Automatic Differentiation Variational Inference

Efe Acer [*]
efe.acer@cs.ox.ac.uk

Lorenz Kuhn
lorenz.kuhn@cs.ox.ac.uk

Jan Stratmann
jan.stratmann@cs.ox.ac.uk

Patrick Zoechbauer
patrick.zoechbauer@cs.ox.ac.uk

Abstract

In this project we reproduce the results of *Automatic Differentiation Variational Inference* by Kucukelbir et al. [2015] in Pyro [Bingham et al., 2018]. We are largely able to reproduce the experimental results presented in the paper with minor exceptions. Additionally, we study potential shortcomings of ADVI in two experimental settings that go beyond the original paper. First, we show that ADVI seems slightly more robust to model misspecification than NUTS. Second, we suggest that ADVI performs well under data subsampling even if it is compared to strong stochastic MCMC methods and not just naive stochastic NUTS as done in the original paper. Our code is publicly available on GitHub[2]. In general, we agree with the authors' claims that ADVI provides a powerful alternative to MCMC algorithms, in particular by being generally easier to use and less sensitive to various hyper-parameter choices.

1 Introduction

In this section, we briefly explain our approach to this reproducibility challenge and provide an overview over the rest of this report.

The original paper that builds the foundation for this review was published by Kucukelbir et al. [2015]. It introduced and evaluated "Automatic Differentiation Variational Inference in Stan". The paper features a theoretical derivation of the ADVI algorithm and demonstrates its performance by applying the ADVI algorithm to a set of Bayesian inference problems from a variety of data domain.

Given the scope of this project we were presented with a choice between two different approaches to reproducing results from this paper. One option was to focus on a low-level implementation of ADVI and evaluate it on one or two simple examples. The other option was to implement ADVI using higher-level primitives and then focus on evaluating the method across a wider range of examples and to investigate some aspects of ADVI beyond those discussed in Kucukelbir et al. [2015]. Based on our interests, we decided to pursue the second approach.

We therefore begin this report by briefly reviewing the theory behind ADVI in section 2. In section 3, we reproduce the experiments done by Kucukelbir et al. [2015] in Pyro, a competing probabilistic programming framework originally introduced by Bingham et al. [2018]. This includes case studies on nonconjugate regression models (see section 3.1) and nonnegative matrix factorizations of video frames (see section 3.2). We omit the experiment focusing on the imageCLEF dataset and instead investigate possible extensions of the original paper.

In section 4, we subsequently extend our analysis of the ADVI method using two additional experiments that go beyond the original paper's contents: First, we use a Gaussian Mixture Model (GMM)

[*]group 17; names in alphabetical order;
[2]`https://github.com/lorenzkuhn/advi`

to showcase inherent strengths and weaknesses of ADVI compared to NUTS. Specifically, we show that if the number of components K is misspecified, ADVI does not achieve the same performance as MCMC. Second, we further detail the topic of subsampling. Subsampling is needed for large datasets where computing gradients over the whole dataset becomes prohibitively expensive. To this end, we compare ADVI with more advanced HMC sub-sampling techniques (e.g., SGHMC) rather than NUTS or naive sub-sampling, as was done in the original paper.

2 Automatic Differentiation Variational Inference

In this section, we briefly review the motivation for and the key ideas of Automatic Differentiation Variational Inference (ADVI).

Variational inference is one approach to approximate posteriors of probabilistic models. A probabilistic model defines a joint distribution over observations $\mathbf{x}$ and latent variables θ. Given a prior over the latent variables $p(\theta)$ and some observations $\mathbf{x}$, one might want to calculate the posterior over the latent variables given these observations: $p(\theta \mid \mathbf{x})$. In general, there is no analytical solution for $p(\theta \mid \mathbf{x})$ and the posterior has to be approximated. Markov Chain Monte Carlo (MCMC) methods and variational inference are the two main classes of approximate inference methods.

The central idea of variational inference is to formulate the posterior approximation as an optimization problem. That is, a family of variational distributions is defined and an optimisation problem is solved to find the member of this family that minimises the KL divergence to the true posterior distribution. More formally, for a variational family $q(\theta; \phi)$ parameterised by a vector $\phi \in \Phi$, VI minimises

$$\phi = \arg\min_{\phi \in \Phi} \mathrm{KL}(q(\theta; \phi) || p(\theta | \mathbf{x}))$$

Since this term contains the intractable posterior distribution $p(\theta|\mathbf{x})$, it cannot be minimized directly. Instead, it can be shown that minimizing the objective above is equivalent to minimizing the Evidence Lower Bound (ELBO) which we can compute:

$$\mathrm{ELBO}(\phi) = \mathbb{E}_{q(\theta)}[\log p(x, \theta)] - \mathbb{E}_{q(\theta)}[\log q(\theta; \phi)]$$

One constraint with this approach is that $\mathrm{supp}(q(\theta; \phi)) \subseteq \mathrm{supp}(p(\theta|\mathbf{x}))$ since the KL divergence between the variational distribution and the posterior would otherwise always be infinite. Without loss of generality, it can be assumed that $\mathrm{supp}(p(\theta)) = \mathrm{supp}(p(\theta|\mathbf{x}))$. Thus, if one wants to use variational inference for a given probabilistic model, the variational family will have to be hand picked to have the same support as the prior distribution of the model.

