Relevance Vector Sampling for Reinforcement Learning in Continuous Action Space

Minwoo Lee
Computer Science Department
Colorado State University
Fort Collins, Colorado 80523
Email: lemin@cs.colostate.edu

Charles W. Anderson
Computer Science Department
Colorado State University
Fort Collins, Colorado 80523
Email: anderson@cs.colostate.edu

Abstract—**To be applicable to real world problems, much reinforcement learning (RL) research has focused on continuous state spaces with function approximations. Some problems also require continuous actions, but searching for good actions in a continuous action space is problematic. This paper suggests a novel relevance vector sampling approach to action search in an RL framework with relevance vector machines (RVM-RL). We hypothesize that each relevance vector (RV) is placed on the modes of the value approximation surface as the learning converges. From the hypothesis, we select actions in RVs to maximize the estimated state-action values. We report the efficiency of the proposed approach by controlling a simulated octopus arm with RV-sampled actions.**

I. Introduction

Reinforcement learning (RL) problems are often modeled as finite Markov decision processes (MDP) [25] with solutions that lie in discrete spaces. Real world problems, however, require multidimensional, even continuous state representation. Thus, the curse of dimensionality gets worse when continuous actions are needed. In the infinite state-action spaces, fine discretization can lead to successful control, but this can result in the need for a prohibitive amount of experience [22].

In practical reinforcement learning studies, function approximation estimates the state-action (Q) values to *generalize* based on limited experience to overcome the lack of experience. Parameterized Q function approximators have been successfully applied to various problems [5], [16], [2], [23], [24]. Some investigated transfer learning approaches to improve learning efficacy over the function approximations [18], [8], [28], [1]. Also, some studies have demonstrated convergence in practical applications involving continuous domains [19], [15], [13]. However, these approaches are still slow for practical applications and require numerous samples to learn convergent, near-optimal policies.

Several studies have shown that continuous actions allow the solution of problems that are impossible to solve with coarse discretization of action space [29], [10], [13]. The following studies considered alternatives to discretization. From a finite set of actions, some researchers obtain real-valued actions by interpolating discrete actions based on the value functions. Millan, et al., [20] sampled real-valued actions from neighbors incrementally based on the approximated value function. Hasselt, et al., [29] select actions that have the highest Q value from the interpolator. Lazaric, et al., [13] use Sequential Monte Carlo methods, which resample real continuous actions according to an importance sampling. Lee, et al., [15] use back-propagation over Greedy-GQ with neural networks to search for actions that maximize the Q estimation. However, these approaches require additional computations to search continuous actions.

Lee and Anderson [14] have shown that relevance vector machine (RVM) function approximation can be successfully adopted and the gradual knowledge augmentation improves the robustness of learning. However, the RVM-based learning framework for reinforcement learning (RVM-RL) with augmentation suffers from a growing search space when a problem requires continuous action control. The sparse nature of RVMs captures the significant masses of an agent's experience, and it is observed that relevance vectors tend to be located in the modes of the Q estimation curve. This happens especially when the Q estimation converges to the true Q values. Thus, we can reuse the actions in a relevance vector set since one of them is likely to be an optimal action.

The major contributions of this paper are the modification of RVM-RL and low-cost, continuous action sampling. To overcome the limitations of RVM-RL for continuous action domain problems, we replace the costly target function approximation shaping with significant (or relevant) sample storage for relevant experience replay. In addition, by reusing already discovered relevance vectors, our approach lowers the continuous action search cost to constant time. The approach proposed here to achieve the action sets from relevance vectors resembles importance sampling in sequential Monte Carlo learning [13], so its computation load is less intense than the direct line search as in [15]. Since RVM-RL stores the sparse relevance vectors from the learning process, there is no additional sampling cost required.

In Section II, we briefly summarize MDP and reinforcement learning and introduce RVM-based reinforcement learning framework. In Section III, we introduce our novel approach of continuous relevance vector action sampling with relevanct experience replay. Octopus arm control experiments and results of the proposed approach are summarized in Section IV, and we summarize our findings in Section V.

