Statistical Predicate Invention

Stanley Kok KOKS@CS.WASHINGTON.EDU
Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Department of Computer Science & Engineering, University of Washington, Seattle, WA 98195, USA

Abstract

We propose statistical predicate invention as a key problem for statistical relational learning. SPI is the problem of discovering new concepts, properties and relations in structured data, and generalizes hidden variable discovery in statistical models and predicate invention in ILP. We propose an initial model for SPI based on second-order Markov logic, in which predicates as well as arguments can be variables, and the domain of discourse is not fully known in advance. Our approach iteratively refines clusters of symbols based on the clusters of symbols they appear in atoms with (e.g., it clusters relations by the clusters of the objects they relate). Since different clusterings are better for predicting different subsets of the atoms, we allow multiple cross-cutting clusterings. We show that this approach outperforms Markov logic structure learning and the recently introduced infinite relational model on a number of relational datasets.

1. Introduction

In the past few years, the statistical relational learning (SRL) community has recognized the importance of combining the strengths of statistical learning and relational learning (also known as inductive logic programming), and developed several novel representations, as well as algorithms to learn their parameters and structure (Getoor & Taskar, 2007). However, the problem of statistical predicate invention (SPI) has so far received little attention in the community. SPI is the discovery of new concepts, properties and relations from data, expressed in terms of the observable ones, using statistical techniques to guide the process and explicitly representing the uncertainty in the discovered predicates. These can in turn be used as a basis for discovering new predicates, which is potentially much more powerful than learning based on a fixed set of simple primitives. Essentially all the concepts used by humans can be viewed as invented predicates, with many levels of discovery between them and the sensory percepts they are ultimately based on.

In statistical learning, this problem is known as hidden or latent variable discovery, and in relational learning as predicate invention. Both hidden variable discovery and predicate invention are considered quite important in their respective communities, but are also very difficult, with limited progress to date.

One might question the need for SPI, arguing that structure learning is sufficient. Such a question can also be directed at hidden variable discovery and predicate invention, and their benefits, as articulated by their respective communities, also apply to SPI. SPI produces more compact and comprehensible models than pure structure learning, and may also improve accuracy by representing unobserved aspects of the domain. Instead of directly modeling dependencies among observed predicates, which potentially requires an exponential number of parameters, we can invent a predicate and model the dependence between it and each of the observed predicates, requiring only a linear number of parameters and reducing the risk of overfitting. In turn, invented predicates can be used to learn new formulas, allowing larger search steps, and potentially enabling us to learn more complex models accurately.

Among the prominent approaches in statistical learning is a series of algorithms developed by Elidan, Friedman and coworkers for finding hidden variables in Bayesian networks. Elidan et al. (2001) look for structural patterns in the network that suggest the presence of hidden variables. Elidan and Friedman (2005) group observed variables by their mutual information, and create a hidden variable for each group. Central to both approaches is some form of EM algorithm

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

that iteratively creates hidden variables, hypothesizes their values, and learns the parameters of the resulting Bayesian network. A weakness of such statistical approaches is they assume that the data is independently and identically distributed, which is not true in many real-world applications.

In relational learning, the problem is known as predicate invention (see Kramer (1995) for a survey). Predicates are invented to compress a first-order theory, or to facilitate the learning of first-order formulas. Relational learning employs several techniques for predicate invention. Predicates can be invented by analyzing first-order formulas, and forming a predicate to represent either their commonalities (interconstruction (Wogulis & Langley, 1989)) or their differences (intraconstruction (Muggleton & Buntine, 1988)). A weakness of inter/intraconstruction is that they are prone to over-generating predicates, many of which are not useful. Predicates can also be invented by instantiating second-order templates (Silverstein & Pazzani, 1991), or to represent exceptions to learned rules (Srinivasan et al., 1992). Relational predicate invention approaches suffer from a limited ability to handle noisy data.

Only a few approaches to date combine elements of statistical and relational learning. Most of them only cluster objects, not relations (Popescul & Ungar, 2004; Wolfe & Jensen, 2004; Neville & Jensen, 2005; Xu et al., 2005; Long et al., 2006; Roy et al., 2006). Craven and Slattery (2001) proposed a learning mechanism for hypertext domains in which class predictions produced by naive Bayes are added to an ILP system (FOIL) as invented predicates.[1] The SAYU-VISTA system (Davis et al., 2007) uses an off-the-shelf ILP system (Aleph) to learn Horn clauses on a database. It creates a predicate for each clause learned, adds it as a relational table to the database, and then runs a standard Bayesian network structure learning algorithm (TAN). Both of these systems predict only a single target predicate. We would like SPI to find arbitrary regularities over all predicates. The state-of-the-art is the infinite relational model (IRM, Kemp et al. (2006)), which simultaneously clusters objects and relations. The objects can be of more than one type, and the relations can take on any number of arguments. Xu et al. (2006) propose a closely related model.

