VIPer: Iterative Value-Aware Model Learning on the Value Improvement Path

Anonymous Authors[1]

Abstract

We propose a practical and generalizable Decision-Aware Model-Based Reinforcement Learning algorithm. We extend the frameworks of VAML (Farahmand et al., 2017) and IterVAML (Farahmand, 2018), which have been shown to be difficult to scale to high-dimensional and continuous environments (Lovatto et al., 2020a; Modhe et al., 2021; Voelcker et al., 2022). We propose to use the notion of the Value Improvement Path (Dabney et al., 2020) to improve the generalization of VAML-like model learning. We show theoretically for linear and tabular spaces that our proposed algorithm is sensible, justifying extension to non-linear and continuous spaces. We also present a detailed implementation proposal based on these ideas.

1. Introduction

With the growing success and increasing performance of reinforcement learning methods in recent years, the complexity of environments tackled with these methods has increased correspondingly. The focus of the community has shifted towards leveraging the representational power and flexibility of deep neural networks to tackle challenges in high-dimensional observations, such as playing video games from pixel input (Hafner et al., 2020; 2021) and controlling robots based on cameras (Ibarz et al., 2021). However, using high-dimensional observations has proven to be challenging for applying representation learning or model based techniques, since modelling high dimensional inputs such as video streams with sufficient fidelity to accommodate planning is a challenging task (Lambert et al., 2020; Stone et al., 2021; Tomar et al., 2021).

Therefore, reducing the state space or finding alternative ways to model the environment is an important sub-problem of scaling model-based methods to challenging domains such as video inputs with distracting dimensions or high-dimensional inputs. Recent efforts have focused on using techniques from computer vision to tackle complex vision environment (Yarats et al., 2021), or learning models which aim to capture the underlying decision problems in the MDP (Farahmand et al., 2017; Oh et al., 2017; Farahmand, 2018; Schrittwieser et al., 2020; Zhang et al., 2021; Fu et al., 2021). However, algorithm inspired by computer vision often rely crucially on domain knowledge and heuristics, which leads to failure in cases where these are inadmissible (Tomar et al., 2021; Chen et al., 2021), and decision-centered models can be inefficient in training or fail to generalize.

The aim of this project is to consider representation and model learning as a joint effort, and use our knowledge about representations to directly modify the model learning objective. We consider the framework of *decision-aware model learning* (DAML) (Farahmand et al., 2017), which aims to derive models which are optimal for planning for different sets of policies or value functions. We highlight that by introducing ideas from the representation learning community, we can qualify what sets of value functions and policies to consider for decision-aware model learning. Furthermore, by explicitly considering a range of different value functions as decision-aware targets, we mitigate previously described problems with the Value-aware Model Learning framework. Finally, we show how reasoning about representation learning and decision-aware model learning jointly, we can derive a simple and well-motivated algorithm.

This paper is a Work-in-Progress contribution. In this preliminary report, we present the core framework and theoretical justification for our idea to invite feedback from the community. We leave empirical experiments to a full presentation of the work at a later stage. The main question we seek to address in this paper is how to learn a model which is useful as part of a Dyna update loop.

2. Background

We consider a Markov decision process or MDP M defined as $(\mathcal{X}, \mathcal{A}, P^*, r, \gamma)$, with state space $\mathcal{X}$, action space $\mathcal{A}$, transition kernel $P^* : \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X})$, reward function $r : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. The goal of an agent is to optimize the obtained average discounted infinite horizon reward under its policy: $\max_\pi \mathbb{E}_{\pi,P^*} \left[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t, x_{t+1})\right]$

Given a policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{A})$, the value function is defined as the expected return conditioned on every possible state: $V^\pi(x) = \mathbb{E}_\pi \left[\sum_{t\geq 0} \gamma^t R_t | X_0 = x\right]$, where $R_t = r(X_t, A_t, X_{t+1})$ is the reward at time t. The

action-value function is defined similarly as $Q^\pi(x,a) = \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t R_t | X_0 = x, A_0 = a\right]$. The Bellman operator $\mathcal{T}^\pi$, when applied to Q has the following effect:

$$(\mathcal{T}^\pi Q)(x,a) = \mathbb{E}[r(x,a,X') + \gamma \mathbb{E}_\pi[Q(X',A')]].$$

In this paper, we consider Dyna-based algorithms (Sutton, 1990), in which the environment P^* is replaced with a learned approximate model $\hat{P}$. The learned model is used to gather additional fictitious experience, which can be incorporated into any RL algorithm. In our setting, we specifically consider policy iteration and actor-critic algorithms, such as SAC (Haarnoja et al., 2018). In these, an actor (represented by a policy π) and a critic (an approximate value function $\hat{V}^\pi$) are updated iteratively. The actor maximizes the performance of the policy under the critic, while the critic aims to learn an accurate value function of the policy.

