In *Proceedings of the ICML Workshop on Structural Knowledge Transfer for Machine Learning*. Pittsburgh, PA, July 2006.

Transfer Learning with Markov Logic Networks

Lilyana Mihalkova LILYANAM@CS.UTEXAS.EDU
Raymond J. Mooney MOONEY@CS.UTEXAS.EDU
University of Texas, Department of Computer Sciences, 1 University Station, C0500, Austin, TX 78712

Abstract

We propose a new algorithm for transfer learning of Markov Logic Network (MLN) structure. An important aspect of our approach is that it first diagnoses the provided source MLN and then focuses on re-learning only the incorrect portions. Experiments in a pair of synthetic domains demonstrate that this strategy significantly decreases the search space and speeds up learning while maintaining a level of accuracy comparable to that of the current best algorithm.

1. Introduction

Traditional machine learning algorithms operate under the assumption that learning for each new task starts from scratch, thus disregarding knowledge gained in previous domains. Naturally, if the domains encountered during learning are related, this approach would waste both data and computer time to develop hypotheses that could be recovered by examining and possibly slightly modifying previously acquired knowledge. The field of transfer learning, which has recently greatly increased in popularity, addresses the problem of how to leverage previous knowledge in order to improve the efficiency and accuracy of learning on a new task that is related to the original one. Transfer learning approaches have been developed for a variety of learning settings, including reinforcement learning (Torrey et al., 2005), and Bayesian nets (Niculescu-Mizil & Caruana, 2005), among others.

We propose a new transfer learning algorithm for MLNs. MLNs are a powerful formalism that combines the expressiveness of first-order logic with the flexibility of probability (Richardson & Domingos, 2006). There are two aspects to learning an MLN—the structure and the weights. While weight learning is relatively quick, structure learning is very computationally intensive. Therefore, MLN structure learning, which is the focus of the present work, could particularly benefit from transfer.

We assume the following set-up. A source MLN is learned in the original task and is provided to the learner, along with a mapping from the predicates of the original domain to those of the target domain. Recovering this mapping automatically is another very interesting research problem, but for now we assume the mapping is simply given. A similar assumption is made by Torrey et al. (2005).

The current state-of-the-art structure learning algorithm (Kok & Domingos, 2005), which we will call Alchemy after the open-source system that implements it[1] (Kok et al., 2005), can start learning either from scratch or from a provided MLN and can therefore be used for transfer. However, Alchemy does not explicitly attempt to assess the similarities between tasks or take advantage of them. As a result, it could search through an unnecessarily large number of structures and take a long time to complete. Our proposed algorithm successfully diagnoses the source MLN and exploits the similarities between the tasks by focusing on relearning only the inaccurate parts. In this way, it significantly decreases both the learning time and the number of hypotheses considered, while maintaining a level of learning accuracy similar to that of Alchemy.

2. Background

2.1. Markov Logic Networks

An MLN (Richardson & Domingos, 2006) consists of a set of first-order logic formulae, each with a weight attached, and provides a model for the joint distribution of a set of variables. A useful way of viewing MLNs is as templates for producing fully-grounded Markov networks (Pearl, 1988) when a set of constants is provided. As described by Richardson and Domingos (2006), an MLN, L, can be used to construct a Markov network by including a node for each grounding of each predicate appearing in L and a feature for

Appearing in the *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, June 29, Pittsburgh, PA.

[1] In addition to implementing this algorithm, Alchemy also includes capabilities for performing inference and weight learning.

each formula in L. The value of a particular node is given by the truth value of the corresponding ground literal; similarly, the value of each feature is 1 if the corresponding ground formula is true and 0 otherwise.