In this paper, we present MRC, an algorithm based on Markov logic (Richardson & Domingos, 2006), as a first step towards a general framework for SPI. MRC automatically invents predicates by clustering objects, attributes and relations. The invented predicates capture arbitrary regularities over all relations, and are not just used to predict a designated target relation. MRC learns multiple clusterings, rather than just one, to represent the complexities in relational data. MRC is short for Multiple Relational Clusterings.

We begin by briefly reviewing Markov logic in the next section. We then describe our model in detail (Section 3). Next we report our experiments comparing our model with Markov logic structure learning and IRM (Section 4). We conclude with a discussion of future work.

2. Markov Logic

In first-order logic, formulas are constructed using four types of symbols: constants, variables, functions, and predicates. (In this paper we use only function-free logic.) Constants represent objects in the domain of discourse (e.g., people: `Anna`, `Bob`, etc.). Variables (e.g., `x`, `y`) range over the objects in the domain. Predicates represent relations among objects (e.g., `Friends`), or attributes of objects (e.g., `Student`). Variables and constants may be typed. An *atom* is a predicate symbol applied to a list of arguments, which may be variables or constants (e.g., `Friends(Anna, x)`). A *ground atom* is an atom all of whose arguments are constants (e.g., `Friends(Anna, Bob)`). A *world* is an assignment of truth values to all possible ground atoms. A database is a partial specification of a world; each atom in it is true, false or (implicitly) unknown.

Markov logic is a probabilistic extension of first-order logic. A *Markov logic network (MLN)* is a set of weighted first-order formulas. Together with a set of constants representing objects in the domain, it defines a Markov network (Pearl, 1988) with one node per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability distribution over possible worlds x specified by the ground Markov network is given by

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_{i \in F} \sum_{j \in G_i} w_i g_j(x)\right) \qquad (1)$$

where Z is a normalization constant, F is the set of all first-order formulas in the MLN, G_i is the set of groundings of the ith first-order formula, and $g_j(x) = 1$ if the jth ground formula is true and $g_j(x) = 0$ otherwise. Markov logic enables us to compactly represent complex models in non-i.i.d. domains. General algorithms for inference and learning in Markov logic are

[1] To our knowledge, this is the only previous paper that uses the term 'statistical predicate invention'.

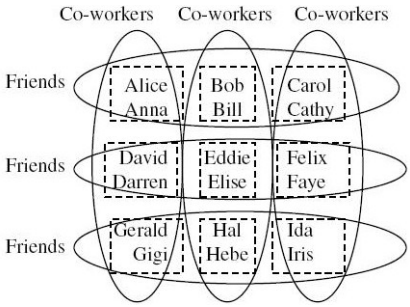

Figure 1. Example of multiple clusterings.

discussed in Richardson and Domingos (2006).

3. Multiple Relational Clusterings

Predicate invention is the creation of new symbols, together with formulas that define them in terms of the symbols in the data. (In a slight abuse of language, we use "predicate invention" to refer to the creation of both new predicate symbols and new constant symbols.) In this section we propose a statistical approach to predicate invention based on Markov logic. The simplest instance of statistical predicate invention is clustering, with each cluster being an invented unary predicate. More generally, all latent variables in i.i.d. statistical models can be viewed as invented unary predicates. Our goal in this paper is to extend this to relational domains, where predicates can have arbitrary arity, objects can be of multiple types, and data is non-i.i.d.

We call our approach MRC, for Multiple Relational Clusterings. MRC is based on the observation that, in relational domains, multiple clusterings are necessary to fully capture the interactions between objects. Consider the following simple example. People have coworkers, friends, technical skills, and hobbies. A person's technical skills are best predicted by her coworkers's skills, and her hobbies by her friends' hobbies. If we form a single clustering of people, coworkers and friends will be mixed, and our ability to predict both skills and hobbies will be hurt. Instead, we should cluster together people who work together, and simultaneously cluster people who are friends with each other. Each person thus belongs to both a "work cluster" and a "friendship cluster." (See Figure 1.) Membership in a work cluster is highly predictive of technical skills, and membership in a friendship cluster is highly predictive of hobbies. The remainder of this section presents a formalization of this idea and an efficient algorithm to implement it.

