Decision-Theoretic Logic Programs

Jianzhong Chen and Stephen Muggleton

Department of Computing, Imperial College London, London SW7 2AZ, UK
{cjz, shm}@doc.ic.ac.uk

Abstract. We propose a new Probabilistic ILP (PILP) framework, Decision-theoretic Logic Programs (DTLPs), in the paper. DTLPs extend PILP models by integrating decision-making features developed in Statistical Decision Theory area. Both decision-theoretic knowledge (e.g. utilities) and probabilistic knowledge (e.g. probabilities) can be represented and dealt with in DTLPs. An implementation of DTLPs using Stochastic Logic Programs (SLPs) is introduced and a DTLP parameter learning algorithm is discussed accordingly. The representation and methods are tested by performing regression on the traditional mutagenesis dataset.

1 Introduction

There is currently considerable interest within *Inductive Logic Programming* (ILP) community in *Probabilistic Inductive Logic Programming* (PILP) [6] and closely allied areas of *Statistical Relational Learning* [8] and *Structured Machine Learning* [7]. PILP naturally extends traditional ILP [13] by introducing probabilities that can explicitly deal with uncertainty such as missing and noisy information [4]. A new framework, *Decision-Theoretic Logic Programs* (DTLPs), is proposed in this paper which extends PILP to *Statistical Decision Theory* [1, 9] area by introducing some decision-theoretic features that can explicitly represent and deal with expected utilities/rewards/losses happened in people's decision making behaviours.

Statistical decision theory is concerned with the making of decisions in the presence of statistical/probabilistic knowledge which sheds light on some of the uncertainties involved in the decision problem [1]. It simulates the process where a decision-maker chooses outcomes of an act given the background information (about the states of the world) that the decision-maker has. It is common to divide decisions into categories based on the scale of the knowledge, i.e. *decisions under certainty* if the information are deterministic, *decisions under risk* when the decision-maker has complete probabilistic knowledge and *decisions under uncertainty* when the probabilistic knowledge are partially known. The dominating approach to decision-making with probabilities is *expected utility* theory, in which utilities are the numerical values the decision-maker has set for the outcomes of an act. The basic decision-rule in the theory is as simple as: choose the outcome with the highest expected utility.

The motivation of DTLPs is shown by an example in Table 1(a), which is adapted from [17] that demonstrates making a decision of whether to bring the

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

(a)	rain	no rain
umbrella	10(dry,heavy)	10(dry,heavy)
no umbrella	0(wet,light)	15(dry,light)

```
(b) 0.4::coin(head). 0.6::coin(tail).
    sequence([]).
    sequence([H|T]):-coin(H),sequence(T).
```

Table 1. (a) Decision matrix of 'umbrella & rain' example; (b) An SLP example.

umbrella or not at a certain day based on the statistical information of whether it rains or not on the day. DTLPs are a framework that can encode such kind of decision-making problems where both utilities and probabilities are involved and handled. An implementation of DTLPs using *Stochastic Logic Programs* (SLPs) has been developed in the paper, based on which a method of learning probabilities and utilities for a given decision-theoretic logic program is introduced. The method has been tested and evaluated by revisiting the traditional mutagenesis regression problem. The experiment results show that the settings and learning method of DTLPs not only successfully integrate decision-theoretic features into PILP, but also provide a way of performing regression using PILP.

2 Decision-Theoretic Logic Programs

2.1 Background

Expected utility theory plays a central role in statistical decision theory.

Definition 1 (Expected utility theory). *Let f be an act and $x_1, x_2, \cdots, x_N$ the states of the world ($n \geq 1$) that might influence the outcome of f. Let o_i ($1 \leq i \leq N$) be an outcome f will have if x_i is the true state. Let Pr be a probability function defined on $\{x_i\}$ and U a utility function defined on $\{o_i\}$. The* expected utility *of f, relative to Pr and U, is $EU(f) = \sum_{i=1}^{N} Pr(x_i)U(o_i)$.*

Theorem 1 (Decision-rule theorem). *If a decision-maker's preferences satisfy certain qualitative conditions then there exists a Pr and a U such that, for all acts f and g, the decision-maker prefers f to g iff $EU(f) > EU(g)$.*

