Group Report

Advanced Topics in Machine Learning
Reproducibility Challenge

April 30, 2020

1 Introduction

In this assessment, we chose to reproduce and extend the results of the paper "Automatic Variational Inference in Stan" written by Kucukelbir et al. [1]. As several members of the group had already been exposed to working with neural networks and frequentist approaches to modelling, we decided to choose a paper in the field of Bayesian inference, a field that no member of the group had been previously exposed to. The central hurdle in Bayesian inference is to approximate the posterior distribution over the parameters of a probabilistic model given a set of data. By exploring the novel approach introduced by the paper on how to tackle this problem, it allowed us to gain a good understanding of the long-standing challenges and new developments in the field.
The paper outlines an algorithm for Automatic Differential Variational Inference (ADVI) for Bayesian inference, which was implemented for the probabilistic programming system (PPS) Stan. It furthermore assesses ADVI experimentally by applying it to five different probabilistic models and comparing its performance to two baseline methods. Building on this, we formulated four goals for our project.

1. Understand the motivation of ADVI, how it works and how it compares to other Bayesian inference methods.

2. Efficiently implement ADVI and the experiments of the paper.

3. Evaluate the reproducibility of the experimental results of the paper.

4. Investigate further properties of ADVI to extend the results of the paper.

We managed to implement ADVI and replicate four out of the five experiments in the PPS TensorFlow Probability. We were able to reproduce the experimental results with varying success. Even though our ADVI implementation performed roughly as reported, we found the baseline methods to outperform ADVI after optimising their hyper-parameters.
This report is structured as follows. In Section 2, we summarise the key aspects of the paper. In Section 3, we explain our code base and discuss what difficulties we faced throughout the implementation process. In Section 4, we discuss the experiments we reproduced and extend the results with our own experiment.

2 Background

In this section, we summarise the key aspects of ADVI as described in the papers [1] and [2]. We will focus on aspects which were relevant to implementing ADVI and how it was used in the experiments. For the derivations and motivations of these respective steps, we refer the reader to the aforementioned papers.

Approximate Bayesian Inference

Given a set of data $\mathbf{X}$ and latent variables θ, consider a Bayesian probability model $p(\mathbf{X}, \theta) = p(\mathbf{X}|\theta) \cdot p(\theta)$. The goal of Bayesian inference is to compute the probability distribution over the latent variables given the data, i.e. the posterior distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta) \cdot p(\theta)}{p(\mathbf{X})}.$$

As computing the posterior is often difficult in machine learning, one usually tries to approximate it using a Markov Chain Monte Carlo (MCMC) method or Variational Inference (VI). Even though VI tends to be faster than MCMC, it typically requires tedious model-specific derivations (see [1]).

Automatic Differentiation Variational Inference (ADVI)

The authors of the paper develop ADVI, a method that automatically performs VI for any Bayesian model that has continuous latent variables $\theta \in \mathbb{R}^K$ and is differentiable w.r.t. θ. For that purpose, they define a family of probability distributions $q(\theta; \mu, \omega)$ parameterised over $\mu, \omega \in \mathbb{R}^K$ and try to find μ^*, ω^* such that $q(\theta; \mu^*, \omega^*)$ approximates $p(\theta|\mathbf{X})$ as closely as possible, i.e. minimises the Kullback-Leibler divergence $\mathrm{KL}\left(q(\theta; \mu, \omega) \;||\; p(\theta|\mathbf{X})\right)$ and, equivalently, maximises the ELBO $\mathcal{L}(\mu, \omega)$.
For specifying the variational family $q(\theta; \mu, \omega)$, the authors posit a diagonal (mean-field) Gaussian on an intermediate latent variable vector $\zeta \in \mathbb{R}^K$:

$$q(\zeta; \mu, \omega) = \prod_{k=1}^{K} \mathcal{N}(\zeta_k; \mu_k, \sigma_k^2) \quad \text{with } \sigma_k = \exp(\omega_k).$$

