École des Ponts ParisTech
Department of Statistics, University of Oxford

2017
Master's Internship Report

Émile Mathieu
Élève ingénieur, Third year

Bayesian Nonparametric Inference within Probabilistic Programming Languages

Internship carried out at Department of Statistics, University of Oxford
From the 22nd of May, to the 15th of September 2017.

Company tutor: Teh, Yee Whye
Training supervisor: Obozinski, Guillaume

Acknowledgments

First of all, I would like to express my indebtedness appreciation to my departmental supervisor Prof. Yee Whye Teh. His belief in me and his advices played a decisive role in making the execution of my work and thus this report.

I also express my deepest thanks to Benjamin Bloem-Reddy, who as a postdoc, oversaw me during this internship and with whom I frequently work.

Moreover, my gratitude goes to Guillaume Obozinski, my school training supervisor, whose guidance has continually shaped my career path since I have been at Ecole des Ponts ParisTech.

Abstract

Keywords : Probabilistic Programming, Bayesian Non-parametric, Bayesian Inference, Sampling methods

Résumé

Contents

Acknowledgments **iii**

Abstract **iv**

Résumé **v**

Table of contents **vii**

List of figures **viii**

Glossary **ix**

1 Introduction **1**

2 Presentation of the Department of Statistics **2**
2.1 Creation 2
2.2 Activities 3

3 Mission **4**
3.1 Themes of research 4
3.2 Context 4
3.3 Reading group 5
3.4 Organisation 5
3.4.1 Managing papers 5
3.4.2 Managing research 6
3.4.3 Managing projects 6

4 Probabilistic programming **7**
4.1 What is it ? 7
4.2 Why is it useful ? 7
4.3 History 7
4.4 Design 8
4.5 Inference schemes 8
4.5.1 MCMC 8
4.5.2 Importance Sampling 8
4.5.3 Particle MCMC 8
4.5.4 Hamiltonian 9
4.5.5 Variational Inference 9

4.6 Contributions 9

5 Bayesian nonparametric **10**
5.1 Definition 10
5.2 Usefulness 10
5.3 Canonical models 10
5.4 MCMC Inference 10
5.4.1 Marginal Samplers 10
5.4.2 Conditional Samplers 11
5.4.3 Hybrid Samplers 11
5.4.4 SMC 11

6 BNP sampling in PPL **12**
6.1 Link between BNP and High order PPL 12

7 Future Work **13**
7.1 Learning parameters in PPL 13
7.1.1 Motivation 13
7.1.2 Choice of language/library 14
7.1.3 New models 14
7.1.4 Difficulties 14
7.2 Variational Inference for Bayesian Nonparametric (BNP) in Probabilistic Programming Language (PPL) 15
7.3 Adversarial Inference for BNP in PPL 15
7.4 Piecewise Deterministic Markov Processes 15

8 Conclusion and personal review **17**
8.1 Conclusion 17
8.2 Personal review 17

Appendices **18**
A Variational Inference ? 18
B Automatic Differentiation ? 18

List of Figures

Glossary

- **BNP**: *Bayesian Non-Parametric*, explained in Section 5.1.
- **PPL**: *Probabilistic Programming Language*, explained in Section 4.1.

Introduction

cf Research Proposal ?

Presentation of the Department of Statistics

2.1 Creation

The Department of Statistics [1] is part of the University of Oxford, along with the other departments and the 38 constituent colleges.

The University of Oxford was founded in the 11th century, which makes it the oldest university in the English-speaking world and the world's second-oldest university in continuous operation.

The Department of Statistics was officially created in 1988, even though first moves in the development of Oxford statistics can be dated to the 19th century.

Indeed, In the 1870s, Florence Nightingale – the pioneer of modern nursing – discussed the possibility of endowing a Professorship of Statistics in Oxford, but the proposal eventually foundered. However, Oxford did appoint a statistician to a chair in 1891, although not to a chair in statistics.

