Mitigating Planner Overfitting in Model-Based Reinforcement Learning

Dilip Arumugam [1] **David Abel** [2] **Kavosh Asadi** [2] **Nakul Gopalan** [2] **Christopher Grimm** [3] **Jun Ki Lee** [2]
Lucas Lehnert [2] **Michael L. Littman** [2]

Abstract

An agent with an inaccurate model of its environment faces a difficult choice: it can ignore the errors in its model and act in the real world in whatever way it determines is optimal with respect to its model. Alternatively, it can take a more conservative stance and eschew its model in favor of optimizing its behavior solely via real-world interaction. This latter approach can be exceedingly slow to learn from experience, while the former can lead to "planner overfitting"—aspects of the agent's behavior are optimized to exploit errors in its model. This paper explores an intermediate position in which the planner seeks to avoid overfitting through a kind of regularization of the plans it considers. We present three different approaches that demonstrably mitigate planner overfitting in reinforcement-learning environments.

1. Introduction

Model-based reinforcement learning (RL) has proven to be a powerful approach for generating reward-seeking behavior in sequential decision-making environments. For example, a number of methods are known for guaranteeing near optimal behavior in a Markov decision process (MDP) by adopting a model-based approach (Kearns & Singh, 1998; Brafman & Tennenholtz, 2002; Strehl et al., 2009). In this line of work, a learning agent continually updates its model of the transition dynamics of the environment and actively seeks out parts of its environment that could contribute to achieving high reward but that are not yet well learned. Policies, in this setting, are designed specifically to explore unknown transitions so that the agent will be able to exploit (that is, maximize reward) in the long run.

A distinct model-based RL problem is one in which an agent has explored its environment, constructed a model, and must then use this learned model to select the best policy that it can. A straightforward approach to this problem, referred to as the *certainty equivalence approximation* (Dayan & Sejnowski, 1996), is to take the learned model and to compute its optimal policy, deploying the resulting policy in the real environment. The promise of such an approach is that, for environments that are defined by relatively simple dynamics but require complex behavior, a model-based learner can start making high-quality decisions with little data.

Nevertheless, recent large-scale successes of reinforcement learning have not been due to model-based methods but instead derive from value-function based or policy-search methods (Mnih et al., 2015; 2016; Schulman et al., 2017; Hessel et al., 2018). Attempts to leverage model-based methods have fallen below expectations, particularly when models are learned using function-approximation methods. Jiang et al. (2015) highlighted a significant shortcoming of the certainty equivalence approximation, showing that it is important to hedge against possibly misleading errors in a learned model. They found that reducing the effective planning depth by decreasing the discount factor used for decision making can result in improved performance when operating in the true environment.

At first, this result might seem counter intuitive—the best way to exploit a learned model can be to exploit it incompletely. However, an analogous situation arises in supervised machine learning. It is well established that, particularly when data is sparse, the representational capacity of supervised learning methods must be restrained or regularized to avoid overfitting. Returning the best hypothesis in a hypothesis class relative to the training data can be problematic if the hypothesis class is overly expressive relative to the size of the training data. The classic result is that testing performance improves, plateaus, then drops as the complexity of the learner's hypothesis class is increased.

In this paper, we extend the results on avoiding planner overfitting via decreasing discount rates by introducing several other ways of regularizing policies in model-based RL. In each case, we see the classic "overfitting" pattern in which resisting the urge to treat the learned model as correct and to search in a reduced policy class is repaid by improved performance in the actual environment. We believe this re-

[1]Department of Computer Science, Stanford University [2]Department of Computer Science, Brown University [3]Department of Computer Science & Engineering, University of Michigan. Correspondence to: Dilip Arumugam <dilip@cs.stanford.edu>.

search direction may hold the key to large-scale applications of model-based RL.

Section 2 provides a set of definitions, which provide a vocabulary for the paper. Section 3 reviews the results on decreasing discount rates, Section 4 presents a new approach that plans using epsilon greedy policies, and Section 5 presents results where policy-search is performed using lower capacity representations of policies. Section 6 summarizes related work and Section 7 concludes.

2. Definitions

An MDP M is defined by the quantities $\langle S, A, R, T, \gamma \rangle$, where S is a state space, A is an action space, $R : S \times A \to \mathbb{R}$ is a reward function, $T : S \times A \to \mathbb{P}(S)$ is a transition function, and $0 \leq \gamma < 1$ is a discount factor. The notation $\mathbb{P}(X)$ represents the set of probability distributions over the discrete set X. Given an MDP $M = \langle S, A, R, T, \gamma \rangle$, its optimal value function Q^* is the solution to the Bellman equation:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a)_{s'} \max_{a'} Q^*(s',a').$$

This function is unique and can be computed by algorithms such as value iteration or linear programming (Puterman, 1994).

A (deterministic) policy is a mapping from states to actions, $\pi : S \to A$. Given a value function $Q : S \times A \to \mathbb{R}$, the *greedy policy* with respect to Q is $\pi_Q(s) = \operatorname{argmax}_a Q(s,a)$. The greedy policy with respect to Q^* maximizes expected discounted reward from all states. We assume that ties between actions of the greedy policy are broken arbitrarily but consistently so there is always a unique optimal policy for any MDP.

