Learning RoboCup-Keepaway with Kernels

Tobias Jung* Daniel Polani†

Abstract

We give another success story of using kernel-based methods to solve a difficult reinforcement learning problem, namely that of 3vs2 keepaway in RoboCup simulated soccer. Key challenges in keepaway are the high-dimensionality of the state space (rendering conventional grid-based function approximation like tilecoding infeasable) and the stochasticity due to noise and multiple learning agents needing to cooperate. We use approximate policy iteration with sparsified regularization networks to carry out policy evaluation. Preliminary results indicate that the learned behavior is even better than Stone et al. [5].

1 The Keepaway problem

RoboCup is a multi-agent domain with two opposing teams of agents struggling to beat the other one in a game of simulated soccer. Agents are autonomous entities; they sense and act independently and asynchronously, run as individual processes and cannot directly communicate. Each agent only has a restricted view of the world (hidden states). In addition, random noise affects both the agents sensors as well as their actuators (stochasticity). State description consists of relative distances and angles to visible objects in the world, such as the ball, other agents or fixed beacons for localization (high dimensionality). And finally the agents must make their decisions in real-time. These properties make RoboCup a more than worthy challenge for current reinforcement learning (RL).

Here, we consider a subtask of the full problem, namely the 3vs2 *keepaway* problem. In *keepaway* we have two smaller teams: one team (3 keepers) must try to maintain possession of the ball as long as possible while staying within a tight rectangular space. The other team (2 takers) tries to gain possession of the ball. Stone *et al.* [5] formulated keepaway as RL benchmark problem;[1] the keepers must *learn* how to maximize the time they control the ball (as a team!) against the team of opposing takers playing a fixed strategy. However, each keeper only learns *individually* from its own (noisy) actions and its own (noisy) perceptions of the world. The decision-making

*University of Mainz, Germany. E-mail: tjung@informatik.uni-mainz.de

†University of Hertfordshire, UK. E-mail: daniel.polani@herts.ac.uk

[1] sources available from `http://www.cs.utexas.edu/users/AustinVilla/sim/keepaway/`

happens at an intermediate level using multi-step macro-actions; the keeper currently controlling the ball must decide between holding the ball or passing it to one of its two teammates. The remaining keepers automatically try to position themselves such to best receive a pass. The immediate reward is the time that passes between individual calls to the acting agent. The task is episodic; it starts with the keepers controlling the ball and endures as long as neither the ball leaves the region nor the takers succeed in gaining control. The two central challenges to overcome is 1.) the high dimensionality of the state space (each sensation consists of 13 measurements, see Fig. 1) meaning that conventional approaches to function approximation, like grid-based tilecoding, will have a hard time. And 2.) the stochasticity due to noise and the uncertainty in control due to the multi-agent nature, meaning that the dynamics of the environment are both unknown and cannot be obtained easily. Hence we need model-free methods.
Stone *et al.* [5] successfully applied RL to *keepaway*, using the textbook approach with online Sarsa(λ) and tilecoding as underlying function approximator [6]. However, tilecoding is a local method and places parameters (i.e. basis functions) regularly throughout the entire state space, such that the number of parameters grows exponentially with the dimensionality. In [5] this very serious shortcoming was adressed by exploiting problem-specific knowledge of how the various state variables interact. In particular, each state variable was considered independently from the rest. Here in this short paper we will instead apply kernel-based methods which bypasses the exponential growth of parameters and represents the solution through the data. We demonstrate that we can learn using the full state information without toning down anything.

2 Our methodology: tools and algorithms

Approximate policy evaluation. In reinforcement learning the goal is to estimate the value function from observed transitions and rewards, which is then used to derive good decisions. Here we use the framework of approximate policy iteration, i.e. alternate between estimating the value function for the current policy π (policy-evaluation) and deriving a new policy from the approximation (policy-improvement). Other than in traditional function approximation the desired target values in policy-evaluation are not directly known but must be inferred from the information contained in the Bellman operator $\mathcal{T}_\pi$. Since we know that the true (unknown) value function is a fixed point under $\mathcal{T}_\pi$ two possibilities are [3]: (1) the Bellman residual

