Active learning using Hamiltonian Monte Carlo

Ward Janssens, Ádám Arany and Rosa Friesacher

KU Leuven, ESAT, STADIUS, Leuven, Belgium

1 Introduction

In certain situations, obtaining new samples for analysis is costly or hard to do, think, for instance, of drilling for oil deposits, drug discovery, or plant breeding. Active learning addresses this challenge by selecting new candidates that are expected to be the most informative, and consequently, improving the machine learning models efficiently. There are many ways to select these new candidates in an informed way. A common strategy leverages Bayesian uncertainty estimation of the posterior predictive distribution (PPD), selecting samples with the largest posterior variance, which correspond to regions of greatest model uncertainty [5].

This form of active learning lies in accurately estimating the PPD. Within the Bayesian framework, multiple approaches have been proposed to approximate this distribution, but modern deep learning models, such as Bayesian neural networks (BNN), pose difficulties due to their large number of parameters. Hamiltonian Monte Carlo (HMC), which was developed in a physics context, is well-suited to estimate the PPD. HMC uses a physics-based approach to take efficient steps in the posterior space. By making the negative log-posterior distribution proportional to the potential energy and introducing an additional conjugate momentum variable for all parameters, which is resampled for each step. In this work, we investigate the suitability of HMC for uncertainty estimation in an active learning context [1][4].

2 Problem Setup

To evaluate our hypothesis, we conduct a simulation study on a regression problem. Training, validation, and test data are generated from an exponentially decaying sine wave, $f(x)$, within the interval $[0, 1]$, supplemented with frequency-based features to enhance model expressiveness and improve model performance. We then train three models, a classical feedforward neural network, a Bayesian Neural Network (BNN) and a Monte Carlo dropout (MC dropout) model. The latter estimates the target function by generating stochastic forward passes by randomly deactivating neurons and averaging the resulting predictions. For the BNN, posterior inference is performed using HMC. To improve the performance of the MC dropout and classical models, several restarts are performed, with the best performing being reported [3, 6].

For the BNN, new samples x_{new} are selected by estimating the posterior predictive variance at multiple x inputs and querying the point with the largest variance. This same acquisition strategy is applied to the MC dropout model by using the variance across stochastic forward passes. In contrast, the classical feedforward model selects new candidates randomly from the input range and is thus considered the (random) baseline. Note that each model starts with the same training data, but newly acquired samples are added only to the training data of the corresponding model. The architecture of the networks, as well as the dropout rate specifically for the MC dropout model, was tuned on the validation set and updated every five queried samples. For simplicity, the number of hidden layers was fixed at one for all models [2].

3 Results

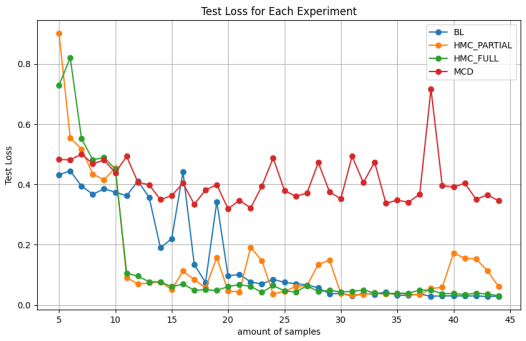

Figure 1: Learning curves using the MSE loss for the baseline, MC dropout, Bayesian neural network with architecture tuned via either a feed-forward neural network or a Bayesian neural network also estimated via HMC.

Figure 1 shows the mean squared error (MSE) of each model as new samples are added during the simulation. The baseline (BL) and MC dropout (MCD) serve as reference methods, while HMC(full) and HMC(partial) both employ HMC for Bayesian neural networks, differing only in how the network architecture is tuned. In HMC(full), the architecture is optimised using the BNN itself, whereas HMC(partial) adopts the architecture obtained from a classical feedforward model. For all this was done after five new samples were added, except for HMC(full), which was done after ten samples. Lowering the amount of architecture tuning needed for HMC(full) has the advantage of reducing the computational cost of the experiment, which is very high when training many BNNs.

The results demonstrate that the HMC-based BNN achieves the lower MSE faster than the random baseline, converging more quickly toward the true underlying signal $f(x)$ and thus beating the baseline. The two observed MSE levels around a loss of 0.4 and 0.05 correspond respectively to approximating the flat mean of the sine wave (mainly for the MCD) or to successfully capturing $f(x)$. This last level corresponds to reaching the true, asymptotic minimum. The performance gap between HMC(full) and HMC(partial) is small, with HMC(full) performing slightly better due to consistent hyperparameter tuning under

the Bayesian setting, with a downside being a larger computational cost. The fluctuations visible in the learning curves are largely attributable to the prevalence of local minima, a well-known problem in training neural networks. By contrast, the MC Dropout estimator performs poorly and doesn't seem to learn $f(x)$. Further experiments showed that for a larger number of samples in the training data, the MC Dropout model eventually converges. This can be partially alleviated by increasing the number of restarts.

References

[1] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2018.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1050–1059. JMLR: W&CP, 2016.

[4] Radford M. Neal. Probabilistic inference using markov chain monte carlo methods. *Technical Report*, 2004. Available online at https://www.cs.toronto.edu/ radford/reviewMCMC.pdf.

[5] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 1 2009.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. Submitted November 2013; Published June 2014.