Probabilistic Explanation Based Learning

Angelika Kimmig[1], Luc De Raedt[1], and Hannu Toivonen[2]

[1] Department of Computer Science, Katholieke Universiteit Leuven
[2] Department of Computer Science, University of Helsinki

Abstract. Explanation based learning produces generalized explanations from examples. These explanations are typically built in a deductive manner and they aim to capture the essential characteristics of the examples.
Probabilistic explanation based learning extends this idea to probabilistic logic representations, which have recently become popular within the field of statistical relational learning. The task is now to find the most likely explanation why one (or more) example(s) satisfy a given concept. These probabilistic and generalized explanations can then be used to discover *similar* examples and to reason by *analogy*. So, whereas traditional explanation based learning is typically used for speed-up learning, probabilistic explanation based learning is used for discovering new knowledge. Probabilistic explanation based learning has been implemented in a recently proposed probabilistic logic called ProbLog, and it has been applied to a challenging application in discovering relationships of interest in large biological networks.

1 Introduction

During the 80s, explanation based learning (EBL) was a popular research theme within the field of machine learning. It aimed at finding plausible and generalized explanations for particular examples using a logical domain theory, cf. [2] for an overview and introduction. These generalized explanations were then typically turned into rules that would be added to the theory, often with the aim of speeding up further inference or extending an imperfect domain theory. Traditional explanation based learning was largely studied within first order logic representations and explanations were built using deduction [8, 16], though there was sometimes also an abductive component [1]. In the past few years, the machine learning and artificial intelligence communities have become interested in statistical relational learning [7] and probabilistic logic learning [4]; these are techniques that lie at the intersection of machine learning, logical representations and probabilistic modelling. These fields have contributed many different probabilistic logics, and have used them for various applications.

A natural question that arises in this context is whether explanation based learning can be applied to these probabilistic logics. A first step in this direction is done in [3], where logical and empirical evidence are combined in explanation based learning to get explanations with a certain confidence. The question

is investigated and positively answered within this paper. More specifically, we introduce probabilistic explanation based learning within a recently introduced simple extension of Prolog, called ProbLog [5], and demonstrate its use on a challenging application within biological link mining. Probabilistic explanation based learning computes the most likely generalized explanation from one or more positive examples and then uses these generalized explanations to identify other examples that possess this explanation (with high probability). In this way, probabilistic explanation based learning realizes a kind of probabilistic similarity or analogical reasoning. This type of reasoning is required within scientific link mining tasks in large biological networks. These are networks that consist of a large set of entities (such as proteins, tissues, diseases, genes, ...) as well as the relationships that hold amongst them. The task faced by the biologist is to investigate these networks in order to understand particular phenomenae and to discover new knowledge and relationships, cf. [15, 11]. Using probabilistic explanation based learning with ProbLog allows the life scientist, for instance, to discover probable explanations for specific phenomenae (such as a gene being related to a particular disease – say Alzheimer disease), and then to apply the discovered generalized explanation to identify other genes that are related to this disease with a similar explanation. Furthermore, it allows to rank these genes according to the likelihood of the explanation.

Probabilistic explanation based learning as introduced here is also related to probabilistic abduction, as studied by Poole [12], and to abductive explanation based learning. The difference with Poole's work however is that we compute *generalized* explanations and also apply them for analogical reasoning. In contrast to abductive explanation based learning, probabilistic reasoning is employed here.

This paper is organized as follows: we briefly review ProbLog in Section 2 and explanation based learning in Section 3. Section 4 introduces probabilistic EBL, which is evaluated using experiments in biological link mining in Section 5. Finally, Section 6 concludes and touches upon related work.

2 ProbLog: Probabilistic Prolog

ProbLog is a simple probabilistic extension of Prolog introduced in [5]. A ProbLog program consists – as Prolog – of a set of definite clauses. However, in ProbLog every clause c_i is labeled with the probability p_i that it is true, and those probabilities are assumed to be mutually independent.