The next significant moves in the development of Oxford statistics were by economists, who were increasingly keen to build economic theory on a foundation of sound data analysis. This led to the creation in 1935 of an Institute of Statistic, swhich was then renamed as the Institute of Economics and Statistics in 1962.

The sequence of events which led directly to the establishment of the present Department of Statistics began with the appointment in 1945 of David Finney as the universitys first Lecturer in the Design and Analysis of Scientific Experiment (LIDASE).

Then in the1980s, after the Department of Biomathematics' head increasingly felt that Oxford was losing out in the face of developments in statistics, a working party appointed by the general board of the university to assess a careful analysis of the organisation of statistics in Oxford. They found fragmentation to be the dominant feature of Oxford statistics and concluded that fragmentation has serious disadvantages ..... The working

[1] https://www.stats.ox.ac.uk

partys report recommended the creation of a university statistics department, which were to include the former Department of Biomathematics, together with a new Professorship in Statistical Science and the two existing lecturerships in statistics within the Mathematical Institute.

These major recommendations were all accepted by the university and the new Department of Statistics was created in 1988.

2.2 Activities

The Department of Statistics at Oxford is a world leader in research including computational statistics and statistical methodology, applied probability, bioinformatics and mathematical genetics. The main research groups in the Department are Computational statistics and machine learning, Probability, Statistical genetics and bioinformatics, Protein Informatics and Statistical Genetics.

I am part of the Computational Statistics and Machine Learning Group (OxCSML) [2], which have research interests spanning Statistical Machine Learning, Monte Carlo Methods and Computational Statistics, Statistical Methodology and Applied Statistics.

The department offers an undergraduate degree (BA or MMath) in Mathematics and Statistics, jointly with the Mathematical Institute. At postgraduate level there is an MSc course in Applied Statistics (MSc in Statistical Science from 2017), as well as a lively and stimulating environment for postgraduate research (DPhil or MSc by Research). The department also has a consulting activity called *Oxford University Statistical Consulting*.

[2] `http://csml.stats.ox.ac.uk/people/mathieu/`

Mission

3.1 Themes of research

Prof. Yee Whye Teh [1] has worked for a long time on inference sampling schemes for BNP mixture models [17, 16, 26, 27], but also on stick-breaking constructions [42, 15]. He also has recently been interested in PPL and consequently in inference schemes within PPL for BNP models.

This theme requires knowledge in several fields – Probabilities, Computational Statistics, Programming Languages – which makes it deeply interesting. He proposed me working with him on this topic as part of a 3-years DPhil program, and to start earlier as an intern.

3.2 Context

In addition to inviting me to work with him, Prof. Yee Whye Teh also opened two postdoctoral positions for working on the same project, which have been filled by Tom Rainforth and Benjamin Bloem-Reddy.

Tom Rainforth [2] is finishing is third year of DPhil in the Dept. of Engineering Science in Oxford, supervised by Prof. Frank Wood. His interests include probabilistic programming, Bayesian optimization, probabilistic numerics, sequential Monte Carlo and particle Markov Chain Monte Carlo methods. He will join the group in October, but he has already attended several reading group meetings.

On the other hand, Benjamin Bloem-Reddy [3] arrived in May in Oxford and has already started working on the project. He was supervised by Peter Orbanz at Columbia University, and his research were focused on probabilistic and statistical analysis of networks and other discrete data.

[1] https://www.stats.ox.ac.uk/~teh
[2] http://www.robots.ox.ac.uk/~twgr
[3] http://www.stats.ox.ac.uk/~bloemred/

3.3 Reading group

Four reading groups are organised with a bi-weekly period: Kernel methods, Deep Learning, Bayesian Nonparametrics and Probabilistic Inference. I have been leading the Probabilistic Inference reading group [4] since July. Since, Ben and I have presented four papers [38, 43, 39, 11] with an emphasise on probabilist programming.

3.4 Organisation

In this section I develop my current organisation and workflow as a researcher. At first, I had much trouble to organise my workflow, I wrote my papers' review and new ideas on flying sheets, papers were saved in my computer's folders, citations for report was time-consuming, my code was locally saved, etc...

