MOPO: Model-based Offline Policy Optimization

Tianhe Yu [*1] **Garrett Thomas** [*1] **Lantao Yu** [1] **Stefano Ermon** [1] **James Zou** [1] **Sergey Levine** [2] **Chelsea Finn** [†1] **Tengyu Ma** [†1]

Abstract

Offline reinforcement learning (RL) refers to the problem of learning policies entirely from a batch of previously collected data. Despite significant progress in model-free offline RL, the most successful prior methods constrain the policy to the support of the data, precluding generalization to new states and actions. In this paper, we propose a model-based approach that enables some degree of generalization beyond the behavioral distribution. To limit exploitation of errors in the model, we penalize the reward function by a measure of model uncertainty, in such a way that the expected return in the penalized MDP is a lower bound of the return in the true MDP. Based on our theoretical results, we develop a practical model-based offline RL algorithm that outperforms both standard model-based RL methods and state-of-the-art model-free offline RL approaches on existing offline RL benchmarks, as well as two challenging continuous control tasks that require generalizing from data collected for a different task.

1. Introduction

Recent advances in machine learning using deep neural networks have shown significant successes in scaling to large datasets, such as ImageNet (Deng et al., 2009) in computer vision, SQuAD (Rajpurkar et al., 2016) in NLP, and RoboNet (Dasari et al., 2019) in robot learning. Reinforcement learning (RL) methods, in contrast, struggle to scale to many real-world applications, e.g., autonomous driving (Yu et al., 2018) and healthcare (Gottesman et al., 2019), because they rely on costly online trial-and-error. Designing RL algorithms that can learn from diverse, static datasets would enable more practical RL training in the real world.

[*]Equal contribution [†]Equal advising [1]Stanford University [2]UC Berkeley. Correspondence to: Tianhe Yu <tianheyu@cs.stanford.edu>, Garrett Thomas <gwthomas@stanford.edu>.

Accepted at Workshop on Inductive Biases, Invariances and Generalization, International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

While off-policy RL algorithms (Lillicrap et al., 2015; Haarnoja et al., 2018; Fujimoto et al., 2018b) can in principle utilize previously collected datasets, they perform poorly without online data collection. These failures are generally caused by large extrapolation error when the Q-function is evaluated on out-of-distribution actions (Fujimoto et al., 2018a; Kumar et al., 2019), which can lead to unstable learning and divergence. Offline RL methods propose to mitigate bootstrapped error by constraining the learned policy to the behavior policy induced by the dataset (Fujimoto et al., 2018a; Kumar et al., 2019; Wu et al., 2019; Jaques et al., 2019; Nachum et al., 2019; Peng et al., 2019; Siegel et al., 2020). While these methods achieve reasonable performances in some settings, their learning is limited to behaviors within the data manifold. We argue that it is important for an offline RL algorithm to be equipped with the ability to leave the data support to learn a better policy for two reasons: (1) the provided batch dataset is usually sub-optimal in terms of both the states and actions covered by the dataset, and (2) the target task can be different from the tasks performed in the batch data for various reasons, e.g., because data is not available or hard to collect for the target task.

We hypothesize that model-based RL methods (Sutton, 1991; Deisenroth & Rasmussen, 2011; Levine & Koltun, 2013; Kumar et al., 2016; Janner et al., 2019; Luo et al., 2018) make a natural choice for enabling generalization, for a number of reasons. First, model-based RL algorithms effectively receive more supervision, since the model is trained on every transition, even in sparse-reward settings. Second, they are trained with supervised learning, which provides more stable and less noisy gradients than bootstrapping. Lastly, uncertainty estimation techniques, such as bootstrap ensembles, are well developed for supervised learning methods (Lakshminarayanan et al., 2017; Kuleshov et al., 2018; Snoek et al., 2019) and are known to perform poorly for value-based RL methods (Wu et al., 2019). All of these attributes have the potential to improve or control generalization.

