Negation Without Negation in Probabilistic Logic Programming

David Buchman and **David Poole**
Department of Computer Science, University of British Columbia, Vancouver, BC, Canada
http://www.cs.ubc.ca/~davidbuc davidbuc@cs.ubc.ca
http://www.cs.ubc.ca/~poole poole@cs.ubc.ca

Abstract

Probabilistic logic programs without negation can have cycles (with a preference for *false*), but cannot represent all conditional distributions. Probabilistic logic programs with negation can represent arbitrary conditional probabilities, but with cycles they create logical inconsistencies. We show how allowing negative noise probabilities allows us to represent arbitrary conditional probabilities without negations. Noise probabilities for non-exclusive rules are difficult to interpret and unintuitive to manipulate; to alleviate this we define "probability-strengths" which provide an intuitive additive algebra for combining rules. For acyclic programs we prove what constraints on the strengths allow for proper distributions on the non-noise variables and allow for all non-extreme distributions to be represented. We show how arbitrary CPDs can be converted into this form in a canonical way. Furthermore, if a joint distribution can be compactly represented by a cyclic program with negations, we show how it can also be compactly represented with negative noise probabilities and no negations. This allows algorithms for exact inference that do not support negations to be applicable to probabilistic logic programs with negations.

Introduction

In recent years, there is rising interest in combining probabilistic reasoning and logic formalisms. Logic programs have the advantage of an intuitive declarative and procedural interpretation (Kowalski 2014). Probabilistic modeling, on the other hand, is a well-understood formalism for representing quantified uncertainty. Such combined formalisms include ICL (Poole 1997; 2000), which confines randomness to independent probabilistic "noise variables", Prism (Sato and Kameya 1997), ProbLog (De Raedt, Kimmig, and Toivonen 2007), and CP-logic (Vennekens, Denecker, and Bruynooghe 2009).

In this paper, we introduce negative noise probabilities as an alternative to negations, to avoid logical inconsistencies, and we show that program expressiveness is not reduced. Introducing "negative probabilities" means that probabilities are allowed to be negative *as if* that was meaningful, as long as the marginal distribution over the variables of interest (i.e., the non-noise variables) is nonnegative.

Dıez and Galán (2003) used negative probabilities to optimize computation for the noisy-or in probabilistic graphical models. Kisynski and Poole (2009) examined extending this approach to noisy-or to first order logic. Jha and Suciu (2012) examined using negative probabilities for answering queries in probabilistic databases. Van den Broeck, Meert, and Darwiche (2013) use negative probabilities to allow for skolemization in the presence of existential quantifiers, thus allowing efficient model counting.

The full version of this paper, available from the authors' web sites, contains proofs, further examples and insights.

Programs

We use capital letters for random variables, and lower-case letters for them being $true$, e.g., b means $B = true$. A **(probabilistic) rule** has the form $p : head \leftarrow body$, where p is a probability, $head$ is a positive literal, and $body$ is a conjunction of other, positive or negative, literals. When $p = 1$, it can be omitted, and the rule is called a **deterministic rule**. The probabilistic aspect is captured using a set of special **"noise" variables** $N_1, N_2, \ldots, N_{\mathcal{N}}$. Each noise variable appears exactly once as a rule head, in special probabilistic rules called **probabilistic facts** with the form $p_i : n_i$. Other probabilistic rules are actually only syntactic sugar: $p : head \leftarrow body$ is short for $p : n_i$ and $head \leftarrow n_i \wedge body$, where N_i is a new noise variable not used by any other rule.

A **program** is a **multiset of rules**. For convenience, we sometimes treat programs as sets; however, unions may produce programs with recurring rules. A program with no noise variables is a **deterministic program**. Programs can be either **cyclic** or **acyclic**.

