ECE 8803: Online Decision Making in Machine Learning **Fall 2021**

Lecture 24: November 22

Lecturer: Vidya Muthukumar

Until now, we have engaged with the so-called *tabular RL* setting, in which MDPs are modeled with a finite number of states and actions. In this setting, all RL methods scale, both in statistical and computational complexity, directly proportional to the size of the state space, $|\mathcal{S}|$, and the size of the action space $|\mathcal{A}|$. This is, of course, a massive improvement over the exponential scaling in state-space size ($|\mathcal{A}|^{|\mathcal{S}|}$) that a naive optimal-policy-search would necessitate.

However, even linear-in-state-space-size time is often not acceptable for several modern RL applications, where the state-space size may be on the order of a billion or even a trillion. In applications that use RL in conjunction with other ML technologies (such as computer vision for self-driving cars), the state space may be high-dimensional and continuous in nature (for example, the pixels of an image), in which case tabular RL is not even applicable out-of-the-box.

Of course, if the parameters of such large MDPs (i.e. the transitions and probabilities) are really independently set, then we cannot do much better than simply using this extremely large tabular MDP. However, in most of these real-world settings, there is ostensibly some sort of *common structure* across the states and actions, allowing us to infer about rewards and transitions of "neighboring" states and actions without visiting them explicitly. Concretely, there may be a lower-dimensional (typically unknown) parameter that itself expresses the entire MDP, value function, or policy space through a function (such as a linear model, or a neural network). The problem of trying to infer this unknown parameter, or more directly, estimate an optimal policy through this lower-dimensional parameterization, is commonly called *function approximation in RL*.

In today's lecture, we will take a brief tour through function approximation techniques in RL. The first part of the lecture will touch upon different ML models that are used in function approximation as well as different types of approximation (in MDP space, value space or policy space). The second part of the lecture will highlight key challenges that are introduced with function approximation in the realms of generalization, exploration and optimization. Determining the exact best types of function approximation for a RL application is an ongoing research area both in theory and practice. Through this exposition, I will try and give a sense of where we are at, and what remains to be done.

24.1. Where function approximation is applied

Over the past few weeks, we have seen three types of RL methods: a) model-based methods (that estimate a MDP and then return an optimal policy from that estimated MDP), b)

value-based methods (that estimate the value, or Q-function, from samples), and c) policy-based methods (that attempt to optimize over policy space directly). Accordingly, function approximation can be applied to the MDP (mostly in the transition probabilities), the value function, or the space of policies. In all cases, the role is one of *dimensionality reduction*: we want to reduce the dependence of sample complexity (or regret) of RL methods from the size of the state and action space ($|\mathcal{S}|, |\mathcal{A}|$) to the effective dimension of the chosen function class. We will see how this happens through this lecture, through a series of examples. We begin with brief, abstract descriptions of how function approximation is used in model-based, value-based, and policy-based methods (assuming a generative/"parallel-sampling" model; we defer issues of exploration to later in this lecture note). In all of the definitions provided below, we will use the abstract notation $f(\cdot)$ to denote a function-approximator. We also link to concrete examples of linear function approximators for each of these, provided in more detail in Section 24.2.1.

Definition 1 *[Model-based function approximation] The model-based function approximation paradigm considers common functions $f_P(\cdot)$ and $f_R(\cdot)$ such that the transition probabilities and rewards respectively can be written as*

$$P(s'|s,a) \approx f_P(s,a,s') \text{ and} \tag{24.1}$$

$$r(s,a) \approx f_R(s,a) \tag{24.2}$$

where the identities of $f_P(\cdot), f_R(\cdot)$ are unknown, but are assumed to lie in known function classes $\mathcal{F}_P, \mathcal{F}_R$. Typically, the effective size/dimension of these function classes will be much smaller than the size of the state-space $|\mathcal{S}|$; therefore, learning $f_P(\cdot), f_R(\cdot)$ from samples will require significantly fewer samples than treating the entries $P(s'|s,a), r(s,a)$ as independent entities.

An example of linear model-based function approximation is the *linear MDP*, defined in Definition 4 later in the lecture. A typical model-based RL algorithm imbued with function approximation would follow a three-step approach as below (ignoring exploration issues for now):

- Estimating functions $\widehat{f}_P(\cdot), \widehat{f}_R(\cdot)$ from samples.
- Plugging these estimates into Equations (24.1) to construct an estimated MDP.
- Compute the optimal policy for the estimated MDP.

