Memory Development with Heteroskedastic Bayesian Last Layer Probabilistic Deep Neural Networks

George Velentzas, Costas S Tzafestas, Mehdi Khamassi

Extended Abstract: Memory Development with Heteroskedastic Bayesian Last Layer Probabilistic Deep Neural Networks

Georgios Velentzas[1], Costas Tzafestas[2], Mehdi Khamassi[1]

***Abstract*— Learning world models in model-based Reinforcement Learning (MBRL) enables sample-efficient learning, while avoiding model-bias via uncertainty estimates of the transition dynamics. Moreover, it enables using Model Predictive Control (MPC) or Recurrent Neural Networks (RNN) for efficient action planning. However, current uncertainty estimates of the prediction (transition) step are not provided in a closed-form that could potentially be used for analytically estimating changes in the dynamics function. Here, we propose a hybrid method to capture both the aleatoric (data) uncertainty via Probabilistic Deep Neural Networks (PrDNN), and the epistemic (model) uncertainty through a Bayesian Last Layer.**

I. INTRODUCTION AND RELATED WORK

Learning world models of the environment dynamics enables Model Predictive Control (MPC), a process through which an agent can anticipate environment states resulting from specific actions, allowing to plan future action sequences. This has been widely used in MBRL and applied to real-world robotic tasks (e.g., target reaching with a robotic arm, locomotion control of a half-cheetah, etc.) [1]. However, when uncertainty of transitions is overlooked, model-bias [2] and sample-inefficiency are induced, as described in [3].

Probabilistic MPC methods have been proposed [4], while methods for estimating the uncertainty of predictions in deep learning have been thoroughly studied [5], [6], [7], [3], [8], [9]. The validity of some approaches has been criticized [10], nevertheless, the enriched usability of such estimates is unquestionable; for example, adding an epistemic component to the cost function would be a way to instantiate Active Inference [11] in the Bayesian RL framework. However, these methods are usually applied to stationary tasks, while the absence of an analytic expression of the total uncertainty of the predictive distributions limits the tools needed for real-time capture of changes in the dynamics function.

Here, we derive an analytic way of estimating the predictive (transition) distribution of the dynamics function by incorporating multivariate Bayesian regression at the last layer of a PrDNN [3], [12]. Including heteroskedastic noise, we utilize both types of uncertainty [13], aleatoric (data) and epistemic (model), to recognize changes in the dynamics that require to learn and memorize different world models. We introduce a memory schema, and we show numerical simulations of a 3-dof robotic arm in a target-reaching task.

*This work was supported by the European Commission's CAVAA Project (EIC 101071178) and the CNRS-INS2I APIER Project.

[1]Sorbonne University, CNRS, Institute of Intelligent Systems and Robotics, Paris, France velentzas@isir.upmc.fr, mehdi.khamassi@sorbonne-universite.fr

[2]Division of Signals, Control and Robotics, School of Electrical and Computer Engineering, NTUA, Greece. ktzaf@cs.ntua.gr

II. METHODS

We assume a world of infinite environments, where each environment with index $e = 1, 2, ...$ is characterised by its own dynamics. If $\mathbf{s} \in \mathbb{R}^p$ denotes the state representation vector, $\mathbf{u} \in \mathbb{R}^u$ denotes the action vector, $\mathbf{x} = [\mathbf{s}^\top, \mathbf{u}^\top]^\top$, where $\mathbf{x} \in \mathbb{R}^s$, denotes the concatenation of those and $\mathbf{y} = \mathbf{s}' - \mathbf{s}$ denotes the state displacement vector after a control $\mathbf{u}$ is applied for a time period of dt, we make the assumption that the transition dynamics of the environment with index e is described by $\mathbf{y} = f^e(\mathbf{x}) + n^e(\mathbf{x})$, where $f^e : \mathbb{R}^s \mapsto \mathbb{R}^p$ is the dynamics function, and $n^e(\mathbf{x}) \sim \mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma}^e(\mathbf{x}))$ is a heteroskedastic transition noise which encapsulates aleatoric uncertainty. With $\mathcal{P}^e$ denoting the state displacement transition distribution, the above implies that $\mathcal{P}^e(\mathbf{y}|\mathbf{x}) = \mathcal{N}(f^e(\mathbf{x}), \mathbf{\Sigma}^e(\mathbf{x}))$. The agent's objective is to be able to perform a task in each different environment, without having any prior knowledge of the environment's index, and quickly adapt after each transition. In this work, we assume that each interaction with the environment is taking place in episodes, where each episode's time horizon is unknown.

