Reinforcement Learning

R. B. Alexander

Reinforcement learning (RL) is a technique for training an agent in an environment to take actions that maximize the expected utility over a given horizon - solving a Markov decision process (MDP). We focus here on infinite horizon problems with discount factor γ. At time t, the agent is in state $s_t \in \mathcal{S}$ and can take action $a_t \in \mathcal{A}$. Using the value function $U(s)$ or the state-action value function $Q(s, a)$ (or an approximation thereof), which describes the discounted future expected utility of being in state s and taking action a, the agent selects the optimal action according to the optimal policy $a_t = \pi^*(s_t)$. The agent takes action a_t, receives reward r_t, and transitions to state s_{t+1}. Under the Markov assumption, the state transition model $T(s_{t+1} \mid s_t, a_t)$ and reward model $R(s_t, a_t)$ are stationary. Various techniques can be used to compute optimal or approximately optimal policies for acting in the environment. When the transition model and reward model are known exactly, an optimal policy can be computed using dynamic programming algorithms (value iteration, policy iteration, direct policy search); when the transition model and reward model are not known exactly, an approximately optimal policy can be computed using approximate dynamic programming algorithms (maximum-likelihood estimation of T and R, Q-learning, SARSA, local approximation, global approximation). In the following paper, we utilize these algorithms, discuss and utilize methods for improving generalization, and show results for three datasets of varying size that have discretized state and action spaces.

Modifications for Improving Generalization

Among the challenges present in reinforcement learning – assigning credit to past actions, generalizing from limited experience, and balancing exploration and exploitation – generalization is arguably the most difficult. We present methods below that we used in improving generalization of the policies found using general RL algorithms.

Modal Policy Updating

We introduce modal policy updating as a way of locally generalizing from limited experience. The principle is that the policies in the neighborhood of an unvisited state are a good approximation for the policy of the unvisited state. The policy of the unvisited state is the most likely policy of the neighborhood, so we use the mode of the neighborhood's policy. A suitable representation for the neighborhood of s used herein is that the ℓ_1 norm of two state representations should be less than some distance r. It is appropriate to normalize the components of the state representation relative to the size of the state space along each dimension. For example, suppose we have a state space $\mathcal{S}$ with 50,000 states that can be decomposed into a representation in $s_1 \times s_2 \subset \mathbb{Z}^2$ with $s_1 = [1, 100], s_2 = [1, 500]$. A suitable condition for the neighborhood around s' could be the following condition.

$$(s_1 - s')^2 + \left(\frac{s_2 - s'}{5}\right)^2 \leq R^2$$

For modal policy updating, we first compute the value function and optimal policy for the discretized state space using any algorithm without generalization techniques (value iteration, policy iteration, policy search, Q-learning, Sarsa, eligibility trace methods) and make a copy of the optimal policy called $\tilde{\pi}$. Then for each state in $\mathcal{S}$, we follow the following procedure. First, compute the neighborhood of s, $N(s) \subset \mathcal{S}$, where the neighborhood excludes s. The structure of the state space can be exploited to define an efficient neighborhood representation. Then, remove the states in the neighborhood where the value function at s is 0 to select the valid neighborhood $N'(s) = s \in N(s) | U(s) \neq 0$. If the valid neighborhood is non-empty, the policy at s, updated to the mode of the policies of the valid neighborhood $\tilde{\pi}(s) = mode(\pi(N'(s)))$. After iterating over the state space, the modally-updated policy is $\tilde{\pi}$.

Global Approximation using Fourier Series Expansion

Global approximation is used to approximate the value function over the entire state space. Since the Fourier series can approximate any function, we use a several-term decomposition to approximate the value function without resorting to arbitrary basis functions. Using an efficient representation of the state space leads to better approximation. In the cases presented here, the most efficient representation was 2D, so a 2D Fourier series expansion $f_a(s_1, s_2)$ with lengthscales $\lambda_{s_1}, \lambda_{s_2}$ was used with three terms and decomposed along the action space for $a \in \mathcal{A}$:

$$f_a(s_1,s_2) = \sum_{n=1}^{3}\sum_{m=1}^{3} \alpha_{n,m} \cos\left(\frac{2\pi n s_1}{\lambda_{s_1}}\right)\cos\left(\frac{2\pi m s_2}{\lambda_{s_2}}\right) + \sum_{n=1}^{3}\sum_{m=1}^{3} \beta_{n,m} \cos\left(\frac{2\pi n s_1}{\lambda_{s_1}}\right)\sin\left(\frac{2\pi m s_2}{\lambda_{s_2}}\right) + \sum_{n=1}^{3}\sum_{m=1}^{3} \gamma_{n,m} \sin\left(\frac{2\pi n s_1}{\lambda_{s_1}}\right)\cos\left(\frac{2\pi m s_2}{\lambda_{s_2}}\right) + \sum_{n=1}^{3}\sum_{m=1}^{3} \delta_{n,m} \sin\left(\frac{2\pi n s_1}{\lambda_{s_1}}\right)\sin\left(\frac{2\pi m s_2}{\lambda_{s_2}}\right) + c$$