II. Background

A. Reinforcement Learning

A Markov decision process (MDP) [25] is a mathematical framework for modeling decision making problems. MDP is defined as a tuple $(S, A, P^a_{ss'}, R, \gamma)$, where for each time step $t = 0, 1, 2, \ldots$, with probability $P^a_{ss'}$, action $a_t \in A$ in state $s_t \in S$ transitions to state $s_{t+1} = s' \in S$, and the environment emits a reward $r_{t+1} \in R$.

Reinforcement learning involves interactions between an agent and an environment, and it can be modeled as a sequential decision problem, thus MDP. In an environment specified by the given MDP, a reinforcement learning agent aims to maximize the reward in the long run. For control problems, to estimate how good an action is in a given state, we can define the action value method for policy π, $Q^\pi(s, a)$, as expected sum of rewards:

$$Q^\pi(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_t = s, a_t = a, \pi]$$

where $\gamma \in (0, 1]$ is a discounting factor. Reinforcement learning looks for an optimal policy that maximizes Q^π, which can be denoted Q^*.

$$Q^*(s, a) = \mathbb{E}[r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s = s_t, a = a_t]$$

Without an environmental model, temporal difference (TD) learning learns directly from experience and bootstraps to update value function estimates—it updates the estimates based on the previously learned estimates.

For control problems, on-policy TD, SARSA [26], estimates $Q^\pi(s, a)$ for the current behavior policy π. The Q estimate for next state and action s_{t+1} and a_{t+1} is fed in for bootstrap update as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)].$$

Here, the action value function Q is for current behavior policy π. For simplicity, π superscript is omitted. Independently from the current behavior policy, off-policy TD, Q-learning [30], directly approximates Q^*. From $Q^*(s, a) = \max_{a'}(s, a')$, Q-learning updates is defined by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)].$$

From the estimated Q, the current best policy can be chosen greedily:

$$\pi^*(s) \leftarrow \arg\max_a Q^\pi(s, a).$$

However, greedy action selection can result in not enough samples collected for correct estimates of value function. In this paper, we use ϵ-greedy that selects a random action with probability ϵ and chooses a greedy action with probability $1-\epsilon$. By decreasing ϵ as learning goes on, an agent exploits the learned best actions more.

B. RVM-RL

Lee, et al., [14] suggested a knowledge augmentation framework with relevance vector machines (RVMs) [27] as Q function approximations. The reinforcement learning framework, RVM-RL, extended fitted-Q [21] and transferred learned experiences to next batch in a mini-batch training framework. The framework gradually augments the experience to learn efficiently and robustly.

The learning framework includes five steps: 1) collect samples, 2) train an RVM, 3) evaluate the regression training, 4) decide heuristics for shaping and transfer, 5) transfer learned knowledge for next batch learning. Through the repetitions of these steps, the target function approximator is adapted by adding the relevance vectors (RVs) and by updating the weights with stochastic gradient descent as below.

$$\begin{aligned} \texttt{RV}(\mathbf{RVM}_{target}) &= \texttt{RV}(\mathbf{RVM}_{target}) \cup \texttt{RV}(\mathbf{RVM}_{t+1}) \\ \texttt{W}(\mathbf{RVM}_{target}) &= (1 - c)\texttt{W}(\mathbf{RVM}_{target}) + c\texttt{W}(\mathbf{RVM}_{t+1}) \end{aligned}$$

Here, $\texttt{RV}(\cdot)$ retrieves the relevance vectors of RVM and $\texttt{W}(\cdot)$ retrieves the weights (mean estimates) of the RVM. c is a learning rate that determines how fast it reflects new knowledge.

In these series of learning steps, this approach resembles models of redundancy elimination (infomax principle) [17], [9] and the sequential building of encoded knowledge. Through RVM optimization, we apply the principle of efficient coding to represent knowledge. By transferring and updating the target function approximation, knowledge is accumulated to help find an optimal solution to a particular problem. That is, RVM-RL gradually increases its knowledge about the RL task from experience until it eventually reaches a good solution.

III. Methods

The main goal of relevance vector sampling is minimizing the computation costs for continuous action search. By using the sparse solutions of relevance vector machines, we can easily construct a continuous action set that has the best action with maximum Q value. To tackle the large search space problem with real-valued continuous actions, it is required to improve the RVM-RL as follows.

