How well does your sampler really work?

Ryan Turner*
MILA, Université de Montréal
turnerry@iro.umontreal.ca

Brady Neal*
MILA, Université de Montréal
bradyneal11@gmail.com

Markov chain Monte Carlo (MCMC) methods have seen a huge increase in use over the last few decades. The goal in MCMC methods is to sample from a complex probability distribution $p^\star$ given access only to its unnormalized density $\tilde{p}$, where $\tilde{p}$ is typically a posterior distribution. These posteriors are intractable to normalize and sample from in complex models such as deep networks; in deep networks, posterior integration has the potential to alleviate problems like adversarial examples [4].

Each machine learning conference contains a publication proposing a new variation on MCMC methods. The community lacks a method to determine if these new methods actually sample from posteriors found in real problems with improved accuracy over existing samplers. New methods are benchmarked either 1) via hand-crafted toy problems (where a ground-truth is known) or 2) via test set performance on real problems. The issue with hand-crafted examples is obvious: Performance on these problems may have little relation to performance on real problems. Benchmarking via test set performance on real problems is laudable. However, it confounds the specification of the model and priors with the performance of the sampler. In a misspecified model, it is possible that a sampler stuck in an unrepresentative part of the posterior could actually have higher test set performance.

Whether current samplers are providing samples from anything close to the true posterior on difficult problems is of critical importance for determining future research directions. Are samplers with higher test set performance actually sampling from real posteriors more faithfully? Can we sample with any fidelity from complex high dimensional distributions? Is that merely a "fool's errand"? The answers to these questions will determine if it is a worthwhile endeavor to continue to hone MCMC methods for application in successful modern models such as deep neural nets.

We propose a new data-driven approach to create a benchmark that estimates how well various MCMC procedures work on real problems. Arguably, algorithms in machine learning and statistics rely on the "workhorses" of either optimization or sampling methods. The world of (non-convex) optimization has already tackled this challenge with the COCO benchmark [6], which contains a test battery of difficult optimization problems. Our approach is an analogous system for sampling methods. However, we aim to further improve upon this using flexible (including neural net based) benchmark examples that have been trained to match posteriors found in practice.

The notion of a black box is highly relevant to conceptually understanding this work. Fundamentally, an MCMC sampler is a system that takes a black box that computes an unnormalized density $\tilde{p} \propto p^\star$ (and possibly $\nabla \log \tilde{p}$) and a previous sample $\mathbf{x}_{t-1} \in \mathbb{R}^D$ in the Markov chain; it outputs another sample $\mathbf{x}_t \in \mathbb{R}^D$. Once the Markov chain has converged, these samples are theoretically guaranteed to marginally come from the density $p^\star$, albeit with temporal correlation. If the previous sample was drawn exactly, $\mathbf{x}_{t-1} \sim p^\star$, then $\mathbf{x}_t \sim p^\star$ exactly as well; this is a result of *detailed balance*.

By analogy, optimization algorithms take an objective function $f \in \mathbb{R}^D \to \mathbb{R}$ (and ∇f) as a black box and produce points $\mathbf{x}_t \in \mathbb{R}^D$ that successively minimize f as much as possible. Just as COCO provides its objectives f as a black box to the optimizers and keeps hidden the true optimum, our benchmark provides the unnormalized density $\tilde{p}$ as a black box to the samplers. Our benchmark keeps hidden the parameterization of $\tilde{p}$ needed to efficiently take iid samples from $p^\star$.

Contribution We build a novel benchmark to test various samplers on realistic problems. This includes designing sensible metrics to score samplers across problems. Our system will serve as a practical tool in research like MLcomp/COCO. We also explore the validity of MCMC diagnostics.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

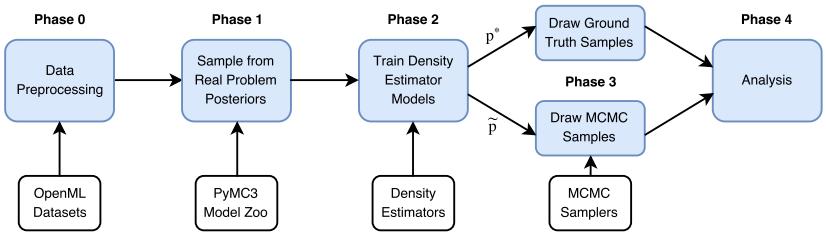

Figure 1: Flowchart showing the five phases of methodology. Phases 0–2 are for creating benchmark examples and are not re-run when new samplers are tested. Phase 2 includes mixture models, RNADE, and Real NVP [3].

