Imitation Learning in Relational Domains Using Functional Gradient Boosting

Sriraam Natarajan, Saket Joshi[+], Prasad Tadepalli[+], Kristian Kersting[#], Jude Shavlik[*]
Wake Forest University School of Medicine, [+] Oregon State University
[#] Fraunhofer IAIS, Germany, [*] University of Wisconsin-Madison

It is common knowledge that both humans and animals learn new skills by observing others. This problem, which is called *imitation learning*, can be formulated as learning a representation of a policy – a mapping from states to actions – from examples of that policy. Our focus is on relational domains where states are naturally described by relations among an indefinite number of objects. Examples include real time strategy games such as Warcraft, regulation of traffic lights, logistics, and a variety of planning domains. In this work, we employ two ideas. First, instead of learning a deterministic policy to imitate the expert, we learn a stochastic policy where the probability of an action given a state is represented by a sum of potential functions. Second, we leverage the recently developed functional-gradient boosting approach to learn a set of regression trees, each of which represents a potential function. The functional gradient approach has been found to give state of the art results in many relational problems [6, 5]. Together, these two ideas allow us to overcome the limited representational capacity of earlier approaches, while also giving us an effective learning algorithm. Indeed, the functional-gradient approach to boosting has already been found to yield excellent results in imitation learning in robotics in propositional setting [7].

Functional Gradient Boosting: Assume that the training examples are of the form $(\mathbf{x}_i, y_i)$ for $i = 1, ..., N$ and $y_i \in \{1, ..., K\}$. The goal is to fit a model $P(y|\mathbf{x}) \propto e^{\psi(y,\mathbf{x})}$. The standard method of supervised learning is based on gradient-descent where the learning algorithm starts with initial parameters θ_0 and computes the gradient of the likelihood function. Dietterich et al. used a more general approach to train the potential functions based on Friedman's [3] gradient-tree boosting algorithm where the potential functions are represented by sums of regression trees that are grown stage-wise.

In this work, the goal is to find a policy μ that is captured by the trajectories provided by the expert and represented using features $\mathbf{f_i^j}$ of the state and actions a_i^j of the expert, i.e., the goal is to determine a policy $\mu = P(a_i|\mathbf{f_i}; \psi)$ $\forall a, i$ where the features are relational. These features could define the objects in the domain (squares in a gridworld, players in robocup, blocks in blocksworld etc.), their relationships (type of objects, teammates in robocup etc.), or temporal relationships (between current state and previous state) or some information about the world (traffic density at a signal, distance to the goal etc.). We assume a functional parametrization over the policy and consider the conditional distribution over actions a_i given the features to be $P(a_i|\mathbf{f_i}; \psi) = e^{\psi(a_i;\mathbf{f_i})} / \sum_{a_i'} e^{\psi(a_i';\mathbf{f_i})}$, $\forall a_i \in \mathbf{A}$ where $\psi(a_i; \mathbf{f_i})$ is the potential function of a_i given the grounding $\mathbf{f_i}$ of the feature predicates at state s_i and the normalization is over all the admissible actions in the current state. Instead of computing the functional gradient over the potential function, the functional gradients are computed for each training example (i.e., trajectories):$\Delta_m(a_i^j; \mathbf{f_i^j}) = \nabla_\psi \sum_j \sum_i log(P(a_i^j|\mathbf{f_i^j}; \psi))|_{\psi_{m-1}}$. These are point-wise gradients for examples $\langle \mathbf{f_i^j}, a_i^j \rangle$ on each state i in each trajectory j conditioned on the potential from the previous iteration(shown as $|_{\psi_{m-1}}$). Now this set of local gradients form a set of training examples for the gradient at stage m. The most important step in functional gradient boosting is the fitting of a regression function (typically a regression tree) h_m on the training examples $[(\mathbf{f_i^j}, a_i^j), \Delta_m(a_i^j; \mathbf{f_i^j})]$ [3]. Dietterich et al. [2] point out that although the fitted function h_m is not exactly the same as the desired Δ_m, it will point in the same direction (assuming that there are enough training examples). So ascent in the direction of h_m will approximate the true functional gradient.

Proposition 0.1. *The functional gradient w.r.t* $\psi(a_i^j; \mathbf{f_i^j})$ *of the likelihood for each example* $\langle \mathbf{f_i^j}, a_i^j \rangle$ *is given by:*
$\frac{\partial \log P(a_i^j|\mathbf{f_i^j};\psi)}{\partial \psi(\hat{a}_i^j;\mathbf{f_i^j})} = I(a_i^j = \hat{a}_i^j|\mathbf{f_i^j}) - P(a_i^j|\mathbf{f_i^j}; \psi)$
where $\hat{a}_i^j$ *is the action observed from the trajectory and* I *is the indicator function that is* 1 *if* $a_i^j = \hat{a}_j^i$ *and* 0 *otherwise.*

