Probabilistic Rule Learning through Integer Linear Programming

Radomír Černoch and Filip Železný

Department of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University Prague, Czech Republic
{cernorad,zelezny}@fel.cvut.cz

Abstract. Recent interest in probabilistic logic as a formalism for machine learning has motivated the formulation of "probabilistic rule learning", where the task is to induce a set of logical rules from a probabilistic database. We start by defining rule learning within propositional, probabilistic logic. Then we show that this problem can be viewed as a regression task with integer variables. This problem is translated into Mixed Linear Programming, which is experimentally shown to provide a speedup over the current implementation. The advantage of this approach is the ability to switch between logically interpretable rule learning with binary coefficients and classical regression with rational coefficients.

Keywords: probabilistic logic, rule learning, problog, probfoil, integer linear programming

1 Introduction

For long, rule learning systems have been a popular choice in the machine learning community, mainly due to the comprehensibility of the encoded knowledge. However applications using rules (or logic-based representation in general) usually fail to capture uncertainty. The attempts to overcome these shortcomings have lead to introducing measures (e.g. *rule coverage* in ILP systems), which are defined on top of the rule representation and therefore lack the elegance of a unified framework.

Recently there has been a growing interest in merging the mathematical logic with probability theory, which is known to capture uncertainty well, but has limited capabilities in capturing structured knowledge. Most of the formalisms, which deal with probabilities are propositional. These efforts have spawned many formalisms such as *Markov logic* [4], which merges first-order logic and Markov networks, or *ProbLog* [1], which is a probabilistic extension of Prolog.

Despite these advances, the rule learning task within probabilistic logic has been overlooked for a long time. *Probabilistic Rule Learning* (PRL) [2] was first introduced using the declarative setting of ProbLog.

We consider it useful to elaborate the core concepts of PRL in simpler declarative settings before upgrading it to richer formalisms such as ProbLog. Indeed,

the risk associated with exploring PRL in expressive settings prematurely is that some necessary trade-offs in expressiveness will be ad-hoc. For instance, learning examples in PRL as defined in [2] are ground facts whereas in ILP they are usually clauses; it is not clear just whether and why this limitation is needed.

Below we propose a bottom-up approach to build the PRL framework, starting from the simplest case of learning from propositional interpretations without intensional background knowledge. In fact we do not go any further except for outlining the possible upgrading directions.

The concepts of PRL which we do maintain below are as follows

- Learning examples and background knowledge contain probabilistic assertions.
- Learned knowledge has a dual interpretation: it is a probability estimator (like e.g. a Bayesian network) but it has a form that allows it to be also interpreted as a logical rule expressing crisp relationships between the involved propositions.

2 Task Formulation

Given a set of propositions $a_1 \dots a_n$, a *probabilistic interpretation* is the assignment of probabilities to these propositions, i.e. a real vector $\mathbf{p} \in [0;1]^n$. Probabilistic interpretations will act as learning samples. A sample set is thus a matrix

$$\mathbf{S} = \begin{bmatrix} p_1^1 & p_2^1 & \dots & p_n^1 \\ p_1^2 & p_2^2 & \dots & p_n^2 \\ \dots & \dots & \dots & \dots \\ p_1^m & p_2^m & \dots & p_n^m \end{bmatrix}$$

wherein rows correspond to the m samples. The matrix is simply a real-valued attribute-value table; we use the seemingly superfluous word *interpretation* since it will enable us to maintain terminology when upgrading to the first-order case.

There is further a distinguished ('target') proposition a_0 which is assigned a probability p_0^i for each learning sample ($1 \leq i \leq m$). The goal is to learn a function $f : [0,1]^n \to [0,1]$ that estimates the probability of a_0 given a probabilistic interpretation.