To answer a query about the probability of a set of ground literals or formulae, one can perform Gibbs sampling over the Markov Network. Gibbs sampling starts by assigning a random truth value to each query literal. It then proceeds in rounds, recomputing the probability of a ground literal X given its Markov Blanket MB_X (i.e. its neighboring nodes). As given by Richardson and Domingos (2006), this probability is recomputed using the following equation:

$$P(X = x|MB_X = m) = \frac{e^{S_X(x,m)}}{e^{S_X(0,m)} + e^{S_X(1,m)}} \quad (1)$$

where, if F is the set of ground formulae in which X participates, S_X is defined as follows.

$$S_X(x,m) = \sum_{f_i \in F} w_i f_i(X = x, MB_x = m) \quad (2)$$

It is not necessary to fully ground the MLN in order to perform inference on it—formulae that are already trivially satisfied by the evidence can be omitted because they have no effect on the value of Equation (1), and the only ground literals that need to appear in the Markov network are the query variables and those that are present in the Markov blanket of a literal with an unknown value.

Kok and Domingos (2005) introduce an algorithm for learning MLN structure that can start either from an empty MLN or from a previously-constructed one. Candidate clauses are generated by considering all possible additions and deletions of literals to the existing clauses as well as all possible sign flips. Two search strategies are proposed—beam search, which maintains a beam of best clauses, and shortest-first search, which considers adding shorter clauses before moving on to longer ones. Candidates are scored using a weighted pseudo-log-likelihood measure. In this paper, we compare to the faster, beam search, version of the algorithm, which we call Alchemy after its open source implementation (Kok et al., 2005).

3. New Algorithm

Recall that the learner is given the MLN obtained from the source domain and a mapping from the predicates in the source domain to those in the target domain. In addition, we assume that the formulae of the provided MLN are disjunctions of literals. The learner is *not* told which parts of the source MLN are useful in the new task and which may need to be relearned. Thus, the algorithm first needs to diagnose the given MLN.

The general skeleton of our algorithm proceeds in two stages and is similar to that of FORTE (Richards & Mooney, 1995), which revises first-order theories.

1. **Self-Diagnosis:** In this step, the algorithm inspects the given MLN and determines for each formula whether it is too general, too specific, or requires no change. The purpose of this step is to focus the search for new formulae to those parts of the MLN that truly need to be updated.

2. **Structure Update:** In this step we carry out the actual updates to the clauses by specializing the ones marked as too general and generalizing those marked as too specific.

Next, we describe these steps in more detail.

3.1. Self-Diagnosis

One natural approach to self-diagnosis is to attempt to use the source MLN while observing where its formulae fail. In the case of FORTE where the formulae are part of a first-order theory, this is done by attempting to prove positive examples in the data. Our self-diagnosis algorithm proceeds analogously.

At the onset, the learner is provided with a source MLN and a relational dataset. Each of the predicates in the target domain is examined in turn. The current predicate under examination is denoted as P^*. The algorithm performs a slightly modified version of Gibbs sampling with P^* serving as a query predicate whose groundings have their values set to unknown, while evidence is given by the values of all other predicate groundings in the data. In each round of sampling, in addition to recalculating the probability of a ground literal X, the algorithm considers all clauses in which X participates. Even though the truth value of X is set to unknown for the purposes of sampling, its value is known from the data. Let the actual value of X be v (true or false).

Each participating clause C can be placed in one of four bins with respect to X. For the purposes of exposition, let $\sigma = false$ if X appears negated in C and $\sigma = true$ if X appears non-negated in C. For a running example, we will use the following simple relational database: $\{Student(Ann)$, $\neg HasJob(Ann)$, $Sleepy(Ann)$, $Sociable(Ann)$, $InClass(Ann)\}$ where $P^* = Student$, $X = Student(Ann)$, and $v = true$.

- **[Applies;Good]** The value of X is crucial in evaluating C, with C being true only when $X = v$, as in $\neg InClass(Ann) \vee Student(Ann)$.

- **[Applies;Bad]** C is true only when $X = \neg v$, e.g. $\neg Sociable(Ann) \vee \neg Student(Ann)$.

- **[Does not apply;Good]** C is true regardless of the value of X (i.e. it holds trivially), and $\sigma \neq v$. For example, $Sleepy(Ann) \vee \neg Student(Ann)$.