Model-based structure is typically the strongest posited structure and will imply value-based and policy-based approximation structure as defined below; it also has the strongest theoretical guarantees. Weaker forms of function approximation include value-based and policy-based function approximation, which are defined below.

Definition 2 *[Value-based function approximation] Value-based function approximation typically approximates **only the optimal** value function $\boldsymbol{V}^*$, or equivalently, the optimal Q-function $\{Q^*(s,a)\}_{(s,a)\in\mathcal{S}\times\mathcal{A}}$. This function approximation model considers a function $f_Q(\cdot) \in \mathcal{F}_Q$ such that*

$$Q^*(s,a) \approx f_Q(s,a) \text{ for all } (s,a) \in \mathcal{S}\times\mathcal{A}. \tag{24.3}$$

As with the model-based case, the identity of $f_Q(\cdot)$ is unknown, but is assumed to lie in a known function class $\mathcal{F}_Q$. Moreover, as in the model-based case, the effective size/dimension of $\mathcal{F}_Q$ will be much smaller than the size of the state-space $|\mathcal{S}|$.

Value-based approximation structure is weaker than model-based approximation structure: we can construct several large tabular MDPs for which Equation (24.3) holds but not Equation (24.1). This makes it a better or worse assumption to make, depending on the underlying structure of the MDP (see Section 24.3.1 more details on these modeling conundrums). An example of value-based approximation structure with a linear model is provided in Equation (24.7).

From an implementation perspective, value-based methods for tabular RL like Q-learning are easily adaptible to the function approximation setting with *parametric* function classes (including neural networks). Let the function used to parameterize the Q-function at round t be parameterized by $\boldsymbol{\theta}_t$. Then, a typical run of Q-learning with function approximation simply updates at step t as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t(r(s,a) + \gamma \max_{a' \in \mathcal{A}} Q_t(s,a') - Q_t(s,a)) \cdot \nabla_{\boldsymbol{\theta}_t} Q_t(s,a). \tag{24.4}$$

It is a worthwhile exercise to verify that this corresponds to the standard Q-learning update in the tabular RL setting.

Definition 3 *[Policy-based function approximation] Finally, policy-based function approximation specifies a policy class by $f_\Pi(\cdot)$. According to this function approximation, we can write the optimal policy as*

$$\pi(a|s) \approx f_\Pi(s,a). \tag{24.5}$$

As with the model-based and value-based cases, the identity of $f_\Pi(\cdot)$ is typically unknown, but is assumed to lie in a known function class $\mathcal{F}_\Pi$.

Policy-based approximation structure is the weakest of the three function approximation frameworks: we can find examples of large tabular MDPs for which neither model-based nor value-based structure holds, but the optimal policy has a relatively simple representation. An example of policy-based approximation structure with linearly parameterized soft-max policies is provided in Equation (24.8).

Algorithms for policy optimization with function approximation are also conceptually simple: they simply operate by choosing $f_\Pi \in \mathcal{F}_\Pi$ as:

$$f_\Pi = \arg\max_{f \in \mathcal{F}_\Pi} V^\pi(s_0),$$

where π is the policy induced by f. Thus, we are now maximizing over a much smaller space ($\mathcal{F}_\Pi$) than the space of all possible policies.

Sometimes, value-based and policy-based function approximation are also used together (as in actor-critic methods).

24.2. Types of approximators

Now that we have completed our (high-level) discussion of types of function approximation in RL, we now describe concrete examples of function approximators and take a closer look at the afforded benefits of function approximation.

24.2.1 Linear function approximation

Linear function approximation is one of the simplest forms of parametric function approximation, and also the one for which the strongest theoretical guarantees exist. In its most basic form, it is a *model-based* approximation and called a linear MDP, formally defined below.