3 A Regression-based Formulation

The most standard way to solve the above task is to approach it as regression. Assuming a linear form of f

$$p_0 \approx f(\mathbf{p}) = x_1 p_1 + x_2 p_2 + \dots x_n p_n = \mathbf{x}^{\mathrm{T}} \mathbf{p} \tag{1}$$

we would get[1]

$$\mathbf{x} = \arg\min_{\mathbf{x}} ||\mathbf{S}\mathbf{x} - \mathbf{p}_0|| \tag{2}$$

[1] All vectors in this text are column vectors unless transposed.

where $\mathbf{p}_0 = \left[p_0^1, p_0^2 \ldots p_0^m\right]^{\mathrm{T}}$ are the target proposition's probabilities corresponding to the samples $1 \ldots m$ and $||.||$ is some suitable norm. This problem has a simple analytical solution [3] under the L_2 norm if $\mathbf{x} \in R^n$. In general, such a solution would have no *rule* interpretation (because rules must be expressed in propositional logic) and thus would not comply with the PRL framework.

One way to accomplish the rule-interpretability of the result is to require $\mathbf{x} \in \{0,1\}^n$. In this case, Eq. 1 translates into

$$p_0 = \sum_{x_i=1} p_i \tag{3}$$

which can be interpreted as (a consequence of) the disjunctive rule

$$a_0 \leftrightarrow \bigvee_{x_i=1} a_i \tag{4}$$

on the condition that the propositions a_i are mutually exclusive. If we adopted the closed-world assumption as is done in ProbLog, we could as well write the implication sign $\leftarrow$ in the rule instead of the equivalence sign.

Analogically to the above we can arrive at a result interpretable as a *conjunctive* rule. All it takes is to substitute all probabilities with their logarithms[2]

$$\log p_0 \approx f(\mathbf{p}) = x_1 \log p_1 + x_2 \log p_2 + \ldots + x_n \log p_n \ .$$

Again, if $\mathbf{x} \in \{0,1\}^n$, we get this time

$$p_0 = \prod_{x_i=1} p_i \tag{5}$$

which can be interpreted as (a consequence of) the conjunctive rule

$$a_0 \leftrightarrow \bigwedge_{x_i=1} a_i \tag{6}$$

on the condition that the propositions a_i are mutually independent, which is arguably a more reasonable assumption than that of mutual exclusivity as made in the disjunctive case. Again, we could replace $\leftrightarrow$ with $\leftarrow$ under the CWA.

In summary, finding a single propositional disjunctive or conjunctive probabilistic rule can be approached as a linear regression problem with binary coefficients.

3.1 Regression by Integer Linear Programming

In this section we present a strategy to reduce the regression problem into the Linear Programming ($\mathcal{LP}$) formalism based on [5]. Generally, we can distinguish three forms of $\mathcal{LP}$ depending on the type of the output (seeked) variables:

[2] Zero probabilities would be replaced by a very small positive number.

a classical, which deals with rational numbers, Integer-$\mathcal{LP}$ (whose special case is a Boolean-$\mathcal{LP}$), and the mixture case (M$\mathcal{LP}$) where some of the output variables are real and others are integer. In this article we will be employing M$\mathcal{LP}$.

In general, the goal of $\mathcal{LP}$ is, given matrix $\mathbf{A}$ and vectors $\mathbf{b}$, $\mathbf{c}$, to find vector $\mathbf{y}$ that minimizes

$$\mathbf{c}^{\mathrm{T}}\mathbf{y} \tag{7}$$

subject to

$$\mathbf{A}\,\mathbf{y} \leq \mathbf{b}\ . \tag{8}$$

The approach we adopt here aims at tightening the lower and upper bounds on the difference between the actual and predicted probabilities. Formally, we require that for each example i,

$$\left| p_0^i - \left(\mathbf{p}^i\right)^{\mathrm{T}} \cdot \mathbf{x} \right| \leq \epsilon^i$$

where $\epsilon^i \in R$. This can be rewritten into

$$-\,\epsilon^i \leq \left(\left[-\left(\mathbf{p}^i\right)^{\mathrm{T}} \mid p_0^i \right] \cdot \left[\frac{\mathbf{x}}{1} \right] \right) \leq \epsilon^i \tag{9}$$