Notice that multiple clusterings may also be useful in propositional domains, but the need for them there is less acute, because objects tend to have many fewer properties than relations. (For example, `Friends(Anna,x)` can have as many groundings as there are people in the world, and different friendships may be best predicted by different clusters Anna belongs to.)

We define our model using finite second-order Markov logic, in which variables can range over relations (predicates) as well as objects (constants). Extending Markov logic to second order involves simply grounding atoms with all possible predicate symbols as well as all constant symbols, and allows us to represent some models much more compactly than first-order Markov logic. We use it to specify how predicate symbols are clustered.

We use the variable r to range over predicate symbols, x_i for the ith argument of a predicate, γ_i for a cluster of ith arguments of a predicate (i.e., a set of symbols), and Γ for a clustering (i.e., a set of clusters or, equivalently, a partitioning of a set of symbols). For simplicity, we present our rules in generic form for predicates of all arities and argument types, with n representing the arity of r; in reality, if a rule involves quantification over predicate variables, a separate version of the rule is required for each arity and argument type.

The first rule in our MLN for SPI states that each symbol belongs to at least one cluster:

$$\forall x \; \exists \gamma \; x \in \gamma$$

This rule is hard, i.e., it has infinite weight and cannot be violated. The second rule states that a symbol cannot belong to more than one cluster in the same clustering:

$$\forall x, \gamma, \gamma', \Gamma \;\; x \in \gamma \wedge \gamma \in \Gamma \wedge \gamma' \in \Gamma \wedge \gamma \neq \gamma' \Rightarrow x \notin \gamma'$$

This rule is also hard. We call it the *mutual exclusion* rule.

If r is in cluster γ_r and x_i is in cluster γ_i, we say that $r(x_1, \ldots, x_n)$ is in the *combination of clusters* $(\gamma_r, \gamma_1, \ldots, \gamma_n)$. The next rule says that each atom appears in exactly one combination of clusters, and is also hard:

$$\forall r, x_1, \ldots, x_n \; \exists! \gamma_r, \gamma_1, \ldots, \gamma_n \\ r \in \gamma_r \wedge x_1 \in \gamma_1 \wedge \ldots \wedge x_n \in \gamma_n$$

The next rule is the key rule in the model, and states that the truth value of an atom is determined by the cluster combination it belongs to:

$$\forall r, x_1, \ldots, x_n, +\gamma_r, +\gamma_1, \ldots, +\gamma_n \\ r \in \gamma_r \wedge x_1 \in \gamma_1 \wedge \ldots \wedge x_n \in \gamma_n \Rightarrow r(x_1, \ldots, x_n)$$

This rule is soft. The "+" notation is syntactic sugar that signifies the MLN contains an instance of this rule *with a separate weight* for each tuple of clusters $(\gamma_r, \gamma_1, \ldots, \gamma_n)$. As we will see below, this weight is the log-odds that a random atom in this cluster combination is true. Thus, this is the rule that allows us to predict the probability of query atoms given the cluster memberships of the symbols in them. We call this the *atom prediction* rule. Combined with the mutual exclusion rule, it also allows us to predict the cluster membership of evidence atoms. Chaining these two inferences allows us to predict the probability of query atoms given evidence atoms.

To combat the proliferation of clusters and consequent overfitting, we impose an exponential prior on the number of clusters, represented by the formula

$$\forall \gamma \exists x \; x \in \gamma$$

with negative weight $-\lambda$. The parameter λ is fixed during learning, and is the penalty in log-posterior incurred by adding a cluster to the model. Thus larger λs lead to fewer clusterings being formed.[2]

A *cluster assignment* $\{\Gamma\}$ is an assignment of truth values to all $r \in \gamma_r$ and $x_i \in \gamma_i$ atoms. The MLN defined by the five rules above represents a joint distribution $P(\{\Gamma\}, R)$ over $\{\Gamma\}$ and R, the vector of truth assignments to the observable ground atoms. Learning consists of finding the cluster assignment that maximizes $P(\{\Gamma\}|R) \propto P(\{\Gamma\}, R) = P(\{\Gamma\})P(R|\{\Gamma\})$, and the corresponding weights. $P(\{\Gamma\}) = 0$ for any $\{\Gamma\}$ that violates a hard rule. For the remainder, $P(\{\Gamma\})$ reduces to the exponential prior. It is easily seen that, given a cluster assignment, the MLN decomposes into a separate MLN for each combination of clusters, and the weight of the corresponding atom prediction rule is the log odds of an atom in that combination of clusters being true. (Recall that, by design, each atom appears in exactly one combination of clusters.) Further, given a cluster assignment, atoms with unknown truth values do not affect the estimation of weights, because they are graph-separated from all other atoms by the cluster assignment. If t_k is the empirical number of true atoms in cluster combination k, and f_k the number of false atoms, we estimate w_k as $\log((t_k + \beta)/(f_k + \beta))$, where β is a smoothing parameter.