Example 1. Within bibliographic data analysis, the similarity structure among items can improve information retrieval results. Consider a collection of papers $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$ and some pairwise similarities $\texttt{similar}(\texttt{a}, \texttt{c})$, e.g., based on key word analysis. Two items `X` and `Y` are $\texttt{related}(\texttt{X}, \texttt{Y})$ if they are similar (such as `a` and `c`) or if `X` is similar to some item `Z` which is related to `Y`. Uncertainty can

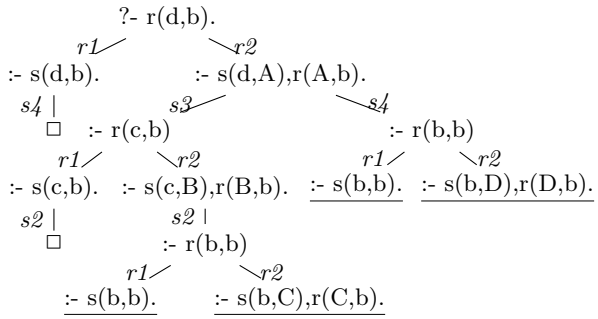

Fig. 1. SLD tree for `related(d,b)`

elegantly be represented by the attached probabilities:

$$
\begin{array}{ll}
1.0 : \texttt{related}(\texttt{X},\texttt{Y}) :- \texttt{similar}(\texttt{X},\texttt{Y}). & \\
0.8 : \texttt{related}(\texttt{X},\texttt{Y}) :- \texttt{similar}(\texttt{X},\texttt{Z}), \texttt{related}(\texttt{Z},\texttt{Y}). & \\
0.8 : \texttt{similar}(\texttt{a},\texttt{c}). & 0.7 : \texttt{similar}(\texttt{c},\texttt{b}). \\
0.6 : \texttt{similar}(\texttt{d},\texttt{c}). & 0.9 : \texttt{similar}(\texttt{d},\texttt{b}). \\
0.7 : \texttt{similar}(\texttt{e},\texttt{c}). & 0.5 : \texttt{similar}(\texttt{f},\texttt{a}).
\end{array}
$$

A ProbLog program $T = \{p_1 : c_1, \cdots, p_n : c_n\}$ defines a probability distribution over logic programs $L \subseteq L_T = \{c_1, \cdots, c_n\}$ in the following way:

$$P(L|T) = \prod_{c_i \in L} p_i \prod_{c_i \in L_T \setminus L} (1 - p_i). \tag{1}$$

Unlike in Prolog, where one is typically interested in determining whether a query succeeds or fails, ProbLog specifies the probability that a query succeeds. The *success probability* $P(q|T)$ of a query q in a ProbLog program T is defined by

$$P(q|T) = \sum_{L \subseteq L_T} P(q, L|T) = \sum_{L \subseteq L_T} P(q|L) \cdot P(L|T), \tag{2}$$

where $P(q|L) = 1$ if there exists a θ such that $L \models q\theta$, and $P(q|L) = 0$ otherwise. In other words, the success probability of query q corresponds to the probability that the query q has *a* proof in a logic program randomly sampled from T.

The evaluation of the success probability of ProbLog queries is computationally hard. In [5], this problem is tackled by employing a reduction to the computation of the probability of a monotone DNF formula, an NP-complete problem.

Example 2. Figure 1 shows the SLD-tree for proving `related(d,b)` in our example program (see Section 3 for details). This tree contains two successful proofs, and therefore the success probability of `related(d,b)` corresponds to $P((r_1 \wedge s_4) \vee (r_2 \wedge s_3 \wedge r_1 \wedge s_2))$. This probability cannot be computed as $P((r_1 \wedge s_4)) + P((r_2 \wedge s_3 \wedge r_1 \wedge s_2))$ because the two expressions are not mutually disjoint.

The key contribution of our previous work on ProbLog was the implementation of an efficient approximative inference procedure for computing the success probabilities in large ProbLog programs for the biological network mining domain (cf. also Section 5). The inference procedure employs Binary Decision Diagrams in combination with an approximation algorithm based on iterative deepening, cf. [5] for more details.

However, in probabilistic explanation based learning as introduced here, cf. Section 4, the probability of interest will not be the total probability of a query but rather the probability of a single derivation d for a given example. This probability corresponds to

$$P(d|T) = \prod_{c_i \in d} p_i \tag{3}$$

and can thus be computed exactly in an efficient way. Intuitively, it corresponds to the probability that a randomly sampled subprogram of T contains all clauses employed in the derivation d.