Related work The closest existing system is SamplerCompare of Thompson [12], which tests samplers on a handful of hand-crafted stress-tests such as Neal's funnel. However, SamplerCompare is more an R package to aid sampler evaluation than a full benchmark. Recent work from systems biology [1] compares various samplers for dynamical systems (filtering) on a set of hand-crafted ODE systems inspired by biological applications, and is not very relevant for machine learning.

1 Methodology

In our approach we use a large "data set of data sets" and a diverse "model zoo" to create a representative set of examples. Long MCMC chains are drawn (using NUTS [7]) from each of these posteriors. Flexible unsupervised models that serve as a *ground-truth* in the benchmarking phase are fit to the chains to construct the *benchmark example distribution* (BED), which include deep models like RNADE [13]. Once trained, these BEDs are functionally equivalent to hand-crafted examples such as the toy posterior distributions usually used to benchmark samplers (or such as those in COCO). However, these examples are not hand-crafted but, rather, are much more representative of real problems. Because it is possible to draw exact (iid) samples from the BEDs, we now have a ground-truth set of samples to validate the accuracy of the sampling methods, and assess MCMC diagnostics. In particular, we look at the effective sample size (ESS) because it provides a concrete statement on the quality of an MCMC chain [8]. Our approach is a form of *meta-learning* [14].

Our benchmark system follows a five phase approach, with a graphical summary in Figure 1.

In phase 0, we create a "corpus" of data sets that we refer to as a "data set of data sets." This is meant to create a realistic sample of problems that a practitioner may encounter "in the wild." Such an approach was also taken in the AutoML competition [5] and the automated statistician project [9].

In phase 1, we use a model zoo to simulate a variety of (Bayesian) models that a practitioner might attempt to apply to a real problem. There are models for regression and classification. Each model/data set pair results in a posterior over a parameter space, which varies in dimensionality depending on the problem. Except in very simple cases (e.g., linear regression), we are not able to obtain samples from these posteriors exactly. We use NUTS, the default sampler in PyMC3 [11] and Stan [2], because it is considered to be a good off-the-shelf sampler. By running multiple long chains of NUTS on the posteriors, we obtain a sufficient approximation and representation for phase 2.

In phase 2, we run various density estimation models (e.g., RNADE) to generate BEDs on the Markov chains from phase 1. We run a separate training procedure on each model/data set pair. These BEDs serve as surrogates for the real posteriors found in phase 1. We do not aim to replicate the posteriors from phase 1 exactly, but generate examples that are *qualitatively similar* to the real posteriors in phase 1. This gives us examples that are more realistic than the usual hand-crafted toy problems. Nonetheless, we train multiple models and take the one with the highest held-out likelihood on the last 20% of the Markov chain from phase 1. Model checking diagnostics can also be used to verify the similarity of the BEDs (surrogates) and their corresponding Markov chains from the real posteriors.

When selecting models for use as BEDs in phase 2, we have the following requirements: 1) The models are flexible enough to closely fit the posteriors found in phase 1. 2) They serve as a black box, providing an unnormalized density $\tilde{p}$ (and $\nabla\tilde{p}$) for an arbitrary point $\mathbf{x}$. 3) We can efficiently sample (ground-truth) from them given their parameters (which are hidden from the samplers).

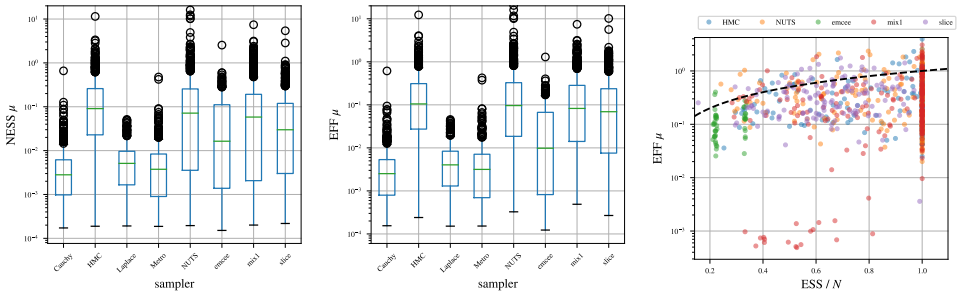

Figure 2: Performance summaries: The box plots demonstrate the distribution on NESS (left) and efficiency (center) conditional on the sampler achieving an RESS of at least 12 to only show the mode where the samplers don't completely fail. We also show a calibration plot to assess if ESS is a good predictor of efficiency with the diagonal in dashed black. The ESS diagnostic appears to have an optimistic bias.

