Useful Uncertainties in Reinforcement Learning

Christian Weilbach

October 29, 2018

University of British Columbia

Overview

1. About me
2. Motivation
3. Types of Uncertainties
4. Example of Uncertainties in Reinforcement Learning
5. Methods for epistemic Uncertainty

About me

About me

- computer science PhD student with Frank
- love functional programming (Clojure) and (de)composable systems
- have a background in distributed databases
- like Bayesian statistics as composable statistics and unified modeling framework
  $\Rightarrow$ like probabilistic programming languages: Anglican, pyro
- Bachelor thesis on training RBMs with STDP in the HBP
- studied philosophy and cultural anthropology in a former life

Motivation

Why do we need uncertainties?

- reason about model confidence
- reason about predictive confidence
- prevent adversarial examples, Gal and Smith 2018
- active learning

Types of Uncertainties

Aleatoric vs. Epistemic Uncertainty

- *aleatoric*: inherent stochastic uncertainty in the data
- *epistemic*: subjective uncertainty of the model, i.e. marginalized posterior predictive over the model distribution

$$p(y|x) = \int p(y|x, \theta) d\theta \tag{1}$$

Uncertainties explained through Vision

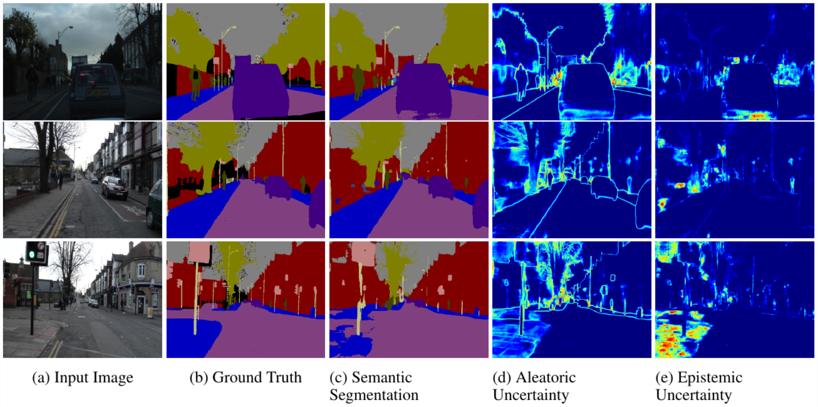

Figure 1: Comparison of uncertainties in Vision, Kendall and Gal 2017

Comparison of models Chua et al. 2018

Model		Aleatoric uncertainty	Epistemic uncertainty
Baseline Models			
Deterministic NN	(D)	No	No
Probabilistic NN	(P)	Yes	No
Deterministic ensemble NN	(DE)	No	Yes
Gaussian process baseline	(GP)	Homoscedastic	Yes
Our Model			
Probabilistic ensemble NN	(PE)	**Yes**	**Yes**

Example of Uncertainties in Reinforcement Learning

Approach of Chua et al. 2018

- Model-based RL works well in small data regime
- *Problem:* not competitive to model-free RL in large data regime
- Neural Network models overfit in model-based RL too early
- Contribution: can be addressed by incorporating aleatoric uncertainties in NN

Probabilistic NN-model for continuous probabilistic output

$$\text{loss}_{\text{Gauss}}(\theta) = \sum_{n=1}^{N} \left[\mu_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n) - \boldsymbol{s}_{n+1}\right]^{\top} \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{s}_n, \boldsymbol{a}_n) \left[\mu_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n) - \boldsymbol{s}_{n+1}\right] \tag{2}$$

$$+ \log \det \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{s}_n, \boldsymbol{a}_n) \tag{3}$$

Note: For discrete output we can just pick a softmax distribution as usual.

