SCIENCE CHINA
Information Sciences

• REVIEW •

February 2024, Vol. 67, Iss. 2, 121101:1–121101:26
https://doi.org/10.1007/s11432-022-3696-5

A survey on model-based reinforcement learning

Fan-Ming LUO[1,3], Tian XU[1], Hang LAI[2], Xiong-Hui CHEN[1,3], Weinan ZHANG[2*] & Yang YU[1,3*]

[1]*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China;*
[2]*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[3]*Polixir.ai, Nanjing* 211106*, China*

Received 19 June 2022/Revised 23 September 2022/Accepted 31 January 2023/Published online 23 January 2024

Abstract Reinforcement learning (RL) interacts with the environment to solve sequential decision-making problems via a trial-and-error approach. Errors are always undesirable in real-world applications, even though RL excels at playing complex video games that permit several trial-and-error attempts. To improve sample efficiency and thus reduce errors, model-based reinforcement learning (MBRL) is believed to be a promising direction, as it constructs environment models in which trial-and-errors can occur without incurring actual costs. In this survey, we investigate MBRL with a particular focus on the recent advancements in deep RL. There is a generalization error between the learned model of a non-tabular environment and the actual environment. Consequently, it is crucial to analyze the disparity between policy training in the environment model and that in the actual environment, guiding algorithm design for improved model learning, model utilization, and policy training. In addition, we discuss the recent developments of model-based techniques in other forms of RL, such as offline RL, goal-conditioned RL, multi-agent RL, and meta-RL. Furthermore, we discuss the applicability and benefits of MBRL for real-world tasks. Finally, this survey concludes with a discussion of the promising future development prospects for MBRL. We believe that MBRL has great unrealized potential and benefits in real-world applications, and we hope this survey will encourage additional research on MBRL.

Keywords reinforcement learning, model-based reinforcement learning, planning, model learning, model learning with reduced error, model usage

1 Overview of model-based RL

Reinforcement learning (RL) is the study of methods for enhancing the sequential decision-making performance of autonomous agents [1]. Since the success of deep RL in playing Go [2] and video games [3] demonstrates a decision-making ability beyond that of humans, it is of great interest to expand its application to include real-world tasks.

Deep RL algorithms typically require many training samples, resulting in extremely high sample complexity. In general RL tasks, the sample complexity of an algorithm refers to the number of samples required to learn an approximately optimal policy. Unlike the supervised learning paradigm, which learns from labeled historical data, typical RL algorithms require interaction data by executing the most recent policy in the environment. Once the policy has been updated, the underlying data distribution (previously the occupancy measure [4]) changes, and the data must be recollected by rerunning the policy. Therefore, applying RL algorithms with high sample complexity to real-world tasks is difficult, where trial-and-error can be extremely expensive.

Therefore, a major focus of recent deep RL (DRL) research has been on enhancing sample efficiency [5]. Model-based RL (MBRL) is one of the most important research directions that have the potential to make RL algorithms significantly more sample efficient [6]. This belief is based on an intuitive analogy to human intelligence. Humans can imagine a world in which it is possible to predict the outcomes of various actions. In this manner, appropriate actions can be chosen based on imagination, with minimal costs associated

* Corresponding author (email: wnzhang@sjtu.edu.cn, yuy@nju.edu.cn)

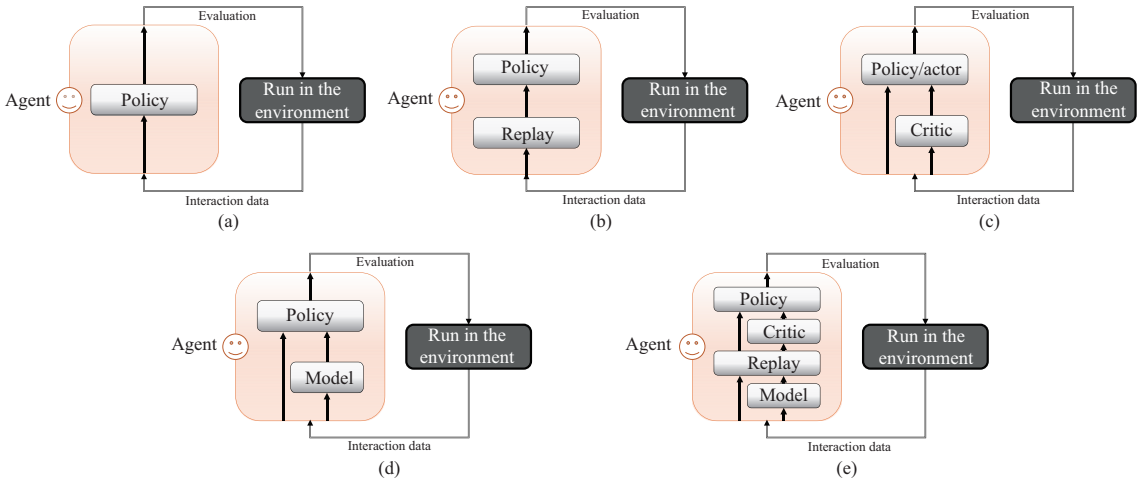

Figure 1 (Color online) Architectures of RL algorithms. The figures show a training iteration of the RL, focusing on how the interaction data are utilized. (a) On-policy RL; (b) off-policy RL; (c) actor-critic RL; (d) model-based RL; (e) off-policy actor-critic model-based RL.

with trial and error. In MBRL, the term model refers to the environment model expected to perform the same function as the imagination.

The environment model (or model) is the abstraction of the environment dynamics with which the learning agent interacts in MBRL. The dynamics environment in RL is typically formulated as a Markov decision process (MDP), denoted by the tuple $\langle S, A, M, R, \gamma \rangle$, where S, A, and γ represent the state space, action space, and discount factor for future rewards, respectively, while $M : S \times A \mapsto S$ represents the state transition dynamics and $R : S \times A \mapsto \mathbb{R}$ denotes the reward function. Given the state and action spaces and the discount factor, the state transition dynamics and the reward function are typically the most important aspects of the environment model. Consequently, learning the model entails recovering the state transition dynamics M and the reward function R. In many instances, the reward function is also explicitly defined; therefore, the primary objective of model learning is to learn the state transition dynamics [7, 8].

An agent interacts with the environment and generates experienced data that is factored into the learning algorithm's sample complexity. With an environment model, the agent can utilize their imagination. It can interact with the model to sample interaction data, also called simulation data, which is typically not accounted for in sample complexity. Idealistically, if the model is sufficiently accurate, a good policy can be learned from the model. Compared with the model-free reinforcement learning (MFRL) methods [3, 9–13], where the agent does not model the environment transitions, MBRL allows the agent to perform cost-free explore in the learned model. Notable alternatives to MBRL that attempt to better utilize the experienced data include the off-policy algorithms, which use a replay buffer to store the old data, and the actor-critic algorithms, which can be viewed as learning a critic to facilitate policy updates. Figure 1 illustrates various types of RL structures. The simplest on-policy RL is depicted in Figure 1(a), where the agent uses the most recent data to update the policy. In the off-policy, as illustrated in Figure 1(b), the agent collects historical data in the replay buffer, where the policy is learned. In actor-critic RL, as illustrated in Figure 1(c), the agent first learns a critic, the value function of the long-term return, and then learns the policy (actor) with the assistance of the critic. Figure 1(d) demonstrates that MBRL explicitly learns a model. MBRL reconstructs the state transition dynamics, whereas off-policy RL estimates the value more robustly using the replay buffer. Although calculating the value function, or the critic, involves the information of the transition dynamics, the learned model in MBRL is decoupled with the policy and thus can be used to evaluate other policies, while the value function is bound to the sampling policy. Note that off-policy, actor-critic, and model-based are three parallel structures; Figure 1(e) illustrates a possible combination.

Recent studies from both theoretical [14] and empirical [6, 8] perspectives demonstrate that, given a sufficiently accurate model, it is intuitive that MBRL yields higher sample efficiency than MFRL. However, it is not simple to learn an ideal model in various DRL tasks with relatively complex environments. Therefore, we must carefully consider model learning and application strategies. Furthermore, a learned model is inevitably inaccurate to some extent, resulting in compounding errors when simulating

an episode [15].

In this survey, we examine the model-based RL techniques in depth. As described in Sections 2 and 3, our initial focus is on how models are learned and utilized in a fundamental context. For model learning, we begin with the classic tabular represented models; then for approximation models such as neural networks, we review the theories, the key challenges when confronting complex environments, as well as the latest developments in reducing model errors. For model usage, we categorize the literature into two sections: blackbox model rollout for trajectory sampling and Whitebox model for gradient propagation. Regarding the model used as a subsequent task of model learning, we also discuss efforts to bridge model learning and model usage, namely value-aware and policy-aware model learning. In addition, we examine the combination of model-based methods in other types of RL, such as offline RL, goal-conditioned RL, multi-agent RL, and meta-RL. In addition, we discuss the applicability and benefits of MBRL in real-world scenarios. This paper concludes with a discussion of the most promising emerging research perspectives and future trends in the evolution of MBRL.

2 Model learning

For MBRL, the first component to consider is the learning of the environment model. As introduced above, the model is formulated as the MDP $\langle S, A, M, R, \gamma\rangle$, where S, A, and γ are commonly predefined, and the state transition dynamics M and the reward function R are to be learned. We assume that some historical data is available for learning. Usually, the historical data can be in the form of trajectories $\{\tau_1, \tau_2, \ldots, \tau_k\}$, and each trajectory is a sequence of state-action-reward pairs, $\tau_i = (s_0^i, a_0^i, r_0^i, \ldots, s_{L-1}^i, a_{L-1}^i, r_{L-1}^i, s_L^i)$. It is easy to discover that the data records the past transitions (s_t, a_t, s_{t+1}) corresponding to the input and output of the transition dynamics M, and the past rewards (s_t, a_t, r_t) corresponding to the input and output of the reward function R. Therefore, it is straightforward to borrow ideas from supervised learning for model learning.

2.1 Model learning in tabular setting

At the early stage of RL research, the state and action spaces are finite and small, and the model learning is considered with the tabular MDPs [16], where policies, transition dynamics, and reward functions can all be recorded in a table. To learn the transition dynamics, let $C[s, a, s']$ record the counting of the state-action-next-state (s, a, s'). The transition dynamics is then

$$\hat{M}(s'|s,a) = \begin{cases} \frac{C[s,a,s']}{\sum_{s''\in S} C[s,a,s'']}, & \sum_{s''\in S} C[s,a,s''] > 0, \\ \frac{1}{|S|}, & \text{otherwise.} \end{cases} \tag{1}$$

For the reward function, let $\mathrm{Sum}[s, a]$ record the sum of rewards received by taking action a on state s. The reward function is then

$$\hat{R}(s,a) = \begin{cases} \frac{\mathrm{Sum}[s,a]}{C[s,a]}, & C[s,a] > 0, \\ R_{\min}, & \text{otherwise,} \end{cases} \tag{2}$$

where $R_{\min}$ is the preset minimum value of the immediate reward.

The above simple calculation of the transition dynamics and the reward function corresponds to the maximum likelihood estimation (MLE) under the tabular setting. Notice that $\hat{M}$ and $\hat{R}$ are an unbiased estimation of the true transition M^* and the true reward function R^*, respectively, and thus converge to M^* and R^* as the samples approach infinity.

For collecting samples, sampling trajectories from the environment are not as straightforward as sampling a coin. R-MAX [17] is a representative algorithm for joint model learning and exploration. In R-MAX, a state transits to itself, and the immediate reward is set to the maximum value by default, but only when a state-action pair has been visited sufficiently many times, i.e., larger than K, the transition probability, and the reward are assigned to their average value. This is implemented by using $\tilde{M}$ and $\tilde{R}$ as in (3).

$$\tilde{M}(s'|s,a) = \begin{cases} \frac{C[s,a,s']}{\sum_{s''\in S} C[s,a,s'']}, & C[s,a] \geqslant K, \\ I[s'=s], & C[s,a] < K, \end{cases} \qquad \tilde{R}(s,a) = \begin{cases} \frac{\mathrm{Sum}[s,a]}{C[s,a]}, & C[s,a] \geqslant K, \\ R_{\max}, & C[s,a] < K, \end{cases} \tag{3}$$

Table 1 State-of-the-art regret bounds for model-based and model-free algorithms on episodic tabular MDPs[a)]

	Algorithm	Regret
Model-based	UCBVI [19]	$\widetilde{\mathcal{O}}(\sqrt{H^3\|\mathcal{S}\|\|\mathcal{A}\|K})$
Model-free	UCB-ADVANTAGE [20]	$\widetilde{\mathcal{O}}(\sqrt{H^3\|\mathcal{S}\|\|\mathcal{A}\|K})$
	Lower bound [21]	$\Omega(\sqrt{H^3\|\mathcal{S}\|\|\mathcal{A}\|K})$

a) $|\mathcal{S}|$ is the state space size, $|\mathcal{A}|$ is the action space size, H is the planning horizon, and K is the number of episodes. $\widetilde{\mathcal{O}}()$ omits any log factors.

where I is the indicator function that is 1 when the inner expression is true and 0 otherwise.

In every iteration, R-MAX solves an ϵ-optimal policy in the fictitious model $\langle S, A, \tilde{M}, \tilde{R}, \gamma\rangle$, which naturally tries to explore unvisited states due to the set of the $R_{\max}$ reward, and applies the policy in the environment to collect more samples to update the fictitious model. With a properly set K, R-MAX requires $\tilde{O}\big(\frac{|\mathcal{S}|^2|\mathcal{A}|}{\epsilon^3(1-\gamma)^2}\log\frac{1}{\delta}\big)$ episodes to achieve a high accuracy (ℓ_1 difference on transition $< \epsilon/2$) model [18].

Another important theoretical property of RL is the regret,

$$\text{Regret}(K) = \sum_{k=1}^{K} V^*(s_0) - V^{\pi_k}(s_0),$$

where K is the total number of episodes. The regret measures cumulative value different from the optimal policy. Table 1 [19–21] lists the current best results. A model-based method UCBVI [19], which uses the learned model to estimate the values, has achieved the regret close to the lower bound. Meanwhile, a model-free method UCB-ADVANTAGE [20] also has the same regret bound. This comparison implies that model-based methods are unlikely to have any advantage over model-free methods, in the studied situations. This conclusion, however, contradicts many practical experiences. A key limitation of the analysis is the assumption of the tabular setting, where models are not able to generalize across states. General theoretical understandings of MBRL with function approximations are still unknown.

2.2 Model learning via prediction loss

While the counting method and the theory of model learning under the setting of tabular MDPs are clear, it is not feasible to use tabular representation for large-scale MDPs and MDPs with continuous state space and action space. Approximation functions are therefore employed in the general setting. The approximation functions can be implemented by machine learning models, such as linear models, neural networks, and decision tree models. In this survey, we focus on neural network models, for which let M_θ with parameter θ being the network weights denote the transition model. Meanwhile, in order to explicitly indicate the real transition dynamics, let M^* denote it.