A. Relevant Experience Replay

RVM-RL can be slow to learn when the problem is extremely complex. When the number of input dimensions grow and the state and action spaces are continuous, RVM-RL spends much time on maintaining multiple RVMs and updating relevant parameters. Having only one RVM can reduce the required computation or transfer processes. However, a single RVM can be unstable with overfitting on each batch of training data. As a solution, we suggest a relevance vector storage for experience replay–we named this as *relevant experience replay*. Thus, the modified RVM-RL, after each batch, stores relevance vectors and replays them as a training sample that is combined with the new minibatch sample. From pilot tests, it is observed that the relevant experience replay reduces the

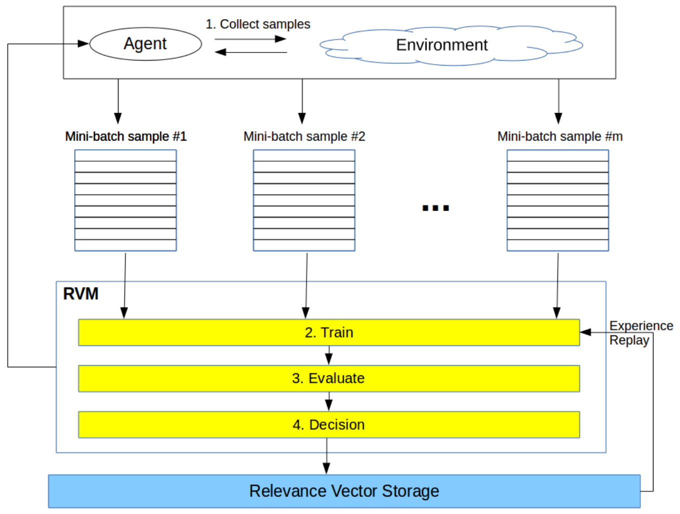


Fig. 1. The modified RVM-RL learning framework. Instead of maintaining multiple RVMs, it shapes a single RVM with relevant experience replay.

possibility of losing sparsity with augmentation. Fig. 1 depicts the modified RVM-RL with experience replay.

B. Relevance Vector (RV) Sampling

Lee, et al., [15] have demonstrated continuous action search with back-propagation in neural networks. The best action leads to the maximum estimated Q value on each step. At time t, with trained weights $\mathbf{v_t}$ and $\mathbf{w_t}$, the estimated optimal action $\mathbf{a_t}^*$ is determined by this one step search [22]:

$$\mathbf{a_t}^* = \operatorname*{argmax}_{\mathbf{a}} Q_{\mathbf{v_t},\mathbf{w_t}}(\mathbf{s_t}, \mathbf{a}),$$

where $\mathbf{v_t^{(a)}}$ are the weights applied to action input ($\mathbf{v} = [\mathbf{v^{(s)}}, \mathbf{v^{(a)}}]$). The best action $\mathbf{a}^*$ that maximizes $Q(\mathbf{s_t}, \mathbf{a})$ was found by using gradient ascent of $Q(\mathbf{s_t}, \mathbf{a})$ with respect to $\mathbf{a}$:

$$\frac{\partial Q(\mathbf{s_t}, \mathbf{a})}{\partial \mathbf{a}} = \mathbf{w_t}^\top \odot (1 - \mathbf{z}^2) \mathbf{v_t^{(a)}}^\top.$$

Here $\odot$ represents a component-wise multiplication, and $\mathbf{z}$ are the outputs from hidden units with $\tanh$ activation function.