Expected utility theory could, more precisely, be called "probability-weighted utility theory". When applying the above theory and theorem to the example in Table 1(a), the decisions are made based on Pr(rain), the probabilities of rain. For example, if Pr(rain)=0.2 (e.g. on 1st June), then a rational person will prefer not bringing the umbrella, as EU(no_umbrella)=12 $>$ EU(umbrella)=10; if Pr(rain)=0.4 (e.g. on 1st May), then a rational person might have to bring the umbrella (although it may result in carrying a heavy bag), as EU(umbrella)=10 $>$ EU(no_umbrella)=9.

Stochastic Logic Programs (SLPs) [10] are one of the developed PILP frameworks that provide a natural way of associating probabilities with logical rules and have been applied in some real applications [3, 4]. An SLP S is a definite logic program, where each clause C is a first-order range-restricted definite clause and some of the definite clauses are labelled/parameterised with non-negative numbers, $l :: C$. In a pure normalised SLP, each choice for a clause C has a

10 :	umbrella(A):-rain(A,y).
10 :	umbrella(A):-rain(A,n).
0 :	no_umbrella(A):-rain(A,y).
15 :	no_umbrella(A):-rain(A,n).
0.4 ::	rain('01/05',y).
0.6 ::	rain('01/05',n).
0.2 ::	rain('01/06',y).
0.8 ::	rain('01/06',n). (a)

-1.91 :	mut(A):-active(A,1).
-2.32 :	mut(A):-active(A,2).
-1.99 :	mut(A):-active(A,3).
-1.32 :	mut(A):-inactive(A).
0.47::	active(A,1):-logp(A,B),gteq(B,4.18).
0.32::	active(A,2):-lumo(A,B),lteq(B,-1.937).
0.21::	active(A,3):-logp(A,B),gteq(B,2.74), ring_size_5(A,C).
	inactive(A):-not(active(A,1)), not(active(A,2)),not(active(A,3)).
(b)	logp('d63',2.79).lumo('d63',-3.768)....

Table 2. (a) a SLP-based DTLP for the 'umbrella & rain' example (Table 1(a)), where the variable A stands for a certain date; (b) a SLP-based DTLP for the mutagenesis data set learned in section 4, where A is a compound, B is a real number and C is a structure.

parameter attached and the parameters sum to one, so they can therefore be interpreted as probabilities. Normalised SLPs are defined such that each parameter l denotes the probability that C is the next clause used in a derivation given that its head C^+ has the correct predicate symbol. Table 1(b) shows an impure normalised SLP that represents a sequence of tossed coin each of which comes up either head (with probability 0.4) or tail (with probability 0.6).

Generally speaking, an SLP S has a *distribution semantics* [11] Learning SLPs has been studied in [5], which solves the parameter estimation problem by developing *failure-adjusted maximisation* (FAM) algorithm, and in [11], which presents a preliminary approach to structure learning. FAM is designed to deal with SLP parameter learning from incomplete or ambiguous data in which the atoms in the data have more than one refutation that can yield them.

2.2 DTLPs

Definition 2 (Decision-theoretic logic programs). *A DTLP D is a definite logic program that consists of three types of first-order range-restricted definite clauses $\{C\}$: a set of* deterministic clauses $\{C^+ :- C^-\}$*, where C^+ and C^- are the head and the body of C respectively; a set of* probabilistic clauses*, $\{p :: C^+ :- C^-\}$, where p stands for conditional probability $Pr(C^+|C^-)$ (based on some probability function); and a set of* decision-theoretic clauses*, $\{u : C^+ :- C^-\}$, where u is a utility value (defined by some utility function) of making a decision C^+. Accordingly, three types of predicates are defined in D:* deterministic predicates *that specify non-probabilistic facts;* decision-theoretic predicates *that represent decisions (outcomes of acts); and* probabilistic predicates *that represent the states of world affecting decisions. The subset D_q of clauses in D whose heads share the same predicate symbol q is called the definition of q. For each q, if q is a probabilistic predicate, we assume the sum of the probabilities of the clauses in D_q is normalised to 1.*

