Bayesian Machine Learning - Project : Challenges in Markov chain Monte Carlo for Bayesian neural networks

Ben Kabongo
Ecole Normale Supérieure Paris-Saclay
`ben.kabongo_buzangu@ens-paris-saclay.fr`

Hugo Queniat
Télécom Paris, Institut Polytechnique de Paris
`hugo.queniat@telecom-paris.fr`

Thibault Robine
Télécom SudParis, Institut Polytechnique de Paris
`thibault.robine@telecom-sudparis.eu`

Abstract

This study is an extensive review and critic of the 2022 paper **Challenges in Monte Carlo Markov Chain for Bayesian Neural Networks** written by Theodore Papamarkou, Jacob Hinkle, M. Todd Young and David Womble [15]. This paper explores the difficulties of using MCMC methods to sample parameters posteriors for neural networks. We will first highlight the obstacles faced by MCMC sampling for such models, before applying them ourselves for MLPs and compare the results obtained for different MCMC methods. We created our own implementation to reproduce the paper's experiments and run our own. The code is available here: `https://github.com/hugo-qea/BayesianML-Project`.

1 Challenges in MCMC sampling

Computational coast. Monte Carlo Markov Chain methods suffer from the curse of dimensionality. The computational complexity increases dramatically as the parameters space dimension grow or the number of data points. As a result, in most cases, other methods such as Variational Inference are preferred.

Weight symmetries. Furthermore, some intrinsic characteristics of neural networks make this specific type of non-linear hierarchical model rather unfriendly to MCMC methods. Mainly, it is known that the output of the neural networks are invariant under a certain number of common transformations [2]. Hence, the neural network parameters are non-identifiable. Even though we could restrain ourselves to a certain subset of $\mathbb{R}^n$ to avoid such complications, in practice, the parameter space is set to be the whole $\mathbb{R}^n$. This specific property of neural networks parameters become then a major drawback for MCMC-based sampling. The posterior will showcase several equally likely modes and local optima. This is a really damaging assumption for our sampling methods since first, computational time will be wasted while trying to explore equivalent modes of the parameter posterior and, secondly, because the density will be multi-modal which is one of the biggest challenges for MCMC sampling.As a result, the algorithms will most likely suffer from low acceptance rates and entrapment in local modes.

Prior specification. An other question while building this Bayesian model, is to introduce a parameter prior. In the context of neural network, the relationship between input and output is usually unclear and the interpretation of weights and biases in neural network remains an open question. Hence, the typical choice would be to choose a parameter prior which reflects this lack of a priori knowledge. In this specific case, it is common, in a Bayesian setting, to opt for an improper prior, such as an hypothetical uniform density over the whole space. However, it has been shown [12] that an improper prior can lead to an improper posterior. Following [12] predicament, it is usually preferred to select a flat and truncated density to reflect this lack of knowledge.

Convergence. Convergence in fixed MCMC sampling time is challenging for small-scale neural networks fitted to non-linearly separable datasets. Assessing whether a finite sample from an MCMC algorithm represents the target density accurately is inherently uncertain. Diagnostics designed for asymptotically exact MCMC may fail to identify convergence issues accurately. Thus, combinations of diagnostics are employed to reduce the risk of false diagnosis. Recent research [20, 3, 4] has focused on approximate MCMC methods, including minibatch MCMC and alternative techniques without minibatching. Quantization and discrepancy are key concepts in these methods. Quantization approximates the target density using an empirical measure [9], while discrepancy measures how well this empirical measure approximates the target [3]. Techniques like kernel Stein discrepancy (KSD) [3] and maximum mean discrepancy (MMD) [10] quantify discrepancy. Rudolf and Schweizer [20] propose using Wasserstein distance to assess the quality of approximation in approximate MCMC.

2 Inferential framework overview

2.1 Multilayer perceptron model

A multilayer perceptron is a neural network whose layers are completely connected. Let $\rho \geq 2$ be the number of layers. Let $j \in \{0, \cdots, \rho\}$, the number of a model layer; $j = 0$ corresponds to the model input dimension and $j = \rho$ corresponds to the model output dimension. Let k_j be the number of neurons in layer j. We'll therefore note that $k_{0:p} = (k_0, \cdots, k_\rho)$ and in the following each model will be denoted MLP($k_{0:\rho}$).