A **model** is an assignment to all the variables, represented as a set of positive literals. We use the stable-model semantics (Gelfond and Lifschitz 1988). We take the semantics of a deterministic program to be its unique stable model. If it does not have a unique stable model, we call the program **"(logically) inconsistent"**. Inconsistency only arises in cyclic programs, when a cycle of rules contains a negation (Apt and Bezem 1991).

A **deterministic program realization (DPR)** for a program R is a deterministic program derived from R by having every probabilistic fact $p_i : n_i$ either omitted or converted to the deterministic n_i. The semantics of a probabilistic program is a distribution over its $2^{\mathcal{N}}$ DPRs, where, inde-

pendently, each $p_i : n_i$ is converted to n_i with probability p_i and omitted with probability $1 - p_i$. The unique stable-model semantics provides a semantics of a unique model for each DPR. This provides R with a semantics of a distribution over models, which represents a joint distribution of the variables. However, using negations, a cyclic program may have a positive probability of (logical) inconsistency.

Probabilistic Rule Strengths

When all rules with a common head have disjoint bodies, the noise probabilities can be interpreted as conditional probabilities. When the bodies are not disjoint, their probabilistic influences must be combined.

Example 1. *Let* $R = \{\, p_1 : a,\ p_2 : h,\ p_3 : h \leftarrow a \,\}$. *Then* $\mathrm{P}(h \mid a) = 1 - (1 - p_2)(1 - p_3)$.

Multiple occurrences of the same rule, $p_1 : r, p_2 : r, \ldots$, can be replaced with the single $\left(1 - \prod_i (1 - p_i)\right) : r$.

In both cases, combining the influences of different rules and summing up occurrences of the same rule, the math is similar, but the behavior of the noise probabilities is not very intuitive. We suggest a different representation for probabilistic values that makes the math in both cases additive:

Definition 1. *The* ***strength*** *of a probability* $-\infty < p \leq 1$ *is*

$$\boldsymbol{\sigma \stackrel{\text{def}}{=} -\ln(1-p)}, \quad -\infty < \sigma \leq \infty.$$

Therefore, a strength σ represents the probability:

$$\boldsymbol{p = 1 - e^{-\sigma}}$$

Example 2. *R from Example 1 can be represented as* $R = \{\, \sigma_1 : a,\ \sigma_2 : h,\ \sigma_3 : h \leftarrow a \,\}$. *Writing* $\sigma(h \mid a) = -\ln(1-\mathrm{P}(h \mid a))$, *we get* $\sigma(h \mid a) = -\ln(1-p_2)(1-p_3) = \sigma_2 + \sigma_3$.

The influence of rules can thus be combined with simple addition: Given that $body_1$ and $body_2$ are *true*, then $\sigma_1 : h \leftarrow body_1$ and $\sigma_2 : h \leftarrow body_2$ are equivalent to $\sigma_1 + \sigma_2 : h$. The same is true for multiple occurrences of the same rule: $\sigma_1 : r, \sigma_2 : r, \ldots$ can be replaced with $\left(\sum_i \sigma_i\right) : r$.

Unlike noise probabilities, strengths are interpretable: the strength of a rule represents the amount (or "weight") of information it provides, which is to be added to the information provided by other rules. Strengths make our main results, and especially their proofs (in the full paper), dramatically simpler, and also make negative probabilities intuitive. $\sigma(p)$ is monotonically increasing with p: $\sigma < 0$ corresponds to $p < 0$ (more on that later), $\sigma = 0$ corresponds to $p = 0$, $0 < \sigma < \infty$ corresponds to $0 < p < 1$, and $\sigma = \infty$ corresponds to $p = 1$ (deterministic rules.) The latter is intuitive, as a deterministic rule is equivalent to an infinite number of occurrences of a probabilistic rule. $p > 1$ cannot be represented using a strength σ.

Acyclic Probabilistic Logic Programs

The set of all rules with a variable H as their head, can be seen as a specification of the conditional probability distribution (CPD) of H given (a subset of) the variables that precede it in the ordering (its "parents.")