Thus, I worked on a better workflow and after trials and errors, I eventually arrived on what I describe below. I aim to modify this process with time, so as to continually enhance my productivity and be able to focus on the interesting part of the job.

3.4 Managing papers

My biggest trouble was keeping organised the dozens of new articles I read each week. I was saving them in a tree-like structure of folders, but with the number of articles saved growing, it became more and more difficult to find specific article. Moreover, this structure inherently prohibits cross-categories articles which is annoying for a project situated at the intersections of several fields. Furthermore, I had no fast way to cite an article, neither in plain format (for markdown [5] files for instance) nor in *BibTeX* format.

Then, I have heard of papers managing library such as Papers3 [6] or Mendeley [7]. I have eventually opted for Papers3 but Mendeley is also a popular choice in the academic community. These applications features many tools easing the life of a researcher, the main one being from my point of view:

- Synchronisation: between multiple computers or devices.
- Multi-labels: these are used in the search tool.
- Local search: search in titles, authors, labels and even papers' content.
- Online search: can import articles in a fast manner by being connected with online search engines such as *arXiv*.
- Collections: create a reading list, or group of papers which can be cited at once
- Citations: get *BibTeX* reference or *BibTeX* cite command in clipboard

[4] https://github.com/BigBayes/oxsml/wiki/Probabilistic-Inference-meetings
[5] https://en.wikipedia.org/wiki/Markdown
[6] https://www.readcube.com/papers/mac/
[7] https://www.mendeley.com

3.4 Managing research

Another of my organizational issue was keeping track of ideas. I happened to find that research is a result of a long chain of ideas which were continually iterated upon. I am now maintaining a single *master document* for keeping tracks of this chain of ideas.

It has a bulleted list of all ideas, problems, and topics that Id like to think more carefully about. This list is succinct but subsequent sections go in depth on particular entries. This list is sorted according to what Id like to work on next, but I continually revise my priorities according to whether I think the direction aligns with my broader research vision, and if I think the direction is necessarily impactful for the community at large.

3.4 Managing projects

Then, when an idea has matured enough and I have seriously started working on it, I create a Github [8] repository for the project. Each project has its separated repository. It contains a `/readme.md` file maintaining a list of todos, with also questions (and sometimes answers!) both for myself and collaborators. This makes it transparent how to keep moving forward and whats blocking the work.

There is also a `/doc/` folder for all the write-ups, usually in LaTeXformat. The `etc/` folder is used for everything not relevant to other directories such as pictures of whiteboards during conversations about the project. Finally, the `/src/` folder is where all code is written. Runnable scripts are written directly in `/src/` , and classes and utilities are written in `/src/codebase/`.

[8] `https://github.com`

Probabilistic programming

4.1 What is it ?

At a high level, PPL are Programming Languages (PL) techniques to abstract inference algorithms from stats/ML such that they apply automatically and correctly to the broadest possible set of model-based reasoning applications

Probabilistic programming systems [6, 7, 9, 31] represent generative models as programs written in a specialized language that provides syntax for the definition and conditioning of random variables.

Probabilistic programs are usual functional or imperative programs with two added constructs: (1) the ability to draw values at random from distributions, and (2) the ability to condition values of variables in a program via observations. from Gordon, Henzinger, Nori, and Rajamani Probabilistic programming. In Proceedings of On The Future of Software Engineering (2014).

TEST

4.2 Why is it useful ?

Increase productivity: savings to be found in the amount of code that needs to written in order to prototype and develop models.

remove the burden of having to develop inference code for each new model: which is error-prone and time consuming This is done by providing a modeling language abstraction layer in which developers can denote their models. If done, generic inference is provided for free.

4.3 History

The first generation of PPLs had limitation in the range of models that could be represented and in which inference could be performed. BUGS [28] and STAN [9] can only work with graphical models. Similarly Factorie [32] and Infer.NET [34] only handle factor

graphs. These so-called *First-Order* Probabilistic Programming Languages (PPLS), can only represent finite dimensional model and have bounded loops.