In phase 3, we benchmark the samplers. If someone invents and provides a new sampling algorithm, it is added in phase 3. Phases 0–2 remain fixed as new samplers are submitted to be benchmarked. Each sampler is run on each of the BEDs for multiple chains for a fixed period of time; meaning, N is different for each chain. The raw samples from these Markov chains form the output of phase 3.

In phase 4, we take a large number (e.g., $\sim 10^5$) of exact iid samples from the BEDs as a ground-truth. The square loss between unbiased point estimates $\hat{\theta}$ (e.g., $\hat{\mu}_d$ or $\hat{\sigma}_d^2$) taken from the Markov chains from phase 3 and the point estimates from the exact chains are aggregated. We also compute and store the MCMC diagnostics for each chain.

We then aggregate the performance results by looking at the real effective sample size (RESS) as derived from the square errors in point estimation $\hat{\theta}$. In analogy to the justifications for the ESS diagnostic we define: RESS $:= R/\text{mean sq. error} = RK/\sum_{k=1}^{K}(\hat{\theta}_k - \theta)^2$, where K is the number of MCMC chains and $R = N \operatorname{Var}_{p^\star}[\hat{\theta}]$ is a constant to make RESS comparable across different estimators $\hat{\theta}$. When looking at the distribution of sampler performance across examples it is more appropriate to use normalized ESS: NESS $:=$ RESS$/\operatorname{median} N$, where the median is taken across different samplers on the same example. We also define *efficiency* (EFF) as RESS$/N$.

2 Results

The box plots in Figure 2 provide a sense of the variation. The NESS of the samplers is generally bimodal: either the samplers achieve $>1\%$ efficiency or they completely fail with an RESS < 1. To illustrate a single mode, in Figure 2 we show the box plots after excluding the "failed" chains. Inspired by the rule of $N = 12$ from MacKay [10], we use an RESS of 12 to threshold failure-vs-success. Emcee is the most bimodal: while it sometimes has a high NESS competitive with advanced methods, it has the lowest success probability (49% vs. 78% for NUTS). Emcee also has the lowest efficiency of any method except random walk Metropolis, but has the highest per sample speed.

Other results shown include: NUTS and HMC are the highest performers, despite their higher per sample cost. Slice sampling also makes a "strong showing", being more competitive in the lower dimensional cases. Random walk metropolis methods generally have an efficiency in the 0.1% to 1% range while slice sampling and HMC based methods have efficiencies in the ball park of 2% to 40%, with NUTS showing the highest performance. Emcee seems to vary widely. Using a compound proposal "mix" (alternating NUTS and random walk Metropolis) does not substantially increase NESS when methods succeed. However, mix increases the chance of success: to 83% from 78% for NUTS when $\hat{\theta} = \hat{\mu}$, to 91% from 87% for KS distance, but only to 72% from 71% when $\hat{\theta} = \hat{\sigma}^2$.

3 Conclusions

We have presented a novel data-driven system to benchmark the real performance of MCMC samplers on realistic problems. This benchmark is intended to become a general service that will become as widespread as COCO or MLcomp, and evolve with time by including more models in phases 1 and 2.

References

[1] B. Ballnus, S. Hug, K. Hatz, L. Görlitz, J. Hasenauer, and F. J. Theis. Comprehensive benchmarking of Markov chain Monte Carlo methods for dynamical systems. *BMC Systems Biology*, 11(1):63, 2017.

[2] B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 20:1–37, 2016.

[3] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv:1605.08803*, 2016.

[4] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv:1703.00410*, 2017.

[5] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macia, B. Ray, M. Saeed, A. Statnikov, et al. Design of the 2015 ChaLearn AutoML challenge. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[6] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *arXiv:1603.08785*, 2016.

[7] M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[8] R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal. Markov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100, 1998.

[9] J. R. Lloyd, D. K. Duvenaud, R. B. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence*, pages 1242–1250, 2014.

[10] D. J. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[11] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[12] M. B. Thompson. Introduction to SamplerCompare. *Journal of Statistical Software*, 43(12): 1–10, 2011.

[13] B. Uria, I. Murray, and H. Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183, 2013.

[14] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.