Let $\mathrm{supp}(p) = \{\theta \mid p(\theta) > 0\}$ be the support of p. The shape of $\mathrm{supp}(p)$ depends on the Bayesian model. However, ζ has support over the whole real

coordinate space $\mathbb{R}^K$. ADVI uses a bijection T between $\text{supp}(p)$ and $\mathbb{R}^K$ to transform ζ into θ, i.e. $\theta = T^{-1}(\zeta)$. Hence, one can derive that

$$q(\theta; \mu, \omega) = \mathcal{N}\left(T^{-1}(\zeta); \mu, \sigma^2\right) \cdot |\det J_{T^{-1}}(\zeta)|$$

defines an implicitly non-Gaussian variational family with $\text{supp}(q) = \text{supp}(p)$. Note that the choice of T depends on the Bayesian model. In practice, Stan automatically selects T from a library of transformations (see [3]). We refer to $\text{supp}(p)$ as the constrained space and to $\mathbb{R}^K$ as the unconstrained space. Using the re-parametrisation trick with $\zeta = \mu + \text{diag}(\sigma)\eta$ and $\eta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the authors show that the ELBO and its gradient with respect to the variational parameters can be written as

$$\mathcal{L}(\mu, \omega) = \mathbb{E}_{\mathcal{N}(\eta)}\left[\log p\left(\mathbf{X}, T^{-1}(\zeta)\right) + \log |\det J_{T^{-1}}(\zeta)|\right] + \sum_{k=1}^{K} \omega_k, \tag{1}$$

$$\nabla_\mu \mathcal{L} = \mathbb{E}_{\mathcal{N}(\eta)}\left[\nabla_\theta \log p(\mathbf{X}, \theta) \nabla_\zeta \log T^{-1}(\zeta) + \nabla_\zeta \log |\det J_{T^{-1}}(\zeta)|\right], \tag{2}$$

$$\nabla_{\omega_k} \mathcal{L} = \mathbb{E}_{\mathcal{N}(\eta_k)}\left[\left(\nabla_{\theta_k} \log p(\mathbf{X}, \theta) \nabla_{\zeta_k} \log T^{-1}(\zeta) + \nabla_{\zeta_k} \log |\det J_{T^{-1}}(\zeta)|\right) \eta_k \sigma_k\right] + 1. \tag{3}$$

The gradients inside the expectations in equations (2)-(3) can be computed by automatic differentiation. To approximate the expectations in equations (1)-(3), ADVI uses MC integration with M η-samples. Building on that, ADVI initialises $\mu = \omega = \mathbf{0}$ and optimises these parameters via a gradient descent algorithm.

Baseline Methods

The performance of ADVI is compared to two MCMC methods - Hamiltonian Monte Carlo (HMC) and the No-U-Turn Sampler (NUTS). These are the standard methods for Bayesian inference in Stan. In general, MCMC methods compute one θ-sample per step by applying a randomised transition function to the previous θ-sample. The transition function is designed such that the sampling process converges to sampling from the posterior $p(\theta|\mathbf{X})$.
Hamiltonian Monte Carlo (HMC) [4, 5] avoids the slow exploration of state space experienced by random walk MCMC algorithms such as Gibbs sampling by applying Hamiltonian dynamics. By taking first-order gradients, trajectories can be approximated to accelerate convergence. However, the performance of HMC is highly dependent on its step size and the number

of steps, these are hyper parameters that need to be hand-tuned. The No U-Turn Sampler (NUTS) [6] algorithm extends HMC by adding a stopping criterion removing the need for a step limit. Additionally, the step size can be adjusted by using primal-dual averaging [7], so that NUTS can then be used with no hand-tuning at all.

Experimental Setup

The authors compare ADVI with HMC and NUTS on 5 different Bayesian models and data sets. For that purpose, they define the following performance measure. To place ADVI on the same scale as HMC and NUTS, one θ-sample is drawn from the variational ADVI model after every optimisation step. For each method, they approximate the predictive likelihood on held out data $\hat{p}_t \approx p(\mathbf{X}^*|\mathbf{X}) = \int p(\mathbf{X}^*|\theta)p(\theta|\mathbf{X})d\theta$ using Monte Carlo estimation with all samples drawn until step t. More precisely, they report the averaged logarithm of the approximate predictive likelihood, i.e. $\frac{1}{ND}\log\hat{p}_t$, where N is the number of data points in $\mathbf{X}^*$ and D the number of dimensions of one data point. The authors refer to this performance measure as "average log predictive". The larger the average log predictive, the higher the accuracy of the method.