The aim of ADVI is to offer a way of applying variational inference to a given probabilistic model without having to think about this support matching constraint. Put simply, ADVI achieves this by bijecitvely mapping the original latent variable space to the real coordinate space and then performing variational inference in the transformed space.

Let's explore this in detail: for a one-to-one differentiable function $T : \mathrm{supp}(p(\theta)) \to \mathbb{R}^K$, the transformed joint density $p(x, \zeta)$ has the representation $p(x, \zeta) = p(x, T^{-1}(\zeta))|\mathrm{det}J_{T^{-1}}(\zeta)|$. Intuitively, the Jacobian here describes how the transformation warps unit volumes and thus ensures that the integral of the transformed distribution equals 1.

The ELBO in real coordinate space then takes the form:

$$\mathrm{ELBO}(\phi) = \mathbb{E}_{q(\zeta;\phi)}[\log p(x, T^{-1}(\zeta))) + \log |\mathrm{det}J_{t^{-1}}(\zeta)|] - \mathbb{H}[q(\zeta; \phi)]$$

For example, assume that $p(\theta) = \mathrm{Gamma}(1, 1)$, i.e., $\mathrm{supp}(p(\theta)) = \mathbb{R}_{>0}$. Then ADVI uses the transformation $T : \mathbb{R}_{>0} \to \mathbb{R}$ given by $\zeta = T(\theta) = \log(\theta)$ to map the constrained posterior space onto the real coordinate space.

The authors then choose $q(\zeta; \phi)$ to be a Gaussian distribution with mean $\boldsymbol{\mu}$ and variance $\mathrm{diag}(\boldsymbol{\sigma})$. However, the ELBO involves an expectation over $q(\zeta; \phi)$ which depends on $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Unfortunately, this leads to an intractable integral and hence automatic differentiation cannot be applied to this term directly. Instead, an additional transformation is applied to transform the variational distribution into a standard Gaussian. This second transformation is given by $S : \mathbb{R}^K \to \mathbb{R}^K$ is given by

$$S(\boldsymbol{\zeta}) = \mathrm{diag}(\boldsymbol{\sigma})^{-1} (\boldsymbol{\zeta} - \boldsymbol{\mu}) .$$

By doing so the expectation in the ELBO is computed with respect to a standard Gaussian rather than the original variational distribution. This means that automatic differentiation can be applied to the expression inside of the expectation. The expectation is then approximated via Monte Carlo sampling from the standard Gaussian.

Figure 1 visualizes the different transformation applied in the ADVI algorithm.

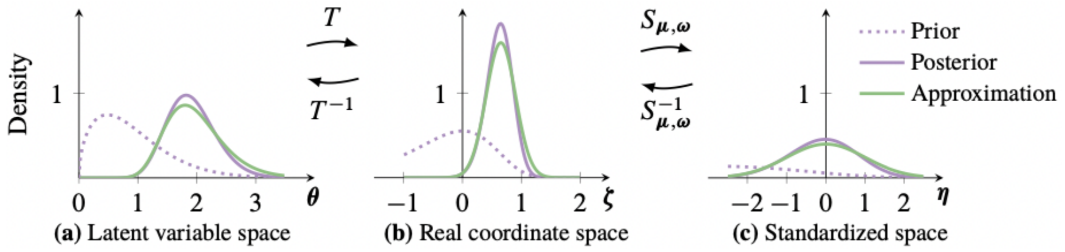


Figure 1: Visualization shown in [Kucukelbir et al., 2015] to demonstrate relevance of transformation in ADVI.

Putting everything together, the final ELBO takes the following form:

$$\text{ELBO}(\phi) = \mathbb{E}_{\mathcal{N}((\eta;0,I)}[\log p(x, T^{-1}(S^{-1}(\eta)))) + \log|\det J_{t^{-1}}(S^{-1}(\eta))|] - \mathbb{H}[q(\zeta;\phi)]$$

This ELBO is then optimised using stochastic gradient ascent with an adaptive step size sequence which is an extension of RMSPROP [Tieleman et al., 2012].

3 Reproducing Automatic Differentiation Variational Inference

Since the experimental results presented by Kucukelbir et al. [2015] were obtained using the authors' own probabilistic programming language Stan (see [Stan Development Team, 2018a]), we opted to verify the results using similar functionality offered in Pyro (see [Bingham et al., 2018]), a probabilistic programming language built on top of Pytorch (see [Paszke et al., 2019]).

When presenting their results, the original paper's authors make a point of comparing ADVI to conventional MCMC sampling methods, namely HMC and NUTS, an extension of HMC. Throughout this review, we thus also compare ADVI to these MCMC methods.

The authors of [Kucukelbir et al., 2015] evaluated their newly proposed ADVI algorithm through multiple empirical experiments. The remainder of this section will focus on reproducing these results. As the paper does not specify the hardware used to achieve the results, we cannot meaningfully compare raw times or make any assumptions on effective parallelization. For sake of completeness, we want to note that we are computing on two Intel Xeon cores clocked at 2.2GHz via Google Colab with access to 13 GB of RAM. Given more time, we would recommend rerunning the experiments in Stan on the same hardware to get even more efficiency insights, too.