The goal of the of the M$\mathcal{LP}$ solver will be to maintain the inequalities above and at the same time minimize the bounds ϵ^i.[3]

As commented earlier, we need the vector $\mathbf{x}$ to consist of only zeros and ones so that the probability estimator can be dually interpreted as a logical rule. In the M$\mathcal{LP}$ framework, this is achieved by declaring components of $\mathbf{x}$ as integer and at the same time stipulating a further constraint

$$0 \leq x_i \leq 1 \tag{10}$$

for all components ($1 \leq i \leq n$) of $\mathbf{x}$. However these constraints are optional and omitting them leads to a standard regression under the L_1 norm. This can be exploited in estimating the amount of precision lost due to logical interpretability as shown in the experiments.

Notice that the there are two types of variables: the states of all propositional symbols $\mathbf{x}$ (which are binary) and the errors of prediction for each example ϵ (which are rational). As both types of variables change their value during induction, they both are included in the set of M$\mathcal{LP}$ vector of variables $\mathbf{y}$.

Instantiation of the M$\mathcal{LP}$ problem. We want to instantiate Eq. 8 so as to enforce the desired inequalities (Eqs. 9 and 10), and Eq. 7 to minimize the bounds ϵ^i from Eq. 9. Denoting $\boldsymbol{\epsilon} = \left[\epsilon^1, \epsilon^1, \ldots \epsilon^m\right]^{\mathrm{T}}$, the inequalities are enforced by prescribing

[3] In principle we could have all ϵ^i equal to each other but since some examples may be harder to fit than others (e.g., they may be outliers), we choose not to equalize them.

$$\mathbf{A} = \left[\begin{array}{rrr} \mathbf{S} & -\mathbf{p}_0 & -\mathbf{I}\,\epsilon \\ \hline -\mathbf{S} & \mathbf{p}_0 & -\mathbf{I}\,\epsilon \\ \hline \mathbf{I} & 0 & \mathbf{0}^{\mathrm{T}} \\ \hline -\mathbf{I} & 0 & \mathbf{0}^{\mathrm{T}} \\ \hline \mathbf{0'}^{\mathrm{T}} & 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0'}^{\mathrm{T}} & -1 & \mathbf{0}^{\mathrm{T}} \end{array}\right] \tag{11}$$

$$\mathbf{y} = \left[\begin{array}{c} \mathbf{x} \\ \hline r \\ \hline \boldsymbol{\epsilon} \end{array}\right] \quad \mathbf{b} = \left[\begin{array}{r} \mathbf{0} \\ \hline \mathbf{0} \\ \hline \mathbf{1} \\ \hline \mathbf{0} \\ \hline 1 \\ -1 \end{array}\right] \tag{12}$$

Here, $\mathbf{1}$ ($\mathbf{0}$) is the vector of n 1's (0's, respectively), $\mathbf{0'}$ is the vector of m 0's, and $\mathbf{I}$ is a $n \times n$ unit matrix (1's in the main diagonal, 0's elsewhere). Thus $\mathbf{A}$ has $n + 1 + m$ columns and $2m + 2n + 2$ rows. For ease of reference we partitioned the matrix and the vectors by vertical lines and will refer to the partitions as multirows. Plugging $\mathbf{A}$, $\mathbf{y}$ and $\mathbf{b}$ into Eq. 8 we see that the first (second) multirow of $\mathbf{A}$ corresponds to the left (right, respectively) inequality in Ineq. 9 provided that $r = 1$. This latter condition is in turn enforced by the last multirow (i.e., last two rows) of $\mathbf{A}$. Lastly, the third and forth multirow of $\mathbf{A}$ implement the inequalities in Eq. 10.