Results

In our implementation, we used a variety of methods including:

1. value iteration using the maximum-likelihood estimate of T and R
2. Gauss-Seidel value iteration using the maximum-likelihood estimate of T and R
3. sarsa(λ)
4. sarsa(λ) with local approximation (k-nearest neighbors)
5. sarsa(λ) with global approximation (fourier series)

For each of the above, we performed modal policy updating.

Small Dataset

The `small` dataset describes a 10x10 grid-world with $|\mathcal{S}| = 100$ and $|\mathcal{A}| = 4$. The discount factor for the problem is given to be $\gamma = 0.95$. The dataset has a convenient representation as two position (x, y) coordinates in space and the actions represent moving left, right, up, or down. For this dataset, we show the results of Gauss-Seidel value iteration and sarsa(λ) with global approximation. Value iteration performs best here, even with the tuning of the learning rate and eligibility trace parameter for sarsa. We observe the convergence of the Fourier series to an approximate value function, though it does not yield as good of a policy as value iteration.

Since the state space was relatively small and most states had a nonzero value function, the modally-updated policies are essentially equivalent to the original policies.

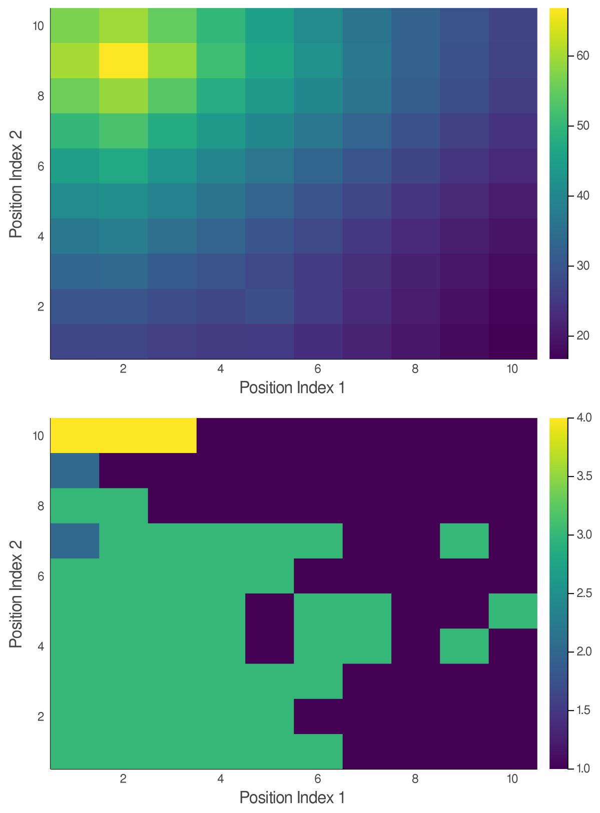


Figure 1 Optimal value function and policy in the positional state space representation using Gauss-Seidel value iteration ($\epsilon = 0.1$) for the `small` dataset.

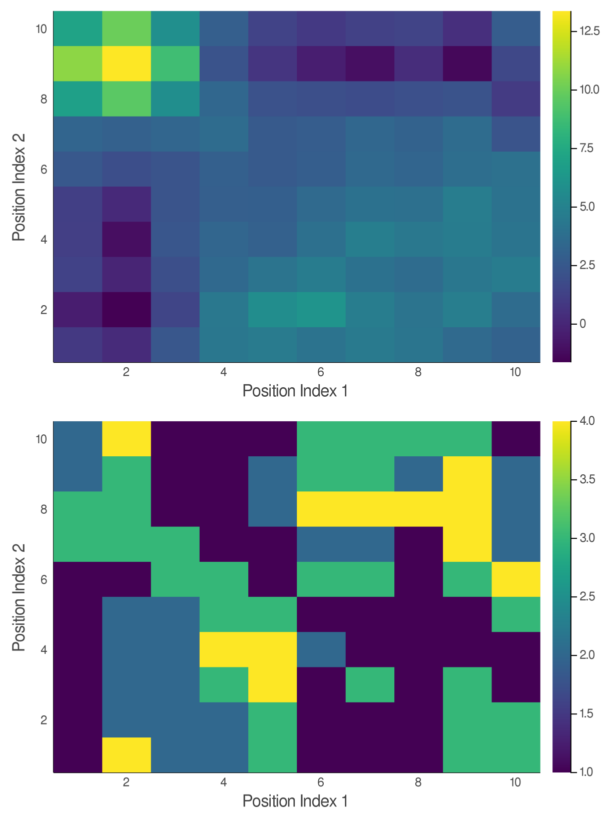


Figure 2 Optimal value function and policy in the positional state space representation using sarsa(λ) with global approximation ($\lambda = 0.9; \alpha = 0.05$) for the `small` dataset.