Example 3. There are two proofs for `related(d,b)`. The first one uses the base case of `related/2` and the fact `similar(d,b)` and thus has a probability of $1.0 \cdot 0.9 = 0.9$, the second one uses the recursive case and two facts, thus getting a probability of $0.8 \cdot 0.6 \cdot 1.0 \cdot 0.7 = 0.336$.

3 Explanation Based Learning

The central idea of explanation-based learning (EBL) as conveniently formalized for Prolog [8, 16] is to compute a generalized explanation from a concrete proof of an example. Explanations use only so-called *operational* predicates, i.e. predicates that capture essential characteristics of the domain of interest and should be easy to prove. Operational predicates are to be declared by the user as such.

Following the work by [8, 16], explanation based learning starts from a definite clause theory T, that is a pure Prolog program, and an example in the form of a ground atom $p(t_1, ..., t_n)$. It then constructs a refutation proof of the example using SLD-resolution. SLD-resolution takes a goal of the form $?- g, g_1, ..., g_n$, a clause $h \leftarrow b_1, ..., b_m$ such that g and h unify with most general unifier θ, and then produces the resolvent $?- b_1\theta, ..., b_m\theta, g_1\theta, ..., g_n\theta$. This process then continues until the empty goal is reached. SLD-resolution is illustrated in Figure 1, where each path from the root of the tree to the empty clause $\Box$ corresponds to a refutation proof of `related(d,b)`. Given the resolution proof for the example $p(t_1, ..., t_n)$, explanation based learning will generalize the proof and produce a generalized explanation. To realize this, it starts from the variabelized goal, i.e. $p(X_1, ..., X_n)$ where the X_i are different variables, and then performs the same SLD-resolution steps as in the proof for the example. The only difference is that in the general proof constructed in explanation based learning atoms $q(s_1, ..., s_r)$ for operational predicates q in a goal $?- g_1, ..., g_i, q(s_1, ..., s_r), g_{i+1}, ..., g_n$ are not resolved away. Also, the proof procedure stops when the goal contains only atoms for operational predicates. The resulting goal provides a *generalized explanation*

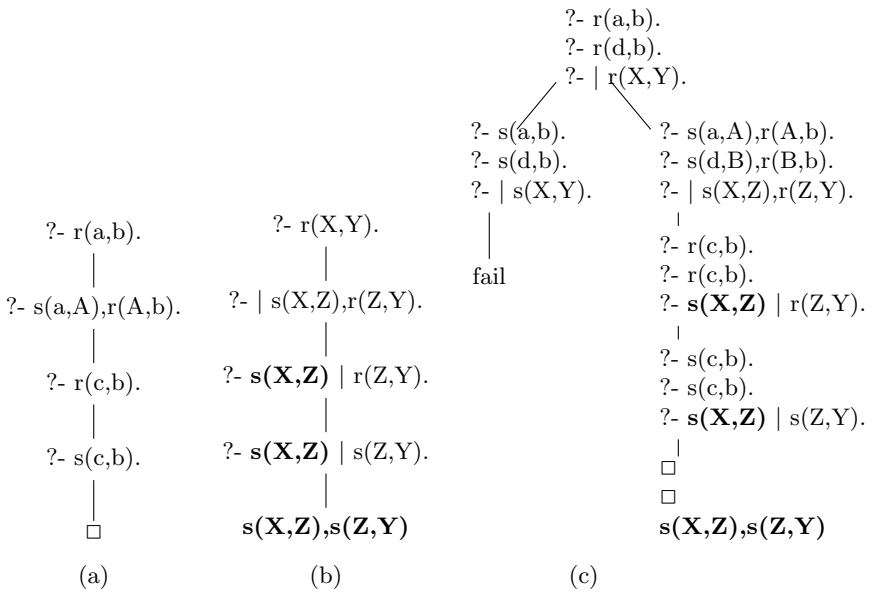

Fig. 2. (a) The successful branch of the SLD tree for `related(a,b)`. (b) The corresponding branch for general goal `related(X,Y)`, where bold atoms are part of the explanation and the bar marks the position to continue the proof. (c) A partial SLD tree for Example 6, where each node contains the current status for the two training examples as well as the general version.

for the example. In terms of the SLD-resolution proof tree, explanation based learning cuts off branches below operational predicates. It is easy to implement the explanation based proof procedure as a meta-interpreter for Prolog [16, 8].