Definition 4 (Linear MDP) *A (stationary) linear MDP is specified with the following ingredients:*

- *A **known** state and action space $\mathcal{S}, \mathcal{A}$, and a **known** feature map $\boldsymbol{\phi}(s,a) \in \mathbb{R}^d$ for any state-action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$.*
- *An **unknown** set of parameters $\boldsymbol{\theta^*} \in \mathbb{R}^d$ and $\boldsymbol{\mu^*} \in \mathbb{R}^{|\mathcal{S}| \times d}$ such that the rewards and transitions respectively can be expressed as*

$$r(s,a) = f_{\boldsymbol{\theta^*}}(s,a) := \langle \boldsymbol{\theta^*}, \boldsymbol{\phi}(s,a) \rangle \tag{24.6a}$$

$$P(\cdot|s,a) = f_{\boldsymbol{\mu^*}}(s,a) := \boldsymbol{\mu^*}\boldsymbol{\phi}(s,a). \tag{24.6b}$$

In a typical linear MDP specification, $d \ll |\mathcal{S}|$ and so this type of function approximation is performing a form of dimensionality reduction. In a tabular MDP, the total number of unknown parameters is $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$. On the other hand, the number of unknown parameters required to specify a linear MDP is given by $\mathcal{O}(|\mathcal{S}|d)$ (i.e. the dimension of $\boldsymbol{\theta^*}$ plus the dimension of $\boldsymbol{\mu^*}$), which is clearly much smaller. In fact, as we will see subsequently, it is possible to derive sample complexity bounds that do not depend on $|\mathcal{S}|$ at all, but only on the latent dimension $d \ll |\mathcal{S}|$. This demonstrates the power of such function approximation, when it is correct.

The dimensionality reduction afforded by linear function approximation is in the specification of the known features $\phi(s,a)$ for state-action pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$: these specify that, while the state/action space may be very large, it only impacts rewards and transitions through its (lower-dimensional) feature map. Implicitly, because the feature map is lower-dimensional, different state-action pairs may have very similar feature mappings (i.e. $\boldsymbol{\phi}(s,a) \approx \boldsymbol{\phi}(s',a')$ even when $(s,a) \neq (s',a')$): so information can be shared across these states and actions. Definition 4 provides one concrete model through which to formalize such structure. Alternative to the linear MDP, we may only have a linear parameterization of the *optimal Q-function*, i.e.

$$Q^*(s,a) = f_{\boldsymbol{\theta^*}}(s,a) := \langle \boldsymbol{\theta^*}, \boldsymbol{\phi}(s,a) \rangle \tag{24.7}$$

for some unknown $\boldsymbol{\theta^*} \in \mathbb{R}^d$ — or only allow a linear (soft-max) parameterization of policies of the form

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{\exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a))}{\sum_{a' \in \mathcal{A}} \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(s,a'))} \tag{24.8}$$

These are much weaker forms of linear function approximations than the linear MDP: there are several known examples for which Equation (24.7) holds, but no linear MDP can be

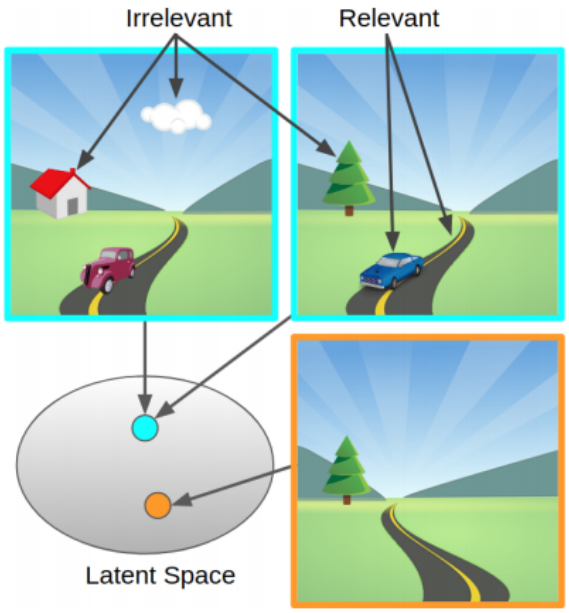

Figure 24.1: A schematic depiction of hidden structure in rich observations: the two distinct observations on the top essentially map to the same latent state, while the bottom observation is a different latent state.

defined according to Definition 4. Accordingly, our guarantees on RL are also significantly weaker for these modeling assumptions; see Section 24.3.1 for a discussion.