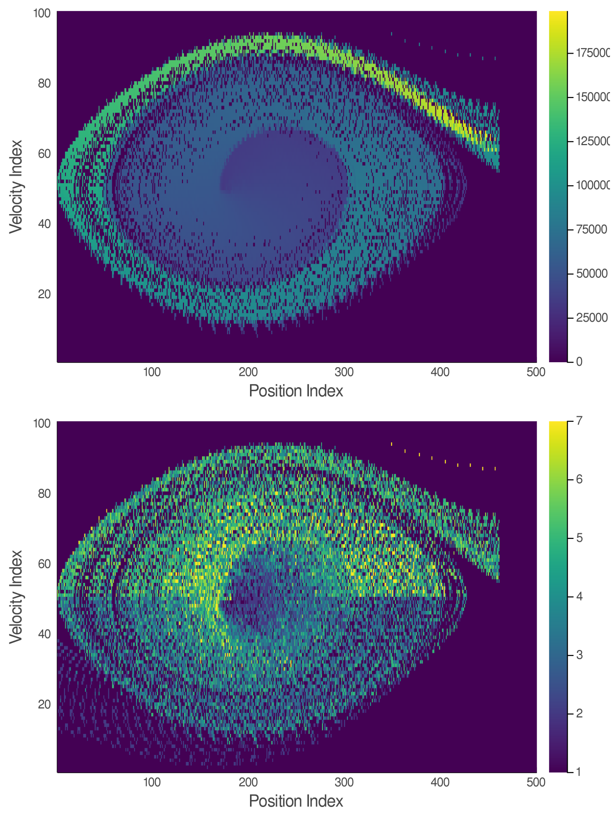


Figure 3 Optimal value function and policy in the phase space representation using Gauss-Seidel value iteration ($\gamma = 0.99; \epsilon = 1000$) for the `medium` dataset.

Medium Dataset

The `medium` dataset describes mountain car environment with $|\mathcal{S}| = 50000$ and $|\mathcal{A}| = 7$. The problem is given to be undiscounted. The dataset has a convenient representation as position and velocity (x, v) coordinates in phase space and the actions represent varying degrees of positive and negative acceleration. For this dataset, we performed Gauss-Seidel value iteration and sarsa(λ). Value iteration performs best for the cost and ease of use. Since the state space is relatively large and we have many missing states in our dataset, the modally-updated policies are even better than the original policies, which are missing reasonable policies for the unvisited states. Our condition for the neighborhood was: $(v - v')^2 + ((x - x')/5)^2 \leq (1.5)^2$.

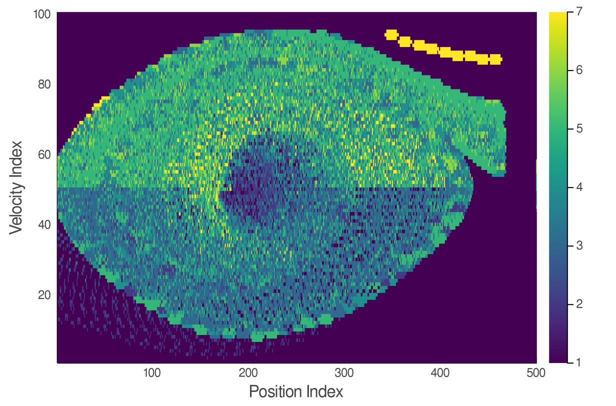
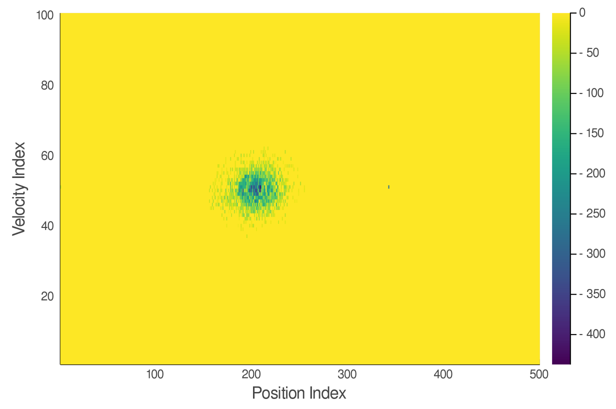


Figure 4 Optimal modally-updated policy in the phase space representation using Gauss-Seidel value iteration ($\gamma = 0.99; \epsilon = 1000$) for the `medium` dataset.

