they keep the features having non-zero weights, and add all conjoined features whose all subsets have non-zero weights in the previous step. Then they run their learning again. They continue this process until no more features can be added.

### Relational Logistic Regression

Relational logistic regression (RLR) (Kazemi et al., 2014) is the analogue of LR for relational models. RLR can be also considered as the directed analogue of Markov logic networks (Richardson and Domingos, 2006). In order to describe RLR, first we need to introduce some definitions and terminologies used in relational domains.

A **population** refers to a set of **individuals** and corresponds to a domain in logic. **Population size** of a population is a non-negative number indicating its cardinality. For example, a population can be the set of planets in the solar system, where *Mars* is an individual and the population size is 8.

**Logical variables** start with lower-case letters, and **constants** start with upper-case letters. Associated with a logical variable  $x$  is a population  $pop(x)$  where  $|x| = |pop(x)|$  is the size of the population. A lower-case and an upper-case letter written in bold refer to a set of logical variables and a set of individuals respectively.

A **parametrized random variable (PRV)** (Poole, 2003) is of the form  $F(t_1, \dots, t_k)$  where  $F$  is a  $k$ -ary (continuous or categorical) function symbol and each  $t_i$  is a logical variable or a constant. If all  $t_i$ s are constants, the PRV is a random variable. If  $k = 0$ , we can omit the parentheses. If  $F$  is a predicate symbol,  $F$  has range  $\{\text{True}, \text{False}\}$ , otherwise

<sup>1</sup>Note that there are  $2^n$  degrees of freedom and any representation that can represent arbitrary conditional probabilities may require  $2^n$  parameters. The challenge is to find a representation that can often use fewer.

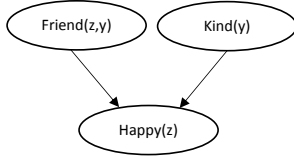


Figure 1: A relational model taken from (Kazemi et al., 2014).

the range of  $F$  is the range of the function. For example,  $LifeExistsOn(planet)$  can be a PRV with predicate function  $LifeExistsOn$  and logical variable  $planet$ , which is true if life exists on the given  $planet$ .

A **literal** is an assignment of a value to a PRV. We represent  $F(\cdot) = true$  as  $f(\cdot)$  and  $F(\cdot) = false$  as  $\neg f(\cdot)$ . A **formula** is made up of literals connected with conjunction or disjunction.

A **weighted formula (WF)** for a PRV  $Q(\mathbf{z})$ , where  $\mathbf{z}$  is a set of logical variables, is a tuple  $\langle F, w \rangle$  where  $F$  is a Boolean formula of the parents of  $Q$  and  $w$  is a weight.

**Relational logistic regression (RLR)** defines a conditional probability distribution for a Boolean PRV  $Q(\mathbf{z})$  using a set of WFs  $\psi$  as follows:

$$P(q(\mathbf{Z}) | \Pi) = \sigma \left( \sum_{\langle F, w \rangle \in \psi} w * F_{\Pi, \mathbf{z} \rightarrow \mathbf{Z}} \right) \quad (3)$$

where  $\sigma(x) = \frac{1}{1 + \exp(-x)}$  is the Sigmoid function,  $\Pi$  represents the assigned values to parents of  $Q$ ,  $\mathbf{Z}$  represents an assignment of individuals to the logical variables in  $\mathbf{z}$ , and  $F_{\Pi, \mathbf{z} \rightarrow \mathbf{Z}}$  is formula  $F$  with each logical variable  $\mathbf{z}$  in it being replaced according to  $\mathbf{Z}$ , and evaluated in  $\Pi$ .

**Example 1.** Consider the relational model in Fig. 1 taken from (Kazemi et al., 2014) and suppose we want to model "someone is happy if they have at least 5 friends that are kind". The following WFs can be used to represent this model:

$$\langle True, -4.5 \rangle$$

$$\langle friend(z, y) \wedge kind(y), 1 \rangle$$

RLR sums over the above WFs resulting in:

$$\forall Z \in z: P(Happy(Z) = True | Friend(Z, y), Kind(y)) = \text{Sigmoid}(-4.5 + 1 * n_T)$$

where  $n_T$  represents the number of individuals in  $y$  for which  $Friend(Z, y) \wedge Kind(y)$  is true. When  $n_T \geq 5$ , the probability is closer to one than zero and when  $n_T < 5$ , the probability is closer to zero than one.