3 Implementation

In this section, we explain how we implemented ADVI and our experiments. Besides giving an overview of the code structure, we focus on the motivations behind the key choices we made and outlining the difficulties we faced. With these difficulties in mind, we evaluate the ease of reproducing the paper by outlining the extra resources required. The code for our project can be found here on our anonymous git repository: `https://github.com/AdvancedML2020/ADVI`

TensorFlow Probability

Our first step was to chose a probabilistic programming system (PPS) for reproducing the results. In the paper [1], ADVI and the experiments are written in Stan. However, we decided to work with the Python library TensorFlow Probability (`tfp`) for the following reasons.

- TensorFlow (Probability) is widely in the Machine Learning research community. We believe that the experience we gain from learning TensorFlow will also help us in future endeavours.

- To the best of our knowledge, ADVI has not been implemented for TensorFlow Probability before, giving us the opportunity to make a contribution to the community and introduce ADVI to a wider audience.

- TensorFlow Probabilty comes with comprehensive libraries of distributions (`tfp.distributions`) and bijections (`tfp.bijectors`), both needed for effectively implementing ADVI and the probabilistic models for our experiments. It also provides implementations of the baseline methods HMC and NUTS.

Code Structure

Please refer to our anonymous git repository linked above for the code.

- ADVI implementation: `advi/`
  There are two components to our implementation of ADVI. First, an instance of the `ADVIModel` class must be instantiated from `advi/model.py`. The `ADVIModel` object represents the variational family $q(\theta; \mu, \omega)$. Then, inference can be run using `advi/core.py`, where our gradient descent algorithm is applied on the variational parameters in the unconstrained space, minimising the ELBO with respect to μ and ω. These parameters and the ELBO are instance variables of the `ADVIModel` class, allowing for an easy access during run-time via a tracing function.

- Bayesian models for the experiments: `models/`
  The probabilistic models which were fitted are defined as classes. We decided to formalise how we define these classes so that they had common methods which were necessary for running inference using ADVI or the sampling methods. This allowed us to use the different inference methods interchangeably with different models in a streamlined manner.

- Training and logging: `train_log.py`
  For initial testing, we used TensorBoard to as a sanity check to see that our models were converging. However we found that it did not have enough flexibility, therefore we decided to implement our own logging

and plotting functionality which can be found under `train_log.py` and `utils/plot.py`. Tracing functions were used to output the value for the average log predictive for the sampling methods. These are functions which can be passed in to the TensorFlow Probability function call `tfp.mcmc.sample_chain()`, which allows the average log predictive to be calculated and logged using the intermediate results.

- Testing: `testing/`
  We developed test cases for ensuring the correctness of our implementation. We focused on checking if the gradients in ADVI are computed correctly and whether ADVI finds the correct parameters of a simple multi-dimensional Gaussian toy posterior distribution.

Difficulty 1 – Transformation Function T

Recall from Section 2 that a crucial component of ADVI is the choice of the function T that transforms $\text{supp}(p)$ into $\mathbb{R}^K$. In Stan, any Bayesian model is implicitly associated with a transformation function. The library of transformation functions covers a range of different support sets, e.g. real number vectors with lower and upper bounds, ordering or simplex constraints. Using these, Stan automatically transforms the parameters of every model into the unconstrained real coordinate space before applying ADVI, HMC or NUTS (see [3], Section 10).
To the best of our knowledge, TensorFlow Probability does not automatically transform the parameter space into the unconstrained space. However, it provides a rich library of bijector objects (`tfp.bijectors`) with all functionalities we need for defining, composing and applying transformation functions. Fortunately, the Stan manual [3] describes what transformation is used in different cases. This allowed us to manually reproduce the transformation functions of Stan in TensorFlow and run ADVI, HMC and NUTS on the unconstrained parameter space as done in the paper [1]. However, with our resources, we cannot reproduce ADVI to automatically chose T, since this is a feature provided by Stan rather than something which was introduced and outlined in the paper.