2.1. Decision-aware and value equivalent model learning

Although traditional model-based methods have mostly focused on finding the maximum likelihood estimate for the model, it has been shown that this is not always an optimal design decision. Several recent works discuss variations of this approach for model learning that aim to align the model with the way it is used in planning. Following the nomenclature of Farahmand et al. (2017) we will refer to these different ideas and proposals collectively as *Decision-Aware Model Learning*. One example of DAML is "Value-Aware Model Learning" which we review in the next section.

2.1.1. Value-aware model learning

The *Value-Aware Model Learning* (VAML) framework (Farahmand et al., 2017) proposes to train a model by penalizing the differences in value function prediction under a learned model $\hat{P}$ and the true environment transition model P^*. Given a distribution over the state-action space ν and a value function V, the VAML loss function can be expressed as follows, with notation taken from Voelcker et al. (2022):

$$\mathcal{L}_V(\hat{P}, P^*, \nu) = \int \nu(s,a) \Bigg| \overbrace{\int P^*(s'|s,a)V(s')\mathrm{d}s'}^{\text{environment value estimate}} - \overbrace{\int \hat{P}(s'|s,a)V(s')\mathrm{d}s'}^{\text{model value estimate}} \Bigg|^2 \mathrm{d}(s,a).$$

Since the value function is generally not known during training, Farahmand et al. (2017) proposes to use a worst-case value function from a function class $\mathcal{F}$:

$$\mathcal{L}_{\text{VAML}}(\hat{P}, P^*, \nu) = \sup_{V\in\mathcal{F}} \mathcal{L}_V(\hat{P}, P^*, \nu).$$

The VAML loss function was proposed to directly control the error when using the model to compute the Bellman backup operator, which can be used for a (Fitted) Value Iteration algorithm (Ernst et al., 2005; Gordon, 1995; Munos & Szepesvári, 2008).

The biggest issue with using this loss in practice is choosing a meaningful set of value functions, $\mathcal{F}$, and computing the supremum. In practice, it is common to use neural networks to parameterize the value function. The supremum over this class of flexible models can be very conservative (although still less conservative than the MLE) and is hard to compute in practice.

To mitigate this, Farahmand (2018) proposes to use the most recent value function estimate V_i produced during learning, where $(V_1, \ldots, V_i)$ is the sequence of value estimates produced during the value iteration procedure. This leads to the *Iterative Value-Aware Model Learning* (IterVAML) algorithm:

$$\mathcal{L}_{\text{IterVAML}}(\hat{P}, P^*, \nu) = \mathcal{L}_{V_i}(\hat{P}, P^*, \nu).$$

Intuitively, the current value function estimate is used directly in the Bellman backup computation for the next value function update, so it is sufficient for the model to match this value function. However, several authors (Lovatto et al., 2020b; Voelcker et al., 2022; Modhe et al., 2021) note that the empirical results of IterVAML are lacking. Voelcker et al. (2022) show that the problem reliably appears if the same model is used to compute several value function update steps, as the learned models performance can quickly deteriorate when the value function changes. This inspires us to take a closer look at the value function set choices for learning a decision-aware model.

When looking at IterVAML and VAML as two algorithms from a general family, one addressing the supremum over a whole function space and one operating on a single value function, it is easy to see that we can devise additional algorithms by considering different sets of value functions. This view naturally leads to a connection of the VAML paradigm with recent developments that investigate the geometry of the value function space and the value functions encountered during value function training (Dadashi et al., 2019; Dabney et al., 2020).

2.1.2. The value-equivalence principle

A complementary view on the DAML framework is presented by Grimm et al. (2020) and Grimm et al. (2021). Instead of discussing loss functions for DAML models, they instead characterize the set of models on which the Bellman backup behaves the same as on the true model. To bridge the two perspectives, we can see that the set of Value-Equivalent Models for a given value function is exactly comprised of those that achieve zero error on the VAML loss for the value

function set.

In the *Proper Value Equivalence* approach (Grimm et al., 2021), a loss function for the abstract planning model MuZero (Schrittwieser et al., 2020) is proposed. This loss is motivated as expanding the single value function used in MuZero to include past value function iterates to constrain the set of value-equivalent models further (note that this is different from the Value Improvement Path discussed in the next section, as the past value function iterates are not due to improving policies but are rather improving approximations of the value function for an individual policy). We hypothesize that a similar construction can be used to constrain IterVAML using information about the value function space that our policy iterates on.