Conversely, given the model weights, we can use inference to assign probabilities of membership in combinations of clusters to all atoms. Thus the learning problem can in principle be solved using an EM algorithm, with cluster assignment as the E step, and MAP estimation of weights as the M step. However, while the M step in this algorithm is trivial, the E step is extremely complex. We begin by simplifying the problem by performing hard assignment of symbols to clusters (i.e., instead of computing probabilities of cluster membership, a symbol is simply assigned to its most likely cluster). Since performing an exhaustive search over cluster assignments is infeasible, the key is to develop an intelligent tractable approach. Since, given a cluster assignment, the MAP weights can be computed in closed form, a better alternative to EM is simply to search over cluster assignments, evaluating each assignment by its posterior probability. This can be viewed as a form of structure learning, where a structure is a cluster assignment.

Algorithm 1 shows the pseudo-code for our learning algorithm, MRC. The basic idea is the following: when clustering sets of symbols related by atoms, each refinement of one set of symbols potentially forms a basis for the further refinement of the related clusters. MRC is thus composed of two levels of search: the top level finds clusterings, and the bottom level finds clusters. At the top level, MRC is a recursive procedure whose inputs are a cluster of predicates γ_r per arity and argument type, and a cluster of symbols γ_i per type. In the initial call to MRC, each γ_r is the set of all predicate symbols with the same number and type of arguments, and γ_i is the set of all constant symbols of the ith type. At each step, MRC creates a cluster symbol for each cluster of predicate and constant symbols it receives as input. Next it clusters the predicate and constant symbols, creating and deleting cluster symbols as it creates and destroys clusters. It then calls itself recursively with each possible combination of the clusters it formed. For example, suppose the data consists of binary predicates $r(x_1, x_2)$, where x_1 and x_2 are of different type. If r is clustered into γ_r^1 and γ_r^2, x_1 into x_1^1 and x_1^2, and x_2 into x_2^1 and x_2^2, MRC calls itself recursively with the cluster combinations $(\gamma_r^1, \gamma_1^1, \gamma_2^1)$, $(\gamma_r^1, \gamma_1^1, \gamma_2^2)$, $(\gamma_r^1, \gamma_1^2, \gamma_2^1)$, $(\gamma_r^1, \gamma_1^2, \gamma_2^2)$, $(\gamma_r^2, \gamma_1^1, \gamma_2^1)$, etc.

Within each recursive call, MRC uses greedy search with restarts to find the MAP clustering of the subset of predicate and constant symbols it received. It begins by assigning all constant symbols of the same type to a a single cluster, and similarly for predicate symbols of the same arity and argument type. The search operators used are: move a symbol between clusters, split a cluster, and merge a cluster. (If clusters are large, only a random subset of the splits is tried at each step.) A greedy search ends when no operator increases posterior probability. Restarts are performed,

[2] We have also experimented with using a Chinese restaurant process prior (CRP, Pitman (2002)), and the results were similar. We thus use the simpler exponential prior.

Algorithm 1 *MRC*(C,R)

inputs: $C = (\gamma_{r_1}, \ldots, \gamma_{r_m}, \gamma_1, \ldots, \gamma_n)$, a combination of clusters, where γ_{r_i} is a cluster of relation symbols with the same number and type of arguments, and γ_j is a cluster of constant symbols of the same type
R, ground atoms formed from the symbols in C
output: $D = \{(\gamma'_{r_1}, \ldots, \gamma'_{r_m}, \gamma'_1, \ldots, \gamma'_n)\}$, a set of cluster combinations where $\gamma'_i \subseteq \gamma_i$
note: Γ_i is a clustering of the symbols in γ_i, i.e., $\Gamma_i = \{\gamma_i^1, \ldots, \gamma_i^k\}$, $\gamma_i^j \subseteq \gamma_i, \bigcup_{j=1}^k \gamma_i^j = \gamma_i$, and $\gamma_i^j \cap \gamma_i^k = \emptyset, j \neq k$. $\{\Gamma_i\}$ is a set of clusterings.

```
Γ_i ← {γ_i} for all γ_i in C
{Γ_i^Best} ← {Γ_i}
for s ← 0 to MaxSteps do
  {Γ_i^Tmp} ← best change to any clustering in {Γ_i}
  if P({Γ_i^Tmp}|R) > P({Γ_i}|R)
    {Γ_i} ← {Γ_i^Tmp}
    if P({Γ_i}|R) > P({Γ_i^Best}|R)
      {Γ_i^Best} ← {Γ_i}
  else if for the last MaxBad consecutive iterations
       P({Γ_i^Tmp}|R) ≤ P({Γ_i}|R)
    reset Γ_i ← {γ_i} for all γ_i in C
if Γ_i^Best = {γ_i} for all γ_i in C
  return C
D ← ∅
for each C' ∈ Γ_{r_1}^Best × ... × Γ_{r_m}^Best × Γ_1^Best × ... × Γ_n^Best
  R' ← ground atoms formed from the symbols in C'
  D ← D ∪ MRC(C',R')
return D
```