Example 4. Reconsider the logic program of Example 1, ignoring the probability labels for now. We define `similar/2` to be the only operational predicate, and use `related(a,b)` as training example. EBL proves this goal using two instances of the operational predicate, namely `similar(a,c)` and `similar(c,b)`, and then produces the explanation `similar(X,Z), similar(Z,Y)` for the generalized example `related(X,Y)`. The result can be represented as the clause `exp_related(X,Y)`← `similar(X,Z), similar(Z,Y)`. We will call such clauses *explanation clauses*. To be able to identify the examples covered by this clause, we rename the predicate in the head of explanation clauses. The successful branches of the SLD trees for `related(a,b)` and the generalized example `related(X,Y)` are depicted in Figure 2.

4 Probabilistic Explanation Based Learning

Probabilistic explanation based learning (PEBL) extends EBL to probabilistic logic representations. In this paper, we use ProbLog as the underlying language

for PEBL, but the general idea can easily be transferred to other probabilistic logics, such as Poole's ICL [12] or Sato's PRISM [14].

Within ProbLog, as already discussed in Section 2, a probability is associated to each proof of a query. Therefore, the adaptation of explanation based learning to ProbLog is direct. The key difference now is that for each example, we compute the most likely proof and then compute the generalized explanation as sketched in the previous section.

The probability of a given single proof is calculated simply as the product $\prod_i p_i$ of probability labels of all clauses c_i used (at least once) in that proof, as defined in Equation 3 and illustrated in Example 3. This corresponds to the probability of randomly sampling a subprogram of the ProbLog program that contains all clauses used in the proof and thus admits the proof. The set of clauses C used in the proof of the example that is to be generalized can be partitioned into two sets. Indeed, define $G = \{c|c \in C$ such that c is used in the generalized proof$\}$, and $E = C - G$, i.e. E contains the example-specific clauses used to prove operational predicates. We then have that

$$\prod_{c_i \in C} p_i = \prod_{c_j \in G} p_j \prod_{c_k \in E} p_k.$$

Thus, the probability of the orginal proof is equal to the product of the probability of the generalized proof and the probability of the operational predicates for the example.

Example 5. In Example 4, $C = \{r_2, s_1, r_1, s_2\}$, $G = \{r_1, r_2\}$ and $E = \{s_1, s_2\}$. The probability $\prod_{c_j \in G} p_j = 0.8 \cdot 1.0$ also denotes the probability that the explanation clause `related(X,Y)`← `similar(X,Z), similar(Z,Y)` is logically entailed by the original ProbLog program.

Computing the most likely proof for a given goal in ProbLog is straightforward: instead of traversing the SLD-tree in a left-to-right depth-first manner as in Prolog, nodes are expanded in order of the probability of the derivation leading to that node. This realizes a best-first search with the probability of the current proof as an evaluation function. In the application sketched in Section 5, we need to deal however with goals having very many candidate proofs (each corresponding to a particular path in a large biological network). Implementing best-first search in a naive manner rapidly results in memory problems. We therefore employ the traditional solution of iterative deepening [13] to avoid these problems in our implementation. Using iterative deepening depth first search, we cut off the search at proofs with probability below a threshold. We start with a minimum probability of 0.5. If at least one explanation was found in the last completed iteration, the algorithm outputs the most probable one and stops, otherwise the next iteration uses half the probability threshold of the last one. The algorithm can also be used to return the k most probable structurally distinct explanations.