3.1 Nonconjugate regression models

The first two examples that were studied in [Kucukelbir et al., 2015] concern two non-conjugate regression models: (1) Linear regression with automatic relevance detection (ARD) (see [Drugowitsch, 2013]) and (2) Logistic regression with spatial hierarchical priors (see [Gelman and Hill, 2007]).

3.1.1 Automatic relevance detection

In linear regression, it is commonly assumed that the outcome y is sampled from a normal distribution. The mean is given by the linear function $\boldsymbol{w}^T\boldsymbol{x}$, where $\boldsymbol{w}$ is an unknown weight vector and $\boldsymbol{x}$ is a vector of feature variables associated with y. For multiple independent observations $\boldsymbol{y} = (y_1, ..., y_n)$, it therefore holds that

$$p(\boldsymbol{y} \mid X, \boldsymbol{w}, \sigma) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid \boldsymbol{w}^T\boldsymbol{x}_n, \sigma),$$

where σ denotes the standard deviation.

To enable automatic relevance detection, this model is extended such that each weight w_k is associated with a shrinkage prior, i.e.,

$$p(\boldsymbol{w} \mid \boldsymbol{\alpha}, \sigma) = \prod_{k=1}^{D} \mathcal{N}\left(w_k \mid 0, \frac{\sigma}{\sqrt{\alpha_k}}\right)$$

with

$$p(\boldsymbol{\alpha}) = \prod_{k=1}^{D} \text{Gam}(\alpha_k \mid 1, 1) \quad \text{and} \quad p(\sigma) = \text{InvGam}(\sigma \mid 1, 1).$$

Note that α_k acts as a shrinkage parameter for each w_k, which sees the prior on w_k become narrower around 0 (and hence more informative) as α_k becomes very large.

To test the posterior distribution inference for this model in an experiment, we have implemented the data generating process according to [Kucukelbir et al., 2015]. This process is given by

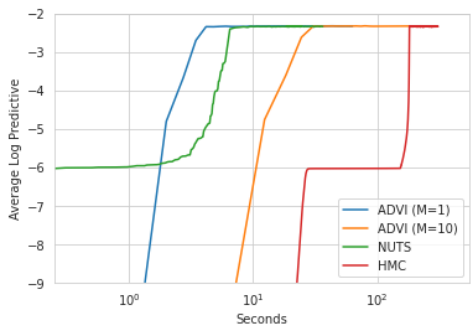
$$\begin{aligned}
x_k^i &\sim \mathcal{N}(0, 1) \quad \text{for} \quad i = 1, ..., 11'000 \quad k = 1, ..., 250 \\
w_k &\sim \mathcal{N}(0, 25) \quad \text{for} \quad k = 1, ..., 125 \quad \text{and} \quad w_k = 0 \quad \text{for} \quad k = 126, ..., 250 \\
\varepsilon_i &\sim \mathcal{N}(0, 5) \quad \text{for} \quad i = 1, ..., 11'000 \\
y_i &= \boldsymbol{w}^T \boldsymbol{x}_i + \varepsilon
\end{aligned}$$

This yields a data set containing 250 simulated features, of which only half are used to generate the labels y_i. For the experiments, we simulated 10'000 training data points and 1'000 testing data points. Analogously to [Kucukelbir et al., 2015], we report and study two different settings for ADVI by varying the number of samples drawn to get a Monte Carlo estimate for the gradients.

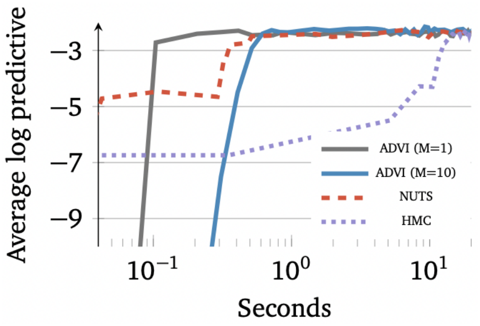
Figure 2a shows the performance achieved by ADVI as well as MCMC algorithms based on our implementation in Pyro. In contrast, Figure 2b shows the results as reported in the original paper [Kucukelbir et al., 2015]. Our replication results confirm the observations made by Kucukelbir et al. [2015], namely that using a single sample is sufficient for the inference algorithm to converge to an optimum.

In comparison, Figures 2a and 2b also show that the replication in Pyro delivers very similar results to those reported in the original paper. All methods converge to the same performance, although the convergence is slightly slower in our implementation. This could, however, be explained by the choice of underlying computing infrastructure or the tuning parameter of the inference algorithms. We note that the paper does not report many details with respect to those parameters.

Further, we have observed that the runtime of HMC is very sensitive to the tuning parameters *step_size* and *number_of_steps* (i.e., the number of discrete steps over which to simulate Hamiltonian dynamics).



(a) Replication of results using pyro



(b) Copy of figure from original ADVI paper.

Figure 2: Replication analysis for linear regression with automatic relevance determination. For HMC, we set *step_size* to 0.1 and *number_of_steps* to 10.

3.1.2 Logistic regression with spatial hierarchical prior

The second regression example discussed in the ADVI paper concerns hierarchical logistic regression applied to real-life data from political sciences.

