Transfer Learning by Mapping and Revising Boosted Relational Dependency Networks

Rodrigo Azevedo Santos[1], **Aline Paes**[2], **Gerson Zaverucha**[1]

[1]PESC/COPPE– Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ – Brazil

[2]Departamento de Ciência da Computação – Universidade Federal Fluminense (UFF)
Niterói, RJ – Brazil

Abstract. *Statistical machine learning algorithms usually assume that there is considerably-size data to train the models. However, they would fail in addressing domains where data is difficult or expensive to obtain. Transfer learning has emerged to address this problem of learning from scarce data by relying on a model learned in a source domain where data is easy to obtain to be a starting point for the target domain. On the other hand, real-world data contains objects and their relations, usually gathered from noisy environment. Finding patterns through such uncertain relational data has been the focus of the Statistical Relational Learning (SRL) area. Thus, to address domains with* scarce, relational, and uncertain data*, in this paper, we propose TreeBoostler, an algorithm that transfers the SRL state-of-the-art Boosted Relational Dependency Networks learned in a source domain to the target domain. TreeBoostler first finds a mapping between pairs of predicates to accommodate the additive trees into the target vocabulary. After, it employs two theory revision operators devised to handle incorrect relational regression trees aiming at improving the performance of the mapped trees. In the experiments presented in this paper, TreeBoostler has successfully transferred knowledge among several distinct domains. Moreover, it performs comparably or better than learning from scratch methods in terms of accuracy and outperforms a transfer learning approach in terms of accuracy and runtime.*

Accepted in the journal Machine Learning (https://doi.org/10.1007/s10994-020-05871-x). Research supported by CNPq, FAPERJ and CAPES.

1. Introduction

Machine learning algorithms have been successfully used in many areas such as computer vision, robotics, etc. However, this success usually comes with the presence of large amounts of data. When the number of examples is relatively small, learning good models can be a challenging task. This is often the case of several real-world problems where collecting data is expensive or even impossible to obtain. To handle this issue, transfer learning techniques [Pan and Yang 2010] leverage a model learned from a source domain with more examples to learn from another related domain where data is more scarce.

Transfer learning has been widely employed in classical machine learning settings. However, most of them do not take into account the relationships between entities of the domain and the fact that the examples may not be identically and in-

dependently distributed, which is the case of a number of real-world data. In addition, real-world data have noise and are generally uncertain which is the focus of the area called Statistical Relational Learning (SRL) [Getoor and Taskar 2007]. Transfer Learning algorithms have also been developed in the context of SRL. Two of these algorithms [Davis and Domingos 2009, Van Haaren et al. 2015] transfer relational knowledge by creating a second-order representation of formulas from learned Markov Logic Networks (MLN) [Richardson and Domingos 2006]. Other three algorithms [Mihalkova et al. 2007, Mihalkova and Mooney 2009, Kumaraswamy et al. 2015] find predicate mappings through search methods to perform transference of clauses learned from MLNs by mapping their predicates.

In this work, we present a transfer learning algorithm called TreeBoostler that transfers Boosted Relational Dependency Networks [Natarajan et al. 2012] by mapping the predicates appearing in the trees. At a higher level, the algorithm generates the possible predicate mappings as it tries to recursively transfer nodes from the source regression trees. After finding such mappings, they are propagated to the rest of the tree and the other trees of the next iterations. To adjust the mapped trees to the new target domain, TreeBoostler also includes a theory revision [Wrobel 1996, Paes et al. 2017] algorithm for proposing modifications to the mapped models in order to handle incorrectness and improve the performance.

We evaluated TreeBoostler in several real-world datasets and simulated the scenario where only a few data are available by training on one single fold and testing on the remaining folds. Our results demonstrate that our method has successfully transferred learned knowledge across different domains in a smaller time compared to other transfer learning algorithms. In addition, transference showed to be very useful in terms of accuracy compared to learning from scratch methods based on RDNs. Additional experiments were performed to investigate the behavior of the algorithm as the number of examples increases and when provided minimal target data. The results demonstrate that our algorithm can be very competitive to traditional methods that learn from scratch even with the increase of the amount of data, also when provided only with a few examples.

