Information Maximizing Exploration with a Latent Dynamics Model

Trevor Barron, Heni Ben Amor
Department of Computing, Informatics and Decision Systems Engineering
Arizona State University
United States
tpbarron,hbenamor@asu.edu

Oliver Obst
Centre for Research in Mathematics, School of Computing, Engineering and Mathematics
Western Sydney University
Australia
o.obst@westernsydney.edu.au

Abstract

All reinforcement learning algorithms must handle the trade-off between exploration and exploitation. Many state-of-the-art deep reinforcement learning methods use noise in the action selection, such as Gaussian noise in policy gradient methods or ϵ-greedy in Q-learning. While these methods are appealing due to their simplicity, they do not explore the state space in a methodical manner. We present an approach that uses a model to derive reward bonuses as a means of intrinsic motivation to improve model-free reinforcement learning. A key insight of our approach is that this dynamics model can be learned in the latent feature space of a value function, representing the dynamics of the agent and the environment. This method is both theoretically grounded and computationally advantageous, permitting the efficient use of Bayesian information-theoretic methods in high-dimensional state spaces. We evaluate our method on several continuous control tasks, focusing on improving exploration.

1 Introduction

Model-free reinforcement learning (RL) has enjoyed significant success over the past few years. While model-based RL has not enjoyed the same successes, models have appealing properties that could potentially be leveraged to increase sample-efficiency, guide exploration, or enable high-level reasoning about internal actions.

In this paper, we propose using a model to derive reward bonuses to accelerate a model-free RL technique. The key insight in our approach is that the dynamics model can be learned in the latent feature space of a value function. This builds upon existing work that formalizes an equivalence between a linear model and linear value estimate. We learn an actor-critic style policy and value function using standard model-free RL techniques, but simultaneously learn a Bayesian dynamics model (BNN) based on the *latent feature* representation found by the final hidden layer of the value function approximator. The BNN takes as input the latent feature representation for a state and an action and predicts the latent feature representation for the state at the subsequent time step. Both the Bayesian model and the value function are linear in these features. The result is a compact representation of the task dynamics. In turn, the latent dynamics model can be used for intelligent, information theoretic exploration of the state space. The main contributions of this work include:

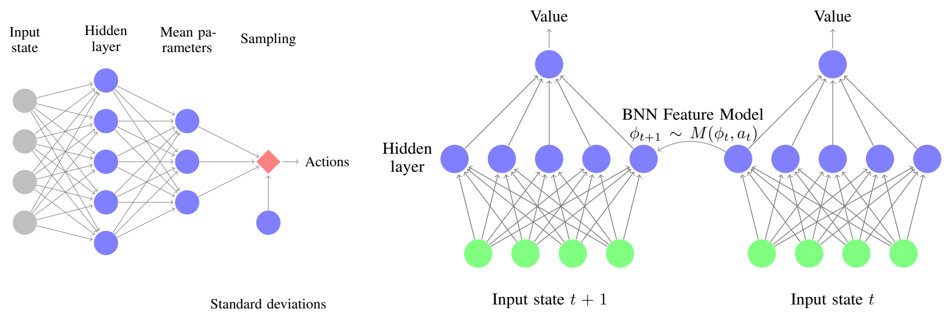


Figure 1: The network structures used to approximate the policies in the vector and high-dimensional cases (left). A Bayesian network is trained in the latent feature space of a value function approximator (right).

1. Empirical results indicating that information maximizing exploration on latent features performs as good or better than an equivalent method on the true state space.
2. A Bayesian transition model learned in the latent feature space of a neural network value function capitalizing on already learned features that can be used to compute quantities such as information gain for exploration bonuses.
3. An analysis of the relationship between the model distribution over latent features and the value distribution.

The introduced method, which we call Information Maximizing Latent Exploration (IMLE), is both theoretically grounded and computationally advantageous. We evaluate our method on several continuous control, focusing on improving exploration.

2 Methodology

In this work we assume a finite-horizon discounted Markov decision process (MDP) defined by a tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is the state set, $\mathcal{A} \subseteq \mathbb{R}^m$ is the action set, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the state transition distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward, ρ_0 is the start state distribution and $\gamma \in (0, 1]$ is the discount factor. We are interested in finding a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ that maximizes the expected discounted reward, $\mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right]$.