2.2.1 *Prediction model loss*

A straightforward approach to model learning fits one-step transitions, which has been widely employed [7, 8, 22–24]. When M_θ is deterministic, the model learning objective can be the mean squared prediction error of the model M_θ on the next state [25].

$$\min_\theta \mathbb{E}_{(s,a)\sim\rho^{M^*}_{\pi_D}, s'\sim M^*(\cdot|s,a)}\left[\|s' - M_\theta(s,a)\|_2^2\right], \tag{4}$$

where M^* is the real transition dynamics, π_D is the data-collecting policy, and $\rho^{M^*}_{\pi_D}$ is the stationary state-action distribution induced by π_D and M^*. Intuitively, $\rho^{M^*}_{\pi_D}$ is the data collected by running π_D for a long period.

However, the deterministic transition model fails to capture the aleatoric uncertainty [26], which arises from the inherent stochasticities of the environment. To model the aleatoric uncertainty, a natural idea is to utilize the probabilistic transition model $M_\theta(\cdot|s,a)$ [26]. Under this case, the objective can be minimizing the KL divergence between $M^*(\cdot|s,a)$ and $M_\theta(\cdot|s,a)$ as in (5).

$$\min_\theta \mathbb{E}_{(s,a)\sim\rho^{M^*}_{\pi_D}}\left[D_{\text{KL}}\left(M^*(\cdot|s,a), M_\theta(\cdot|s,a)\right)\right] := \mathbb{E}_{(s,a)\sim\rho^{M^*}_{\pi_D}, s'\sim M^*(\cdot|s,a)}\left[\log\left(\frac{M^*\left(s' \mid s,a\right)}{M_\theta\left(s' \mid s,a\right)}\right)\right]. \tag{5}$$

The probabilistic transition model is often instantiated as a Gaussian distribution [7, 8, 26], i.e., $M_\theta(\cdot|s,a) = \mathcal{N}(\mu_\theta(s,a), \Sigma_\theta(s,a))$ with parameterized models of μ_θ and Σ_θ. Then Eq. (5) becomes

$$\min_\theta \mathbb{E}_{(s,a)\sim\rho^{M^*}_{\pi_D}, s'\sim M^*(\cdot|s,a)} \left[(\mu_\theta(s,a) - s')^{\mathrm{T}} \Sigma_\theta^{-1} (\mu_\theta(s,a) - s') + \log(\det \Sigma_\theta(s,a)) \right].$$

Using the prediction model loss of either (4) or (5), we can see that the model learning task has been transformed to be a supervised learning task. Any supervised learning technique can be employed to achieve efficient and effective model learning.

2.2.2 *Model properties*

When a model has been obtained using the prediction model loss, we then care how well the model can help. We can evaluate the model with some static metrics which can be evaluated on a static dataset [27], e.g., likelihood ratio, outlier rate, and explained variance. Moreover, for RL tasks, it is also significant to know how different a policy π performs in the model and in the real environment, i.e., the value evaluation error,

$$\|V^\pi_{M_\theta} - V^\pi_{M^*}\|_\infty.$$

The Simulation Lemma firstly proven in [28] says the following.

Theorem 1 (Simulation Lemma)**.** Given an MDP with reward upperbound $R_{\max}$ and transition model M^*, a learned transition model M_θ with $\max_{s,a} \|M_\theta(s,a) - M^*(s,a)\|_1 \leqslant \epsilon_m^{\max}$, and a learned reward function with $\max_{s,a} |R_\theta(s,a) - R(s,a)| \leqslant \epsilon_r$, the value evaluation error of any policy π is bounded as

$$\|V^\pi_{M_\theta} - V^\pi_{M^*}\|_\infty \leqslant \frac{\gamma \epsilon_m^{\max} R_{\max}}{2(1-\gamma)^2} + \frac{\epsilon_r}{1-\gamma}. \tag{6}$$

The Simulation Lemma shows that the value loss corresponding to the model error $\epsilon_m^{\max}$ has a quadratic coefficient on the effective horizon $\frac{1}{1-\gamma}$. This means that the value loss grows quadratically fast as the horizon grows. When the reward function is out of consideration, as discussed above, we can omit the ϵ_r term. Meanwhile, the Simulation Lemma also shows that the reward error is not severe compared with the model error.

We can also note the limitations of the Simulation Lemma. Compared with the learning loss (4) where the data follows the distribution of the data-collecting policy, the model error $\epsilon_m^{\max}$ is measured as the maximum difference over all state-action pairs, which is not easily achieved or assessed in a practical way.

In the recent analysis [7,8,29], the error of the learned transition model M_θ is measured over the data distribution, and thus directly connects with the learning loss (4). We present the result in Theorem 2, named Simulation Lemma II. Note that we omit the reward function error since it is not essential.

Theorem 2 (Simulation Lemma II)**.** Given an MDP with reward upper bound $R_{\max}$ and transition model M^*, a data-collecting policy π_D, and a learned transition model M_θ with

$$\mathbb{E}_{(s,a)\sim\rho^{M^*}_{\pi_D}} \left[D_{\mathrm{KL}}\big(M^*(\cdot|s,a), M_\theta(\cdot|s,a)\big) \right] \leqslant \epsilon_m^\rho,$$

for an arbitrary policy π with bounded divergence,

$$\max_s D_{\mathrm{KL}}\big(\pi(\cdot|s), \pi_D(\cdot|s)\big) \leqslant \epsilon_\pi,$$

the value evaluation error of the policy is bounded as

$$|V^\pi_{M_\theta} - V^\pi_{M^*}| \leqslant \frac{\sqrt{2} R_{\max}\gamma}{(1-\gamma)^2}\sqrt{\epsilon_m} + \frac{2\sqrt{2} R_{\max}}{(1-\gamma)^2}\sqrt{\epsilon_\pi}. \tag{7}$$

Note that let ϵ_m^ρ denote the distributional model error to distinguish the uniform model error $\epsilon_m^{\max}$ in Theorem 1.

The policy evaluation error of Simulation Lemma II contains two terms, the bias of the learned model, and the policy divergence between the evaluating policy π and the data-collecting policy π_D. Theorem 2 indicates that the policy evaluation error w.r.t (with respect to) the model error ϵ_m^ρ also has a quadratic coefficient on the effective horizon $\frac{1}{1-\gamma}$, similar to Theorem 1. The coefficient means a quadratic compounding error of learning in the model, which is the reason that studies such as [8] only adopt short rollouts, say, less than 10 steps, in the learned model.

Moreover, Xu et al. [30] provided a finite sample complexity result for the evaluation error by further relating the model error ϵ_m^ρ with the samples and model space of the model learning. Interested readers are referred to [30, Corollary 2] for the detailed analysis.

2.2.3 *Model variants*

Since the compounding error is due to the recursive state-action generation using the one-step transition model, a way of alleviating the issue is to predict many steps at a time. A multistep model [15] takes the current state s_t and a sequence of actions $(a_t, a_{t+1}, \ldots, a_{t+h})$ with length h as the input, and predicts the future h states. A (deterministic) multistep model is represented as

$$(s_{t+1}, \ldots, s_{t+h}) = M_\theta^h(s_t, a_{t+1}, \ldots, a_{t+h}),$$

which can also be trained by supervised learning. Intuitively, compared with the one-step model, the multistep model does not take the predicted "fake" states as the input and hence could avoid the compounding error within h steps [15]. Meanwhile, the dynamics across multi-steps can be much more complex than one-step dynamics; therefore, multistep transition predictions can have a larger error.

The transition models discussed before are all forward models, i.e., predict along time. In addition to the forward model, there also exist studies on the backward transition model in MBRL [31–35]. The backward transition model takes the future state s_{t+1} and action a_t as inputs and predicts the state s_t. The backward model is often used to generate reverse data, which can help reduce the rollout horizon [33] and could improve the sample efficiency of MBRL [35].

2.3 Model learning with reduced error

In the above model properties, we can see the horizon-squared compounding error is a major issue of model learning. The issue is mainly due to the use of prediction loss to learn an unconstrained model.

2.3.1 *Model learning with Lipschitz continuity constraint*

To reduce the compounding error, one way is to constrain the model. Venkatraman et al. [36] and Asadi et al. [37] employed the Lipschitz continuity constraint for the models. They firstly employed Wasserstein distance [38] to measure the similarity between two transition distributions. For any two distributions P and Q over space $\mathcal{X}$, the Wasserstein distance between P and Q is defined as

$$W(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma} \left[\mathrm{d}(x, y)\right],$$

where $\Pi(P, Q)$ denotes the set of all joint distributions $\gamma(x, y)$, whose marginals are respectively P and Q, and d is the distance metric on $\mathcal{X}$. Compared with other divergence measures (e.g., KL divergence and TV distance), Wasserstein distance can appropriately measure the similarity between two distributions with disjoint supports [37].

When considering a state distribution ρ over the state space S rather than a single state s, we can also define the transition model, i.e., the generalized transition model, $M_\theta(\cdot|\rho, a)$ as

$$M_\theta(s'|\rho, a) := \int M_\theta(s'|s, a)\rho(s)\mathrm{d}s.$$

M_θ is called ϵ_w-accurate w.r.t. the Wasserstein distance if and only if $\sup_{s,a} W(M_\theta(\cdot|s, a), M^*(\cdot|s, a)) \leqslant \varepsilon_w$. Notice that ϵ_w measures the one-step error of the transition model and is connected to $\epsilon_m^{\max}$ in Theorem 1.

In MBRL, we desire an upper bound on the n-step error. For a fixed initial state distribution μ and a fixed sequence of actions $(a^0, \ldots, a^{n-1})$, the n-step error is defined as

$$W(M_\theta^n(\cdot|\mu), M^{*,n}(\cdot|\mu)),$$
$$\text{where } M^n(s|\mu) := \mathbb{P}\left(s_n = s|s_0 \sim \mu(\cdot), a_0 = a^0, s_1 \sim M(\cdot|s_0, a_0), a_1 = a^1, \ldots, a_{n-1} = a^{n-1}\right).$$

The Lipschitz continuity is introduced for probabilistic transition models.

Definition 1. A probabilistic transition model M is K-Lipschitz if and only if

$$\sup_{a\in\mathcal{A}}\sup_{\rho_1,\rho_2\in\Delta(\mathcal{S})}\frac{W(M(\cdot|\rho_1,a),M(\cdot|\rho_2,a))}{W(\rho_1,\rho_2)}\leqslant K.$$

With a Lipschitz continuity-constrained model, the n-step error can be bounded.

Theorem 3 (Theorem 1 in [37])**.** Suppose the real transition model M^* is K^*-Lipschitz and the learned transition model M_θ is K-Lipschitz and ε_w-accurate, then $\forall n\geqslant 1$,

$$W(M_\theta^n(\cdot,\mu),M^{*,n}(\cdot,\mu))\leqslant\varepsilon_w\sum_{i=1}^{n-1}(\overline{K})^i,$$

where $\overline{K}=\min\{K^*,K\}$.

Asadi et al. [37] further introduced two assumptions to simplify the analysis. One assumption is a single action, i.e., $\mathcal{A}=\{a\}$. The other assumption is the state-only K_R-Lipschitz continuous reward function. Under the two assumptions, the value function under the transition M is simplified as

$$V_M(s)=\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t)|s_0=s,a_t=a,s_{t+1}\sim M(\cdot|s_t,a_t),t=0,1,2,\ldots\right].$$

Then the value function error (of the policy taking the only action) can be bounded.

Theorem 4 (Theorem 2 in [37])**.** Under the same assumptions as in Theorem 3. Further, suppose that $\mathcal{A}=\{a\}$ and the reward function only depends on the state and is K_R-Lipschitz continuous. Then $\forall s\in\mathcal{S}$ and $\overline{K}\in[0,\frac{1}{\gamma})$,

$$|V_{M_\theta}(s)-V_{M^*}(s)|\leqslant\frac{\gamma K_R\epsilon_w}{(1-\gamma)(1-\gamma\overline{K})},\tag{8}$$

where $\overline{K}=\min\{K^*,K\}$.

Although the analysis is over-simplified, we can still notice that when $\overline{K}$ is small, the evaluation error can be small. In other words, the compounding error can be controlled, compared with those in the simulation lemmas (i.e., Theorems 1 and 2).

Since $\overline{K}$ is the minimum between the Lipschitz constants of M^* and M_θ, the theorem suggests a trade-off on the Lipschitz constant of M_θ. When K is small and $K\leqslant K^\star$, $\overline{K}$ is also small. Meanwhile, with a small K, the model might be hard to approximate M^* with a large K^*, and thus ϵ_w could be large.

2.3.2 *Model learning by distribution matching*

The prediction loss employed in Theorems 1 and 2 minimizes the model error on each point of the state-action data. While the prediction loss minimization can be straightforwardly solved by supervised learning, the long-term effect of transitions is hard to capture, resulting in the horizon-squared compounding error issue. Therefore, to learn the long-term effect of transitions, an idea is to match the distributions between the real trajectories and the trajectories rolled out in the learned model.

The idea of distribution matching has been employed in imitation learning through adversarial learning such as the GAIL method [39] that imitates the expert policy in an adversarial manner, where a discriminator $\mathfrak{D}$ learns to identity whether a state-action pair comes from the expert demonstrations and a generator π imitates the expert policy by maximizing the discriminator score. This corresponds to the minimax optimization problem,

$$\min_{\pi\in\Pi}\max_{\mathfrak{D}\in(0,1)^{S\times A}}\mathbb{E}_{(s,a)\sim\rho_{\pi_E}}\left[\log\left(\mathfrak{D}(s,a)\right)\right]+\mathbb{E}_{(s,a)\sim\rho_\pi}\left[\log(1-\mathfrak{D}(s,a))\right],$$

recalling that ρ_π is the state-action distribution by running the policy π, and π_E is the expert policy. When the discriminator is optimal to the inner objective, i.e., $\mathfrak{D}^*(s,a)=\rho_{\pi_E}(s,a)/\left(\rho_{\pi_E}(s,a)+\rho_\pi(s,a)\right)$, the generator essentially minimizes the Jensen-Shannon (JS) divergence between ρ_{π_E} and ρ_π (up to a constant),

$$\min_{\pi\in\Pi}D_{\mathrm{JS}}(\rho_{\pi_E},\rho_\pi):=\frac{1}{2}\left[D_{\mathrm{KL}}\left(\rho_{\pi_E},\frac{\rho_\pi+\rho_{\pi_E}}{2}\right)+D_{\mathrm{KL}}\left(\rho_\pi,\frac{\rho_\pi+\rho_{\pi_E}}{2}\right)\right],\tag{9}$$

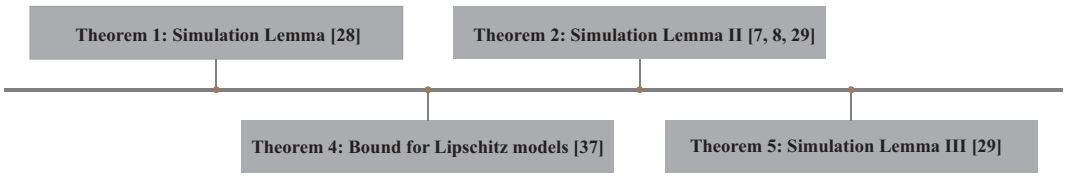

Figure 2 Illustration of milestones of theoretical results about model learning.

which achieves the goal of distribution matching and solves the compounding error issue of imitation learning theoretically [29, 40–42] and empirically [39, 43, 44].