2. Contributions

To sum up, the main contributions of this dissertation include:

- A transfer learning algorithm, namely TreeBoostler, that constructs a target set of relational regression trees biased by a predicate mapping found through the transfer process given the structure of the source regression trees. This is found by applying all legal mappings to a node and selecting the one which gives the best split.
- A revision theory system that proposes modifications to boosted trees through two revision operators. These revision operators are: (1) pruning operator, which deletes nodes from a tree and (2) expansion operator, which expandes nodes in each tree.
- Three types of experiments to evaluate TreeBoostler against baseline approaches. The experiments were conducted as follows: (1) simulating a transfer learning environment with limited target data, (2) providing to the system a scenario with increasing amounts of target data and (3) providing a scenario with learning from minimal target data.

3. TreeBoostler: The proposed algorithm

We propose a method that transfers learned boosted trees from a source domain to a target domain. The approach is divided into two major steps: first, the source boosted trees structure is transferred to the target domain by finding an adequate predicate mapping, and second, the algorithm revises its trees by pruning and expanding nodes in order to better fit the target data. Figure 1 illustrates the entire process of the algorithm.

3.1. Transferring the structure

A fundamental problem when tackling transfer learning on relational domains is to automatically find how to map the source vocabulary to the target domain. In this way, the first step of the overall process is to find this mapping, where we reduce the overall vocabulary of both domains to their set of predicates. With that, the boosted trees learned from the source domain are transferred sequentially to the target domain and the parameters relearned to fit the target data. There are two approaches for establishing a predicate mapping: (1) a global mapping, which finds a corresponding target predicate to each source predicate and applies this mapping to the entire source structure (i.e. all clauses) at once; and (2) a local mapping, which finds an independent predicate mapping for each independent part of the entire structure (i.e. each clause).

In this work, we choose to follow the local approach, by finding the best local predicate mapping for transferring the boosted trees. Thus, the algorithm translates the predicates presented in the inner nodes according to the previously found translations in order to keep the found predicates mapping through the entire process of learning trees.

3.2. Revising the structure

When transferring learned theories from one domain to another it is usually not enough to map the vocabularies from both domains to achieve a model representative of the target domain [Mihalkova et al. 2007]. Such theories may contain multiple faults that prevent them to correctly predict examples due to the difference in the distribution of both domains. These faults can be repaired through the process of theory revision. The main idea of theory revision is to search for points in the theory that are preventing the examples to be correctly classified and propose modifications to them. In a transfer learning scenario, the revision process attempts to adjust the initial mapped source theory to fit the target data. The goal is to achieve more accurate theories due to the fact that the theory revision allows the learning algorithm to build clauses from partial or incomplete theories that would otherwise not be found in the constrained search space.

Our theory revision component follows the three major steps:

1. Searching for paths in the trees responsible for bad predictions of examples and defining them as revision points.
2. Proposing possible modifications to the revision points by applying the revision operators.
3. Scoring both transferred and revised theory and choosing to stay with the best one.

The revision points, which are responsible for ”bad”predictions, need to be modified during the revision process in order to increase accuracy. We define as a revision point any leaf that has a ”bad”weighted variance. Arguably, modifications on the paths ending

