Interpretable Dynamics Models for Data-Efficient Reinforcement Learning

Markus Kaiser[1,2*] and Clemens Otte[1] and Thomas Runkler[1,2] and Carl Henrik Ek[3]

1- Siemens AG, Germany. 2- Technical University of Munich, Germany.
3- University of Bristol, United Kingdom.

Abstract. In this paper, we present a Bayesian view on model-based reinforcement learning. We use expert knowledge to impose structure on the transition model and present an efficient learning scheme based on variational inference. This scheme is applied to a heteroskedastic and bimodal benchmark problem on which we compare our results to NFQ and show how our approach yields human-interpretable insight about the underlying dynamics while also increasing data-efficiency.

1 Introduction

In reinforcement learning (RL) [8], an agent's task is to learn a policy π which, given the current state $\boldsymbol{s}$ of an environment, chooses an action $\boldsymbol{a}$ to achieve the goal specified by a reward function r mapping states to numerical rewards. The next state $\boldsymbol{s}' = f(\boldsymbol{s}, \boldsymbol{a})$ is determined by the latent and possibly stochastic transition function f. We consider batch RL problems [6], where we are presented with a set of state transitions $\mathcal{D} = \{(\boldsymbol{s}_n, \boldsymbol{a}_n, \boldsymbol{s}'_n)\}_{n=1}^N$ and are unable to interact with the original system to find a policy. This setup is common in industrial applications of RL, where deploying an untrusted policy can lead to safety issues. Similarly, gathering exploration data can be costly, calling for data-efficient methods.

Deisenroth et al. [2] showed how data-efficiency in model-based RL can be increased with probabilistic models for the transition dynamics f. They provide a principled way of taking model uncertainty into account when evaluating the performance of a policy, thereby reducing the impact of model-bias. A shortcoming of this approach is the limitations put on modelling choices by the inference scheme. Transition dynamics are modelled as standard Gaussian processes (GPs) and policies and rewards must be of specific forms. In this work, we extend this approach by allowing and imposing additional structure. In many environments, experts can describe abstract properties of the system even if no closed form models are available. Incorporating this knowledge facilitates learning and allows us to precisely state what we want to learn from data.

The paper is outlined as follows. After introducing the heteroscedastic and bimodal Wet-Chicken benchmark, we show how high-level knowledge about this system can be used to impose Bayesian structure. We derive an efficient inference scheme for both the dynamics model and for probabilistic policy search based on variational inference. We show that this approach yields interpretable models and policies and is significantly more data-efficient than less interpretable alternatives.

*The project this report is based on was supported with funds from the German Federal Ministry of Education and Research under project number 01 IS 18049 A.

2 The Wet-Chicken Benchmark

In the Wet-Chicken problem [3], a canoeist is paddling in a two-dimensional river. The canoeist's position at time t is given by $\boldsymbol{s}_t = (x_t, y_t)$, where x_t denotes the position along the river and y_t the position across it. The river is bounded by its length $l = 5$ and width $w = 5$. There is a waterfall at the end of the river at $x = l$. The canoeist wants to get close to the waterfall to maximize the reward $r(\boldsymbol{s}_t) = x_t$. However, if the canoeist falls down the waterfall he has to start over at the initial position $(0, 0)$.

The river's flow consists of a deterministic velocity $v_t = y_t \cdot {}^3\!/_w$ and stochastic turbulence $b_t = 3.5 - v_t$, both of which depend on the position on the y-axis. The higher y_t the faster the river flows but also the less turbulent it becomes. The canoeist chooses his paddle direction and intensity via an action $\boldsymbol{a}_t = (a_{t,x}, a_{t,y}) \in [-1, 1]^2$. The transition function $f : (\boldsymbol{s}_t, \boldsymbol{a}_t) \mapsto \boldsymbol{s}_{t+1} = (x_{t+1}, y_{t+1})$ is given by

$$x_{t+1} = \begin{cases} 0 & \text{if } \hat{x}_{t+1} > l \\ 0 & \text{if } \hat{x}_{t+1} < 0 \\ \hat{x}_{t+1} & \text{otherwise} \end{cases} \qquad y_{t+1} = \begin{cases} 0 & \text{if } \hat{x}_{t+1} > l \text{ or } \hat{y}_{t+1} < 0 \\ w & \text{if } \hat{y}_{t+1} > w \\ \hat{y}_{t+1} & \text{otherwise} \end{cases} \tag{1}$$

where $\hat{x}_{t+1} = x_t + (1.5 \cdot a_{t,x} - 0.5) + v_t + b_t \cdot \tau_t$ and $\hat{y}_{t+1} = y_t + a_{t,y}$ and $\tau_t \sim \mathcal{U}(-1, 1)$ is a uniform random variable that represents the turbulence.