Difficulty 2 – Gradient Descent Algorithm

In the paper [1], a modified version of adaGrad [8] is used: instead of regularizing by the $\ell 2$-norm of all previous gradients, only gradients of the past 10

iterations are taken into account. While this breaks the convergence properties of adaGrad, it makes training much faster, especially in the case of ADVI where the initial state can be far from the true posterior.
Since this algorithm is not implemented in TensorFlow, we first started experimenting with the original adaGrad algorithm. Since convergence was slow, we then switched to Adam [9]. Adam combines both the speed of RMSprop [10] and (given a sensible choice of hyperparameters) convergence guarantees of adaGrad [11].

Difficulty 3 – Termination of ADVI

According to Algorithm 1 in the paper [1], ADVI stops optimising when the change in the ELBO is below a threshold ϵ, without further specifying ϵ. In general, a good choice of ϵ depends on the problem and the variance induced by MC integration (see equation (1) in Section 2), which is higher for a smaller number of samples. As this hyper-parameter does not have significant effects in our experimental setup, we set $\epsilon = 0.01$ per default. Furthermore, we decided to use the same number of samples M for computing the ELBO as for the gradients.

Difficulty 4 – HMC and NUTS

While we attempted to match Stan's implementation of HMC and NUTS as closely as possible, there are some important differences that deserve highlighting: Stan uses the aforementioned primal-dual averaging algorithm to automatically adjust step sizes during HMC and NUTS. In contrast, we have used a much more simple step size adaption scheme (equation 19 from [12]). Additionally, Stan by default initializes latent variables in unconstrained space with values drawn from the uniform distribution on the interval $[-2, 2]$. In TensorFlow, however, the initial state is required as an input to all MCMC-algorithms, and has to be defined by the user. We find that the performance of HMC and NUTS heavily depends on the initial state (see Section 4). Therefore, instead of mimicking the automatic initialisation of Stan, we experiment with different initialisation techniques, including sampling from the model priors, taking their means or using the μ-values from the initial variational ADVI distribution.

Required Resources

We roughly estimate that building the code base needed to perform our experiments took us a total of 100 programming hours. This includes implementing ADVI, the logging and plotting, the Bayesian models and the tests.

4 Experiments

In this section, we discuss the experiments we conducted, both from the paper [1] and our own extension. We start with some general considerations, followed by detailed descriptions of the individual experiments.

4.1 General Considerations

Recall from Section 2 that 5 experiments with different probabilistic models and datasets were described in the paper, and that the performance measure for the inference methods is the average log predictive on held-out data. The goal of the authors is to compare the performance of ADVI to the performance of HMC and NUTS by means of accuracy and speed – the larger the average log predictive, the higher the accuracy.

Choosing Experiments

Due to time constraints, we were not able to reproduce all five experiments conducted. In Section 3.3 of the paper, the authors develop an experiment to evaluate the performance of ADVI on a large data set, i.e. 250,000 images. For reproducing this, we would have had to implement stochastic minibatch subsampling to our ADVI implementation, which is not required for the other experiments. Since we suspected that our computational resources might not suffice to execute this experiment anyway, we focused on reproducing the four experiments from Section 3.1 and 3.2 of [1] (see Section 4.2 and 4.3) as well as one additional experiment to illustrate the importance of the transformation function T and state initialisation in ADVI (see Section 4.4).

Discussion of the Experimental Setup

Before diving into the individual experiments, we critically assess the goals and the design of the experiments conducted in the paper.

- **Computational resources**. Even though ADVI is compared to HMC and NUTS by depicting their performance measure over time, the authors do not reveal what computational resources they used. This makes it difficult to evaluate if the performance differences in our implementation are due to inefficient implementations in our code or computational power. More specifically, despite our ADVI showing similar behaviour as in the paper [1], in all experiments, it is consistently slower than reported. We assume that this is caused by having access to less computational power, but we cannot be certain. All our experiments were performed on our laptops, i.e. MacBook Pro, Intel Core i5/i7 processors (1 x 2.3 - 4 x 2.8 GHz), 4-16 GB DDR3 RAM.