## Handling Continuous Variables

RLR was initially designed for Boolean or multi-valued parents. If *True* is associated with 1 and *False* is associated with 0, we can substitute  $\wedge$  in our WFs with  $*$ . Then we can allow continuous PRVs in WFs. For example if for some  $X \in x$  we have  $R(X) = True$ ,  $S(X) = False$  and  $T(X) = 0.2$ , then  $\langle r(X) * s(X), w \rangle$  evaluates to 0,  $\langle r(X) * \neg s(X), w \rangle$  evaluates to  $w$ , and  $\langle r(X) * s(X) * t(X), w \rangle$  evaluates to  $0.2 * w$ .

## Learning Relational Logistic Regression

The aforementioned learning algorithm for LR is not directly applicable to RLR because the number of potential weights in RLR is unbounded. We develop a learning algorithm for RLR which can handle the unbounded number of potential weights.

Learning RLR from data consists of two parts: structure learning (learning the set of WFs) and parameter learning (learning the weight of each WF).

### Parameter Learning

Suppose we are given a set  $\psi$  of WFs defining the conditional probability of a PRV  $Q(\mathbf{z})$ , and we want to learn the weight of each WF. We can convert this learning problem into a LR learning problem by generating a flat data-set in single-matrix form. To do so, for each assignment  $\mathbf{Z}$  of individuals to the logical variables in  $\mathbf{z}$ , we generate one data row  $\langle x_{i1}, x_{i2}, \dots, x_{i|\psi|}, y_i \rangle$  in which  $y_i = Q(\mathbf{Z})$  and  $x_{ij}$  is the number of times the formula of the  $j$ -th WF in  $\psi$  is true when the logical variables in  $\mathbf{z}$  are replaced with the individuals in  $\mathbf{Z}$ . Once we do this conversion, we have a single-matrix data for which we can learn a LR model. The weight learned for the  $j$ -th input gives the weight for our  $j$ -th WF. Note that the conversion is complete and specifies the values of all relations.

**Example 2.** Consider the relational model in Fig. 1 and suppose we are given the following WFs:

$$\langle True, w_1 \rangle$$

$$\langle friend(z, y), w_2 \rangle$$

$$\langle friend(z, y) * kind(y), w_3 \rangle$$

In this case, we generate a matrix data having a row for each individual  $Z \in z$  where the row consists of four values: the first number is 1 serving as the intercept, the second one is the number of people that are friends with  $Z$ , the third is the number of kind people that are friends with  $Z$ , and the fourth one represents whether  $Z$  is happy or not. These are sufficient statistics for learning the weights. An example of the generated matrices is as follows:

Bias	#Friends	#Kind Friends	Happy?
1	5	3	Yes
1	18	2	No
1	1	1	Yes
1	12	10	Yes
...	...	...	...

In the above matrix, the first four people have 5, 18, 1 and 12 friends respectively, out of which 3, 2, 1, and 10 are kind.

### Structure Learning

Learning the structure of an RLR model refers to selecting a set of WFs that should be used. By conjoining different relations and adding attributes, one can generate an infinite number of WFs. As an example, suppose we want to predict the gender of users ( $G(u)$ ) in a movie rating system, where we are given the occupation ( $O(u)$ ), age ( $A(u)$ ) and the movies that these users have rated ( $Rated(u, m)$ ), as well

as the set of (possibly more than one) genres that each movie belongs to. Our WFs can have formulae such as:

$$\begin{aligned} a(u) &= \text{young} \\ \text{rated}(u, m) * \text{drama}(m) \\ \text{rated}(u, m) * \text{rated}(u', m) * g(u') &= \text{male} \end{aligned}$$

and many other WFs with much more conjoined relations. Not all of these WFs may be useful though. As an example, a WF whose formula is  $\text{action}(m)$  is not a useful feature in predicting  $G(u)$  as it evaluates to a constant number for all users. We avoid generating such features in an RLR learning model. To do this in a systematic way, we need a few definitions:

**Definition 1.** Let  $\psi$  denote the set of WFs defining the conditional probability of a PRV  $Q(\mathbf{z})$ . A logical variable  $v$  in a formula  $f$  of a WF  $\in \psi$  is a:

- **target** if  $v \in \mathbf{z}$
- **connector** for  $v_1$  and  $v_2$  if there are at least two relations in  $f$  one having  $v$  and  $v_1$  and the other having  $v$  and  $v_2$
- **attributed** if there exists at least one PRV in  $f$  having only  $v$  (e.g.,  $g(v)$ )
- **hanging** if it fits in none of the above definitions

**Definition 2.** A WF is a *chain* (Schulte and Khosravi, 2012) if its literals can be ordered as a list  $[r_1(\mathbf{x}_1), \dots, r_k(\mathbf{x}_k)]$  such that each literal  $r_{i+1}(\mathbf{x}_{i+1})$  shares at least one logical variable with the preceding literals  $r_1(\mathbf{x}_1), \dots, r_i(\mathbf{x}_i)$ . A chain is *targeted* if it has at least one target logical variable. A WF is *k-BL* if it contains no more than  $k$  binary literals, and is *r-UL* if it contains no more than  $r$  unary literals.