The linear model on function approximation at least simplifies some aspects of the RL pipeline: for example, estimating the parameters of the MDP, or the Q-function, is at least straightforward to do via linear regression routines. Thus, the linear model constitutes an excellent mathematical testbed for various RL algorithms deployed with function approximation.

24.2.2 Combining RL and representation learning: The "rich observations" framework

Large state-spaces, rather than large action-spaces, are the more commonly encountered obstacle to making tabular RL methods scale well. In particular, our state may constitute of an observation of the world that is typically quite *rich* in content: it may comprise of, for example, the pixels of an image representing the state of the game, or an observation of traffic by an autonomous vehicle. These *rich observations* nonetheless have several irrelevant pieces of information: for all practical purposes, very different-looking observations might have the same properties. Motivated by this, a common mode of function approximation involves what is called *state abstraction*: learning a mapping from this set of rich observations into a much smaller-sized latent state space. The state abstraction model is formally defined below.

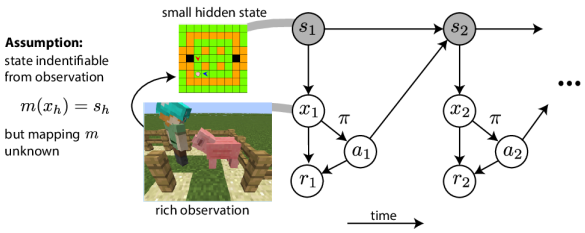

Figure 24.2: An example of state abstraction: observations that are labeled with the same color correspond to the same latent state.

Definition 5 *[State abstraction] Consider a* ***latent state space*** *of size equal to* $K \ll |\mathcal{S}|$*. Corresponding to any state* $s \in \mathcal{S}$*, there exists an* ***unknown*** *encoding (also called representation)* $\phi : \mathcal{S} \to \{1, \ldots, K\}$ *such that for all* $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$*, we have*

$$P(s'|s, a) = P(\phi(s')|\phi(s), a) \text{ and}$$
$$r(s, a) = r(\phi(s), a).$$

An example of this kind of state abstraction is depicted in Figure 24.2 for the case of game playing. A very attractive feature of this model is the *hidden tabular MDP* structure: if we knew the representation $\phi(\cdot)$, then we could apply any of the tabular methods that we have learned about in the last few weeks and obtain RL algorithms with sample complexity scaling as $\mathcal{O}(K)$ (omitting the dependence on the other parameters).

Of course, we typically do not know this representation and need to *learn* it from samples. Typical algorithms for the rich-observation setting in RL combine advances in representation learning (including deep RL) with a state-of-the-art tabular RL algorithm, applied on the *learned* latent-state MDP. Suppose that we are exploring our environment and see a sequence of states and actions $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$. The central challenge in learning a latent representation here is that we *do not observe* the latent states $\phi(s_0), \phi(s_1), \ldots$, and yet we need to learn this representation from data. There are several possible approaches one can take to successfully learn the representation of varying degrees of complexity. A schematic of these approaches is provided in Figure 24.3.

- *Unsupervised* learning methods, that take the size of the latent state space K as input and apply an unsupervised learning method, such as K-means clustering or variational auto-encoders, on the states. If the chosen policy to generate the sequence of states and actions is sufficiently exploratory, we would expect to explore the state-space sufficiently well so that unsupervised learning works. This may require more samples than necessary (and we will see why below), but is the conceptually simplest of the representation learning approaches in RL.

- *Supervised* learning methods, which look at contiguous pairs s_t, s_{t+1} and attempt to learn a mapping $\phi(\cdot)$ that maximizes the likelihood of the observed transition pairs $(s_0, s_1), (s_1, s_2), (s_2, s_3), \ldots$. These would be ostensibly more sample-efficient than unsupervised methods, as they utilize the transition information across rounds. However, they are significantly more complicated to implement efficiently and analyze.

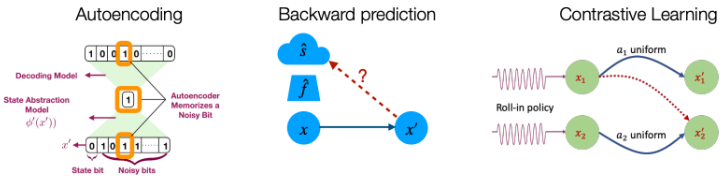

Figure 24.3: A schematic of various approaches to representation learning in RL.

