Learning in Non-Stationary MDPs as Transfer Learning (Extended Abstract)

M. M. Hassan Mahmud
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB
hmahmud42@gmail.com

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB
s.ramamoorthy@ed.ac.uk

ABSTRACT

In this paper we introduce the MDP-with-agents model for addressing a particular sub-class of non-stationary environments where the learner is required to interact with other agents. The behavior-policies of the agents are determined by a latent variable that changes rarely, but can modify the agent policies drastically when it does change (like traffic conditions in a driving problem). This unpredictable change in the latent variable results in non-stationarity. We frame this problem as transfer learning in a particular sub-class of MDPs, which we call MDPs-with-agents (MDP-A), where each task/MDP requires the learner to learn to interact with opponent agents with fixed policies. Across the tasks, the state and action space remains the same (and is known) but the agent-policies change. We transfer information from previous tasks to quickly infer the combined agent behavior policy in a new task after some limited initial exploration, and hence rapidly learn an optimal/near-optimal policy. We propose a transfer algorithm which given a collection of source behavior policies, eliminates, using a novel statistical test, the policies that do not apply in the new task in time polynomial in the relevant parameters. We also perform experiments in three interesting domains and show that our algorithm significantly outperforms relevant alternative algorithms.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Intelligent Agents

General Terms

Algorithms

Keywords

Single agent Learning, Planning, Agent theories, Modelling other agents and self, Computational architectures for learning

1. THE MDP-A MODEL

In this paper we introduce the MDP-with-agents model for learning in non-stationary environments where the non-stationarity arise from the actions of K other agents whose behavior policy may change over time. The behavior-policies are determined by a latent variable τ that changes infrequently, but when it does, may drastically alter the policies of the agents. That is, each value of τ defines a regime, and the change in value causes a regime change. Our model captures domains that require interacting with robots or other agents who operate in accordance with distinct behavior-profiles. For instance, in a driving problem, the overall behavior of other drivers depend on the latent variable 'traffic conditions'. We use standard and novel statistical tests to quickly determine whether any of the previous profiles are being used in the current task, and if so use that fact to compute an optimal policy for the new task. We give sample complexity bounds for the algorithm to eliminate incorrect profiles when we use our tests and then demonstrate its efficacy in experiments. Full details appear in the long version of the paper [3], and below we describe our approach and results briefly.

We model the above problem as transfer learning for reinforcement learning (TLRL) in MDPs where each MDP represents a particular regime. In TLRL we try to use solutions of MDPs solved at a prior point in time (the *source* MDPs) to solve a new but related MDP (the *target* MDP) much faster (see [4],[5] for an introduction reinforcement learning in MDPs and [6] for a survey on TLRL). In our case we use agent behavior in source MDPs (previous regimes) to infer their behavior in the target MDP (current regime).

We assume that the reader is familiar with MDPs, MDP policies and value functions [5]. To model regimes, we introduce a sub-class of MDPs with the additional presence of K other agents with fixed policies called MDP-with-agents, (MDP-A in short form). Each MDP-A is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, L, \gamma, \mathcal{A}', \tau)$, where $\mathcal{A}'$ is a joint action space of K different agents and τ is the joint behavior profile/*type* of the K different agents. Each τ is a distribution over $\mathcal{A}'$ indexed by s, a – i.e. $\tau_{s,a}$ is a distribution over $\mathcal{A}'$. The transition distribution and reward function, respectively have the form $T(\cdot|s,a,a')$ and $L(s,a,a')$ where $s \in \mathcal{S}$, $a \in \mathcal{A}$, $a' \in \mathcal{A}'$. The joint-action a' is such that $a' \sim \tau_{s,a}(a')$. Since $\tau_{s,a}$ depends only on s and a, each MDP-A is indeed a MDP; we recover the standard MDP transition and reward functions as follows: $P(s'|s,a) = \sum_{a'} T(s'|s,a,a')\tau_{s,a}(a')$ and $R(s,a) = \sum_{a'} L(s,a,a')\tau_{s,a}(a')$. The class of MDPs we consider are identical with the exception of the type τ. Hence each possible value of τ implies an MDP and we associate with each τ the optimal policy of the corresponding MDP and denote it by π^*_τ.

When solving the target MDP-A, we assume that all the components defining it are known except for the type τ. Given τ, we would be able to compute the optimal policy immediately. In the transfer formulation of our problem, we are given as input a set of source types/MDPs τ_i, $1 \leq i \leq N$. If we can determine quickly that $\tau = \tau_i$, we can use $\pi^*_{\tau_i}$ to attain optimal performance rapidly. This is precisely the idea of our algorithm Type-Elimination, given in sketch form as Algorithm 1 (see [3] for the full algorithm). The main feature of Type-Elimination is the use of a statistical test $f(p,x)$ that given a distribution p and a sample x, returns the prob-

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.

ability that p could *not* have generated x. So given a set of opponent agent actions $smp(s,a)$ observed at state s after choosing action a, if $f(\tau_i, smp(s,a)) > \alpha$ we can eliminate τ_i as the true type with probability α. We introduce a novel test f based on the Hoeffding bounds, and is a 'inverting' of the typical way the bounds are used. Instead of determining the accuracy of the estimate of the mean given a sample, we measure how likely it is that a given distribution could have generated a sample with the observed mean.