However, because offline model-based algorithms cannot improve the dynamics model using additional experience, we expect that such algorithms require careful use of the model in regions outside of the data support. To do so, we modify the model MDP via a *reward penalty* based on an

estimate of the model error. Crucially, this estimate is model-dependent, and does not necessarily penalize all out-of-distribution states and actions equally, but rather prescribes penalties based on the estimated magnitude of model error. Further, this estimation is done both on *states* and *actions*, allowing generalization to both, in contrast to model-free approaches that only reason about uncertainty with respect to actions.

The primary contribution of this work is an offline model-based RL algorithm that optimizes a policy in an uncertainty-penalized MDP, where the reward function is penalized by an estimate of the model's error, yielding a lower bound of the return in the true MDP. We present a practical algorithm that estimates model error using the predicted variance of a Lipschitz-regularized model. We empirically compare this approach, model-based offline policy optimization (MOPO), to existing state-of-the-art model-free offline RL algorithms. Our results suggest that MOPO substantially outperforms these prior methods on the offline RL benchmark D4RL (Fu et al., 2020) as well as on offline RL problems where the agent must generalize to out-of-distribution states in order to succeed.

2. Preliminaries

We consider a standard Markov decision process (MDP) $M = (\mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state space and action space respectively, $T(s' \mid s, a)$ the transition dynamics, $r(s, a)$ the reward function, μ_0 the initial state distribution, and $\gamma \in (0, 1)$ the discount factor. The goal in RL is to optimize a policy $\pi(a \mid s)$ that maximizes the expected discounted return $\eta_M(\pi) := \mathop{\mathbb{E}}_{\pi, T, \mu_0}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. The value function $V_M^\pi(s) := \mathop{\mathbb{E}}_{\pi, T}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s]$ gives the expected discounted return under π if starting from state s.

In the model-based approach we will have a dynamics model $\widehat{T}$ estimated from the transitions in the static dataset $\mathcal{D}_{\text{env}} = \{(s, a, r, s')\}$. This *estimated dynamics* defines a *model MDP* $\widehat{M} = (\mathcal{S}, \mathcal{A}, \widehat{T}, r, \mu_0, \gamma)$. Let $\mathbb{P}^\pi_{\widehat{T}, t}(s)$ denote the probability of being in state s at time step t if actions are sampled according to π and transitions according to $\widehat{T}$. Let $\rho^\pi_{\widehat{T}}(s)$ be the discounted state distribution of policy π under dynamics $\widehat{T}$: $\rho^\pi_{\widehat{T}}(s) := \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi_{\widehat{T}, t}(s)$. We also define (abusing notation) the discounted state-action distribution $\rho^\pi_{\widehat{T}}(s, a) := \rho^\pi_{\widehat{T}}(s)\pi(a \mid s)$. Note that $\rho^\pi_{\widehat{T}}$, as defined here, is not a properly normalized probability distribution, as it integrates to $1/(1-\gamma)$. We will denote (improper) expectations with respect to $\rho^\pi_{\widehat{T}}$ with $\bar{\mathbb{E}}$, as in $\eta_{\widehat{M}}(\pi) = \bar{\mathbb{E}}_{\rho^\pi_{\widehat{T}}}[r(s, a)]$.

We summarize model-based policy optimization (MBPO) (Janner et al., 2019), which we build on in this work, in Appendix D.

3. MOPO: Model-Based Offline Policy Optimization

We now present our approach to model-based offline RL. We first bound the true return from below by the return of a constructed model MDP penalized by the uncertainty of the dynamics (Section 3.1). Then we maximize the conservative estimation of the return by an off-the-shelf reinforcement learning algorithm, which gives MOPO, a generic model-based off-policy algorithm (Section 3.2). We discuss important practical implementation details in Section 3.3.