- *Semi-supervised, or contrastive learning* methods, which hit a kind of sweet spot between the complexities of unsupervised and supervised methods. These generate "fake transitions" and mix them in with the true transitions, and label each transition pair with 0 or 1 according to whether the transition was fake or real. Then, a good learned representation would ostensibly be able to effectively distinguish the fake and real transitions. While the contrastive learning approach is conceptually the most difficult to describe (and is an active research area even in simpler ML settings), it has achieved some remarkable successes in difficult RL environments (see the additional notes and references).

To analyze the above approaches, we typically assume that $\phi(\cdot)$ lies in a function class $\mathcal{F}$, and seek algorithms that achieve a sample complexity given by $\mathcal{O}(K \cdot \text{comp}(\mathcal{F}))$, where $\text{comp}(\mathcal{F})$ is some typical complexity measure of learning a representation in $\mathcal{F}$. (For example, if the representation is linear with dimension d, we would expect $\text{comp}(\mathcal{F}) = d$.) Provided that the state-abstraction structure of Definition 5 actually exists, this approach is a powerful, flexible and interpretable approach to plug-and-play advances in representation learning with state-of-the-art tabular RL algorithms.

24.3. Enduring challenges in function approximation

Now that we have provided a brief description of various types of function approximation, we now turn to describing some open challenges in our understanding of function approximation. While several of the challenges we will present here are intellectual in nature (involving a lack of theoretical guarantees), the challenges also have important methodolgical implications for a) what ML model to use for function approximation, b) where to apply the function approximation, and c) which are the best RL algorithms to use in conjunction with function approximation.

Disclaimer: In general, research into RL with function approximation is very active and I expect several details in this note to change a few months from now :-).

24.3.1 Estimation-approximation tradeoffs

Whether the type of function approximation to be applied is model-based, value-based or policy-based, the choice of approximation model is an extremely challenging one in theory and practice, and can have critical ramifications for performance. In particular, both *under-specifying* or *over-specifying* a function approximation model can have significant

consequences. We show how this is the case through two examples (which we only describe in high-level detail).

Example 1 *[Temporal difference learning with linear function approximation] For this example, we consider the problem of* ***policy evaluation*** *with linear function approximation, i.e. evaluating $\boldsymbol{V}^\pi$ for a fixed stationary policy π. In the case of tabular MDPs, policy evaluation can be done efficiently via the popular* ***temporal difference (TD) learning***[1] *algorithm, which makes updates like*

$$V_{t+1}(s) = (1-\alpha_t)V_t(s) + \alpha_t \cdot (r(s,\pi(s)) + \gamma \cdot V_t(s')) \tag{24.9}$$

where $s' \sim P(\cdot|s,\pi(s))$. In the tabular setting, we know that $\boldsymbol{V}_t \to \boldsymbol{V}^\pi$ and we can also derive convergence rates. There is a natural version of TD learning that can be applied with linear function approximation. When the linear MDP model is exact (i.e. Equation (24.6) *holds), this works quite well. However, if the linear MDP model is* not *exact the TD learning is not quite as robust as we'd like it to be. In particular, let $\boldsymbol{V}_{\mathsf{best},d}$ denote the best d-dimensional linear function approximation to $\boldsymbol{V}^\pi$, and $\boldsymbol{V}_{\mathsf{lim}} := \lim_{t\to\infty} \boldsymbol{V}_t$. Then, we have*

$$\|\boldsymbol{V}_{\mathsf{lim}} - \boldsymbol{V}^\pi\|^2 \le \frac{1}{1-\gamma} \cdot \|\boldsymbol{V}_{\mathsf{best},d} - \boldsymbol{V}^\pi\|^2; \tag{24.10}$$

in other words, not only does the TD algorithm ***not*** *converge to the "best-in-class" estimate of the policy evaluation, given by $\boldsymbol{V}_{\mathsf{best},d}$; but also, there is a blow-up in the effective horizon length of the approximation error! This blow-up turns out to be tight and unimprovable for a certain class of iterative algorithms like TD. This is much worse than a standard supervised-learning guarantee, in which we would instead have $f_{\mathsf{lim}} = f_{\mathsf{best},d}$ and we can often also obtain finite-sample rates. This example suggests that algorithm selection needs to be done carefully under model misspecification.*

Example 1 demonstrates the perils of model misspecification in function approximation in RL for a specific example (and for a specific algorithm). This suggests that we do not want to pick an approximation class that is too small—in fact, most theoretical guarantees for RL with function approximation assume *realizability*, i.e. that the function approximation is exact.

