MSc Project - Reflective Essay

Project Title:	Designing a benchmarking suite for probabilistic programming languages
Student Name:	Patrick Pagni
Student Number:	232046906
Supervisor Name:	Dr. Fredrik Dahlqvist
Programme of Study:	Artificial intelligence MSc

The aim of this project was to design a benchmarking suite which can measure the performance of probabilistic programming languages with reference to an exact posterior distribution. This essay will analyse the strengths and weaknesses of the project from a holistic perspective. Additionally, there will be a discussion on the relationship between the practical deliverable and the theoretical research completed. Finally, there will be a section detailing further work which could be conducted with regards to this project.

1 Project strengths and weaknesses

This section will discuss strengths and weaknesses of the project. This project was divided into three work streams: developing distance measures to compare probability distributions; building the infrastructure to evaluate PPLs scalably; and defining a problem set which provides good coverage over the infinite set of problems. These work streams were prioritised in the same order they have been listed above. The end of this section will discuss the effect of prioritising these work streams in this way.

This project explores the usefulness of two performance measures in terms of measuring the distance between high-dimensional distributions. There is a loop-based implementation of an extension to d-dimensional Kolmogorov-Smirnov Test where instead of comparing two empirical CDFs; an empirical CDF is compared to an analytic CDF. This required significant effort to develop an algorithm where on can calculate the orthant density arithmetic in a manner which is agnostic to the dimensionality of the space. Developing this algorithm can be considered a success.

Nevertheless, the current implementation of this algorithm is flawed as shown in the research paper: the time taken to compute the significance score is too high to be considered usable for high-dimensional distributions. This should not necessarily be considered a weakness of the project in itself; rather it is an area for further development of the algorithm in order to accelerate the computation. While it would have been preferred to have an optimised version of the algorithm, the inability for the loop-based implementation to scale effectively is simply a finding of this project after evaluating the benchmarking suite.

One could contend that two implemented performance measures cannot result in a robust evaluation of probabilistic languages. The author is sympathetic with this perspective. The aim while developing this benchmarking suite was to capture as much information as possible, which is why the model execution time was also captured. While completing this project, one direction the author considered going in was to develop more efficient methodologies for measuring the distance between high-dimensional probability distributions, and significant energy went into this

during the early phases of the project. As a result, one could say the author had tunnel vision in wanting to implement fewer more complex distance measures, rather than implementing several simple measures.

This exacerbates the fact that the d-dimensional KS test has a sub-optimal implementation since one would hope that if most efforts went into developing the algorithm for this, it would be implemented optimally. Unfortunately, the loop-based approach was in itself very challenging to implement; the approach for implementing it PyTorch will be quite different the author suspects; and to have a minimum viable project for this project it was important to focus on other work streams.

Another area of strength for this project is that this benchmarking suite is provably compatible with three different PPLs. The three PPLs implemented for this benchmarking suite are PyMC, Pyro and Stan. Two of these were implemented very easily since they are Python libraries, but Stan was more complicated since it is a separate programming language in its own right. By developing a benchmarking pipeline which allows one to define models in a PPL and then applying the problem parameters to each model, it made adding new models for each PPL very straightforward. Indeed, this modular approach means the author anticipates that adding more probabilistic languages to this benchmarking suite should not provide a significant challenge.

Furthermore, by testing three separate PPLs using the benchmarking suite, one was able to show the potential for comparing the performance of the PPLs (albeit the problem set did not find any areas of separation performance-wise.) While it shows the performance of each PPL is similar, the format of the output shows that comparison between each PPL is achievable using this benchmarking suite.

On the other hand, this benchmarking could be improved by packaging it properly as a python library. Currently, the benchmarking suite is run from the command line and one runs the main.py script with optional arguments for the problems to run; after the results are extracted it saves these results to a csv file. While this is an acceptable implementation for the purposes of a prototype, it would be more elegant to package it as a python library which can be run in a python terminal or jupyter notebook.

A major area for improvement for this project would have been to develop a more curated problem set to test the limits of the probabilistic programming languages and try to identify areas in which one language performs better than another. A significant challenge for the author to account for is the fact that there is no clear separation in the performance of PPLs according to this problem set. This means that while it has been shown that one can analyse the performance of PPLs in isolation using this benchmarking suite, there is no evidence yet that this methodology (of comparing a PPL's found distribution against a reference) surfaces performance difference between PPLs.

The author believes, however, that the failure to produce compelling comparisons between the performance of each PPL (besides Stan's significantly faster model execution time) is the product of the limited problem set rather than the overall approach; or at least one should extend the problem set sufficiently that such a conclusion can be drawn.