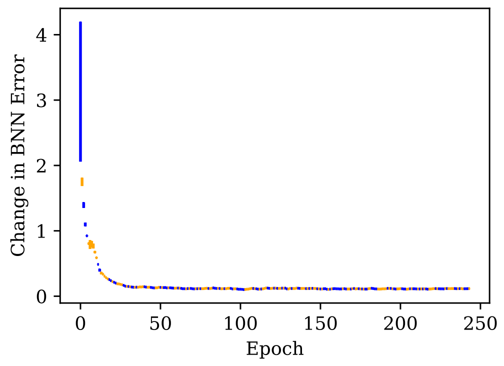
2.1 Relationship between model and value function

Several researchers have noted a relationship between transition models and value estimates [Parr et al., 2008, Sutton et al., 2012]. Our approach extends previous work that studied the relationship between the solutions reached by a linear value function approximation and a one-step linear model. Parr et al. [2008] proved that the solution found by approximating a value function given an exact model and by solving for an exact value function given an approximate model are equivalent. This relationship motivates the decision to learn a dynamics model, not in the state space, but rather in the latent feature space of the value function. That is, we assume that features found to be useful for value estimation will also be useful for one-step model prediction. Furthermore, we argue that because the value estimate is linear in the latent features the model should be as well. Hence, we formulate the model as a linear Bayesian network.

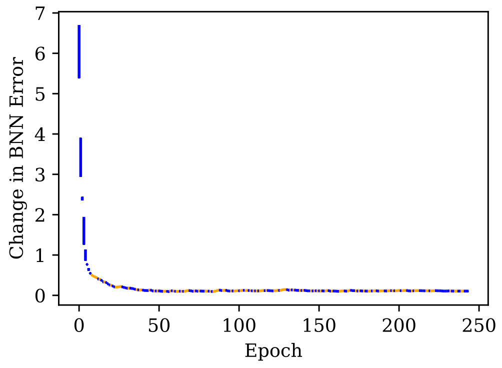
2.2 A latent dynamics model

By logically decomposing the structure of a neural network into two components, first a feature learning component and second a linear approximation, we convert our problem into a form that closely mirrors the linear-case analysis equivalence by Parr et al. [2008]. A graphical representation of our network structure is shown in Figure 1.

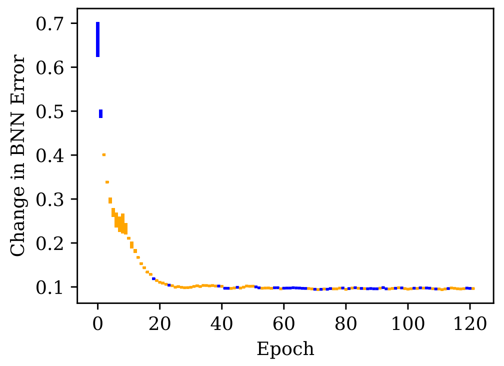
We alternate between approximating the value function and a linear Bayesian dynamics model given the current value function's latent features. Using value function features as basis for model learning results in non-stationary features. In practice we do not find this to be a problem and observe that the latent dynamics model converges rapidly. In fact, we empirically find that updating the value function often decreases the error of the dynamics model, especially early in training (see Figure 2). This finding highlights the relationship between features for value estimation and features for prediction and validates our approach.



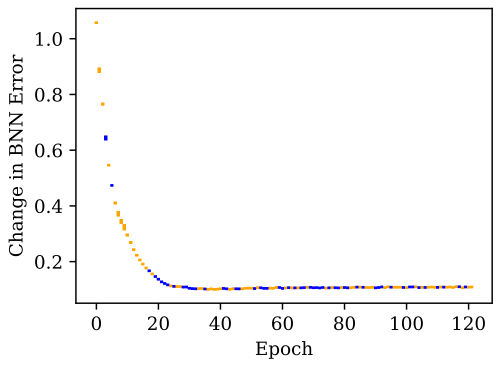
(a) Dense Walker2D



(b) Dense Ant



(c) Sparse Mountain Car



(d) Sparse Acrobot

Figure 2: Blue bars represent BNN improvement during a value function update, orange bars represent BNN accuracy decline. Early in training updating the value function reduces BNN model error. This empirical finding supports the theoretical model-value relationship and motivates learning the model in the feature space of the value function.

Employing a Bayesian dynamics model provides additional useful information including an uncertainty measure regarding a prediction and a means of computing information gained by visiting a state (through KL divergence). A single forward step can be expressed as $\Pr(\phi_{V_\theta}(\hat{s}_{t+1}) \mid \phi_{V_\theta}(s_t), a_t; \omega)$ where $\phi_{V_\theta}(s_t)$ is the output from the final hidden layer of the value function (pre-activations), V_θ, and ω are the parameters of the model sampled from a random variable $\boldsymbol{\Omega}$. A distribution over possible models is maintained through a prior $\Pr(\omega)$, which we choose to be Gaussian.