Diagram

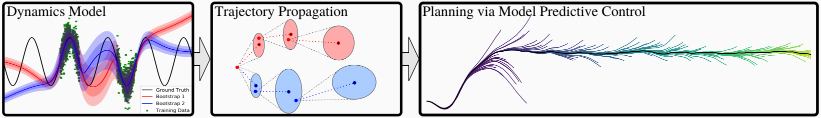

Figure 2: System decomposition of Chua et al. 2018

Cross-Entropy Method (CEM)

Idea: Learn proposal distribution f for Importance Sampling of rare events:

$$\mathrm{KL}\left(g^*||g\right) = \int g^*(\mathbf{x}) \ln \frac{g^*(\mathbf{x})}{g(\mathbf{x})} \mathrm{d}\mathbf{x} = \mathbb{E}\left[\ln \frac{g^*(\mathbf{X})}{g(\mathbf{X})}\right], \quad \mathbf{X} \sim g^* \tag{4}$$

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmin}} \mathrm{KL}\left(g^*||f(\cdot;\mathbf{v})\right) \tag{5}$$

$$= \underset{\mathbf{v}}{\operatorname{argmax}} \mathbb{E}_{\mathbf{u}} \mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}} \ln f(\mathbf{X};\mathbf{v}) \tag{6}$$

$$= \underset{\mathbf{v}}{\operatorname{argmax}} \mathbb{E}_{\mathbf{w}} \mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}} \ln f(\mathbf{X};\mathbf{v}) \frac{f(\mathbf{X};\mathbf{u})}{f(\mathbf{X};\mathbf{w})} \tag{7}$$

http://iew3.technion.ac.il/CE/files/Misc/tutorial.pdf

Cross-Entropy Method (CEM)

Algorithm 2.1 (CE Algorithm for Rare-Event Estimation) *Given the sample size N and the parameter ϱ, execute the following steps.*

1. *Define* $\widehat{\mathbf{v}}_0 = \mathbf{u}$. *Let* $N^{\mathrm{e}} = \lceil \varrho N \rceil$. *Set* $t = 1$ *(iteration counter).*
2. *Generate* $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\text{iid}} f(\cdot; \widehat{\mathbf{v}}_{t-1})$. *Calculate* $S_i = S(\mathbf{X}_i)$ *for all* i, *and order these from smallest to largest:* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. *Let* $\widehat{\gamma}_t$ *be the sample* $(1 - \varrho)$*-quantile of performances; that is,* $\widehat{\gamma}_t = S_{(N - N^{\mathrm{e}} + 1)}$. *If* $\widehat{\gamma}_t > \gamma$, *reset* $\widehat{\gamma}_t$ *to* γ.
3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *to solve the stochastic program* (7), *with* $\mathbf{w} = \widehat{\mathbf{v}}_{t-1}$. *Denote the solution by* $\widehat{\mathbf{v}}_t$.
4. *If* $\widehat{\gamma}_t < \gamma$, *set* $t = t + 1$ *and reiterate from Step 2; otherwise, proceed with Step 5.*
5. *Let* $T = t$ *be the final iteration counter. Generate* $\mathbf{X}_1, \ldots, \mathbf{X}_{N_1} \sim_{\text{iid}} f(\cdot; \widehat{\mathbf{v}}_T)$ *and estimate* ℓ *via importance sampling, as in* (3) *with* $\mathbf{u} = \widehat{\mathbf{v}}_T$.

Figure 3: `http://iew3.technion.ac.il/CE/files/Misc/tutorial.pdf`

Probabilistic Ensembles with Trajectory Sampling (PETS) method

Algorithm 1 Our model-based MPC algorithm '*PETS*':

Initialize data $\mathbb{D}$ with a random controller for one trial.
for Trial $k = 1$ to K **do**
    Train a *PE* dynamics model $\widetilde{f}$ given $\mathbb{D}$.
    for Time $t = 0$ to TaskHorizon **do**
        for Actions sampled $\boldsymbol{a}_{t:t+T} \sim \text{CEM}(\cdot)$, 1 to NSamples **do**
            Propagate state particles $\boldsymbol{s}_\tau^p$ using *TS* and $\widetilde{f}|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
            Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_\tau^p, \boldsymbol{a}_\tau)$
            Update $\text{CEM}(\cdot)$ distribution.
        Execute first action $\boldsymbol{a}_t^*$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^*$.
        Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t^*, \boldsymbol{s}_{t+1}\}$.