- **Performance measure**. Recall that for measuring the performance of HMC, NUTS and ADVI, one sample per step is drawn, and all samples drawn until step t are used to estimate the predictive likelihood. In practice, however, this is not how ADVI is used. Instead, a large number of samples would be drawn from the final variational distribution after termination of ADVI. As a consequence, samples drawn from ADVI before termination should not matter. However, they heavily affect the performance measure. We conclude that the experimental setup does not properly capture the accuracy of ADVI, but acknowledge that VI and MCMC methods are difficult to compare in general.

- **Beyond accuracy and speed**. A downside of MCMC methods is that they tend to focus on one mode in multi-modal posterior distributions (see e.g. [13]). Variational inference methods might mitigate this issue. Thus, it could be interesting to explore whether or not ADVI can express multi-modal posteriors better than HMC and NUTS. The performance measure used in the paper does not capture this.

4.2 Hierarchical Regression Models

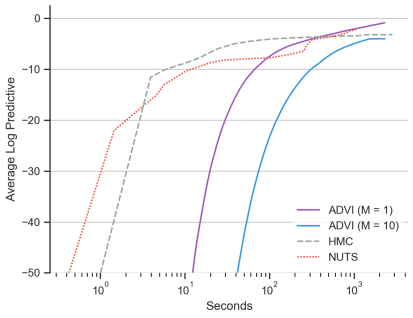
To show that ADVI is at least as fast as the baseline methods in basic settings, the authors investigate two rather simple regression models (see Appendix F and G of [1] for details).

Linear Regression with ARD

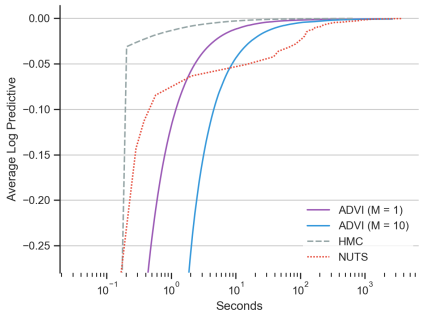
Linear regression with Automatic Relevance Determination was performed on generated data. The likelihood of each prediction $\mathbf{y}$ is assumed to be normally distributed around mean $\mathbf{w}^\intercal\mathbf{x}$ with unknown variance τ^{-1}. The prior for w is normally distributed around 0 with variance $\tau\text{diag}[a]^{-1}$. In short, the priors of the model parameters are hierarchical, as well as each parameter having a different prior (see Appendix F of [1] for details). The likelihood is maximised by finding parameters w, τ and α. The model has 250 parameters, half of which were set to 0, as specified in the paper [1]. The way in which the data was generated was not clear in the paper. We decided to generate the parameters in the model by sampling from a normal distribution, then generating values of $\mathbf{x}$ and $\mathbf{y}$ from these parameters by calculating the dot product of the parameters with $\mathbf{x}$ to obtain $\mathbf{y}$. Because of this ambiguity, μ and σ of the distribution of the parameters was varied to see if it would affect the performance of the different methods. When σ was 1, HMC and NUTS outperformed ADVI. When σ was greater than 10, HMC and NUTS failed to converge within a time frame of 20 minutes, and ADVI out performed the sampling methods. This was seen to be the case even when varying the step size of HMC from 0.001 to 1. Changing the mean of the generated parameters did not influence the performance of the methods.
HMC and NUTS made progress when the initial starting state was the mean of the priors. On occasion, when randomly sampled from the prior, the sampling methods converged, but the majority of the time, successive chain steps did not increase the average log predictive with random initialisation. For optimum performance, HMC and NUTS where therefore initialised on the mean of the priors. The experiment seen on Figure 1a shows results of an experiment with data generated using $\sigma = 1$ and $\mu = 0$. HMC and NUTS initialised on the mean of the priors clearly out performs ADVI in terms of convergence speed. The conclusion is that the authors of the paper did not investigate different initial starting states of HMC and NUTS and failed to recognise that there their initial state was not the optimum.