2.2. The geometry of the value function set

Dadashi et al. (2019) showed that the space of value functions for an MDP $\mathcal{V} = \{V^\pi : \pi \in \mathcal{P}(\mathcal{A})^\mathcal{X}\}$, i.e. the set of all value functions attainable by some policy, is characterized by a polytope, which we will refer to as the value function polytope. This set is generally different from the set $\mathcal{F}$ considered by Farahmand et al. (2017), which is the set of value functions that can be represented by a certain choice of parameterization or representation. In Figure 1 we illustrate the value function polytope for a 2-state MDP, as well as a particular choice of $\mathcal{F}$ that contains it (which may not be true in general).

In Dabney et al. (2020), it was shown that the sequence of improving value functions goes along a curve in the value function polytope (see Fig 1). The sequence of improving value functions, or "the value improvement path" as referred to by Dabney et al. (2020), is defined as the sequence $(V^{\pi_0}, V^{\pi_1}, \dots, V^{\pi^*})$ where $V^{\pi_{i+1}} \succeq V^{\pi_i}$, that is, $V^{\pi_{i+1}}(x) \geq V^{\pi_i}(x)$ for all $x \in \mathcal{X}$.

We can see that this set is smaller than $\mathcal{V}$, yet captures the value functions that are likely to be important during learning. It is this property that we will use to motivate our approach in subsequent sections.

3. Model generalization along the value improvement path

One crucial problem with value-aware models is ensuring that they are useful for several value function updates. This is a simple fact that is often not clearly stated, so it deserves some motivation and consideration.

As Voelcker et al. (2022) show, IterVAML fails if the model is reused for several value function update steps. They highlight that this happens because the predictions of the model are not guaranteed to be value equivalent for a wider set of value functions. It is possible for the model to predict

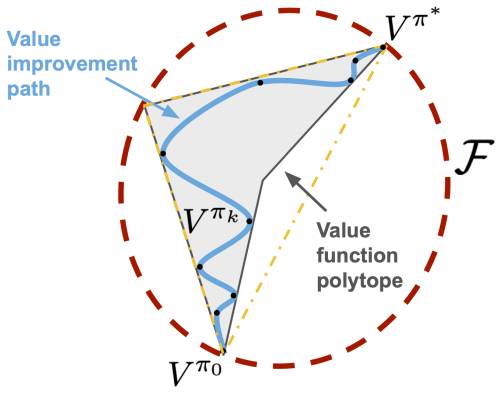

Figure 1. A two dimensional representation of the value function sets discussed. The yellow outline indicates the convex hull of the value function polytope

out-of-distribution states that are value equivalent under the current value function estimate, but wrong when updating the value function. This would not be a problem if the model was only used for a single update step and retrained from scratch after, as was assumed in the theoretical analysis by Farahmand (2018). However, such an algorithm can become prohibitively expensive when using a gradient descent based optimization which requires thousands of updates and have no guarantees of convergence when using neural networks.

In the case that we want to use our model for multiple value function updates, we require that the model generalize to future updates of the value function and choose a sufficiently large set for value equivalence. Model learning should be done so that future value updates result in small model loss and do not invalidate the model's predictions. The insights presented in Section 2.2 allow us to highlight several sets which can be considered to achieve this goal.

3.1. Choosing a proper value function set

In VAML, we explicitly, through the use of supremum, enforce that the model loss be small for any value function representable in the function space. While this ensures that the error in the Bellman backup is bounded for any function choice from the value function space, this may be too conservative in practice. We may never encounter most of the value functions representable in the function space. However, by enforcing that our model and value functions be able to represent them, we lose some of the advantage of being "value-aware" as this may require us to learn many parts of the observation space that will never be useful to the planning problem. To highlight this, consider the following theorem on the VAML loss due to Farahmand et al. (2017), for a set of value functions:

Theorem 3.1 (VAML for linear value function model)**.** *If*

the value-function class is given as $V_\theta(x) = \phi(x)^\top \theta$ with $\|\theta\|_2 \leq B$, the supremum over the value function class can be computed as a difference in expectation over next-state features

$$\begin{aligned}
&\sup_{V_\theta \in \mathcal{F}_\theta} \left\langle \hat{P} - P^*, V_\theta \right\rangle = \\
&\quad = B \left\| \int (\hat{P}(x'|x,a) - P^*(x'|x,a))\phi(x')\mathrm{d}x \right\|_2^2 \\
&\quad = B \left\| \mathbb{E}_{\hat{P}}\left[\phi(X')\right] - \mathbb{E}_{P^*}\left[\phi(X')\right] \right\|_2^2
\end{aligned}$$

This is a conservative loss as it requires full matching of the next states features in expectation. But if some of these features are irrelevant for the task, this is not taken into account.