There is almost no turbulence at $y = w$, but the velocity is too high to paddle back. Similarly, the velocity is zero at $y = 0$, but the canoeist can fall down the waterfall unpredictably due to the high turbulence. A successful canoeist must find a trade-off between the stochasticity and uncontrollable velocities in the river to get as close to the waterfall as possible.

3 Probabilistic Policy Search

We are interested in finding a policy specified by the parameters $\boldsymbol{\theta}_\pi$ which maximizes the discounted return $J^\pi(\boldsymbol{\theta}_\pi) = \sum_{t=0}^{T} \gamma^t r(\boldsymbol{s}_t) = \sum_{t=0}^{T} \gamma^t r_t$. Starting from an initial state $\boldsymbol{s}_0$ we generate a trajectory of states $\boldsymbol{s}_0, \dots, \boldsymbol{s}_T$ obtained by applying the action $\boldsymbol{a}_t = \pi(\boldsymbol{s}_t)$ at every time step t. The next state is generated using the (latent) transition function f, yielding $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t)$.

Many environments have stochastic elements, such as the random drift in the Wet-Chicken benchmark from Section 2. We take this stochasticity into account by interpreting the problem from a Bayesian perspective where the discounted return specifies a generative model whose graphical model is shown in Fig. 1. Because of the Markov property assumed in RL, conditional independences between the states yield a recursive definition of the state probabilities given by

$$\begin{aligned} \mathrm{p}(\boldsymbol{s}_{t+1} \,|\, f, \boldsymbol{\theta}_\pi) &= \int \mathrm{p}(f(\boldsymbol{s}_t, \boldsymbol{a}_t) \,|\, \boldsymbol{s}_t, \boldsymbol{a}_t)\, \mathrm{p}(\boldsymbol{a}_t \,|\, \boldsymbol{s}_t, \boldsymbol{\theta}_\pi)\, \mathrm{p}(\boldsymbol{s}_t)\, \mathrm{d}\boldsymbol{a}_t\, \mathrm{d}\boldsymbol{s}_t, \\ \mathrm{p}(r_t \,|\, \boldsymbol{\theta}_\pi) &= \int \mathrm{p}(r(\boldsymbol{s}_t) \,|\, \boldsymbol{s}_t)\, \mathrm{p}(\boldsymbol{s}_t \,|\, \boldsymbol{\theta}_\pi)\, \mathrm{d}\boldsymbol{s}_t. \end{aligned} \tag{2}$$

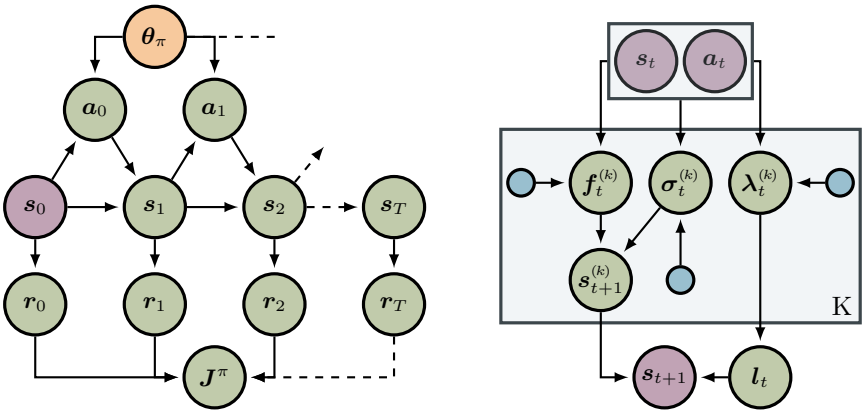


Fig. 1: The graphical models considered in this work, where violet nodes are observed, parameters are shown in yellow and variational parameters are blue. The generative process for the return J^π (left) shows how starting from $\boldsymbol{s}_0$, a trajectory of length T is generated with the policy parameterized by $\boldsymbol{\theta}_\pi$. The return is generated by the rewards which depend on their respective states only. The transition model (right) separates the flow-behaviour of the river $\boldsymbol{f}_t$, the heteroscedastic noise process $\boldsymbol{\sigma}_t$ and the possibility of falling down $\boldsymbol{\lambda}_t$. Latent variables $\boldsymbol{l}_t$ represent the belief that the t^{th} data point is a fall-down event.