Sarsa runs can give good policies but at the expense of having to tuning the learning rate and eligibility trace parameter and re-run the algorithm. As a result, we did not find a decent policy using sarsa.



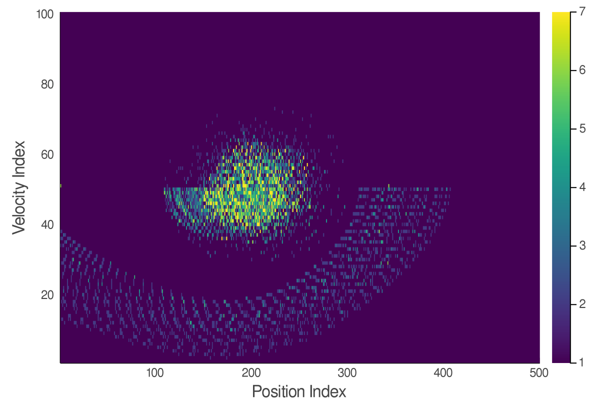
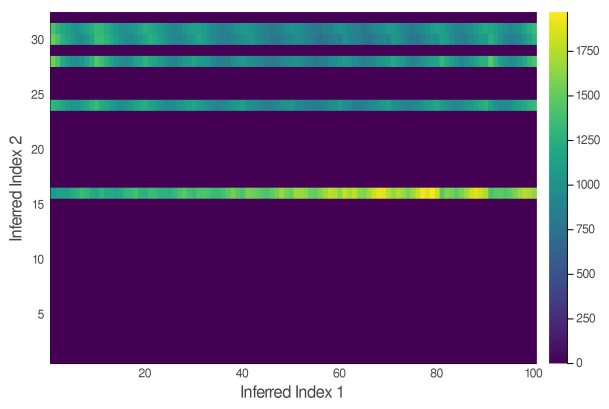


Figure 5 Optimal value function and policy in the phase space representation using sarsa(λ) ($\gamma = 0.99; \lambda = 0.9; \alpha = 0.05$) for the `medium` dataset.

Large Dataset

The `large` dataset describes an unknown environment with $|\mathcal{S}| = 312020$ and $|\mathcal{A}| = 9$. The discount factor for the problem is given to be $\gamma = 0.95$. The dataset appears to have a convenient representation as three sets of coordinates $(ceil(s/10000), mod(s, 10000), mod(s, 100))$. We used only Gauss-Seidel value iteration to find an approximately optimal policy. The representation shown below is the compressed representation with the first coordinate along the y-axis, and the second coordinate along the x-axis (which is a composition of only the unique second coordinates, implying periodicity over the third coordinate). The challenge in this dataset is that the state space is quite large and so an efficient representation is needed to achieve a fast algorithm. Since we can only iterate on the states in our dataset, we have a reasonable compression of the state space itself, so the algorithm is faster that in the `medium` dataset despite the increased size of the state space and action space.



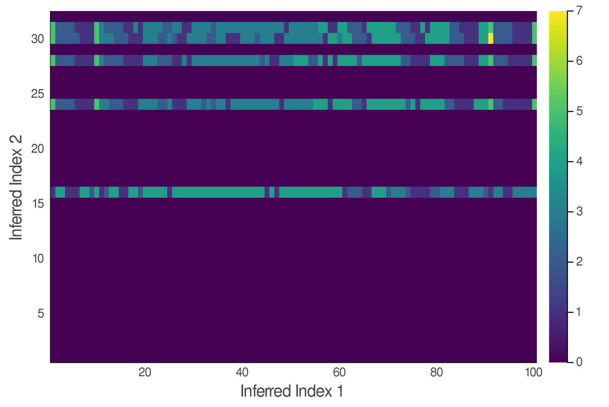


Figure 6 Optimal value function and policy in the compressed representation using Gauss-Seidel value iteration ($\gamma = 0.95; \epsilon = 0.1$) for the `large` dataset.

Table 1 Runtimes in Seconds for Various Algorithms

	Algorithm		
Dataset	VI	GSVI	Sarsa(λ)
`small`	1.264	0.922	1.046
`medium`	-	5460	628.9
`large`	-	220.4	-