The example aims to model the impact that the state of a voter has on their voting preference in the US presidential election. Formally, let α_j, for $j = 1, ..., 51$ (the District of Columbia is treated as a state in this model) be the state-level effect. Then a simple linear model for the probability that voter i supports the Republican candidate is given by

$$\mathrm{P}(y_i = 1) = \mathrm{logit}^{-1}(\alpha_{j(i)} + \beta^T x_i), \text{ for } i = 1, ..., n,$$

where $j(i)$ is the mapping that provides the state j of voter i, x_i denotes a set of additional features, and y_i denotes the binary variable indicating party preference.

In the example discussed in the ADVI paper, the data includes features for age, sex, ethnicity, and education. Instead of adding those variable only as linear terms, the model trained by Kucukelbir et al. [2015] includes interaction terms for gender $\times$ ethnicity as well as varying intercepts for age, education and the interaction age $\times$ education. This results in an updated model equation given by

$$\begin{aligned}\mathrm{logit}\left(\mathrm{P}(y_i = 1)\right) = \beta_0 &+ \beta^{\text{female}}\text{female}_i + \beta^{\text{black}}\text{black}_i + \beta^{\text{female}\times\text{black}}\text{female}\times\text{black}_i \\ &+ \alpha^{\text{age}}_{k(i)} + \alpha^{\text{edu}}_{l(i)} + \alpha^{\text{age}\times\text{edu}}_{k(i),l(i)} + \alpha^{\text{state}}_{j(i)}, \text{ for } i = 1, ..., n,\end{aligned}$$

with the following prior distributions on the model parameters

$$\begin{aligned}\alpha^{\text{age}}_k &\sim \mathcal{N}(0, \sigma_{\text{age}}) \\ \alpha^{\text{edu}}_l &\sim \mathcal{N}(0, \sigma_{\text{edu}}) \\ \alpha^{\text{age}\times\text{edu}}_{k,l} &\sim \mathcal{N}(0, \sigma_{\text{age}\times\text{edu}}) \\ \alpha^{\text{region}}_m &\sim \mathcal{N}(0, \sigma_{\text{region}}) \\ \alpha^{\text{state}}_j &\sim \mathcal{N}(\alpha^{\text{region}}_{m(j)} + \beta^{\text{prev\_vote}}\text{prev\_vote}, \sigma_{\text{state}})\end{aligned}$$

Note that the model imposes a hierarchical prior on the individual state effect, i.e., the mean of the individual state effect depends on the region to which the state belong as well as the previous voting behavior.

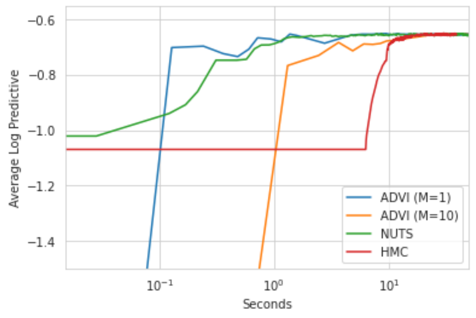
Figure 3 show the model performance on a dataset from the United States 1988 presidential election. It consists of 11'536 data points from an opinion poll gathered by media organizations before the election. Figure 3a show the results obtained through the replication in Pyro. As in the original paper, the models were trained on 10'000 data points and the remaining 1'536 were used for testing. Unfortunately, the original ADVI paper does not provide any additional details regarding the train/test split, hence we decided to split randomly.

Comparing both Figures 3a and 3b, we observe similar results between the original paper and our replication:

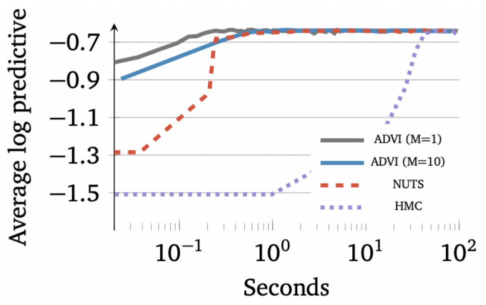
1. All four inference algorithms eventually converge to the same average log predictive accuracy.
2. MCMC with NUTS sampler and ADVI with $M = 1$ converge fastest to the optima.
3. For the MCMC methods, the NUTS sampler converges faster to the optima than using HMC.

However, we also observe some small differences between Figures 3a and 3b. These differences occur mainly for ADVI during the early phase of the training. In contrast to the original paper, our replication of ADVI achieves significantly worse performance during the first few iterations compared to the results reported in [Kucukelbir et al., 2015].

This could be explained by the approach used to initialize the model parameters. Therefore, we implemented the same approach as described by the authors of using draws from a standard normal for parameter initialization. However, we could not replicate the strong performance of ADVI in the first 10^{-1} seconds. Nevertheless, both ADVI methods in our replication achieve the optima within 1 second.



(a) Replication of results using pyro



(b) Copy of figure from original ADVI paper.

Figure 3: Replication analysis for logistic regression with spatial hierarchical prior.

3.2 Non-negative matrix factorization

To further demonstrate ADVI's potential in nonconjugate models, Kucukelbir et al. [2015] call on the Frey Face dataset ([Frey and Roweis, 2008]). This dataset contains 1956 frames that make up a short video sequence showing Brendan Frey's moving head in 28×20 resolution.

The authors' goal in this example is to fit two types of non-negative matrix factorization models to the frames. This is supposed to showcase Stan's built-in competence in applying ADVI to constrained latent variable spaces. For both models, Kucukelbir et al. [2015] measure success through the average log predictive accuracy per reconstructed pixel.