**Example 3.** Suppose  $\langle \text{rated}(u, m) * \text{rated}(u', m) * \text{rated}(u', m') * \text{comedy}(m), w \rangle$  is a WF belonging to the set of WFs defining the conditional probability of  $G(u)$ . Then  $u$  is a target,  $m$  and  $u'$  are connectors,  $m$  is also attributed, and  $m'$  is a hanging logical variable. This WF is a chain because the second literal shares a  $m$  with the first literal, the third one shares a  $u'$  with the second one, and the fourth one shares a  $m$  with the first and second literal. This chain is targeted because it contains  $u$ , which is the only target logical variable. The WF is 3-BL, 1-UL as it contains no more than 3 binary and no more than 1 unary literals.  $\langle \text{age}(u) = \text{young} * \text{comedy}(m), w \rangle$  is a non-chain WF, and  $\langle \text{drama}(m) * \text{comedy}(m), w \rangle$  is a chain which is not targeted.

We avoid generating two types of WFs: 1- WFs that are not targeted chains, 2- WFs that contain hanging logical variables. We do this because non-targeted chains (e.g.,  $\langle \text{drama}(m) * \text{comedy}(m), w \rangle$  for predicting  $G(u)$ ) always evaluates to a constant number, and WFs with hanging logical variables (e.g.,  $\langle \text{rated}(u, m) * \text{acted}(a, m), w \rangle$  for predicting  $G(u)$ ) can be replaced with more informative WFs. In the rest of the paper, we only consider the WFs that are targeted chains and have no hanging logical variables.

Having these definitions, we state the hierarchical assumption as follows:

**Hierarchical Assumption:** Let  $f$  be a  $k$ -BL,  $r$ -UL WF. Let  $\psi_f$  be the set of all  $k$ -BL,  $j$ -UL ( $j < r$ ) WFs having the same  $k$  binary literals and a strict subset of the unary literals as

$f$ .  $f$  is *useless* if L1-regularized logistic regression assigns a zero weight to it, or there exists a useless WF in  $\psi_f$ .

**Example 4.** A WF  $\langle \text{rated}(u, m) * \text{drama}(m) * \text{comedy}(m), w \rangle$  is useless if either  $\langle \text{rated}(u, m) * \text{drama}(m), w_1 \rangle$  or  $\langle \text{rated}(u, m) * \text{comedy}(m), w_2 \rangle$  is useless, or L1-regularization sets  $w$  to 0.

In order to learn the structure of an RLR model, we select a value  $k$  and generate all allowed  $k$ -BL, 1-UL WFs. We find the best value of  $k$  by cross-validation. Then we add WFs with more unary literals by making the hierarchical assumption and using a similar search strategy as in (Schmidt and Murphy, 2010) by following the algorithm below:

```

curWFs ← set of k-BL, 1-UL WFs
removedWFs ← ∅
r ← 1
while(stoppingCriteriaMet()){
  M = L1-Regularized-Logistic-Regression(curWFs)
  removedWFs += curWFs having w = 0 in M
  r ← r + 1
  curWFs = HA(removedWFs, r)
}

```

The algorithm starts with  $k$ -BL, 1-UL WFs. Initially, no WF is labeled as removed.  $r$  is initially set to 1 to indicate the current maximum number of unary literals. Then until the stopping criteria is met, the weights of the WFs in  $curWFs$  are learned using an L1-regularized logistic regression. If the weight of a WF is set to zero, we add it to the  $removedWFs$ . Then we increment  $r$  and update the  $curWFs$  to the  $k$ -BL,  $r$ -UL WFs obeying the hierarchical assumption with respect to the  $removedWFs$  (we assume  $HA(removedWFs, r)$  is a function which returns such WFs). The stopping criteria is met when no more WFs can be generated as  $r$  increases.