With stochasticity or an uncertain transition model, the discounted return becomes uncertain and the goal can be reformulated to optimizing the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)] = \sum_{t=0}^{T} \gamma^t \, \mathbb{E}_{\mathrm{p}(\boldsymbol{s}_t|\boldsymbol{\theta}_\pi)}[r_t]$.

A model-based policy search method consists of two key parts [2]: First, a dynamics model is learned from state transition data. Second, this dynamics model is used to learn the parameters $\boldsymbol{\theta}_\pi$ of the policy π which maximize the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)]$. We discuss both steps in the following.

3.1 An Interpretable Transition Model

We formulate a probabilistic transition model based on high-level knowledge about the Wet-Chicken benchmark. Importantly, we do not formulate a specific parametric dynamics model as would be required to derive a controller. Instead, we make assumptions on a level typically available from domain experts.

We encode that given a pair of current state and action $\hat{\boldsymbol{s}}_t = (\boldsymbol{s}_t, \boldsymbol{a}_t)$, the next state $\boldsymbol{s}_{t+1}$ is generated via the combination of three things: the deterministic flow-behaviour of the river $\boldsymbol{f}_t$, some heteroscedastic noise process $\boldsymbol{\sigma}_t$ and the possibility of falling down $\boldsymbol{\lambda}_t$. This prior imposes structure which allows us to explicitly state what we want to learn from the data and where we do not assume prior knowledge: How does the river flow? What kind of turbulences exist? When does the canoeist fall down? How do the actions influence the system?

We formulate a graphical model in Fig. 1 using the data association with GPs

(DAGP) model [5], which allows us to handle the multi-modality introduced by falling down the waterfall. We specify this separation via the marginal likelihood

$$\begin{aligned}
\mathrm{p}(\boldsymbol{s}_{t+1} \,|\, \hat{\boldsymbol{s}}_t) &= \int \mathrm{p}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{\sigma}_t, \boldsymbol{f}_t, \boldsymbol{l}_t)\, \mathrm{p}(\boldsymbol{\sigma}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{p}(\boldsymbol{f}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{p}(\boldsymbol{l}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{d}\boldsymbol{\sigma}_t\, \mathrm{d}\boldsymbol{l}_t\, \mathrm{d}\boldsymbol{f}_t, \\
\mathrm{p}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{\sigma}_t, \boldsymbol{f}_t, \boldsymbol{l}_t) &= \prod_{k=1}^{K} \mathcal{N}\Big(\boldsymbol{s}_{t+1} \,\Big|\, \boldsymbol{f}_t^{(k)}, \big(\boldsymbol{\sigma}_t^{(k)}\big)^2\Big)^{\mathbb{I}(l_t^{(k)}=1)},
\end{aligned} \tag{3}$$

where $\boldsymbol{f}_t = \big(\boldsymbol{f}_t^{(1)}, \ldots, \boldsymbol{f}_t^{(K)}\big)$ and $\mathrm{p}(\boldsymbol{l}_t \,|\, \hat{\boldsymbol{s}}_t) = \int \mathcal{M}(\boldsymbol{l}_t \,|\, \mathrm{softmax}(\boldsymbol{\lambda}_t))\, \mathrm{p}(\boldsymbol{\lambda}_t \,|\, \hat{\boldsymbol{s}}_t)\, \mathrm{d}\boldsymbol{\lambda}_t$ with $\mathcal{M}$ denoting a multinomial distribution. In our case, we use $K = 2$ modes, one for staying in the river and one for falling down the waterfall. For every data point we infer a posterior belief $\mathrm{p}(\boldsymbol{l}_t)$ about which mode the data point belongs to as we assume this separation can not be predetermined using expert knowledge. We place independent GP priors on the $\boldsymbol{f}^{(k)}$, $\boldsymbol{\sigma}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$.