For the M$\mathcal{LP}$ solver, the last multirow of $\mathbf{y}$ will be declared as rational while the remaining components will be declared as integer.

Finally, to minimize the error bounds $\boldsymbol{\epsilon}$ through Eq. 7, we set

$$\mathbf{c} = \left[\begin{array}{c} \mathbf{0} \\ \hline 0 \\ \hline \mathbf{1'} \end{array}\right] \tag{13}$$

4 Experiments

We have performed an experimental evaluation of the proposed algorithm. As the the Mixed-LP solver, we have used the `lp_solve` implementation, which can be found at `http://lpsolve.sourceforge.net`.

Unless stated otherwise, the experiments were performed on a `windsurfing` dataset [2]. All the experiments used the default setting (unless stated otherwise):

1. The algorithm was implemented as stated in this article.
2. The ProbFOIL algorithm was adapted to solve the same problem as stated here: Instead of finding a first-order clause, the target clause was limited to a conjunction of propositional literals.

3. The `windsurfing` dataset contains 4 propositional symbols: `precipitation`, `windOk`, `sunShine` and `surfing`. The target hypothesis was generated as an implication having the `surfing` symbol in its head and the body consisting of the remaining literals.
4. The first comparison was made using the original dataset with 20 real-world measurements.
5. In order to measure performance, we have further extended the dataset artificially: Examples were generated by sampling the probabilities of the first 3 proposition symbols: `precipitation` $\sim Beta(a = 2, b = 8)$, `windOk` $\sim Beta(a = 7, b = 3)$ and `sunShine` $\sim Beta(a = 6, b = 4)$.
6. The probability of the `surfing` proposition was calculated according to the target hypothesis by following the *ProbLog* semantics. Afterwards a noise was added, which followed a uniform distribution, whose width was set to $k/2$. For $k = 1$ the noise is high and the prediction is likely to be randomous. For $k < 0.01$ the noise is irrelevant and we expect an exact inference.
7. Each generated dataset contains 20 learning examples.

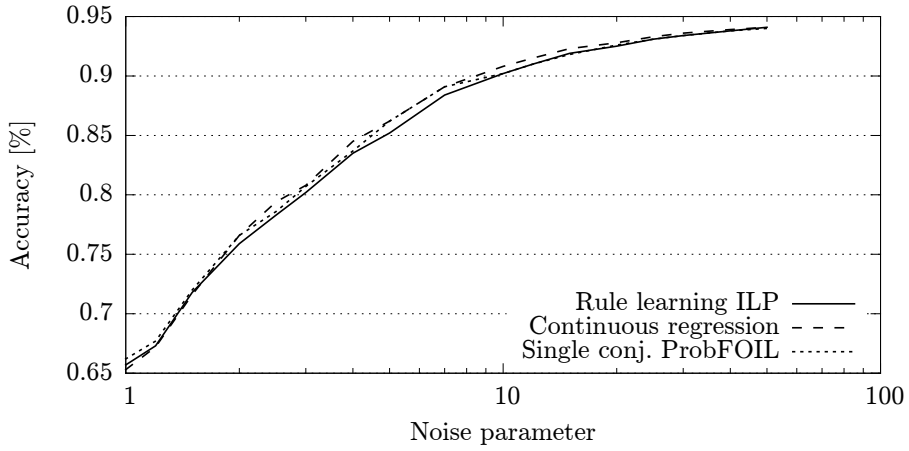

Fig. 1. Dependency between accuracy of prediction and the k parameter, which sets the amount of noise in the data. The solid line represents the proposed algorithm. The "continuous regression" is the same algorithm without the constraints on the $\mathbf{x}$ vector, which sets an upper bound on the prediction.

Using the real-world dataset, we have performed a basic test of the system. The induced rule was found in less than 1 second. This shows an advantage over the full first-order ProbFOIL implementation, which is reported to achieve a runtime around 40 s [2].