A. Modeling the dynamics function for each environment

If $\mathbf{X}$ and $\mathbf{Y}$ denote the matrices of n transition observations from a single environment, where the i-th rows of $\mathbf{X}$ and $\mathbf{Y}$ are $\mathbf{x}_i^\top$ and $\mathbf{y}_i^\top$ respectively, we model the problem as multivariate Bayesian linear regression with learned non-linear features and heteroskedasticity, encapsulating both the epistemic and aleatoric uncertainty, such as

$$\mathbf{Y} = \Phi_{\boldsymbol{\omega}}(\mathbf{X})\mathbf{B} + \mathbf{E}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{X}) \tag{1}$$

where $\mathbf{\Phi}_{\boldsymbol{\omega}}(\mathbf{X})$ is a $n \times d$ matrix with its i-th row being $\boldsymbol{\phi}_{\boldsymbol{\omega}}(\mathbf{x}_i)^\top$, and $\boldsymbol{\phi}_{\boldsymbol{\omega}} : \mathbb{R}^s \mapsto \mathbb{R}^d$ denoting a parameterized (by weights $\boldsymbol{\omega}$) non-linear feature mapping function, $\mathbf{B}$ is a $d \times p$ matrix of Bayesian weights. If $\mathbf{\Sigma}_{\boldsymbol{\omega},\boldsymbol{\theta}} : \mathbb{R}^s \mapsto \mathcal{A}$ denotes a parameterized function (by weights $\boldsymbol{\omega}, \boldsymbol{\theta}$), with $\mathcal{A}$ being the set of positive definite matrices, $\mathbf{E}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{X})$ is a $n \times p$ random matrix with independent rows, where the i-th row is $\mathbf{e}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{x}_i)^\top$, with $\mathbf{e}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{x}_i) \sim \mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{x}_i))$. For simplicity, we omit the parameter weights notation and take the matrix vectorization of both sides of equation Eq.1, thus,

$$\mathrm{vec}(\mathbf{Y}^\top) = [\Phi(\mathbf{X}) \otimes \mathbf{I}_p]\,\mathrm{vec}(\mathbf{B}^\top) + \mathrm{vec}(\mathbf{E}(\mathbf{X})^\top) \tag{2}$$

We denote $y = \mathrm{vec}(\mathbf{Y}^\top)$, $\beta = \mathrm{vec}(\mathbf{B}^\top)$, $\varepsilon = \mathrm{vec}(\mathbf{E}(\mathbf{X})^\top)$, $\mathbf{\Phi}_I = \Phi(\mathbf{X}) \otimes \mathbf{I}_p$, therefore Eq. 2 can be simply written as $y = \mathbf{\Phi}_I\beta + \varepsilon$, which implies that $y|\beta \sim \mathcal{N}_{n\times p}(\mathbf{\Phi}_I\beta, \mathbf{S}_{NET})$, with $\mathbf{S}_{NET} = \mathrm{diag}(\mathbf{\Sigma}(\mathbf{x}_1), ..., \mathbf{\Sigma}(\mathbf{x}_n))$. This notation is convenient, as y denotes the concatenation of state displacement vectors, with $y = \text{vec}([\mathbf{y}_1, ..., \mathbf{y}_n])$, thus for $n = 1$, $y = \mathbf{y}_1$. Setting a matrix-normal prior for the matrix of Bayesian weights $\mathbf{B}$, such that $\mathbf{B} \sim \mathcal{MN}(\mathbf{B}_0, \mathbf{U}, \mathbf{V})$, is equivalent with $\beta \sim \mathcal{N}(\beta_0, \mathbf{\Sigma}_\beta)$, such that $\beta_0 = \text{vec}(\mathbf{B}_0^\top)$ and $\mathbf{\Sigma}_\beta = \mathbf{U} \otimes \mathbf{V}$. The posterior distribution is then analytically derived as $\beta|y \sim \mathcal{N}(\boldsymbol{\mu}_{\beta|y}, \mathbf{\Sigma}_{\beta|y})$, with $\mathbf{\Sigma}_{\beta|y} = [\mathbf{\Phi}_I^\top \mathbf{S}_{NET}^{-1} \mathbf{\Phi}_I + \mathbf{\Sigma}_\beta^{-1}]^{-1}$ and $\boldsymbol{\mu}_{\beta|y} = \mathbf{\Sigma}_{\beta|y}[\mathbf{\Phi}_I^\top \mathbf{S}_{NET}^{-1} y + \mathbf{\Sigma}_\beta^{-1}\beta_0]$. Finally, the predictive distribution of the concatenation of the state displacement vectors for given inputs $\mathbf{x}_1^*, \mathbf{x}_2^*, ..., \mathbf{x}_k^*$ is,