Hierarchical Logistic Regression

Our second experiment was a Logistic Regression model predicting the likelihood of a person voting Republican in the 1988 US presidential election based on properties like gender, age or income obtained from multiple sur-



(a) Linear Regression with ARD



(b) Hierarchical Logistic Regression

Figure 1: Hierarchical Generalised Linear Model Experiments.

veys. Both the model and data for this came from [14]. Since the model is non-conjugate, it is an example of family of models where traditional mean-field methods for variational inference cannot be directly applied and model specific algorithms were previously required for variational inference [15].
We ran the algorithm using $M = 1$ and $M = 10$ samples to calculate gradients and check convergence of our variational model. When comparing it to the results presented in the paper [1], we noticed our model was performing vastly better (see Figure 1b). After further investigation of the model's implementation in Stan (Figure 7 in [1]), we noticed that the model implemented in Stan differs from the model description in the paper and the original book [14]. This makes it very hard to compare our results with the results achieved in the paper. But since this seems to be a mistake made by the authors, we have decided not to change our model to reflect the changes made in the Stan implementation but keep the model described in the paper.

4.3 Non-negative Matrix Factorisation Models

The paper further explores two non-negative matrix factorisation models. The models take images from the Frey Faces data set, which includes 1965 images of faces with dimensions 20 x 28 taken from frames in a video. In the Constrained Gamma Poisson Model, the priors are gamma distributions, whereas for the Dirichlet Exponential Model, the priors are exponential and Dirichlet distributions. A non-conjugate Poisson distribution is used as likelihood (see Appendix H and I of [1] for details). The goal of both experiments

is to show that ADVI enables the exploration of new models for which HMC and NUTS struggle to find good results.

Matrix Factorisation with Constrained Gamma Poisson model

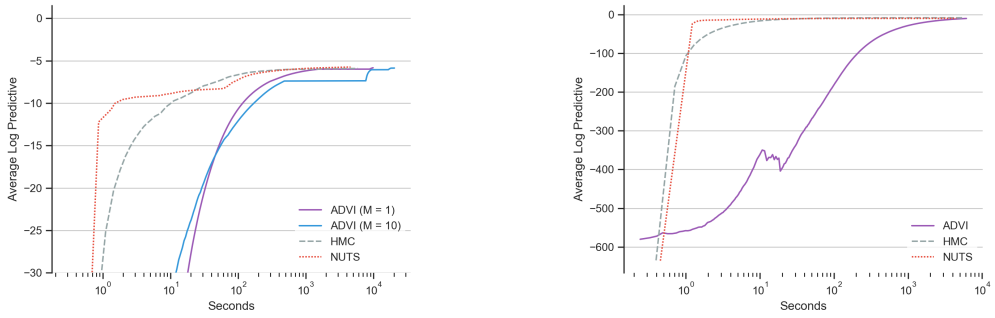
While for the other experiments we relied on Stan's reference manual [3] for the right choice of transformation function, it lists no mapping between positive ordered vectors and multidimensional real space. But since it contained mappings between ordered vectors and real space, and positive real space to real space, we decided to use function composition to create the mapping required. But since parameters μ and ω are set to zero initially, the choice of mapping determines the distribution of the initial state. The inverse of the mapping we created used a double exponential, which both caused numerical instabilities and poor initialisation of the initial state.
We then tried to come up with our own transformation. For simplicity, we describe the inverse mapping T^{-1} from $\mathbb{R}^K$ to K-dimensional positive ordered real space. Let $\mathbf{x} = T^{-1}(\mathbf{y})$, then

$$\begin{aligned}\mathbf{x}_1 &= \exp(\mathbf{y}_1)\\ \mathbf{x}_i &= \exp(\mathbf{y}_i) + \mathbf{x}_{i-1}\end{aligned}$$

With this new mapping only a single exponential is computed in the inverse transformation, making the training more stable and leading to better initial states.
Since the paper didn't state how the data set was split into training and test set, we decided to hold out 20% of the data as test data. The results of this experiment can be seen in Figure 2a using $M = 1$ and $M = 10$ samples to estimate the ELBO. The results are unexpected as the authors claim that HMC does not converge on this model. We found in our experiment that NUTS outperforms HMC and HMC outperforms ADVI. We attribute this to the authors not exploring the range of possibilities for initialisation of HMC. Similarily to other experiments, initialising HMC and NUTS on the mean of the priors results in superior performance than what is seen in the paper.