The *value function for a policy π deployed in M* can be found by solving

$$Q^\pi_M(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a)_{s'} Q^\pi_M(s', \pi(s')).$$

The value function of the optimal policy is the optimal value function. For a policy π, we also define the scalar $V^\pi_M = \sum_s w_s Q^\pi_M(s, \pi(s))$, where w is an MDP-specific weighting function over the states.

The *epsilon-greedy policy* (Sutton & Barto, 1998) is a stochastic policy where the probability of choosing action a is $(1-\epsilon) + \epsilon/|A|$ if $a = \operatorname{argmax}_a Q(s,a)$ and $\epsilon/|A|$ otherwise. The optimal epsilon greedy policy for M is not generally the epsilon greedy policy for Q^*. Instead, it is necessary to solve a different set of Bellman equations:

$$Q^\epsilon(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a)_{s'} \times \left((1-\epsilon) \max_{a'} Q^\epsilon(s',a') + \epsilon/|A| \sum_{a'} Q^\epsilon(s',a') \right).$$

The optimal epsilon-greedy policy plays an important role in the analysis of learning algorithms like SARSA (Rummery, 1994; Littman & Szepesvári, 1996).

These examples of optimal policies are with respect to all possible deterministic Markov policies. In this paper, we also consider optimization with respect to a restricted set of policies $\breve{\Pi}$. The optimal restricted policy can be found by comparing the scalar values of the policies: $\rho^* = \operatorname{argmax}_{\rho \in \breve{\Pi}} V_\rho$.

3. Decreased Discounting

Let $M = \langle S, A, R, T, \gamma \rangle$ be the evaluation environment and $\widehat{M} = \langle S, A, R, \widehat{T}, \breve{\gamma} \rangle$ be the planning environment, where $\widehat{T}$ is the learned model and $\breve{\gamma} \leq \gamma$ is a smaller discount factor used to decrease the effective planning horizon.

Jiang et al. (2015) proved a bound on the difference between the performance of the optimal policy in M and the performance of the optimal policy in $\widehat{M}$ when executed in M:

$$\frac{\gamma - \breve{\gamma}}{(1-\gamma)(1-\breve{\gamma})} R_{\max} + \frac{2R_{\max}}{(1-\breve{\gamma})^2} \sqrt{\frac{1}{2n} \log \frac{2|S||A||\Pi_{R,\breve{\gamma}}|}{\delta}}. \quad (1)$$

Here, $R_{\max} = \max_{s,a} R(s,a)$ is the largest reward (we assume all rewards are non-negative), δ is the certainty with which the bound needs to hold, n is the number of samples of each transition used to build the model, and $|\Pi_{R,\breve{\gamma}}|$ is the number of distinct possibly optimal policies for $\langle S, A, R, \cdot, \breve{\gamma} \rangle$ over the entire space of possible transition functions.

They show that $|\Pi_{R,\breve{\gamma}}|$ is an increasing function of $\breve{\gamma}$, growing from 1 to as high as $|A|^{|S|}$, the size of the set of all possible deterministic policies. They left open the shape of this function, which is most useful if it grows gradually, but could possibly jump abruptly.

To help ground intuitions, we estimated the shape of $|\Pi_{R,\breve{\gamma}}|$ over a set of randomly generated MDPs. Following Jiang et al. (2015), a "ten-state chain" MDP $M = \langle S, A, T, R, \gamma \rangle$ is drawn such that, for each state–action pair, $(s,a) \in S \times A$, the transition function $T(s,a)$ is constructed by choosing 5 states at random from S, then assigning probabilities to these states by drawing 5 independent samples from a uniform distribution over $[0,1]$ and normalizing the resulting numbers. The probability of transition to any other state

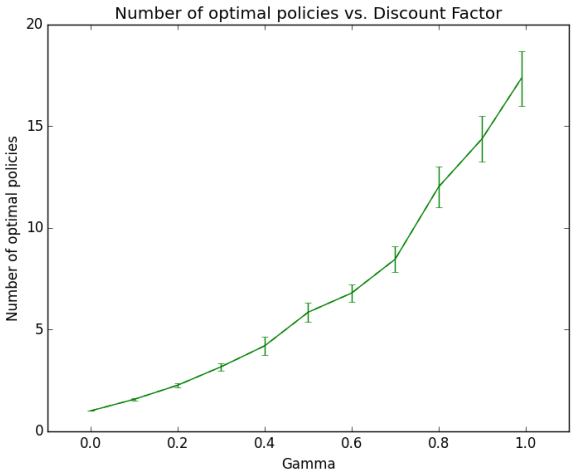

Figure 1. The number of distinct optimal policies found generating random transition functions for a fixed reward function varying $\breve{\gamma}$ in random MDPs.

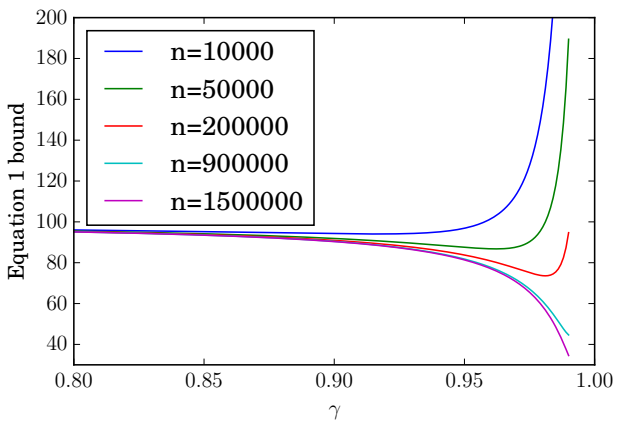

Figure 2. Bound on policy loss for randomly generated MDPs, showing the tightest bound for intermediate values of γ for intermediate amounts of data.

is zero. For each state–action pair $(s, a) \in S \times A$, the reward $R(s, a)$ is drawn from a uniform distribution with support $[0, 1]$. For our MDPs, we chose $|S| = 10$, $|A| = 2$ and $\gamma = 0.99$. We examined $\breve{\gamma}$ in $\{0.0, 0.1, ..., 0.9, 0.99\}$, computed optimal policies by running value iteration with 10 iterations. We sampled repeatedly until no new optimal policy was discovered for 5000 consecutive samples.