$$\mathcal{P}(y|\mathbf{x}_1^*, ..., \mathbf{x}_k^*) = \mathcal{N}(\mathbf{\Phi}_I^* \boldsymbol{\mu}_{\beta|y}, \mathbf{\Phi}_I^* \mathbf{\Sigma}_{\beta|y} \mathbf{\Phi}_I^{*\top} + \mathbf{S}_{NET}^*) \quad (3)$$

where $\mathcal{P}$ is the transition model that encapsulates the full uncertainty (aleatoric and epistemic), $\mathbf{\Phi}_I^* = \Phi(\mathbf{X}^*) \otimes \mathbf{I}_p$, and $\mathbf{S}_{NET}^* = \text{diag}(\mathbf{\Sigma}(\mathbf{x}_1^*), ..., \mathbf{\Sigma}(\mathbf{x}_k^*))$. For estimating the weight parameters $\boldsymbol{\omega}, \boldsymbol{\theta}$, and therefore, for analytically deriving $\boldsymbol{\mu}_{\beta|y}$ and $\mathbf{\Sigma}_{\beta|y}$, we use the transition observation matrices $\mathbf{X}, \mathbf{Y}$ to train a PrDNN [3] and extract $\mathbf{\Phi}_I$ and $\mathbf{S}_{NET}$. The first N layers of the network are used to model the feature mapping function $\phi_{\boldsymbol{\omega}}(\cdot)$, while the last layer is comprised by a linear and a non linear part. The linear part with weights $\boldsymbol{\zeta}$ predicts the state displacement vector as $\mathbf{y}_{\boldsymbol{\omega},\boldsymbol{\zeta}}(\mathbf{x})$, and the non-linear part with weights $\boldsymbol{\theta}$ outputs the covariance matrix $\mathbf{\Sigma}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{x})$. The loss function used is the negative log-likelihood, which for one training example $(\mathbf{x}_i, \mathbf{y}_i)$ simplifies to

$$\log|\mathbf{\Sigma}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{x}_i)| + (\mathbf{y}_i - \mathbf{y}_{\boldsymbol{\omega},\boldsymbol{\zeta}}(\mathbf{x}_i))^\top \mathbf{\Sigma}_{\boldsymbol{\omega},\boldsymbol{\theta}}(\mathbf{x}_i)^{-1}(\mathbf{y}_i - \mathbf{y}_{\boldsymbol{\omega},\boldsymbol{\zeta}}(\mathbf{x}_i))$$

After the network is trained, the full batch of data is used once for extracting $\mathbf{\Phi}_I$ and $\mathbf{S}_{NET}$. During the prediction stage (i.e., model is used for MPC), the linear part is removed and substituted by the Bayesian layer which encapsulates the epistemic uncertainty of the mapping between the features and the predictions. The non-linear part is kept and represents the aleatoric uncertainty, providing the full uncertainty quantification described by the predictive distribution of Eq.3.

B. Memory Development and Adaptive Control

We make the assumption that the dynamics function $f^e(\cdot)$ and the transition noise $n^e(\cdot)$ are static within environments, but might be different between environments. The agent collects experiences and develops an incremental memory with $\mathcal{M} = \bigcup_j \mathcal{M}_j$, where $\mathcal{M}_j = \{\mathcal{P}_j, \mathcal{D}_j\}$ denotes a memory block consisting of a dynamics model $\mathcal{P}_j$ and the transition observations $\mathcal{D}_j = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ from which $\mathcal{P}_j$ is trained. The experiment is performed in episodes, where at each episode the agent is arbitrarily placed in an environment. During the first episode, the transition observations $\mathcal{D}_1 = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^T$ are collected, the probabilistic transition function $\mathcal{P}_1$ is trained, and the memory is populated as $\mathcal{M} = \{\mathcal{M}_1\}$, with $\mathcal{M}_1 = \{\mathcal{P}_1, \mathcal{D}_1\}$.

In each subsequent episode, the last recently used dynamics model is utilized (after the first episode we default this to $\mathcal{P}_1$) for taking actions using MPC with trajectory sampling [3] (sampling from the predictive distribution) and a sample-efficient Cross Entropy Method [14]. At each time step t of the control phase, a sliding window of length W of the most recent transition observations $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=t-W}^t$ is used for computing the likelihood $\mathcal{P}_j(y|\mathbf{x}_{t-W}, ..., \mathbf{x}_t)$ for all dynamic models j in the memory. The index of the dynamics model utilized at timestep $t+1$ is then $\arg\max_j \mathcal{P}_j(y|\mathbf{x}_{t-W}, ..., \mathbf{x}_t)$.

