Can A Reinforcement Learning Agent Practice Before It Starts Learning?

Minwoo Lee* and Charles W. Anderson†
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523-1873
Email: *lemin@cs.colostate.edu, †anderson@cs.colostate.edu

***Abstract*—A reinforcement learning (RL) agent needs a fair amount of experience to find a near-optimal policy. Transfer learning has been investigated as a means to reduce the amount of experience required. Transfer learning, however, requires another similar reinforcement learning task as a transfer source, which can also be costly in the amount of experience required. In this research, we examine the possible "practice" approach that transfers knowledge from a non-RL task to a target RL task to avoid the expensive data sampling. We analyze how practice captures the distributions of state and action spaces in an environment. For this, we develop a novel learning approach that acquires important samples from practice and then applies them to a target RL task without changing learned bases. Results show an improved learning efficiency through practice in classical benchmark problems and limitations in OpenAI Gym problems.**

I. Introduction

Natural intelligence is developed from acquired experience and adaptation to new environments, resulting in the learning of new knowledge. The repetition of this cycle constructs new knowledge and sharpens acumen. Machine learning studies have tried to mimic this behavior and have shown successful transfer of knowledge in various domains. The benefits from transfer learning in supervised learning have been shown in textual data classification [22], natural language processing [4], image classification [34], WIFI localization [35], spam filtering and intrusion detection [8].

Transfer learning can boost reinforcement learning (RL) in many tasks, such as video games [19], robot soccer [28], and even the complex game of Go [25]. In reinforcement learning, since the samples for training contain an agent's behavior, schemes in addition to simple parameter value transfer can be adopted, such as smooth landing imitation [17], [6] and advising [32], [31]. However, these approaches are limited to knowledge exchange between two reinforcement learning tasks. When the target task or the goal is not unveiled, it is impossible to collect knowledge for transfer.

This study is motivated by Anderson, et al., [1] that have shown that pretraining neural networks for state transition prediction resulted in successful and more efficient training of a reinforcement learning agent. By using simple weight transfer, they examined how the transfer of learned weight values contributes to the subsequent reinforcement learning problem once the reinforcement signal is introduced. However, with weights from pretrained networks, it is hard to understand or explain how pretraining helps learning the successive tasks.

To better understand the benefits of practice, we focus on recent studies with relevance vector machines (RVM) for reinforcement learning tasks. Their additional analytical power through a Bayesian interpretation can help us observe how practiced knowledge assists successive learning. Lee and Anderson [14] showed that function approximation with relevance vector machines (RVM) can retrieve key experiences from learning and the gradual knowledge augmentation improves the robustness of learning. Also, they further improved the framework with efficient continuous action sampling [13]. Rexakis and Lagoudakis [23] successfully use the sparsity of an RVM classifier on their directed rollout classification policy iteration (DRCPI-RVM) framework. Because of the sparse nature of RVMs that capture the significant mass of an agent's experience, an RVM-based reinforcement learning framework (RVM-RL) [14] can help us understand how state transition prediction in Anderson, et al., [1] improves the performance of reinforcement learning. For this, we use a modified version of RVM-RL as a learning framework to collect the experiences in practice to explain what knowledge an agent obtains from practice and how it can be applied as a feature space for reinforcement learning.

A major contribution of our work is examining the effect of practice by an RL agent before starting to learn the solution to an RL problem. As humans practice to develop faster reactions or better performance in real situations, practice in reinforcement learning even without any goal-directives is expected to improve subsequent learning with reinforcement signals. Additional experience with knowledge construction from practice, in the form of sets of relevance vectors (RVs), turns out to be the key reason that makes this work. We demonstrate how practiced knowledge contributes to reinforcement learning by fixing the RV bases with the suggested framework. We will discuss this in detail in the following sections.

In Section II, we briefly summarize pretraining deep networks and RVM-based RL framework, and in Section III we introduce our novel approach for practice and fixed-basis training. Benchmark experiments and results of the proposed approach are summarized in Section IV, and we discuss how practice helps reinforcement learning and possible improvement in Section V.

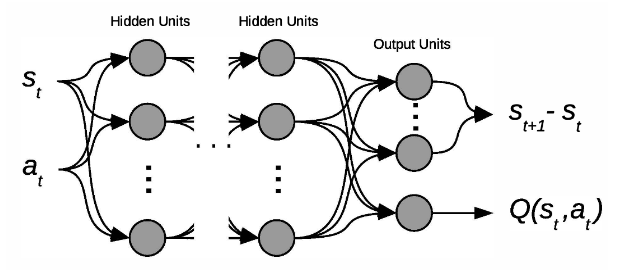


Fig. 1. Neural networks for state change prediction and Q estimation. First, neural networks are trained to predict state changes. Then the role of the final layer is changed from predicting state change to predicting future reinforcement as the value of a Q function.

II. Background

A. Pretraining Deep Networks for Reinforcement Learning

Anderson, et al., [1], demonstrated how learning the dynamics of an environment can facilitate the learning of a Q function in multilayered neural networks. For complex problems in practice, pretraining neural networks can greatly reduce the number of interactions with an environment that are required to achieve good performance.