3.1. Quantifying the uncertainty: from the dynamics to the total return

Our key idea is to build a lower bound for the return of a policy π under the true dynamics and then maximize the lower bound over π. A natural estimator for the true return $\eta_M(\pi)$ is $\eta_{\widehat{M}}(\pi)$, the return under the estimated dynamics. The error of this estimator depends on, potentially in a complex fashion, the error of $\widehat{T}$, which may compound over time. In this subsection, we characterize how the error of $\widehat{T}$ influences the uncertainty of the total return. We will state a lemma that bounds the difference between the performance of a policy under dynamics T and dynamics $\widehat{T}$. All proofs are given in Appendix B. We first require some assumptions.

Assumption 3.1. Assume a scalar $c \geq 0$ and a function class $\mathcal{F}$ such that $V_M^\pi \in c\mathcal{F}$ for all π.

In our analysis, the integral probability metric (IPM) (Müller, 1997) associated with $\mathcal{F}$, defined as $d_{\mathcal{F}}(P, Q) := \sup_{f \in \mathcal{F}} \left|\mathbb{E}_{X \sim P}[f(X)] - \mathbb{E}_{Y \sim Q}[f(Y)]\right|$, relates the error of the model with the error of the return. Although Assumption 3.1 is somewhat abstract, it typically holds in practice: if we assume that the reward function is bounded by $r_{\max}$, then V_M^π is bounded by $r_{\max}/(1-\gamma)$ for any π, so Assumption 3.1 holds with $c = r_{\max}/(1-\gamma)$ and $\mathcal{F} = \{f : \|f\|_\infty \leq 1\}$. In this case $d_{\mathcal{F}}$ is the *total variation distance*. We note that with stronger assumptions on the MDP, one can also obtain bounds in terms of 1-Wasserstein distance or maximum mean discrepancy.

Assumption 3.2. We assume a function $u : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which is an *admissible error estimator* for $\widehat{T}$, meaning that $d_{\mathcal{F}}(\widehat{T}(s, a), T(s, a)) \leq u(s, a)$ for all s, a.

Lemma 3.3. *Let M and $\widehat{M}$ be two MDPs with reward function r but different dynamics T and $\widehat{T}$ respectively. Then, under Assumptions 3.1 and 3.2, letting $\lambda := c\gamma$ we have*

$$|\eta_{\widehat{M}}(\pi) - \eta_M(\pi)| \leq \lambda \mathop{\bar{\mathbb{E}}}_{(s,a) \sim \rho^\pi_{\widehat{T}}}[u(s, a)] \tag{1}$$

Thus,

$$\eta_M(\pi) \geq \mathop{\bar{\mathbb{E}}}_{(s,a) \sim \rho^\pi_{\widehat{T}}}[r(s, a) - \lambda u(s, a)] \tag{2}$$

Algorithm 1 Framework for Model-based Offline Policy Optimization (MOPO) with Reward Penalty

Require: Dynamics model $\widehat{T}$ with admissible error estimator $u(s,a)$; constant λ.

1: Define $\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$. Let $\widetilde{M}$ be the MDP with dynamics $\widehat{T}$ and reward $\tilde{r}$.
2: Run any RL algorithm on $\widetilde{M}$ until convergence to obtain

$$\hat{\pi} = \text{argmax}_{\pi} \eta_{\widetilde{M}}(\pi) \tag{4}$$

Equation (2) suggests that a policy that obtains high reward in the estimated MDP while also minimizing model error will obtain high reward in the real MDP. Motivated by Lemma 3.1, we define the *uncertainty-penalized reward* $\tilde{r}(s,a) := r(s,a) - \lambda u(s,a)$, and the *uncertainty-penalized MDP* $\widetilde{M} := (\mathcal{S}, \mathcal{A}, \widehat{T}, \tilde{r}, \mu_0, \gamma)$. We observe that $\widetilde{M}$ is conservative in that the return under it bounds from below the true return:

$$\eta_M(\pi) \geq \bar{\mathbb{E}}_{(s,a)\sim\rho^{\pi}_{\widehat{T}}}[\tilde{r}(s,a)] = \eta_{\widetilde{M}}(\pi) \tag{3}$$

3.2. Policy optimization on the penalized MDP

Motivated by (3), we optimize the policy on the uncertainty-penalized MDP $\widetilde{M}$ in Algorithm 1. We establish a lower bound on the performance of the policy returned by the algorithm.