Probabilistic explanation based learning as incorporated in ProbLog offers natural solutions to two issues traditionally discussed in the context of explanation based learning [10, 9]. The first one is the *multiple explanation* problem,

which is concerned with choosing the explanation to be generalized for examples having multiple proofs. This problem arises in many applications such as the one sketched in Section 5 on mining biological networks, where there are various possible explanations as to why a particular query succeeds, for instance, a gene being linked to a particular disease. The use of a sound probabilistic framework naturally deals with this issue by selecting the *most likely* proof. The second problem is that of *generalizing from multiple examples*, another issue that received quite some attention in traditional explanation based learning. To realize this in our setting, we modify the best-first search algorithm so that it searches for the most likely generalized explanation shared by the n examples $e_1, ..., e_n$. Starting from the variabelized atom e, we compute $n + 1$ SLD-resolution derivations in parallel. A resolution step resolving an atom for a non-operational predicate in the generalized proof for e is allowed only when the same resolution step can also be applied to each of the n parallel derivations. Atoms corresponding to operational predicates are – as sketched above – not resolved in the generalized proof, but it is nevertheless required that for each occurrence of these atoms in the n parallel derivations, there exists a resolution derivation.

Example 6. Consider again our running example, and assume that we now want to construct a common explanation for `related(a,b)` and `related(d,b)`. We thus have to simultaneously prove both examples and the variabelized goal `related(X,Y)`. This is illustrated in Figure 2(c). After resolving all three goals with the first clause for `related/2`, we reach the first instance of the operational predicate `similar/2` and thus have to prove both `similar(a,b)` and `similar(d,b)`. As proving `similar(a,b)` fails, the last resolution step is rejected and the second clause for `related/2` used instead. As both `similar(a,A)` and `similar(d,B)` can be proven, `similar(X,Z)` is added to the explanation, and the procedure continues with the goals `related(c,b)`, `related(c,b)` and `related(Z,Y)`. This succeeds using the base case and adds `similar(Z,Y)` to the explanation, which thus is `similar(X,Z),similar(Z,Y)`.

Because PEBL generates a generalized explanation, which can be turned into an explanation clause, the technique can be employed to identify similar instances and to reason by analogy. Indeed, by asserting such an explanation clause and posing queries to the resulting predicate one obtains similar examples. Furthermore, the success probabilities of the examples can be used to rank them according to likelihood, and hence, similarity.

Example 7. Using the explanation clause `exp_related(X,Y)`← `similar(X,Z), similar(Z,Y)` to query for covered instances would return the following answer: `exp_related(a,b)` $(0.8 \cdot 0.7 = 0.56)$, `exp_related(e,b)` $(0.7 \cdot 0.7 = 0.49)$, `exp_related(d,b)` $(0.6 \cdot 0.7 = 0.42)$, `exp_related(f,c)` $(0.5 \cdot 0.8 = 0.40)$.

5 Experiments

Research on ProbLog and PEBL is meant to support the life scientist analysing large biological networks that can be automatically constructed from the enormous amounts of molecular biological data that are available from public sources,

```
e_path(A,B) ← node(A,gene), edge(A,C,belongs_to), node(C,homologgroup),
      edge(B,C,refers_to), node(B,phenotype), nodes_distinct([B,C,A]).
e_path(A,B) ← node(A,gene), edge(A,C,codes_for), node(C,protein),
      edge(D,C,subsumes), node(D,protein), edge(D,E,interacts_with),
      node(E,protein), edge(B,E,refers_to), node(B,phenotype),
      nodes_distinct([B,E,D,C,A])
e_path(A,B) ← node(A,gene), edge(A,C,participates_in), node(C,pathway),
      edge(D,C,participates_in), node(D,gene), edge(D,E,codes_for),
      node(E,protein), edge(B,E,refers_to), node(B,phenotype),
      nodes_distinct([B,E,D,C,A])
e_path(A,B) ← node(A,gene), edge(A,C,is_found_in),
      node(C,cellularcomponent), edge(D,C,is_found_in), node(D,protein),
      edge(B,D,refers_to), node(B,phenotype), nodes_distinct([B,D,C,A])
```

Fig. 3. Some explanation clauses for `path(A,B)`, connecting gene `A` to phenotype `B`.

such as Ensembl[3], NCBI Entrez[4], OMIM[5], and many others. They contain knowledge about various types of objects, such as genes, proteins, tissues, organisms, biological processes, and molecular functions. Information about their known or predicted relationships is also available, e.g., that gene A of organism B codes for protein C, which is expressed in tissue D, or that genes E and F are likely to be related since they co-occur often in scientific articles. Analysing such networks and data has been identified as an important and challenging task (see, e.g., [11]) and only few tools exist that support life scientists in this task.