Figure 1 is an estimate of how $|\Pi_{R,\breve{\gamma}}|$ grows in this class of randomly generated MDPs. Fortunately, the set appears to grow gradually, making $\breve{\gamma}$ an effective parameter for fighting planner overfitting.

Estimating $|\Pi_{R,\breve{\gamma}}| \approx 11e^{\breve{\gamma}} - 10$, Figure 2 shows the bound of Equation 1 applied to the random MDP distribution ($|S| = 10$, $|A| = 2$, $R_{\max} = 1$, $\gamma = .99$).

Note that the expected "U" shape is visible, but only for a relatively narrow range of values of n. For under 50k samples, the minimal loss bound is achieved for $\breve{\gamma} = 0$. For over 900k samples, the minimal loss bound is achieved for $\breve{\gamma} = \gamma$. (Note that the pattern shown here is relatively insensitive to the estimated shape of $|\Pi_{R,\breve{\gamma}}|$.)

For actual MDPs, the "U" shape is much more robust. Using this same distribution over MDPs, Figure 3 replicates an empirical result of Jiang et al. (2015) showing that intermediate values of $\breve{\gamma}$ are most successful and that this value grows as the model used in planning becomes more accurate (having been trained on more trajectories). We sampled MDPs from the same random distribution and, for each value of $n \in \{5, 10, 20, 50\}$, we generated 1000 datasets each consisting of n trajectories of length 10 starting from a state selected uniformly at random and executing a random policy.

In all experiments, the estimated MDP ($\widehat{M}$) was computed using maximum likelihood estimates of T and R with no additive Gaussian noise. Optimal policies were all found by running value iteration in the estimated MDP $\widehat{M}$. The empirical loss (Equation 14 of Jiang et al. (2015)) was computed for each value of $\gamma \in \{0.0, 0.1, 0.2, \ldots, 0.9, 0.99\}$. The error bars shown in the figure represent 95% confidence intervals.

4. Increased Exploration

In this section, we consider a novel regularization approach in which planning is performed over the set of epsilon-greedy policies. The intuition here is that adding noise to the policies makes it harder for them to be tailored explicitly to the learned model, resulting in less planner overfitting.

In Section 4.1, a general bound is introduced and then Section 4.2 applies the bound to the set of epsilon greedy policies.

4.1. General Bounds

We can relate the structure of a restricted set of policies $\breve{\Pi}$ to the performance in an approximate model with the following theorem.

Theorem 1. *Let $\breve{\Pi}$ be a set of policies for an MDP $M = \langle S, A, T, R, \gamma \rangle$. Let $\widehat{M} = \langle S, A, \widehat{T}, R, \gamma \rangle$ be an MDP like M, but with a different transition function. Let π be the optimal policy for M and $\widehat{\pi}$ be the optimal policy for $\widehat{M}$. Let ρ be the optimal policy in $\breve{\Pi}$ for M and $\widehat{\rho}$ be the optimal policy in $\breve{\Pi}$ for $\widehat{M}$. Then,*

$$|V^{\pi}_{M} - V^{\widehat{\rho}}_{M}| \leq |V^{\pi}_{M} - V^{\rho}_{M}| + 2 \max_{p \in \breve{\Pi}} |V^{p}_{M} - V^{p}_{\widehat{M}}|.$$

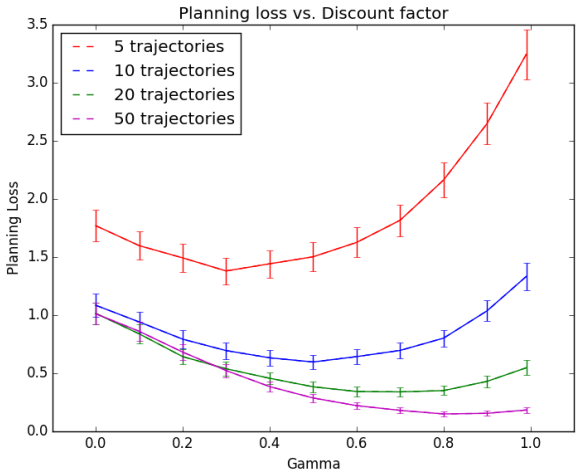

Figure 3. Reducing the discount factor used in planning combats planner overfitting in random MDPs.