As discussed in the paper, the potential set of problems is infinite. Even when constrained to problems which have exact reference solutions using conjugate priors, the parameters one can change for the problem set are sample size; distribution of samples; data dimensionality; prior distribution family of estimated parameters; prior distribution hyperparameters for distribution of estimated parameters; implementation of probabilistic model. Granted, after some analysis it did not appear that sample size had a significant bearing on the problems contained within the problem set, but it is conceivable that for more complex probabilistic models, sample size could still have some effect on the outcome. Developing a curated set of problems which manipulate these levers

should be the highest priority work for further development of this benchmarking suite.

The author believes developing a curated problem set will be one which is driven by more research into the theory that underpins PPL models, especially for problems where the exact solution is calculated using conjugate priors. It will require more research into the theory of how to apply priors onto different parameters, what is considered an effective prior for a mean versus a covariance for example. Again, the author's research has found that there are 5 different kinds of priors: flat prior; vague but proper prior; weakly informative prior; generic weakly informative prior; specific informative prior. [6] This is a useful starting point for thinking about how to develop a problems which cover each of these kinds of prior, but what weakly informative priors are for one PPL model with a certain set of parameters may be different for another PPL model. Therefore this will require a deeper understanding of the relationship between the parameters and the prior choice.

For example, when estimating the covariance of a multivariate normal distribution; the PPL models performed well and found covariance matrices which appeared to fit the data well (not rigorously tested so not present in the research paper.) When the PPL models were designed to find the scale matrix of a multivariate Student's T distribution, the performance appeared to be much worse, with the model having a much smaller scale than the conjugate prior solution. Unfortunately, these outcomes did not make it into the research paper since they required more work to verify the outputs from the conjugate prior model and the PPL models.

A core challenge faced during the development of this benchmarking suite is that to develop a minimum viable product it was necessary to develop a minimum viable product in each of these work streams. It would have been impossible to deliver a benchmarking suite which was missing performance measures; a benchmarking infrastructure or a problem set. As a result, a minimum amount of development had to be completed in three orthogonal directions.

This caused the project to suffer because time which could have been spent developing an optimal implementation of the d-dimensional Kolmogorov-Smirnov Test, for example, was spent in developing the benchmarking infrastructure; or developing a limited problem set. The author tried to limit the scope of the project in order to prioritise the development of the distance metrics, and the building of the infrastructure to utilise these metrics. By sacrificing the quality of the problem set, however, the author is concerned that the quality of the results has suffered since he does not feel the PPLs a rigorously tested at this stage.

If one were to repeat the project, one might sacrifice the quality of the benchmarking infrastructure itself to prioritise the quality of the problem set. The reason for this is that armed with performance measures and a problem set, one can complete ad hoc analyses which generate more interesting results than those produced by the testing of this iteration of the benchmarking suite. In many ways this benchmarking suite was developed more with the idea of how it can be extended beyond the scope of this project, rather than focusing on one of the core aims of the project: generate and analyse results.

To conclude this section, the project has been a success insofar as developing a prototype for a benchmarking suite which compares PPL distributions against a reference distribution. Having said that, there are several areas for improvement, such as the implementation of the d-dimensional KS test and the limited problem set. The main weakness of this project is in the focus on developing the benchmarking infrastructure at the expense of expanding the problem set. This mistake in prioritisation resulted in the results and analysis portion of this project being insubstantial. While it was possible to derive some results and analysis, adding more problems would have allowed for a more comprehensive evaluation of PPLs.

2 Relationship between practical work and theoretical work

This section will discuss how to derive a reference distribution using conjugate priors. It will also outline the research required for the development of the performance measures.

The original brief for this benchmarking suite proposed measuring the performance of probabilistic programming languages against a reference distribution and this reference distribution can be exactly calculated using conjugate priors. At the beginning of this project, the author was unfamiliar with conjugate priors and how they can be used to calculate the exact posterior distribution given some data. After some research, it became clear that conjugate priors could be used to state the posterior parameters in terms of the prior parameters and the data. [3] For example, take the scenario where one modelling the prior and posterior as a Normal distribution with unknown mean but known variance (σ^2). The prior hyperparameters are μ_0, σ_0^2. With a dataset $x_i \in X$ one can exactly calculate the posterior hyperparameters:

$$\frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}} \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} x_i}{\sigma^2} \right), \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2} \right)^{-1}$$

After calculating these parameters, the posterior predictive distribution (i.e. the reference distribution) is given by $\mathcal{N}(\tilde{x}|\mu_0', \sigma_0^{2'} + \sigma^2)$. Understanding conjugate priors and how they were used was a keystone for this project since it was vital that this could be implemented. In fact, after realising that conjugate priors essentially provide straightforward formulae from the priors to the predictive posterior distribution, the implementation was straightforward.