This may suggest that we want to pick function approximation classes that are as large as possible, to make the realizability assumption more realistic[2]. On the other hand, too much flexibility in modeling can pose its own problems, as evidenced by our second example below.

Example 2 *[Fundamental limits on learning for linear-Q^*] Here, we consider the linear-Q^* approximation assumption that was described in Equation* (24.7). *As we have discussed, this*

1. The definition that we provide here is called TD(0). More generally, temporal difference learning can be parameterized by a parameter $\lambda \in (0,1)$ that interpolates between the iterative TD(0) approach and a Monte-Carlo type simulation (TD(1)). The details of this are out-of-scope for our purposes.
2. This is possibly one way to interpret the success of deep RL, which involves highly expressive function classes. However, note that most success stories in deep RL involve billions and trillions of simulations, and it would be of great interest to bring this sample complexity down to enable its applicability to more high-stakes scenarios.

is significantly weaker structure than the linear-MDP (model-based) structure. It turns out that there is a price to pay for this weaker structure; it can be shown that any *algorithm needs at least* $\min(e^{\Omega(d)}, \Omega(2^H))$ *samples (in the finite-horizon setting; similar scalings also hold for the effective horizon length in the discounted case) to estimate an approximately optimal policy. In fact, this lower bound holds even when* (s, a) *pairs can be sampled adaptively. This is significantly worse than the sample complexity of* $\text{poly}(d, H)$ *that is afforded by model-based linear-MDP structure.*

The above examples demonstrate that function approximation selection is an important, complex and challenging *model selection problem*. This model selection needs to be done in conjunction with algorithm choice, and is typically a function of the instance and the environment in which we are operating. While our discussion above has focused on challenges related to sample complexity and generalization, exploration and optimization with function approximation also pose significant open challenges. We will not discuss these in detail, but I am happy to chat about them in office hours/discussion.

24.4. Additional references and notes

- Function approximation was first proposed by Bellman in the 1960's in a *computation* context, rather than a learning context. While we have emphasized sample complexity issues with large tabular MDPs (because of $|\mathcal{S}|$ being large), it is worth noting that these same large tabular MDPs would also be difficult to *compute* an optimal policy for, as we need to update (and store) values for all $|\mathcal{S}|$ states at every iteration, regardless of whether DP, value iteration or policy iteration is being used. Experimental evidence was provided in early works that the *optimal* value function could be well-approximated by the linear combination of a few basis functions (i.e very similar to the linear-Q^* assumption); thus motivating the *approximate dynamic programming* approach. Indeed, in early days approximate DP and RL were used almost interchangeably.

- These notes only scratch the surface of function approximation in RL. For an excellent in-depth tutorial, please see the recent tutorial that was given at COLT 2021: `https://rltheorybook.github.io/colt21tutorial`. The tutorial page also contains several references for recent theoretical results in the area.

- Example 1 is from Tsitsiklis and Van Roy (1997) and Example 2 is from Weisz et al. (2021). At a high level, the issues arising in both Examples 1 and 2 are due to error-propagation in dynamic programming across a long horizon (although this manifests in very different ways).

- There are several types of structural assumptions that are in between the extremes of a linear MDP assumption and a linear-Q^* assumption. Jiang et al. (2017) defined a fundamental quantity called the *Bellman rank* and showed that whenever this is low, efficient estimation of an optimal policy is possible. This low-Bellman-rank assumption encapsulates linear MDPs, but also several other function classes (but not the linear-Q^* class). Zanette et al. (2020) showed that the estimation error gracefully degrades with a certain notion of approximation error called the *inherent Bellman error*. While positive results have been shown for these settings (which are significantly richer

than linear MDPs), computational efficiency of the developed algorithms remains an important open question.

References

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2017.

John N Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE transactions on automatic control*, 42(5):674–690, 1997.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory*, pages 1237–1264. PMLR, 2021.

Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR, 2020.