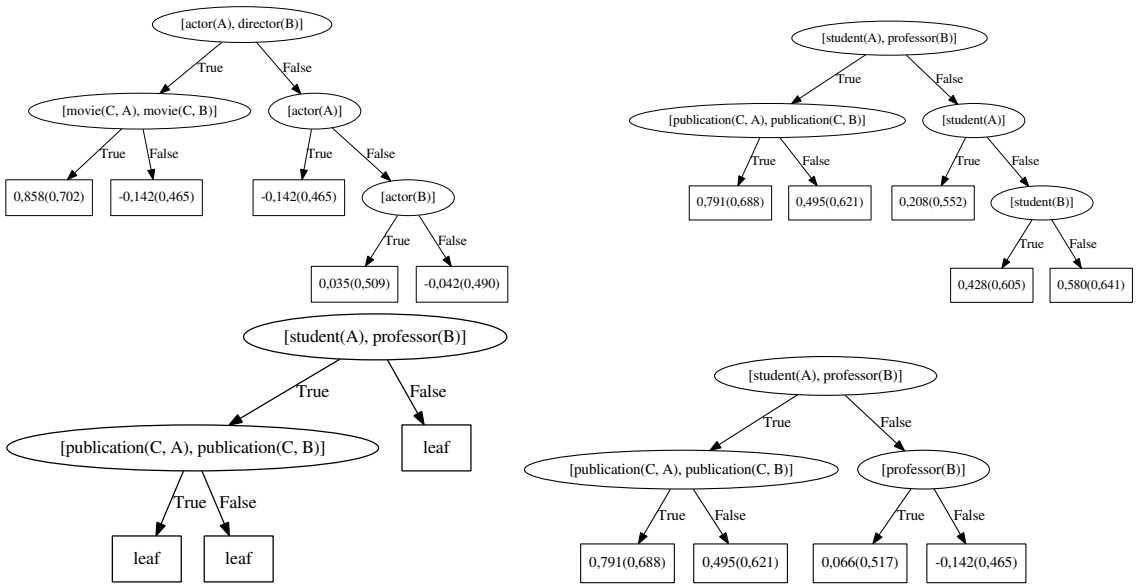

Figura 1. The transfer learning process stages with their respective trees. Obtained from source domain by learning from scratch (top-left); transferred by mapping predicates (top-right); after the pruning process (down-left) and after the expansion of nodes (down-right).

up on such leaves will change the way an example is covered resulting in a differently weighted variance.

We considered two types of revision operators: (1) a pruning operator, which increases the coverage of examples by deleting nodes from a tree (and in such a way, it may be seen as a generalization operator); and (2) an expansion operator, which decreases the coverage of examples by expanding nodes in each tree (in the same way, it can be seen as a specialization operator). We describe them as follows:

- **Pruning** operator prunes the tree from the bottom to top by removing a node whose children are leaves marked as revision points.
- **Expansion** operator recursively adds nodes that give the best split in a leaf considered as a revision point.

4. Experimental Results and Conclusions

We conducted the experiments considering the following questions:

- Does it learn more accurate models than the baselines?
- Does theory revision improve the performance of the transfer process?
- Does it transfer well across domains?
- Is it faster than the baselines?
- Does it perform better than the baselines with increasing amount of examples in the target data?
- Does it perform better than the baselines with minimal target data?

We have performed three types of experiments: (1) an approach simulating a transfer learning environment with limited target data, (2) a scenario with increasing amounts

Tabela 1. Results on transference from Twitter to Yeast dataset and NELL Sports domain to Finances domain.

	Twitter → Yeast				NELL Sports → NELL Finances			
Algorithm	CLL	AUC ROC	AUC PR	Time	CLL	AUC ROC	AUC PR	Time
RDN	-0.182	0.695	0.081	4.46 s	-0.180	0.532	0.020	4.59 s
RDN-B	-0.257	0.919	0.231	18.80 s	-0.317	0.713	0.083	22.12 s
TODTLER	-0.023	0.497	0.002	39 min	NA	NA	NA	NA
TreeBoostler*	-0.180	**0.986**	0.273	4.14 s	-0.164	0.978	0.062	46.63 s
TreeBoostler	-0.180	0.986	0.272	60.99 s	**-0.161**	**0.979**	0.074	229.36 s