Theorem 3.4. *Suppose Assumption 3.1 and 3.2 hold, and define* $\epsilon_u(\pi) := \bar{\mathbb{E}}_{\rho^{\pi}_{\widehat{T}}}[u(s,a)]$. *Then the learned policy* $\hat{\pi}$ *in MOPO (Algorithm 1) satisfies*

$$\eta_M(\hat{\pi}) \geq \sup_{\pi}\{\eta_M(\pi) - 2\lambda\epsilon_u(\pi)\} \tag{5}$$

The quantity $\epsilon_u(\pi)$ characterizes how erroneous the model is along trajectories induced by π. Crucially, it depends on the quality of the model and the uncertainty quantification, rather than an explicit divergence between π and the behavioral distribution. We provide commentary on the theorem in Appendix C.

3.3. Practical implementation

The method is summarized in Algorithm 2 in Appendix E, and largely follows MBPO with a few key exceptions. We model the dynamics using a neural network that outputs a Gaussian distribution over the next state and reward: $\widehat{T}_{\theta,\phi}(s_{t+1}, r | s_t, a_t) = \mathcal{N}(\mu_\theta(s_t,a_t), \Sigma_\phi(s_t,a_t))$. We learn an ensemble of N dynamics models $\{\widehat{T}^i_{\theta,\phi} = \mathcal{N}(\mu^i_\theta, \Sigma^i_\phi)\}_{i=1}^N$. Unlike standard MBPO, the model means μ^i_θ are Lipschitz-regularized. This is shown to be helpful in various settings in Appendix G. We implement this regularization using spectral normalization (Miyato et al., 2018); for details, see Appendix F.

In this work we take $u(s,a) = \max_{i=1}^N \|\Sigma^i_\phi(s,a)\|_{\text{F}}$. The learned variance of a Gaussian probabilistic model can theoretically recover the true aleatoric uncertainty when the model is well-specified, and taking the maximum over the ensemble causes u be more conservative. While this estimator lacks theoretical guarantees, we find that it is sufficiently accurate to achieve good performance in practice. Hence the practical uncertainty-penalized reward of MOPO is computed as $\tilde{r}(s,a) = \hat{r}(s,a) - \lambda \max_{i=1,\ldots,N} \|\Sigma^i_\phi(s,a)\|_{\text{F}}$, where $\hat{r}$ is the mean of the predicted reward output by $\widehat{T}$.

We treat the penalty coefficient λ as a user-chosen hyperparameter. Since we do not have a true admissible error estimator, the value of λ prescribed by the theory may not be an optimal choice in practice; it should be larger if our heuristic $u(s,a)$ underestimates the true error and smaller if u substantially overestimates the true error.

4. Experiments

In our experiments, we aim to study the follow questions: (1) How does MOPO perform on standard offline RL benchmarks in comparison to prior state-of-the-art approaches? (2) Can MOPO solve tasks that require generalization to out-of-distribution behaviors? (3) How does each component in MOPO affect performance?

To study (2), we construct two additional continuous control tasks that demand out-of-distribution generalization, as described in Section 4.2. To answer question (3), we conduct a complete ablation study to analyze the effect of each module in MOPO in Appendix G. For more details on the experimental setup and hyperparameters, see Appendix H.