We remark that the choice of this dataset and respective models to demonstrate ADVI's abilities is rather unconventional. Researching the Gamma Poision model (one of the models that was used by the authors for this task), we find that it was originally proposed by Canny [2004] to store document-term data and similar discrete information. Alternatively, matrix factorization models are also commonly used in recommender systems [Koren et al., 2009], which could have served as a more natural choice of dataset, providing many performance baselines available to compare ADVI with. Nevertheless, we proceed to reproduce implement this model in Pyro in order to be able to compare our results with the original paper.

3.2.1 Constrained Gamma Poisson model

For the first of the two models the following generative process is assumed to produce a sample frame $Y \in \mathbb{N}^{U \times I}$:

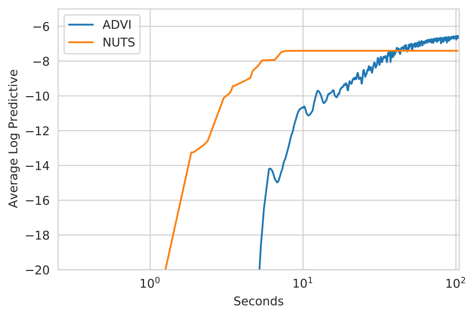
1. Draw i.i.d values from Gam(1,1) to fill $\theta \in \mathbb{N}^{K \times U}$ and $\beta \in \mathbb{N}^{K \times I}$.
2. Draw $Y \sim$ Poisson($\theta^\top \beta$), where the distribution is applied element-wise to each pixel.

This setup effectively learns $K = 10$ latent 'factors', each consisting of two vectors of size $I = 20$ and $U = 28$, respectively.

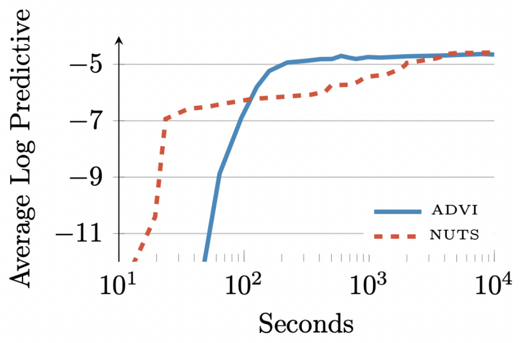
The constraints relevant to ADVI arising from this model are mainly that all values in θ and β are limited to the positive real axis. To ensure uniqueness of each factor, the authors impose an additional ordering constraint along the first dimension of θ (else scaling either parameter by $c \in \mathcal{R}$ and the other by $\frac{1}{c}$ yields indistinguishable solutions).

3.2.2 Dirichlet Exponential model

The second non-negative matrix factorization model proposed by Kucukelbir et al. [2015] is very similar to section 3.2.1. It only differs in how the latent variables θ and β are drawn: Each K-vector θ_u is drawn according to Dir($\boldsymbol{\alpha}_0$) and each value in β is drawn from Exponential(0.1), where $\boldsymbol{\alpha}_0$ is set to 1000 in each component. From here, step two from section 3.2.1 is applied. Note that the Dirichlet prior implicitly yields a simplex constraint on each resulting θ_u, which is modeled via a stick-breaking transform readily available in both Pyro and Stan.



(a) Replication of results using pyro.



(b) Copy of figure from original ADVI paper.

Figure 4: Replication analysis for non-negative matrix factorization using the Gamma Poisson model.

3.2.3 Pyro implementation

Implementing the generative model in Pyro is straightforward. Similarly, the NUTS algorithm runs out of the box. To perform variational inference, Pyro relies on so-called 'guides': Essentially, a guide allows to specify a parameterized distribution $q(\theta; \phi)$, where ϕ are the parameters to be optimized (refer to section 2 for details). In contrast to the hierarchical regression models presented in section 3.1, however, we cannot use any of Pyro's built-in 'autoguides' for variational inference, which assume certain distribution families and construct the guides from there. This is because autoguides do not automatically capture either the ordering constraint along the columns of θ in the gamma poisson model, nor the simplex constraint imposed on the rows of θ in the dirichlet exponential model. We therefore resort to implement custom guides that manually declare the variables to optimize over in real coordinate space, as well as specify which transformation T^{-1} (see section 2) to use to get the corresponding values in the latent variable space.

Thanks to the constraint-transformation mappings in Stan's reference manual ([Stan Development Team, 2018b]), we can instruct Pyro to use the exact same transformations in all but one case: The positivity and ordering constraints simultaneously imposed on the columns of θ in section 3.2.1, as the choice of transformation function T^{-1} for this specific case remains unspecified in both the original paper and Stan's reference manual. To fill this blank, our first intuition was to simply chain the transforms used for ordered and positive vectors in Stan, respectively. But since both of them rely on an exponential, this effectively leads to the calculation of a double exponential (e^{e^x}) that is numerically unstable.