On the other hand, *High Order* PPLS are Turing complete, allow unbounded recursion and thus can denote infinite dimensional objects. They are easy to program in, natural to express certain models, but it is hard to perform inference in these PPLs: Anglican [49], Venture [31], Church [20].

4.4 Design

`sample/assume` and `observe` statements

$\mathbf{y} := (y_j)_{j=1}^{N}$

A (almost-surely terminating) probabilistic program defines a probability distribution over finite feasible traces x with probability density $\pi(\mathbf{x}) := \gamma(\mathbf{x})/Z$ where

$$\gamma(\mathbf{x}) := \prod_{i=1}^{|\mathbf{x}|} f(x_i \mid x_{1:i-1}) \prod_{j=1}^{|\mathbf{y}|} g(y_j \mid x_{1:\tau(j)})$$

and Z is the normalizing constant $Z := \int \gamma(\mathbf{x}) d(\mathbf{x})$.

4.5 Inference schemes

Typically inference can be performed for any probabilistic program using one or more generic inference techniques provided by the system back end, such as Metropolis-Hastings [47, 31], Hamiltonian Monte Carlo [9], expectation propagation [34], and extensions of Sequential Monte Carlo [44, 37, 50] methods.

4.5 MCMC

[47] [39]

4.5 Importance Sampling

IS

SMC

4.5 Particle MCMC

breakpoints are needed. Continuation-Passing Style (CPS) in *Anglican* [49] and *WebPPL* [21] coroutine copying in [18]

PG

PMMH

IPMCMC

4.5 Hamiltonian

HMC

HMCDA

NUTS In [23], the authors address the issue of choosing the two hyperparameters of HMC: a step size ϵ and a desired number of steps L, since HMCs performance is highly sensitive on those. [36]

SGLD mini-batch / online setting, scale to bug dataset [1] [46]

SGHMC Same setting as SGLD Naive version is wrong (posterior is not the invariant distribution), see [29] friction term [10]

4.5 Variational Inference

MCMC methods can be slow to converge and their convergence can be difficult to diagnose. To my knowledge, *Edward* [43] is the only PPL handling variational inference.

4.6 Contributions

During this internship I have taken the time to actually implement several inference algorithms, by contributing to two existing PPLs. First, I implemented [2] both the Stochastic Gradient Langevin Dynamics (SGLD) and Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) inference algorithms in Turing.jl [18], a PPL based on Julia and developed at the University of Cambridge. Then, I implemented [3] the Dual Averaging extension of HMC [13] for Edward [43], a PPL built on top of Tensorflow [1] by Blei's group at Columbia University.

PMMH: [4]

PGAS / IPMCMC for Turing.jl

[1] See for instance, SGLD applied to a Bayesian logistic regression at `https://github.com/yebai/Turing.jl/blob/master/example-models/sgld-paper/lr_sgld.jl`

[2] See `https://github.com/yebai/Turing.jl/tree/master/src/samplers`

[3] See `https://github.com/blei-lab/edward/pull/728`

[4] `https://github.com/yebai/Turing.jl/pull/339`

became a *contributor* of Turing repository.

Bayesian nonparametric

5.1 Definition

5.2 Usefulness

5.3 Canonical models

??? DP, PYP, etc

5.4 MCMC Inference

Constructing MCMC schemes for models with one or more Bayesian nonparametric components is an active research area since dealing with the infinite dimensional component P forbids the direct use of standard simulation-based methods.These methods usually require a finite-dimensional representation. The general idea for designing inference schemes is to find finite dimensional representations to be able to store the model in a computer with finite capacity.

There are two main sampling approaches to facilitate simulation in the case of Bayesian nonparametric models: random truncation and marginalisation. These two schemes are known in the literature as conditional and marginal samplers.

5.4 Marginal Samplers

Marginal samplers bypass the need to represent the infinite-dimensional component by marginalising it out. These schemes have lower storage requirements than conditional samplers because they only store the induced partition, but could potentially have worse mixing properties.