We compare against several baselines, including MBPO and the current state-of-the-art model-free offline RL algorithms. BEAR aims to constrain the policy's actions to lie in the support of the behavioral distribution (Kumar et al., 2019). BRAC is a family of algorithms that penalize the value function by some measure of discrepancy between the actor and the behavioral policy (Wu et al., 2019). BRAC-v uses this penalty both when updating the critic and when updating the actor, while BRAC-p uses this penalty only when updating the actor and does not explicitly penalize the critic.

4.1. Evaluation on the D4RL benchmark

To answer question (1), we evaluate our method on a large subset of datasets in the D4RL benchmark[1] (Fu et al., 2020), including three environments (halfcheetah, hopper, and

[1] https://sites.google.com/view/d4rl

Dataset type	Environment	Batch Mean	Batch Max	MOPO (ours)	MBPO	SAC	BEAR	BRAC-v
random	halfcheetah	-303.2	-0.1	**3679.8** ± 70.7	3533.0 ± 201.8	3502.0	2885.6	3207.3
random	hopper	299.26	365.9	**412.8** ± 30.7	126.6 ± 173.9	347.7	289.5	370.5
random	walker2d	0.9	57.3	**596.3** ± 121.8	395.9 ± 371.7	192.0	307.6	23.9
medium	halfcheetah	3953.0	4410.7	4706.9 ± 61.1	3230.0 ± 2543.6	-808.6	4508.7	**5365.3**
medium	hopper	1021.7	3254.3	840.9 ± 99.3	137.8 ± 87.5	5.7	**1527.9**	1030.0
medium	walker2d	498.4	3752.7	645.5 ± 464.8	582.6 ± 348.8	44.2	1526.7	**3734.3**
mixed	halfcheetah	2300.6	4834.2	**6418.3** ± 47.4	5598.4 ± 1285.1	-581.3	4211.3	5413.8
mixed	hopper	470.5	1377.9	**2988.7** ± 186.3	1599.2 ± 969.6	93.3	802.7	5.3
mixed	walker2d	358.4	1956.5	**1963.5** ± 383.8	1021.8 ± 585.8	87.8	495.3	44.5
med-expert	halfcheetah	8074.9	12940.2	**6913.5** ± 2793.0	929.6 ± 903.2	-55.7	6132.5	5342.4
med-expert	hopper	1850.5	3760.5	1663.5 ± 1375.6	**1803.6** ± 1102.4	32.9	109.8	5.1
med-expert	walker2d	1062.3	5408.6	2527.1 ± 879.8	351.7 ± 170.6	-5.1	1193.6	**3058.0**

Table 1. Results for D4RL datasets. Each number is the average undiscounted return of the policy at the last iteration of training, averaged over 3 random seeds, ± standard deviation. Numbers for model-free methods taken from (Fu et al., 2020), which does not report standard deviations. We omit BRAC-p in this table for space because BRAC-v outperforms it on these tasks.

Environment	Batch Mean	Batch Max	MOPO (ours)	MBPO	SAC	BEAR	BRAC-p	BRAC-v
halfcheetah-jump	-1022.6	1808.6	**4140.6±88.6**	2971.4±1262.1	-3588.2±1436.3	16.8±59.5	1069.9±232.0	871±41.4
ant-angle	866.7	2311.9	**2502.2±15.0**	13.6±65.5	-966.4±777.5	1658.2±16.0	1806.7±265.0	2333±138.9

Table 2. Average returns `halfcheetah-jump` and `ant-angle` that require out-of-distribution policy. The results are averaged over 3 random seeds. Our method outperforms all the baselines by a large margin, indicating that our approach is effective in generalizing to out-of-distribution states where model-free offline RL methods struggle.

walker2d) and four dataset types (random, medium, mixed, medium-expert), yielding a total of 12 problem settings. The datasets in this benchmark have been generated as follows: **random**: roll out a randomly initialized policy for 1M steps. **medium**: partially train a policy using SAC, then roll it out for 1M steps. **mixed**: train a policy using SAC until a certain (environment-specific) performance threshold is reached, and take the replay buffer as the batch. **medium-expert**: combine 1M samples of rollouts from a fully-trained policy with another 1M samples of rollouts from a partially trained policy or a random policy.

