Non-disjoint modularity in reinforcement learning through boosted policies

Kurt Driessens KURT.DRIESSENS@CS.KULEUVEN.BE
Katholieke Universiteit Leuven, Belgium

Real world applications of reinforcement learning techniques are often limited by the monolithic results reinforcement learning agents produce. Small changes in the environment of the learning agent, due to deployment at a different spatial location, an evolving world or additional behavioral requirements cause the learned results to become suboptimal or, unfortunately more often than not, entirely unusable. It is therefore not surprising that the production and reuse of modular knowledge in reinforcement learning is an important topic for research, as is evident from the high number of reinforcement learning researchers that are involved or interested in inductive transfer. Inductive transfer or transfer learning is meant to enable the reuse of automatically learned results in different but related domains, either by making them directly applicable (possibly producing non-optimal but still high quality behavior) or as a starting point for further, more focussed and productive, learning.

The production and reuse of modular knowledge through and in reinforcement learning has, so far, been limited mostly to the generation of partial policies used either in the form of temporally extended actions or in a temporally or spatially disjointed decomposition of the learning task. Often the choice for this kind of decomposition had its roots in history, as the used techniques were initially developed to counter the curse of dimensionality and enable reinforcement learning agents to deal with larger problem domains.

In this abstract we will promote the production and use of a different type of modular knowledge that allows the decomposition of concurrent or overlapping subtasks or goals. Imagine an acrobat-waiter that delivers drinks while on stilts. This agent has to combine balancing his weight on the stilts and navigating between customers and the bar. He can however not interleave these two subtasks but has to choose his navigational actions in consideration to the balancing aspect of his task. Another example is that of user-adaptable software-assistants. We would like to be able to learn a user-preference module that can be shared between software assistants that exert subtle influences on how these assistants perform their tasks.

1. Modular Knowledge in Reinforcement Learning

Modularity in reinforcement learning has been studied mainly through hierarchical approaches (Barto & Mahadevan, 2003; Dietterich, 2000). Using goal-driven spatial and temporal divisions of the state-action space, control over the actions of the learning agent is passed to one of its hierarchically ordered control structures. Even in the 'options' framework (Sutton et al., 1999) where an agent can choose to perform a temporally extended action or option, options can terminate before their actual goal is reached, but only one aspect of the task decomposition of the agent influences its action choice at any given time-step.

Two exceptions to these approaches, which do not require disjoint policies, are the work by Torrey et al. (2005) and Driessens and Blockeel (2001) . These approaches allow different aspects of the task to be represented by separate Q-functions. Because policies based on the combination of Q-values are often hard to predict, Torrey et al. use the learned Q-values and combinations of them (e.g. the sum) in advice available to a new Q-learning agent. Driessens and Blockeel on the other hand add the learned Q-functions to the state representation of a new Q-learner and allow it to use the extra information as features to describe its Q-function.

While these approaches move further into the direction of the non-disjoint modularity we are interested in, the modules or policies are reused only in an indirect way. While this may be a safe option, we believe it is possible to do better through boosted policies.

2. Non-parametric boosted policies

Policy definition through a so-called potential function is nothing new in reinforcement learning. Often, a Q-

function is to serve as the potential function, but even in the more general case of a function $\psi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ that expresses preference of state-action pairs

$$\pi(s,a) : \mathcal{S} \times \mathcal{A} \rightarrow [0,1] = \frac{e^{\psi(s,a)}}{\sum_b e^{\psi(s,b)}} \qquad (1)$$

presents a probability distribution that can represents a stochastic policy for a learning agent.

Traditionally, these potential functions are expressed through a parameter vector ϕ and optimized using a gradient search mechanism. Recently, a non-parametric policy gradient approach was proposed that eliminates the need for a fixed parameterization. It employs samples of the functional gradient of the current policy to learn an approximate gradient update over the entire state-action space. These updates are combined in an approach very similar to boosting, i.e. $\psi(s,a) = \psi_1(s,a) + \psi_2(s,a) + \ldots \psi_n(s,a)$ where each ψ_i is a learned approximate gradient update and ψ is used to generate the agent's policy as described above. This enables the use of any standard regression algorithm and means that different gradient updates can use different machine learning algorithms and even the different representational formats they require.

3. Modular Policies

The inherent incremental structure of gradient policy updates and the possible representational distinctness of policy updates in non-parametric policy gradient learning give rise to an elegant modularity in the learned results. By separating the gradient updates relating to different parts of the agent's task, one can create separate potential functions each belonging to a separate part of the task. Because these "modular" functions are added together to give rise to a policy, the subtasks for which they serve are not required to be spatially or temporally disjoint.

Imagine a reinforcement learner operating in the (overused but often very illustrative) blocks world. One "module" or potential function could drive the agent to stack blocks and making the stacks as high as possible. When used alone, this would give rise to an agent that stacks all blocks into a single tower. A second module could drive an agent to combine blocks of similar colors. The combination of both modules could then give rise to an agent that builds stacks of blocks with similar colors. A third module could represent the fact that certain blocks give rise to stable combinations (e.g. small blocks on top of large blocks are stable, the other way around is not) and combinations of all three modules can give rise to an agent that builds stable, similar colored stacks.

Two challenges are apparent to get this approach to work. One is devising a way to learn the separate modules or to separate out the relevant parts of a learned policy, and the second is to recombine them into a policy that indeed spends the correct amount of attention to each of the modules. Early experiments seem to indicate that the approach is at least feasible for some domains. In the blocks domain discussed above, we have observed increased learning speeds when an agent confronted with an unstable blocks world is initialized with a previously learned stacking module from a stable world. The main advantage of policy gradient methods is that they are self-correcting and as such quite robust against aggressive initializations.

We are confident that further research into this direction will give rise to a well rounded approach to modular reinforcement learning based on non-parametric policy gradients.

Acknowledgments
Kurt Driessens is a post-doctoral fellow of the Fund for Scientific Research (FWO) of Flanders.

References

Barto, A., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete-Event Systems journal*, *13*, 41–77.

Dietterich, T. (2000). Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Driessens, K., & Blockeel, H. (2001). Learning digger using hierarchical reinforcement learning for concurrent goals. European Workshop on Reinforcement Learning, EWRL, Utrecht, the Netherlands, October 5-6, 2001.

Sutton, R., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learningx,. *Artificial Intelligence*, *112*, 181–211.

Torrey, L., Shavlik, J., Walker, T., & Maclin, R. (2006). Skill acquisition via transfer learning and advice taking. *Proceedings of the 17th European Conference on Machine Learning* (pp. 425–436).

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* (pp. 412–424).