Fig. 1 explains how network weights are pretrained and transferred for learning a Q function. To *pretrain* the neural networks, state transition samples of state s_t, action a_t, and next state s_{t+1} are collected. From the samples, s_t and a_t are used as input and $s_{t+1} - s_t$ forms the target output to fit. The scaled conjugate gradient (SCG) algorithm [20] was used to train the network to minimize the mean squared error in the outputs. For this pretraining stage, any reinforcement learning related information such as rewards, goals or objectives is not provided.

After pretraining the networks, a reinforcement learning agent collects mini-batches of s_t, a_t, reward r_{t+1}, s_{t+1}, and a_{t+1} samples. With these mini-batches, the pretrained neural networks are further trained to estimate Q values by minimizing the Bellman error with SARSA [27] update. Again, the SCG algorithm is applied. To lessen the chance of overfitting with each mini-batch, the number of iterations of the SCG algorithm is limited to a small number.

B. RVM-RL

To apply relevance vector machines (RVMs) [30] to reinforcement learning, Lee, et al., [14] extended a mini-batch training framework, fitted-Q [24], to transfer learned RVs in each step as illustrated in Fig. 2. The framework gradually augments the experience to learn efficiently. From a collection of mini-batch samples, an RVM is trained. After evaluating the trained RVM according to its error in reinforcement prediction, a heuristic decision is made to transfer the learned RVs to the next mini-batch iteration or to discard the learned RVs. When RVs pass this evaluation test, the target function approximator is adapted by adding the RVs and by updating the weights with stochastic gradient descent.

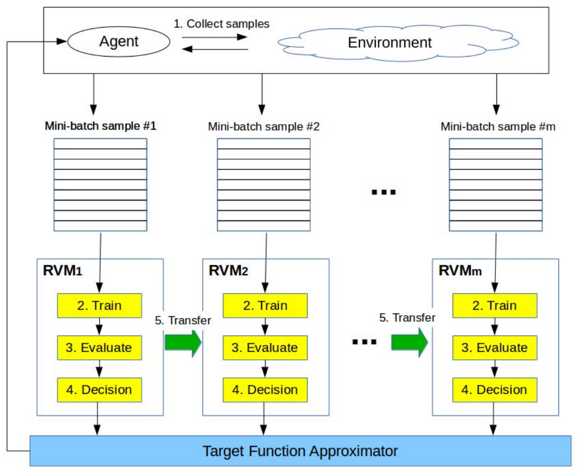


Fig. 2. RVM-RL learning framework. Based on the sparse solution from an RVM, knowledge is gradually augmented and heuristics maintain the sparsity after learning. The sparse solutions provide additional interpretation about the learned policy.

In these series of learning steps, this approach resembles models of redundancy elimination (infomax principle) [16], [7] 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. This knowledge accumulation has been tested in continuous state benchmark problems. In the next section, an approach is presented to quickly discover the core information to hasten the learning process, which can lessen the required sampling time.

III. RVM Practice for Reinforcement Learning

Here, we replace the term *pretraining* with the more general term *practice*. We focus on discovering knowledge that summarizes the dynamics of an environment. We adopt the relevance vector machines to discover such knowledge. We develop fixed-basis RVMs to examine the obtained knowledge efficacy for efficient reinforcement learning. For this, we first summarize the slight modifications to the RVM-RL framework with a fixed-basis and describe how to incorporate practice in the learning of an agent and how to apply this to actual problems.

A. Fixed-basis RVM (fRVM)

For efficient learning, the fast marginal likelihood maximization algorithm for RVMs [30] adds or removes bases, as defined by the set of RVs, to find the best fit that maximizes the log-likelihood. However, when we already know the best bases or alternative ones, it is not necessary to go through the RV addition or removal process. Assuming that $\mathbf{X}^{\mathbf{RVM}}$

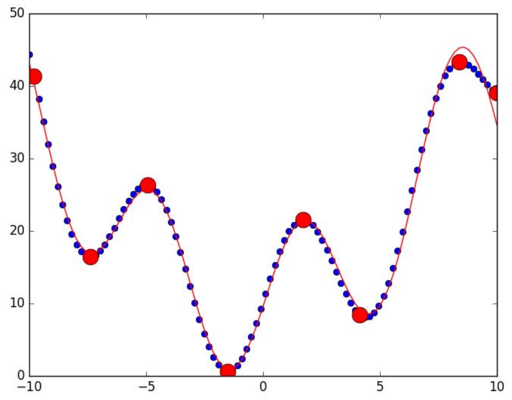


Fig. 3. fRVM with preset RVs (red dots). Blue dots represent the training samples and red line shows the prediction curve fit.