W_j and b_j represent respectively the weights and biases of the j layer of a model. $W_{j,k,l}$ refers to the (k-l)-th element of the W_j matrix and $b_{j,k}$ the k-th element of the b_j vector. The weights and biases of the j layer will be collected using the notation $\theta_j = (W_j, b_j)$ and $\theta = (\theta_1, \cdots, \theta_p)$ will represent all the weights and biases of the model for all its layers. ϕ_j will denote the activation function of the j layer.

Let x_i be a data point, and two functions are associated with each layer: g_j and h_j, such that:

$$\begin{aligned} g_j(x_i, \theta_{1:j}) &= W_j h_{j-1}(x_i, \theta_{1:j-1}) + b_j \\ h_j(x_i, \theta_{1:j}) &= \phi_j(g_j(x_i, \theta_{1:j})) \end{aligned}$$

The final output of a multilayer perceptron can thus be denoted $h_\rho(x_i, \theta)$.

2.2 Likelihood for classification

Let's take a multilayer perceptron MLP($k_{0:\rho}$) for binary ($k_\rho = 1$) or multiclass ($k_\rho \geq 2$) classification task and $D_{1:s} = \{(x_i, y_i)_{i=1,\cdots,s}\}$ is the training dataset.

In the case of binary classification, the activation function at the output of the neural network is the sigmoid function and in the case of multiclass classification it is the softmax function. These functions convert the network output into value that can be interpreted as probabilities.

$$\phi_\rho(g_\rho) = \begin{cases} \sigma(g_\rho) = \frac{1}{1+e^{-g_\rho}} & \text{binary} \\ \text{softmax}(g_\rho) = \frac{e^{g_\rho}}{\sum_{k=1}^{k_\rho} e^{g_k}} & \text{multiclass} \end{cases}$$

In the binary case, we assume $\forall i, P(y_i = 1|x_i, \theta) = h_\rho(x_i, \theta)$, and in the multiclass case we assume $\forall i, P(y_i = k|x_i, \theta) = h_{\rho,k}(x_i, \theta)$; h_ρ being the last layer of the MLP. Thus, by assuming that the data is independent and identically distributed random variables, the likelihood of the multiclass case is given by:

$$L(y_{1:s}|x_{1:s}, \theta) = \prod_{i=1}^{s} \prod_{k=1}^{k_p} h_{\rho,k}(x_i, \theta)_k^{1_{y_i=k}}$$

2.3 MCMC sampling for parameter estimation

We want to sample the θ parameters with MCMCs to solve this classification task. Let $\pi(\theta)$ denote the prior distribution of the parameters and $L(y_{1:s}|x_{1:s}, \theta)$ the likelihood knowing the parameters. The posterior distribution of the parameters θ is given by:

$$P(\theta|x_{1:s}, y_{1:s}) \propto L(y_{1:s}|x_{1:s}, \theta)\pi(\theta)$$

In the original paper 3 MCMC methods are considered, namely: Metropolis-Hastings algorithm (MH) [13, 11], Hamiltonian Monte Carlo (HMC) [14] and Power Posterior Sampling (PP) [6].

For diagnostics on the convergence of MCMC methods, we use PSRF (Potential Scale Reduction Factor) [7, 1] and ESS (Effective Sample Size) [21, 8]. PSRF and its variants can fail to diagnose poor mixing of a Markov chain, whereas low values of ESS are an indicator of poor mixing. It is thus recommended to check both PSRF and ESS [22].

2.4 Bayesian marginalization for prediction

Posterior predictive distribution. Let (x, y) be a test example. The predictive distribution of this example is given by:

$$P(y|x, D_{1:s}) = \int P(y|x, \theta)P(\theta|D_{1:s})d\theta$$

with $P(y|x, \theta)$ the likelihood and $P(\theta|D_{1:s})$ the posterior distribution of the parameters θ.