Through the bridge of duel MDP [45, 46] that treats the environment also as an agent, the idea of distribution matching is brought for model learning [29, 46–48]. The transition model M_θ, which takes the current state-action pair as inputs and predicts a distribution over the next state, is regarded as a policy in the imitation view. A discriminator is employed to distinguish expert state-action-next-state tuples from the "fake" ones. The minimax optimization problem in [29] is then formulated as

$$\min_{M_\theta}\max_{\mathfrak{D}}\mathbb{E}_{(s,a,s')\sim\mu^{M^*}}\left[\log\left(\mathfrak{D}(s,a,s')\right)\right]+\mathbb{E}_{(s,a,s')\sim\mu^{M_\theta}}\left[\log(1-\mathfrak{D}(s,a,s'))\right], \tag{10}$$

where μ^{M^*} and μ^{M_θ} are the joint state-action-next-state distributions induced by the data-collecting policy π_D in the true environment M^* and M_θ, respectively. Formally,

$$\mu^{M^*}(s,a,s')=\rho^{M^*}_{\pi_D}(s,a)M^*(s'|s,a)\ ,\ \mu^{M_\theta}(s,a,s')=\rho^{M_\theta}_{\pi_D}(s,a)M_\theta(s'|s,a).$$

The optimal solution of M_θ minimizes the JS divergence between $\mu^{M^\star}$ and μ^{M_θ}, matching the trajectory distribution. Different from [29], Wu et al. [47] chose to minimize the Wasserstein distance between μ^{M^*} and μ^{M_θ}. In [29, 47], they kept the data-collecting policy π_D fixed during the training process of the transition model M_θ. On the other hand, Shi et al. [46] optimized the transition model and policy jointly, which forms a multi-agent adversarial imitation learning.

By the distribution matching, Xu et al. [29] (Theorem 3) proved an improved policy evaluation error bound as in Theorem 5, which is named Simulation Lemma III.

Theorem 5 (Simulation Lemma III)**.** Given an MDP with reward upper bound $R_{\max}$ and transition model with M^*, a data-collecting policy π_D, and a learned transition model M_θ with

$$D_{\mathrm{JS}}(\mu^{M_\theta},\mu^{M^*})\leqslant\epsilon_m^{\mathrm{JS}},$$

for an arbitrary policy π with bounded divergence,

$$\max_s D_{\mathrm{KL}}\big(\pi(\cdot|s),\pi_D(\cdot|s)\big)\leqslant\epsilon_\pi,$$

the policy evaluation error is bounded as

$$|V^\pi_{M_\theta}-V^\pi_{M^*}|\leqslant\frac{2\sqrt{2}R_{\max}}{1-\gamma}\sqrt{\epsilon_m^{\mathrm{JS}}}+\frac{2\sqrt{2}R_{\max}}{(1-\gamma)^2}\sqrt{\epsilon_\pi}. \tag{11}$$

We can see in Theorem 5 that the coefficient on the model error ϵ_m^{JS} is linear w.r.t. the effective horizon, i.e., $\frac{1}{1-\gamma}$, which meets the lower bound [30] and thus cannot be further improved in general. This means the compounding error issue is solved.

One may further notice that the model error ϵ_m^{JS} is a different quantity from the error ϵ_m in Theorem 2. This is true. To achieve the same value, the matching loss using the JS-divergence requires more samples than the prediction loss. In [42], the adversarial imitation approach has been improved to have a lower sample complexity than that of supervised learning.

Finally, we summarize the main theoretical results about model learning in Figure 2.

2.3.3 *Robust model learning*

While in Simulation Lemma III the compounding error is reduced, the policy divergence term about ϵ_π can still be large. The policy divergence is the difference between the data-collecting policy and the target policy. In order to reduce the divergence, one direction is to use a data-collecting policy with a

wide distribution. Zhang et al. [49] proposed to use a distribution of policy to collect data, instead of a single policy, such that the divergence can be reduced. When using a distribution of the data-collecting policy, the model learning objective can be formulated as

$$\min_{M_\theta} \mathbb{E}_{\pi\sim\mathcal{P}(\pi)} \left[\max_{\mathfrak{D}_\pi} \mathbb{E}_{(s,a,s')\sim\mu_\pi^{M_*}} [\log \mathfrak{D}_\pi(s,a,s')] + \mathbb{E}_{(s,a,s')\sim\mu_\pi^{M_\theta}} [\log(1-\mathfrak{D}_\pi(s,a,s'))] \right], \tag{12}$$

where $\mu_\pi^{M_*}$ and $\mu_\pi^{M_\theta}$ are the joint state-action-next-state distributions. Empirically, the expectation operation $\mathbb{E}_{\pi\sim\mathcal{P}(\pi)}$ can be approximated by taking the average over a set of sampled policies.

Furthermore, to particularly deal with concern-case interacting policies, i.e., the outlier policies with much different behavior from the majority, Zhang et al. [49] designed conditioned value at risk (CVaR) [50] objective which focuses on the ϵ-percentile of policies with the largest value discrepancy in simulation.

2.4 Model learning for complex environments dynamics

The mainstream realization of the environment dynamics model is an ensemble of Gaussian processes where the mean vector and covariance matrix for the distribution of the next state are built based on neural networks fed in the current state-action pair [26]. Such an architecture is shown to work well on MuJuCo robot locomotion environments, where the state observations are sufficient statistics for future derivation. However, there still exist many complex environments that are hard to be directly modeled via the above method. Below we discuss two important aspects of complex environment dynamics modeling.

Partial observability. For partially observable environments, the observations may not be sufficient statistics for future derivation, which makes the environment a partially observable MDP (POMDP) [51]. For model learning in a POMDP, belief state estimation is the classic solution, where an observation model $p(o_t|s_t)$ and a latent transition model $p(s_{t+1}|s_t,a_t)$ are learned via maximizing a posterior and the posterior distribution $p(s_t|o_1,\ldots,o_t)$ can be inferred. With deep learning, the latent state distribution $p(s_t|o_1,\ldots,o_t)$ can be obtained via a recurrent neural network [52]. Hausknecht and Stone [53] further introduced a delta distribution for deterministic settings or a Gaussian distribution for stochastic settings.

Representation learning. For high-dimensional state space such as images, representation learning that learns informative latent state or action representation will much benefit the environment model building so as to improve the effectiveness and sample efficiency of model-based RL [54] on different aspects, including value prediction [55] and model rollout [25]. Ha and Schmidhuber [52] applied an autoencoder network to encode the latent state that can reconstruct the image. Hafner et al. [56] proposed Dreamer to learn the latent dynamics for visual control tasks, in which an environment model (called world model) with visual encoder and latent dynamics is learned based on collected experience, and the model is shown to have the capability of performing long rollout and value estimation. Hafner et al. [57] further proposed DreamerV2 that supports the agent to learn purely from the model rollout data and achieve human-level performance on 55 Atari game tasks. Compared with Dreamer, DreamerV2 replaces the Gaussian latent as proposed in PlaNet [58] with the discrete latent, which brings superior performance. The possible reason for such effects would be that the discrete latent representation can better fit the aggregate posterior and handle multi-modal cases. Considering the state-action pair distribution mismatch between the model training and model rollout stages, Shen et al. [59] incorporated a domain adaptation objective into the model learning task to encourage the model to learn invariant representations of state-action pairs between the real data and rollout data.

3 Model usage and integration with model learning

In this section, we will start by introducing three kinds of model usages, which are categorized as in Figure 3. We will first introduce the methods that do not utilize the internal structure of the model, which require decision-time model simulations (Subsection 3.1) or use the data simulated by the model for RL (Subsection 3.2). Then we introduce the methods that use a differentiable model for gradient generation (Subsection 3.3). Finally, we cover the attempts to bridge model learning and model usage, i.e., value-aware and policy-aware model learning (Subsection 3.4).

3.1 Planning with model simulation

When a model is available, the most straightforward idea of utilizing the model is to plan in it. Planning denotes any computational process that takes a model as input and produces or improves a policy for

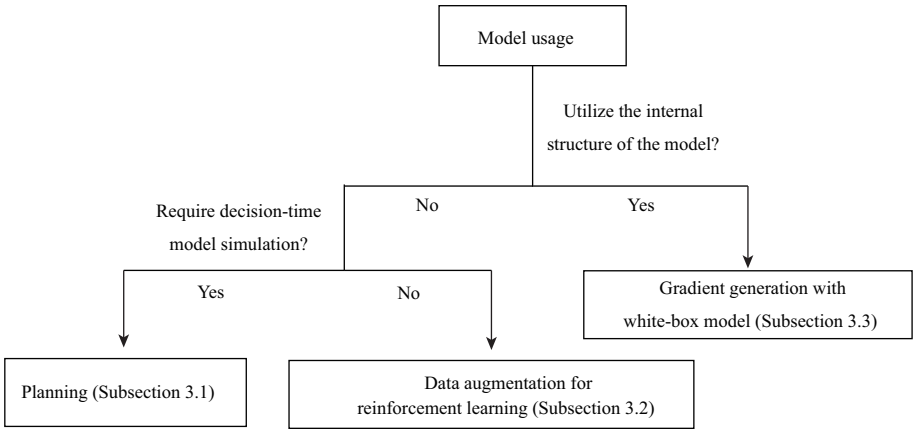

Figure 3 Illustration for the taxonomy formulation of the model usages.

interacting with the modeled environment [1, 60, 61]. We will list the MBRL approaches that integrate planning into their methods or frameworks. We will categorize these approaches according to the planning methods they adopt.

Model predictive control (MPC). MPC [62] is a kind of model-based control method that plans an optimized sequence of actions in the model. The classical MPC approach requires to design a parametric model and policy, such as in the linear quadratic form, to fit the optimizer such as a quadratic optimization solver. Learning-based MPC [63] has a tight connection with MBRL, particularly for nonlinear and black-box models. In general, at each time step, MPC obtains an optimal action sequence by sampling multiple sequences and applying the first action of the sequence to the environment. Formally, at time step t, an MPC agent will seek an action sequence $a_{t:t+\tau}$ by optimizing

$$\max_{a_{t:t+\tau}} \mathbb{E}_{s_{t'+1}\sim p(s_{t'+1}|s_{t'},a_{t'})}\left[\sum_{t'=t}^{t+\tau} r(s_{t'},a_{t'})\right], \tag{13}$$

where τ denotes the planning horizon. Then the agent will choose the first action a_t from the action sequence and apply it to the environment.

Black-box MPC regards (13) as a black-box optimization problem and adopts some zero-order optimization methods to solve it. MB-MF [25] adopts a basic optimization method, i.e., the Monte Carlo (MC) method (also known as "random shooting"), which samples a number of action sequences $a_{t:t+\tau}$ from the space of action sequence uniformly and randomly. By applying the action sequences in the model, the current state s_t can be transited to $s_{t+\tau}$ following the transition distribution. The returns accumulated during the transition process are used to evaluate the action sequences. The action sequence with the highest evaluation will be preserved as the solution of (13). The MC method is simple to implement and does not require many computational resources. However, because of the low efficiency of the random sampling process, it also suffers from high variance and could fail to sample a high reward action sequence when the action space is large. Recent advances in MPC methods focus on altering the sampling strategies [26, 58] and the sampling space [64].

Replacing the MC method with CEM [65], PETS [26], and PlaNet [58] improves the optimization efficiency. Instead of sampling randomly and uniformly, CEM samples the action sequences from a multivariate normal distribution, which will be adjusted according to the evaluation of the sampled sequences such that the high-reward sequences can be sampled with a higher probability. This principle resembles many other derivative-free optimization methods [66–68]. As a result, other derivative-free optimization methods can also be used to solve (13) and integrated into the MPC framework. In addition to altering the optimization method, POPLIN-A [64] further improves the optimization efficiency by altering the sampling space to a space of action bias. Specifically, POPLIN-A seeks a sequence of action residuals to adjust an action sequence proposed by a policy. For example, POPLIN-A-Replan, a variant of POPLIN-A, alters (13) to (14) by introducing a policy function $\pi(s)$:

$$\max_{\delta_{t:t+\tau}} \mathbb{E}_{s_{t'+1}\sim p(s_{t'+1}|s_{t'},\pi(s_{t'})+\delta_{t'})}\left[\sum_{t'=t}^{t+\tau} r(s_{t'},\pi(s_{t'})+\delta_{t'})\right]. \tag{14}$$

The policy learns to imitate the MPC results. As a result, at the state the policy has met, it can propose a nearly optimal action sequence as an initial solution. Thus, POPLIN-A largely simplifies the optimization problem and improves planning efficiency.

Another way to improve MPC is using a faster simulator, which implies more simulations and action-sequence evaluations in a fixed time period. As a result, we can obtain an optimal action sequence with higher probability. He et al. [69] proposed an influence-augmented local simulator (IALS), which only simulates the transitions of part of the state variables, namely local state variables. Specifically, IALS first uses influence-based abstraction (IBA) [70–72] to split the set of state variables into local state variables and non-local state variables, where the local state variables directly affect the observation and reward of the agent. Then IALS adopts a factorizable simulator and only predicts the transitions of the local state variables with the corresponding local simulator. As it only calls the part of the simulator related to the transition of the local state variables, IALS can enjoy better time-efficiency. However, there exist local state variables transited based on some non-local state variables, which IALS obtains by a recurrent neural network (RNN) based on past local state variables and actions. The RNN could introduce prediction errors and make IALS an imperfect simulator/transition model.

With an imperfect model, the result of MPC could not be unreliable. However, we can still make use of the planning result by integrating it into a MFRL framework. I2A [73] integrates Monte Carlo planning results into the MFRL framework as the auxiliary information. Instead of applying the first action of the sequence with the highest return, I2A encodes several rollouts from the model to rollout embeddings. The embeddings will then be aggregated and used to augment the input of the MFRL agent. As I2A does not rely on the planning results, it can successfully use imperfect models.

Monte Carlo tree search (MCTS). MCTS [2, 74–76] is an extension of Monte Carlo sampling methods. MCTS also aims at solving (13). Unlike the MC methods mentioned in the MPC part, MCTS adopts a tree-search method. At each time step, MCTS incrementally extends a search tree from the current environment state [74, 75]. Each node in the tree corresponds to a state, which will be evaluated by some approximated value functions or the return obtained after rollouts in the model with a random policy [75] or a neural network policy [2, 76, 77]. Finally, action will be chosen such that the agent can be more likely transited to a state which has a higher evaluated value. In MCTS, models are generally used to generate the search tree and evaluate the state.

AlphaGo [2] first uses MCTS to beat professional human players in the game of Go. AlphaGo first utilizes human expert data to pre-train a policy network, which is used to generate the search tree. For each decision timestep, AlphaGo uses MCTS to decide where to play the next stone on board. AlphaGo Zero [76] is able to defeat professional human players without any human knowledge. AlphaGo Zero trains a policy to mimic the planning outputs of MCTS, like POPLIN-A [64]. The policy is then used to generate the search tree in MCTS. The procedure of AlphaGo Zero can be regarded as an iteration between improving a policy by planning and enhancing the planning by the policy. This training procedure has also been adopted by recent work to play the board game Hex [78].