5.4 Conditional Samplers

Conditional samplers replace the infinite-dimensional prior by a finite-dimensional representation chosen according to a truncation level. Since these samplers do not integrate out the infinite-dimensional component, their output provides a more comprehensive representation of the random probability measure. thinning vs stick-breaking

5.4 Hybrid Samplers

YW paper on PK ?

5.4 SMC

Review of SMC ? cf Maria Lomeli thesis

BNP sampling in PPL

Stochastic Memoization with DPmem: $\alpha = 0$, deterministic memoization, $\alpha = \inf$ no memoization
https://probmods.org/chapters/12-non-parametric-models.html

6.1 Link between BNP and High order PPL

See Frank Wood meeting

Future Work

In this chapter are presented several ideas which I intend to further developed. When matured enough, this ideas may become a project on itself.

7.1 Learning parameters in PPL

7.1 Motivation

Since variational methods have arisen, ideas of mixing sampling with variational inference (VI) have emerged, including in the PPL literature.

In [48], the authors introduce the idea of automatically learning the parameters of proposals for Sequential Monte Carlo (SMC) within a PPL. A lower bound on the KL divergence between the proposal and the true posterior distribution is optimize via gradient descent. In [38, 24], this idea is further developed using neural networks (such as LSTMs [22]) to parametrize these proposal distributions. These networks are fed with the previous latent and observed variables. In AESMC [25], FIVO [30] and VSMC [35], both the model and the SMC's proposal are learned by maximizing the marginal likelihood estimator given by the SMC.

The interest in learning parameters (for proposals and for the model) and performing inference on some random variables at once is thus great. PPLs allow to easily write probabilistic models and perform inference on latent variables. One may be interested in building a PPL with the capability of automatically optimizing some parameters given a loss/estimator.

Automatic Differentiation (AD) methods [2] enable the computation of gradients of some variables with respect to some parameters. The reverse differentiation is particularly popular in the machine learning community, where the history of each variable (how it has been constructed) is saved as a computational graph, and gradients can then be computed via the *chain rule*.

Some libraries such as TensorFlow [1] require the users to define static computation graphs within the syntactic and semantic constraints of a domain-specific mini language with limited support for control flow whereas the lineage of projects leading from autograd [1]

[1] https://github.com/HIPS/autograd

to PyTorch [2] provide truly general-purpose reverse mode AD capability. The latter mode is to be preferred for fully and easy support of control flow such as stochastic recursion which is needed for stick-breaking processes.

7.1 Choice of language/library

We this idea in mind, one can now think how to pragmatically build such a AD PPL.

Python: One of the most famous language for scientific computing is Python [40]. As *Edward* [43] is built on top of *Tensorflow*, one could build a PPL layer on top of PyTorch. *Edward* implements each Markov chain Monte Carlo (MCMC) step (specific for each algorithms) as a computational graph in *Tensorflow* which is thereafter run with the updated input so as to sample a new value. Similarly for VI, *Edward* implements a loss function as a computational graph, for which its gradient can be computed via auto-differentiation.

However, *Edward* focuses on VI and Hamiltonian Monte Carlo (HMC)-like schemes and does not handle particle algorithms. Indeed, so as to handle such algorithms, a PPL must have access to *breakpoints* at `assume` statement. This can be implemented via CPS [3] or coroutine [4] copying. Unfortunately implementing CPS is something non-trivial.

Julia: One could also think of using Julia [3], which has been specifically built for scientific usage. Julia has the advantage of natively handling coroutine copying, which is used in Turing [18] to implement particle methods.

Reverse mode AD libraries exist in Julia, *ReverseDiff.jl* [5] and *Knet.jl* [6] which respectively build a static and dynamic graph.

I am particularly interested in the perspective of adapting a AD library for Turing [18].

7.1 New models

With such as PPL in mind, one can think of new model or algorithms to be developed.

The idea of AESMC [25] might be extended to Particle Gibbs (PG) and Particle Marginal Metropolis-Hastings (PMMH) so as to learn proposals' parameters for their SMC and for $p(\theta^\star|\theta)$ parameters (specific of PMMH).