and they give different results because of the random split operator used.

Notice that in each call of MRC, it forms a clustering for each of its input clusters, thereby always satisfying the first two hard rules in the MLN. MRC also always satisfies the third hard rule because it only passes the atoms in the current combination to each recursive call.

MRC terminates when no further refinement increases posterior probability, and returns the finest clusterings produced. In other words, if we view MRC as growing a tree of clusterings, it returns the leaves. Conceivably, it might be useful to retain the whole tree, and perform shrinkage (McCallum et al., 1998) over it. This is an item for future work. Notice that the clusters created at a higher level of recursion constrain the clusters that can be created at lower levels, e.g., if two symbols are assigned to different clusters at a higher level, they cannot be assigned to the same cluster in subsequent levels. Notice also that predicate symbols of different arities and argument types are never clustered together. This is a limitation that we plan to overcome in the future.

4. Experiments

In our experiments, we compare MRC with IRM (Kemp et al., 2006) and MLN structure learning (Kok & Domingos, 2005).

4.1. Infinite Relational Model

The IRM is a recently-published model that also clusters objects, attributes, and relations. However, unlike MRC, it only finds a single clustering. It defines a generative model for the predicates and cluster assignments. Like MRC, it assumes that the predicates are conditionally independent given the cluster assignments, and the cluster assignments for each type are independent. IRM uses a Chinese restaurant process prior (CRP, Pitman (2002)) on the cluster assignments. Under the CRP, each new object is assigned to an existing cluster with probability proportional to the cluster size. Because the CRP has the property of exchangeability, the order in which objects arrive does not affect the outcome. IRM assumes that the probability p of an atom being true conditioned on cluster membership is generated according to a symmetric Beta distribution, and that the truth values of atoms are then generated according to a Bernoulli distribution with parameter p. IRM finds the MAP cluster assignment using the same greedy search as our model, except that it also searches for the optimal values of its CRP and Beta parameters.

4.2. MLN Structure Learning

We also compare MRC to Kok and Domingos' (2005) MLN structure learning algorithm (MSL, beam search version) implemented in the Alchemy package (Kok et al., 2006). MSL begins by creating all possible unit clauses. Then, at each step, it creates candidate clauses by adding literals to the current clauses. The weight of each candidate clause is learned by optimizing a weighted pseudo-log-likelihood (WPLL) measure, and the best one is added to the MLN. The algorithm continues to add the best candidate clauses to the MLN until none of the candidates improves WPLL.

4.3. Datasets

We compared MRC to IRM and MSL on all four datasets used in Kemp et al. (2006).[3]

Animals. This dataset contains a set of animals and their features (Osherson et al., 1991). It consists exclusively of unary predicates of the form `f(a)`, where `f` is a feature and `a` is an animal (e.g., `Swims(Dolphin)`). There are 50 animals, 85 features, and thus a total of 4250 ground atoms, of which 1562 are true. This is a simple propositional dataset with no relational struc-

[3]The IRM code and datasets are publicly available at http://web.mit.edu/∼ckemp/www/code/irm.html.

ture, but it is useful as a "base case" for comparison. Notice that, unlike traditional clustering algorithms, which only cluster objects by features, MRC and IRM also cluster features by objects. This is known as biclustering or co-clustering, and has received considerable attention in the recent literature (e.g., Dhillon et al. (2003)).

UMLS. UMLS contains data from the Unified Medical Language System, a biomedical ontology (McCray, 2003). It consists of binary predicates of the form $\mathtt{r}(\mathtt{c}, \mathtt{c}')$, where $\mathtt{c}$ and $\mathtt{c}'$ are biomedical concepts (e.g., `Antibiotic`, `Disease`), and $\mathtt{r}$ is a relation between them (e.g., `Treats`, `Diagnoses`). There are 49 relations and 135 concepts, for a total of 893,025 ground atoms, of which 6529 are true.