For the development of the performance measures, the research going into the development of the KL divergence had to be stopped because the specific idea could not be implemented in the time. This research is detailed in further work. The research for d-dimensional KS test, however, was almost exclusively underpinned by the work completed by Hagen et al. in [5]. Their work did still need to be adapted and changed to compare an analytic CDF with an empirical CDF. Developing the algorithm to segment a CDF into orthants and find the density of each orthant was an original undertaking (from the author's perspective). It was not developed in tandem with reading any theoretical work but was developed by identifying patterns in the arithmetic as the dimensions increase. It would be reasonable to say that this extends the work completed by Hagen et al. and it is more in keeping with the original KS test which compares an empirical CDF and an analytic CDF. [2]

3 Further work

This section will discuss further work which can be completed to enhance this project. It will also discuss further research opportunities: either building on this project, or research opportunities which came to light during research into this project.

To get the most out of this benchmarking suite, it will be important to extend the problem set to comprehensively evaluate probabilistic programming languages. This is the highest priority item since it will enable proper evaluation of the viability of the methodology of comparing a found distribution with a reference distribution.

It would also be interesting to invest more time into the development of distance measures for the benchmarking suite. For starters, one could optimize the implementation of the d-dimensional KS test. This is a challenging problem since achieving the same outcome using approaches which take advantage of accelerated computing (such as pytorch) may be quite different to the loop-

based implementation. Alternatively one could manually parallelize the computation for different subsets of test points using threading. The advantage would be one could keep the loop-based implementation. It would be preferable however, if one could implement it using a library designed for accelerated computation such as pytorch, then it would be simple to use GPUs for these computations too. Beyond the optimization of the d-dimensional KS test, one could add more metrics which capturing more information about PPL performance. For example, one could add the Gelman-Rubin convergence statistic which measures for which problems the models converged to a stationary distribution versus which models require longer runs. [1]

For the benchmarking suite itself, the author would have liked to have added unit testing to the functions within the repository to ensure they were robust and reliable. It would have also been preferable to have packaged the prototype as a python library, and built more robust distribution architecture. Nevertheless, this is work which can be completed in the coming weeks after this assessment process is completed.

Finally, during the development of the performance measures of the benchmark suite, the intention was to explore efficient ways to make the Kullback-Leibler divergence calculation scale up to very high dimensions. Much of the research done in this area strayed too far from the brief and would not have been applicable to the project itself, but it is interesting to briefly mention it here. The trouble with calculating the KL divergence is that it is a computationally expensive task since one is numerically integrating high dimensional distributions. This can result in wasted computational resources since if the algorithm is integrating across the entire support of the distribution it can be integrating over areas where the probability density is very low. An idea was explored to use Hamiltonian Monte Carlo Markov Chains to identify the typical set of the probability distribution. This technique finds the areas of the probability distribution which are high density and so will contribute significantly to the integral. [4] Once the typical set of the probability distribution has been found, one can constrain the support for the KL divergence calculation to it. Unfortunately, after some research into practical implementations of HMCMC sampling, it seemed there was no implementation suited to this purpose. As a result, this line of research was stopped to prioritise the development of the benchmarking suite. It is still an idea which the author stumbled upon during this project and would be interested to explore further.

To summarise, opportunities for further research extending this benchmarking suite would be to extend the problem set, optimise the d-dimensional KS test and add further metrics. It would also be to add testing to the repository and improve the distribution of the package. There is also the proposal to use HMCMC to set a dynamic support the KL divergence to prevent integrating over low density areas, but this work is seaparate from the project itself.

References

[1] Andrew Gelman and Donald B. Rubin. “Inference from Iterative Simulation Using Multiple Sequences”. In: *Statistical Science* 7.4 (1992), pp. 457–472. ISSN: 08834237. URL: http://www.jstor.org/stable/2246093 (visited on 08/20/2024).

[2] Kevin Ford. “From Kolmogorov’s theorem on empirical distribution to number theory”. In: *Kolmogorov’s Heritage in Mathematics*. Ed. by Éric Charpentier, Annick Lesne, and Nikolaï K. Nikolski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 97–108. ISBN: 978-3-540-36351-4. DOI: 10.1007/978-3-540-36351-4_5. URL: https://doi.org/10.1007/978-3-540-36351-4_5.

[3] Kevin Murphy. “Conjugate Bayesian analysis of the Gaussian distribution”. In: (Nov. 2007).

[4] Michael Betancourt. *A Conceptual Introduction to Hamiltonian Monte Carlo*. 2018. arXiv: 1701.02434 [stat.ME]. URL: https://arxiv.org/abs/1701.02434.

[5] Alex Hagen et al. *Accelerated Computation of a High Dimensional Kolmogorov-Smirnov Distance*. 2021. arXiv: 2106.13706 [stat.CO]. URL: https://arxiv.org/abs/2106.13706.

[6] Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual*. Version 2.35. 2023. URL: https://mc-stan.org.