Following the insights from (Dadashi et al., 2019), we can see that the set of value functions for policies in the actual MDP is constrained in the value function polytope. Furthermore, we can use the structure of this polytope to derive the following result:

Theorem 3.2. *Assume a set of linear value functions $\mathcal{V} = \left\{V(x) = \phi(x)^\top \theta : \theta \in \{\theta_1, \ldots, \theta_m\}\right\}$ and a model $\hat{P}$ with*

$$\forall \theta : \left\| \mathbb{E}_{\hat{P}(\cdot|x,a)}\left[\phi(X')^\top \theta\right] - \mathbb{E}_{P^*(\cdot|x,a)}\left[\phi(X')^\top \theta_i\right] \right\|^2 \leq \epsilon(x,a).$$

Then for any convex combination of the value functions $V(x) = \phi(x)^\top \sum_i \alpha_i \theta_i$ with $\sum_i \alpha_i = 1$ and $\alpha_i \geq 0$, the prediction error of the model wrt the new value function can be bounded as follows:

$$\begin{aligned}
&\left\| \mathbb{E}_{\hat{P}}\left[V(X')\right] - \mathbb{E}_{P^*}\left[V(X')\right] \right\|^2 \\
\leq &\sum_i \alpha_i \left\| \mathbb{E}_{\hat{P}}\left[\phi(X')^\top \theta\right] - \mathbb{E}_{P^*}\left[\phi(X')^\top \theta\right] \right\|^2 \\
\leq &\epsilon(x,a).
\end{aligned}$$

If the value functions V_i are chosen as the vertices of the value function polytope, their convex hull covers the whole polytope. The proof of the second line follows from the linearity of the expectations and an application of Jensen's inequality.

The corresponding VAML loss over this set can be written as the maximum over a finite set of weight vectors θ_i. This allows us to concentrate on those value function that can actually appear for policies in the given MDP, so the set of value-aware models can be substantially larger depending on the actual behavior of value function on the MDP. For example, if the value functions are independent of one of the observation dimensions, the value function polytope will capture this while our set defined a priori might not.

A perfect model under this loss for all vertex policies in the value function polytope also correspond to the largest set of proper value equivalent models inthe framework of (Grimm et al., 2021), as it allows the computation of the full Bellman backup for any policy on the MDP without error.

However, this theorem still has a problem: to cover the whole value function polytope, we now have to compute the value functions for all vertex policies. As Dadashi et al. (2019) shows, this set can be prohibitively large, growing exponentially with the number of states, and is infinite for continuous state-action spaces. Obtaining the value functions also requires us to solve the policy evaluation problem for all deterministic policies, which would require an accurate model for each a priori to obtaining the value functions. If we had information about the shape of the value function polytope before running VAML, we could use this, but for practical algorithms that are able to operate tabula rasa, we need to restrict our set even further.

As highlighted by Dabney et al. (2020), the value functions an agent will encounter follow the value improvement path, which is substantially smaller than the whole value function polytope. In their paper, they conjecture and show empirically that a learned representation over past policies and value functions along the value improvement path contains sufficient information to allow generalization to future value functions. We hypothesize that the same holds for a model which is value equivalent to the set of past value functions.

4. Connecting representation and model learning

To focus on representing the value functions along the value improvement path, we argue that the model and value functions should be learned in the same embedding space, and a model that is learned in this space is sufficiently expressive for the task of learning any value function that can be represented.

Any model learning objective induces a representation learning objective in the underlying state space. For example, in a tabular domain, given certain restrictions on the model class, it can be shown (see Appendix A) that learning an IterVAML model is equivalent to finding the MLE over a Q^*-state abstraction. Given this viewpoint, we can shift our focus and talk about the representations learned by the model, and those learned by the value function. We use $\mathcal{Z}$ to refer to the representation space and assume (for now) that ϕ is a fixed embedding function with $\phi : \mathcal{X} \to \mathcal{Z}$.

Assume the value-function class is $\mathcal{F}_\theta = \{V_\theta(x) = \phi(x)^\top \theta; \|\theta\| \leq B\}$. Note that here we denote the embedding space as a mapping from the observation space through $\phi(\cdot)$, i.e. $\mathcal{Z} = \phi(\mathcal{X})$. Then the VAML model with zero error, considering the result of Theorem 3.1, satisfies

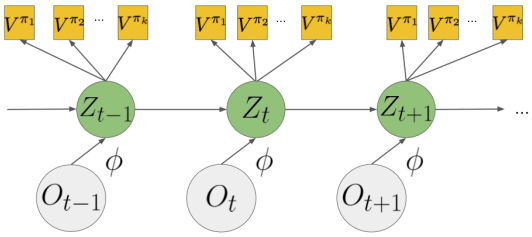

Figure 2. Illustration of the underlying model structure of the ViPer model. At each timestep t, the observation O is encoded into a latent space which serves as the input to a set of value functions for past policies π_i.