We represent a joint assignment to $(A_1, \ldots, A_n)$ using the set $s \stackrel{\text{def}}{=} \{a_i : A_i = \mathit{true}\}$. Given a set of positive literals L, we use $\bigwedge L$ for $\bigwedge_{l \in L} l$, and $\bigwedge \neg L$ for $\bigwedge_{l \in L} \neg l$.

Negations Needed for Expressiveness

A CPD $\mathrm{P}(h \mid A_1, \ldots, A_n)$ is **monotonic** if changing some A_i from *false* to *true* can only increase $\mathrm{P}(h)$. Using negation, it is easy to represent any CPD, e.g., by creating a rule for every possible assignment s. Consider a positive-probability rule without negations $b \leftarrow a \wedge \ldots$. The rule increases $\mathrm{P}(b \mid a)$ without changing $\mathrm{P}(b \mid \neg a)$. A similar rule in which a does not appear would increase both $\mathrm{P}(b \mid a)$ and $\mathrm{P}(b \mid \neg a)$. Negation is therefore needed for representing some CPDs, e.g., non-monotonic CPDs, where $\mathrm{P}(b \mid a) < \mathrm{P}(b \mid \neg a)$.

Negative Noise Probabilities

Distributions are non-negative functions that sum to 1. The semantics of a probabilistic program R was defined as a distribution over its $2^{\mathcal{N}}$ DPRs, leading to a joint distribution $\mathrm{P}(\mathbf{V})$ over the variables. When some noise probability is negative, $\mathrm{P}(\mathbf{V})$ still sums to 1, but is not necessarily non-negative. However, the noise variables may be treated as auxiliary variables and be marginalized out to give the joint distribution of interest, $\mathrm{P}(\mathbf{V} \backslash \mathbf{N})$, and the marginalization may make $\mathrm{P}(\mathbf{V} \backslash \mathbf{N})$ nonnegative. When this happens, we can ignore the meaninglessness of the intermediate negative probabilistic values, since R has a proper joint distribution semantics $\mathrm{P}(\mathbf{V} \backslash \mathbf{N})$. If this does not happen, we say the distribution and R are **improper**.

Example 3. *The following acyclic program defines an improper* $\mathrm{P}(N_1, N_2, N_3, A, B)$, *but a proper* $\mathrm{P}(A, B)$*:*

$$R = \{\ \begin{array}{lll} p_1 : n_1, & a \leftarrow n_1, & p_1 = 0.5 \\ p_2 : n_2, & b \leftarrow n_2, & p_2 = 0.7 \\ p_3 : n_3, & b \leftarrow n_3 \wedge a\ \} & p_3 = -\frac{4}{3} \end{array}$$

Negative Strengths A probability $p < 0$ corresponds to a strength $\sigma < 0$. Using σ-notation, negative probabilities become intuitive: They allow to subtract the weights of evidence, or to subtract rule weights.

Example 4. *Let R contain a rule $\sigma : r$. Adding the new rule $-\sigma : r$ to R is equivalent to removing $\sigma : r$ from R.*

$\sigma = -\infty$, which corresponds to $\mathrm{P} = -\infty$, is not allowed, because it makes the semantics ill defined. Therefore, adding a new rule cannot cancel out a deterministic rule.

Representing CPDs Without Negations

Negative noise probabilities in acyclic programs allow us to express CPDs that cannot otherwise be expressed without negations. For example, the last two lines in Example 3 represent a non-monotonic CPD $\mathrm{P}(B \mid A)$.