Algorithm 1 (*Sketch*) Type-Elimination$(\{\tau_1, \tau_2, \cdots, \tau_N\}, s_0, f, \alpha)$

1: **Input:** Previous types $\tau_j, 1 \leq j \leq N$, initial state s_0, statistical test f and confidence level α.
2: **Initialize:** Candidate types $\mathsf{VT} = \{\tau_i : 1 \leq i \leq N\}$, $t = 0$, $\forall s, a\ smp(s,a) = \emptyset$.
3: **for** $t = 1$ to T and $|\mathsf{VT}| > 0$ **do**
4: **if** $\tau_{s_t,a}$ defined for some a [* hence an opportunity to possibly eliminate a type from VT *] **then**
5: Take action $a_t \triangleq \arg\max_a \min_{a',\tau_i \in \mathsf{VT}} [f(smp(s_t,a)a', \tau_i(s_t,a)) - f(smp(s_t,a), \tau_i(s_t,a))]$, where $smp(s_t,a)$ are the agent actions a' observed at s, a so far [* so in the worst case, a_t maximizes the increase in f, and moves us most toward eliminating a type in VT *].
6: **else**
7: Take action $a_t \triangleq \rho(s_t)$ where $\rho \triangleq \arg\max_{\pi^*_{\tau_i}} V^{\pi^*_{\tau_i}}(s_t)$ [* choose optimistic action given types in VT *].
8: **end if**
9: Observe s_{t+1}, r_{t+1}, a'_t; update transition and reward estimates $\hat{P}(.|s_t,a_t)$ and $\hat{R}(s_t,a_t)$, update $smp(s_t,a_t) \leftarrow smp(s_t,a_t) \cup \{a'_t\}$.
10: Remove τ_i from VT if $f(\tau_i, smp(s_t,a_t)) > \alpha$.
11: **end for**
12: **if** $t < T$ [* true type was not in VT, *] **then**
13: Run R-Max for T-t steps, initialized with $\hat{P}$ and $\hat{R}$.
14: **end if**

2. EXPERIMENTS: NICHE FACTORY

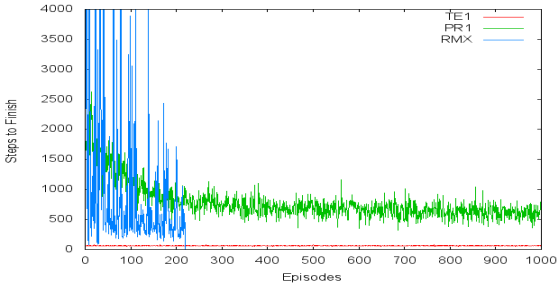

Figure 1: Results for the niche factory domain. Policy Reuse (PR1) eventually converges (not shown) but Type-Elimination (TE1) completely dominates both R-Max (RMX) and PR1.

From a transfer learning perspective, our algorithm tries to efficiently reuse the source policies $\pi^*_{\tau_i}$ and so solves the same problem as the Policy-Reuse algorithm [2]. Hence, in our experiments, we compare our method to this algorithm. We performed a set of experiments on 3 domains and due to space reasons we only report results from one, the *niche factory* domain (the results from others are comparable). Here the goal of the learner is to effectively compete with other agents for shared resources in a factory floor that tries to address ever changing demands of customers in niche markets [1]. The learner and other agents are each responsible for assembling different products, aimed at different niche markets. The customer demands/market conditions change unpredictably, which translates to an unpredictable change in demand on resources. Figure 1 presents the result for a particular target task (market condition) given a set of source task (previously encountered tasks (market conditions), averaged over 10 trials. The result are representative of other target tasks. We compared Policy Reuse (PR1) with Type-Elimination (TE1) and R-Max (RMX) as an optimal pure RL baseline. Our algorithm is able to quickly identify the true type and starts performing optimally after only a few episodes.

3. RELATED WORK AND CONCLUSION

Several algorithms for dealing with non-stationary MDPs have been developed (for instance [7] – see long version [3] for full discussion, including other algorithms). The main difference between our method and these previous methods is that we consider a very specific type of non-stationarity, which we believe to be useful in practice, and obtain an efficient algorithm for that scenario, whereas they consider the problem of non-stationary MDPs in full generality. Correspondingly, as we discuss in the long version of our paper, they have fundamentally weaker performance guarantees for our setting. A similar relationship holds between MDP-A and more general models of handling multi-agent systems like Markov games and I-POMDPs. In terms of transfer learning methods, Policy Reuse is the only one that is most directly related to ours. For future work, our goal is to relax the assumptions of known transition and reward functions, allow types to change between and within episodes, complete the picture by introducing a method to acquire the types in a continual 'lifelong' learning setting and do more experiments in interesting domains. We believe all of the above can be done using the tools we have already developed.

4. ACKNOWLEDGEMENTS

We undertook the work in the Robust Autonomy and Decisions group, School of Informatics, University of Edinburgh. The group is supported in part by grants from the UK Engineering and Physical Sciences Research Council (EP/H012338/1), the European Commission (TOMSY Grant 270436, FP7-ICT-2009.2.1 Call 6).

5. REFERENCES

[1] A third industrial revolution. *Economist*, April 21^{st}, 2012.
[2] F. Fernandez, J. Garcia, and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, 206.
[3] M. M. H. Mahmud and S. Ramamoorthy. Learning in non-stationary mdps as transfer learning. Technical report, University of Edinburgh, 2013.
[4] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.
[5] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
[6] M. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.
[7] J. Y. Yu and S. Mannor. Arbitrarily modulated markov decision processes. In *Proceedings of the IEEE Conference on Decision and Control*, 2009.