are relevance vectors, we can define the feature vector $\boldsymbol{\phi}$ with kernel $k(\cdot)$ as follows:

$$\boldsymbol{\phi}(\mathbf{x}) = k(\mathbf{x}, \mathbf{X}^{\mathbf{RVM}}).$$

Now, we can alternatively compute the following weight mean and covariance and prior distributions without modifying RV bases. First, compute the mean and covariance of the weights. Here, $\boldsymbol{\Phi}$, representing $\boldsymbol{\phi}(\mathbf{x})$, is the similarity of x to the preset relevance vectors:

$$\boldsymbol{\mu} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \mathbf{t}$$
$$\boldsymbol{\Sigma} = (\beta \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\alpha} \mathbf{I})^{-1}.$$

From the weight estimation, the hyper-parameters are computed:

$$\begin{aligned} \gamma_i &= 1 - \alpha_i \Sigma_{ii}, \\ \alpha_i &\leftarrow \frac{\gamma_i}{\mu_i^2}, \\ \beta &\leftarrow \frac{N - \sum_i \gamma_i}{\|\mathbf{t} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2}. \end{aligned}$$

Here, $\boldsymbol{\alpha}$ and β represent the hyper-parameters for the prior distribution of weights and target. γ_i is interpreted as a measure of how well-determined the weight w_i is by the data. $\boldsymbol{\Phi}$ can be defined as a matrix composed of training vectors transformed by the basis function. $\mathbf{t}$ is the Q-learning target.

Fig. 3 shows the application of fRVM to a classic quadratic sine curve fit problem from Tipping, et al., [29]. Manually setting the seven fixed RVs, we can obtain good predictions with the radial basis function (RBF) kernel parameter $\gamma_k = 0.06$ and tolerance 1×10^{-3}.

B. RVM-based Practice for RL

In the practice stage, a regular RVM predicts state changes and discovers the relevance vectors for reinforcement learning tasks. Our hypothesis is that learning the dynamics of the world can result in the discovery of knowledge that can be applicable to reinforcement learning tasks. This was empirically examined with deep networks [1]. We expect this will be the same with a RVM function approximation. RVMs relate the learned experience to input samples, so we expect to interpret what was learned in the practice stage more easily than from the pretrained neural network structures.

To improve the learning speed and examine the practice contribution, we adopt fRVM-based reinforcement learning that do not change RV bases. In previous section, we observed fRVMs can fit well when the bases are known. From the randomly explored or collected samples, we can train an RVM to predict the next state or the difference between current and next state. The learned RVM produces relevance vectors that capture key dynamics of an environment. Assuming these RVs are known bases, we can build a fixed-basis RVM for reinforcement learning (fRVM-RL). Now the fRVM-RL adjusts weights, so it estimates Q vales based on similarity kernel features to the learned RVs.

In a reinforcement learning framework, we use fRVM as a function approximator that estimates Q-values. Unlike RVM-RL, it does not need to maintain multiple RVMs and does not need to transfer RVs in each step. It simply updates weights following the RVM update rules. Thus, after practice results in good bases for reinforcement learning, fRVM-RL can learn a policy very efficiently. Algorithm 1 describes the learning algorithm for practice and fRVM-based reinforcement learning.

Algorithm 1 RVM-Practice and fRVM-RL

```
Collect $L$ samples of tuple $(s, a, s')$ using environment dynamics.
Set regression target $z = s'$ or the state changes $z = s' - s$.
**Train RVM** and discover basis $\mathbf{RV}_{practice}$ and weights $\mathbf{w}_{practice}$.
**Initialization:** the basis sample $\mathbf{X}^{\mathbf{RVM}}$ and weights $\mathbf{w}$ of **fRVM** with practiced $\mathbf{RV}_{practice}$ and $\mathbf{w}_{practice}$.
**Choose** discounting factor $\gamma \in (0, 1]$ and learning rate $c$.
**for** each mini-batch **do**
    **Select** action $\mathbf{a_t}$ given state $\mathbf{s_t}$ by $\epsilon$-greedy action selection. Apply $\mathbf{a_t}$ to arrive at $\mathbf{s_{t+1}}$.
    **Observe** $N$ samples, $(\mathbf{s_t}, \mathbf{a_t}, r_{t+1}, \mathbf{s_{t+1}})$ at consecutive time steps $t$.
    **Set** target $y = r_{t+1} + \gamma \max_a Q_{\mathbf{w},\boldsymbol{\alpha},\beta}(s_{t+1}, a)$
    **Train fRVM**
    $\mathbf{w} = (1 - c)\mathbf{w}_t + c\mathbf{w}_{t+1}$
    Decreases $\epsilon$
**end for**
```

How to collect practice samples to improve the target learning performance is a significant issue. Various practice approaches can be investigated but in this paper, we focus on previously examined state-transition dynamics samples and random sample collection. First, we can use a simulation of a dynamic system to generate samples. In this case, we can have two different options for the regression target, the next state or the difference between the next state and current state. When using dynamics simulation, the next state is close to the previous state and the samples are more likely dependent on each other, which can require more samples for the practice stage. With the state difference target, we can reduce the required number of samples and increase the independence of samples. Random sampling can be used when there is no simulation model for dynamic sampling. Without knowing the dynamics of the world, it randomly generates samples with a certain distribution. Also, it can reduce the possible biased sampling from simulated dynamics. However, it is difficult to understand what it learned from this randomly sampled practice by disconnecting correlation between s_t and s_{t+1} since s_{t+1} is not dependent on s_t and a_t. Further strategies should be investigated for better practice models and efficient reinforcement learning. We will discuss more about this issue in following sections.