The expression is very similar to the one derived in [2]. The key feature of the above expression is that the functional gradient at each state of the trajectory is dependent on the observed action $\hat{a}$. If the example is positive (i.e., it is an

action executed by the expert), the gradient $(I - P)$ is positive indicating that the policy should increase the probability of choosing the action. On the contrary if the example is a negative example (i.e., for all other actions), the gradient is negative implying that it will push the probability of choosing the action towards 0. Following prior work [4, 6, 5], we use *Relational Regression Trees* (RRTs)[1] to fit the gradient function at every feature in the training example (since we are in the relational setting). These trees upgrade the attribute-value representation used within classical regression trees. The key idea in this work is to represent the distribution over each action as a set of RRTs on the features. These trees are learned such that at each iteration the new set of RRTs aim to maximize the likelihood of the distributions w.r.t ψ. Hence, when computing $P(a(X)|\mathbf{f}(X))$ for a particular value of state variable X (say x), each branch in each tree is considered to determine the branches that are satisfied for that particular grounding (x) and their corresponding regression values are added to the potential ψ. For example, X could be a certain block in the blocksworld, a player in Robocup or a square in the gridworld.

Experiments: For our experiments we chose four domains. These domains range from blocksworld which has a long tradition in planning, to a grid world domain that has a rich relational hierarchical structure to a multi-agent system for traffic control to Robocup domain that has continuous features. We compare the performance of our tree-boosting for relational imitation learning *TBRIL* system to a classical RRT learner, *TILDE* [1] and to propositional functional gradient boosting (*PFGB*) and empirically show the superiority of our method. In the blocksworld domain, the goal is to reach a target configuration where there is no tower of more than three blocks. In the traffic signal domain, the goal is to ensure smooth flow of traffic by controlling the signals (8 different configurations each for 4 signals). In the gridworld domain, the goal is to navigate the grid by opening doors and collect resources or destroy enemies. In the Robocup domain, the task is to breakaway from an opposition player in a 2-on-1 game. For each domain, we obtained traces of an expert's policy and created training and test sets. Our performance metric is the area under the precision-recall (PR) curve in the test set against the number of trajectories from the expert. The results are presented in Figure 1. As can be seen, *TBRIL* system outperforms both TILDE and *PFGB* systems. PFGB showed very poor performance in the blocksworld and the traffic signal domain and hence we do not present the results here.

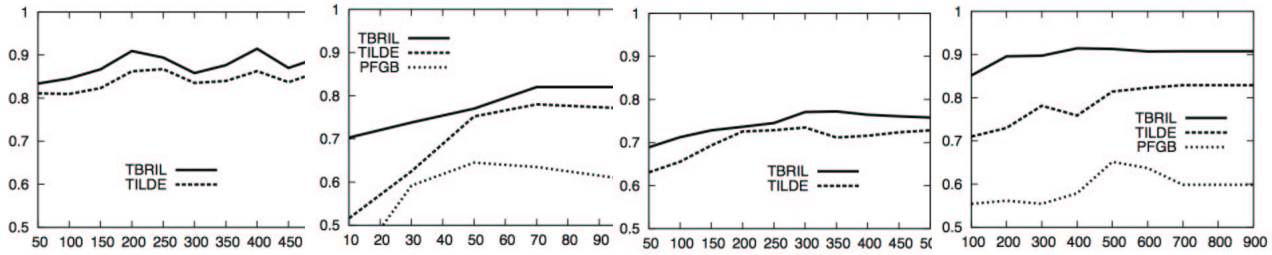

Figure 1: AUC-PR vs. number of trajectories (a) Blocksworld (b) Gridworld (c) Traffic Signal (d) Robocup

To our knowledge this is the first adaptation of a Statistical Relational technique to the problem of learning relational policies from an expert. In this sense, our work can be understood as making a contribution in both ways: applying gradient boosting for relational imitation learning and identifying one more potential application of SRL.

Acknowledgements: SN acknowledges the support of Translational Sciences Institute, Wake Forest University School of Medicine. SJ is funded by the CI Fellowship (2010), under NSF subaward No. CIF-B-211. PT gratefully acknowledges the support of NSF under grant IIS-0964705. KK is supported by Fraunhofer ATTRACT fellowship STREAM and by the EC under contract number FP7-248258-First-MM. JS acknowledges the support of Defense Advanced Research Projects Agency under DARPA grant FA8650-06-C-7606.

References

[1] H. Blockeel. Top-down induction of first order logical decision trees. *AI Commun.*, 12(1-2), 1999.

[2] T.G. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *ICML*, 2004.

[3] J.H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 2001.

[4] B. Gutmann and K. Kersting. TildeCRF: Conditional random fields for logical sequences. In *ECML*, 2006.

[5] K. Kersting and K. Driessens. Non–parametric policy gradients: A unified treatment of propositional and relational domains. In *ICML*, 2008.

[6] S. Natarajan, T. Khot, K. Kersting, B. Guttmann, and J. Shavlik. Boosting Relational Dependency networks. In *ILP*, 2010.

[7] N. Ratliff, D. Silver, and A. Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, pages 25–53, 2009.

Topic: Learning algorithms
Poster/Presentation