We restrict that no decision-theoretic predicates will occur in the body of probabilistic and deterministic clauses, and only probabilistic and deterministic predicates could occur in the body of decision-theoretic clauses. Definition 2 implies a hierarchical framework of building a DTLP, where decision-theoretic predicates are on top of and made up of probabilistic and deterministic predicates that are made up of other probabilistic and deterministic predicates. The setting is such that DTLPs could be built upon any PILP model that can define and manipulate probabilistic and deterministic knowledge. A DTLP built upon SLPs representing the "umbrella & rain" example is listed in Table 2(a). Therefore, the semantics of a PILP-built DTLP closely depend on the semantics of PILP model that express and interpret probabilities.

As in PILP, two types of probability semantics could be encoded in DTLPs, i.e. *possible worlds* probabilistic structure and *domain-frequency* probabilistic structure [4, 12]. With possible worlds semantics (encoded in most PILP models except SLPs), for a decision-theoretic predicate, a set of utilities are assigned on its definition, which is made up of a set of exclusive possible worlds. In this case, the expected utility of a decision should be computed by considering all the possible worlds. For example, in the DTLP shown in Table 2(a), utilities are assigned on the definitions of the two decision-theoretic predicates, umbrella(A) and no_umbrella(A), in which two possible worlds (it rains or it does not rain) are specified respectively. With domain-frequency semantics (e.g. in SLPs), for a decision-theoretic predicate, a set of utilities are assigned on its definition that is made up of a set of domain features. As in SLPs, the domain features could be partially overlapped between each other such that an object/example is ambiguous in the sense that it could have more than one yields in its proof [5]. In this case, the expected utility of a decision could be computed by considering one or more domain features. For instance, in the SLP-based DTLP example shown in Table 2(b), the mutagenic status of a compound A could be either active or inactive. If A is active, then A could be probabilistically categorised into one or more active types, e.g. active(A,1), active(A,2) and active(A,3).

In addition to the proof settings and probability computation of PILP models used to build DTLPs, the calculation of expected utilities for a decision is a distinct step in DTLPs. If possible worlds semantics are encoded, the expected utility rule defined in Definition 1 should be followed. On the other hand, if domain-frequency semantics are the case, the expected utility of a decision should be the weighted mean of utilities of clauses involved in decision-making (see in Table 3 step 4).

3 Learning DTLPs

SLPs are used as the base PILP model to build DTLPs in this paper. SLPs are also used to simulate DTLPs with possible worlds probabilistic structure as shown in the example of Table 2(a). With the help of SLP learning, we develop a DTLP parameter learning method in which both probabilities and utilities could be estimated for a given definite logic program. This implies a three-stage DTLP learning strategy: (1)learning a logic program *LP* by an ILP system (e.g. Progol [14]); (2)building an SLP *S* with *LP* and learning parameters for *S* by performing FAM algorithm [5]; (3)building a DTLP *D* on top of *S*, estimating

```
for each fold in an $n$-fold cross-validation
  1.Estimate parameters for $S$ using FAM from train set;
  2.Compute posterior probability $Pr(e|S)$ given $S$ for each example $e$ in train set;
  3.for each decision-theoretic clause $C$ in $D$
    Compute $EU(C) = \frac{\sum_i Pr(e_i|S)U(e_i)}{\sum_i Pr(e_i|S)}$, where $e_i$ is predicted by $C^-$;
  4.Compute $EU(f)$ for each example $f$ in test set using
    $EU(f) = \frac{\sum_{i=1}^M Pr(C_i)EU(C_i)}{\sum_{i=1}^M Pr(C_i)}$, where $f$ is predicted by $\{C_i\}_{i=1}^M$;
  5.Compute mean squared error (MSE) by comparing $EU(f)$ and $U(f)$ for all $f$.
```

Table 3. DTLP parameter learning algorithm

Background Knowledge	B1	B2	B3	B4
Mean Squared Error (MSE)	0.265	0.192	0.174	0.170
Standard Error (SE)	0.098	0.099	0.086	0.084
Progol Predictive Accuracy [18]	0.76	0.81	0.83	0.88
Linear Regression Predictive Accuracy [18]	0.89 ± 0.02			

Table 4. Experiment results. The standard error in the estimate is $\sqrt{(1 - MSE)MSE/N}$, where N is the number of test examples.

utilities for the decision-theoretic clauses in D from train datasets, and evaluating D by predicting expected utilities for test datasets. The last stage also suggests the possibility of performing regression which could not be done by proper ILP and PILP systems. A detailed algorithm of stage 3 is listed in Table 3.