Despite the conventional MCTS algorithm can only be used in discrete action space, Couëoux et al. [79] and Moerland et al. [80] extended the MCTS framework to continuous action space by progressive widening [81, 82], which adaptively determines the number of child actions of a state in the tree according to the total number of visits of the state. Further, when the true model is not provided, MCTS can be applied to a learned model. Value prediction network (VPN) [55] learns an abstract state transition model. The abstract state transition model infers the next abstract state by taking the current abstract state and action as input, which is the same as the transition function in typical MDP. However, the abstract state has no semantics of the corresponding state. The purpose of the abstract transition model is to convert the abstract state to an abstract state that can be used to make more precise value and reward predictions. As a result, given an action sequence, the VPN can predict the reward and the state value for the future states after taking the action sequence to the environment. VPN applies MCTS to the learned model to search for an action sequence that has the highest bootstrapped environment return. With the abstract transition model, VPN can be applied to the tasks where the observation is the image, e.g., Atari games. The experiments show that VPN can outperform DQN [3] in several Atari games. With a similar framework, MuZero [83] further improves the performance in Atari games. MuZero also learns a transition model but additionally learns an abstract policy, which outputs actions with the abstract states as inputs. Empirical results have shown huge advantages of MuZero in Atari games, Go, chess, and shogi.

Background planning. MPC and MCTS always begin and complete planning after the agent en-

counters a new state. This kind of method is also known as decision-time planning [1]. Another way of planning is background planning, which uses the simulated data obtained from the model to improve the policy or value learning [1]. Dynamic programming [1], tabular Dyna [16, 84], and prioritized sweeping [85] all belong to background planning methods. Here, we introduce VIN [86], which implements a dynamic programming method, value iteration (VI), with a neural network. VIN reveals that the classic VI planning algorithm [87] may be represented by a specific type of convolutional neural network (CNN). A step of VI can be achieved by passing a reward tensor to a CNN followed by a max-pooling layer. They embed such a module inside a standard feed-forward network and thus obtain an NN model that can learn to plan implicitly and provide the policy with useful planning results.

3.2 Data augmentation with model simulation

With a model, we can generate any number of simulated samples as we want. We can use the simulated data for policy learning or value approximation. These kinds of integration of policy/value learning and models are known as Dyna-style methods. Dyna-style methods [16] utilize the learned transition model to generate more experiences and then perform RL on the dataset augmented by the model experiences. As a result, the models in Dyna-style methods are regarded as the data-augmenter for the policies. The main purpose of models is to generate simulated experiences for policy learning. In this subsection, we will introduce value learning and policy learning with the simulated experience obtained from a model.

Value estimation. MC value estimation [1, 88] is the original method to approximate state values. Specifically, for a state s_t, it uses MC search to estimate $Q^\pi(s_t, a)$ by performing action a in state s_t and subsequently executing policy π in all successor states. In MC search, many simulated trajectories starting from (s_t, a) are generated following π. The value function $Q^\pi(s_t, a)$ will be estimated by averaging the cumulative reward of the trajectories. MC value estimation depicts an original model usage for value approximation, i.e., averaging the cumulative reward of the simulated trajectories. Another value approximation method is temporal-difference (TD) prediction [89], which is broadly used in many value-based RL methods [3, 12]. In one step TD, the update target of the value of state s_t, $V(s_t)$, is determined by the value of the next state $V(s_{t+1})$ and the received reward r_t: $r_t + \gamma V(s_{t+1})$. Compared with the MC methods, one step TD does not need an environment model and thus is preferred by many model-free methods. The intermediate between MC methods and one-step TD is H-step TD, whose update target for $V(s_t)$ is $\sum_{t'=t}^{t+H-1} \gamma^{t'-t} r_{t'} + \gamma^H V(s_{t+H})$. Feinberg et al. [23] proposed model-based value expansion (MVE), which indicates that H-step TD value prediction can reduce the value estimation error under some conditions, which is concluded in Theorem 6.

Theorem 6 (Theorem 3.1 in [23])**.** Define s_t, a_t, and r_t to be the states, actions, and rewards resulting from following policy π using the true dynamics f starting at $s_0 \sim v$ and analogously define $\hat{s}_t$, $\hat{a}_t$, and $\hat{r}_t$ using the learned dynamics $\hat{f}$ in place of f. Let the reward function r be L_r-Lipschitz and the value function V^π be L_V-Lipschitz. Let ϵ be an upper bound,

$$\max_{t \in [H]} \mathbb{E} \left\| \hat{s}_t - s_t \right\|^2 \leqslant \epsilon^2,$$

on the model risk for an H-step rollout. Then for any parameterized value function $\hat{V}$, H-step model value expansion estimate $\hat{V}_H(s_t) = \sum_{t'=t}^{H+t-1} \gamma^{t'-t} \hat{r}'_t + \gamma^H \hat{V}(s_{t+H})$ satisfies

$$\underset{\nu}{\mathrm{MSE}}(\hat{V}_H) \leqslant c_1^2 \epsilon^2 + (1 + c_2 \epsilon) \gamma^{2H} \underset{(\hat{f}^\pi)^H \nu}{\mathrm{MSE}}(\hat{V}),$$

where c_1 and c_2 grow at most linearly in L_r and L_V and are independent of H for $\gamma < 1$, $\mathrm{MSE}_\nu(V) = \mathbb{E}_{S \sim \nu}[V(S) - V^\pi(S)]$ measures the value estimation error, $(\hat{f}^\pi)^H \nu$ denotes the measure obtained by iteratively applying the policy π for H steps starting from the initial state distribution ν. We assume $\mathrm{MSE}_{(\hat{f}^\pi)^H \nu}(\hat{V}) \geqslant 2$ for simplicity of presentation, but an analogous result holds when the critic outperforms the model.

Theorem 6 implies that, if ϵ is small and the value function $\hat{V}$ is at least as accurate on imagined states $(\hat{f}^\pi)^H \nu$ as on those sampled from ν:

$$\underset{\nu}{\mathrm{MSE}}(\hat{V}) \geqslant \underset{(\hat{f}^\pi)^H \nu}{\mathrm{MSE}}(\hat{V}), \tag{15}$$

the mean squared error (MSE) of the H-step value estimation will approximately contract by γ^{2H}. Thus, the primary principle of MVE is forming H-step TD targets by unrolling the model dynamics for H steps. Moreover, to satisfy (15), MVE trains the value function on the data sampled in the environment as well as the imagined data sampled in a learned model. However, MVE relies on fixed horizon H, which is task-specific and may change in different training phases and state space. Stochastic ensemble value expansion (STEVE) [90] further improves MVE by interpolating between different horizons H based on the uncertainty calculated using ensemble. It reweights the value targets of different depths according to their uncertainty, which is derived from both the value function and transition dynamics uncertainty. The rollout length with lower uncertainty will be assigned with a higher weight.

Policy learning. The data augmented by the model can also be used by MFRL methods for policy improvement. The learned dynamic model can be regarded as a simulator where the policy can be trained. Kurutach et al. [22] proposed model ensemble trust region policy optimization (ME-TRPO), which learns a policy via TRPO [9] from a set of learned models. The training process iterates between collecting data using the current policy, training the ensemble model with the environment data, and improving the policy in the ensemble model. Each model is learned from the real trajectories by minimizing the prediction error. When interacting with the ensemble model, in every step, ME-TRPO randomly chooses a model to predict the next state given the current state and action. The collected imagined trajectories are used to update the policy via TRPO. Further, Luo et al. [7] studied such a learning framework from a theoretical perspective. The authors find that if we establish a discrepancy bound

$$D_{\pi_{\text{ref}}}(\hat{M}) = L \cdot \mathbb{E}_{S_0,\ldots,S_t\sim\pi_{\text{ref}},M^\star}\left[\|\hat{M}(S_t) - S_{t+1}\|\right],$$

and update a model $\hat{M}$ and a policy π following

$$\pi, \hat{M} \leftarrow \underset{\tilde{\pi}\in\Pi,\tilde{M}\in\mathcal{M}}{\arg\max}\ V^{\tilde{\pi},\tilde{M}} - D_{\pi_{\text{ref}}}(\tilde{M}), \quad \text{s.t.}\ \ \mathrm{d}(\tilde{\pi},\pi) \leqslant \delta, \tag{16}$$

the policy performance in a true dynamical model $M^\star$ will improve monotonically. Here, π_{ref} is a reference policy, $\hat{M}$ is the estimated model, $V^{\tilde{\pi},\tilde{M}}$ denotes the value of $\tilde{\pi}$ in model $\tilde{M}$, Π is the policy space, $\mathcal{M}$ is the model space. Eq. (16) can be divided into two terms. Based on (16), as a result, Luo et al. [7] proposed stochastic lower bound optimization (SLBO), which could be regarded as a variant of ME-TRPO. They discarded the gradient of the first term w.r.t. the model for an approximation. As a result, the maximization of the first term is equal to updating the policy by an RL algorithm in the current model. The second term implies minimizing an H-step prediction of the model. The training procedure is quite similar to ME-TRPO but uses a multi-step L2-norm loss to train the dynamics. SLBO is built with theoretical guarantees but still has a problem. SLBO uses the model to roll out whole trajectories from the start state. However, due to the compounding error of the model, we may not rollout so long horizon. Model-based policy optimization (MBPO) [8], on the other hand, samples the branched rollout in the model. MBPO begins a rollout from a state sampled in the real environment and runs k steps according to policy π and the learned model p_θ. Moreover, MBPO also adopts soft actor-critic [13], which is an off-policy RL algorithm, to update the policy with the mixed data from the real environment and learned model. MBPO also gives a monotonic improvement theorem with model bias and k-branch rollouts.

Theorem 7 (Theorem 4.3 in [8])**.** Let the expected TV-distance between two transition distributions be bounded at each timestep by ϵ_m, the policy divergence be bounded by ϵ_π, the model error on the distribution of the current policy π be bounded by $\epsilon_{m'}$, the true returns $\eta[\pi]$ and the returns from the k-branched rollout method satisfy:

$$\eta[\pi] \geqslant \eta^{\text{branch}}[\pi] - C(\epsilon_m,\epsilon_\pi,\epsilon_{m'},k), \quad C(\epsilon_m,\epsilon_\pi,\epsilon_{m'},k) = 2r_{\max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k\epsilon_\pi}{1-\gamma} + \frac{k}{1-\gamma}\epsilon_{m'}\right]. \tag{17}$$

Theorem 7 implies if we can improve the returns under the model $\eta^{\text{branch}}[\pi]$ by more than $C(\epsilon_m,\epsilon_\pi,\epsilon_{m'},k)$, the policy improvement under the true returns can be guaranteed. Further the authors also prove that the optimal $k = \arg\min_k C(\epsilon_m,\epsilon_\pi,\epsilon_{m'},k) > 0$ for sufficiently low $\epsilon_{m'}$, which indicates that roll-out in the learned model with $k^\star$ steps is better than sampling full trajectories or discarding the model. Further, bidirectional model-based policy optimization (BMPO) [33] adopts a backward model to reduce $C(\epsilon_m,\epsilon_\pi,\epsilon_{m'},k)$. BMPO introduces a backward dynamic model $q(s_t|s_{t+1},a_t)$ and a backward policy

$\pi(a_t|s_{t+1})$ to accomplish backward trajectory sampling. Starting from a state s_t, BMPO will sample k_1 steps backward rollouts and k_2 steps forward rollouts. The policy will be learned from the mixture of the forward rollouts, backward rollouts, and real environment data. The true returns of the policy learned by BMPO will be bounded by

$$\eta[\pi] \geqslant \eta^{\text{branch}}[\pi] - C^{\text{BMPO}}(\epsilon_m, \epsilon_\pi, \epsilon_{m'}, k_1, k_2),$$
$$C^{\text{BMPO}}(\epsilon_m, \epsilon_\pi, \epsilon_{m'}, k_1, k2) = 2r_{\max}\left[\frac{\gamma^{k_1+k_2+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k_1+k_2}\epsilon_\pi}{1-\gamma} + \frac{\max(k_1,k_2)}{1-\gamma}\epsilon_{m'}\right].$$

Note that the bound of MBPO with the same rollout length to BMPO is $C(\epsilon_m, \epsilon_\pi, \epsilon_{m'}, k_1 + k_2)$. It is evident that BMPO obtains a tighter upper bound of the return discrepancy by employing bidirectional models.

Typically, MBPO uses a short rollout length to avoid large compounding errors, which may limit the model usage. Recently, Pan et al. [91] proposed a masked model-based actor-critic (M2AC), which can choose a longer rollout length to leverage the model better by discarding the samples with high uncertainty. M2AC computes the uncertainty of each sample by measuring the disagreement between one model versus the rest of the models in an ensemble model, and stores only a batch of samples with the least uncertainty. Lin et al. [92] proposed MPPVE to estimate a multi-step plan value function. MPPVE updates the value function avoiding gradient propagation through the multi-step plan, thus reducing the effect of the model-error.

Dyna-style methods can integrate model learning and MFRL naturally. These methods have an impressive performance as well as a theoretical bound. As a result, Dyna-style algorithms attract lots of research interest in the MBRL community. A common and significant problem for these methods is how to tackle or alleviate the compounding errors. How to use the model to generate more reliable data and how to make better use of the imagined data are still open problems.

3.3 Gradient generation with white box model simulation

In the former subsections, we regard the dynamic model as a black box, with which we can transfer a state to another conditioned on an action. However, in many MBRL scenarios, the dynamic models are differentiable. A dynamic model could be a neural network [93], Gaussian process [94], or a differentiable physics engine [95]. We can utilize the internal structure of the models to facilitate policy learning. In this subsection, we will introduce the approaches that use a white box dynamic model for policy learning. We list two categories of these approaches: differential planning and value gradient, both of which use the internal structure of the model to plan or learn a policy.

Differential planning. Planning in a white box model could be more data-efficient. The Monte Carlo trials discussed in Subsection 3.1 can be altered by gradient-based search. In some cases, we can obtain the analytic form of the optimal policy for an MDP. Linear quadratic regulator (LQR) [96, 97] studies the MDP where the dynamic is linear, and the reward is quadratic. Particularly, the dynamic function is a linear function of state and action. Meanwhile, the reward function is a quadratic function of state and action. In this case, the optimal policy for each step is a linear function of the current state and can be derived from the parameters of the dynamic and reward functions. In the non-linear model, we can approximate the dynamic model to the first order and the cost function to the second order. LQR can then be applied to the approximated model, which is known as iterative LQR (iLQR) [98, 99]. As a result, we can linearize a learned dynamic model and use iLQR to determine the approximately optimal action at each step [100–102]. Guided policy search (GPS) [101] uses iLQR to draw samples from a white-box model. The samples are used in two ways: producing an initial neural network policy by BC and updating the policy via policy gradient. GPS is a successful integration of planning and RL. The GPS framework has been generalized to the case where the dynamic model is unknown [102], or the input is high-dimensional images [103,104]. In recent work, Zhang et al. [105] proposed stochastic optimal control with latent representations (SOLAR) based on the GPS framework. SOLAR defeats an MPC method [106] in the task of learning image-based control policy on a real robot. This result indicates the promising potential of GPS for solving real-world tasks.