Theorem 1. *Consider a set R of negation-free probabilistic rules (possibly with negative probabilities), in which the head is h and the bodies are conjunctions of subsets $s \subseteq \{a_1, \ldots, a_n\}$. Let σ_s be the probabilistic strength of the rule $h \leftarrow \bigwedge s$. Then R represents a "proper" CPD (i.e., $\forall s, \mathrm{P}(h \mid s) \in [0, 1]$) if and only if:*

$$\forall s \subseteq \{a_1, \ldots, a_n\},\ \textstyle\sum_{s' \subseteq s} \sigma_{s'} \geq 0.$$

Theorem 2. *Using negative noise probabilities, any CPD $\mathrm{P}(h \mid A_1, \ldots, A_n) < 1$ can be expressed without negations.*

The theorem follows from the correctness of Algorithm 1.

Algorithm 1.

Input: CPD $\mathrm{P}(h \mid A_1, \ldots, A_n) < 1$,
represented as $\sigma(h \mid s) < \infty$
Output: set of rules R *representing the CPD*

$S \leftarrow$ *set of all subsets of* $\{a_1, \ldots, a_n\}$
$R \leftarrow \emptyset$
while $S \neq \emptyset$ ***do***
  $s \leftarrow$ *some minimal subset in* S
  $\sigma_s \leftarrow \sigma(h \mid s) - \sum_{s' \subset s} \sigma_{s'}$
  $rule_s \leftarrow (\sigma_s : h \leftarrow \bigwedge s)$
  $R \leftarrow R \cup \{rule_s\}$
  $S \leftarrow S \setminus s$
return R

Sparsity Rules with $\sigma_s = 0$ have no effect and can be pruned; e.g., for noisy-or, only $n \ll 2^n$ rules remain.

Canonical Form

Corollary 1. *Algorithm 1 provides a* canonical form *for representing CPDs without negations. When identical rules are summed up, this representation is unique.*

Note the clear mathematical similarity of this canonical form to the "canonical parametrization" for undirected probabilistic graphical models (Koller and Friedman 2009; Lauritzen 1996; Buchman et al. 2012).

Using negations, however, representation is not unique.

Cyclic Probabilistic Logic Programs

Motivating Negations in Cyclic Programs

Consider the following first-order probabilistic cyclic logic program, loosely based on Richardson and Domingos (2006), and also based on the ProbLog tutorial[1]:

(1) $0.3:$ $smokes(X)$
(2) $0.1:$ $friends(X, Y)$
(3) $0.9:$ $friends(X, Y) \leftarrow friends(Y, X)$
(4) $0.6:$ $susceptible(X)$
(5) $0.2:$ $smokes(X) \leftarrow susceptible(X) \wedge friends(X, Y) \wedge smokes(Y)$
(6) $friends(chris, sam)$

There is a 30% smoking baseline, and, if X is susceptible, then every smoking friend has a 20% chance of also causing X to smoke. Consider now modeling nonconformity instead of susceptibility. The probability a "nonconformist" person smokes increases with every *non-smoking* friend they have. The straight-forward approach is to make the changes below. Unfortunately, the negation added appears inside a cycle, thus the program is not logically consistent.

(4) $0.6:$ $\boldsymbol{nonconformist(X)}$
(5) $0.2:$ $smokes(X) \leftarrow \boldsymbol{nonconformist(X)} \wedge friends(X, Y) \wedge \neg\, smokes(Y)$

The rest of the paper does not deal with first-order programs.

Negations Needed for Expressiveness

Negations are best avoided, because they may create logical inconsistencies with cyclic rules. However, not all joint distributions can be expressed in cyclic programs without negation (and without negative probabilities). Characterizing what can be represented is complex.

Proposition 1. *Not all distributions can be represented using two-variable cyclic probabilistic negation-free programs, if noise probabilities are limited to* $[0, 1]$.

The proof shows the distribution a general-form program represents must satisfy: $\mathrm{P}(a \wedge b) \geq \mathrm{P}(a \mid \neg b)\,\mathrm{P}(b \mid \neg a)$.