Proof. We can write

$$\begin{aligned} & V_M^\pi - V_M^{\widehat{\rho}} \\ &= (V_M^\pi - V_M^\rho) + (V_M^\rho - V_{\widehat{M}}^\rho) \\ &\quad -(V_M^{\widehat{\rho}} - V_{\widehat{M}}^{\widehat{\rho}}) - (V_{\widehat{M}}^{\widehat{\rho}} - V_{\widehat{M}}^\rho) \\ &\leq (V_M^\pi - V_M^\rho) + (V_M^\rho - V_{\widehat{M}}^\rho) - (V_M^{\widehat{\rho}} - V_{\widehat{M}}^{\widehat{\rho}}) \quad (2) \\ &\leq |V_M^\pi - V_M^\rho| + |V_M^\rho - V_{\widehat{M}}^\rho| + |V_M^{\widehat{\rho}} - V_{\widehat{M}}^{\widehat{\rho}}| \\ &\leq |V_M^\pi - V_M^\rho| + 2 \max_{p \in \breve{\Pi}} |V_M^p - V_{\widehat{M}}^p|. \quad (3) \end{aligned}$$

Equation 2 follows from the fact that $V_{\widehat{M}}^{\widehat{\rho}} - V_{\widehat{M}}^\rho \geq 0$, since $\widehat{\rho}$ is chosen as optimal among the set of restricted policies with respect to $\widehat{M}$. Equation 3 follows because both ρ and $\widehat{\rho}$ are included in $\breve{\Pi}$. The theorem follows from the fact that $V_M^\pi - V_M^{\widehat{\rho}} \geq 0$ since π is chosen to be optimal in M. □

Theorem 1 shows that the restricted policy set $\breve{\Pi}$ impacts the resulting value of the plan in two ways. First, the bigger the class is, the closer V_M^ρ becomes to V_M^π—that is, the more policies we consider, the closer to optimal we become. At the same time, $\max_{p \in \breve{\Pi}} |V_M^p - V_{\widehat{M}}^p|$ grows as $\breve{\Pi}$ gets larger as there are more policies that can differ in value between M and $\widehat{M}$.

Jiang et al. (2015) leverage this structure in the specific case of defining $\breve{\Pi}$ by optimizing policies using a smaller value for γ. Our Theorem 1 generalizes the idea to arbitrary restricted policy classes and arbitrary pairs of MDPs M and $\widehat{M}$.

In particular, Consider a sequence of Π_i such that $\Pi_i \subseteq \Pi_{i+1}$. Then, the first part of the bound is monotonically non-increasing (it goes down each time a better policy is included in the set) and the second part of the bound is monotonically non-decreasing (it goes up each time a policy is included that magnifies the difference in performance possible in the two MDPs).

In Lemma 1, we show that the particular choice of $\widehat{M}$ that comes from statistically sampling transitions as in certainty equivalence leads to a bound on $|V_M^p - V_{\widehat{M}}^p|$, for an arbitrary policy p.

Lemma 1. *Given true MDP M, let $\widehat{M}$ be an MDP comprised of a reward function R and transition function $\widehat{T}$ estimated from n samples for each state–action pair, and let p be a policy, then the following holds with probability at least $1 - \delta$:*

$$|V_M^p - V_{\widehat{M}}^p| \leq \frac{2R_{\max}}{(1-\gamma)^2} \sqrt{\frac{1}{2n} \log \frac{2|S||A||\breve{\Pi}|}{\delta}}.$$

Proof. This lemma is a variation of the classic "Simulation Lemma" (Kearns & Singh, 1998; Strehl et al., 2009) and is proven in this form as Theorem 2 of Jiang et al. (2015). Note that their proof, though stated with respect to a particular choice of $\breve{\Pi}$ set, holds in this general form. □

4.2. Bound for Epsilon-Greedy Policies

It remains to show that $\|V_M^\pi - V_M^{\widehat{\rho}}\|_\infty$ is bounded when restricted to epsilon-greedy policies. For the case of planning with a decreased discount factor, Jiang et al. (2015) provide a bound for this quantity in their Lemma 1. For the case of epsilon-greedy policies, the corresponding bound is proven in the following lemma.

Lemma 2. *For any MDP M, the difference in value of the optimal policy π and the optimal ϵ-greedy policy ρ is bounded by:*

$$|V_M^\pi - V_M^\rho| \leq R_{\max} \frac{\epsilon}{(1-\gamma)(1-\gamma(1-\epsilon))}.$$

Proof. Let π be the optimal policy for M and u be a policy that selects actions uniformly at random. We can define π_ϵ, an ϵ-greedy version of π, as:

$$\pi_\epsilon^a(s) = (1-\epsilon)\pi^a(s) + \epsilon u^a(s). \quad (4)$$

where $\pi^a(s)$ refers to the probability associated with action a under a policy π. Let T^π denote the transition matrix from states to states under policy π. Using the above definition, we can decompose the transition matrix into