Figure 4: The model-based MPC algorithm *PETS*. Chua et al. 2018

Results

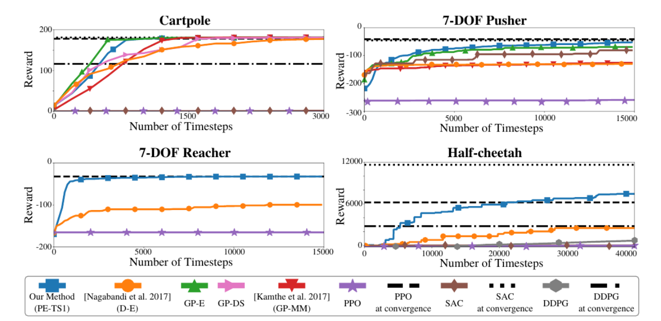

Figure 5: Performance of *PETS* on different tasks. Chua et al. 2018

Methods for epistemic Uncertainty

Bayesian Neural Networks

- Regularizer: Dropout as Bayesian Approximation, Gal and Ghahramani 2016
- Optimizer: Stochastic Gradient HMC
- dedicated Variational Inference based models

Motivation for SGHMC

Method/Dataset	Boston Housing	Yacht Hydrodynamics	Concrete	Wine Quality Red
SGHMC (best average)	-3.474 $\pm$ 0.511	-13.579 $\pm$ 0.983	-4.871 $\pm$ 0.051	-1.825 $\pm$ 0.75
SGHMC (tuned per dataset)	**-2.489 $\pm$ 0.151**	-1.753 $\pm$ 0.19	-4.165 $\pm$ 0.723	-1.287 $\pm$ 0.28
SGHMC (scale-adapted)	-2.536 $\pm$ 0.036	**-1.107 $\pm$ 0.083**	**-3.384 $\pm$ 0.24**	**-1.041 $\pm$ 0.17**
VI	-2.903 $\pm$ 0.071	-3.439 $\pm$ 0.163	-3.391 $\pm$ 0.017	-0.980 $\pm$ 0.013
PBP	-2.574 $\pm$ 0.089	-1.634 $\pm$ 0.016	**-3.161 $\pm$ 0.019**	**-0.968 $\pm$ 0.014**

Figure 6: Bayesian NN comparison. Springenberg et al. 2016

Monte Carlo

- sample from posterior distribution over model parameters
- *Goal*: draw samples from posterior distribution $\pi(\theta)$, e.g. to get uncertainties
- MCMC is often used for Bayesian inference
- uses a proposal distribution q (typically Gaussian) for diffusion
- problem: high rejection rate in Metropolis-Hastings acceptance in high dimensions:

$$A(\theta_{i+1}|\theta_i) = \min(1, \underbrace{\frac{\pi(\theta_{i+1})q(\theta_i|\theta_{i+1})}{\pi(\theta_i)q(\theta_{i+1}|\theta_i)}}_{\text{acceptance probability}}) \tag{8}$$

Hamiltonian Monte Carlo

- idea: exploit gradient to move around in typical set
  $\Rightarrow$ better mixing
- used for example in the Stan probabilistic programming environment

$$\mathcal{M} \overset{?}{\to} \mathcal{TM}^* \xrightarrow{\text{Hamiltonian Flow}} \mathcal{TM}^* \to \mathcal{M} \tag{9}$$

$$\theta \to (\theta, p) \overset{H}{\to} (\theta', p') \to \theta' \tag{10}$$

$$H(\theta, p) := \underbrace{-\log \pi(p|\theta)}_{\substack{\text{“kinetic'''} \\ K(\theta,p)}} \underbrace{-\log \pi(\theta)}_{\substack{\text{“potential'''} \\ U(\theta)}} \tag{11}$$

$$\tag{12}$$

Hamiltonian system analogy

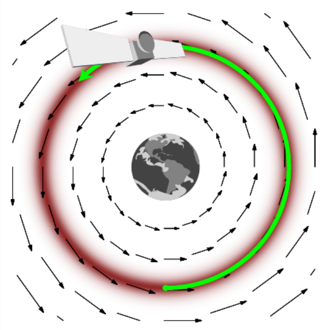

Figure 7: Properly adjusted momentum to stay in orbit, i.e. typical set around the mode. Betancourt 2017