## Adding Hidden Features

While we exploit the observed features of the objects in making predictions, each object may contain really useful information that has not been observed. As an example, in predicting the gender of the users given the movies they liked, some movies may only be appealing to males and some only to females. Or there might be features in movies that we do not know about, but they contribute to predicting the gender of users.

In order to incorporate hidden features in our RLR model, we add continuous unary PRVs such as  $H(m)$  with (initially) random values to our dataset. Then we generate all  $k$ -BL, 1-UL WFs and learn the weights as well as the values of the hidden features using stochastic gradient descent with L1-regularization. Once we learn the values of the hidden features, we treat them as normal features and use our aforementioned structure and parameter learning algorithms to learn an RLR model.

## Experiments and Results

We test our learning algorithm on the 0.1M Movielens dataset (Harper and Konstan, 2015) with the modifications made by Schulte and Khosravi (2012). This data-set contains information about 940 users (nominal variables for age, occupation, and gender), 1682 movies (binary variables for ac-

Table 1: ACLL and accuracy on predicting the gender and age of the users in the MovieLens dataset.

		Learning Algorithm				
		Baseline	LR	RDN-Boost	RLR-Base	RLR-H
Gender	ACLL	-0.6020	-0.5694	-0.5947	-0.5368	-0.5046
	Accuracy	71.0638%	71.383%	70.6383%	73.8298%	77.3404%
Age	ACLL	-0.6733	-0.5242	-0.5299	-0.5166	-0.5090
	Accuracy	60.1064%	76.0638%	76.4893%	77.1277%	77.0212%

tion, horror, and drama), the movies rated by each user containing 79,778 user-movie pairs, and the actual rating the user has given to a movie. In our experiments, we ignored the actual ratings and only considered if a movie has been rated by a user or not.

We learned RLR models for predicting the age and gender of users once with no hidden features (RLR-Base), and once with one hidden feature (RLR-H) for the movies. We regularize the predictions of both RLR-Base and RLR-H towards the mean as:

$$Probability = \lambda * mean + (1 - \lambda) * (RLRsignal) \quad (4)$$

When predicting the age of the users, we only considered two instead of three age classes (we merged the age\_1 and age\_2 classes). For learning Logistic regression models with L1-regularization, we used the open source codes of (Schmidt, Fung, and Rosales, 2007) and we learned the final logistic regression model with Weka software (Hall et al., 2009). We compared the proposed method with a baseline model always predicting the mean, standard logistic regression (LR) not using the relational information, and the RDN-Boost.

The performance of all learning algorithms were obtained by 5-folds cross-validation. In each fold, we divided the users in the MovieLens data-set randomly into 80% training set and 20% test set. We learned the model on the train set and measured the accuracy (the percentage of correctly classified instances) and the average conditional log-likelihood (ACLL) on the test set, and averaged them over the 5 folds. ACLL is computed as follows:

$$ACLL = \frac{1}{m} \sum_{i=1}^m \ln(P(G(U_i) | data, model)) \quad (5)$$

Obtained results are represented in Table 1. They show that RLR utilizes the relational features to improve the predictions compared to the logistic regression model that does not use the relational information, and the RDB-Boost. Obtained results also represent that adding hidden features to the RLR models may increase the accuracy and reduce the MAE. However, we observed that adding hidden features makes the model over-confident by pushing the prediction probabilities towards zero and one, thus requiring more regularization towards the mean.

## Discussion

In our first experiment on predicting the gender of the users, we found that on average men have rated more action movies

than women. This means for predicting the gender of the users, the feature  $male(u) \Leftarrow rated(u, m) \wedge action(m)$  is a useful feature for both RLR and MLNs as it counts the number of action movies. Many of the current relational learning algorithms/models, however, rely mostly on the existential quantifier as their aggregator (e.g., (Horsch and Poole, 1990; De Raedt, Kimmig, and Toivonen, 2007; Huynh and Mooney, 2008; Natarajan et al., 2012)). By relying on the existential quantifier, these models either have to use many complex rules to imitate the effect of such rules, or lose great amounts of relational information available in terms of counts.

As a particular example, consider the discriminative structure learning of Huynh and Mooney (2008) for MLNs. First they learn a large set of features using ALEPH, then learn the weights of the features with L1-regularization to enable automatic feature selection. In cases where everyone has rated an action movie,  $male(u) :- rated(u, m) \wedge action(m)$  is not a useful feature for ALEPH (and so it will not find it) because it does not distinguish males from females. Therefore, this rule will not be included in the final MLN.