Results are given in Table 1. Our method is the strongest by a significant margin on all the mixed datasets and most of the medium-expert datasets, while also achieving the best performance on all of the random datasets. Our model-based approach performs less well on the medium datasets. We hypothesize that the lack of action diversity in the medium datasets make it more difficult to learn a model that generalizes well. Fortunately, this setting is one in which model-free methods can perform well, suggesting that model-based and model-free approaches are able to perform well in complementary settings.

4.2. Evaluation on tasks requiring out-of-distribution generalization

To answer question (2), we construct two environments `halfcheetah-jump` and `ant-angle` where the agent must solve a task that is different from the purpose of the behavioral policy. The trajectories of the batch data in the these datasets are from policies trained for the original dynamics and reward functions `HalfCheetah` and `Ant` in OpenAI Gym (Brockman et al., 2016) which incentivize the cheetah and ant to move forward as fast as possible. Note that for `HalfCheetah`, we set the maximum velocity to be 3. Concretely, we train SAC for 1M steps and use the entire training replay buffer as the trajectories for the batch data. Then, we assign these trajectories with new rewards that incentivize the cheetah to jump and the ant to run towards the top right corner with a 30 degree angle. Thus, to achieve good performance for the new reward functions, the policy needs to leave the observational distribution. We include the exact forms of the new reward functions in Appendix H. In these environments, learning the correct behaviors requires leaving the support of the data distribution; optimizing solely within the data manifold will lead to sub-optimal policies.

In Table 2, we show that MOPO significantly outperforms the state-of-the-art model-free approaches. In particular, model-free offline RL cannot outperform the best trajectory in the batch dataset, whereas MOPO exceeds the batch max by a significant margin. This validates that MOPO is able to generalize to out-of-distribution behaviors while existing model-free methods are unable to solve those challenges.

5. Conclusion

We develop an algorithm, model-based offline policy optimization (MOPO), based on a modified MDP that penalizes states and actions according to model uncertainty. MOPO is theoretically motivated and trades off the risk of making mistakes and the benefit of diverse exploration from escaping the behavioral distribution. In our experiments, MOPO outperforms state-of-the-art offline RL methods in both standard benchmarks (Fu et al., 2020) and out-of-distribution generalization environments.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., Singh, S., Levine, S., and Finn, C. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018a.

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018b.

Gottesman, O., Johansson, F., Komorowski, M., Faisal, A., Sontag, D., Doshi-Velez, F., and Celi, L. A. Guidelines for reinforcement learning in healthcare. *Nat Med*, 25(1): 16–18, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pp. 12498–12509, 2019.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Kuleshov, V., Fenner, N., and Ermon, S. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pp. 11761–11771, 2019.

Kumar, V., Todorov, E., and Levine, S. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 378–383. IEEE, 2016.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pp. 6402–6413, 2017.

Levine, S. and Koltun, V. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Luo, Y., Xu, H., Li, Y., Tian, Y., Darrell, T., and Ma, T. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Müller, A. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29 (2):429–443, 1997.

Nachum, O., Dai, B., Kostrikov, I., Chow, Y., Li, L., and Schuurmans, D. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Siegel, N. Y., Springenberg, J. T., Berkenkamp, F., Abdolmaleki, A., Neunert, M., Lampe, T., Hafner, R., and Riedmiller, M. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado,

Z. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pp. 13969–13980, 2019.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Wei, C. and Ma, T. Data-dependent sample complexity of deep neural networks via lipschitz augmentation, 2019a.

Wei, C. and Ma, T. Improved sample complexities for deep networks and robust classification via an all-layer margin. *arXiv preprint arXiv:1910.04284*, 2019b.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., and Darrell, T. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018.