We therefore opt to implement and use a custom transform that closely resembles the ordering transform detailed in Stan's reference manual. The resulting inverse transform $T^{-1}(\mathbf{y})$, which maps real vectors $\mathbf{y} \in \mathbb{R}^K$ to positive ordered vectors $\mathbf{x} \in \mathbb{R}^K_{\geq 0}$ is:

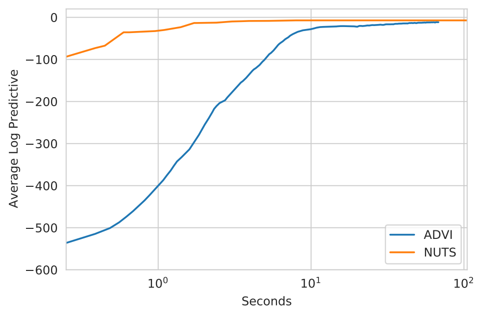
$$x_1 = e^{y_1} \qquad \text{if } i = 1, \text{ and} \tag{1}$$

$$x_i = e^{y_i} + x_{i-1} \qquad \text{if } 1 < i \leq K \tag{2}$$

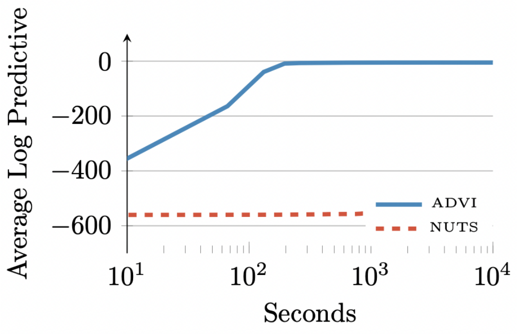
3.2.4 Results & Comparison

Based on the implementation described in section 3.2.3, Pyro is able to run variational inference on both models using the custom guides. The paper does not specify initialization of model parameters, therefore we choose to initialize every single parameter according to $\mathcal{N}(0, 1)$. Similarly, for lack of specification in the original paper, we use a random $80/20$ split between training and testing frames. In doing so, we generally attain the same average log probability per pixel as the authors in the paper and observe similar convergence trends for ADVI. Just like in the original paper, out-of-the-box HMC also did not produce any meaningful samples within a one-hour time budget.

In contrast to the paper, however, we cannot confirm the trends observed when comparing NUTS to ADVI (see Figure 4a). The biggest surprise was the competitive performance of NUTS in both models. Especially with the Dirichlet Exponential model, NUTS continuously outperforms ADVI. We suspect this might be due to the authors of the original paper not exploring many runs or initialization



(a) Replication of results using pyro.



(b) Copy of figure from original ADVI paper.

Figure 5: Replication analysis for non-negative matrix factorization using the Dirichlet Exponential model.

methods when running the NUTS algorithm; the algorithm might therefore have gotten stuck in a local optimum. Another possibility is that Stan's NUTS solver suffered from numerical instabilities that weren't properly reported by the software, as the average log predictive of the NUTS run in Figure 5b is very close to the score ADVI yields after random initialization.

It should also be noted that the paper is not very clear about how the NUTS 'score' was calculated. To get the latent variables needed to calculate the log probability from our implementation, we generally considered the most recent point sampled, while Kucukelbir et al. [2015] might have used a running average of e.g. the last 20 points sampled. This could explain the slower observed convergence in Figure 4b when compared to the graph we obtained in Figure 4a.

Finally, our raw computation times are about an order of magnitude faster than the ones reported in the paper. This is impressive, even though the original data comes from hardware that is at least 7 years old.

4 Extension experiments beyond the original paper

4.1 Gaussian Mixture Models

Kucukelbir et al. [2015] evaluate ADVI on nonconjugate Gaussian Mixture models. They explore the ImageCLEF dataset [Ionescu et al., 2020], where they place a Dirichlet prior on the mixing weights, a Gaussian prior on the component means, and a log-normal prior on the component standard deviations.

We tested the validity of their results for this problem setting of models by creating a synthetic data set. Our synthetic data set is a mixture of K 2-dimensional Gaussians with equal mixture proportions. We choose to use a synthetic data set as this allows us to control the number of mixture components K when generating the data. Additionally, this setup allows us to explicitly study the case in which the number of mixture components in the model matches the number of components in the data, and the case in which the number of components is misspecified in the model.

Formally, we use the following data generating process: Given $\mu_1^0, ..., \mu_K^0$ then

$$\begin{aligned} &\mu_i \sim p \quad \text{with} \quad p(\mu_K^0) = 1/K \quad \text{for } k = 1, ..., K \\ &y_i \sim \mathcal{N}(\mu_i, \text{diag}(0.1, ..., 0.1)) \end{aligned}$$

Further, as in [Kucukelbir et al., 2015], we assume the following probabilistic model for the data

$$p(\mathbf{Y} \mid \boldsymbol{\theta}, \mu, \sigma) = \prod_{n=1}^{N} \sum_{k=1}^{\kappa} \theta_k \mathcal{N}\left(y_i \mid \mu_i, \text{diag}(\boldsymbol{\sigma}_i)\right)$$

with a Dirichlet prior for the mixture proportions, i.e., $p(\theta) = \mathrm{Dir}(\theta \mid \alpha_0)$. A Gaussian prior is selected for the mixture means, i.e.,

$$p(\boldsymbol{\mu}) = \prod_{k=1}^{\kappa} \mathcal{N}(\boldsymbol{\mu}_i \mid 0, 1)$$

and a lognormal prior for the mixture standard deviations, i.e.,

$$p(\boldsymbol{\sigma}) = \prod_{k=1}^{\kappa} \mathrm{logNormal}(\sigma_k \mid 0, 1)$$

4.1.1 Performance with correctly specified K

As Figure 6 shows, ADVI finds a strongly performing solution quicker than NUTS. In fact, NUTS struggles to converge to a solution of the same quality as ADVI. HMC is not shown in the figure as it fails entirely. This is also the case in Kucukelbir et al. [2015] where the authors suggest that this is due to the effect of label switching on HMC-based algorithms when mixture models are given. Figure 7 shows a similar result regarding convergence time of the algorithms, however ADVI significantly outperforms NUTS in terms of log predictive accuracy there. This is not the case for our synthetic data set and the reason is perhaps the difference between the complexity of our data set and ImageCLEF.