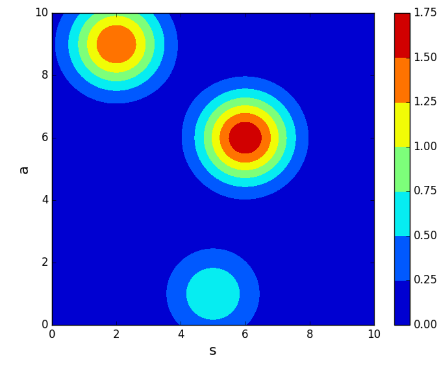
Similarly, using the derivative of the radial basis function kernel, the gradient for continuous action search over RVM function approximation can be defined:

$$\frac{\partial Q(\mathbf{s_t}, \mathbf{a})}{\partial \mathbf{a}} = -2\gamma(\mathbf{w_t} \odot \phi)^\top (\mathbf{a} - \mathbf{a}'). \tag{1}$$

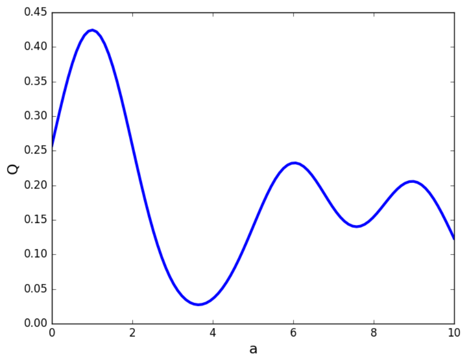
where $\mathbf{a}'$ is actions in a set of relevance vectors.

These gradient search approaches require additional computational costs for optimization. Furthermore, there is no guarantee for global maximum solution with the additional computation since the Q approximation can have multiple local maxima.

Fig. 2 shows that with multiple RVs, depending on the selection of kernel parameters, it is likely to have more than one local maximum, which makes it difficult to find a global maximum with a gradient descent approach. Along with the



(a) Q plots



(b) Example Q on 4

Fig. 2. Multi-modal Q function approximation with RVM-RL. As learning converges, relevance vectors are place the modes of Q function approximation. Restricting the search of continuous actions to relevance vectors lead the proposed RV sampling.

multimodality, it is observed that the placements of relevance vectors are located at the modes of the Q estimation surface.

Now, we suggest a way to use RVM as an action sampler based on the observation. With an assumption that the current Q estimation is valid, the action sets that are stored in relevance vectors lead local maxima. For greedy action selection, the RV action with highest Q value in the action set can be chosen. From state s_t, when current relevance vectors are stored in $X^{(RVM)} = [s^{(RVM)}, a^{(RVM)}]$, the candidate actions are $\tilde{a}^{(RVM)}$ after removing duplicate elements. From the candidates, we select action as follows:

$$a_t^* = \operatorname*{argmax}_{a \in \tilde{a}^{(RVM)}} Q_{\mathbf{v_t}, w_t}(s_t, a). \tag{2}$$

By using RVM-RL framework, the relevance vector sampling simplifies the continuous action set construction greatly. Additional computation is only required to remove the duplicate actions in relevance vectors that are taken in different states.

C. Kernel for Bases

For the given MDP model, RVM function approximation maps state and action inputs to the estimated Q values, ie. $Q: S \times A \rightarrow \mathbb{R}$ for state set S and action set A. Since the positive definite kernels are closed under multiplication, Engel,

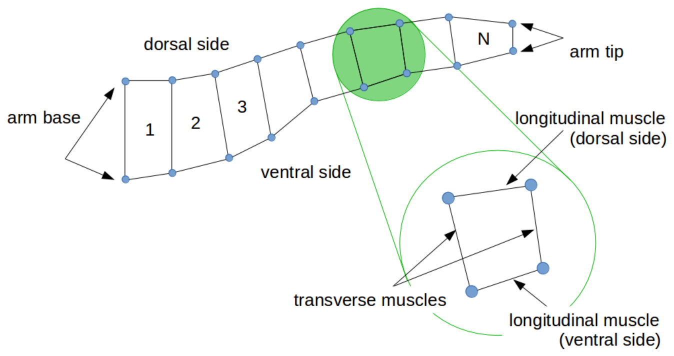

dorsal side
N
arm tip
arm base
1
2
3
ventral side
longitudinal muscle (dorsal side)
transverse muscles
longitudinal muscle (ventral side)

Fig. 3. 2-dimensional octopus arm model with N compartments [31]. Spring-like muscles, 2 longitudinal (dorsal and ventral) and 2 transverse, surround a constant area C, and $2N + 2$ masses connect the muscles.