Operating in the latent space also provides the means to obtain a measure of uncertainty not only over our state transition but also over the future value approximation. Note that since the value estimate is simply a linear transformation of the latent features, the output distribution of the model can be transformed to represent a distribution over the value estimate. The value function approximation remains a deterministic linear transform but a distribution over the *inputs* is given by the dynamics model. If the probability of a latent encoding of the value function given a state, s, is drawn from a normal distribution $\Pr(\boldsymbol{L}) = \Pr(\phi_{V_\theta}(s_t)) \sim \mathcal{N}(\mu_s, \sigma_s^2)$, as is the case with our linear Bayesian network, then a distribution over the value function itself is given by:

$$\mathrm{E}(\boldsymbol{V}) = W\mathrm{E}(\boldsymbol{L}) + \boldsymbol{b} \tag{1}$$

$$\mathrm{Var}(\boldsymbol{V}) = W\mathrm{Cov}(\boldsymbol{L})W^T, \tag{2}$$

where W and $\boldsymbol{b}$ are the weights and biases of the final linear layer. Accordingly, the distribution of the value estimate is $\Pr(V_\theta(s_t)) \sim \mathcal{N}(\mathrm{E}(\boldsymbol{V}), \mathrm{Var}(\boldsymbol{V}))$. Hence by learning a Bayesian dynamics model in latent feature space we get an uncertainty measure of the value function for free without incorporating any weight uncertainty into the value function itself.

2.3 Incentivizing exploration with reward bonuses and intrinsic motivation

In this work we focus on exploration and evaluate a method akin to Variational Information Maximizing Exploration (VIME) [Houthooft et al., 2016] in the latent space. Reinforcement learning requires a careful balance between exploration and exploitation. One approach is to provide reward bonuses in order to incentivize an agent to take actions even in the absence of well-shaped rewards from the environment.

This exploration strategy is based on the idea that an agent should behave in a manner to reduce its uncertainty regarding the environment dynamics. Bayesian models handle the trade off between exploration and exploitation naturally. In VIME, optimizing a Bayesian model is made tractable with variational inference and reward bonuses are derived from the amount of information gained by visiting a state. In particular, an intrinsic reward is derived for reductions in model entropy. This encourages the agent to take actions that provide maximal information about the environment. Following the notation in Houthooft et al. [2016], this quantity can be expressed formally as the information gain of observing a future state s_{t+1}, taking action a_t, with history ζ_t,

$$I(S_{t+1}; \Omega \mid \zeta_t, a_t) = \mathbb{E}_{s_{t+1} \sim \mathcal{P}(\cdot|\zeta_t, a_t)}[D_{\mathrm{KL}}(p(\omega|\zeta_t, a_t, s_{t+1}) \parallel p(\omega|\zeta_t))]. \tag{3}$$

The reward bonus is then formulated as,

$$r'(s_t, a_t, s_{t+1}) = r(s_t, a_t) + \eta D_{\mathrm{KL}}(p(\omega|\zeta_t, a_t, s_{t+1}) \parallel p(\omega|\zeta_t)), \tag{4}$$

where η is a parameter that controls the trade-off between exploration and exploitation.

In practice, computing the distributions within the KL divergence are usually intractable. Accordingly, we approximate this distribution over the parameters given the agent's experience using variational inference [Hinton and Van Camp, 1993]. This approach minimizes $D_{\mathrm{KL}}(q(\omega;\theta) \parallel p(\omega))$ where $q(\omega;\theta)$ is a probabilistic model parameterized by θ. The information gain for visiting a state is then approximated using the KL divergence between the parameters of the model before and after observing s_{t+1}, $D_{\mathrm{KL}}(q(\omega;\theta_{t+1}) \parallel q(\omega;\theta_t)$ and the reward bonus becomes,

$$r'(s_t, a_t, s_{t+1}) = \underbrace{r(s_t, a_t)}_{\text{Environ. reward}} + \eta \underbrace{D_{\mathrm{KL}}(q(\omega;\theta_{t+1}) \parallel q(\omega;\theta_t))}_{\text{Intrinsic reward}}. \tag{5}$$

As in VIME we model the Bayesian network as a fully-factored Gaussian distribution, which makes it possible to approximate the KL divergence with a single second order gradient step.