Matrix Factorisation with Dirichlet Exponential Model

For reproducing this experiment, we set $M = 1$ and held out 20% of the data set as test data. The data split was not mentioned in the paper. Figure 2b illustrates the performance of our ADVI implementation. Apart from being



(a) Constrained Gamma Poisson (b) Dirichlet Exponential

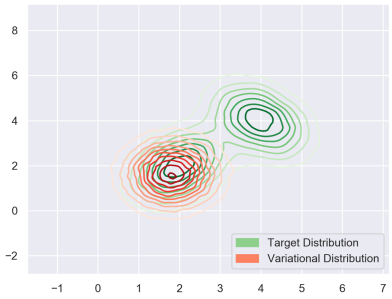
Figure 2: Non-negative Matrix Factorisation Model Experiments.

slower than reported (see discussion in Section 4.1), ADVI showed roughly the same behaviour as in the paper. We assume that the volatility between 10 seconds and 40 which we consistently observed for different parameter settings, was cut out in [1] Figure 5b. Our ADVI terminated after about 75 minutes.

For the baseline methods, we observed a different behaviour than in the paper. Firstly, we initialised HMC and NUTS by sampling from the prior distributions. Both did not find sensible results, exactly as reported in [1]. The same held for sampling uniformly from $[-2, 2]$ in the unconstrained parameter space as done in Stan. However, this dramatically changed after experimenting with the initialisation method. Figure 2b shows the performance of HMC and NUTS if initialised to the μ-values of the initial variational ADVI distribution, i.e. to $\mathbf{0}$ in the unconstrained space or $T^{-1}(\mathbf{0})$ in the constrained space (see Section 2). Both HMC and NUTS achieved good results in significantly less time than ADVI. This held true even when the initial Dirichlet parameters were sampled from the prior and the exponential parameters were (randomly) initialised close to the mean of the exponential prior. We conclude that the baselines used in the paper were not sufficiently optimised.

4.4 Extension: Importance of Choice of Transformation

Stan's Reference Manual includes a guide to the mappings used between $\text{supp}(p)$ and $\mathbb{R}^K$ (see Section 3). However there are many possible choices



(a) ADVI with Identity Transformation

(b) ADVI with shifted Identity Transformation

Figure 3: Density plots for Mixed Gaussian Experiments.

of mappings. There exists an infinite number of bijections $f : \mathbb{R}^K \to \mathbb{R}^K$ that, given a mapping T from $\text{supp}(p)$ to $\mathbb{R}^K$, we can use to create a new transformation $f \circ T$.

We therefore asked ourselves how the choice of transformation influences the performance of ADVI. While we have already seen the Gamma Poisson model fail for our initial choice of transformation, this might have also been affected by numerical instabilities.

Conveniently, for all models presented in the paper [1], the mean of the initial variational distribution in ADVI matches the prior mean of the latent variables. So what happens if this is not the case? While gradient descent can guarantee to converge at an optimum, that optimum is not necessarily global. As a result, the choosing the right starting point can be crucial in finding the global optimum. We tested our hypothesis with a simple two-dimensional Gaussian Mixture model that we tried to approximate with a single Gaussian. In Figure 3 we present two density plots for ADVI with different transformations. Figure 3a shows ADVI's performance when using the identity bijection as suggested by the paper, while Figure 3b shows ADVI's performance when the same mapping was shifted so that the initial state lies at the centre of our target distribution. Note that this does not change gradients but only affects the initial state of ADVI. Our results seem to suggest that the choice of transformation is extremely important: Clearly, when using the identity transformation ADVI fails to converge close to the target distributions mean, while the shifted mapping seems to capture the

target distribution much better.