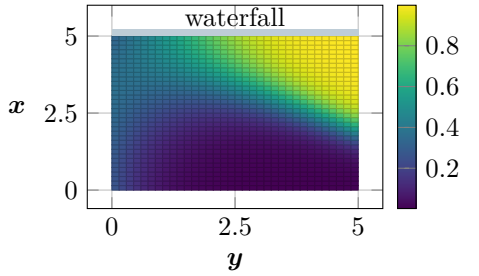
We approximate the exact posterior via a factorized variational distribution $\mathrm{q}(\boldsymbol{f}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \boldsymbol{U}) = \prod_{k=1}^{K} \prod_{t=1}^{T} \mathrm{q}(\boldsymbol{f}_t^{(k)}, \boldsymbol{u}^{(k)})\, \mathrm{q}(\boldsymbol{\lambda}_t^{(k)}, \boldsymbol{u}_{\boldsymbol{\lambda}}^{(k)})\, \mathrm{q}(\boldsymbol{\sigma}_t^{(k)}, \boldsymbol{u}_{\boldsymbol{\sigma}}^{(k)})$ which introduces variational inducing inputs and outputs $\boldsymbol{U}$ as described in [4, 5]. The variational parameters are optimized by minimizing a lower bound to the marginal likelihood which can be efficiently computed via sampling and enables stochastic optimization. For details we refer to [5]. We obtain an explicit representation of the GP posteriors during variational inference which allows us to efficiently propagate samples through the model to simulate trajectories used for policy search.

3.2 Policy Learning

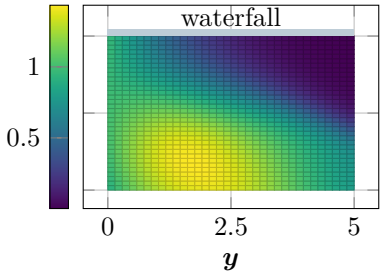
After training a transition model, we use the variational posterior $\mathrm{q}(\boldsymbol{s}_{t+1} \,|\, \hat{\boldsymbol{s}}_t)$ to train a policy by sampling roll-outs and optimizing policy parameters via stochastic gradient descent on the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)]$. The expected return is approximated using the variational posterior given by

$$\begin{aligned}
\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)] &= \sum_{t=0}^{T} \gamma^t\, \mathbb{E}_{\mathrm{p}(\boldsymbol{s}_t \,|\, \boldsymbol{\theta}_\pi)}[\boldsymbol{r}_t] \approx \sum_{t=0}^{T} \gamma^t\, \mathbb{E}_{\mathrm{q}(\boldsymbol{s}_t \,|\, \boldsymbol{\theta}_\pi)}[\boldsymbol{r}_t] \\
&= \int \sum_{t=0}^{T} \Big[\gamma^t\, \mathbb{E}_{\mathrm{q}(\boldsymbol{s}_t \,|\, \boldsymbol{\theta}_\pi)}[\boldsymbol{r}_t]\Big]\, \mathrm{p}(\boldsymbol{s_0}) \prod_{t=0}^{T-1} \mathrm{q}(\boldsymbol{s}_{t+1} \,|\, \boldsymbol{s}_t, \boldsymbol{\theta}_\pi)\, \mathrm{d}\boldsymbol{s}_0 \ldots \mathrm{d}\boldsymbol{s}_T \\
&\approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t=0}^{T} \gamma^t r_t^p.
\end{aligned} \tag{4}$$

We expand the expectation to explicitly show the marginalization of the states in the trajectory. Due to the Markovian property of the transition dynamics, the integral factorizes along t. The integral is approximated by averaging over P samples propagated through the model starting from a known distribution of initial states $\mathrm{p}(\boldsymbol{s}_0)$. State transitions can efficiently be sampled from the variational posterior of the dynamics model by repeatedly taking independent samples of the different GPs.



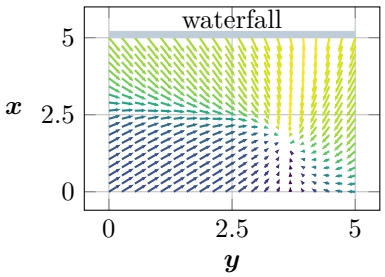
(a) Fall-down probability $\boldsymbol{\lambda}$