The algorithm pseudo-code for IMLE is specified in Algorithm 1. For more details on VIME specifics we refer readers to the original paper [Houthooft et al., 2016].

2.4 A relationship between model-based information gain-based intrinsic motivation and Bayesian Q-Learning in the Linear Case

We now discuss a relationship between intrinsic rewards based on information gain of a dynamics model and information gain of a value estimate when both the model and value estimate are linear in the feature representation. Specifically, we find that if the Bayesian dynamics model represents the true distribution over the future latent features, then the intrinsic reward bonus derived from model-based information gain is greater than or equal to a bonus derived from information gain regarding the Q-value estimate. We assume $M(s_t, a_t) \overset{d}{=} \phi^*_{t+1}$, where $M(s_t, a_t)$ is the distribution over latent features predicted by the model at time $t+1$ and ϕ^*_{t+1} represents the true distribution over the latent features at time $t+1$.

Algorithm 1 Information maximizing exploration with a latent Bayesian dynamics model

```
1: Initialize policy $\pi_0$, Bayesian model $M$, replay memory $R$
2: for epoch $= 0, 1, 2, \ldots$ until convergence do
3:     for time step in epoch do
4:         Behave according to policy $a_t = \pi_{\theta_p}(s_t)$
5:         Save transition $(s_t, a_t, s_{t+1})$ to replay memory, $R$
6:         Project $l_t = \phi(s_t)$ and $l_{t+1} = \phi(s_{t+1})$
7:         Compute $D_{\text{KL}}(q(\omega; \theta'_{n+1}) \parallel q(\omega; \theta_{n+1}))$ by approximation $\nabla_\theta^T H^{-1} \nabla_\theta$
8:         Compute rewards $r'(s_t, a_t, s_{t+1}) = r(s_t, a_t) + D_{\text{KL}}(q(\omega; \theta'_{n+1}) \parallel q(\omega; \theta_{n+1}))$
9:     Draw samples, $S$, from replay, $R$
10:    Project samples to latent representation $l_s = \phi(s)$
11:    Minimize variational loss of Bayesian dynamics model, $M$, on one-step latent feature
12:        dynamics, $l_{s_{t+1}} \sim M(l_{s_t}, a_t)$
13:    Update policy, $\pi$ , given augmented rewards with standard RL method
```

The KL divergence of the dynamics model is computed as a sum of the KL divergence over each parameters' old and new distribution after observing a new sample. Specifically, let $w_1 \sim \mathcal{N}(\mu_{w_1}, \sigma^2_{w_1})$ and $b_1 \sim \mathcal{N}(\mu_{b_1}, \sigma^2_{b_1})$. We then let the agent take an action a and transition to state s. Updating the dynamics model with this new information gives $w_2 \sim \mathcal{N}(\mu_{w_2}, \sigma^2_{w_2})$ and $b_2 \sim \mathcal{N}(\mu_{b_2}, \sigma^2_{b_2})$. Then, the information gain according to the model is $IG_{model} = D_{\text{KL}}(w_1 \parallel w_2) + D_{\text{KL}}(b_1 \parallel b_2)$. Given the corresponding parameter distributions, we find the output distributions for sample, s, before and after observing s to be $o_1 \sim \mathcal{N}(x\mu_{w_1} + \mu_{b_1}, x^2\sigma^2_{w_1} + \sigma^2_{b_1})$ and $o_2 \sim \mathcal{N}(x\mu_{w_2} + \mu_{b_2}, x^2\sigma^2_{w_2} + \sigma^2_{b_2})$. This follows because the prior is Gaussian and the model is linear.

Moreover, since the KL divergence is preserved under linear transformation (proof in Appendix), the KL divergence of the output distribution is identical to the KL divergence of the Q-value distribution before and after observing s. Specifically, assume a mean, μ_{m_1}, and standard deviation, σ_{m_1}, defined by the output of the dynamics model before observing s and a new distribution defined by μ_{m_2} and σ_{m_2} after observing s. We then examine the KL divergence between $m_1(x) = \mathcal{N}(\mu_{m_1}, \sigma^2_{m_1})$ and $m_2(x) = \mathcal{N}(\mu_{m_2}, \sigma^2_{m_2})$. Since we assume a network structure such that the value estimate is only a linear transformation of the output of the dynamics model, it too defines a Gaussian distribution. Given parameters w and b of the linear transformation, these transformed distributions, $q_1(x) = \mathcal{N}(w\mu_{m_1} + b, w^2\sigma^2_{m_1})$ and $q_2(x) = \mathcal{N}(w\mu_{m_2} + b, w^2\sigma^2_{m_2})$ represent the distribution of $Q(s, a)$ before and after observing s. The KL divergence of the output of dynamics model is equivalent to the KL divergence of the Q-value estimate.