5 Conclusion

We were successful in implementing ADVI and replicating four of the five experiments from the paper [1] using TensorFlow Probability. Throughout the implementation process, we faced some difficulties (see Section 3). Most of them stemmed from switching the code base from Stan to TensorFlow Probability, as many Stan-specific aspects are crucial to ADVI, for example the automatic choice of the transformation function. Some difficulties however, were due to information not being available such as, when exactly ADVI terminates. In general, we experienced that reproducing the paper required us to source a significant amount of resources from outside the paper, and a longer time investment in to the project than what was initially planned.
In the four experiments we reproduced from the paper, the curves for the average log predictive of ADVI resembles the curves in the paper. However we cannot confirm that ADVI performs better than the baseline methods. On the contrary, with optimised hyper-parameters, we found that both HMC and NUTS performed vastly better than the respective algorithms from the paper. HMC outperformed ADVI in all of the experiments and NUTS in three out of the four experiments. This is especially surprising for the performance of HMC. The authors reported that HMC did not result in any convergence for experiments involving the non-negative matrix factorisation models and therefore omitted these results entirely from their graphs. Consequently, we believe that the baselines experiments conducted in the paper were weak, and the authors did not explore all the possibilities of initialisation states to optimise HMC and NUTS.
The hardware used for the experiment was not specified in the paper, therefore we assumed that the results obtained would be universal across any device. However, it could be that the authors' use of a GPU in their experiments resulted in a speed up of ADVI disproportionately more than that of HMC and NUTS, resulting in some of the discrepancy between our results and theirs. ADVI which uses automatic differentiation is more amenable to parallel processing. Whereas the sampling methods HMC and NUTS in their sequential nature does not lend itself well to added speed improvements as a result of parallel processing, unless when using multiple Markov chains, which we assumed is not the case as this was not specified. Therefore with

GPU support our results could have shown ADVI to outperform these sampling methods, as shown in the paper, even with the added speed up we introduced to HMC and NUTS with optimal initialisation. If this were to be the case, the authors would have proven not that ADVI is a tool for faster inference, but a tool which leverages GPU acceleration better than other inference methods.

The accounts of the experiments in the paper contained several flaws. Some crucial information was not available, for example how to generate the data for the Linear Regression with ARD experiment (see Section 4.2) or which transformation function to use in the Constrained Gamma Poisson experiment (see Section 4.3). Furthermore, the Hierarchical Logistic Regression experiment contained a mistake that changed the result significantly (see Section 4.2). Additionally, the performance measure used for evaluating the inference methods does not fully capture the actual accuracy of ADVI.

We also noticed that some potential benefits of ADVI over MCMC methods were not investigated in the paper, this includes a higher robustness to changes in hyper-parameters or a more accurate description of highly multi-modal posterior distributions.

As an extension to the paper, we analysed the impact of the transformation function T on the performance of ADVI. In a simple experimental setting, we showed how T determines the initial state of ADVI, which heavily affects how close the final variational distribution is to the actual posterior (see Section 4.4). We also observed this in the Constrained Gamma Poisson model, where our initial choice of T included a double exponential causing our initialisation to be off, so that ADVI did not converge. Hence, suggest the following change to ADVI: Initialise μ such that $T^{-1}(\mu)$ agrees with the model's prior mean. This mitigates the need to pick a good transformation function, since the initial parameters will be independent of the transformation (other than potential numerical issues).

Given more time, we would have tried to perform the experiments with more computational power, particularly with GPUs, in order to evaluate the hypothesis of GPU usage accelerating ADVI more than MCMC methods. In addition, we would have like to implemented stochastic mini-batch optimisation for ADVI and tried to reproduce the experiment from Section 3.3. of the paper. Building on our results, we would like to further explore the properties of ADVI, particularly its expressiveness given different transformation functions.

References

[1] Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David Blei. Automatic variational inference in stan. In *Advances in neural information processing systems*, pages 568–576, 2015.

[2] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

[3] Stan Development Team. Stan modeling language users guide and reference manual, version 2.22, 2020.

[4] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216 – 222, 1987.

[5] RM Neal. Mcmc using hamiltonian dynamics. *arXiv preprint arXiv:1206.1901*, 2012.

[6] Matthew D Hoffman and Andrew Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[7] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[11] Sashank Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[12] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and computing*, 18(4):343–373, 2008.

[13] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[14] Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

[15] Chong Wang and David M. Blei. Variational inference in nonconjugate models. *J. Mach. Learn. Res.*, 14(1):1005–1031, April 2013.