More evaluation of the time requirements was done evaluating scalability using a generated dataset with an increasing number of examples. The results

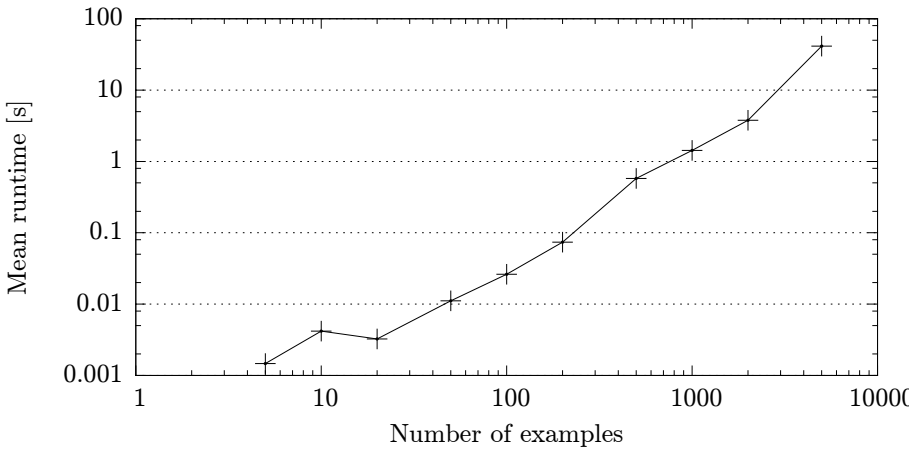

Fig. 2. Scalability of the proposed algorithm. In this case we have varied the number of examples in the dataset and measured the time for finding the target clause.

shown in Figure 2 indicate that the proposed system should be able to handle datasets of more than 10 000 examples. The roughly exponential growth of the time is an expected result; caused by the fact that every added example into the dataset adds one more variable into the $\mathcal{LP}$ and increases its dimensionality.

Figure 1 shows that the predictions by rules produced by ProbFOIL and the proposed system are nearly identical.

5 Discussion

5.1 Variations of the ILP strategy

Several potentially useful alterations of the above formulation are possible. First, by inserting $\mathbf{0}'$ instead of $\boldsymbol{\epsilon}$ into the first (second) multirow of $\mathbf{b}$, we can stipulate that the prediction error on each training sample may only be positive (negative, respectively). This would be appropriate e.g. if the $\mathcal{LP}$ procedure is used to learn a single conjunctive rule, and is iterated within a procedure for DNF learning.

Second, the last multirow of $\mathbf{c}$ can generally contain a vector of real numbers which would act as *weights* attached to individual learning samples. These could weight our confidences about the probabilities attached to the samples. A motivating scenario for doing this is as follows. A natural language processor reads the sentence: "Gene a interacts with gene b with 30% probability." Moreover since most NLP are statistics-based, it can also state that the statement was parsed correctly with 80% probability. Those two probabilities cannot be mixed. Nevertheless the second one can be considered as the weight of the sample acquired from the parsed sentence.

5.2 Regression with Regularization

As follows from the above, we need to enforce the condition $\mathbf{x} \in \{0,1\}^n$ to guarantee that probability estimators are interpretable not only as as rules about the probabilities of propositions but also about the propositions themselves. A 'soft' way to enforce the condition in the regression tasks above is the *ridge regression* strategy [3]. This amounts to extending the optimization problem in Eq. 2 with a regularization term. This is normally adopted in order to keep the norm of $\mathbf{x}$ small. Here we want to keep it close to 0 or 1. Therefore we choose the regularization term $\mathbf{x}^{\mathrm{T}}(\mathbf{1}-\mathbf{x})$ which is continuous, and is zero for vectors $\mathbf{x}$ consisting only of zeros or ones. To keep the regularizer non-negative we still need to stipulate that $\mathbf{x} \in [0;1]^n$. In summary, we arrive at the optimization problem