Empirically, by sampling, we find $D_{\text{KL}}(q_1 \parallel q_2) = D_{\text{KL}}(m_1 \parallel m_2) \leq IG_{model}$. (It may be possible to prove generally that $D_{\text{KL}}(m_1 \parallel m_2) \leq IG_{model}$ when both are linear in the feature representation though thus far the authors have been unable to show this in a general case.) Intuitively, this means that incentivizing exploration to areas of high value uncertainty, possibly derived from a Bayesian Q-network, provides a weaker intrinsic reward than incentivizing exploration to areas of high model uncertainty. Further analysis of when the divergence of the model and the divergence of the Q-value estimate differ is left to future work.

3 Experimental Results

We focus our study on actor-critic policy gradient methods where both policy and value function approximators are learned. While some methods have merged the policy and value function into a single function approximator with two heads we instead learn separate models for each in order to align with the theory regarding model-value equivalence as closely as possible [Mnih et al., 2016, Schulman et al., 2017]. In this work we use a Proximal Policy Optimization (PPO) method as a baseline [Schulman et al., 2017]. We evaluate our method on continuous control benchmark tasks from the OpenAI Gym based on Box2D and the Pybullet simulator [Brockman et al., 2016, Coumans and Bai, 2016–2017, Catto, 2017]. We compare our method to PPO with and without VIME. All experiments are averaged over three random seeds. In general we find that IMLE performs slightly better and VIME on continuous control tasks.

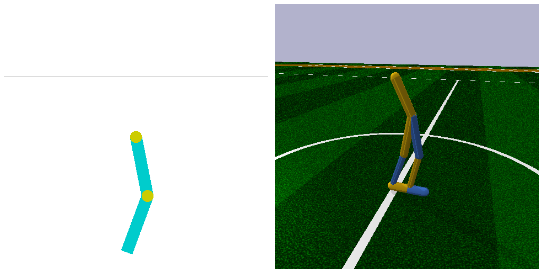


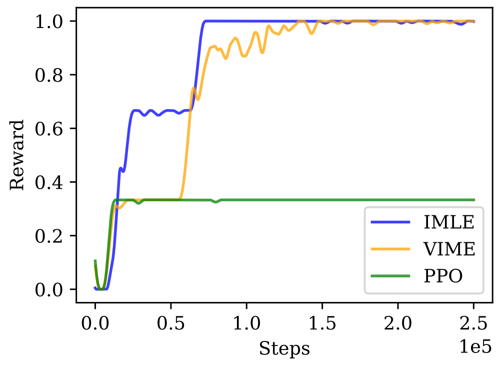
Figure 3: Example renders of the Acrobot and Walker2D tasks.

3.1 Implementation Details

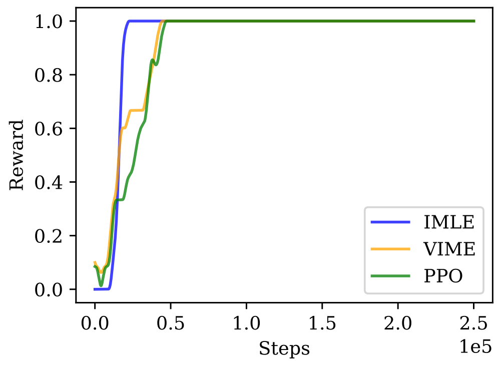
The IMLE and VIME, BNN updates and intrinsic reward generation is integrated into the PPO algorithm. The agent interacts with the environment for a defined number of steps, K. Each transition is added to a FIFO replay memory. After K steps have passed the RL policy and value function are updated according to the PPO update algorithm. If the replay memory is of sufficient size, the BNN is then updated as well. Once the BNN has been updated once, it is used to derive an intrinsic reward on for each transition experienced by the agent. The intrinsic reward given to the agent is normalized by the median of the 10 previous KL divergence epoch means and then scaled by η. In IMLE, the encoded representation is taken before applying any non-linearity but is then normalized using a running mean and standard deviation. A full table of hyperparameters is in the Appendix. Both the policy and value function are represented as neural networks. In all tasks the policy had two hidden layers of 64 nodes and the value function had a two hidden layers of size 32. The latent feature size is therefore 32 as well.