IV. Experiments

We investigate the efficacy of RV bases that is built by RVM-based practice with two classic reinforcement learning benchmark tasks. The first task is the mountain-car problem consisting of an under-powered car that must climb up a hill is tested. For the second task, we test the pole balancing problem in which a pole must be balanced by pushing on the cart to which it is attached. We compare RVM-based practice methods on these tasks with the following RL algorithms. Neural fitted-Q learning (NN) is a well-known successful learning framework with a similar mini-batch learning structure. The Gaussian process temporal difference (GPTD) algorithm has a Bayesian structure similar to RVMs. Finally, RVM-RL is included to allow the direct examination of the efficiency of practice.

To examine the cases with large search space for practice, we apply the proposed fixed RVM-RL to two Box2D problems in OpenAI Gym. The first task is the lunar lander that controls a spaceship control task that fires main and side engines to smoothly reach on the landing pad. The second task, car racing, is to learn how to control a car from the top-down racing track image pixels. Both problems require large amounts of exploration to obtain a good policy. Although solutions are found for lunar lander, yet no one found solutions for the car racing task. With these two examples, we discuss the limitations of fixed bases learning approach.

A. Mountain Car

The mountain car is a popular dynamics problem that controls an under-powered car that cannot climb directly up the right hill, but must first be pushed up the left hill to gain enough momentum to reach the goal at the top of the right hill. Available actions are push forward (+1), push backward (-1), and no acceleration (+0). The optimal solution that needs to initially drive the car away from the goal makes the problem harder. This continuous control dynamics are described in detail in [26].

The state is represented in two dimensions: the car position x_t and its velocity $\dot{x_t}$. Following the classic mountain car, we assign a reward -1 on each time step. When it reaches the goal ($x_t = 0.5$) at the top of the right hill, the agent gets the reward 0 and is restarted from a random position. With this random position restarting, the fixed number of samples are collected for training. The described reinforcement function is defined as follows:

$$r_t = \begin{cases} 0 & \text{if } x_t \geq 0.5 \\ -1 & \text{otherwise} \end{cases}$$

For the practice stage, we randomly generate samples $p = (x_t, \dot{x_t})$ in the range of $x_t \in [-1.2, 0.5]$ and $\dot{x_t} \in [-2.0, 2.0]$. 10 repetitions of the generation of 1000 practice samples result in different numbers of RVs, ranging from 12 to 18. During this practice stage, the RBF kernel parameter γ_k is set to 1.0, and the maximum number of iterations is limited to 100.

The reinforcement learning discount factor γ is set to 0.9. To test with a small number of exploration actions, we exponentially decrease ϵ from 1 to 0.1 with a factor 0.9885. With the decreasing ϵ, actions are chosen by ϵ-greedy algorithm. This is repeated 1000 times. The mini-batch of 1000 steps is used to update the fRVM weights for Q function estimation. This is repeated for 200 mini-batches. For fRVM-RL training, we preset the fRVM with the $RV_{practice}$ achieved from the practice stage. The RBF kernel parameter γ_k is not changed from 1.0 to accommodate the achieved knowledge. The learning rate $c = 0.2$ was best-performing in our pilot tests and used for the mountain car task tests. The fRVM maximum number of iterations is set to 10. Neural networks with two hidden layers, each of 20 units, are chosen for comparison. Moller's scaled conjugate gradient optimization algorithm [20] was limited to 20 iterations to avoid overfitting. For GPTD, the RBF kernel parameter $\gamma_k = 0.01$, the accuracy threshold $v = 0.1$, the convergence threshold $\eta = 1 \times 10^{-4}$, and initial standard deviation $\sigma_0 = 0.1$ result in the best performance.

fRVM updates only the weights in the middle of training, and as we can see in Fig. 4, fRVM-RL quickly finds the best policy. Comparing the convergence point, fRVM-RL with practice converges to good performance with 100 fewer mini-batches than the previously best performing algorithm, RVM-RL. Since we start $\epsilon = 1$, we observe that the starting points of the curves are not different, and the transferred weights are not utilized for jumpstart test. However, by not adding or removing bases in the middle of training, the training is simplified with linear weight updates that reduces learning time considerably. Most of all, this exemplifies the RV bases obtained from practice well capture the distributions of main factors for correct Q estimation.

B. Pole Balancing

Adding a pole to a cart that swings in two dimensions, Barto, et al., [2] first introduced the benchmark pole-balancing problem. The objective is to apply forces to a cart in a given track and to keep the pole from falling over. Three actions to control the cart are defined: move left, move right, and apply zero force. We define this problem as a continuing task with discounting factor γ, so the goal is to maintain the pole upright as long as possible.