- [**Does not apply;Bad**] C is trivially true, and $\sigma = v$, e.g. $\neg HasJob(Ann) \vee Student(Ann)$.

This taxonomy is motivated by a close inspection of Equation (1). The probability of $X = x$ is increased only by clauses in the [**Applies;Good**] bin and is decreased by clauses in the [**Applies;Bad**] bin. Clauses in the other two bins do not have an effect on this equation. However, if some of the literals other than X in a [**Does not apply;Bad**] clause, are deleted so that it no longer holds trivially, it will be moved to the [**Applies;Good**] bin and will help to increase the probability of the correct value of X. Similarly, if we add some literals to an [**Applies;Bad**] clause so that it becomes trivially satisfied, it will enter the [**Does not apply;Good**] bin and will no longer decrease the probability of the correct value of X.

With these observations in mind, we can complete the description of the self-diagnosis step. As the probability of a literal is recalculated in each iteration of Gibbs sampling, for each clause in which the literal participates, we keep a tally of the number of times it falls into each of the bins. Finally, if a clause was placed in the [**Applies;Bad**] bin more than p percent of time, it is marked for specialization and if it fell in the [**Does not apply; Bad**] bin more than p percent of time, it is marked for generalization. We anticipated that in the highly sparse relational domains in which we tested, clauses would fall mostly in the [**Does not apply; Good**] bin. To prevent this bin from swamping the other ones, we set p to the low value of 10%.

This process is repeated for each predicate, P^*, in the target domain.

3.2. Structure Updates

The updates are performed using beam search starting from the clauses identified in the previous step. Unlike Kok and Domingos (2005), however, we do not consider all possible additions and deletions of a literal to each clause. Rather, we only try removing literals from the clauses marked as too specific and we try literal additions only to the clauses marked as too general. These restrictions apply also to the candidates produced from a particular clause. The candidates are scored using the weighted pseudo-log-likelihood measure of Kok and Domingos (2005). Thus, the search space is constrained first by limiting the number of clauses considered for updates, and second, by restricting the kind of update performed on each clause.

4. Experimentation

4.1. Data and Methodology

We used two synthetic domains—Academic (the source), which provides knowledge about academic departments and is similar to that of Richardson and Domingos (2006) but contains fewer predicates, and Industrial (the target), which provides an analogous description of a company. Figure 1 lists the mapping between the predicates in the two domains. The domains additionally contain equality predicates. Each training example represents a single organization and contains between 50 to 150 true ground literals. To emphasize the size of each example, we call it a *mega-example*. Mega-examples are artificially generated by first producing a skeleton by fixing the values of the groundings of the unary predicates and partially specifying some of the binary ones and then performing maximum aposteriori inference over a hand-written MLN to assign values to the unspecified groundings.

Academic	**Industrial**
President(X)	Chair(X)
Professor(X)	Employee(X)
Student(X)	Intern(X)
AdvisedBy(X, Y)	SupervisedBy(X, Y)
Publication(P, X)	Project(P, X)
Area(A, X)	Department(A, X)
None	Secretary(X)
None	AssistedBy(X, Y)

Figure 1. Predicate mappings in the two domains

We compared the performance of our new algorithm (**TransferNew**) to that of Alchemy, starting both from scratch (**ScratchAlchemy**) and from the same source MLN provided to the new algorithm (**TransferAlchemy**). Our accuracy metrics were the area under the precision-recall curve (AUC) and the conditional log-likelihood (CLL), as used in prior work (Kok & Domingos, 2005). Each point on the learning curves is the average of 5 independent learning runs where accuracy at each point was measured on an independently generated mega-example, different for each run. Testing was done for the predicates *supervisedBy* and *secretary*. The former was picked because it represents an interesting relation, and the latter—because it was absent in the source domain. The algorithms were *not* told on which predicates they would be tested. All timing experiments were run one by one on the same dedicated machine.