et al., [6] separately compute the correlation between different state values and the correlation between different action values with the product kernel.

$$k(\mathbf{s}, \mathbf{a}, \mathbf{s}', \mathbf{a}') = k_s(\mathbf{s}, \mathbf{s}')k_a(\mathbf{a}, \mathbf{a}')$$

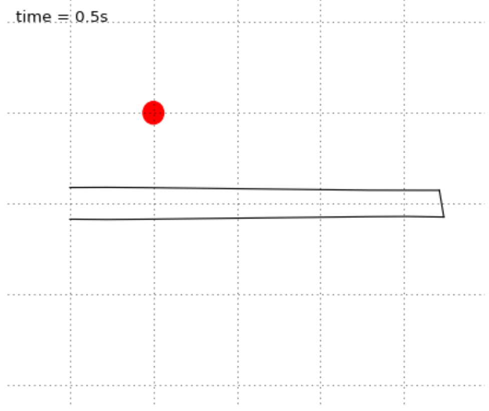
Interestingly, the kernel approach enables us to handle both parametric and nonparametric problems. In this paper, our target problem includes finding a general solution for different goals, so we include goal position in the input, ie. $Q : S \times A \times G \to \mathbb{R}$ for the goal position set G. For this, we extend the product kernel with an additional kernel for the goal inputs.

$$k(\mathbf{s}, \mathbf{a}, \mathbf{g}, \mathbf{s}', \mathbf{a}', \mathbf{g}') = k_s(\mathbf{s}, \mathbf{s}')k_a(\mathbf{a}, \mathbf{a}')k_g(\mathbf{g}, \mathbf{g}')$$

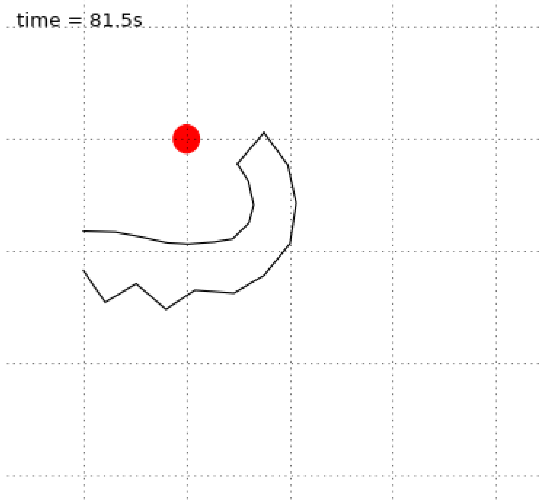
For the continuous action space, this product kernel can increase the search space. Thus, the proposed search of actions with relevance vectors can simplify the problem.

IV. Experiments

A real octopus arm is a complex organ with many degrees of freedom. Here, the Yekutieli's two-dimensional model [31] is used (Fig. 3). The model is composed of spring muscles for each compartment. The basis of the model is that muscular hydrostats maintain a constant volume [11], so forces are transferred among the segments. Gravity, buoyancy, fluid friction, internal particle repulsion and pressure, and muscle contractions are computed each time step. For our experiments, we implemented a Python version of the octopus arm model defined in the RL-competition [3] (Fig. 4), which simulates the simplified physics in Yekutieli, et al., [31]. The problem contains 10 compartments, and the ventral, dorsal, and transverse muscles are controlled by independent activation variables. For the experiments, we fix the base and do not consider rotation about the base. To make the problem simple, Engel, et al., [7], used 6 predefined discrete actions. Here we do not predefine actions. Without reducing the action space manually, we train the arm to learn from the full action space defined by 33-dimensional real-valued actions. For the 10-compartment



(a) Initial State



(b) Rereaching the goal

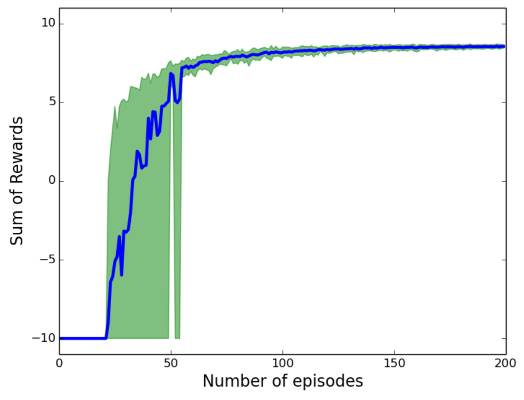
Fig. 4. Octopus arm control task (10 compartments)

example, the state space is defined by 82 continuous values: x-y coordinate positions of each joint, and their velocities.