minimization approach. Here we want to make small the magnitude of the change when applying $\mathcal{T}_\pi$, i.e. we seek an approximation that minimizes (in a regularized least-squares sense) the Bellman residuals. However, this approach is really feasable only for deterministic environments, since else we would need 'doubled samples' [3]. The alternative choice we employ for *keepaway* is (2) the least-squares fixed point approximation. Here we want to make small the direction of the change when applying $\mathcal{T}_\pi$, i.e. we seek an approximation that is invariant under application of $\mathcal{T}_\pi$ followed by 'projection' (in a regularized least-squares sense) into the space of approximate value functions spanned by the currently selected basis functions.

Model-free learning with Q-values. Without explicit knowledge of the underlying system's dynamics we must estimate the augmented state-action values (i.e. Q-values) in order to be able to determine the best action given the state [3].

Regularization networks. We use a non-parametric kernel-based approach; the approximate Q-value function is represented as a sum of kernels centered on the data. To carry out the 'projection' during policy-evaluation we minimize a regularization functional that in addition to minimizing the squared approximation error also penalizes complexity. The resulting solution hence equals a regularization network (with suitable inputs and targets) and is effectively the mean prediction obtained by GP-regression (without predictive variance).

Subset of regressors and online selection. We use the subset of regressors method, e.g. see [4], to approximate the full kernel using a much reduced subset of basis functions. To select this subset we employ online greedy selection, similar to [1, 2], that adds a candidate basis function based on its distance to the span of the previously chosen ones. One improvement is that we consider a *supervised* criterion for the selection of the relevant basis functions that takes into account the reduction of the cost in the original learning task (here it is the related task of minimizing the Bellman-residuals) in addition to reducing the error incurred from approximating the kernel. Since the per-step complexity during training and prediction depends on the size of the subset, making sure that no unnecessary basis functions are selected ensures more efficient usage of otherwise scarce resources. This way learning in real-time (a necessity for *keepaway*) becomes possible.

3 Results and Discussion

Figure 1 shows the results we are able to achieve with our approach; due to the lack of space we can't describe the experiments in more detail. The results are preliminary but seem to indicate that a kernel-based approach is indeed superior to convential approaches like grid-based tilecoding, as far as high-dimensional learning tasks in RL are concerned. Also see the very comparable GPTD of Engel et al. in [2] and follow-up work.

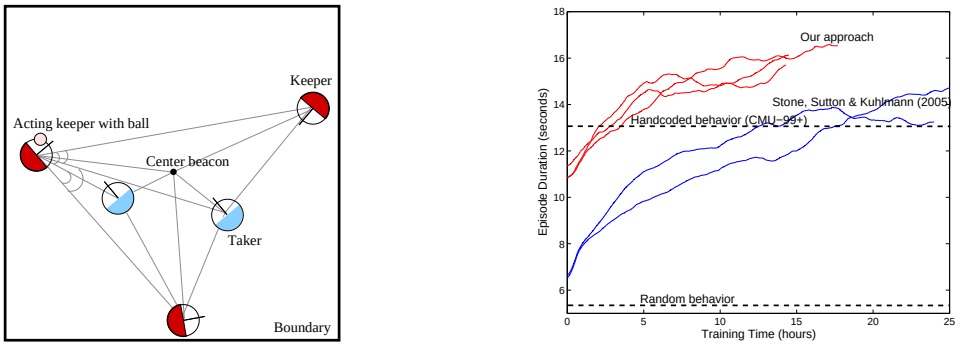

Figure 1: Left: Illustrating *keepaway*. The various lines and angles indicate the 13 state variables making up each sensation. Right: Learning curves for our approach. We plot training time (after interacting for 20 hours an agent experiences roughly 50,000 state transitions) against the average time the keepers are able to control the ball (quality of learned behavior).

References

[1] L. Csató and M. Opper. Sparse representation for Gaussian process models. In *Advances in NIPS 13*, pages 444–450, 2001.

[2] Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proc. of ICML 20*, pages 154–161, 2003.

[3] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *JMLR*, 4:1107–1149, 2003.

[4] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[5] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[6] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.