$\mathbb{E}_{\hat{P}(\cdot|x,a)}[z'] = \mathbb{E}_{P^*(\cdot|x,a)}[z']$. In other words, instead of a model that acts in the observation space $\mathcal{X}$, an equivalent one acts directly in the representation space, which we denote as a "lifted" model.

Parr et al. (2008) states that if a linear value function approximation over feature embeddings $\mathcal{Z}$ is used, then the least-squares linear model over the same feature space has precisely the same value function for a given policy. Therefore it is sufficient to learn a lifted linear model in the space Z to evaluate any given policy (under the constraint of linear value function approximation).

This result helps us see that if the representation space of the model is shared with the representation space of the value function, there is sufficient flexibility for the optimal model to be learned, and therefore we do not need to learn separate features ϕ for the model and values.

Although the linear argument cannot be transferred easily into the embedding space of a deep neural network, we argue that it is a sensible extension of these results. First of all, if we assume that the value function is represented by a learned function on top of an embedding function, any point in the observation space will be mapped into the embedding space before predicting the value. Therefore, by lifting the model into this space, we cannot lose information that is necessary for representing the value function. In fact, a similar assumption has been made implicitly in works such as VPN (Oh et al., 2017) and MuZero (Schrittwieser et al., 2020). However the losses used in these works are different, and they do not explicitly consider larger value function sets.

We hypothesize that learning a model and value function in the same representation space is sufficient for generalizable models even in the deep learning setting.

In addition, the arguments from Section 3 allow us to expand the simple idea of a lifted model in an embedding space: the exact choice of the embedding and its expressiveness directly corresponds with the expressiveness of the model.

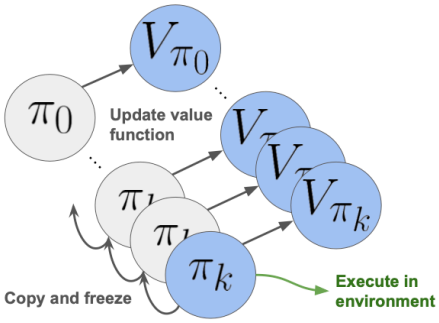

Figure 3. The algorithmic setup of the actor and critic components of our algorithm. Blue indicates quantities that are updated at each step using gradient descent. Past policies are frozen, but the value functions are still updated on new data.

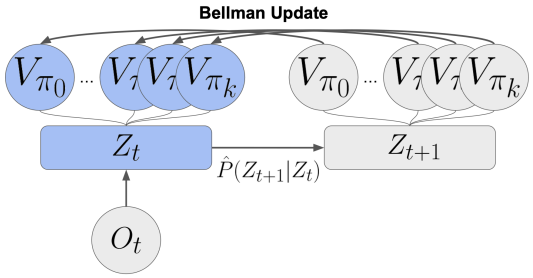

Figure 4. Value and embedding update procedure given a model defined on the embedding space and set of k value functions corresponding to k past policies. The loss function is the soft Bellman error due to Haarnoja et al. (2018).

If the representation space is able to capture the whole value function polytope, a (good) model in it will be equally useful to express the Bellman backup of any policy. If the representation space captures the value functions along the value improvement path, the model will equally be useful to compute value function predictions for value functions contained therein.

To leverage this knowledge, we propose a joint learning of a agent, embedding space and latent model that is built on the insights of the Value Improvement Path. A schematic view of our proposed agent is shown in Figure 2. We discuss details of our setup in the next section.

5. Proposal: Experimental setup

As mentioned in the introduction, this Work-in-Progress report only presents a preliminary framework and theoretical background of the algorithm, which we call *Value Improvement Path Iterative Model Learning* or VIPer. An implementation based on deep neural networks and flexible actor-critic algorithms such as SAC leaves open many design decisions and hyper-parameters. In this section, we

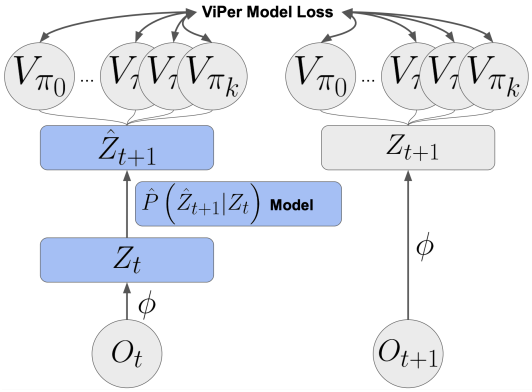

Figure 5. Model updating procedure. Model loss defined in equation 1. Color scheme as above.

describe the algorithm and the planned experiments to investigate the ideas presented in this work.