Another way of differential planning is utilizing the gradient of the dynamic model for action sequences search. Eq. (13) can be optimized by gradient descent methods if the dynamic model is differentiable. Srinivas et al. [107] proposed universal planning networks (UPN), which involves a gradient descent

planner (GDP). The GDP uses gradient descent to optimize an action sequence to reach a goal. In UPN, the reward is related to the distance between the final state $s_{t+\tau}$ and a goal state s_g. The reward is given at the last step of the planned rollout. Thus, in UPN, Eq. (13) is instantiated by

$$\min_{a_{t:t+\tau}} \ \mathbb{E}_{s_{t'+1}=f(s_{t'},a_{t'})}\|s_{t+\tau}-s_g\|_2^2, \tag{18}$$

where $f(s_{t'},a_{t'})$ is the dynamic model. In fact, we can write $s_{t+\tau}$ as iteratively calling the dynamic model:

$$s_{t+\tau}=f(\cdots f(f(f(s_t,a_t),a_{t+1}),a_{t+2}),\ldots,a_{t+\tau-1}). \tag{19}$$

As $f(s_{t'},a_{t'})$ is differentiable, the gradient of (18) can be passed to $a_{t:t+\tau-1}$ through the dynamic model. GDP alters the Monte Carlo search in MPC to gradient-based search, which is more data-efficient. Despite the high data efficiency, gradient-based methods are prone to sticking to local minimums. However, sample-based methods, e.g., CEM, do not suffer from this problem. Bharadhwaj et al. [108] integrated CEM and gradient-based search to optimize the action sequence. For each CEM iteration, they will use a gradient-based search to refine the action sequences sampled by CEM before the sequences are evaluated. Their experiments show that their method can also avoid the local minima. Compared with CEM, their methods can converge faster and obtain better or equal performance.

Value gradient. The policy gradient can also be passed through a white-box model. Probabilistic inference for learning control (PILCO) [94] models the dynamic model by the Gaussian process [109]. PILCO contains four stages, (1) learning a probabilistic Gaussian process dynamic model from data; (2) evaluating the policy via approximate inference in the learned model; (3) obtaining the gradient of the policy w.r.t. the policy evaluation; (4) updating the policy parameters to maximize the policy evaluation via conjugate gradient or L-BFGS [110]. The training process iterates between collecting data using the current policy and improving the policy. Although PILCO has a graceful mathematics form for the policy gradient and is able to estimate the model uncertainty with by Gaussian process naturally, its GP model is hard to scale in high-dimensional environments. To improve the scalability of PILCO, Gal et al. [111] replaced the GP with a Bayesian neural network to model the dynamic [112]. They scale PILCO to high dimensions as well as retain the probabilistic nature of the GP model.

For the dynamic model instantiated by a common neural network, e.g., fully-connected or convolutional neural network, the policy gradient can be estimated by backpropagating through the learned model [113]. These methods typically involve two parts: re-parameterization of distributions [114, 115] and policy gradient backpropagation through a model. We will take stochastic value gradient (SVG) [93] as an example. SVG re-parameterizes the policy $\pi(a|s)$ by regarding the policy as deterministic function of state s and a noise η: $a=\pi(s,\eta;\theta)$, where $\eta\in\rho(\eta)$ is a random vector, θ is the parameter of π. Similarly, the dynamic model is $s'=f(s,a,\xi;\phi)$, where $\xi\in\rho(\xi)$ is a random vector, ϕ is the parameter of f. For any state-action sequence $(s_t,a_t,s_{t+1},a_{t+1},\ldots)$, we can infer the corresponding $(\eta_t,\xi_t,\eta_{t+1},\xi_{t+1},\ldots)$ with $\pi(s,\eta)$ and $f(s,a,\xi)$, such that $\pi(s_{t'},\eta_{t'};\theta)=a_{t'}$, $f(s_{t'},a_{t'},\xi_{t'};\phi)=s_{t'+1}$, $\forall t'\in\{t,t+1,\ldots\}$. Further, the value function $V^\pi(s_t)$ can be estimated by H-step return plus a parameterized value function $v(\cdot)$:

$$\begin{aligned} V^\pi(s_t)&=\sum_{t'=t}^{t+H-1}\left[\gamma^{t'-t}r(s_{t'},a_{t'})+\gamma^H v(s_{t+H})\right],\quad a_{t'}=\pi(s_{t'},\eta_{t'};\theta),\\ s_{t'+1}&=f(s_{t'},a_{t'},\xi_{t'};\phi)=f\big(f(s_{t'-1},a_{t'-1},\xi_{t'-1};\phi),\pi(s_{t'},\eta_{t'};\theta),\xi_{t'}\big)=\cdots. \end{aligned} \tag{20}$$

Note that each state $s_{t'}$ can be obtained by recursively calling the dynamic function like (19). As a result, we can write $V^\pi(s_t)$ as a function of $s_t,(\eta_t,\xi_t,\eta_{t+1},\xi_{t+1},\ldots)$, parameterized by θ,ϕ:

$$V^\pi(s_t)=F\left(s_t,\eta_t,\xi_t,\eta_{t+1},\xi_{t+1},\ldots;\theta,\phi\right). \tag{21}$$

As the objective of RL is maximizing $V^\pi(s_t)$, the policy gradient at s_t is $-\nabla_\theta V^\pi(s_t)$. In SVG(1), SVG($\infty$) [93], and SVG-$H$ [116], the value function is estimated with 1, ∞, and H-step return. Moreover, the estimation is based on real-world trajectories. It will introduce a likelihood ratio term for the model predictions and increase the variance of the gradient estimate. Model-augmented actor-critic (MAAC) [117], dreamer [56], and imagined value gradients (IVG) [118], instead, entirely rely on the predictions of the model, removing the need for likelihood ratio terms. Particularly, to estimate the action or state values, MAAC will sample an H-step rollout in the model with the current policy. Additionally,

MAAC estimates the action-value function, i.e. Q-function $Q^{\pi}(s,a)$, rather than the state value function $V^{\pi}(s)$. It has been proven that the gradient error of MAAC can be bounded by model and Q-function errors. Clavera et al. [117] further gave the lower bounds of improvement for MAAC in terms of model error and function error, which implies that the policy can be improved monotonically with a precise dynamic model and an accurate Q-function.

As neural network models are broadly used for model learning, the white box model-based methods can be naturally applied to these models. These methods can directly calculate the gradient of the RL objective w.r.t. the policy of action sequences. As a result, these methods have the potential to alter the trials-and-errors exploration and policy improvement paradigm to gradient descent. However, these methods currently suffer from gradient bias because of the model error. How to reduce the gradient bias and variance is a future direction of these kinds of methods.

3.4 Value-aware and policy-aware model learning

Previous MBRL studies mainly treat model learning and model usage separately, which may result in a mismatch of learning objectives between models and policies. That is, the model is trained to give accurate predictions on all training data, while the policy is optimized to achieve high performance in the true environment. Therefore, a model with a small prediction error on the training dataset does not always imply a policy with high rewards [119]. To this end, Farahmand et al. [120] proposed the value-aware model learning (VAML) framework to address this problem by incorporating value function information into model learning. Intuitively, when the model-generated data is used to update value functions, we should focus more on the estimation error of the value target rather than the error of the next state. To be more specific, VAML optimizes the model to minimize the one-step value estimation difference between using environment and model:

$$\mathcal{L}_V(\hat{p},p,\mu)=\int \mu(s,a)\left|\int p\left(s'\mid s,a\right)V\left(s'\right)\mathrm{d}s'-\int \hat{p}\left(s'\mid s,a\right)V\left(s'\right)\mathrm{d}s'\right|^2\ \mathrm{d}(s,a). \tag{22}$$

By minimizing the above loss, the bootstrap target calculated using the model will be close to that using the true environment. Voelcker et al. [121] further improved VAML by taking the gradient of value function into account, and proposed to learn a model that is more accurate on the state-action pairs with large value function gradients. Similarly, for policy gradient based algorithm [122], the model should provide accurate estimate of policy gradient:

$$\nabla_\theta \hat{J}(\theta)=\mathbb{E}_{(s,a)\sim\hat{\rho}_{\pi_\theta}}\left[Q^{\pi_\theta}(s,a)\nabla_\theta\log\pi_\theta(a\mid s)\right], \tag{23}$$

and can be optimized by minimizing the difference between $\nabla_\theta\hat{J}(\theta)$ and the ground-truth $\nabla_\theta J(\theta)$. By integrating value or policy information into model learning, the learned model can be more suitable for current updates and more robust than traditional maximum likelihood methods, especially when the model capacity is insufficient to fully represent the environment [121].

4 Model-based methods in other forms of RL

4.1 Offline RL

Offline RL studies the methodologies that enable the agent to directly learn an effective policy from an offline experience dataset without any interaction with the environment dynamics [123]. In general, given a collected experience dataset $\mathcal{D}=\{(s,a,r,s')\}$, the whole offline RL processing can be framed as

$$\min_{\pi}\mathcal{L}(\mathcal{D},\pi), \tag{24}$$

where the design of the loss function $\mathcal{L}$ is the focus of different offline RL work. With such a setting, the agent will not be required to interact with the environment before a satisfying policy has been learned. Thus, offline RL techniques can be applied to a much wider range of real-world applications.

Despite the non-interaction nature of offline RL making its training objective (24) similar to that of supervised learning, the key challenge of offline RL is the extrapolation error, also studied as an out-of-distribution (OOD) problem from the data perspective, caused by the discrepancy between the underlying behavior policy generating the dataset and the current learning policy [124].

Model-free offline RL mainly designs algorithms that are constrained by the offline dataset to avoid extrapolation errors [124–127] without using extra data. As a result, the learned policies are usually conservative since the dataset itself always limits the appropriate generalization of the learning policy beyond the offline dataset [35].

Model-based offline RL, on the other hand, first builds an environment model based on the offline dataset and then trains the policy based on the model (and the data). The key advantage of building the environment model in offline RL is to leverage the model's generalization ability to perform a certain level of exploration and also to generate additional training data to improve the policy performance.

Nevertheless, since the offline data are often quite limited, the learned model is considered untrustworthy. As a result, most methods take a conservative strategy. MORel [128] constructs a pessimistic MDP (P-MDP) using the offline dataset, where the state transition model is additionally trained and used for detecting whether the current state-action pair is OOD. If OOD is detected, the model will transit to a terminal state and output a negative reward. As such, the agent in the P-MDP tends to learn to avoid OOD situations and thus reduce the extrapolation error in the learning process. MOPO [129] derives a policy value lower bound based on a learned model and incorporates a penalty term into the reward function based on the model uncertainty so as to discourage the agent from entering the OOD region. To bypass the error caused by regarding uncertainty estimated via the deep neural networks as the guidance of avoiding OOD problems, COMBO [130] trains a value function based on both the offline data and the rollout data generated by the learned model. Moreover, the value function is regularized on the OOD data generated via model rollout.

Other than the conservative strategies that avoid entering into OOD regions, it is possible to generalize. MAPLE [131] was the first to borrow the generalization ability of meta-RL for offline RL. It derives an adaptable policy through a meta-RL method to make adaptive decisions directly in OOD regions, which provides an alternative way to deal with OOD data.

4.2 Goal-conditioned RL

Goal-conditioned RL (GCRL), also named goal-oriented RL [132], deals with the tasks where agents are expected to achieve different goals [133] in an environment or complete a complex task via achieving a series of goals. In GCRL, the observation (or the state) of the agent is usually augmented with a goal, which is normally represented as a mapped vector $g \in \mathcal{G}$ from a target state, i.e., $g = \phi(s_{\text{target}})$. The mapping function $\phi : \mathcal{S} \mapsto \mathcal{G}$ can be designed based on specific tasks or just the identity function. The resulting goal can be regarded as being sampled from a distribution p_g. Then, the reward function is defined based on the (state, action, goal) tuple as $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \mapsto \mathbb{R}$. In such a setting, the agent policy $\pi : S \times \mathcal{G} \mapsto \Omega(\mathcal{A})$ is trained to maximize the expected goal-conditioned return as

$$\max_{\pi} J(\pi) = \mathbb{E}_{p,p_g,\pi}\left[\sum_t \gamma^t r(s_t, a_t, g)\right]. \tag{25}$$

To efficiently train the agent to achieve various goals, goal relabeling techniques are widely used. In hindsight, experience replay (HER) [134], the goal is relabeled from the trajectories where the original goal may not be achieved, which is motivated by learning from failure cases. Specifically, in HER, the relabeled goals are built via mapping a randomly picked state in a trajectory of the replay buffer. Other following studies seek different ways to generate the goal, including using GANs to generate goals with different difficulty scores [135] and planning the goals with search techniques based on experience data [136, 137].

To further enhance the diversity of the generated goals, thus the robustness and effectiveness of the trained goal-conditioned policy, recent attempts have been made for goal planning via model-based methods [138] studied visual prediction models to build environment dynamics based on visual observations to enable subgoal generation and planning for robot manipulation tasks, which demonstrates substantial performance gain over the baseline MFRL methods and planning methods without subgoals. Zhu et al. [139] proposed the foresight goal inference (FGI) method to plan goals based on a learned environment dynamics model, then the simulated trajectories are generated based on the goal, goal-conditional policy, and the dynamics model. With model-based RL methods, the training scheme achieves superior sample efficiency than model-free GCRL baselines.

With the high-fidelity environment dynamics model, it is promising to leverage various techniques such as graph search, model inference, and heuristics created by human domain knowledge to generate highly

useful goals to train the goal-conditioned policy and guide the policy's decision-making at the inference stage.

4.3 Multi-agent RL

Multi-agent RL (MARL) studies the sequential interaction strategies among a set of agents $i = 1, 2, \ldots, n$ in the environment, where each agent i is self-interested and aims to maximize its own payoff in terms of expected return as

$$\max_{\pi^i} J^i(\pi^i, \pi^{-i}) = \mathbb{E}_{p,\pi^i,\pi^{-i}}\left[\sum_{t=0}^{\infty} \gamma^t r^i(s_t, a_t^i, a_t^{-i})\right], \tag{26}$$

where the state transition dynamics $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^n \mapsto \Omega(\mathcal{S})$ now depends on the joint action (a_t^i, a_t^{-i}) from all the interacting agents, $\mathcal{A}^i$ denotes the action space of agent i, $r^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^n \mapsto \mathbb{R}$ denotes the reward function for agent i.

Different from single-agent RL, an extra dynamics when seeking the solution in MARL comes from the non-stationarity of the multi-agent game [140]. Ideally, in a Markov game, the Markov perfect equilibrium (MPE) is a profile of policies of the participating agents, where each agent i has no incentive to change its current policy π^i since for each state s, $J^i(\pi^i, \pi^{-i}) \geqslant J^i(\tilde{\pi}^i, \pi^{-i}), \forall \tilde{\pi}^i$ [141]. Ideally, the MPE is the solution sought by MARL algorithms, while an approximate α-MPE is a scenario where every agent yields a value that is within an α margin of the value of its MPE policy [142].