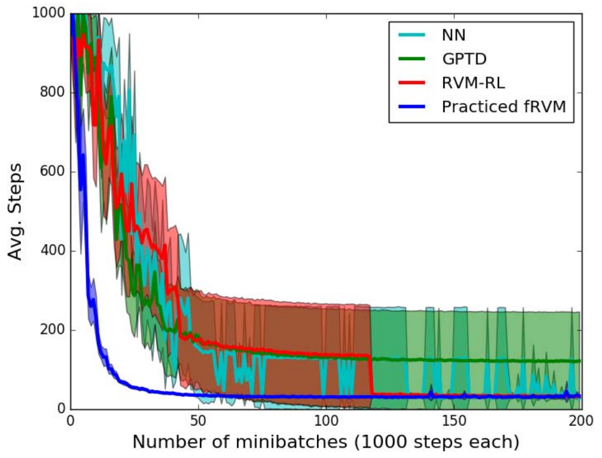


Fig. 4. Average of steps to reach the goal in mountain car problem. The average curve line is computed from 10 experiments. Practice reduces the required number of samples greatly. 1000 samples (the number of steps in one minibatch) are used for practice. The shaded areas represent 95 % confidence interval.

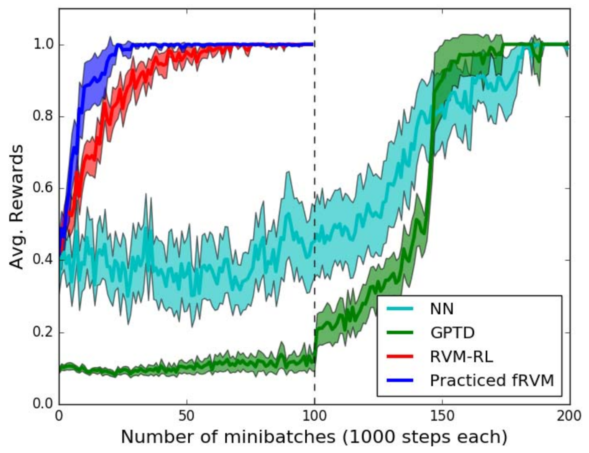


Fig. 5. Average of rewards for each episode in cart-pole balancing. Again, practice helps to converge quickly at the optimal policy. 100 samples (10 % of the number of steps in one minibatch) are used for practice. The shaded areas represent 95 % confidence interval.

The state of this system is four dimensional: the cart position x_t, its velocity $\dot{x_t}$, the pole angle θ_t, and the angular velocity $\dot{\theta}_t$. When the angle $\theta_t = \pi$, the pole is upright. The reward function is defined in terms of the angle as follows:

$$r_t = \begin{cases} 1 & \text{if } |\theta_t - \pi| < \frac{\pi}{3} \\ 0 & \text{otherwise} \end{cases}$$

Thus, when it can balance the pole through the simulation time, the optimal policy will lead to an average reward of 1.0.

For the practice stage in the pole balancing environment, we use dynamic simulation and train an RVM to predict state changes. 100 practice samples are good enough to produce the necessary bases for fRVM-RL training. In 30 practice stages, RVMs produced from 7 to 20 of RVs. For practice, the RBF kernel parameter $\gamma_\text{k} = 20$, and the maximum number of iterations is limited to 100.

All tests share the discount factor $\gamma = 0.99$ and decrease ϵ from 1.0 to 0.1 exponentially. For training, 100 mini-batches with 1000 samples each are collected. fRVM-RL uses the RBF kernel parameter $\gamma_\text{k} = 20$ and the learning rate $c = 0.2$. The best parameters for the neural networks and GPTD were found from pilot tests. Neural networks with 10 units in each of two hidden layers was the best performing structure with the maximum number of iterations for SCG was set to 80. GPTD performed best when $v = 1 \times 10^{-5}$ and $\eta = 0.1$ Initial standard deviation $\sigma_0 = 10$ and the RBF kernel parameter γ_k is set to 1×10^{-5}. As we can see in following results, even with the best parameters, we cannot make the two function approximators work in 100 minibatches. They required twice as many samples to find an optimal policy.

Similar to the mountain car task, we can observe that practice greatly contribute establishing good basis for reinforcement learning function approximation. The fRVM-RL quickly reaches the optimal point and steadily converges. Comparing to RVM-RL, fRVM-RL can save more than 40 mini-batches that contain more than 40,000 samples. For the pole balancing task, we found that adding 100 samples for practice results in learning good performance quickly, reducing the number of samples needed to approximate Q function correctly by 40,000 samples.

C. *Racing Car and Lunar Lander*

To examine the efficacy of practice in complex problems, we applied the fRVM-RL to Box2D problems in OpenAI Gym such as CarRacing-v0 and LunarLander-v2. In the LunarLander-v2, an agent chooses one of four actions: nothing, fire left orientation engine, fire main engine, and fire right orientation engine. The goal is landing the craft on the landing zone smoothly. Thus, the reward between 100 and 140 is given when it lands near zero speed. When it lands in the goal and rests, it gets additional 100 while it gets -100 when it crashes on the surface. Firing main engine cost -0.3, and each leg contact to ground gives 10.