Monte Carlo Approximation. $P(y|x, D_{1:s})$ can be approximated using a Monte Carlo approximation:

$$P(y|x, D_{1:s}) = \mathbb{E}_{\theta|D_{1:s}}\big[P(y|x, \theta)\big] = \sum_{k=1}^{v} P(y|x, \omega_k)$$

$\omega_{1:v}$ is the markov chain realization obtained from $P(\theta|D_{1:s})$.

Classification rule. For prediction, we apply the following rule:

- *Binary classification*: $\hat{y} = 1 \quad \text{if} \quad P(y = 1|x, D_{1:s}) \geq 0.5 \quad \text{else} \quad 0$
- *Multiclass classification*: $\hat{y} = \text{argmax}_y\{p(y|x, D_{1:s})\}$

3 Experiments

To begin with, we chose to carry out our experiments on the same foundations chosen by the authors. Hence, we tested our implementation a selection of their datasets, with the same MLP architectures shown below in table 1 and the same parameter prior : $\pi(\theta) \sim \mathcal{N}(0, 10I)$.

Table 1: Dataset sample sizes and model configurations.

Dataset	Sample size		Model	$\lvert\theta\rvert$
	Training	Test		
Noisy XOR	500	120	$\text{MLP}(2, 2, 1)$	9
Pima	262	130	$\text{MLP}(8, 2, 2, 1)$	27
Penguins	223	110	$\text{MLP}(6, 2, 2, 3)$	29
Iris	120	30	$\text{MLP}(4, 2, 2, 3)$	22

Datasets. The experiments are carried out on 4 different data set: the Noisy XOR, and 3 real datasets, Pima, Penguins and Iris. All datasets are classification task, and the input dimension varies between 2 and 8. The XOR dataset consists of four exact data points defining the XOR function and is expanded by simulating a larger dataset with noisy versions of XOR. The Pima dataset comprises observations from female patients of Pima Indian heritage, with a binary output variable indicating the presence or absence of diabetes. The penguin dataset includes body measurements for three species of penguins observed on three islands in the Palmer Archipelago, Antarctica. In our experiments, we added the Iris dataset. The Iris dataset is a classic dataset in machine learning and consists of observations for three species of iris flowers. The size of the training and testing set as well as the models associated with each dataset are described in table 1.

MCMC samplers. The MCMC litterature is really diverse and comports many distinct methods of sampling. Hence, while the original paper focused on Random Walk Metropolis Hastings [11], Hamiltonian Monte Carlo [5, 14] and Parallel Tempering [6], we chose to keep MH and HMC and to add another traditional sampler, Metropolis Adjusted Langevin Algorithm [19].

Table 2: Results of the experiments. We report for each dataset, the accuracy obtained by performing an SGD on the associated model, the accuracy obtained with the prior, and for each MCMC method considered (MH, HMC and MALA), we report the accuracy, the PSRF and the ESS.

Dataset	Accuracy		MCMC			
	SGD	Prior	Sampler	ACC	PSRF	ESS
XOR	89.00	51.67	MH	85.00	10.01	240047.67
			HMC	68.33	05.04	304402.21
			MALA	77.00	03.28	307765.59
Pima	71.00	66.88	MH	71.97	29.28	747596.30
			HMC	69.50	13.04	566984.81
			MALA	67.52	09.15	627153.94
Penguins	69.00	37.31	MH	71.64	200.07	768408.32
			HMC	26.87	010.63	565941.17
			MALA	26.87	007.39	688418.42
Iris	97.00	30.00	MH	100.00	05.90	177906.83
			HMC	63.00	10.08	521050.67
			MALA	63.00	06.60	700913.17

The results of our experiments are presented in table 2.

Performance of SGD versus MCMC methods. SGD (Stochastic Gradient Descent) is used as a performance benchmark for each dataset. In some cases, MCMC methods appear to underperform SGD. This suggests that the MCMC approach may require more work to compete with SGD on these specific datasets.

Impact of MCMC methods. On some datasets, such as Pima and Iris, performance varies considerably between these methods. This indicates that the choice of MCMC method can have a significant impact on performance.