3.2 Tasks with Sparse Rewards

To examine how effectively our method incentivizes exploration we modify the Acrobot ($\mathcal{S} \in \mathbb{R}^6, \mathcal{A} \in \mathbb{R}$) and MountainCar ($\mathcal{S} \in \mathbb{R}^2, \mathcal{A} \in \mathbb{R}$) environments to have very sparse rewards, therefore requiring an efficient exploration strategy [Sutton, 1996]. A reward of positive one is given when the goal state is reached and the reward is zero everywhere else. The Acrobot environment is also modified to have continuous torque input with a maximum magnitude of one. Figure 4a and 4b show the performance of IMLE versus VIME and PPO. Without intrinsic rewards, PPO is unable to solve the Acrobot task consistently. While both IMLE and VIME are able to solve the task, IMLE reaches the optimal reward of one more rapidly. Note that the unusual step-like nature of the IMLE learning curve is not reflective of the algorithm, instead it is a remnant of averaging multiple training runs. In the MountainCar environment all three algorithms solve the task but IMLE solves the task most rapidly.



(a) Sparse Acrobot



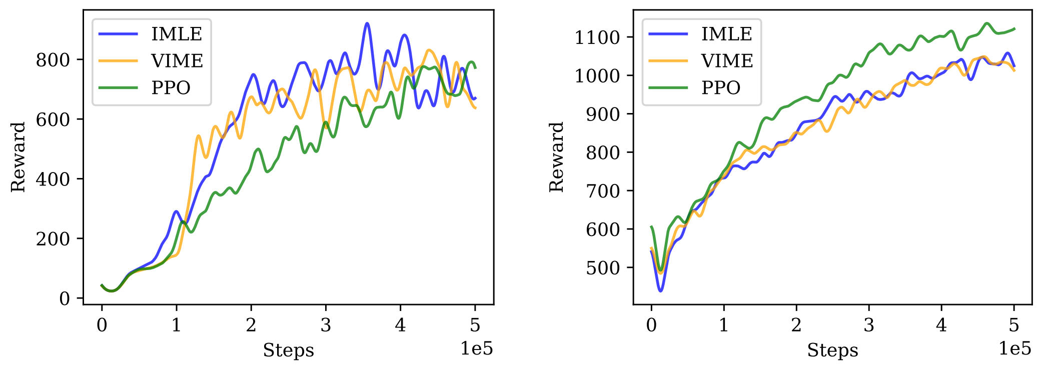
(b) Sparse Mountain Car

Figure 4: We evaluate IMLE vs VIME and PPO on the Acrobot and MountainCar tasks with sparse rewards.

4 Tasks with Well-Shaped Rewards

In addition to the Acrobot and MountainCar tasks, we also test our method on continuous control tasks in the Pybullet simulator. We run experiments on Walker2D ($\mathcal{S} \in \mathbb{R}^{22}, \mathcal{A} \in \mathbb{R}^{6}$) and Ant ($\mathcal{S} \in \mathbb{R}^{28}, \mathcal{A} \in \mathbb{R}^{8}$).

In these tasks we do not modify the reward. Instead we wish to observe how IMLE fares in tasks with carefully shaped rewards. Interestingly, we find that even with well-shaped rewards IMLE can still provide improvement and, moreover, that it can outperform VIME (Figure 5a). In the Walker2D task both IMLE and VIME out-pace vanilla PPO with IMLE slightly above VIME. This is not always the case, though, as additional exploration can also reduce performance. In the Ant environment, both IMLE and VIME marginally under perform relative to vanilla PPO (Figure 5b).



(a) Dense Walker2D (b) Dense Ant

Figure 5: We evaluate IMLE vs VIME and PPO on the Walker2D and Ant tasks with dense rewards.

4.1 Computational Considerations

Bayesian models are known to have high computational requirements as a result of the need to make multiple predictions in order to estimate evidence lower bound. Operating in the latent feature space also provides appealing computational improvements over VIME as the observation space grows. When the dynamics model is trained directly on state inputs, more parameters are needed in the BNN for good accuracy.

For example, consider the Walker2D task, which has $\mathcal{S} \in \mathbb{R}^{22}$ and $\mathcal{A} \in \mathbb{R}^{6}$. Then the BNN used in VIME has an input size of 28 (concatenated state and action vectors), two layers of size 32, and an output size of 22. If we draw 10 samples to estimate the variational lower bound, then the number of multiplications required for this operation is $(n_{\text{input}} \cdot n_{\text{out1}} + n_{\text{out1}} \cdot n_{\text{out2}} \cdot n_{\text{out2}} \cdot n_{\text{pred}}) \cdot n_{samples} = (28 \cdot 32 + 32 \cdot 32 + 32 \cdot 22) \cdot 10 = 26240$.