For this problem, most samples that are collected during practice are only the crashing on the surface. Thus, without strategic practice sampling, it gathers samples without any positive bases around the high rewarding states, and resulting bases make feature values to near-zero, preventing a good estimation of Q values.

This problem gets worse in CarRacing-v0. CarRacing is a problem that controls a racing car from the top-down image of the racing environment. An agent controls steering, acceleration and deceleration. The states are represented by 96 by 96 image pixels, and each frame costs -0.1 reward value. Visiting each track tile is worth $1000/N$ when the number of track tiles are N.

Similar to LunarLander-v2, CarRacing practice does not sample enough. Mostly it gathers samples around the starting position and makes it hard to estimate Q values when a car travels far from the starting region with zero feature values. Thus, more strategic practice approaches are required for complex domain problems. We will discuss this issue in next section in detail.

V. Discussion

From the results in the previous section, two questions arise. How does the kernel parameter affect learning? After fRVM-RL training, does it select right basis only for the RL task? We discuss these questions in this section.

A. Analysis of Practice

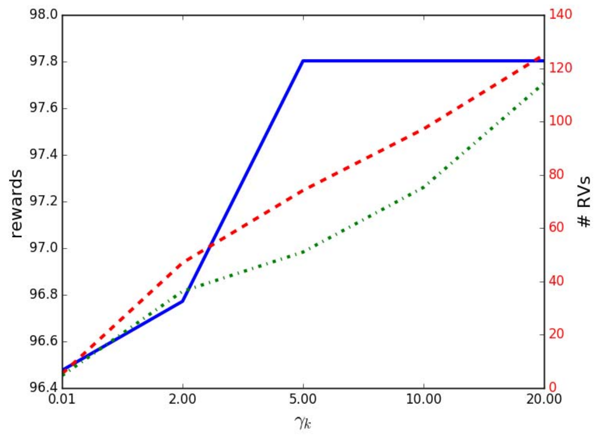
To answer the first question, we plot Fig. 6. The blue line represents the mean of the area under the mean reward curve. For this plot, 10 test results for each γ_k value are collected. The red line and green line represent the number of RVs after practice and active RVs after RL-train. Here, the number of active RVs are recorded by counting the weights greater than 1.0×10^{-5} in magnitude.

As γ_k grows, the base width for RBF gets smaller, which results in more of RVs. However, the number of active RVs decreases because only task-related bases will capture the significant mass [33]. Thus, we can observe that with an RBF kernel, the selection of the kernel parameter greatly affects basis construction and reinforcement learning performance.

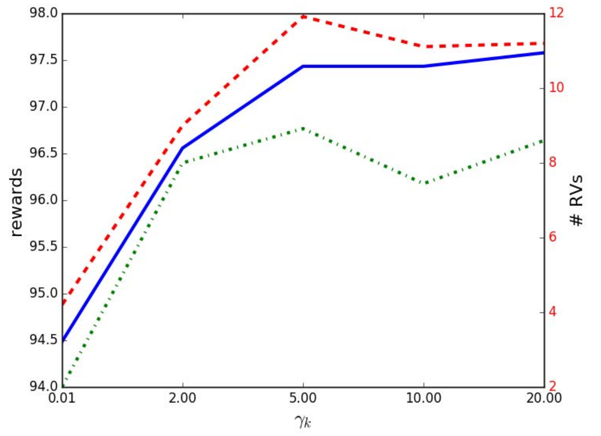
Another observation from Fig. 6 is the difference between sampling methods. With random sampling, the practice stage generates a larger number of RV bases while dynamic sampling eliminates unnecessary RVs. In the case of next state prediction, the small state change makes the sample similar to existing bases so that it can discard similar RVs. With state change prediction, the RVM can remove samples that do not incur state changes, which can also result in a reduction in the number of RVs. Interestingly, when a large number of RVs are used as a fixed basis, the active RVs are spread over almost all of the basis and the number is not reduced. We can intuitively assume that this is caused by the small likelihood that randomly generated samples coincide with the true basis. This investigation answers the question that we raised in our previous pretraining study with neural networks. Random generation of samples seems to be less likely to generate a good basis that is near-orthogonal.

B. Construction of Bases

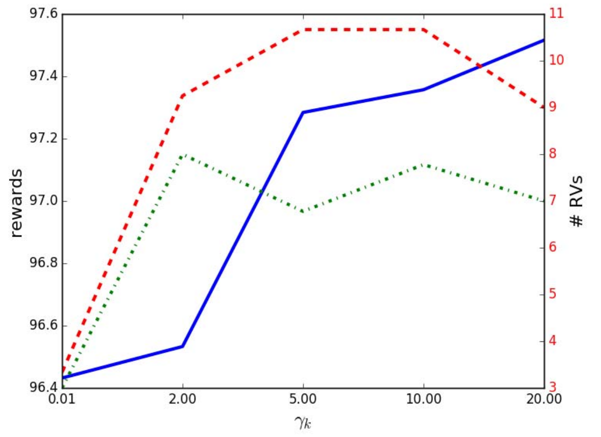
A few authors have previously studied basis construction for supervised learning or semi-supervised learning. Raina, et al., [21] posed the self-taught learning approach that requires the learned structure (or basis) from unlabeled data to be applicable to labeled classification tasks. This enables transfer from unsupervised learning to supervised learning tasks by building a basis from unsupervised learning training. Deep learning [10], [9], [5], [3] pretrains hidden layers of neural networks in unsupervised ways and learns the