$$T^{\pi_\epsilon} = (1-\epsilon)T^\pi + \epsilon T^u. \quad (5)$$

Similarly, we have for the reward vector over states,

$$R^{\pi_\epsilon} = (1-\epsilon)R^{\pi} + \epsilon R^{u}. \tag{6}$$

To obtain our bound, note

$$\begin{aligned}
V_M^{\pi} - V_M^{\pi_\epsilon} & \\
&= \sum_{t=1}^{\infty} \gamma^{t-1} \left[T^{\pi}\right]^{t-1} R^{\pi} - \gamma^{t-1} \left[T^{\pi_\epsilon}\right]^{t-1} R^{\pi_\epsilon} \\
&= \sum_{t=1}^{\infty} \gamma^{t-1} \left[T^{\pi}\right]^{t-1} R^{\pi} \\
&\quad -\gamma^{t-1} \left[T^{\pi_\epsilon}\right]^{t-1} \left[(1-\epsilon)R^{\pi} + \epsilon R^{u}\right] \\
&= \sum_{t=1}^{\infty} \gamma^{t-1} \left[T^{\pi}\right]^{t-1} R^{\pi} \\
&\quad -\gamma^{t-1} \left[T^{\pi_\epsilon}\right]^{t-1} (1-\epsilon)R^{\pi} - \underbrace{\gamma^{t-1} \left[T^{\pi_\epsilon}\right]^{t-1} \epsilon R^{u}}_{\geq 0} \\
&\leq \sum_{t=1}^{\infty} \gamma^{t-1} \left[T^{\pi}\right]^{t-1} R^{\pi} \\
&\quad -\gamma^{t-1} \left[(1-\epsilon)T^{\pi} + \epsilon T^{u}\right]^{t-1} (1-\epsilon)R^{\pi}. \qquad (7)
\end{aligned}$$

Since T^{π} is a transition matrix, all its entries lie in $[0,1]$; hence, we have the following element-wise matrix inequality:

$$\left[(1-\epsilon)T^{\pi} + \epsilon T^{u}\right]^{t-1} \geq \left[(1-\epsilon)T^{\pi}\right]^{t-1}. \tag{8}$$

Plugging inequality 8 into the bound 7 results in

$$\begin{aligned}
V_M^{\pi} - V_M^{\pi_\epsilon} & \\
&\leq \sum_{t=1}^{\infty} \gamma^{t-1} \left[T^{\pi}\right]^{t-1} R^{\pi} - \gamma^{t-1}(1-\epsilon)^{t} \left[T^{\pi}\right]^{t-1} R^{\pi} \\
&\leq \| \sum_{t=1}^{\infty} \gamma^{t-1}(1-(1-\epsilon)^{t}) \left[T^{\pi}\right]^{t-1} R^{\pi} \|_{\infty}
\end{aligned}$$

Since $\|R^{\pi}\|_{\infty} = R_{\max}$ we can upper bound the norm of difference of the values vector over states with

$$\begin{aligned}
\|V_M^{\pi} - V_M^{\pi_\epsilon}\|_{\infty} &\leq \sum_{t=1}^{\infty} \gamma^{t-1}(1-(1-\epsilon)^{t}) R_{\max} \\
&= \frac{\epsilon}{(1-\gamma)(\epsilon\gamma - \gamma + 1)} R_{\max}.
\end{aligned}$$

Using this inequality, we can bound the difference in value of the optimal policy π and the optimal ϵ-greedy policy ρ by

$$\begin{aligned}
\|V_M^{\pi} - V_M^{\rho}\|_{\infty} &\leq \|V_M^{\pi} - V_M^{\pi_\epsilon}\|_{\infty} \\
&\leq \frac{\epsilon}{(1-\gamma)(\epsilon\gamma - \gamma + 1)} R_{\max}.
\end{aligned}$$

□

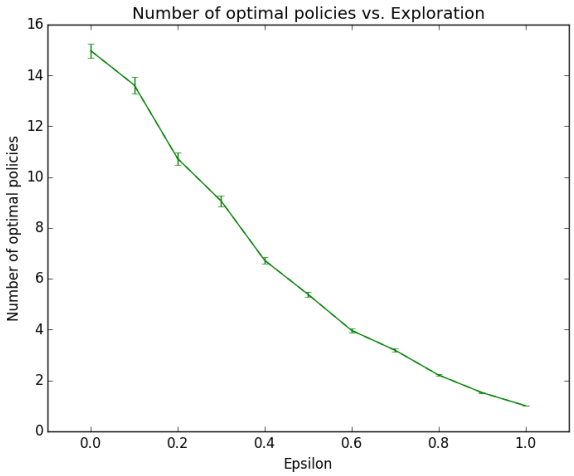

Figure 4. The number of distinct optimal policies found generating random transition functions for a fixed reward function varying ϵ.

Figure 4 is an estimate of how $|\Pi_{R,\epsilon}|$ grows over the class of randomly generated MDPs. Again, the set appears to grow gradually as ϵ decreases, making ϵ another effective parameter for fighting planner overfitting.

4.3. Empirical Results

We evaluated this exploration-based regularization approach in the distribution over MDPs used in Figure 3. Figure 5 shows results for each value of $\epsilon \in \{0.0, 0.1, 0.2, \ldots, 0.9, 1.0\}$. Here, the maximum likelihood transition function $\widehat{T}$ was replaced with the epsilon-softened transition function T_ϵ. In contrast to the previous figure, regularization increases as we go to the right. Once again, we see that intermediate values of ϵ are most successful and the best value of ϵ decreases as the model used in planning becomes more accurate (having been trained on more trajectories). The similarity to Figure 3 is striking—in spite of the difference in approach, it is essentially the mirror image of Figure 3.

We see that manipulating either $\breve{\gamma}$ or ϵ can be used to modulate the impact of planner overfitting. Which method to use in practice depends on the particular planner being used and how easily it is modified to use these methods.