Such a collection of interlinked heterogeneous biological data can be seen as a weighted network, where nodes are entities and the weight of an edge corresponds to the probability that the corresponding nodes are related [15]. It can thus be represented as a ProbLog database, which in the most simple case consists of probabilistic `edge/2` facts. Probabilities of the edges can be obtained from methods that predict their existence based on, e.g., co-occurrence frequencies or sequence similarities [15]. Within ProbLog, it is straightforward to add more information such as the type of relation encoded by an edge or explicit information on nodes, see Figure 3 for some examples.

Using ProbLog, one can pose queries that ask for the probability that a particular relationship holds. Typically, this will involve background knowledge of biological concepts as well as types of relations that form strong chains between two given nodes. In ProbLog this can be modeled by using a `path` predicate that computes the probability of a path as the product of the probabilities of the used edges, thus assuming that they are mutually independent [15, 5].

We implemented PEBL in Yap-5.1.2 and performed experiments in the context of the weighted biological database of [15]. As an example problem to be studied, we looked at connections between disease genes and the correspond-

[3] www.ensembl.org
[4] www.ncbi.nlm.nih.gov/Entrez/
[5] www.ncbi.nlm.nih.gov/Omim

ing phenotype for Alzheimer disease (resp. asthma). Since the key motivation for employing probabilistic explanation based learning is to be able to reason by analogy or to find similar examples, we set up experiments to answer the following questions:

Q1 Does PEBL produce meaningful examples when reasoning by analogy?
Q2 Can we find common explanations?
Q3 Can PEBL's explanations induced on one domain (say Alzheimer disease) be transferred to another one (say Asthma)?

To answer those questions, we extracted graphs around both diseases from the database. The genes were obtained by searching Entrez for human genes with the relevant annotation (AD or asthma); phenotypes are from OMIM. Most of the other information comes from EntrezGene, String, UniProt, HomoloGene, Gene Ontology, and OMIM databases. We did not include articles since they would dominate the graphs otherwise. Weights were assigned to edges as described in [15]. In the experiments below, we used a fixed number of randomly chosen (Alzheimer disease or asthma) genes for graph extraction. Subgraphs were extracted by taking all acyclic paths of length at most 4 or 5, and of probability at least 0.01, between any given gene and the corresponding phenotype. Some of the genes did not have any such paths to the phenotype and are thus disconnected from the rest of the graph. Table 1 gives a summary of the properties of the resulting graphs, obtained using two diseases, a varying number of annotated example genes, and a search depth of 4 or 5. (Remaining columns are explained below). Graphs Alz1 and Alz2 were obtained using the same 17 Alzheimer disease genes, but three of them were not connected to the Alzheimer disease phenotype with a path of length at most 4 (Alz1). Alz3 and Alz4 were extracted starting with all 142 annotated Alzheimer disease genes, and Ast1 and Ast2 with 17 asthma genes. As a basic representation, a modified path-predicate was employed that takes into account the node and edge types. We defined predicates related to node and edge types as operational. To answer question **Q1**, we start by studying example explanations for `path(A,B)` obtained from the graphs, where `A` is a gene and `B` a phenotype (Figure 3). These explanations are all semantically meaningful. For instance, the first one indicates that gene A is related to phenotype B if A belongs to a group of homologous (i.e., evolutionarily related) genes that relate to B. The three other explanations are based on interaction of proteins: either an explicit one, by participation in the same pathway, or by being found in the same cellular component. This last discovery suggests that a clause to describe different kinds of possible interactions would be a useful feature in the logical theory. It thus seems that PEBL can produce useful explanations and can help the user discover and synthesize new information, which answers **Q1** positively.

To further study the questions more objectively, we consider a slightly artifical setting. We define a target predicate `connect/3` as `connect(X,Y,Z) :- path(X,Z),path(Y,Z),path(X,Y).` This predicate succeeds if three nodes are connected to each other. We use examples where genes `X` and `Y` and phenotype `Z` are connected, and construct explanations on graphs Alz1 and Ast1.

	depth	nodes	edges	ag	ng	pt	pos	neg
Alz1	4	122	259	14	15	3	182	2254
Alz2	5	658	3544	17	20	4	272	5056
Alz3	4	351	774	72	33	3	5112	27648
Alz4	5	3364	17666	130	55	6	16770	187470
Ast1	4	127	241	7	12	2	42	642
Ast2	5	381	787	11	12	2	110	902

Table 1. Graph characteristics: search depth used during graph extraction, numbers of nodes and edges, number of genes annotated resp. not annotated with the corresponding disease and number of phenotypes, number of positive and negative examples for connecting two genes and a phenotype.