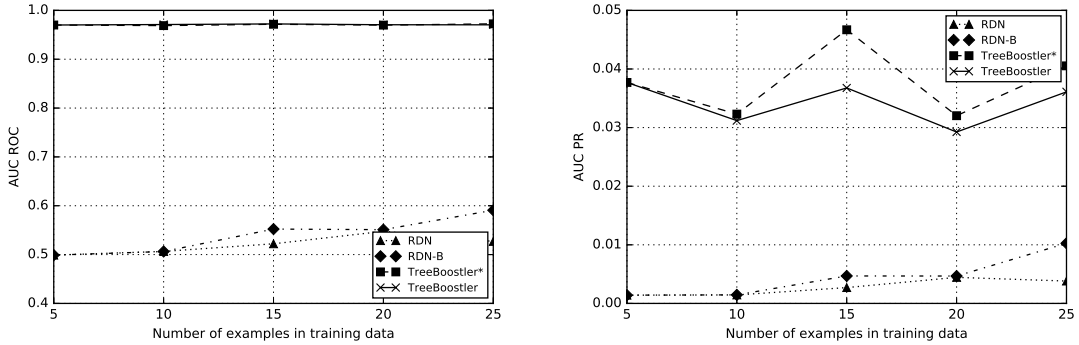

Figura 2. Learning curves from minimal target data for AUC ROC (left) and AUC PR (right) obtained from NELL Sports → NELL Finances.

of target data and (3) a scenario that represents learning from minimal target data. The first question is important to evaluate the algorithm and conclude if it performs better than learning from scratch approaches and related transfer learning approaches. The second question evaluates the effectiveness of a theory revision process and demonstrates that the process is capable of improving the performance of the transferred model in the target domain. The third question addresses if the transfer process is capable of providing good models while the fourth question asks if the algorithm is faster than related transfer learning algorithms and also learning from scratch algorithms. The fifth question investigate the behavior of the algorithm with increasing amounts of data while the sixth question addresses the problem of minimal target data where the learner is provided with only a few examples.

We compared the performance of TreeBoostler against two baseline approaches that learn from scratch from target data: RDN-Boost [Natarajan et al. 2012], which learns a set of regression trees using boosting method and RDN [Neville and Jensen 2007] which learns a single large regression tree. We also compared it against TODTLER [Van Haaren et al. 2015], a transfer learning method that lifts a source structure to second-order logic. In the results we presented two stages of the algorithm: structure transference (TreeBoostler*) and the complete transfer system including theory revision (TreeBoostler).

The most relevant results are presented in Table 1 and Figure 2. It can be observed that our algorithms are competitive or better than TODTLER and learning from scratch

methods. Bold results are significantly better than the performance of all baselines for at least one TreeBostler algorithm. The statistical significance was measured using a paired t-test at the 95% confidence level. Considering the minimal target data problem, as shown in Figure 2, our algorithms easily outperform the learning from scratch algorithms RDN-B and RDN. The results show that only mapping the predicates and learning the parameters for the mapped trees may be very useful when target training data is scarce. In addition, the results presented in the work answer positively all the questions. TreeBoostler is capable of transferring well across domains and learning more accurate models. For experiments considering other datasets, TreeBoostler was also capable of learning faster than baselines and the theory revision showed an improvement in the performance.

References

Davis, J. and Domingos, P. (2009). Deep transfer via second-order markov logic. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Kumaraswamy, R., Odom, P., Kersting, K., Leake, D., and Natarajan, S. (2015). Transfer learning via relational type matching. In *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)*, ICDM '15, pages 811–816. IEEE Computer Society.

Mihalkova, L., Huynh, T., and Mooney, R. J. (2007). Mapping and revising markov logic networks for transfer learning. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 608–614. AAAI Press.

Mihalkova, L. and Mooney, R. (2009). Transfer learning from minimal target data by mapping across relational domains. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1163–1168, Pasadena, CA.

Natarajan, S., Khot, T., Kersting, K., Gutmann, B., and Shavlik, J. (2012). Gradient-based boosting for statistical relational learning: The relational dependency network case. *Mach. Learn.*, 86(1):25–56.

Neville, J. and Jensen, D. D. (2007). Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692.

Paes, A., Zaverucha, G., and Costa, V. S. (2017). On the use of stochastic local search techniques to revise first-order logic theories from examples. *Machine Learning*, 106(2):197–241.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

Van Haaren, J., Kolobov, A., and Davis, J. (2015). TODTLER: Two-order-deep transfer learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Conference on Artificial Intelligence*.

Wrobel, S. (1996). First order theory refinement. In *Advances in inductive logic programming*. IOS Press.