Relational learners based on existential quantifier can potentially imitate the effect of counts by using many complex rules. As an example,  $male(u) :- rated(u, m_1) \wedge action(m_1) \wedge rated(u, m_2) \wedge action(m_2) \wedge m_1 \neq m_2$  may be used to assign a maleness probability to people rating two action movies. But this approach requires a different rule for each count, and the rules become more and more complex as the count grows because they require pairwise inequalities. To see this in practice, we ran experiments with ALEPH on synthesized data and observed that, even though it could learn such rules to enhance its predictions, it failed at finding them.

Based on the above observations, we argue that relational learning algorithms/models need to allow for a richer set of predefined aggregators, or enable non-predefined aggregators to be learned from data similar to what RLR does. We also argue that our structure learning algorithm has the potential to explore more features, and may also be a good candidate for discriminative structure learning of MLNs.

## Conclusion

Relational logistic regression (RLR) can learn complex models for multi-relational data-sets. In this paper, we developed and tested a structure and parameter learning for these models based on the hierarchical assumption. We compared our model with the standard logistic regression model

and the RDN-Boost, and represented that, on the MovieLens data-set, RLR achieves higher accuracies. We also represented how hidden features can boost the performance of RLR models. The results presented in this work are only preliminary results. Future direction includes testing our learning algorithm on more complex data-sets having much more relational information, comparing our model with other relational learning and aggregation models in the literature, making the learning algorithm extrapolate properly for the un-seen population sizes (Poole et al., 2014), and testing the performance of our structure learning algorithm for discriminative learning of Markov logic networks.

## References

- Allison, P. 1999. *Logistic regression using SAS: theory and application*. SAS Publishing.
- Buchman, D.; Schmidt, M.; Mohamed, S.; Poole, D.; and De Freitas, N. 2012. On sparse, spectral and other parameterizations of binary probabilistic models. In *AISTATS*.
- Cessie, S. L., and van Houwelingen, J. 1992. Ridge estimators in logistic regression. *Applied Statistics* 41(1):191–201.
- De Raedt, L.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10(2):1–189.
- De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7.
- Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, 1300–1307. Sweden: Morgan Kaufmann.
- Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.
- Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter* 11(1):10–18.
- Harper, M., and Konstan, J. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5(4):19.
- Horsch, M., and Poole, D. 1990. A dynamic approach to probability inference using Bayesian networks. In *Proc. sixth Conference on Uncertainty in AI*, 155–161.
- Huynh, T. N., and Mooney, R. J. 2008. Discriminative structure and parameter learning for markov logic networks. In *Proc. of the international conference on machine learning*.
- Kazemi, S. M.; Buchman, D.; Kersting, K.; Natarajan, S.; and Poole, D. 2014. Relational logistic regression. In *Proc. 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.
- Kisynski, J., and Poole, D. 2009. Lifted aggregation in directed first-order probabilistic models. In *Twenty-first International Joint Conference on Artificial Intelligence*, 1922–1929.
- Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA.
- Mitchell, T. 1997. *Machine Learning*. McGraw Hill.
- Natarajan, S.; Khot, T.; Lowd, D.; Tadepalli, P.; and Kersting, K. 2010. Exploiting causal independence in Markov logic networks: Combining undirected and directed models. In *European Conference on Machine Learning (ECML)*.
- 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):25–56.
- Neville, J.; Simsek, O.; Jensen, D.; Komoroske, J.; Palmer, K.; and Goldberg, H. 2005. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. MIT Press.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.
- Perlish, C., and Provost, F. 2006. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning* 62:65–105.
- Poole, D.; Buchman, D.; Kazemi, S. M.; Kersting, K.; and Natarajan, S. 2014. Population size extrapolation in relational probabilistic modelling. In *Proc. of the Eighth International Conference on Scalable Uncertainty Management*.
- Poole, D. 2003. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, 985–991.
- Popescul, A., and Ungar, L. H. 2004. Cluster-based concept invention for statistical relational learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 665–670. ACM.
- Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.
- Schmidt, M. W., and Murphy, K. P. 2010. Convex structure learning in log-linear models: Beyond pairwise potentials. In *International Conference on Artificial Intelligence and Statistics*.
- Schmidt, M.; Fung, G.; and Rosales, R. 2007. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *Machine Learning: ECML 2007*. Springer. 286–297.
- Schulte, O., and Khosravi, H. 2012. Learning graphical models for relational data via lattice search. *Machine Learning* 88(3):331–368.
- Srinivasan, A. 2001. The aleph manual.