We hope that our approach stabilizes and speeds up the learning of models on high-dimensional observation environments such as image-based robotics control. Especially in the presence of distracting state information, current model-based algorithms struggle to filter the necessary information from the complicated observation space.

Following the discussions of Sections 3 and 4, we present the following experimental setup for verifying our hypotheses. Referring to Figure 3, an agent interacts with the environment using a policy π_i, and its policy parameters are updated. At regular intervals between policy updates, the policy parameters are "frozen", and copied and stop becoming updated. As a result, a set of k policies are maintained in memory, where the kth policy is the behaviour policy, and is updated using an actor-critic algorithm. The critics associated with each of these policies are continually updated in an off-policy manner so that they are closer to the value functions of each policy. In this way, we have access to the past k policy and value functions that represent the values in the value improvement path. Crucially, these critic share a learned representation function which is forced to learn a sufficiently expressive embedding to capture the necessary representation for all past policy value functions.

Figures 4 and 5 illustrate the training procedure for the model, the embedding function ϕ, and the value functions. Referring to Figure 4, the model is used to predict the next state in the embedding space, from which the targets for updating the value functions are calculated. These targets are then used to update the value functions and the embedding function.

The model is updated as shown in Figure 5 (along with ϕ). The indicated ViPer model loss for a transition at timestep t is defined as:

$$\mathcal{L}^t_{\text{ViPer}}(\hat{P}, \phi) = \sum_{i=1}^{k} \left(\mathbb{E}_{\hat{P}(\cdot|\phi(o_t))}[V^{\pi_i}(\hat{z}_{t+1})] - V^{\pi_i}(\phi(o_{t+1})) \right)^2, \quad (1)$$

where the expectation is approximated using Monte Carlo samples from the model. Note that the loss is calculated across a batch of transitions over multiple timesteps.

6. Related work

Several lines of work have focused on the idea of learning abstract planning models or decision-aware models. One major concept is the idea of *bisimulation metrics* (Ferns et al., 2004; 2011; Zhang et al., 2021) which can be used to define representation learning or model learning objectives. Bisimiulation metrics describe a way to measure the distance between two MDPs (for example between a model and the ground truth environment) by the reward outcome of policies on these two MDPs. If any policy leads to the same reward sequence on the model as on the real environment, then these are equivalent. Similar to VAML or PVE models on a large policy class, the Bisimulation distance can be conservative, as it accounts for all policies on the given MDP.

Another concept that combines characteristics of model and representation learning is the concept of *successor features*. Lehnert & Littman (2020) show that successor features, which are empirical averages of future state occupancy under a policy, encode information about the transition function of the environment. Similar concepts such as *proto-value functions* (Mahadevan & Maggioni, 2007) are similarly built on the characteristics of the environment, and encode the underlying structure of the MDP's transition graph in the form of the eigenvectors of the graph Laplacian. These encode temporal information about the transition structure and can be used to define representations based on the behavior of the transition function. Both of these concepts are however difficult to scale up to large continuous environments and do not account for the plicies actually encountered during reinforcement learning.

Model learning has also been used to stabilize the value function representation in empirical investigations (Jaderberg et al., 2016). Lyle et al. (2021) presents a theoretical analysis of the use of model learning as an auxiliary task in representation learning and reaches the conclusion that dynamics learning can greatly improve the stability and expressiveness of learned representations. Since the value function and policy often depend on future state information, encoding some predictive information of future states into the current representation makes can force the representation to be more general. In contrast to the work presented

here, these approaches are not connected more deeply to the goal of model-based RL or model learning, the learned models are merely auxiliary components of the representation learning procedure.

7. Conclusions

In the context of Value-Aware Model Learning, the choice of the value functions to be aware of is crucial. We present a discussion of this choice, bringing in insights from representation learning to propose an improvement over IterVAML. We show that considering the sets of value functions encountered in practice during running an actor-critic algorithm is crucial to ensure that the model can be used for several update steps in the value function space. We also highlight that representation sharing can be used to ensure that a model is both value equivalent and simple to learn, and we provide an algorithm sketch for the resulting ViPer Model. Implementation and experiments are left for future work, as well as deeper discussions about the interplay of model and representation learning updates.

References

Chen, X., Toyer, S., Wild, C., Emmons, S., Fischer, I., Lee, K.-H., Alex, N., Wang, S. H., Luo, P., Russell, S., Abbeel, P., and Shah, R. An empirical investigation of representation learning for imitation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=kBNhgqXatI.

Dabney, W., Barreto, A., Rowland, M., Dadashi, R., Quan, J., Bellemare, M. G., and Silver, D. The value-improvement path: Towards better representations for reinforcement learning. *arXiv preprint arXiv:2006.02243*, 2020.