Unintuitive Properties of Cyclic Programs

Given a program R, we use R_h for the set of all rules whose head is h. R_h can be seen as defining a CPD $\mathrm{P}_{R_h}(h \mid \text{neighbors})$. For an acyclic R, its joint distribution $\mathrm{P}_R(\cdot)$ reflects this CPD, i.e., $\mathrm{P}_R(h \mid \text{neighbors}) = \mathrm{P}_{R_h}(h \mid \text{neighbors})$. For cyclic programs, however, this is not the case. Furthermore, with negative noise probabilities, $\mathrm{P}_R(\mathbf{V})$ may even be improper, even when all CPDs $\mathrm{P}_{R_h}(h \mid \text{neighbors})$ are proper and mutually-consistent.

Compact Expressiveness with Negative Noise

With negations, cyclic programs may become logically inconsistent, when there is a negation in a rule cycle. However, this still leaves a wide array of possible logically consistent programs that, are expressive enough to be useful. An interesting example are acyclic structures containing negations that connect negation-free cyclic components.

Proposition 2. *Using negative noise probabilities, any positive non-relational joint distribution can be represented without negations.*

The representation guaranteed by Proposition 2 might be exponential in size. To better motivate using negative probabilities instead of negations, we show that joint distributions that can be compactly represented by cyclic programs with negations can also be *compactly* represented using negative probabilities and no negations (Theorem 6, Corollary 2.)

Definition 2. *A program* R *is* ***strongly consistent*** *if any logically inconsistent DPR of* R *has probability* 0.

We use $\mathrm{sum}(R)$ for the program R after identical rules are summed up. We mark $R_1 \equiv R_2$ if programs R_1 and R_2 are both strongly consistent and represent the same (proper or improper) distribution over models.

Definition 3. *A* ***consistency-maintaining ordering (CMO)*** *for a program* R *is an ordering* $(r_1, r_2, \ldots)$ *of* R*'s rules, such that for all* i, $\{r_1, r_2, \ldots, r_i\}$ *is strongly consistent.*

Theorem 3. *Every logically consistent deterministic program has a CMO.*

Theorem 4. *Every strongly consistent program has a CMO.*

Definition 4. *An arbitrary rule* $r = \sigma : h \leftarrow \bigwedge r^+ \wedge \bigwedge \neg r^-$ *is called* ***translatable*** *if* $\sigma < \infty$ *or* $r^- = \emptyset$*, in which case the* ***translated rule*** r^T *is a set of* $2^{|r^-|}$ *negation-free rules:*

$$r^T \stackrel{\text{def}}{=} \{\, (-1)^{|L|}\sigma : \; h \leftarrow \bigwedge r^+ \wedge \bigwedge L \quad : \quad L \subseteq r^- \,\}$$

Definition 5. *If all rules in a program* R *are translatable, then* R *is* ***translatable****, and the* ***translated program*** R^T *is:*

$$R^T \stackrel{\text{def}}{=} \mathrm{sum}\left(\bigcup_{r \in R} r^T\right)$$

[1] https://dtai.cs.kuleuven.be/problog/tutorial.html#tut_part1_smokers

R is translatable, if deterministic rules have no negations.

Theorem 5. *Let R be a set of negation-free rules, and r be a translatable rule $\sigma : \ h \leftarrow \bigwedge r^+ \wedge \bigwedge \neg r^-$. If $R \cup \{r\}$ is strongly consistent, then $R \cup \{r\} \equiv R \cup r^T$.*

The proof is complex, because R is cyclic, so adding rules to R may create complex "interactions" with rules in R.

Theorem 6. *If R is a strongly consistent translatable program, then R^T is a strongly consistent negation-free program and $R^T \equiv R$.*

Furthermore, $|R^T| \leq \sum_i 2^{k_i} \leq 2^k |R|$, where k_i is the number of negations in rule r_i, and $k = \max_i k_i$.