	Alz1						Ast1					
	pos(1)	pos(3)	pos(5)	pos_n	pos_a	prec	pos(1)	pos(3)	pos(5)	pos_n	pos_a	prec
Alz1	0.95	2.53	3.95	6.91	16.82	0.46	1.00	3.00	4.86	6.86	10.57	0.23
Alz2	0.84	2.24	3.60	7.37	18.65	0.42	0.86	2.86	4.71	6.86	14.56	0.22
Alz3	0.99	2.64	4.09	23.20	126.09	0.48	1.00	2.71	4.14	6.86	28.00	0.24
Alz4	0.84	2.23	3.58	7.37	18.80	0.42	0.86	2.29	3.43	5.14	28.00	0.15
Ast1	0.09	0.26	0.44	2.07	2.07	0.02	1.00	3.00	4.86	17.14	17.14	0.34
Ast2	0.08	0.23	0.38	2.00	2.00	0.01	0.86	2.57	4.29	16.57	16.57	0.20

Table 2. Averaged results over all examples learned on Alz1 (left column) resp. Ast1 (right column) and evaluated on 6 different graphs (lines Alz1–4, Ast1–2): number of positives among the first k answers (pos(k)), number of positives returned before the first negative (pos_n), absolute number of positives among examples with non-zero probability (pos_a), and precision w.r.t. all examples with non-zero probability (prec).

We consider ordered triplets `(G1,G2,P)` of nodes, where `G1` and `G2` are different genes and `P` is a phenotype. We call such a triplet positive with respect to the given network if both genes are annotated with the graph's disease and `P` is the corresponding phenotype, and negative otherwise. Thus for Alz1, there are $14 \cdot 13 \cdot 1 = 182$ positive and $29 \cdot 28 \cdot 3 - 182 = 2254$ negative triplets. Those numbers are also given in Table 1.

We use `connect(G1,G2,P)` for positive triplets as training examples. As `connect` is symmetric in the first two arguments, we only consider one ordering per pair of genes, which yields in total $14 \cdot 13/2 = 91$ training examples for Alz1 (resp. 21 for Ast1). The aim is to construct explanations for connections between the nodes of positive triplets, and use those to obtain for each test graph a ranked list of triplets covered by the explanation. To do so, we first compute the most likely explanation for each individual training example e, and then rank all instances covered by the resulting explanation clause according to their maximal probability. Table 2 summarizes classification accuracies obtained using those rankings and the classification criterion on triplets as described above. Values are averaged over all training examples. On graphs describing the same disease as the training graph, the top k instances for $k = 1, 3, 5$ are mostly positive, which again gives a positive answer to **Q1**. We obtained 26 different

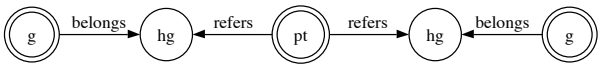

Fig. 4. One explanation for `connect(G1,G2,P)`, where double circles mark answer variables, and node types used are gene (g), phenotype (pt) and homologgroup (hg).

explanations from Alz1 and 3 different explanations from Ast1. Most explanations have been learned on at least two examples, and the biggest set of examples which shared most likely explanation contains 12 examples. Figure 4 shows an example explanation found both on Alz1 and on Ast1. This indicates a positive answer to question **Q2**, discovery of common explanations. The answer to **Q3**, if explanations can be transferred, is less conclusive: while transfer from Asthma to Alzheimer disease achieves good results, the other direction is a lot worse. However, one has to take into account that 23 of the 26 explanations learned on Alzheimer disease do not return any example at all on Asthma graphs, one only returns negative instances and the remaining 2 carry over very well, returning 30 resp 16 positive instances before returning negative ones. At the same time, two of the three explanations learned on Asthma were also learned on Alzheimer disease, which might explain their good performance there.