Dadashi, R., Taiga, A. A., Le Roux, N., Schuurmans, D., and Bellemare, M. G. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pp. 1486–1495. PMLR, 2019.

Ernst, D., Glavic, M., Geurts, P., and Wehenkel, L. Approximate value iteration in the reinforcement learning context. application to electrical power system control. *International Journal of Emerging Electric Power Systems*, 3(1), 2005.

Farahmand, A.-m. Iterative value-aware model learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Farahmand, A.-m., Barreto, A., and Nikovski, D. Value-Aware Loss Function for Model-based Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*, 2017.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *Uncertainty in AI*, 2004.

Ferns, N., Panangaden, P., and Precup, D. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6), 2011.

Fu, X., Yang, G., Agrawal, P., and Jaakkola, T. Learning task informed abstractions. In *International Conference on Machine Learning*, 2021. URL https://proceedings.mlr.press/v139/fu21b.html.

Gordon, G. J. Stable function approximation in dynamic programming. In *International Conference on Machine Learning*, 1995.

Grimm, C., Barreto, A., Singh, S., and Silver, D. The value equivalence principle for model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Grimm, C., Barreto, A., Farquhar, G., Silver, D., and Singh, S. Proper value equivalence. In *Advances in Neural Information Processing Systems*, 2021.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., and Levine, S. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5), 2021.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *International Conference on Learning Representations*, 2016.

Lambert, N., Amos, B., Yadan, O., and Calandra, R. Objective mismatch in model-based reinforcement learning. In *Conference on Learning for Dynamics and Control*, 2020.

Lehnert, L. and Littman, M. L. Successor features combine elements of model-free and model-based reinforcement learning. *J. Mach. Learn. Res.*, 21:196–1, 2020.

Lovatto, A. G., Bueno, T. P., Mauá, D. D., and de Barros, L. N. Decision-aware model learning for actor-critic methods: When theory does not meet practice. In *"I Can't Believe It's Not Better!" at NeurIPS Workshops*, 2020a.

Lovatto, A. G., Bueno, T. P., Mauá, D. D., and de Barros, L. N. Decision-aware model learning for actor-critic methods: When theory does not meet practice. In *"I Can't Believe It's Not Better!" at NeurIPS Workshops*, 2020b.

Lyle, C., Rowland, M., Ostrovski, G., and Dabney, W. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Mahadevan, S. and Maggioni, M. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(10), 2007.

Modhe, N., Kamath, H., Batra, D., and Kalyan, A. Model-advantage optimization for model-based reinforcement learning. *ArXiv*, abs/2106.14080, 2021.

Munos, R. and Szepesvári, C. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9 (5), 2008.

Oh, J., Singh, S., and Lee, H. Value prediction network. *Advances in neural information processing systems*, 30, 2017.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 752–759, 2008.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. The distracting control suite - a challenging benchmark for reinforcement learning from pixels. *ArXiv*, abs/2101.02722, 2021.

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning Proceedings*, pp. 216–224. 1990.

Tomar, M., Mishra, U. A., Zhang, A., and Taylor, M. E. Learning representations for pixel-based control: What matters and why? *arXiv preprint arXiv:2111.07775*, 2021.

Voelcker, C., Liao, V., Garg, A., and Farahmand, A.-m. Value gradient weighted model-based reinforcement learning. *arXiv preprint arXiv:2204.01464*, 2022.

Yarats, D., Kostrikov, I., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

A. ϕ_{Q^*} Abstraction vs. Value-Aware Model Learning

For a finite-state and tabular value functions and models, we can show a relationship between certain representation functions and value-aware model learning. This is formalized in this section.

Suppose we use a Q^*-irrelevant abstraction, which means that for any two states $s_1, s_2 \in \mathcal{S}$, $\phi(s_1) = \phi(s_2) \implies Q^*(s_1, a) = Q^*(s_2, a), \forall a \in \mathcal{A}$. Let $D_{s,a}$ be the collection of transition tuples that start with (s, a) and $D_{x,a} := \sum_{s \in \phi^{-1}(x)} |D_{s,a}|$.