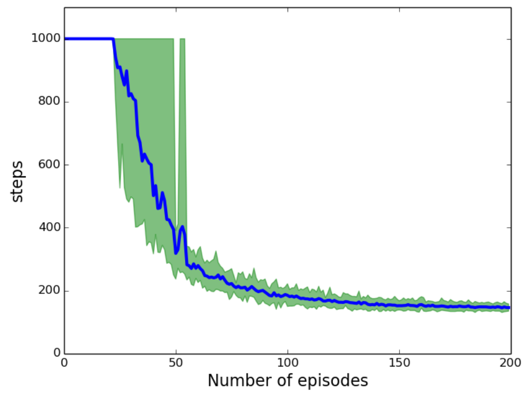
We place the goal at (4, 2) as in Fig. 4. Initially the base of the arm is placed close to (0, 0) and the arm is straight toward the right. The target task is touching the goal with any part of the arm. On each time step, the arm receives -0.01 as a penalty, and if it reaches the goal, it gets 10 as a reward. The maximum number of steps per each episode is limited to 1000 steps. Thus, if the arm touches the goal at the last moment, the total reward will be 0. If it fails to reach the goal, the total will be -10. Positive, larger rewards are obtained when the goal is reached sooner. The episodes are repeated 200 times with exponentially decreasing ϵ value from 1 to 0.01. Each experiment was run 10 times.

Fig. 5 shows the learning performance with RV-based continuous action selection in RVM-RL. For the test, we use the kernel parameter $\gamma_k = 10.0$ and the learning rate $c = 0.6$. The maximum number of RVM iterations was set to 100 and tolerance threshold was set to 1×10^{-5}.

In the beginning, since the RVM-RL agent starts learning with no RV basis, the agent explores the world randomly or



(a) Initial State



(b) Approaching the goal

Fig. 5. Successful learning with RVM-based continuous actions. Blue line represents mean steps and rewards over 10 experiments, and green region shows the min and max values.

revisits one of a few actions in a RV set when ϵ decreases. This pure exploration stage does not have any success record, but it accumulates the RV samples with continuous actions. After about 25 episodes, as the agent applies RV actions more often, instability of learning happens because of poor Q approximation. As learning continues, after 50 episodes of experience the transient curve quickly converges in both reward and step curves. Eventually, it finds a good solution to reach the goal in 137 steps. These results are comparable to our previous continuous action search with neural networks [15] but with very low cost for search. With the similar performance, RV sampling benefits from its sparsity again. By evaluating the small number of actions, it can quickly find a greedy action while neural network back-propagation search spends more time with gradient descent updates. Furthermore, the RV sampling has room for improvement by adopting efficient exploration strategies such as importance sampling [13] and Bayesian exploration-exploitation control [4], [12].

Fig. 6 depicts the continuous actions in the chosen relevance vectors after training. After training, 21 RVs are achieved, and they repeat the two actions, one full contraction on

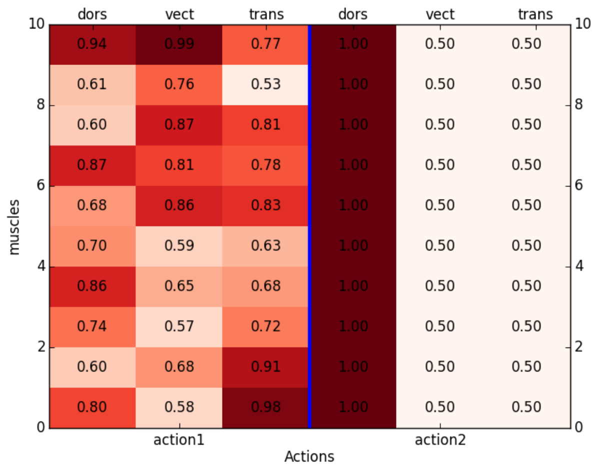


Fig. 6. Two core continuous actions found in 21 RVs after training. As the annotated number in each box represents the contraction force. 1 means full contraction force and 0 means releasing action without any force on a muscle.