7.1 Difficulties

Yet this is not a trivial task, one have to put proper care when computing unbiased gradient of a loss function defined by an expectation over a collection of random variables.

[2] http://pytorch.org/
[3] http://matt.might.net/articles/by-example-continuation-passing-style/
[4] https://en.wikipedia.org/wiki/Coroutine
[5] https://github.com/JuliaDiff/ReverseDiff.jl
[6] https://github.com/denizyuret/Knet.jl

Hopefully, a stochastic computation graph [41] can be converted into a deterministic computation graph, to which the backpropagation algorithm can then be applied on a surrogate loss function which results in an unbiased gradient estimator of the loss.

7.2 Variational Inference for BNP in PPL

To my knowledge, the first article tackling inference in a BNP setting is [6], where a truncated proposal is introduced to approximate a Dirichlet Process Mixture model.

Truncation-free VI methods have also been introduced [7]. These methods adapts model complexity on the fly by lazily representing clusters assignments. Yet, the sticks proportions and mixture components are marginalize out to obtain a closed form distribution for the mixture assignment hidden variables z_i. This marginalization is unfortunately only tractable for few models, such as the Dirichlet Process and the Pitman-Yor process.

However, we may be able to use a similar approach for more flexible BNP models, by extending the latent space with the sticks proportions and mixture components (since they cannot be marginalized out). Moreover, there might be a deeper link between *Truncation-free* VI and stick-breaking processes.

7.3 Adversarial Inference for BNP in PPL

Adversarial inference methods [14, 12, 33] inspired by GANs [19] jointly learns a generation network and an inference network using an adversarial process.

The decoder/generator network $x' \sim p(x|z)$ maps samples from stochastic latent variables to the data space while the encoder/inference network $z' \sim q(z|x)$ maps training examples in data space to the space of latent variables.

An adversarial game is cast between these two networks and a discriminative network is trained to distinguish between joint latent/data-space samples (x', z) from the generative network and joint samples (x, z') from the inference network.

Adversarial inference seems to be closely related to VI. Yet, in adversarial inference the model can also be learned as opposed to VI where only the proposal is learned. Moreover, in VI the marginal likelihood is optimised via a lower bound (ELBO) whereas in adversarial inference, a classification loss is optimised.

This approach could be interesting in the BNP setting, if a tractable and tight lower bound on the marginal likelihood cannot be found.

7.4 Piecewise Deterministic Markov Processes

A novel class of non-reversible Markov chain Monte Carlo schemes relying on continuous-time piecewise deterministic Markov Processes has recently emerged [45]. In these algorithms, the state of the Markov process evolves according to a deterministic dynamics which is modified using a Markov transition kernel at random event times. A general

framework is presented in [4], and includes among others the Zig-Zag Process [5], the Bouncy Particle Sampler [8] and the Generalized Bouncy Particle Sampler [51].

It has been claimed [4] that the non-reversibility property of these algorithms enhances the mixing rate of the chain. I am consequently interested in understanding to what extent this class of MCMC schemes could fit the PPL's setting.

Conclusion and personal review

8.1 Conclusion

Ouverture

8.2 Personal review

Responsability: organization of the probabilistic inference reading group. Research environment Research way of working / thinking: Finding the good questions, etc

Appendices

A Variational Inference ?

B Automatic Differentiation ?

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow: A system for large-scale machine learning*, in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.

[2] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, arXiv.org, (2015).

[3] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A Fresh Approach to Numerical Computing*, SIAM Review, 59 (2017), pp. 65–98.

[4] J. Bierkens, A. Bouchard-Côté, A. Doucet, A. B. Duncan, P. Fearnhead, G. Roberts, and S. J. Vollmer, *Piecewise Deterministic Markov Processes for Scalable Monte Carlo on Restricted Domains*, arXiv.org, (2017).

[5] J. Bierkens, P. Fearnhead, and G. Roberts, *The Zig-Zag Process and Super-Efficient Sampling for Bayesian Analysis of Big Data*, arXiv.org, (2016).