5. Decreased Policy Complexity

In addition to indirectly controlling policy complexity via ϵ and γ, it is possible to control for the complexity via the representation of the policy itself. In this section, we look at varying the complexity of the policy in the context of model-based RL in which a model is learned and then a policy for

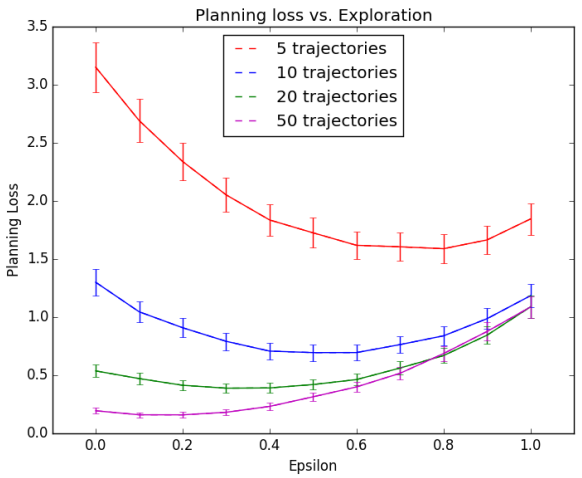

Figure 5. Increasing the randomness in action selection during planning combats planner overfitting in random MDPs.

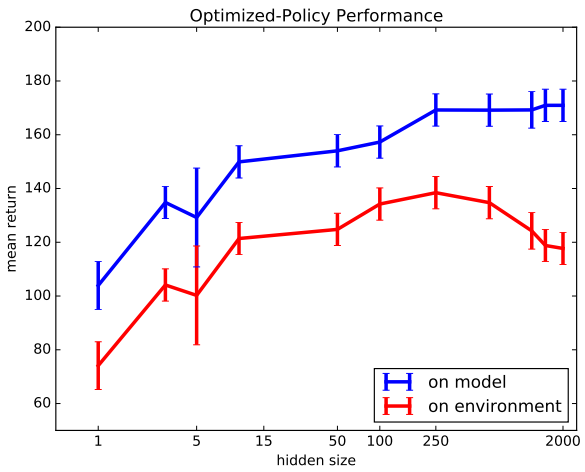

Figure 6. Decreasing the number of hidden units used to represent a policy combats planner overfitting in the Lunar Lander domain.

that model is optimized via a policy search approach. Such an approach was used in the setting of helicopter control (Ng et al., 2003) in the sense that collected data in that work was used to build a model and a policy was constructed to optimize performance in this model (via policy search, in this case) and then deployed in the environment.

Our test domain was Lunar Lander, an environment with a continuous state space and discrete actions. The goal of the environment is to control a falling spacecraft so as to land gently in a target area. It consists of 8 state variables, namely the lander's x and y coordinates, x and y velocities, angle and angular velocities, and two Boolean flags corresponding to whether each leg has touched down. The agent can take 4 actions, corresponding to which of its three thrusters (or no thruster) is active during the current time step. The Lunar Lander environment is publicly available as part of the OpenAI Gym Toolkit (Brockman et al., 2016).

We collected 40k 200-step episodes of data on Lunar Lander. During data collection, decisions were made by a policy-gradient algorithm. Specifically, we ran the REINFORCE algorithm with the state–value function as the baseline (Williams, 1992; Sutton et al., 2000). For the policy and value networks, we used a single hidden layer neural network with 16 hidden units and relu activation functions. We used the Adam algorithm (Kingma & Ba, 2014) with the default parameters and a step size of 0.005. The learned model was a 3-layer neural net with ReLU activation functions mapping the agent's state (8 inputs corresponding to 8 state variables) as well as a one-hot representation of actions (4 inputs corresponding to 4 possible actions). The model consisted of two fully connected hidden layers with 32 units each and ReLU activations. We again used Adam and used step size 0.001 to learn the model.

We then ran policy-gradient RL (REINFORCE) as a planner using the learned model. The policy was represented by a neural network with a single hidden layer. To control the complexity of the policy representation, we varied the number of units in the hidden layer from 1 to 2000. Results were averaged over 40 runs. Figure 6 shows that increasing the size of the hidden layer in the policy resulted in better and better performance on the learned model (top line). However, after 250 or so units, the resulting policy performed less well on the actual environment (bottom line). Thus, we see that reducing policy complexity serves as yet another way to reduce planner overfitting.

6. Related Work

Prior work has explored the use of regularization in reinforcement learning to mitigate overfitting. We survey some of the previous methods according to which function is regularized: (1) value, (2) model, or (3) policy.

6.1. Regularizing Value Functions

Many prior approaches have applied regularization to value function approximation, including Least Squares Temporal Difference learning (Bradtke & Barto, 1996), Policy Evaluation, and the batch approach of Fitted Q-Iteration (FQI) (Ernst et al., 2005).

Kolter & Ng (2009) applied regularization techniques to LSTD (Bradtke & Barto, 1996) with an algorithm they called LARS-TD. In particular, they argued that, without

regularization, LSTD's performance depends heavily on the number of basis functions chosen and the size of the data set collected. If the data set is too small, the technique is prone to overfitting. They showed that L_1 and L_2 regularization yield a procedure that inherits the benefits of selecting good features while making it possible to compute the fixed point. Later work by Liu et al. (2012) built on this work with the algorithm RO-TD, an L_1 regularized off policy Temporal Difference Learning method. Johns et al. (2010) cast the L_1 regularized fixed-point computation as a linear complementarity problem, which provides stronger solution-uniqueness guarantees than those provided for LARS-TD. Petrik et al. (2010) examined the approximate linear programming (ALP) framework for finding approximated value functions in large MDPs. They showed the benefits of adding an L_1 regularization constraint to the ALP that increases the error bound at training time and helps fight overfitting.