dorsal muscles (action2) and a more complex s-shaped muscle contraction (action2). These two actions are alternated to curl the octopus arm to reach the goal. We can observe that sparse (only two) action options are left. This supports the argument about the benefits of RV sampling over slower neural network gradient descent action search. The solutions are sparser, so it is faster to evaluate the actions. However, we need to be careful about extremely sparse solutions that are likely to miss important samples, which can result in a poor policy. We expect this problem could be solved with better exploration strategies.

Now, we generalize the problem to have multiple goals. In this experiment, we change the goal positions to (4, 2) and (4, -2) every the other episodes. Alternating the opposite goal locations can disturb what is learned. Especially when it is overfitted to one goal, this problem ends up with oscillating performance. To simplify this problem, we adopt transfer learning. We train for two goals separately, one with (4, -2) and the other with (4, 2). After finding near-optimal policies from two tasks, an agent learns two curling actions toward the different goal directions. After that, we transfer the learned relevance vectors to tackle the changing goal problem. In this problem, we added goal position in the state input along with state and action. Fig. 7 shows the successful learning curve from 20 experiments. With the transferred relevance vectors, after 20 episodes of oscillation with random exploration (large ϵ), it quickly discovers good continuous actions and a near-optimal policy that reaches the goals quickly. For this experiment, we used the same parameters that we used for the previous single goal experiment.

Videos showing the arm controlling at different learning stages are available at www.cs.colostate.edu/~lemin/octopus.php.

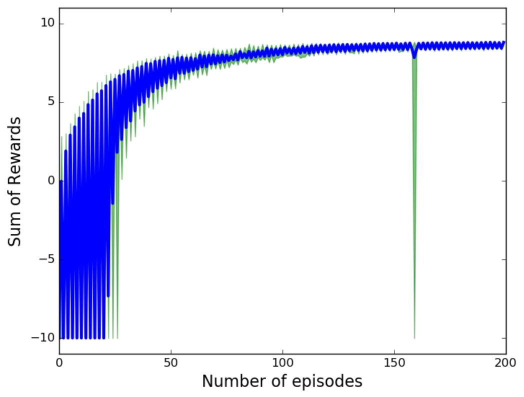


Fig. 7. Successful transfer learning from two separate tasks to a moving goal task on each episode. In the beginning, it oscillates without noticing the changes of the goal, but as it proceeds, it discovers a good policy that can handle both goals. Blue line represents the average over rewards over 20 experiments, and green region represents the minimum and maximum values.

V. Conclusion

In this paper, we proposed a sparse Bayesian reinforcement learning algorithm with novel relevance vector sampling. Lee, et al.'s, [14], RVM-RL framework has been improved to handle problems with large search spaces by using experience replay with relevance vector samples. The proposed approaches are successfully applied to the high dimensional, continuous octopus arm control problem, even with alternating goals.

The following steps will be taken to further improve the approach described here. Since the proposed approach assumes the eventual convergence of Q approximation, it can guarantee fast or stable learning only when it reaches the near-optimal point. Thus, we can combine (1) and (2) by using the RV actions as starting position for gradient search for improved learning performance in early stage. Instead of random exploration, however, if we sample actions efficiently based on the current RVs and Q estimation, RVM-RL is expected to place RVs on the peaks of new Q estimation and the correct action with the highest Q value will be selected with greedy strategy.

References

[1] C. W. Anderson, M. Lee, and D. L. Elliott, "Faster reinforcement learning after pretraining deep networks to predict state dynamics," in *International Joint Conference on Neural Networks (IJCNN)*, 2015.

[2] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems*, vol. 22, no. 3, pp. 283–302, 1997.