Kinship. This dataset contains kinship relationships among members of the Alyawarra tribe from Central Australia (Denham, 1973). Predicates are of the form $\mathtt{k}(\mathtt{p}, \mathtt{p}')$, where $\mathtt{k}$ is a kinship relation and $\mathtt{p}$, $\mathtt{p}'$ are persons. There are 26 kinship terms and 104 persons, for a total of 281,216 ground atoms, of which 10,686 are true.

Nations. This dataset contains a set of relations among nations and their features (Rummel, 1999). It consists of binary and unary predicates. The binary predicates are of the form $\mathtt{r}(\mathtt{n}, \mathtt{n}')$, where $\mathtt{n}$, $\mathtt{n}'$ are nations, and $\mathtt{r}$ is a relation between them (e.g., `ExportsTo`, `GivesEconomicAidTo`). The unary predicates are of the form $\mathtt{f}(\mathtt{n})$, where $\mathtt{n}$ is a nation and $\mathtt{f}$ is a feature (e.g., `Communist`, `Monarchy`). There are 14 nations, 56 relations and 111 features, for a total of 12,530 ground atoms, of which 2565 are true.

4.4. Methodology

Experimental evaluation of statistical relational learners is complicated by the fact that in many cases the data cannot be separated into independent training and test sets. While developing a long-term solution for this remains an open problem, we used an approach that is general and robust: performing cross-validation by atom. For each dataset, we performed ten-fold cross-validation by randomly dividing the atoms into ten folds, training on nine folds at a time, and testing on the remaining one. This can be seen as evaluating the learners in a *transductive* setting, because an object (e.g., `Leopard`) that appears in the test set (e.g., in `MeatEater(Leopard)`) may also appear in the training set (e.g., in `Quadrapedal(Leopard)`). In the training data, the truth values of the test atoms are set to `unknown`, and their actual values (`true`/`false`) are not available. Thus learners must perform generalization in order to be able to infer the test atoms, but the generalization is aided by the dependencies between test atoms and training ones.

Notice that MSL is not directly comparable to MRC and IRM because it makes the closed-world assumption, i.e., all atoms not in its input database are assumed to be false. Our experiments require the test atoms to be open-world. For an approximate comparison, we set all test atoms to false when running MSL. Since in each run these are only 10% of the training set, setting them to false does not greatly change the sufficient statistics (true clause counts) learning is based on. We then ran MC-SAT (Poon & Domingos, 2006) on the MLNs learned by MSL to infer the probabilities of the test atoms.[4]

To evaluate the performance of MRC, IRM and MSL, we measured the average conditional log-likelihood of the test atoms given the observed training ones (CLL), and the area under the precision-recall curve (AUC). The advantage of the CLL is that it directly measures the quality of the probability estimates produced. The advantage of the AUC is that it is insensitive to the large number of true negatives (i.e., atoms that are false and predicted to be false). The precision-recall curve for a predicate is computed by varying the threshold CLL above which an atom is predicted to be true.

For IRM, we used all of the default settings in its publicly available software package (except that we terminated runs after a fixed time rather than a fixed number of iterations). For our model, we set both parameters λ and β to 1 (without any tuning). We ran IRM for ten hours on each fold of each dataset. We also ran MRC for ten hours per fold, on identically configured machines, for the first level of clustering. Subsequent levels of clustering were permitted 100 steps. MRC took a total of 3-10 minutes for the subsequent levels of clustering, negligible compared to the time required for the first level and by IRM. We allowed a much longer time for the first level of clustering because this is where the sets of objects, attributes and relations to be clustered are by far the largest, and finding a good initial clustering is important for the subsequent learning.

4.5. Results

Figure 2 reports the CLL and AUC for MRC, IRM and MSL, averaged over the ten folds of each dataset. We also report the results obtained using just the initial

[4] The parameters for MSL are specified in an online appendix at http://alchemy.cs.washington.edu/papers/-kok07.