Proof. R is strongly consistent, so by Theorem 4 it has a CMO. R can be thus formed by iteratively adding rules, while maintaining strong consistency throughout. For every rule r_i being added, since r_i is translatable, Theorem 5 gives a set of 2^{k_i} negation-free rules, such that adding them instead of r_i gives a strongly consistent program with the same semantics. Repeating this replacement for all rules, we get a negation-free program with the same semantics, with $|\mathrm{sum}(\bigcup_{r \in R} r^T)| \leq \sum_i 2^{k_i} \leq 2^k |R|$ rules. □

Corollary 2. *A given nonnegative-noise-probabilities program R that is consistent with probability 1 and only has negations in non-deterministic rules, can be converted to an equivalent negation-free program (with negative probabilities) R^T with size $|R^T| \leq 2^k |R|$.*

Conclusions

We began by suggesting "probabilistic strengths", a new representation for probabilities in logic programs which is parallel to using log probabilities in graphical models. Strengths make rule algebra additive, and provide deeper insights into probabilistic programs. Strengths also make the concept of negative probabilities simpler and more intuitive.

We showed that without negative probabilities, negations are needed both in acyclic and cyclic programs, to increase expressiveness. Negations may, however, cause difficult-to-avoid logical inconsistencies in cyclic programs.

We suggest using negative noise probabilities in lieu of negations. For acyclic programs, we completely characterized the conditions for avoiding probabilistic improperness, and showed how any positive CPD can be expressed.

For cyclic programs, we first showed the need for negations arises very naturally in real-world applications. We then defined and proved the existence of consistency-maintaining orderings. We then showed how, when a distribution can be compactly expressed using a program with negations, the program can easily be "translated" to an equivalent negation-free program with negative probabilities that is at most 2^k times larger, where k is frequently small.

This translation allows exact inference algorithms that do not support negations to be applicable to probabilistic cyclic programs with negations.

A main open problem is how *approximate inference* can be efficiently carried out in cyclic probabilistic programs with negative probabilities. There is no guarantee that an approximate computation will remain a good approximation when some probabilities are negative.

Extensions to first-order logic are left for future work.

Acknowledgments

This work was supported by an NSERC operating grant to David Poole.

References

Apt, K. R., and Bezem, M. 1991. Acyclic programs. *New Generation Computing* 9(3-4):335–363.

Buchman, D.; Schmidt, M. W.; Mohamed, S.; Poole, D.; and de Freitas, N. 2012. On sparse, spectral and other parameterizations of binary probabilistic models. *Journal of Machine Learning Research - Proceedings Track* 22:173–181.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, 2462–2467.

Dıez, F. J., and Galán, S. F. 2003. An efficient factorization for the noisy max. *International Journal of Intelligent Systems* 18(2):165–177.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, 1070–1080.

Jha, A., and Suciu, D. 2012. Probabilistic databases with Markoviews. *Proceedings of the VLDB Endowment* 5(11):1160–1171.

Kisynski, J., and Poole, D. 2009. Lifted aggregation in directed first-order probabilistic models. In *Proc. Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, 1922–1929.

Koller, D., and Friedman, N. 2009. *Probabilistic graphical models: Principles and techniques*. MIT Press.

Kowalski, R. 2014. *Logic for Problem Solving, Revisited*. BoD–Books on Demand.

Lauritzen, S. L. 1996. *Graphical models*. Oxford University Press, USA.

Poole, D. 1997. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence* 94:7–56. special issue on economic principles of multi-agent systems.

Poole, D. 2000. Abducing through negation as failure: stable models in the Independent Choice Logic. *Journal of Logic Programming* 44(1–3):5–35.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Sato, T., and Kameya, Y. 1997. PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1330–1335.

Van den Broeck, G.; Meert, W.; and Darwiche, A. 2013. Skolemization for weighted first-order model counting. *arXiv preprint arXiv:1312.5378*.

Vennekens, J.; Denecker, M.; and Bruynooghe, M. 2009. CP-logic: A language of causal probabilistic events and its relation to logic programming. *TPLP* 9(3):245–308.