[3] R. Community. (2009) Rl-competition. http://www.rl-competition.org/. [Online; accessed 06-July-2016]. [Online]. Available: http://www.rl-competition.org/

[4] R. Dearden, N. Friedman, and S. Russell, "Bayesian q-learning," in *Proceedings of the National Conference on Artificial Intelligence*, 1998, pp. 761–768.

[5] M. Dorigo and M. Colombetti, "Robot shaping: Developing autonomous agents through learning," *Artificial Intelligence*, vol. 71, no. 2, pp. 321–370, 1994.

[6] Y. Engel, S. Mannor, and R. Meir, "Bayesian reinforcement learning with gaussian process temporal difference methods," 2007.

[7] Y. Engel, P. Szabo, and D. Volkinshtein, "Learning to control an octopus arm with gaussian process temporal difference methods," *Advances in Neural Information Processing Systems*, vol. 18, p. 347, 2006.

[8] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proceedings of the fifth International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM, 2006, pp. 720–727.

[9] K. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[10] C. Gaskett, D. Wettergreen, and A. Zelinsky, "Q-learning in continuous state and action spaces," in *Australasian Joint Conference on Artificial Intelligence*, 1999, pp. 417–428.

[11] W. M. Kier and K. K. Smith, "Tongues, tentacles and trunks: the biomechanics of movement in muscular-hydrostats," *Zoological Journal of the Linnean Society*, vol. 83, no. 4, pp. 307–324, 1985.

[12] J. Z. Kolter and A. Y. Ng, "Near-bayesian exploration in polynomial time," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 513–520.

[13] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement learning in continuous action spaces through sequential monte carlo methods," *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, vol. 20, pp. 833–840, 2008.

[14] M. Lee and C. W. Andersno, "Robust reinforcement learning with relevance vector machines," in *Proceedings of the 1st international workshop on robot learning and planning (RLP)*, 2016.

[15] M. Lee and C. W. Anderson, "Convergent reinforcement learning control with neural networks and continuous action search," in *Proceedings of IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2014.

[16] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[17] R. Linsker, "Perceptual neural organization: some approaches based on network models and information theory," *Annual review of Neuroscience*, vol. 13, no. 1, pp. 257–281, 1990.

[18] M. G. Madden and T. Howley, "Transfer of experience between reinforcement learning environments with progressive difficulty," *Artificial Intelligence Review*, vol. 21, no. 3, pp. 375–398, 2004.

[19] H. R. Maei, C. Szepesvári, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton, "Convergent temporal-difference learning with arbitrary smooth function approximation." in *NIPS*, 2009, pp. 1204–1212.

[20] J. D. R. Millán, D. Posenato, and E. Dedieu, "Continuous-action q-learning," *Machine Learning*, vol. 49, no. 2-3, pp. 247–265, 2002.

[21] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *Machine Learning: ECML 2005*, 2005, pp. 317–328.

[22] J. C. Santamaria, R. S. Sutton, and A. Ram, "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adaptive behavior*, vol. 6, no. 2, p. 163, 1997.

[23] P. Stone and R. S. Sutton, "Scaling reinforcement learning toward robocup soccer," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 1, 2001, pp. 537–544.

[24] P. Stone, R. S. Sutton, and G. Kuhlmann, "Reinforcement learning for robocup soccer keepaway," *Adaptive Behavior*, vol. 13, no. 3, pp. 165–188, 2005.

[25] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 1998.

[27] M. E. Tipping and A. Faul, "Fast marginal likelihood maximisation for sparse bayesian models," in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, vol. 1, no. 3, 2003.

[28] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," in *Machine Learning: ECML 2005*, 2005, pp. 412–424.

[29] H. Van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *Approximate Dynamic Programming and Reinforcement Learning, 2007. IEEE International Symposium on*, 2007, pp. 272–279.

[30] C. J. C. H. Watkins, "Learning from delayed rewards." Ph.D. dissertation, University of Cambridge, 1989.

[31] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov, Y. Engel, B. Hochner, and T. Flash, "Dynamic model of the octopus arm. i. biomechanics of the octopus reaching movement," *Journal of neurophysiology*, vol. 94, no. 2, pp. 1443–1458, 2005.