6 Conclusions and Related Work

We have introduced probabilistic explanation based learning, which applies principles of explanation based learning to probabilistic logic representations. Within this paper this was realized using a simple probablistic extension of Prolog, called ProbLog [5], even though the technique can easily be adapted towards other logics such as ICL [12] or PRISM [14]. Whereas the original version of ProbLog was intended as an inference engine for answering queries to a kind of probabilistic database efficiently, the present work is meant to support reasoning by analogy, similarity, or cases. The idea is that the most likely explanation for a particular example is employed to compute similar instances. Experiments on mining biological networks have shown promising results.

Probabilistic explanation based learning builds upon the existing work in both explanation based learning and probabilistic logics. From an explanation based learning point of view it is a simple extension of the formalisations within Prolog due to [8, 16] with probabilistic inference. From a probabilistic logic point of view, it extends the work of Poole [12] in that it finds *generalized* explanations. We have argued that using probabilities in the explanations provides a natural solution to the multiple explanation problem whereas the use of a resolution based proof procedure allows one to naturally deal with multiple examples and identify common explanations. Finally, the work is related to that on analogical, similarity and case based reasoning. In this regard, it provides a notion of similarity that is based on background knowledge (as in the definition of the `connect` predicate) as well as likelihood.

ProbLog and PEBL have been motivated by and applied to challenging and realistic biological network mining tasks [15]. Within the framework of ProbLog we have also studied the problem of compressing a large network from positive as well as negative examples [6]. Our future work will be aimed at making this set of tools available to the life scientist and to evaluate their utility there.

Acknowledgements This research has been partially supported by IQ (European Union Project IST-FET FP6-516169), Research Foundation-Flanders (FWO-Vlaanderen), Tekes and Humboldt foundation.

References

1. Cohen. W. (1992). Abductive Explanation-Based Learning: A Solution to the Multiple Inconsistent Explanation Problem. In Machine Learning, Vol. 8(2).
2. DeJong, G. (2004). Explanation-based learning. In A. Tucker, editor, Computer Science Handbook. CRC 2nd edition.
3. DeJong, G. (2006). Toward Robust Real-World Inference: A New Perspective on Explanation-Based Learning. In Proc. ECML/PKDD. Vol. 4212 of LNCS.
4. De Raedt, L., & Kersting, K. (2003). Probabilistic Logic Learning. In SIGKDD Explorations, Vol. 5 (2).
5. De Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: A probabilistic Prolog and its application in link discovery. In Proceedings of 20th International Joint Conference on Artificial Intelligence, pp. 2468 – 2473.
6. De Raedt, L., Kersting, K., Kimmig, A., Revoredo, K., & Toivonen, H. (2007). Compressing Probabilistic Prolog Programs. Machine Learning Journal, to appear.
7. Getoor, L., & Taskar, B. (2007). Editors. Statistical Relational Learning. MIT Press. to appear.
8. Hirsh, H. (1987). Explanation-based Generalization in a Logic-Programming Environment. In Proceedings of 15th International Joint Conference on Artificial Intelligence, pp 221 – 227.
9. Langley, P. (1989). Unifying themes in empirical and explanation-based learning. In Proceedings of the sixth international workshop on Machine learning, pp. 2-4.
10. Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47-80.
11. Perez-Iratxeta, C., Bork, P., & Andrade, M.A. (2002). Association of genes to genetically inherited diseases using data mining. Nature Genetics, 31, 316–319.
12. Poole, D. (2003) Probabilistic Horn abduction and Bayesian networks. In Artificial Intelligence. Vol. 64 (1).
13. Russel, S., & Norvig, P. (2002). Artificial Intelligence: A Modern Approach. 2nd edition. Prentice Hall.
14. Sato, T. & Kameya, Y. (2001). Parameter learning of logic programs for symbolic-statistical modeling. Journal of AI Research, 15, 391–454.
15. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., & Toivonen, H. (2006). Link discovery in graphs derived from biological databases. In U. Leser, F. Naumann, and B. Eckman (Eds.), Data Integration in the Life Sciences 2006. Vol. 4075 of *LNBI*. Springer.
16. Van Harmelen, F., & Bundy, A. (1988). Explanation-Based Generalisation = Partial Evaluation. Artificial Intelligence 36(3), pp. 401-412.