(b) heteroscedastic turbulence $\boldsymbol{\sigma}^{(1)}$

N	NFQ	GP	DAGP
100	0.66 ± 0.16	$\mathbf{1.41 \pm 0.01}$	1.18 ± 0.09
250	1.71 ± 0.07	1.54 ± 0.01	$\mathbf{2.33 \pm 0.01}$
500	1.60 ± 0.10	1.56 ± 0.01	$\mathbf{2.25 \pm 0.01}$
1000	1.99 ± 0.06	2.13 ± 0.01	$\mathbf{2.32 \pm 0.01}$
2500	$\mathbf{2.26 \pm 0.02}$	1.91 ± 0.01	$\mathbf{2.28 \pm 0.01}$
5000	$\mathbf{2.33 \pm 0.01}$	1.91 ± 0.01	2.28 ± 0.01

(c) Comparison of expected returns



(d) A successful Wet-Chicken policy

Fig. 2: The separation of different aspects of the Wet-Chicken benchmark yields interpretable information about the probability to fall down the waterfall and the turbulence intensity. Successful policies can be learned based on 250 observations, while about 2500 observations are needed for NFQ.

The expected return in (4) can be optimized using stochastic gradient descent via the gradients $\nabla_{\theta_\pi} J^\pi(\theta_\pi) \approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t=0}^{T} \gamma^t \nabla_{\theta_\pi} r_t^p$ of the Monte Carlo approximation as they are an unbiased estimator of the true gradient. The gradients of the samples can be obtained using automatic differentiation tools such as TensorFlow [1]. The P roll-outs can trivially be parallelized. Importantly, we only need a small number of Monte Carlo samples at every iteration, since we use the gradients of the samples directly.

4 Results

To solve the Wet-Chicken problem, we first train the dynamics model on batch data sampled from the true dynamics. The benchmark has a two-dimensional state and action spaces from which we sample uniform random transitions with varying N in the range 100 to 5000. With $N \geq 250$, our model is able to identify the underlying dynamics. Figures 2a and 2b show how the model has successfully identified the probabilities of falling down the waterfall and the amplitude of turbulence, both with respect to the action $(0, 0)$. We are presented with easily separable posterior belief about different aspects of the Wet-Chicken benchmark. This belief can be reasoned about with experts to evaluate the training result.

Next, we train a neural policy. We sample initial states from the training data, use a horizon of $T = 5$ steps and average over $P = 20$ samples with $\gamma = 0.9$. We use a two-layer neural network with 20 ReLU-activated units each as our policy parametrization. Figure 2d shows an example policy with a trade-off between the unpredictability on the left and the uncontrollable speed on the right.

In Table 2c, we show expected returns averaged over 10 experiments with standard errors. Applying random actions yields a return of about 1.5 and a return above 2.2 indicates that a proper trade-off has been found. We compare our method to a standard GP as the dynamics model and to the model-free NFQ [7] trained for 20 full model learning and sampling iterations using a neural network with one 10-unit hidden layer with sigmoid activations. The GP cannot model heteroscedastic noise or multi-modality. It does not represent the dynamics well enough to derive a policy, illustrating our need for a more structured model. Given enough data, NFQ is able to find successful policies. However, our method requires about an order of magnitude less data, due to the high-level prior knowledge incorporated via the dynamics model.

5 Conclusion

In this paper, we demonstrated how expert knowledge can be incorporated in probabilistic policy search by imposing Bayesian structure on the learning problem. We derived an efficient inference scheme and showed how our approach can solve the Wet-Chicken benchmark, yielding human-interpretable insights about the underlying dynamics and significantly increasing data efficiency.

References

[1] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". In: (2015). Software available from tensorflow.org.

[2] Marc Deisenroth and Carl E. Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 465–472.

[3] Alexander Hans and Steffen Udluft. "Efficient Uncertainty Propagation for Reinforcement Learning with Limited Data". In: *International Conference on Artificial Neural Networks*. Springer, 2009, pp. 70–79.

[4] James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. "Scalable Variational Gaussian Process Classification". In: *Journal of Machine Learning Research* 38 (2015), pp. 351–360.

[5] Markus Kaiser et al. "Data Association with Gaussian Processes". In: (Oct. 16, 2018). arXiv: `1810.07158 [cs, stat]`.

[6] Sascha Lange, Thomas Gabel, and Martin Riedmiller. "Batch Reinforcement Learning". In: *Reinforcement Learning*. Springer, 2012, pp. 45–73.

[7] Martin Riedmiller. "Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method". In: *European Conference on Machine Learning*. Springer, 2005, pp. 317–328.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 1998. 322 pp. ISBN: 978-0-262-19398-6.