(a) random sampling



(b) dynamic sampling - next state prediction



(c) dynamic sampling - state change prediction

Fig. 6. The effects of the RBF kernel γ_k selection with different sampling and target options. The green dashed dot line represents the number RVs after practice, and the red dashed line shows the number RVs with non-zero weights. The blue line depicts the mean of the area under the reward curve. The blue line is scaled on the left reward y-axis and the other two are scaled on the right # RVs y-axis. Only average values are presented for clear reading of plots. The variances of the number of RVs (green and blue) are less than 1 in (b) and (c) and less than 6 in (a). The range of variation in the reward values is between 1.06 and 6.89.

network connectivity structures to be applicable to a target supervised learning task. However, none of these considered reinforcement learning tasks that learn from evaluations of an agent's behavior or reinforcements rather than from known output labels. Anderson, et al., [1] first proposed constructing neural network structures from state dynamics prediction for reinforcement learning. However, it is difficult to interpret the learned network structure. By using RVMs, we clearly see the RV bases and how features are generated with a kernel function. This increases the understanding of the learned bases and environment dynamics.

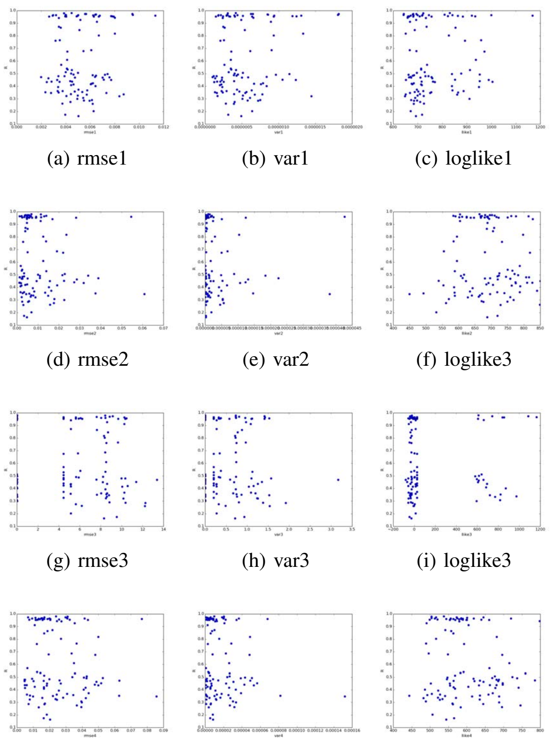
Furthermore, by providing fixed-basis RVM, the approach increases the efficiency of learning. Classical radial basis functions [27], polynomial bases [12], and Fourier bases [11] work well, but how to select a basis is not well understood. Learning in a small source task, proto value functions (PVFs) [18] automatically specify an ortho-normal set of basis functions. This bases can be transferred to tasks with different goals or in a slightly different state space. Practice poses a harder problem and requires samples without reinforcements or objectives. The RVM bases learned through practice can be transferred to a broader range of tasks than PVFs.

When the RV bases are well-established from practice, they support successful learning. As we discussed earlier, when unstable sampling or learning parameters are chosen, it is possible to learn poorly on reinforcement learning tasks. For efficient practice, we can investigate 1) kernel methods for reliable basis construction, 2) practice strategy development, 3) search space reduction, 4) cyclic training of practice and learning, and 5) non-fixed, dynamic learning based on practiced knowledge.

Knowing the effectiveness of bases can automate the basis construction during practice. That is, we can automate the process of finding a good kernel and its parameters. Also, when the found bases are not good enough, we can restart practice or increase the number of samples. Or, with cycles of practice and reinforcement learning, learning can be improved further.

For this, we examine if there is any correlation between the hyperparameters and average rewards. Some preliminary tests were run to evaluate this and collected 100 samples with successful runs ($\sum_t r_t >= 0.9 * N$ where N is the number steps) and poor ones ($\sum_t r_t < 0.9 * N$).

We recorded practice RMSEs, log likelihoods, and variances for each output dimension along with average rewards. We observed that the data is scattered wide and trends are not obvious (Fig. 7). We tested some classification algorithms, such as LDA, QDA, linear and nonlinear logistic regression, to see if it can be classified. The label is set to true if the mean reward is greater than 0.9 and false otherwise. Table I shows the classification accuracy with the 12 features. Nonlinear logistic regression seems to clearly separate successful cases by looking at the RMSE, variance and log-likelihood. This tells us that the selected features are strongly related to the subsequent reinforcement learning performance, so can be the bases for an approach to predicting the success of a practice model. We will investigate this further with more samples and other environments to examine if it can be generalized to other tasks.