In recent work, Subramanian et al. [142] performed an early theoretic analysis on model-based MARL. Particularly, they first proved that if a Markov game (i.e., the real multi-agent environment) is approximated by another game (i.e., the model of the multi-agent environment), then the MPEs from these two games are close to each other. Then, they derived that $\tilde{O}(|\mathcal{S}||\mathcal{A}|(1-\gamma)^{-2}\alpha^{-2})$ samples are sufficient to achieve an α-MPE with high probability. More specifically, for the two-agent zero-sum Markov game, Zhang et al. [143] derived the sample complexity of the model-based MARL methods and showed that it is lower than the complexity of model-free MARL methods as derived in previous work [144].

From the perspective of one agent, the environment it interacts with consists of the opponent agents and the environment dynamics which transits the state according to the joint actions of all agents. As such, the task of learning the multi-agent environment can be decoupled as learning opponent models and the environment dynamics. Opponent modeling [145] is a well-studied topic in multi-agent RL, while the work on environment dynamics learning in multi-agent RL is rare. Mahajan et al. [146] studied the problem of dimension explosion issue of the action space caused by the number agents and proposed model-based Tesseract, where Bellman equation is built as a tensorised form while the reward function and state transition dynamics are realized by low-rank Canonical-Polyadic decomposition. The theoretic analysis shows that the model-based Tesseract achieves an exponential gain in sample efficiency of $O(|\mathcal{A}|^{n/2})$.

From the analysis of [147], the sample efficiency of MARL can be decomposed into two parts, i.e., dynamics sample complexity, which measures the amount of interactions with the real environment, and the opponent sample complexity, which measures the amount of interactions between the ago agent i and other agents $\{-i\}$. With this regard, it is natural to derive the value discrepancy of the agent policy in the multi-agent environment model (i.e., with the state dynamics model and the opponent models) and the real environment with respect to the error terms of the state dynamics model and opponent models when training the policy via Dyna-style model rollout. The bound shows that the opponent models with higher modeling error contribute larger to the discrepancy bound, which motivates the design of the algorithm called adaptive opponent-wise rollout policy optimization (AORPO) [147]. Specifically, the rollout scheme of AORPO allows the opponent models with lower generalization errors to sample longer trajectories while the shorter sampled trajectories can be supplemented with a communication protocol with real opponents. Kim et al. [148] proposed a communication mechanism called intention sharing (IS), where each agent builds the environment dynamics and opponent models and generates the rollout trajectory. Then a compressed representation is learned from the rollout trajectory to carry the intention, which is sent as a message to other agents for better coordination.

As stated in a recent survey on model-based MARL [149], the research in this direction has just started with only a few established studies. The potential topics that are promising for the future development of model-based MARL include improving the scalability of centralized methods, and the new design of decentralized methods and communication protocols based on the learned models.

4.4 Meta-RL

Meta-RL [150–152] studies the methodologies that enable the agent to generalize across different tasks with few-shot samples in the target tasks. In this process, we have a set of tasks for policy training, but the deployed tasks are unknown, can be OOD compared with the distribution of the training tasks [34], and even can be varied when deployed [153]. Tasks have different definitions in different scenarios, e.g., differences in reward functions [154,155], or parameters of dynamics [156,157]. The challenge is to design efficient mechanisms to extract suitable information for the tasks and adjust the behavior of the policy based on the information.

Model-based meta-RL methods learn dynamics models from the training tasks, adapt the dynamics models to the target tasks via few-shot samples, and finally generate actions via model predictive control (MPC) [158] algorithms or a meta-policy (also called contextual policy) trained with RL methods [34]. Nagabandi et al. [159] considered the task distribution is non-stationary when deployed, e.g., legged robots might lose a leg, face to novel terrains and slopes when deployed. It solves the problem by learning an adaptive predictive model $\hat{p}_{\theta'}(s_{t+1}|s_t, a_t)$ with parameters θ', where $\theta' = u_\phi(\mathcal{D}_{\text{test}}, \theta)$ corresponds to model parameters that were updated using an adapter u_ϕ parameterized by ϕ and the collected dataset $\mathcal{D}_{\text{test}}$. ϕ and θ are trained to use the passed M-step data points to compute an optimal θ' which will minimize the negative log-likelihood $\mathcal{L}$ of future K-step data points:

$$\min_{\theta,\phi} \mathbb{E}_{\tau(t-M,t+K)\sim\mathcal{D}} \left[\mathcal{L}(\tau(t, t+K), \theta')\right], \quad \text{s.t. } \theta' = u_\phi(\tau(t-M, t-1), \theta),$$

where $\tau(t-M, t+K) \sim \mathcal{D}$ corresponds to trajectory segments from $t-M$ timestep to $t+K$ timestep sampled from our previous experience. It implements two adapters, i.e., gradient-based adaptive learner (GrBAL), which uses a gradient-based meta-learning as in [154] to perform online adaptation, and recurrence-based adaptive learner (ReBAL), which utilizes a recurrent model to learn to adapt via the gradient of $\mathcal{L}$. When deployed, it runs an MPC to generate actions using the adapted model $\hat{p}_{\theta'}$. MOLe [160] extends the approach to lifelong learning scenarios, i.e., continual online learning from an incoming stream of data. MOLe develops and maintains a mixture of models to handle non-stationary task distributions. The mixture of models is a set of dynamics models split by a task indicator $\hat{p}_{\theta(T_i)}(s_{t+1}|s_t, a_t)$, where T_i denotes a task. It designs an expectation-maximization (EM) algorithm to update all of the model parameters. When deployed, new models will be instantiated for task changes, and old models will be recalled when previously seen tasks are encountered again. CaDM [34] defines the tasks via the dynamics parameters c (e.g., friction) and aims to learn a forward dynamics model from a set of training environments with contexts sampled from $p_{\text{train}}(c)$ that can produce accurate predictions for test environments with unseen contexts sampled from $p_{\text{test}}(c)$. CaDM solves the problem via learning a context-aware dynamics model $\hat{p}_\theta(s_{t+1}|s_t, a_t, z_t)$, where $z_t = u_\phi(\tau(t-M, t-1))$, which is similar to the framework of ReBAL. CaDM proposes three auxiliary tasks, including forward prediction, backward prediction, and future-step prediction, to regularize the representation of z for better generalization ability. Guo et al. [161] further improved the generalization ability of the meta-dynamics model by reducing the redundancy information in the latent variables z. Specifically, in addition to a prediction loss, Guo et al. [161] also introduced a relational loss to force the latent variables corresponding to the same environments being similar. As a result, the learned latent variables were more related to the environments and enjoyed less redundant information than the methods solely considering the prediction loss [34,162]. Belkhale et al. [163] considered a specific application that a policy should control a flight with suspended unknown payloads. As a solution, a context-aware dynamics model $\hat{p}_\theta(s_{t+1}|s_t, a_t, z_t)$ is also constructed to be aware of different payloads. Belkhale et al. [163] inferred the context via a Gaussian with diagonal covariance $\mathcal{N}(\mu_t, \Sigma_t) \approx p_\phi(z \mid \tau_{:t-1})$ and formulated the meta-objective based on variational inference [114].

The basic idea of learning to adapt has been used to solve the dynamics gap, also called reality gap, between training and testing in many real-world applications. OpenAI et al. [164] taught an adaptive controller to solve a Rubik's cube with a humanoid robot hand in a real-world environment with disturbances. Miki et al. [165] learned an adaptive controller for quadrupedal robots which can make robust perceptive locomotion in an assortment of challenging natural and urban environments over different seasons. All of these studies rely on a latent state z to achieve adaptation. However, the generalization ability of z itself is seldom considered, which can be further investigated in future work. Recent applications of model-based meta-RL also have shown its generalization ability in complex tasks [166,167]. These studies also show that model-based meta-RL with MPC is easy to introduce extra constraints of

safety to action generation, which can be a potential advantage of model-based meta-RL compared with model-free methods.

In particular, when the agents have access to a simulator and deal with tasks from both simulation and reality, it relates to Sim2Real [156,168–171], which focuses on how to transfer a policy trained in the simulator to the real world. In such a case, model learning can be easier by making use of the off-the-shelf simulator. For example, Golemo et al. [172] proposed training a neural network to predict the difference between real data and simulated data. The output of the network was then used to compensate for the unreal parts of the simulator. Experiments have shown this neural-augmented method can be more efficient than directly learning a model to predict the next state. In vision-based RL, Chen et al. [171] trained a mapping function to align the representation space of high-dimensional observations in the real environment to low-dimensional state space in the simulator; then a policy can be trained in the simulator and deployed in the real world directly, by mapping observed images from the real world to aligned states and feeding the inferred states into the trained policy. Moreover, Hwangbo et al. [173] trained an actuator network to produce reasonable predictions for the actuator of real robots since the actuator model of the simulator is not accurate enough. Furthermore, the SimGAN framework [174] utilized GAN [175] to generate the simulation parameters, e.g., actual motor forces or contact forces, which were then used to replace the corresponding components of the simulator. As a result, the simulated data can be closer to the real data compared with an original simulator or fully learned dynamics models. In more complicated real-world scenarios, learning an accurate model can be a big challenge due to the high-dimensional state-action space and complex interactions between different objects.

In the future, it is tempting to build more realistic hybrid models combining analytical simulators and neural networks, which can seamlessly take advantage of innovations in both fields.

4.5 Automated methods on model learning and usage

Compared with MFRL, the design and tuning of MBRL methods tend to require more human effort due to their complex algorithm procedure and sensitivity to different hyperparameters. Take Dyna-style methods as an example. Besides designing the model-free counterpart, Dyna-style methods should also consider alternate optimization of model and policy, model planning steps, and the ratio of simulated data to real data [176]. To this end, automated MBRL methods have been investigated to automate the MBRL pipeline and search for better hyperparameters. For example, the reinforcement on reinforcement (RoR) framework proposed by Dong et al. [177] additionally trained a high-level controller through DQN [178] to control the sampling and training process of a Dyna-style MBRL method. Zhang et al. [179] utilized population-based training (PBT) to optimize the hyperparameters of the PETS algorithm [26] during the training process.

In recent work, Lai et al. [176] theoretically analyzed the role of automated hyperparameter scheduling in Dyna-style MBRL, which reveals that a dynamic schedule of data ratio can be more effective than the fixed one. Motivated by such an analysis, they proposed the AutoMBPO framework to automatically schedule the key hyperparameters of the MBPO [8] algorithm. Empirical results show that MBRL methods with automatic hyperparameter optimization can achieve higher sample efficiency compared with those tuned manually [176,179]. How to better incorporate advanced techniques of AutoML [180] into MBRL is a promising direction to investigate further.

5 Applications of model-based RL

MBRL is of particular interest due to its potential application in the real world, where error intolerance is a common characteristic. This feature contradicts the fundamental mechanism of RL methods, namely trial and error. Therefore, training policies must have a playground between real-world applications and RL. For RL to be freely trained, the training environment must have a high level of realism and error tolerance.

In cost-sensitive scenarios, such as autonomous driving [181], industrial control [182], decision optimization traffic control [183], electricity allocation [184] in smart cities, financial trading [185], and control of tokamak plasma [186], a manually crafted simulator may resemble the overall functionality of the real-world task, but it is difficult to achieve a high level of accuracy. The policy trained in a custom-built simulator may not directly apply to the real-world task. This reality-gap can be bridged using the meta-RL methods described previously, in which a meta-policy can be trained to adapt on the

fly to the task. Using data collected from the real-world task, a single, unrealistic simulator may also be used to facilitate policy training [187]. Simulators are also useful for generating specific situations in real-world environments for learning robust policies [181, 188, 189].

Simulators created manually are still expensive and time-intensive to build. Learning environment models from data can be a more effective and cost-effective alternative to custom-built simulators. Virtual-Taobao, an environment model for recommender systems, was first discovered by Shi et al. [46]. The environment model consists of extremely difficult-to-learn customer behaviors, such as clicking, buying, turning the next page, and leaving after reading recommended items. It was demonstrated in [46] that the proposed adversarial model learning method MAIL can accurately model customer behavior, which was later shown to have a smaller compounding error in Subsection 2.3.2. Shang et al. [190] extended the MAIL method to model hidden factors, thereby enhancing the model's ability to learn. Real-world A/B tests demonstrate that the policies trained in the learned models can be implemented in real-world tasks while maintaining the same performance as in the learned models.

From the aforementioned real-world applications, we observe additional practical benefits of model-based methods.

• **Full release of RL capability.** A simulator or a learned model enables any RL algorithm to train a good policy with sufficient explorations. Even if the model is unrealistic, it is possible to constrain the exploration (e.g., [46, 186]) so that the learned policies remain effective.

• **Pre-deployment assessment.** Before a policy is implemented, it must be thoroughly evaluated. However, evaluating a revised policy is extremely challenging. An improved policy can easily derive a state-action distribution that differs from the collected data, as opposed to supervised learning scenarios that typically employ a dataset with identical distribution to validate prediction models. The objective of the off-policy evaluation is to evaluate a policy using non-identical distributed historical datasets. However, current non-policy evaluation methods have not demonstrated reasonable effectiveness in [191]. While a recent study begins to combine off-policy evaluation and model-based policy evaluation [192], running a policy in a simulator/model may be the simplest way to evaluate its effectiveness.

• **Decision explanation.** Running a policy in a simulator/model allows us to not only evaluate the policy's performance but also observe the specific decisions at each state. These decisions are also extremely useful for the decision-maker to subjectively evaluate the policy's credibility. When decisions demonstrate good rationality or even better ideas, the policy can gain the decision maker's trust, which is crucial in practice.

6 Conclusion and future directions

In this survey, we examined MBRL, a classic tabular RL method in the 1990s and has recently experienced a renaissance for deep RL. Sample efficiency has always been the focus of RL optimization, especially in the era of deep learning. To achieve cutting-edge sample efficiency in deep RL, model-based methods play a crucial role. Based on our investigation, we have observed several MBRL developments. Below is a summary of the directions.

• **Learning generalizable models.** As a playground, models must permit the execution of arbitrary policy. Generalizability is the key to MBRL's success. Recent advancements have included the incorporation of causal learning into model learning. A correct causal structure [193], as well as improved modeling of causal effect [194], can aid in the development of effective models. Causal model learning demonstrates a path to robust model generalization.

• **Learning abstract models.** State abstraction [195] and temporal abstraction [196] can map the original MDP to a low-dimensional and compact MDP in which RL tasks can be significantly simplified. Using state and temporal abstraction, model learning can occur in a low-dimensional space, making it a simple task. We have observed a few opportunities for learning abstract models [197, 198], but much more research is required. Abstractions lead naturally to hierarchical RL, which we find to be a nearly unexplored topic.

• **Training generalizable policies.** As discussed previously, meta-RL relies on model randomization and generates a meta-policy that can be generalized to similar environments. Model variations are the source of the meta-ability policies to generalize. Nevertheless, the question of how to generate models so that the trained meta-policy adapts to the target environment is largely ignored.

