Appendix

1 Bayesian neural networks (BNNs)

We demonstrate the behavior of a BNN [1–3] when trained on simple regression data. Figure 1 shows a snapshot of the behavior of the BNN during training. In this figure, the red dots represent the regression training data, which has a 1-dim input x and a 1-dim output. The input to the BNN is constructed as $x = [x, x^2, x^3, x^4]$. The green dots represent BNN predictions, each for a differently sampled θ value, according to $q(\cdot; \phi)$. The color lines represent the output for different, but fixed, θ samples. The shaded areas represent the sampled output mean plus-minus one and two standard deviations.
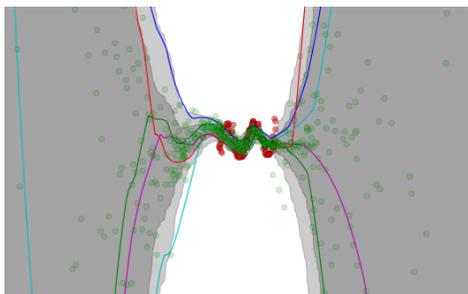
Figure 1: BNN output on a 1D regression task. Shaded areas: sampled output mean $\pm$ one/two standard deviations. Red dots: targets; green dots: prediction samples. Colored lines: neural network functions for different $\theta \sim q(\cdot; \phi)$ samples.
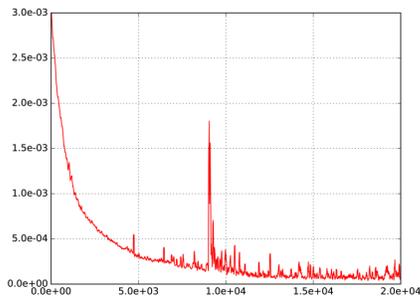
Figure 2: Just before iteration 10,000 we introduce data outside the training data range to the BNN. This results in a KL divergence spike, showing the model's surprise.

The figure shows that the BNN output is very certain in the training data range, while having high uncertainty otherwise. If we introduce data outside of this training range, or data that is significantly different from the training data, it will have a high impact on the parameter distribution $q(\theta; \phi)$. This is tested in Figure 2: previously unseen data is introduced right before training iteration 10,000. The KL divergence from posterior to prior (y-axis) is set out in function of the training iteration number (x-axis). We see a sharp spike in the KL divergence curve, which represents the BNN's surprise about this novel data. This spike diminishes over time as the BNN learns to fit this new data, becoming less surprised about it.

2 Experimental setup

In case of the classic tasks CartPole, CartPoleSwingup, DoublePendulum, and MountainCar, as well as in the case of the hierarchical task SwimmerGather, the dynamics BNN has one hidden layer of 32 units. For the locomotion tasks Walker2D and HalfCheetah, the dynamics BNN has two hidden layers of 64 units each. All hidden layers have rectified linear unit (ReLU) nonlinearities, while no nonlinearity is applied to the output layer. The number of samples drawn to approximate the variational lower bound expectation term is fixed to 10. The batch size for the policy gradient methods is set to 5,000 samples, except for the SwimmerGather task, where it is set to 50,000. The replay pool has a fixed size of 100,000 samples, with a minimum size of 500 samples for all but the SwimmerGather task. In this latter case, the replay pool has a size of 1,000,000 samples. The

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

dynamics BNN is updated each epoch, using 500 iterations of Adam [4], with a batch size of 10, except for the SwimmerGather task, in which 5,000 iterations are used. The Adam learning rate is set to 0.0001 while the batches are drawn randomly with replacement from the replay pool. In the second-order KL divergence update step, λ is set to 0.01. The BNN prior weight distribution is a fully factorized Gaussian with μ sampled from a different Gaussian distribution $\mathcal{N}(\mathbf{0}, I)$, while ρ is fixed to $\log(1 + e^{0.5})$.

The classic tasks make use of a neural network policy with one layer of 32 tanh units, while the locomotion tasks make use of a two-layer neural network of 64 and 32 tanh units. The outputs are modeled by a fully factorized Gaussian distribution $\mathcal{N}(\mu, \sigma^2 I)$, in which μ is modeled as the network output, while σ is a parameter. The classic tasks make use of a neural network baseline with one layer of 32 ReLU units, while the locomotion tasks make use linear baseline function.

All tasks are implemented as described in [5]. The tasks have the following state and action dimensions: CartPole, $\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$; CartPoleSwingup, $\mathcal{S} \subseteq \mathbb{R}^4$, $\mathcal{A} \subseteq \mathbb{R}^1$; DoublePendulum, $\mathcal{S} \subseteq \mathbb{R}^6$, $\mathcal{A} \subseteq \mathbb{R}^1$; MountainCar $\mathcal{S} \subseteq \mathbb{R}^3$, $\mathcal{A} \subseteq \mathbb{R}^1$; locomotion tasks HalfCheetah, $\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$; and Walker2D, $\mathcal{S} \subseteq \mathbb{R}^{20}$, $\mathcal{A} \subseteq \mathbb{R}^6$; and hierarchical task SwimmerGather, $\mathcal{S} \subseteq \mathbb{R}^{33}$, $\mathcal{A} \subseteq \mathbb{R}^2$. The time horizon is set to $T = 500$ for all tasks. For the sparse reward experiments, the tasks have been modified as follows. In MountainCar, the agent receives a reward of $+1$ when the goal state is reached, namely escaping the valley from the right side. In CartPoleSwingup, the agent receives a reward of $+1$ when $\cos(\beta) > 0.8$, with β the pole angle. Therefore, the agent has to figure out how to swing up the pole in the absence of any initial external rewards. In HalfCheetah, the agent receives a reward of $+1$ when $x_{\text{body}} > 5$. As such, it has to figure out how to move forward without any initial external reward.

References

[1] D. Barber and C. M. Bishop, "Ensemble learning in Bayesian neural networks," *NATO ASI Series. Series F: Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.

[2] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *ICML*, 2015.

[4] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[5] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continous control," in *ICML*, 2016.

2