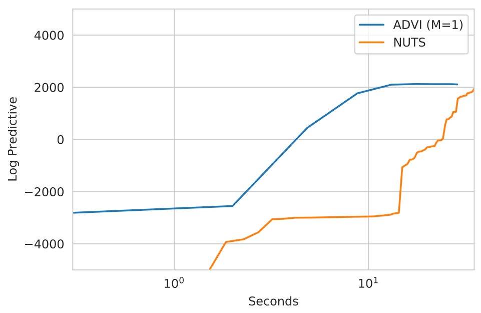


Figure 6: Performance of ADVI and NUTS when applied to the two-component GMM model.

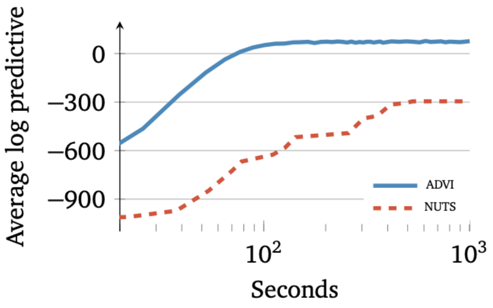


Figure 7: Original performance graphs from [Kucukelbir et al., 2015].

In the original ADVI paper, the authors report that stochastic subsampling of data has improved the model performance. In [Kucukelbir et al., 2015], a significant increase in the performance can be seen after the minibatch size exceeds 500. We also observed an increase in the performance as we increased the size of the minibatches, however the increase was mild. This is perhaps due to the simplicity of our synthetic dataset, even high levels of stochasticity allow the model to generalize well.

Note that the size difference between our synthetic data set and ImageCLEF prevents us from meaningfully comparing the raw times.

4.1.2 Performance with misspecified K

The relation betwen model misspecification and the performance of approximate inference methods is an open research question, see Walker [2013] for instance. In this project, we investigate one very simple form of misspecification: a Gaussian mixture model in which the number of mixture components in the model is smaller than the number of mixture components in the data. We compare how ADVI compares to NUTS under this form of misspecification.

Let K be the actual number of components of the underlying Gaussian Mixture Model that generates our synthetic data, and κ the number of mixture components in our probabilistic model. On real-world data sets correctly specifying K is difficult and oftentimes requires domain knowledge and preliminary data analysis.

To gauge the effect of misspecification we generate a synthetic dataset by sampling from a 5-component Gaussian Mixture Model; then we run ADVI and NUTS with the correctly specified number of components ($K = \kappa = 5$), then with a misspecified number of components ($K \neq \kappa = 3$).

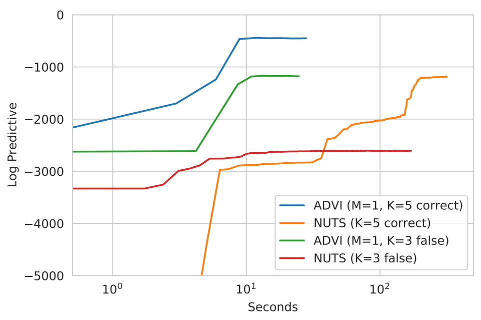


Figure 8: Performance of ADVI when the posterior mode is increased

As seen in Figure 8, we find that that ADVI is more resilient to model misspecification than NUTS. The misspecification leads to a lower log predictive likelihood for both methods, but ADVI consistently outperforms NUTS even under misspecification.

Challenges encountered while implementing and running the experiments in this section were mainly due to Pyro internals. Specific tags and functions are needed in Pyro to properly enumerate discrete latent variables whereas Stan abstracts this part away from the user.

4.2 Subsampling Behaviour of ADVI and Stochastic Gradient MCMC Methods

4.2.1 Motivation

Many of today's most interesting problems in machine learning involve learning from data sets with tens or hundreds of thousands of data points. A central challenge in using approximate inference methods for problems on this scale is that the conventional methods compute the gradients of the objective on the entire data set. This quickly becomes intractable. Thus developing approximate inference methods that work well with data sub-sampling becomes highly relevant.

In section 4.3 of the original ADVI paper, the authors compare the sub-sampling behaviour of ADVI and NUTS. They find that in their specific example ADVI achieves high predictive accuracy while using a batch size of 500. In contrast, they state that NUTS "cannot handle such large datasets [as used in the experiment]". However, based on the paper, it is unclear which batch sizes the authors tried to use with NUTS.

While it is interesting to observe that ADVI works better than NUTS without any modifications to the method, it is well established that HMC is incompatible with naive data sub-sampling [Betancourt, 2015]. However, there is a large literature on stochastic gradient MCMC methods that are are designed to overcome this limitation of HMC. Rather than using naive stochastic HMC as a comparison as done in the paper, we suggest that it is much more interesting to compare the sub-sampling behaviour of ADVI to some of these more advanced stochastic MCMC methods.