(a) rmse1 (b) var1 (c) loglike1

(d) rmse2 (e) var2 (f) loglike3

(g) rmse3 (h) var3 (i) loglike3

(j) rmse4 (k) var4 (l) loglike4

Fig. 7. Scatter plots of features against average rewards. Errors, variances, and log likelihoods for each dimension are selected features.

TABLE I
PRELIMINARY EVALUATIONS FOR EXAMINATION OF AUTOMATED PRACTICE WITH 100 PRACTICE AND FRVM-RL SAMPLES

Classifier	Accuracy
LDA	63 / 100
QDA	69 / 100
Linear Logistic Regression	74 / 100
Nonlinear Logistic Regression	100 / 100

VI. CONCLUSION

We examined the efficacy of practice for reinforcement learning tasks and observed increased efficiency. By training an RVM and obtaining RV bases from dynamics prediction, we were able to successfully transfer the bases and to show improved learning in classical benchmark problems. With fixed basis learning, we demonstrated how effectively practice establishes bases in these examples. Also, we observed the limitations of practice when the tasks get complex. Discussion leads to plans for investigation of more efficient ways to practice and the measure that evaluates how well bases are constructed.

A major contribution of this research is the demonstration of the importance of practice in a machine learning context. Without providing any objective or reinforcement, practice with an RVM extends exploration before starting to solve the actual RL task and generates sparse but helpful bases.

From our discussion about results and practice evaluation, we can further investigate stable kernel methods and search space reduction. Practice strategies such as human or agent guided practice (”coaching”), and cyclic repetition of short practice and short learning will be interesting direction for future study. Allowing adaption of the basis learned from practice might be necessary not only to compensate for lack of practice but also to be easily extended to the continuous action tasks [15], [13]. However, investigation of efficient transfer learning should be studied due to possible disturbances from practice. This line of research will be continued through additional experiments with the OpenAI Gym tasks.

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] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE Transactions on Systems, Man and Cybernetics*, no. 5, pp. 834–846, 1983.

[3] Y. Bengio, “Learning deep architectures for ai,” *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[4] J. Blitzer, R. McDonald, and F. Pereira, “Domain adaptation with structural correspondence learning,” in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 2006, pp. 120–128.

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?” *The Journal of Machine Learning Research (JMLR)*, vol. 11, pp. 625–660, 2010.

[6] 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*, 2006, pp. 720–727.

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

[8] J. Gao, W. Fan, J. Jiang, and J. Han, “Knowledge transfer via multiple model local structure mapping,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 283–291.

[9] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[10] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] G. Konidaris, S. Osentoski, and P. Thomas, “Value function approximation in reinforcement learning using the Fourier basis,” in *Proceedings of the 25th Conference on Artificial Intelligence*, August 2011, pp. 380–385.

[12] M. G. Lagoudakis and R. Parr, “Least-squares policy iteration,” *The Journal of Machine Learning Research (JMLR)*, vol. 4, pp. 1107–1149, 2003.

[13] M. Lee and C. W. Andersno, “Relevance vector sampling for reinforcement learning in continuous action space,” in *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016, (to appear).

[14] ——, “Robust reinforcement learning with relevance vector machines,” in *Robotics: Science and Systems (RSS) Workshop on Robot Learning and Planning*, 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] 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.

[17] 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.

[18] S. Mahadevan and M. Maggioni, “Proto-value functions: A laplacian framework for learning representation and control in markov decision processes.” *The Journal of Machine Learning Research (JMLR)*, vol. 8, no. 2169-2231, p. 16, 2007.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 02 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[20] M. F. Møller, “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural networks*, vol. 6, no. 4, pp. 525–533, 1993.

[21] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: transfer learning from unlabeled data,” in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007, pp. 759–766.

[22] R. Raina, A. Y. Ng, and D. Koller, “Constructing informative priors using transfer learning,” in *Proceedings of the 23rd international Conference on Machine learning (ICML)*, 2006, pp. 713–720.

[23] I. Rexakis and M. G. Lagoudakis, “Directed policy search using relevance vector machines,” in *2012 IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI)*, vol. 1, 2012, pp. 25–32.

[24] 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.

[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

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

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

[28] M. E. Taylor and P. Stone, “Cross-domain transfer for reinforcement learning,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 879–886.

[29] M. E. Tipping, “Sparse bayesian learning and the relevance vector machine,” *The Journal of Machine Learning Research (JMLR)*, vol. 1, pp. 211–244, 2001.

[30] M. E. Tipping, A. C. Faul *et al.*, “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.

[31] L. Torrey, J. Shavlik, T. Walker, and R. Maclin, “Relational skill transfer via advice taking,” in *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[32] 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.

[33] D. Wipf, J. Palmer, and B. Rao, “Perspectives on sparse bayesian learning,” in *Advances in Neural Information Processing Systems 16*, 2003.

[34] P. Wu and T. G. Dietterich, “Improving svm accuracy by training on auxiliary data sources,” in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 110.

[35] V. W. Zheng, S. J. Pan, Q. Yang, and J. J. Pan, “Transferring multi-device localization models using latent multi-task learning.” in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, vol. 8, 2008, pp. 1427–1432.