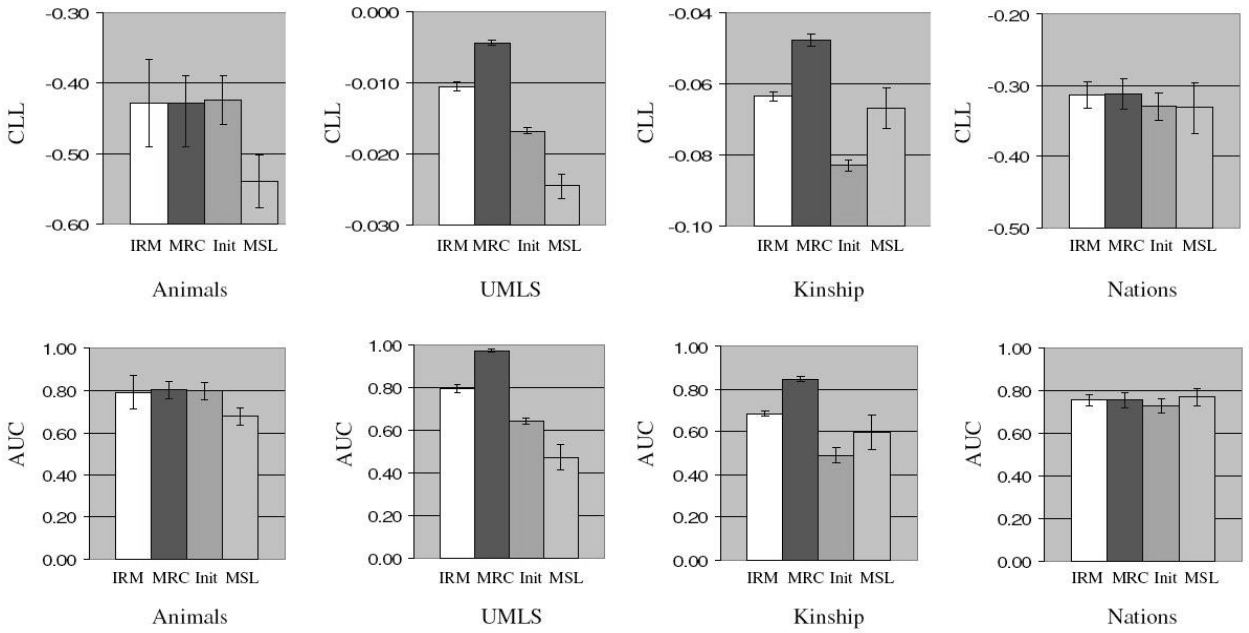

Figure 2. Comparison of MRC, IRM and MLN structure learning (MSL) using ten-fold cross-validation: average conditional log-likelihood of test atoms (CLL) and average area under the precision-recall curve (AUC). Init is the initial clustering formed by MRC. Error bars are one standard deviation in each direction.

clustering formed by MRC, in order to evaluate the usefulness of learning multiple clusterings.

MSL does worse than MRC and IRM on all datasets except Nations. On Nations, it does worse than MRC and IRM in terms of CLL, but approximately ties them in terms of AUC. Many of the relations in Nations are symmetric, e.g., if country A has a military conflict with B, then the reverse is usually true. MSL learns a rule to capture the symmetry, and consequently does well in terms of AUC.

MRC outperforms IRM on UMLS and Kinship, and ties it on Animals and Nations. The difference on UMLS and Kinship is quite large. Animals is the smallest and least structured of the datasets, and it is conceivable that it has little room for improvement beyond a single clustering. The difference in performance between MRC and IRM correlates strongly with dataset size. (Notice that UMLS and Kinship are an order of magnitude larger than Animals and Nations.) This suggests that sophisticated algorithms for statistical predicate invention may be of most use in even larger datasets, which we plan to experiment with in the near future.

MRC outperforms Init on all domains except Animals. The differences on Nations are not significant, but on UMLS and Kinship they are very large. These results show that forming multiple clusterings is key to the good performance of MRC. In fact, Init does considerably worse than IRM on UMLS and Kinship; we attribute this to the fact that IRM performs a search for optimal parameter values, while in MRC these parameters were simply set to default values without any tuning on data. This suggests that optimizing parameters in MRC could lead to further performance gains.

In the Animals dataset, MRC performs at most three levels of cluster refinement. On the other datasets, it performs about five. The average total numbers of clusters generated are: Animals, 202; UMLS, 405; Kinship, 1044; Nations, 586. The average numbers of atom prediction rules learned are: Animals, 305; UMLS, 1935; Kinship, 3568; Nations, 12,169. We provide examples of multiple clusterings that MRC learned for the UMLS dataset in an online appendix (see footnote 4).

5. Conclusion and Future Work

We proposed statistical predicate invention, the discovery of new concepts, properties and relations in structured data, as a key problem for statistical relational learning. We then introduced MRC, an approach to SPI based on second-order Markov logic. MRC forms multiple relational clusterings of the symbols in the data and iteratively refines them. Empirical comparisons with a Markov logic structure learning system and a state-of-the-art relational clustering system on four datasets show the promise of our model.