Consider an approximate MLE model using the abstract representation $\bar{M}_\phi = (\mathcal{S}_\phi, \mathcal{A}, \bar{P}_\phi, \bar{R}_\phi, \gamma)$ on $\phi(\mathcal{S})$. Note that $\phi^{-1}(x)$ consists of all the states in $\mathcal{S}$ with the same Q^*. Let $\mathbf{e}_{\phi(s')}$ be the unit vector whose $\phi(s')$-th entry is 1 and all other entries are 0. Then,

$$\bar{P}_\phi(x, a) = \frac{1}{|D_{x,a}|} \sum_{s' \in D_{x,a}} \mathbf{e}_{\phi(s')}, \qquad \forall (x, a) \in \mathcal{S}_\phi \times \mathcal{A}. \tag{2}$$

Theorem A.1. *The model learned by minimizing the IterVAML loss on the abstract state space is equivalent to $\bar{P}_\phi$. The model learned by minimizing the IterVAML loss on the ground state space, when the model can only represent $n = |\mathcal{S}_\phi|$ number of transitions is equivalent to the model learned on the abstract state space and therefore to $\bar{P}_\phi$.*

Proof. Consider minimizing the VAML loss on the abstract space $\mathcal{S}_\phi$, given the same dataset $D_{x,a}$,

$$\hat{P}_{\phi,\text{IterVAML}}(x, a) = \arg\min_{\hat{P}} \left| \frac{1}{|D_{x,a}|} \sum_{s' \in D_{x,a}} \max_{a'} Q^*(\phi(s'), a') - \sum_{s'' \in \mathcal{S}} \hat{P}(\phi(s'')|x, a) \max_{a'} Q^*(\phi(s''), a') \right|^2. \tag{3}$$

This is minimized when $\hat{P}(\phi(s'')|x, a)$ is $\frac{1}{|D_{x,a}|}$ if s'' appears in the dataset $D_{x,a}$, i.e. when $\hat{P}(\phi(s'')|x, a) = \frac{1}{|D_{x,a}|}, \forall s'' \in D_{x,a}$ and 0 otherwise. This is equivalent to

$$\hat{P}_{\phi,\text{IterVAML}}(x, a) = \frac{1}{|D_{x,a}|} \sum_{s' \in D_{x,a}} \mathbf{e}_{\phi(s')}, \qquad \forall (x, a) \in \mathcal{S}_\phi \times \mathcal{A}.$$

Therefore, given a Q^*-irrelevant abstraction, the VAML model and MLE model ($\bar{P}_\phi$) learned on $\mathcal{S}_\phi$ are equivalent.

On the ground state space, $\mathcal{S}$, the VAML loss is:

$$L_{\text{IterVAML}}(s, a; \hat{P}) = \left| \frac{1}{|D_{s,a}|} \sum_{s' \in D_{s,a}} \max_{a'} Q^*(s', a') - \sum_{s'' \in \mathcal{S}} \hat{P}(s''|s, a) \max_{a'} Q^*(s'', a') \right|^2. \tag{4}$$

Since $|\mathcal{S}_\phi| = n$, that means there are only n unique action-value functions, which we enumerate as $Q_i^*(s, a), i = 0, \dots n-1, \forall a \in \mathcal{A}$. This means that for any $s \in \mathcal{S}$, its action-value function is one of these n action-value functions. Thus, we can rewrite the first term of the VAML loss as:

$$\frac{1}{|D_{s,a}|} \sum_{s' \in D_{s,a}} \max_{a'} Q^*(s', a') = \frac{1}{|D_{s,a}|} \sum_{i=0}^{n-1} n_i(\phi(s'); s, a) \max_{a'} Q^*(\phi(s'), a') \tag{5}$$

where $n_i(s'; s, a)$ denotes the number of times the transition from (s, a) to s' led to a "phase" transition into the i-th abstract state, according to the data $D_{s,a}$. Note that $\sum_{i=0}^{n-1} n_i(\phi(s'); s, a) = |D_{s,a}|$. The second term of the loss can be similarly rewritten:

$$\sum_{s'' \in \mathcal{S}} \hat{P}(s''|s, a) \max_{a'} Q^*(s'', a') = \sum_{i=0}^{n-1} \left(\sum_{s'' \in \phi_i} \hat{P}(s''|s, a) \right) \max_{a'} Q^*(x_i, a'), \tag{6}$$

where ϕ_i is the set of states that map to the same abstract state, and x_i is the ϕ-mapped state in abstract set of states ϕ_i (i.e. $x_i = \phi(s'') \forall s'' \in \phi_i$. Thus, the full loss becomes:

$$L_{\text{IterVAML}}(s, a; \hat{P}) = \left| \frac{1}{|D_{s,a}|} \sum_{i=0}^{n-1} n_i(\phi(s'); s, a) \max_{a'} Q^*(\phi(s'), a') - \sum_{i=0}^{n-1} \left(\sum_{s'' \in \phi_i} \hat{P}(s''|s, a) \right) \max_{a'} Q^*(x_i, a') \right|^2. \tag{7}$$

Thus, the optimal model groups together all s'' that map to the same abstract state, inducing the ϕ_{Q^*} abstraction.