When the episode ends, the new transition data are used to create a candidate memory block $\mathcal{M}_k$. We then compare the predictive distribution of $\mathcal{P}_k$ with the predictive distributions of all the dynamic models in memory $\mathcal{M}$ using a bi-directional hypothesis test. In more specific, when comparing dynamic models $\mathcal{P}_k, \mathcal{P}_j$, the mean predictions of the model $\mathcal{P}_k$ on the inputs $\mathbf{x}$ in $\mathcal{D}_j$ are used to perform a $\mathcal{X}^2$ test on the predictive distribution of $\mathcal{P}_j$ and vice versa. If both the p-values are over a threshold, the two memory blocks are merged to a new memory block. If not, the candidate memory block $\mathcal{M}_k$ is inserted in the main memory $\mathcal{M}$.

III. EXPERIMENTS, RESULTS AND CONCLUSION

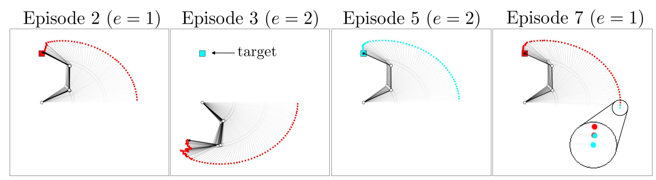

Fig. 1. Robotic arm in a target reaching task. The target's color in each episode denotes the environment's index. The endpoint's color denotes the model $\mathcal{P}_j$ used for MPC at the corresponding timestep within each episode.

Figure 1 shows the results obtained in a target reaching task with a simulated 3-dof robotic arm. During Episode 0 (not shown in the figure), random movements are performed for 1 second in environment $e = 1$ (a control vector is applied every 0.01 seconds, resulting in 100 transition samples). The agent then trains and employs the dynamics model $\mathcal{P}_1$, populating the memory with memory block $\mathcal{M}_1$. In Episode 2 (after 2 seconds of experience), MPC enables the arm to reach the target, demonstrating sample efficient learning. The third episode takes place in environment $e = 2$, where the polarity of two of the actuators is reversed. Due to the different dynamics, the use of model $\mathcal{P}_1$ leads to inappropriate trajectory. The deviation of the predicted dynamics taking into account epistemic and aleatoric uncertainty leads to the creation of a new memory block for learning a different dynamics model: $\mathcal{P}_2$. In Episode 5, $\mathcal{P}_2$ is already sufficient for MPC to drive the arm to the target. In Episode 7 the environment is reversed to $e = 1$. In the first two timesteps (0.02 sec) $\mathcal{P}_2$ is initially employed, but the change in the dynamics is detected and $\mathcal{P}_1$ is re-engaged from timestep 3 onward, with MPC being able to drive the arm to the target. In our experiments, the agent was able to capture all changes in 4 different environments by demonstrating fast adaptation and employing the appropriate model, without populating the memory with redundant memory blocks. Due to space limitations we only show a single example.

In conclusion, our method for deriving predictive distributions with full uncertainty estimates, by disentangling epistemic uncertainty from aleatoric uncertainty, enabled fast recognition and adaptation to changes in the dynamics of the environment. This paves the way for new applications of world models in model-based deep reinforcement learning.

REFERENCES

[1] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[2] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 1–8.

[3] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[4] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1701–1710.

[5] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[6] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," *Advances in neural information processing systems*, vol. 30, 2017.

[7] I. Osband, J. Aslanides, and A. Cassirer, "Randomized prior functions for deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[8] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[9] M. Okada, N. Kosaka, and T. Taniguchi, "Planet of the bayesians: Reconsidering and improving deep planning network by incorporating bayesian inference," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5611–5618.

[10] I. Osband, "Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout," in *NIPS workshop on bayesian deep learning*, vol. 192, 2016.

[11] T. Taniguchi, S. Murata, M. Suzuki, D. Ognibene, P. Lanillos, E. Ugur, L. Jamone, T. Nakamura, A. Ciria, B. Lara *et al.*, "World models and predictive coding for cognitive and developmental robotics: frontiers and challenges," *Advanced Robotics*, pp. 1–27, 2023.

[12] M. Seitzer, A. Tavakoli, D. Antic, and G. Martius, "On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks," *arXiv preprint arXiv:2203.09168*, 2022.

[13] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?" *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[14] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius, "Sample-efficient cross-entropy method for real-time planning," in *Conference on Robot Learning*. PMLR, 2021, pp. 1049–1065.