[6] D. M. Blei and M. I. Jordan, *Variational inference for Dirichlet process mixtures*, Bayesian Analysis, 1 (2006), pp. 121–143.

[7] D. M. Blei and C. Wang, *Truncation-free Stochastic Variational Inference for Bayesian Nonparametric Models*, Advances in Neural Information Processing Systems th Annual Conference on Neural Information Processing Systems, (2012), pp. 422–430.

[8] A. Bouchard-Côté, S. J. Vollmer, and A. Doucet, *The Bouncy Particle Sampler: A Non-Reversible Rejection-Free Markov Chain Monte Carlo Method*, Journal of the American Statistical Association, 4 (2017), pp. 0–0.

[9] B. Carpenter, D. Lee, M. A. Brubaker, A. Riddell, A. Gelman, B. Goodrich, J. Guo, M. Hoffman, M. Betancourt, and P. Li, *Stan: A probabilistic programming language*.

[10] T. D. Chen and C. Fox, Emily B.and Guestrin, *Stochastic gradient hamiltonian monte carlo*, in International Conference on Machine Learning, 2014.

[11] P. Del Moral and L. M. Murray, *Sequential Monte Carlo with Highly Informative Observations*, SIAM/ASA Journal on Uncertainty Quantification, 3 (2015), pp. 969–997.

[12] J. Donahue, P. Krähenbühl, and T. Darrell, *Adversarial Feature Learning*, arXiv.org, (2016).

[13] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, *Hybrid monte carlo*, Physics Letters B, 195 (1987), pp. 216 – 222.

[14] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, *Adversarially Learned Inference*, arXiv.org, (2016).

[15] S. Favaro, M. Lomeli, B. Nipoti, and Y. W. Teh, *On the stick-breaking representation of σ-stable Poisson-Kingman models*, Electronic Journal of Statistics, 8 (2014), pp. 1063–1085.

[16] S. Favaro, M. Lomeli, and Y. W. Teh, *On a class of σ-stable Poisson–Kingman models and an effective marginalized sampler*, Statistics and Computing, 25 (2014), pp. 67–78.

[17] S. Favaro and Y. W. Teh, *MCMC for Normalized Random Measure Mixture Models*, Statistical Science, 28 (2013), pp. 335–359.

[18] H. Ge, A. Ścibior, K. Xu, and Z. Ghahramani, *Turing: A fast imperative probabilistic programming language.*, (2016).

[19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative Adversarial Networks*, arXiv.org, (2014).

[20] N. Goodman, V. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum, *Church: a language for generative models*, arXiv.org, (2012).

[21] N. D. Goodman and A. Stuhlmüller, *The Design and Implementation of Probabilistic Programming Languages*. `http://dippl.org`, 2014. Accessed: 2017-8-29.

[22] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[23] M. D. Homan and A. Gelman, *The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo*, J. Mach. Learn. Res., 15 (2014), pp. 1593–1623.

[24] T. A. Le, A. G. Baydin, and F. Wood, *Inference Compilation and Universal Probabilistic Programming*, arXiv.org, (2016).

[25] T. A. Le, M. Igl, T. Jin, T. Rainforth, and F. Wood, *Auto-Encoding Sequential Monte Carlo*, arXiv.org, (2017).

[26] M. Lomeli, S. Favaro, and Y. W. Teh, *A hybrid sampler for Poisson-Kingman mixture models*, arXiv.org, (2015).

[27] ——, *A Marginal Sampler for σ-Stable Poisson–Kingman Mixture Models*, Journal of Computational and Graphical Statistics, 26 (2017), pp. 44–53.

[28] D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, *Winbugs - a bayesian modelling framework: Concepts, structure, and extensibility*, Statistics and Computing, 10 (2000), pp. 325–337.

[29] Y.-A. Ma, T. Chen, and E. B. Fox, *A complete recipe for stochastic gradient mcmc*, in Proceedings of the 28th International Conference on Neural Information

Processing Systems, NIPS'15, Cambridge, MA, USA, 2015, MIT Press, pp. 2917–2925.