• **Model-based multi-agent RL.** Employing various multi-agent scenarios with a multi-agent environment model is in its infancy. Incorporating model-based methods to improve the coordination among team agents and increase the sample efficiency of training has great potential, as demonstrated by a review of recently published studies. However, the model requires additional research into multi-agent environment learning, planning, and communication mechanism design.

• **Foundation models.** In the machine learning community as a whole, learning foundation models is a recently emerging learning paradigm [199] that exhibits strong performance across a wide variety of vision and natural language processing tasks. By mastering a single policy model, this paradigm is also shifting in decision-making tasks [200]. In addition to foundation policy models, foundation environment models represent a vast area to be investigated.

Other potential directions include enhancing value discrepancy bounds, automatic scheduling in MBRL, adaptive model usage, and lifelong model learning [201]. In the near future, it is reasonable to anticipate a series of breakthroughs toward more efficient and applicable RL techniques in model-based RL.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2020AAA0107200) and National Natural Science Foundation of China (Grant Nos. 61876077, 62076161).

References

1 Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge: MIT Press, 2018
2 Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529: 484–489
3 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533
4 Syed U, Bowling M, Schapire R E. Apprenticeship learning using linear programming. In: Proceedings of the 25th International Conference on Machine Learning, 2008. 1032–1039
5 Yu Y. Towards sample efficient reinforcement learning. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018. 5739–5743
6 Wang T W, Bao X C, Clavera I, et al. Benchmarking model-based reinforcement learning. 2019. ArXiv:1907.02057
7 Luo Y P, Xu H Z, Li Y Z, et al. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. 2018. ArXiv:1807.03858
8 Janner M, Fu J, Zhang M, et al. When to trust your model: model-based policy optimization. In: Proceedings of the Advances in Neural Information Processing Systems, 2019. 12498–12509
9 Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. In: Proceedings of the 32nd International Conference on Machine Learning, 2015. 1889–1897
10 Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning, 2016. 1928–1937
11 Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. 2017. ArXiv:1707.06347
12 Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning. In: Proceedings of the 4th International Conference on Learning Representations, 2016
13 Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 1856–1865
14 Sun W, Jiang N, Krishnamurthy A, et al. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In: Proceedings of the Conference on Learning Theory, 2019
15 Asadi K, Misra D, Kim S, et al. Combating the compounding-error problem with a multi-step model. 2019. ArXiv:1905.13320
16 Sutton R S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Proceedings of the 7th International Conference on Machine Learning, 1990. 216–224
17 Brafman R I, Tennenholtz M. R-MAX-A general polynomial time algorithm for near-optimal reinforcement learning. Journal of Maching Learning Research, 2002, 3: 213–231
18 Jiang N. Notes on Rmax exploration, 2020. https://nanjiang.cs.illinois.edu/files/cs598/note7.pdf
19 Azar M G, Osband I, Munos R. Minimax regret bounds for reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning, 2017. 263–272
20 Zhang Z H, Zhou Y, Ji X Y. Almost optimal model-free reinforcement learning via reference-advantage decomposition. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 15198–15207
21 Jin C, Allen-Zhu Z, Bubeck S, et al. Is Q-learning provably efficient? In: Proceedings of the Advances in Neural Information Processing Systems, 2018. 4868–4878
22 Kurutach T, Clavera I, Duan Y, et al. Model-ensemble trust-region policy optimization. In: Proceedings of the 6th International Conference on Learning Representations, 2018
23 Feinberg V, Wan A, Stoica I, et al. Model-based value estimation for efficient model-free reinforcement learning. 2018. ArXiv:1803.00101
24 Rajeswaran A, Mordatch I, Kumar V. A game theoretic framework for model based reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 7953–7963
25 Nagabandi A, Kahn G, Fearing R S, et al. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2018. 7559–7566
26 Chua K, Calandra R, McAllister R, et al. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Proceedings of the Advances in Neural Information Processing Systems, 2018. 4759–4770
27 Kégl B, Hurtado G, Thomas A. Model-based micro-data reinforcement learning: what are the crucial model properties and which model to choose? In: Proceedings of the 9th International Conference on Learning Representation, 2021
28 Kearns M J, Singh S P. Near-optimal reinforcement learning in polynomial time. Machine Learn, 2002, 49: 209–232
29 Xu T, Li Z N, Yu Y. Error bounds of imitating policies and environments. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 15737–15749
30 Xu T, Li Z N, Yu Y. Error bounds of imitating policies and environments for reinforcement learning. IEEE Trans Pattern Anal Mach Intell, 2022, 44: 6968–6980
31 Edwards A D, Downs L, Davidson J C. Forward-backward reinforcement learning. 2018. ArXiv:1803.10227

32 Goyal A, Brakel P, Fedus W, et al. Recall traces: backtracking models for efficient reinforcement learning. In: Proceedings of the 7th International Conference on Learning Representations, 2019
33 Lai H, Shen J, Zhang W N, et al. Bidirectional model-based policy optimization. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 5618–5627
34 Lee K, Seo Y, Lee S, et al. Context-aware dynamics model for generalization in model-based reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 5757–5766
35 Wang J H, Li W Z, Jiang H Z, et al. Offline reinforcement learning with reverse model-based imagination. 2021. ArXiv:2110.00188
36 Venkatraman A, Hebert M, Bagnell J A. Improving multi-step prediction of learned time series models. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, 2015. 3024–3030
37 Asadi K, Misra D, Littman M L. Lipschitz continuity in model-based reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 264–273
38 Vaserstein L N. Markov processes over denumerable products of spaces, describing large systems of automata. Problemy Peredachi Informatsii, 1969, 5: 64–72
39 Ho J, Ermon S. Generative adversarial imitation learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2016. 4565–4573
40 Zhang Y F, Cai Q, Yang Z R, et al. Generative adversarial imitation learning with neural network parameterization: global optimality and convergence rate. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 11044–11054
41 Wang Y Z, Liu T Y, Yang Z, et al. On computation and generalization of generative adversarial imitation learning. In: Proceedings of the 8th International Conference on Learning Representations, 2020
42 Xu T, Li Z N, Yu Y. On generalization of adversarial imitation learning and beyond. 2021. ArXiv:2106.10424
43 Ghasemipour S K S, Zemel R S, Gu S. A divergence minimization perspective on imitation learning methods. In: Proceedings of the 3rd Annual Conference on Robot Learning, 2019. 1259–1277
44 Ke L Y M, Barnes M, Sun W, et al. Imitation learning as f-divergence minimization. 2019. ArXiv:1905.12888
45 Zhang H F, Wang J, Zhou Z M, et al. Learning to design games: strategic environments in reinforcement learning. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018. 3068–3074
46 Shi J C, Yu Y, Da Q, et al. Virtual-Taobao: virtualizing real-world online retail environment for reinforcement learning. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019. 4902–4909
47 Wu Y H, Fan T H, Ramadge P J, et al. Model imitation for model-based reinforcement learning. 2019. ArXiv:1909.11821
48 Eysenbach B, Khazatsky A, Levine S, et al. Mismatched no more: joint model-policy optimization for model-based RL. 2021. ArXiv:2110.02758
49 Zhang W N, Yang Z Y, Shen J, et al. Learning to build high-fidelity and robust environment models. In: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2021. 104–121
50 Tamar A, Glassner Y, Mannor S. Optimizing the CVaR via sampling. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, 2015
51 Spaan M T. Partially observable Markov decision processes. In: Proceedings of the Reinforcement Learning, volume 12 of Adaptation, Learning, and Optimization, 2012. 387–414
52 Ha D, Schmidhuber J. Recurrent world models facilitate policy evolution. In: Proceedings of the Advances in Neural Information Processing Systems, 2018. 2455–2467
53 Hausknecht M, Stone P. Deep recurrent Q-learning for partially observable MDPs. In: Proceedings of the AAAI Fall Symposium Series, 2015
54 Yang M J, Nachum O. Representation matters: offline pretraining for sequential decision making. In: Proceedings of the 38th International Conference on Machine Learning, 2021. 11784–11794
55 Oh J, Singh S, Lee H. Value prediction network. In: Proceedings of the Advances in Neural Information Processing Systems, 2017. 6118–6128
56 Hafner D, Lillicrap T P, Ba J, et al. Dream to control: learning behaviors by latent imagination. In: Proceedings of the 8th International Conference on Learning Representations, 2020
57 Hafner D, Lillicrap T P, Norouzi M, et al. Mastering Atari with discrete world models. In: Proceedings of the 9th International Conference on Learning Representations, 2021
58 Hafner D, Lillicrap T P, Fischer I, et al. Learning latent dynamics for planning from pixels. In: Proceedings of the 36th International Conference on Machine Learning, 2019. 2555–2565
59 Shen J, Zhao H, Zhang W N, et al. Model-based policy optimization with unsupervised model adaptation. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 2823–2834
60 Moerland T M, Broekens J, Jonker C M. A framework for reinforcement learning and planning. 2020. ArXiv:2006.15009
61 Moerland T M, Broekens J, Jonker C M. Model-based reinforcement learning: a survey. 2020. ArXiv:2006.16712
62 Camacho E F, Alba C B. Model Predictive Control. Berlin: Springer, 2013
63 Hewing L, Wabersich K P, Menner M, et al. Learning-based model predictive control: toward safe learning in control. Annu Rev Control Robot Auton Syst, 2020, 3: 269–296
64 Wang T W, Ba J. Exploring model-based planning with policy networks. In: Proceedings of the 8th International Conference on Learning Representations, 2020
65 Botev Z I, Kroese D P, Rubinstein R Y, et al. The cross-entropy method for optimization. In: Proceedings of the Handbook of Statistics, 2013. 31: 35–59
66 Hansen N. The CMA evolution strategy: a tutorial. 2016. ArXiv:1604.00772
67 Yu Y, Qian H, Hu Y Q. Derivative-free optimization via classification. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence, 2016. 2286–2292
68 Hu Y Q, Qian H, Yu Y. Sequential classification-based optimization for direct policy search. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017. 2029–2035
69 He J, Suau M, Oliehoek F A. Influence-augmented online planning for complex environments. In: Proceedings of the Advances in Neural Information Processing Systems, 2020
70 Oliehoek F A, Witwicki S J, Kaelbling L P. Influence-based abstraction for multiagent systems. In: Proceedings of the 26th Conference on Artificial Intelligence, 2012
71 Oliehoek F, Witwicki S, Kaelbling L. A sufficient statistic for influence in structured multiagent environments. J Artif Intell Res, 2021, 70: 789–870
72 Congeduti E, Mey A, Oliehoek F A. Loss bounds for approximate influence-based abstraction. In: Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems, 2021. 377–385
73 Racanière S, Weber T, Reichert D P, et al. Imagination-augmented agents for deep reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2017. 5690–5701
74 Browne C B, Powley E, Whitehouse D, et al. A survey of monte carlo tree search methods. IEEE Trans Comput Intell AI Games, 2012, 4: 1–43

75 Chaslot G, Bakkes S, Szita I, et al. Monte-Carlo tree search: a new framework for game AI. In: Proceedings of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference, 2008
76 Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. Nature, 2017, 550: 354–359
77 Silver D, Hubert T, Schrittwieser J, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017. ArXiv:1712.01815
78 Anthony T, Tian Z, Barber D. Thinking fast and slow with deep learning and tree search. In: Proceedings of the Advances in Neural Information Processing Systems, 2017. 5360–5370
79 Couëtoux A, Hoock J, Sokolovska N, et al. Continuous upper confidence trees. In: Proceedings of the 5th International Conference on Learning and Intelligent Optimization, 2011. 433–445
80 Moerland T M, Broekens J, Plaat A, et al. A0C: Alpha zero in continuous action space. 2018. ArXiv:1805.09613
81 Coulom R. Computing "Elo ratings" of move patterns in the game of Go. J Int Comput Games Assoc, 2007, 30: 198–208
82 Chaslot G M J B, Winands M H M, Herik H J V D, et al. Progressive strategies for Monte-Carlo tree search. New Math Nat Computation, 2008, 04: 343–357
83 Schrittwieser J, Antonoglou I, Hubert T, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. 2019. ArXiv:1911.08265
84 Sutton R S. Dyna, an integrated architecture for learning, planning, and reacting. SIGART Bull, 1991, 2: 160–163
85 Moore A W, Atkeson C G. Prioritized sweeping: reinforcement learning with less data and less time. Machine Learning, 1993, 13: 103–130
86 Tamar A, Levine S, Abbeel P, et al. Value iteration networks. In: Proceedings of the Advances in Neural Information Processing Systems, 2016. 2146–2154
87 Bellman R. Dynamic programming and stochastic control processes. Inf Control, 1958, 1: 228–239
88 Tesauro G, Galperin G R. On-line policy improvement using Monte-Carlo search. In: Proceedings of the Advances in Neural Information Processing Systems, 1996. 1068–1074
89 Tesauro G. Temporal difference learning and TD-Gammon. Commun ACM, 1995, 38: 58–68
90 Buckman J, Hafner D, Tucker G, et al. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In: Proceedings of the Advances in Neural Information Processing Systems, 2018. 8234–8244
91 Pan F Y, He J, Tu D D, et al. Trust the model when it is confident: masked model-based actor-critic. In: Proceedings of the Advances in Neural Information Processing Systems, 2020
92 Lin H X, Sun Y H, Zhang J J, et al. Model-based reinforcement learning with multi-step plan value estimation. 2022. ArXiv:2209.05530
93 Heess N, Wayne G, Silver D, et al. Learning continuous control policies by stochastic value gradients. In: Proceedings of the Advances in Neural Information Processing Systems, 2015. 2944–2952
94 Deisenroth M P, Rasmussen C E. PILCO: a model-based and data-efficient approach to policy search. In: Proceedings of the 28th International Conference on Machine Learning, 2011. 465–472
95 Degrave J, Hermans M, Dambre J, et al. A differentiable physics engine for deep learning in robotics. Front Neurorobot, 2019, 13: 6
96 Kwakernaak H, Sivan R. Linear Optimal Control Systems. New York: John Wiley & Sons, Inc., 1972
97 Todorov E, Li W. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proceedings of the American Control Conference, 2005. 300–306
98 Li W, Todorov E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In: Proceedings of the 1st International Conference on Informatics in Control, 2004. 222–229
99 Tassa Y, Erez T, Todorov E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012. 4906–4913
100 Watter M, Springenberg J T, Boedecker J, et al. Embed to control: a locally linear latent dynamics model for control from raw images. In: Proceedings of the Advances in Neural Information Processing Systems, 2015. 2746–2754
101 Levine S, Koltun V. Guided policy search. In: Proceedings of the 30th International Conference on Machine Learning, 2013. 1–9
102 Levine S, Abbeel P. Learning neural network policies with guided policy search under unknown dynamics. In: Proceedings of the Advances in Neural Information Processing Systems, 2014. 1071–1079
103 Levine S, Wagener N, Abbeel P. Learning contact-rich manipulation skills with guided policy search. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2015. 156–163
104 Levine S, Finn C, Darrell T, et al. End-to-end training of deep visuomotor policies. J Machine Learning Res, 2016, 17: 1–40
105 Zhang M, Vikram S, Smith L, et al. SOLAR: deep structured representations for model-based reinforcement learning. In: Proceedings of the 36th International Conference on Machine Learning, 2019. 7444–7453
106 Ebert F, Finn C, Dasari S, et al. Visual foresight: model-based deep reinforcement learning for vision-based robotic control. 2018. ArXiv:1812.00568
107 Srinivas A, Jabri A, Abbeel P, et al. Universal planning networks: learning generalizable representations for visuomotor control. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 4739–4748
108 Bharadhwaj H, Xie K, Shkurti F. Model-predictive control via cross-entropy and gradient-based optimization. In: Proceedings of the 2nd Annual Conference on Learning for Dynamics and Control, 2020. 277–286
109 Seeger M. Gaussian processes for machine learning. Int J Neur Syst, 2004, 14: 69–106
110 Peters J, Schaal S. Policy gradient methods for robotics. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006. 2219–2225
111 Gal Y, McAllister R, Rasmussen C E. Improving PILCO with Bayesian neural network dynamics models. In: Proceedings of the 33rd International Conference on Machine Learning Workshop on Data-Efficient Machine Learning Workshop, 2016. 25
112 Mackay D J C. Bayesian methods for adaptive models. Dissertation for Ph.D. Degree. Pasadena: California Institute of Technology, 1992
113 Mohamed S, Rosca M, Figurnov M, et al. Monte Carlo gradient estimation in machine learning. J Machine Learning Res, 2020, 21: 5183–5244
114 Kingma D P, Welling M. Auto-encoding variational Bayes. In: Proceedings of the 2nd International Conference on Learning Representations, 2014
115 Rezende D J, Mohamed S, Wierstra D. Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st International Conference on Machine Learning, 2014. 1278–1286
116 Amos B, Stanton S, Yarats D, et al. On the model-based stochastic value gradient for continuous reinforcement learning. In: Proceedings of the 3rd Annual Conference on Learning for Dynamics and Control, 2021. 6–20
117 Clavera I, Fu Y, Abbeel P. Model-augmented actor-critic: backpropagating through paths. In: Proceedings of the 8th International Conference on Learning Representations, 2020
118 Byravan A, Springenberg J T, Abdolmaleki A, et al. Imagined value gradients: model-based policy optimization with