Matching MCMC methods to data. The performance of MCMC methods also varies according to dataset. For example, on the Penguins dataset, all MCMC methods appear to fail with an accuracy of around 26.87%. This suggests that these methods may not be well suited to this type of data, or that they require more fine-tuning of the hyperparameters.

Sample quality assessment. In addition to accuracy, measures such as PSRF and ESS are used to assess the quality of samples generated by MCMC methods. Low PSRF and high ESS values are generally desirable, indicating good convergence and efficient exploration of the parameter space.

The results highlight the importance of carefully choosing the MCMC method and adapting it to the specificities of the data sets, as well as the importance of assessing the quality of the samples generated to guarantee the reliability of the results.

4 Discussion and conclusion

Bayesian neural network are usually specified as followed for regression task: $y = h_\rho(x_i, \theta) + e$, e being a noise, typically Gaussian noise. This assumption is equivalent to assuming the conditional law of y: $P(y < t|x, \theta) = h_\rho(x, \theta) + P(e < t)$. In the paper, the model specification is not clearly stated and justified. It is to understand that the assumption $P(y = 1|x, \theta) = h_\rho(x, \theta)$ defined the model, and induced a specific optimal law for the parameter θ, that we want to sample. In particular, no noise is added to the model specification, as in the regression case. This lack of precision and justification in this assumption is a weak point.

As we have implemented and run the MCMC algorithm, we have realised that the choice of hyper parameters is very fine, we may easily end up with an acceptance rate of 0.00. A discussion on the difficulties of choosing this hyper parameters are absent of this paper, and would have been welcomed. Indeed, to ensure higher acceptance rates, closer to the theoretical optimas for each methods, such as $a = 0.234$ for MH [16] or $a = 0.574$ for MALA [17], we tried to implement Adaptive variants of our samplers, as described in [18]. However, if asymptotic step-sizes are announced in papers, in practice the multi-modality and the existence of local optima for the density allowed for very low mixing and were source of issues. In order to achieve reliable acceptance rates, we had to consider smaller moves, as most moves would be rejected with a very low Metropolis-Hastings ratio. The plots of the chains show this inefficiency in trying to move to higher probable modes of the density and so does the diagnostics with very noticeably high PSRF values.

Another unaddressed issue we faced was the low efficiency of gradient target density based method. Indeed, methods such as HMC or MALA that rely heavily on $\nabla \log \pi$ with π the target density, showcased lower accuracy in comparison to the very classical Random Walk Metropolis Hastings that does not use the gradient of the density. For instance, we were able to use a classical scheduling for the proposal σ in MH while the Adaptive MALA was clearly struggling to find an adequate step size. In fact, we see down below in figure 2 that the beginning of the chain remains equal to the initialization for a good 15 to 20 thousands samples for this example with the Noisy XOR. On the other hand, MH in figure 1 is able to explore the space and converge to a set of optimal parameters. Hence, we conjecture that gradient computation may lead to numerical instabilities due to very small values as the parameter posterior is probably too flat.