4.2. Results

Figures 2 and 3 compare the accuracy of the three systems on each of the metrics. The error bars give the standard error at each point. The accuracy of the two transfer systems is closely matched on both metrics. The fact that it far exceeds that of **ScratchAlchemy** and improves dramatically after observing a single mega-example, demonstrates that the source MLN captures useful information about the target, which

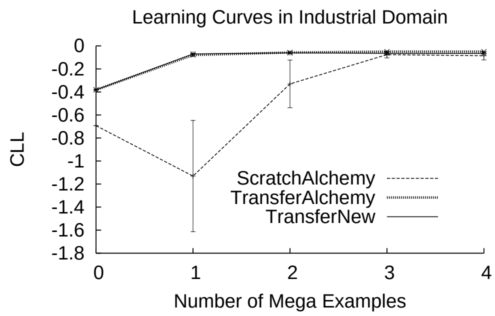

Figure 2. Accuracy on CLL

can be easily recovered after only a few updates. Even though **TransferNew** and **TransferAlchemy** perform similarly, the former considers much fewer candidate clauses during beam search, as listed in Figure 4 and has a significantly reduced running time, as shown in Figure 5. Moreover, the running time of TransferNew shows much less variability accross training runs. This demonstrates the effectiveness of the self-diagnosis step and suggests that our algorithm would be especially well-suited to situations where quick online relearning is important, such as when using MLNs to represent the model of a dynamic environment.

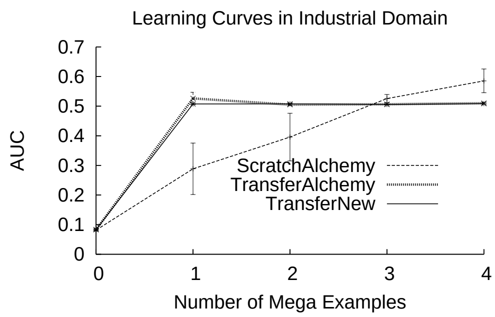

Figure 3. Accuracy on AUC

	TransferNew		**TransferAlchemy**	
Num. Exs.	Mean	Std. Dev.	Mean	Std. Dev.
1	2,645	721	21,610	5,503
2	2,311	1,003	10,555	3,391
3	2,474	441	7,253	2,090
4	2,025	623	7,332	1,393

Figure 4. Average number of candidate clauses considered by the transfer systems

5. Future Work and Conclusions

This paper proposes a new MLN transfer learning algorithm that diagnoses the source MLN and updates only the inaccurate clauses, thus decreasing both the search space and the learning time, while maintaining the accuracy at the level of the current state-of-the-art MLN structure learning algorithm.

	TransferNew		**TransferAlchemy**		
Num. Expls	Mean	St. Dev.	Mean	St. Dev.	Speed-up Factor
1	26	14	690	1165	26.1
2	92	95	835	566	9.1
3	133	43	2320	768	17.4
4	218	130	7208	5597	33.0

Figure 5. Average learning times of the transfer systems (in seconds)

We are currently working on further improving the performance of our learner by adapting relational pathfinding (Richards & Mooney, 1995), a technique for discovering new first-order logic clauses bottom-up instead of via top-down greedy search. In addition, we are planning to test the algorithms on real data.

Acknowledgments

This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed by the Air Force Research Laboratory (AFRL) under contract FA8750-05-2-0283. 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, AFRL, or the United States Government. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. *Proceedings of 22nd International Conference on Machine Learning (ICML-2005).* Bonn,Germany.

Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). *The Alchemy system for statistical relational AI* (Technical Report). Department of Computer Science and Engineering, University of Washington. http://www.cs.washington.edu/ai/alchemy.

Niculescu-Mizil, A., & Caruana, R. (2005). Learning the structure of related tasks. *Proceedings of NIPS-2005 Workshop on Inductive Transfer: 10 Years Later.*

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

Richards, B. L., & Mooney, R. J. (1995). Automated refinement of first-order Horn-clause domain theories. *Machine Learning*, *19*, 95–131.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Torrey, L., Walker, T., Shavlik, J., & Maclin, R. (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. *Proceedings of the 16th European Conference on Machine Learning (ECML-05).* Porto, Portugal.