Farahmand et al. (2008a) and Farahmand et al. (2009) focused on regularization applied to Policy Iteration and Fitted Q-Iteration (FQI) (Ernst et al., 2005) and developed two related methods for Regularized Policy Iteration, each leveraging L_2 regularization during the evaluation of policies for each iteration. The first method adds a regularization term to the Least Squares Temporal Difference (LSTD) error (Bradtke & Barto, 1996), while the second adds a similar term to the optimization of Bellman residual minimization (Baird et al., 1995; Schweitzer & Seidmann, 1985; Williams & Baird, 1993) with regularization (Loth et al., 2007). Their main result shows finite convergence for the Q function under the approximated policy and the true optimal policy. A method for FQI adds a regularization cost to the least squares regression of the Q function. Follow up work (Farahmand et al., 2008b) expanded Regularized Fitted Q-Iteration to planning. That is, given a data set $\mathcal{D} = \langle (s_1, a_1, r_1, s'_1), \ldots, (s_m, a_m, r_m, s'_m) \rangle$ and a function family $\mathcal{F}$ (like regression trees), FQI approximates a Q function through repeated iterations of the following regression problem:

$$\begin{aligned}\widehat{Q}_{t+1} \\ = \quad & \operatorname*{argmin}_{Q \in F} \sum_{i=1}^{m} \left[r_i + \gamma \max_{a' \in A} \widehat{Q}_t(s_i, a) - Q(s'_i, a_i) \right]^2 \\ & + \lambda \text{Pen}(\widehat{Q}),\end{aligned}$$

where $\lambda\text{Pen}(\widehat{Q})$ imposes a regularization penalty term and λ is a regularization coefficient. They prove bounds relating this regularization cost to the approximation error in $\widehat{Q}$ between iterations of FQI.

Farahmand & Szepesvári (2011) and Farahmand (2011) focused on a problem relevant to our approach—regularization for Q value selection in RL and planning. They considered an offline setting in which an algorithm, given a data set of experiences and set of possible Q functions, must choose a Q function from the set that minimizes the true Bellman error. They provided a general complexity regularization bound for model selection, which they applied to bound the approximation error for the Q function chosen by their proposed algorithm, BERMIN.

6.2. Regularizing Models

In model-based RL, regularizaion can be used to improve estimates of R and T when data is finite or limited.

Taylor & Parr (2009) investigated the relationship between Kernelized LSTD Xu et al. (2005) and other related techniques, with a focus on regularization in model-based RL. Most relevant to our work is their decomposition of the Bellman error into transition and reward error, which they empirically show offers insight into the choice of regularization parameters.

Bartlett & Tewari (2009) developed an algorithm, REGAL, with optimal regret for weakly communicating MDPs. REGAL heavily relies on regularization; based on all prior experience, the algorithm continually updates a set $\mathcal{M}$ that, with high probability, contains the true MDP. Letting $\lambda^*(M)$ denote the optimal per-step reward of the MDP M, the traditional optimistic exploration tactic would suggest that the agent should choose the M' in $\mathcal{M}$ with maximal $\lambda^*(M')$. REGAL *also* includes a regularization term to this maximization to prevent overfitting based on the experiences so far, resulting in state-of-the-art regret bounds.

6.3. Regularizing Policies

The focus of applying regularization to policies is to limit the complexity of the policy class being searched in the planning process. It is this approach that we adopt in the present paper.

Somani et al. (2013) explored how regularization can help online planning for Partially Observable Markov Decision Processes (POMDPs). They introduced the DESPOT algorithm (Determinized Sparse Partially Observable Tree), which constructs a tree that models the execution of all policies on a number of sampled scenarios (rollouts). However, the authors note that DESPOT typically succumbs to overfitting, as a policy that performs well on the sampled scenarios is not likely to perform well in general. The work proposes a regularized extension of DESPOT, R-DESPOT, where regularization takes the form of balancing between the performance of the policy on the samples with the complexity of the policy class. Specifically, R-DESPOT imposes a regularization penalty on the utility of each node in the belief tree. The algorithm then computes the policy that maximizes regularized utility for the tree using a bottom up dynamic programming procedure on the tree. The approach

is similar to ours in that it also limits policy complexity through regularization, but focuses on regularizing utility instead of regularizing the use of a transition model. Investigating the interplay between these two approaches poses an interesting direction for future work. In a similar vein, Thomas et al. (2015) developed a batch RL algorithm with a probabilistic performance guarantee that limits the complexity of the policy class as a means of regularization.

Petrik & Scherrer (2008) conducted analysis similar to Jiang et al. (2015). Specifically, they investigated the situations in which using a lower-than-actual discount factor can improve solution quality given an approximate model, noting that this procedure has the effect of regularizing rewards. The work also advanced the first bounds on the error of using a smaller discount factor.

7. Conclusion

For three different regularization methods—decreased discounting, increased exploration, and decreased policy complexity, we found a consistent U-shaped tradeoff between the size of the policy class being searched and its performance on a learned model. Future work will evaluate other methods such as drop out and early stopping.

The plots that varied ϵ and γ were quite similar, raising the possibility that perhaps epsilon-greedy action selection is functioning as another way to decrease the effective horizon depth using in planning—chaining together random actions makes future states less predictable and therefore carry less weight. Later work can examine whether jointly choosing ϵ and γ is more effective than setting only one at a time.