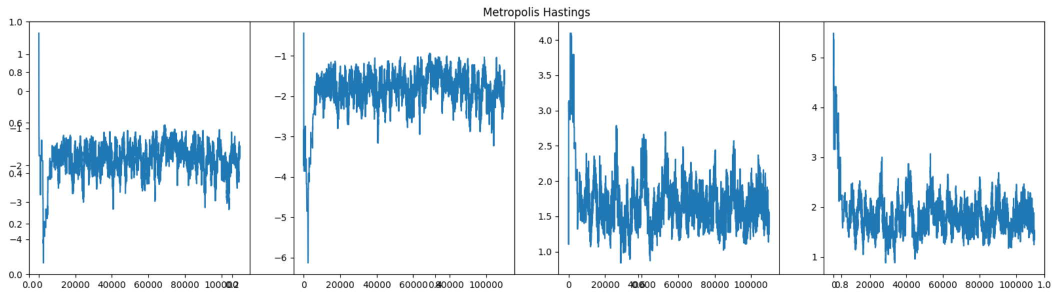

Figure 1: Behaviour of the MC for Random Walk Metropolis Hastings, $a = 0.21$

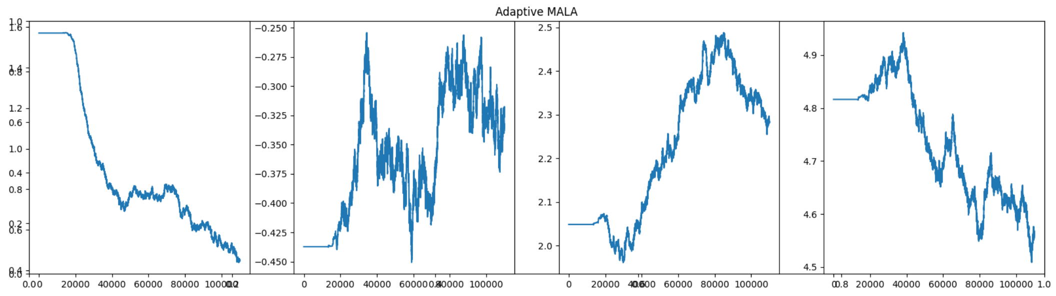

Figure 2: Behaviour of the MC for Metropolis Adjusted Langevin Algorithm, $a = 0.464$

Finally, further than computational, numerical and convergence difficulties, MCMC sampling suffers also from large computing times. Indeed, as our computations and tests have shown us, the computing times, especially for more advanced sampling methods, are rather large and it confirms the observations from the paper as sampling 100000 samples would take us between 5 to 20 minutes depending on the method. As a result, we wonder if, at this time, knowing MCMC's finesse for its parameters and its inability to scale to larger dimension, Bayesian inference via MCMC for neural networks should be used. We have clearly seen its limits when we rose the number of parameters. Hence, the original paper and our observations lead us to believe that we should limit its use to smaller models.

References

[1] Steve P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7:434–455, 1998.

[2] Anthony M. Chen, Haiping Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Computation*, 5:910–927, 1993.

[3] Wenbo Y. Chen, Alessandro Barp, François-Xavier Briol, Jackson Gorham, Mark Girolami, Lester Mackey, and Chris Oates. Stein point markov chain monte carlo. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 1011–1021, 2019.

[4] Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2606–2615, 2016.

[5] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195, 1987.

[6] N. Friel and A. N. Pettitt. Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 70, 2008.

[7] Andrew Gelman and Donald B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7:457–472, 1992.

[8] Lan Gong and James M. Flegal. A practical sequential stopping rule for high-dimensional markov chain monte carlo. *Journal of Computational and Graphical Statistics*, 25:684–700, 2016.

[9] Siegfried Graf and Harald Luschgy. *Foundations of Quantization for Probability Distributions*. Springer, 2007.

[10] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

[11] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57, 1970.

[12] Herbert K H Lee. Neural networks and default priors. *Journal of the Royal Statistical Society*, 2007.

[13] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[14] Radford M. Neal. *MCMC using hamiltonian dynamics*. 2011.

[15] Theodore Papamarkou, Jacob Hinkle, M. Todd Young, and David Womble. Challenges in markov chain monte carlo for bayesian neural networks. *Statistical Science*, 37, 2022.

[16] Gareth O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *Annals of Applied Probability*, 7, 1997.

[17] Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical Science*, 16, 2001.

[18] Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18, 2009.

[19] P. J. Rossky, J. D. Doll, and H. L. Friedman. Brownian dynamics as smart monte carlo simulation. *The Journal of Chemical Physics*, 69, 1978.

[20] Daniel Rudolf and Nikolaus Schweizer. Perturbation theory for markov chains via wasserstein distance. *Bernoulli*, 24(5):2610–2639, 2018. 17.

[21] Dootika Vats and James M. Flegal. Lugsail lag windows and their application to mcmc. *arXiv preprint arXiv:1803.06376*, 2018.

[22] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Burkner. Rank-normalization, folding, and localization: an improved r for assessing convergence of mcmc. *arXiv preprint arXiv:1903.08008*, 2019.