A latent model fares better as the dimensionality of the input increases as the encoding occurs only once and the sampling is then done on a linear model. We assuming a value function with two hidden layers each of size 32 as is used in our work. Then the number of multiplications is $n_{\text{in}} \cdot n_{\text{out1}} + n_{\text{out1}} \cdot n_{\text{out2}} \cdot (n_{\text{modelin}} \cdot n_{\text{modelout}}) \cdot n_{\text{samples}} = 28 \cdot 32 + 32 \cdot 32 + (38 \cdot 32) \cdot 10 = 14080$.

That is, compared to VIME, the IMLE update requires approximately half the number of multiplications to estimate the output distribution and BNN loss. This can be interpreted as a hybrid approach that combines the benefit of neural network feature extraction and Bayesian methods. In general, as the complexity of the state representation grows the deterministic encoding results in an increasingly more efficient BNN update.

5 Related Work

This work received inspiration most directly from VIME [Houthooft et al., 2016] and the analysis of linear models and linear value function approximation by Parr et al. [2008]. VIME provides intrinsic rewards based on information gain computed from a Bayesian dynamics model. Parr et al. [2008] developed the theory equating the solutions of linear fixed point value function methods and linear models.

A large corpus of work has been devoted to developing exploration strategies for reinforcement learning agents. The idea of working in a lower dimensional space is not new. It is common in robotics applications to perform dimensionality reduction to increase convergence rate. Lange et al. [2012] use an autoencoder to extract features on which a policy is trained. Luck et al. [2016] employ a linear probabilistic dimensionality reduction that is learned during training. Both of these methods require learning additional features for the dimensionality reduction, as opposed to our method which operates on existing features being learned by the value function.

Several published methods use prediction for exploration. Stadie et al. [2015] use an autoencoder to derive novelty bonuses based on prediction error of a latent state encoding. While this work does prediction in the latent encoding, it does not take advantage of features learned by the Q-network and learns policies only in discrete action spaces. Pathak et al. [2017] recently published a method that uses intrinsic rewards based on a forward model to provide reward bonuses based on high model error and successfully learn policies in environments with sparse or no rewards.

None of the aforementioned works have considered sharing features between a value function and dynamics model or use a latent feature-level dynamics model for exploration.

6 Conclusion

This work presented a method for learning a probabilistic model, which can be used to derive intrinsic reward bonuses, in the learned feature space of a value function. We show results indicating that our method, IMLE, achieves performance comparable or better than VIME.

Since operating in the latent space unties the dimensionality of the input observation from the dimensionality of the Bayesian model, we believe that IMLE has the potential to scale to much higher dimensional problems including pixel observations.

Additionally, the improvement in model error on value function updates indicate a very close practical model-value relationship. Accordingly, we also believe it is worthwhile to investigate updating a value function using the model *gradient* (as opposed to model-based RL updates).

Finally, while the proposed network structure has theoretical foundations, it is logical to wonder whether a similar approach is possible in the feature space of the policy or a joint policy-value network as in A3C and some PPO implementations [Mnih et al., 2016, Schulman et al., 2017]. A joint structure is appealing as it would provide a means to perform model-based RL updates as well. Our first attempts with such a structure did not perform as well as when learning the model only in the feature space of the value function though we do believe there is future work to be done in this area.

References

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

Erin Catto. A 2d physics engine for games, 2017. URL `http://box2d.org`.

Erwin Coumans and Yunfei Bai. pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org/`, 2016–2017.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

Sascha Lange, Martin Riedmiller, and Arne Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–8. IEEE, 2012.

Kevin Sebastian Luck, Joni Pajarinen, Erik Berger, Ville Kyrki, and Heni Ben Amor. Sparse latent space policy search. In *Association for the Advancement of Artificial Intelligence*, 2016.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. 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*, pages 752–759. ACM, 2008.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363*, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015. URL `http://arxiv.org/abs/1507.00814`.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044, 1996.

Richard S. Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *CoRR*, abs/1206.3285, 2012. URL `http://arxiv.org/abs/1206.3285`.

Appendix

Hyperparameters for PPO, IMLE, and VIME

We use a standard PPO algorithm adapted from the OpenAI baselines implementation (Table 1).