4 Experiments

The DTLP learning algorithms developed in the paper have been tested by performing regression for the mutagenesis dataset (188 'regression friendly' compounds) and the four sets of logic theories obtained by early Prolog implementation of Progol [18], which were learned based on four incremented sets of background knowledge with different predictive accuracies. Non-structural feature LUMO (Lowest Unoccupied Molecular Orbital) is amenable as the target feature for regression. The experiment results are listed in Table 4 in terms of mean squared error and standard error of regression, which show the methods work for the regression with target feature LUMO and the power of different sets of background knowledge [18]. A learned DTLP with background knowledge base B4 is listed in Table 2(b). A predictive accuracy of 0.89 ± 0.02 has been reported using linear regression for the same data set with related attributes in [18].

5 Conclusions and Future work

A preliminary DTLP framework is developed in the paper as well as its learning scheme that has been tested by experiments. The DTLP parameter learning method also makes it possible to perform regression rather than classification using PILP models. The future work include further discussion of the semantics of DTLPs, further development of DTLP learning methods and regression using

DTLP. The work could also benefit from the related work of combining decision theory and various AI methods, such as the Independent Choice Logic (ICL) [15], work done by Scott Sanner and Craig Boutilier et al [16, 2], etc.

References

1. James O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, New York, 2 edition, 1993.
2. C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 355–362. Austin, TX, 2000.
3. J. Chen, L. Kelley, S.H. Muggleton, and M. Sternberg. Protein fold discovery using Stochastic Logic Programs. In L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton, editors, *Probabilistic Inductive Logic Programming*, Lecture Notes in Computer Science, Vol. 4911, pages 244–262. Springer-Verlag, 2007.
4. J. Chen, S.H. Muggleton, and J. Santos. Learning probabilistic logic models from probabilistic examples. *Machine Learning*, 73(1):55–85, 2008.
5. J. Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.
6. L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton. *Probabilistic Inductive Logic Programming - Theory and Applications.* Lecture Notes in Computer Science, Vol. 4911. Springer, Berlin / Heidelberg, 2008.
7. T. Dietterich, P. Domingos, L. Getoor, S.H. Muggleton, and P. Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.
8. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning.* Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Mass., 2007.
9. Patrick Maher. *Betting on theories.* Cambridge University Press, 1993.
10. S.H. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.
11. S.H. Muggleton. Learning structure and parameters of stochastic logic programs. *Electronic Transactions in Artificial Intelligence*, 6, 2002.
12. S.H. Muggleton and J. Chen. Comparison of some probabilistic logic models. In L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton, editors, *Probabilistic Inductive Logic Programming*, Lecture Notes in Computer Science, Vol. 4911, pages 305–324. Springer-Verlag, 2007.
13. S.H. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
14. Stephen Muggleton. Progol version 5.0, 2002. http://www.doc.ic.ac.uk/ shm/Software/progol5.0/.
15. D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):5–56, 1997.
16. S. Sanner and C. Boutilier. Practical solution techniques for first-order mdps. *Artificial Intelligence*, 173(5-6):748–788, 2009.
17. R. D. Shachter and M. A. Peot. Decision making using probabilistic inference methods. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 276–283. San Mateo, CA: Morgan Kaufmann, 1992.
18. A. Srinivasan, S.H. Muggleton, , and R.D. King. Comparing the use of background knowledge by inductive logic programming systems. In L. De Raedt, editor, *Proceedings of the Fifth International Inductive Logic Programming Workshop.* Katholieke Universteit Leuven, 1995.