[30] C. J. Maddison, D. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. W. Teh, *Filtering Variational Objectives*, arXiv.org, (2017).

[31] V. Mansinghka, D. Selsam, and Y. Perov, *Venture: a higher-order probabilistic programming platform with programmable inference*, arXiv.org, (2014).

[32] A. McCallum, K. Schultz, and S. Singh, *FACTORIE: Probabilistic programming via imperatively defined factor graphs*, in Neural Information Processing Systems (NIPS), 2009.

[33] L. M. Mescheder, S. Nowozin, and A. Geiger, *Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks*, in Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, 2017, pp. 2391–2400.

[34] T. Minka, J. Winn, J. Guiver, S. Webster, Y. Zaykov, B. Yangel, A. Spengler, and J. Bronskill, *Infer.NET 2.6*, 2014. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

[35] C. A. Naesseth, S. W. Linderman, R. Ranganath, and D. M. Blei, *Variational Sequential Monte Carlo*, arXiv.org, (2017).

[36] Y. Nesterov, *Primal-dual subgradient methods for convex problems*, Math. Program., 120 (2009), pp. 221–259.

[37] B. Paige, F. Wood, A. Doucet, and Y. W. Teh, *Asynchronous Anytime Sequential Monte Carlo*, arXiv.org, (2014).

[38] D. Ritchie, P. Horsfall, and N. D. Goodman, *Deep Amortized Inference for Probabilistic Programs*, arXiv.org, (2016).

[39] D. Ritchie, A. Stuhlmüller, and N. D. Goodman, *C3: Lightweight Incrementalized MCMC for Probabilistic Programs using Continuations and Callsite Caching*, arXiv.org, (2015).

[40] G. Rossum, *Python reference manual*, tech. rep., Amsterdam, The Netherlands, The Netherlands, 1995.

[41] J. Schulman, N. Heess, T. Weber, and P. Abbeel, *Gradient Estimation Using Stochastic Computation Graphs*, arXiv.org, (2015).

[42] Y. Teh, D. Görür, and Z. Ghahramani, *Stick-breaking construction for the indian buffet process*, in JMLR Workshop and Conference Proceedings Volume 2: AISTATS 2007, Cambridge, MA, USA, Mar. 2007, Max-Planck-Gesellschaft, MIT Press, pp. 556–563.

[43] D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei, *Deep probabilistic programming*, in International Conference on Learning Representations, 2017.

[44] J.-W. van de Meent, H. Yang, V. Mansinghka, and F. Wood, *Particle Gibbs with Ancestor Sampling for Probabilistic Programs*, arXiv.org, (2015).

[45] P. Vanetti, A. Bouchard-Côté, G. Deligiannidis, and A. Doucet, *Piecewise Deterministic Markov Chain Monte Carlo*, arXiv.org, (2017).

[46] M. Welling and Y. W. Teh, *Bayesian learning via stochastic gradient langevin dynamics.*, in ICML, L. Getoor and T. Scheffer, eds., Omnipress, 2011, pp. 681–688.

[47] D. Wingate, N. D. Goodman, and A. Stuhlmüller, *Lightweight Implementations of Probabilistic Programming Languages Via Transformational Compilation*, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, USA, Apr. 2011, pp. 770–778.

[48] D. Wingate and T. Weber, *Automated Variational Inference in Probabilistic Programming*, arXiv.org, (2013).

[49] F. Wood, J. W. van de Meent, and V. Mansinghka, *A new approach to probabilistic programming inference*, in Proceedings of the 17th International conference on Artificial Intelligence and Statistics, 2014, pp. 1024–1032.

[50] F. Wood, J.-W. van de Meent, and V. Mansinghka, *A New Approach to Probabilistic Programming Inference*, arXiv.org, (2015).

[51] C. Wu and C. P. Robert, *Generalized Bouncy Particle Sampler*, arXiv.org, (2017).