transferable latent dynamics models. 2019. ArXiv:1910.04142
119 Lambert N, Amos B, Yadan O, et al. Objective mismatch in model-based reinforcement learning. 2020. ArXiv:2002.04523
120 Farahmand A M, Barreto A, Nikovski D. Value-aware loss function for model-based reinforcement learning. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017. 1486–1494
121 Voelcker C A, Liao V, Garg A, et al. Value gradient weighted model-based reinforcement learning. In: Proceedings of the 10th International Conference on Learning Representations, 2021
122 Abachi R. Policy-aware model learning for policy gradient methods. Dissertation for Ph.D. Degree. Toronto: University of Toronto, 2020
123 Levine S, Kumar A, Tucker G, et al. Offline reinforcement learning: tutorial, review, and perspectives on open problems. 2020. ArXiv:2005.01643
124 Kumar A, Zhou A, Tucker G, et al. Conservative Q-learning for offline reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2020
125 Fujimoto S, Meger D, Precup D. Off-policy deep reinforcement learning without exploration. In: Proceedings of the 36th International Conference on Machine Learning, 2019. 2052–2062
126 Peng X B, Kumar A, Zhang G, et al. Advantage-weighted regression: simple and scalable off-policy reinforcement learning. 2019. ArXiv:1910.00177
127 Chen X Y, Zhou Z J, Wang Z, et al. BAIL: best-action imitation learning for batch deep reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 18353–18363
128 Kidambi R, Rajeswaran A, Netrapalli P, et al. MOReL: model-based offline reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 21810–21823
129 Yu T, Thomas G, Yu L, et al. MOPO: model-based offline policy optimization. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 14129–14142
130 Yu T, Kumar A, Rafailov R, et al. COMBO: conservative offline model-based policy optimization. In: Proceedings of the Advances in Neural Information Processing Systems, 2021
131 Chen X H, Yu Y, Li Q Y, et al. Offline model-based adaptable policy learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2021. 8432–8443
132 Liu M H, Zhu M H, Zhang W N. Goal-conditioned reinforcement learning: problems and solutions. 2022. ArXiv:2201.08299
133 Pitis S, Chan H, Zhao S, et al. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 7750–7761
134 Andrychowicz M, Crow D, Ray A, et al. Hindsight experience replay. In: Proceedings of the Advances in Neural Information Processing Systems, 2017. 5048–5058
135 Florensa C, Held D, Geng X, et al. Automatic goal generation for reinforcement learning agents. In: Proceedings of the 35th International Conference on Machine Learning, 2018. 1514–1523
136 Lai Y Q, Wang W F, Yang Y J, et al. Hindsight planner. In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, 2020. 690–698
137 Eysenbach B, Salakhutdinov R, Levine S. Search on the replay buffer: bridging planning and reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2019. 15220–15231
138 Nair S, Finn C. Hierarchical foresight: self-supervised learning of long-horizon tasks via visual subgoal generation. In: Proceedings of the 8th International Conference on Learning Representations, 2020
139 Zhu M H, Liu M H, Shen J, et al. MapGo: model-assisted policy optimization for goal-oriented tasks. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence, 2021. 3484–3491
140 Papoudakis G, Christianos F, Rahman A, et al. Dealing with non-stationarity in multi-agent deep reinforcement learning. 2019. ArXiv:1906.04737
141 Fink A M. Equilibrium in a stochastic n-person game. Hiroshima Math J, 1964, 28: 89–93
142 Subramanian J, Sinha A, Mahajan A. Robustness and sample complexity of model-based MARL for general-sum Markov games. 2021. ArXiv:2110.02355
143 Zhang K, Kakade S M, Basar T, et al. Model-based multi-agent RL in zero-sum Markov games with near-optimal sample complexity. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 1166–1178
144 Bai Y, Jin C. Provable self-play algorithms for competitive reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 551–560
145 He H, Boyd-Graber J, Kwok K, et al. Opponent modeling in deep reinforcement learning. In: Proceedings of the 33rd International Conference on Machine Learning, 2016. 1804–1813
146 Mahajan A, Samvelyan M, Mao L, et al. Tesseract: tensorised actors for multi-agent reinforcement learning. In: Proceedings of the 38th International Conference on Machine Learning, 2021. 7301–7312
147 Zhang W N, Wang X H, Shen J, et al. Model-based multi-agent policy optimization with adaptive opponent-wise rollouts. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence, 2021
148 Kim W, Park J, Sung Y. Communication in multi-agent reinforcement learning: intention sharing. In: Proceedings of the 9th International Conference on Learning Representations, 2021
149 Wang X H, Zhang Z C, Zhang W N. Model-based multi-agent reinforcement learning: recent progress and prospects. 2022. ArXiv:2203.10603
150 Duan Y, Schulman J, Chen X, et al. RL^2: fast reinforcement learning via slow reinforcement learning. 2016. ArXiv:1611.02779
151 Houthooft R, Chen Y, Isola P, et al. Evolved policy gradients. In: Proceedings of the Advances in Neural Information Processing Systems, 2018. 5405–5414
152 Yu Y, Chen S Y, Da Q, et al. Reusable reinforcement learning via shallow trails. IEEE Trans Neural Netw Learn Syst, 2018, 29: 2204–2215
153 Luo F M, Jiang S Y, Yu Y, et al. Adapt to environment sudden changes by learning a context sensitive policy. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence, 2022
154 Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning, 2017. 1126–1135
155 Rothfuss J, Lee D, Clavera I, et al. ProMP: proximal meta-policy search. In: Proceedings of the 7th International Conference on Learning Representations, 2019
156 Peng X B, Andrychowicz M, Zaremba W, et al. Sim-to-real transfer of robotic control with dynamics randomization. In: Proceedings of the 34th IEEE International Conference on Robotics and Automation, 2018. 1–8
157 Zhang C, Yu Y, Zhou Z H. Learning environmental calibration actions for policy self-evolution. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018. 3061–3067
158 Williams G, Aldrich A, Theodorou E A. Model predictive path integral control using covariance variable importance sampling. 2015. ArXiv:1509.01149
159 Nagabandi A, Clavera I, Liu S, et al. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In: Proceedings of the 7th International Conference on Learning Representations, 2019

160 Nagabandi A, Finn C, Levine S. Deep online learning via meta-learning: continual adaptation for model-based RL. In: Proceedings of the 7th International Conference on Learning Representations, 2019
161 Guo J X, Gong M M, Tao D C. A relational intervention approach for unsupervised dynamics generalization in model-based reinforcement learning. In: Proceedings of the 10th International Conference on Learning Representations, 2022
162 Seo Y, Lee K, Gilaberte I C, et al. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2020
163 Belkhale S, Li R, Kahn G, et al. Model-based meta-reinforcement learning for flight with suspended payloads. IEEE Robot Autom Lett, 2021, 6: 1471–1478
164 OpenAI, Akkaya I, Andrychowicz M, et al. Solving Rubik's cube with a robot hand. 2019. ArXiv:1910.07113
165 Miki T, Lee J, Hwangbo J, et al. Learning robust perceptive locomotion for quadrupedal robots in the wild. Sci Robot, 2022. doi: 10.1126/scirobotics.abk2822
166 Chen B M, Liu Z X, Zhu J C, et al. Context-aware safe reinforcement learning for non-stationary environments. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2021
167 Zhang J, Cheung B, Finn C, et al. Cautious adaptation for reinforcement learning in safety-critical settings. In: Proceedings of the 37th International Conference on Machine Learning, 2020. 11055–11065
168 Yu W, Tan J, Liu C K, et al. Preparing for the unknown: learning a universal policy with online system identification. 2017. ArXiv:1702.02453
169 Tan J, Zhang T N, Coumans E, et al. Sim-to-real: learning agile locomotion for quadruped robots. 2018. ArXiv:1804.10332
170 Rusu A A, Večerík M, Rothörl T, et al. Sim-to-real robot learning from pixels with progressive nets. In: Proceedings of the 1st Annual Conference on Robot Learning, 2017. 262–270
171 Chen X H, Jiang S Y, Xu F, et al. Cross-modal domain adaptation for cost-efficient visual reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, Virtual Event, 2021. 12520–12532
172 Golemo F, Taiga A A, Courville A, et al. Sim-to-real transfer with neural-augmented robot simulation. In: Proceedings of the 2nd Conference on Robot Learning, 2018. 817–828
173 Hwangbo J, Lee J, Dosovitskiy A, et al. Learning agile and dynamic motor skills for legged robots. Sci Robot, 2019, 4: 5872
174 Jiang Y F, Zhang T N, Ho D, et al. SimGAN: hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2021. 2884–2890
175 Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets. In: Proceedings of the Advances in Neural Information Processing Systems, 2014. 2672–2680
176 Lai H, Shen J, Zhang W N, et al. On effective scheduling of model-based reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 2021. 3694–3705
177 Dong L S, Li Y L, Zhou X, et al. Intelligent trainer for dyna-style model-based deep reinforcement learning. In: Proceedings of the IEEE Transactions on Neural Networks and Learning Systems, 2020
178 Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. 2013. ArXiv:1312.5602
179 Zhang B, Rajan R, Pineda L, et al. On the importance of hyperparameter optimization for model-based reinforcement learning. In: Proceedings of the 24th International Conference on Artificial Intelligence and Statistics, 2021. 4015–4023
180 Hutter F, Kotthoff L, Vanschoren J. Automated Machine Learning: Methods, Systems, Challenges. Berlin: Springer, 2019
181 Zhou M, Luo J, Villela J, et al. SMARTS: an open-source scalable multi-agent RL training school for autonomous driving. In: Proceedings of the 4th Conference on Robot Learning, 2020. 264–285
182 Hein D, Depeweg S, Tokic M, et al. A benchmark environment motivated by industrial control problems. In: Proceedings of the IEEE Symposium Series on Computational Intelligence, 2017. 1–8
183 Zhang H C, Feng S Y, Liu C, et al. CityFlow: a multi-agent reinforcement learning environment for large scale city traffic scenario. In: Proceedings of the World Wide Web Conference, 2019. 3620–3624
184 Vázquez-Canteli J R, Kämpf J, Henze G, et al. Citylearn v1.0: an OpenAI Gym environment for demand response with deep reinforcement learning. In: Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, 2019. 356–357
185 Liu X Y, Yang H Y, Chen Q, et al. FinRL: a deep reinforcement learning library for automated stock trading in quantitative finance. 2020. ArXiv:2011.09607
186 Degrave J, Felici F, Buchli J, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. Nature, 2022, 602: 414–419
187 Jiang S, Pang J C, Yu Y. Offline imitation learning with a misspecified simulator. In: Proceedings of the Advances in Neural Information Processing Systems, 2020
188 Chou G, Sahin Y E, Yang L, et al. Using control synthesis to generate corner cases: a case study on autonomous driving. IEEE Trans Comput-Aided Des Integr Circuits Syst, 2018, 37: 2906–2917
189 Sun H W, Feng S, Yan X T, et al. Corner case generation and analysis for safety assessment of autonomous vehicles. Transp Res Record, 2021, 2675: 587–600
190 Shang W J, Li Q Y, Qin Z W, et al. Partially observable environment estimation with uplift inference for reinforcement learning based recommendation. Mach Learn, 2021, 110: 2603–2640
191 Qin R J, Gao S Y, Zhang X Y, et al. NeoRL: a near real-world benchmark for offline reinforcement learning. 2021. ArXiv:2102.00714
192 Jin X K, Liu X H, Jiang S, et al. Hybrid value estimation for off-policy evaluation and offline reinforcement learning. 2022. ArXiv:2206.02000
193 Zhu Z M, Chen X H, Tian H L, et al. Offline reinforcement learning with causal structured world models. 2022. ArXiv:2206.01474
194 Chen X H, Yu Y, Zhu Z M, et al. Adversarial counterfactual environment model learning. 2022. ArXiv:2206.04890
195 Dietterich T G. State abstraction in MAXQ hierarchical reinforcement learning. In: Proceedings of the Advances in Neural Information Processing Systems, 1999. 994–1000
196 Sutton R S, Precup D, Singh S. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artif Intelligence, 1999, 112: 181–211
197 Jiang N, Kulesza A, Singh S. Abstraction selection in model-based reinforcement learning. In: Proceedings of the 32nd International Conference on Machine Learning, 2015. 179–188
198 Zhu Z M, Jiang S, Liu Y R, et al. Invariant action effect model for reinforcement learning. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence, 2022
199 Bommasani R, Hudson D A, Adeli E, et al. On the opportunities and risks of foundation models. 2021. ArXiv:2108.07258
200 Reed S E, Zolna K, Parisotto E, et al. A generalist agent. 2022. ArXiv:2205.06175
201 Wu B, Gupta J K, Kochenderfer M. Model primitives for hierarchical lifelong reinforcement learning. Auton Agent Multi-Agent Syst, 2020, 34: 28