Directions for future work include: evaluating our model in an inductive (rather than transductive) setting; experimenting on larger datasets; using the clusters learned by our model as primitives in structure learning (Kok & Domingos, 2005); learning a hierarchy of multiple clusterings and performing shrinkage over them; and developing more powerful models for SPI (e.g., by allowing clustering of predicates with different arities).

We speculate that all relational structure learning can be accomplished with SPI alone. Traditional relational structure learning approaches like ILP build formulas by incrementally adding predicates that share variables with existing predicates. The dependencies these formulas represent can also be captured by inventing new predicates. For example, consider a formula that states that if two people are friends, either both smoke or neither does: $\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Fr}(\mathtt{x},\mathtt{y}) \Rightarrow (\mathtt{Sm}(\mathtt{x}) \Leftrightarrow \mathtt{Sm}(\mathtt{y}))$. SPI can compactly represent this using two clusters, one containing friends who smoke, and one containing friends who do not. The model we introduced in this paper represents a first step in this direction.

Acknowledgements

We thank Charles Kemp for answering our questions on the IRM. This work was partly funded by DARPA grant FA8750-05-2-0283 (managed by AFRL), DARPA contract NBCH-D030010, NSF grant IIS-0534881, and ONR grant N00014-05-1-0313. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

Craven, M., & Slattery, S. (2001). Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, *43*.

Davis, J., Ong, I., Struyf, J., Burnside, E., Page, D., & Costa, V. S. (2007). Change of representation for statistical relational learning. *Proc. IJCAI'07*.

Denham, W. (1973). *The detection of patterns in Alyawarra nonverbal behavior*. Doctoral dissertation, Department of Anthropology, University of Washington, Seattle, WA.

Dhillon, I. S., Mallela, S., & Modha, D. S. (2003). Information-theoretic co-clustering. *Proc. KDD'03*.

Elidan, G., & Friedman, N. (2005). Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, *6*.

Elidan, G., Lotner, N., Friedman, N., & Koller, D. (2001). Discovering hidden variables: A structure-based approach. *NIPS'01*.

Getoor, L., & Taskar, B. (Eds.). (2007). *Introduction to statistical relational learning*. MIT Press.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., & Ueda, N. (2006). Learning systems of concepts with an infinite relational model. *Proc. AAAI'06*.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. *Proc. ICML'05*.

Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., & Domingos, P. (2006). *The Alchemy system for statistical relational AI* (Technical Report). Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Kramer, S. (1995). *Predicate invention: A comprehensive view* (Technical Report). Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Long, B., Zhang, Z. M., Wu, X., & Yu, P. S. (2006). Spectral clustering for multi-type relational data. *Proc. ICML'06*.

McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. *Proc. ICML'98*.

McCray, A. T. (2003). An upper level ontology for the biomedical domain. *Comparative and Functional Genomics*, *4*.

Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predicates by inverting resolution. *Proc. ICML'88*.

Neville, J., & Jensen, D. (2005). Leveraging relational autocorrelation with latent group models. *Proc. ICDM'05*.

Osherson, D. N., Stern, J., Wilkie, O., Stob., M., & Smith, E. E. (1991). Default probability. *Cognitive Science*, *15*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann.

Pitman, J. (2002). *Combinatorial stochastic processes* (Technical Report 621). Department of Statistics, University of California at Berkeley, Berkeley, CA.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. *Proc. AAAI'06*.

Popescul, A., & Ungar, L. H. (2004). Cluster-based concept invention for statistical relational learning. *Proc. KDD'04*.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*.

Roy, D., Kemp, C., Mansinghka, V. K., & Tenenbaum, J. B. (2006). Learning annotated hierarchies from relational data. *NIPS'06*.

Rummel, R. J. (1999). Dimensionality of nations project: attributes of nations and behavior of nation dyads, 1950 -1965. ICPSR data file.

Silverstein, G., & Pazzani, M. J. (1991). Relational clichés: Constraining constructive induction during relational learning. *Proc. ICML'91*.

Srinivasan, A., Muggleton, S. H., & Bain, M. (1992). Distinguishing exceptions from noise in non-monotonic learning. *Proc. ILP'92*.

Wogulis, J., & Langley, P. (1989). Improving efficiency by learning intermediate concepts. *Proc. IJCAI'89*.

Wolfe, A. P., & Jensen, D. (2004). Playing multiple roles: discovering overlapping roles in social networks. *Proc. of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*.

Xu, Z., Tresp, V., Yu, K., & Kriegel, H.-P. (2006). Infinite hidden relational models. *Proc. UAI'06*.

Xu, Z., Tresp, V., Yu, K., Yu, S., & Kriegel, H.-P. (2005). Dirichlet enhanced relational learning. *Proc. ICML'05*.