Table 1: PPO Hyperparameters

Parameter	Value
# processes	1
Epoch steps	2048
Entropy coef.	0
PPO epochs	10
PPO clip	0.2
PPO batch size	64
Learning rate	3e-4
Discount (γ)	0.99
GAE (τ)	0.95
Horizon	1000 (500 in Acrobot)
Steps	500000 (250000 in Acrobot and MountainCar)

For IMLE and VIME we use hyperparameter settings as close to the original paper as possible (Table 2).

Table 2: IMLE and VIME Hyperparameters

Parameter	Value
BNN updates per step	500
BNN num samples	10
BNN batch size	32
BNN update interval	1
BNN η	0.0001
Min replay size	500
KL queue length	10

Proof of that KL divergence is unchanged under linear transformation

Given an estimate mean, μ_1, and standard deviation, σ_1, defined by the output of the dynamics model for the agent at a given time step, t. We let the agent take an action a and transition to state s_{t+1}. Updating the dynamics model given this new information we have a new output distribution defined by μ_2 and σ_2. We examine the KL divergence between $p(x) = \mathcal{N}(\mu_1, \sigma_1)$ and $q(x) = \mathcal{N}(\mu_2, \sigma_2)$.

Since we assume a network structure such that the value estimate is only a linear transformation of the output of the dynamics model, it too defines a Gaussian distribution. Given parameters w and b of the linear transformation, these transformed distributions, $r(x) = \mathcal{N}(w\mu_1 + b, w^2\sigma_1)$ and $s(x) = \mathcal{N}(w\mu_2 + b, w^2\sigma_2)$ represent the distribution of $Q(s_t, a_t)$ before and after observing s_{t+1}. Moreover, the KL divergence of the output of dynamics model is equivalent to the transformed distribution.

Using that the KL divergence of two multivariate Gaussian distributions is,

$$D_{KL}(p,q) = \int p(x) \frac{\log p(x)}{\log q(x)} dx \tag{6}$$

$$= \frac{1}{2}\left(\log\left(\frac{\det \Sigma_2}{\det \Sigma_1}\right) + \mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)'\Sigma_2^{-1}(\mu_2 - \mu_1) - N\right) \tag{7}$$

then,

$$
\begin{aligned}
D_{KL}(r,s) &= \frac{1}{2}\Bigg(\log\Big(\frac{\det W\Sigma_2 W^T}{\det W\Sigma_1 W^T}\Big) + \mathrm{Tr}((W\Sigma_2 W^T)^{-1}(W\Sigma_1 W^T))+ \qquad (8)\\
&\quad ((W\mu_2+\boldsymbol{b})-(W\mu_1+\boldsymbol{b}))'(W\Sigma_2 W^T)^{-1}((W\mu_2+\boldsymbol{b})-(W\mu_1+\boldsymbol{b}))-N\Bigg)\\
&= \frac{1}{2}\Bigg(\log\Big(\frac{\det \Sigma_2}{\det \Sigma_1}\Big) + \mathrm{Tr}((W\Sigma_2 W^T)^{-1}(W\Sigma_1 W^T))+ \qquad (9)\\
&\quad ((W\mu_2+\boldsymbol{b})-(W\mu_1+\boldsymbol{b}))'(W\Sigma_2 W^T)^{-1}((W\mu_2+\boldsymbol{b})-(W\mu_1+\boldsymbol{b}))-N\Bigg)\\
&= \frac{1}{2}\Bigg(\log\Big(\frac{\det \Sigma_2}{\det \Sigma_1}\Big) + \mathrm{Tr}((W^T)^{-1}\Sigma_2^{-1}W^{-1}W\Sigma_1 W^T)+ \qquad (10)\\
&\quad (W\mu_2-W\mu_1)'(W\Sigma_2 W^T)^{-1}(W\mu_2-W\mu_1)-N\Bigg)\\
&= \frac{1}{2}\Bigg(\log\Big(\frac{\det \Sigma_2}{\det \Sigma_1}\Big) + \mathrm{Tr}(\Sigma_2^{-1}\Sigma_1)+ \qquad (11)\\
&\quad (\mu_2-\mu_1)'W^T(W^T)^{-1}\Sigma_2^{-1}W^{-1}W(\mu_2-\mu_1)-N\Bigg)\\
&= \frac{1}{2}\Bigg(\log\Big(\frac{\det \Sigma_2}{\det \Sigma_1}\Big) + \mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)'\Sigma_2^{-1}(\mu_2-\mu_1)-N\Bigg) \qquad (12)\\
&= D_{KL}(p,q) \qquad (13)
\end{aligned}
$$