More work is needed to identify methods that can learn in much larger domains (Bellemare et al., 2013). One concept worth considering is adapting regularization non-uniformly to the state space. That is, it should be possible to modulate the complexity of policies considered in parts of the state space where the model is more accurate, allowing more expressive plans is some places than others.

References

Baird, Leemon et al. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the twelfth international conference on machine learning*, pp. 30–37, 1995.

Bartlett, Peter L. and Tewari, A. REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 35–42, 2009.

Bellemare, Marc G., Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bradtke, Steven J and Barto, Andrew G. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

Brafman, Ronen I. and Tennenholtz, Moshe. R-MAX—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym, 2016.

Dayan, Peter and Sejnowski, Terrence J. Exploration bonuses and dual control. *Machine Learning*, 25:5–22, 1996.

Ernst, Damien, Geurts, Pierre, and Wehenkel, Louis. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Farahmand, Amir-massoud. *Regularization in reinforcement learning*. PhD thesis, University of Alberta, 2011.

Farahmand, Amir-massoud and Szepesvári, Csaba. Model selection in reinforcement learning. *Machine Learning*, 85:299–332, 2011.

Farahmand, Amir Massoud, Ghavamzadeh, Mohammad, Szepesvári, Csaba, and Mannor, Shie. Regularized Policy Iteration. *Nips*, pp. 441–448, 2008a.

Farahmand, Amir Massoud, Ghavamzadeh, Mohammad, Szepesvári, Csaba, and Mannor, Shie. Regularized fitted Q-Iteration: Application to planning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5323 LNAI:55–68, 2008b.

Farahmand, Amir Massoud, Ghavamzadeh, Mohammad, Szepesvári, Csaba, and Mannor, Shie. Regularized fitted q-iteration for planning in continuous-space markovian decision problems. *Proceedings of the American Control Conference*, pp. 725–730, 2009.

Hessel, Matteo, Modayil, Joseph, van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, Horgan, Dan, Piot, Bilal, Azar, Mohammad Gheshlaghi, and Silver, David. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

Jiang, Nan, Kulesza, Alex, Singh, Satinder, and Lewis, Richard. The dependence of effective planning horizon on model accuracy. In *Proceedings of AAMAS*, pp. 1181–1189, 2015.

Johns, J, Painter-Wakefield, C, and Parr, R. Linear complementarity for regularized policy evaluation and improvement. *Advances in neural information processing systems*, 23:1009–1017, 2010.

Kearns, Michael and Singh, Satinder. Near-optimal reinforcement learning in polynomial time. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 260–268, 1998. URL citeseer.nj.nec.com/kearns98nearoptimal.html.

Kingma, Diederik P. and Ba, Jimmy Lei. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

Kolter, J. Zico and Ng, Andrew Y. Regularization and feature selection in least-squares temporal difference learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, 94305:1–8, 2009.

Littman, Michael L. and Szepesvári, Csaba. A generalized reinforcement-learning model: Convergence and applications. In Saitta, Lorenza (ed.), *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 310–318, 1996.

Liu, Bo, Mahadevan, Sridhar, and Liu, Ji. Regularized off-policy td-learning. In *Advances in Neural Information Processing Systems*, pp. 836–844, 2012.

Loth, Manuel, Davy, Manuel, and Preux, Philippe. Sparse temporal difference learning using lasso. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 352–359. IEEE, 2007.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A., Veness, Joel, Bellemare, Marc G., Graves, Alex, Riedmiller, Martin A., Fidjeland, Andreas, Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Mnih, Volodymyr, Badia, Adrià Puigdomènech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy P., Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

Ng, Andrew Y., Kim, H. Jin, Jordan, Michael I., and Sastry, Shankar. Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 2003.

Petrik, Marek and Scherrer, Bruno. Biasing approximate dynamic programming with a lower discount factor. *Advances in Neural Information Processing Systems (NIPS)*, 1:1–8, 2008.

Petrik, Marek, Taylor, Gavin, Parr, Ron, and Zilberstein, Shlomo. Feature selection using regularization in approximate linear programs for markov decision processes. *arXiv preprint arXiv:1005.1860*, 2010.

Puterman, Martin L. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Rummery, G. A. *Problem solving with reinforcement learning*. PhD thesis, Cambridge University Engineering Department, 1994.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Schweitzer, Paul J and Seidmann, Abraham. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

Somani, A, Ye, Nan, Hsu, D, and Lee, Ws. DESPOT : Online POMDP Planning with Regularization. *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

Strehl, Alexander L., Li, Lihong, and Littman, Michael L. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Sutton, Richard S., Precup, Doina, and Singh, Satinder. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

Sutton, Richard S., McAllester, David, Singh, Satinder, and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pp. 1057–1063, 2000.

Taylor, Gavin and Parr, Ronald. Kernelized value function approximation for reinforcement learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 1–8, 2009.

Thomas, Philip, Theocharous, Georgios, and Ghavamzadeh, Mohammad. High Confidence Policy Improvement. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2380–2388, 2015.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Williams, Ronald J and Baird, Leemon C. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Citeseer, 1993.

Xu, Xin, Xie, Tao, Hu, Dewen, and Lu, Xicheng. Kernel least-squares temporal difference learning. *International Journal of Information Technology*, 11(9):54–63, 2005.