$$\mathbf{x} = \arg\min_{\mathbf{x}} \Big((\mathbf{S}\mathbf{x} - \mathbf{p}_0)^2 + \lambda \mathbf{x}^{\mathrm{T}}(\mathbf{1}-\mathbf{x}) \Big) \tag{14}$$

subject to

$$\mathbf{x} \in [0;1]^n$$

where $\lambda \in R$ is a regularization coefficient. We used the quadratic norm (L_2) in the first term since the regularization term also has a quadratic form.

However large λ is, we still may obtain a solution with components not exactly equal to zero or one. The 'softness' of this approach is in that such components would be simply swayed to zero or one (whichever is closer) before interpreting the rule.

Open question. Is there a standard closed-form solution and/or an effective algorithm available for this formulation of regularized regression?

Even if not, the minimization problem above can of course be approached through various versions of gradient algorithms. Note that such a gradient algorithm would be diferrent from the greedy algorithm ProbFOIL proposed in [2] since here we would work in a real vector space and indeed would use the concept of gradient.

5.3 DNF

Open question. Could we formulate finding a DNF (disjunction of conjunctions) probabilistic rule as a regression problem as above? This seems unlikely and we will probably resort to incremental greedy learning of a DNF by a adding one term (conjunction) to the DNF at a time. By adding a term to the DNF, the probability associated with the DNF will never decrease. Therefore it is reasonable to require that the approximation in Eq. 1 corresponding to each term *underestimates* the probability of the target proposition. This is easily facilitated by the integer linear programming formulation addressed below in Section 3.1.

Note that the relationship between Eq. 3 (5) on one hand and Eq. 4 (6) is not straightforward.

Consider rule $a_0 \leftrightarrow a_1$. From this, it can be soundly (deductively) inferred that also the respective probabilities are equal $p_0 = p_1$. However, inferring the former from the latter is not sound, since a different rule such as $a_0 \leftrightarrow \neg a_1$ may as well explain the equal probabilities. When inferring the declarative (logic) knowledge from the probabilistic result, we are in fact making an unsound *inductive* step. That, as usual in induction, can only be justified on the grounds of statistics, in particular by the multiplicity of learning samples where the probabilities are equal.

In summary, the PRL framework incurs two, rather than just one, inductive steps. One lies in the learning of the probability estimator and the second lies in the transition from the probabilistic to the logic interpretation of the learned estimator. Informally, by the extra inductive we are paying for the weakened requirement on the learning inputs (i.e., for assuming uncertain, rather than crisp, inputs).

Open question. The second inductive step calls for its own statistical learnability model.

6 Conclusions

We have presented a method for translating probabilistic rule learning within propositional logic into Mixed linear programming. The translation leads to a massive speed-up of the induction. A promising feature of this method is the ability to switch between logically interpretable solution and pure regression.

Due to the limitation of the current algorithm (the constructed hypothesis is a single propositional clause), we imagine the algorithm to be employed in some larger framework, in which it can serve as a solver for particular cases of larger problems. However further extensions can be investigated as proposed in the discussion.

Acknowledgment

We gratefully acknowledge the Financial support of the Czech Science Foundation (GA ČR) for the years 2010-2012 (grant no. 103/10/1875).

References

1. De Raedt, L.: Logical and Relational Learning. Springer (2008)
2. De Raedt, L., Thon, I.: Probabilistic rule learning. In: 20th Int. Conf. on Inductive Logic Programming (2010)
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2001)
4. Pedro Domingos, D.L.: Markov Logic: An Interface Layer for Artificial Intelligence. Morgan & Claypool Publishers (2009)
5. Wagner, H.M.: Linear programming techniques for regression analysis. Journal of the American Statistical Association 54(285), 206–212 (Mar 1959), http://www.jstor.org/stable/2282146