Hamiltonian Flow

well-defined physical dynamics:

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial p} = \frac{\partial K}{\partial p} \tag{13}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial \theta} = -\frac{\partial K}{\partial \theta} - \frac{\partial U}{\partial \theta} \tag{14}$$

Problem: How to pick $\pi(p|\theta)$? $K(\theta, p) := \frac{1}{2} p^T \mathbf{M}^{-1} p + ...$ (inverse metric to euclidean metric in sample space) Betancourt 2017

Hamiltonian Flow

$$d\theta = \mathbf{M}^{-1} p \cdot dt \tag{15}$$

$$dp = -\nabla_\theta U(\theta) \cdot dt \tag{16}$$

Implementation problems

1. stay on level-set: symplectic integrators, e.g. leapfrog
2. integration time
3. each step needs a pass through the *whole dataset* for gradient and the Metropolis Hasting-step because of discretization, otherwise we introduce bias

Stochastic Gradient Hamiltonian Monte Carlo
Chen, Fox, and Guestrin 2014

- use stochastic mini-batch sampling for gradient:
  $\nabla \tilde{U} \approx \nabla U(\theta) + \mathcal{N}(0; \mathbf{V}(\theta))$ (CTL)
- equivalent to stochastic gradient with momentum + noise calibration
- we use a Riemannian preconditioning mass matrix to adapt learning rates in the beginning for $\mathbf{M}^{-1}$, but is a free parameter.
  Springenberg et al. 2016
- *Sidenote 1*: still significantly slower in convergence than Adam
- *Sidenote 2*: bias not clear. Betancourt 2015

Naive SGHMC

$$d\theta = \mathbf{M}^{-1} p \cdot dt \tag{17}$$

$$dp = -\nabla_\theta U(\theta) \cdot dt + \mathcal{N}(0; 2B(\theta)dt) \tag{18}$$

Problem: entropy increase: $\lim_{t \to \infty} \pi_t(\theta) = \text{unif.}$

Proper SGHMC

$$d\theta = \mathbf{M}^{-1} p \cdot dt \tag{19}$$

$$dp = -\nabla_\theta \tilde{U}(\theta) \cdot dt - \underbrace{\mathbf{M}^{-1} B p}_{\text{friction}} + \mathcal{N}(0; 2B(\theta)dt) \tag{20}$$

SGD with momentum decay:

$$\triangle\theta = p \cdot \eta \tag{21}$$

$$\triangle p = -\nabla_\theta \tilde{U}(\theta)\eta - \underbrace{\alpha p}_{\text{decay}} \tag{22}$$

Sampling leaves distribution $\pi(\theta)$ invariant (Fokker Planck Equation).
Chen, Fox, and Guestrin 2014

Make Bayesian Neural Networks even sense?

- How well can we explore a million dimensional parameter space?

Demo

Questions?

References

Michael Betancourt. "The Fundamental Incompatibility of Scalable Hamiltonian Monte Carlo and Naive Data Subsampling". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015, pp. 533–540. URL: http://jmlr.org/proceedings/papers/v37/betancourt15.html.

Michael Betancourt. *A Conceptual Introduction to Hamiltonian Monte Carlo*. 2017. eprint: arXiv:1701.02434.

Tianqi Chen, Emily B. Fox, and Carlos Guestrin. *Stochastic Gradient Hamiltonian Monte Carlo*. 2014. eprint: arXiv:1402.4102.

Kurtland Chua et al. *Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models*. 2018. arXiv: 1805.12114v1 [cs.LG].

Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: (2016).

Yarin Gal and Lewis Smith. *Sufficient Conditions for Idealised Models to Have No Adversarial Examples: a Theoretical and Empirical Study with Bayesian Neural Networks*. 2018. arXiv: 1806.00667v3 [stat.ML].

Alex Kendall and Yarin Gal. *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?* 2017. arXiv: 1703.04977v2 [cs.CV].

Jost Tobias Springenberg et al. "Bayesian Optimization with Robust Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 4134–4142. URL: http://papers.nips.cc/paper/6117-bayesian-optimization-with-robust-bayesian-neural-networks.pdf.