We therefore compare stochastic ADVI to two of the most prominent stochastic gradient MCMC methods: Stochastic Gradient Langevin Dynamics (SGLD) [Welling and Teh, 2011] and Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [Chen et al., 2014]. In these experiments, we try to address the following questions:

How robust is ADVI to data subsampling as compared to stochastic gradient MCMC methods? In particular, with memory-bottlenecked settings in mind, at what batch size does the performance of the different approximate inference methods start to deteriorate?

We describe our experimental setup in the following section.

4.2.2 Experimental Setup

To study the sub-sampling behaviour of ADVI, stochastic gradient Langevin dynamics and stochastic gradient HMC, we choose an experimental setting where sub-sampling is virtually always used: training a neural network to perform image classification. To keep things tractable, we choose a two-hidden layer MLP and train it on MNIST [LeCun et al., 1998] and CIFAR-10 [Krizhevsky et al., 2009]. As mentioned above, we compare ADVI, in particular mean-field VI, [Kucukelbir et al., 2015] to two strong stochastic gradient MCMC methods: Stochastic Gradient Langevin Dynamics (SGLD) [Welling and Teh, 2011] and Stochastic Gradient Hamiltonian Monte Carlo (SGHMC) [Chen et al., 2014]. For our experiments, we use the implementations of MFVI, SGLD and SHMC in `https://github.com/google-research/google-research/tree/master/bnn_hmc` which is the code accompanying a large scale comparison of approximate inference methods to HMC by Izmailov et al. [2021]. The paper does not include a study of the sub-sampling behaviour of the various approximate inference, thus our main technical contributions are to build the infrastructure to run each method with various different batch sizes and to build a logging infrastructure to quickly aggregate the results of the various which is not implemented in the repository above. In this example, the latent variables have support in the real coordinate space and we do not need to apply a transformation to them. We use priors of the form $\mathcal{N}(0, \alpha^2 I)$, where we set the prior variance $\alpha^2 = 0.2$. Our model is a two-hidden layer MLP with 256 neurons per layer and ReLU activations. We manually tune the hyper-parameters of the SGLD and SGHMC runs, our final hyperparameters are the ones used in the code in our repository.

4.2.3 Results

In Figure 9 and Figure 10 we compare the accuracy of stochastic ADVI, SGHMC and SGLD of a bayesian neural network on MNIST for various batch sizes. Based on these experiments, we draw the following conclusions.

First, we find that both stochastic ADVI and stochastic MCMC methods achieve high accuracy on this problem even with very small batch sizes. Only SGHMC breaks down for the smallest batch sizes. While the claim in the paper that ADVI works better than HMC without modifying the method holds up, it is possible to use adapted MCMC methods to achieve high performance in a memory-constrained setting. It does however seem like larger changes are necessary to run MCMC with sub-sampling compared to running ADVI with subsampling.

We also evaluate naive stochastic HMC where a new batch is sampled for each trajectory as done in the paper. As found in the paper, we also find that this does not perform well (< 0.6 accuracy). Naive subsampling with HMC generally performs very inconsistently and we only report the best run for a wide range of hyperparameters and batch sizes smaller than 500.

We also note that the stochastic MCMC methods we evaluated seemed much more sensitive to particular choices of hyperparameters than VI. Depending on the time or compute constraints of a given user of these methods, VI might thus be more attractive if one aims to quickly obtain "good enough" results.

Additionally, we wanted to investigate the question which method should be used if we are primarily constrained by time (rather than memory). To this end, we compare the run-times of ADVI, SGLD and SGHMC for a fixed batch size and number of epochs Figure 11. We explore a wide range of hyper-parameters for SGLD and SGHMC but were not able to match the accuracy of ADVI. Among the runs of SGLD and SGHMC that achieved the highest accuracy, we selected those with shortest run time for this comparison. We find that even under this selection mechanism, ADVI achieves a higher accuracy at any given time step than SGLD and SGHMC.

It is also worth noting that SGLD and SGHMC in theory asymptotically produce samples from the true posterior distribution. However, it has been suggested that in practice they rarely do because various necessary requirements and assumptions are not satisfied [Betancourt, 2015, Garriga-Alonso and Fortuin, 2021].

In conclusion, we find that in a memory-constrained setting both ADVI and stochastic MCMC methods achieve high accuracy even when using very small batch sizes. In our experiments, ADVI achieves similar or better results than the stochastic MCMC methods while also being less sensitive to particular choices of hyper-parameters.

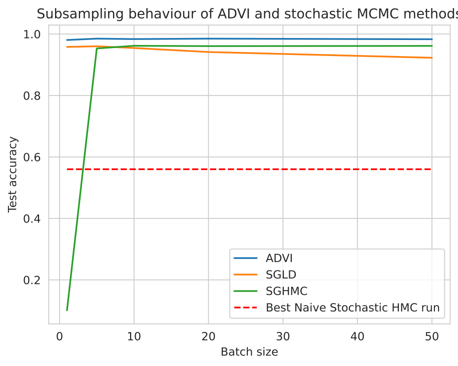


Figure 9: Test accuracies for stochastic ADVI, SGHMC and SGLD for a Bayeisan neural network on MNIST. We also plot the accuracy of the best performing HMC run under naive subsampling.

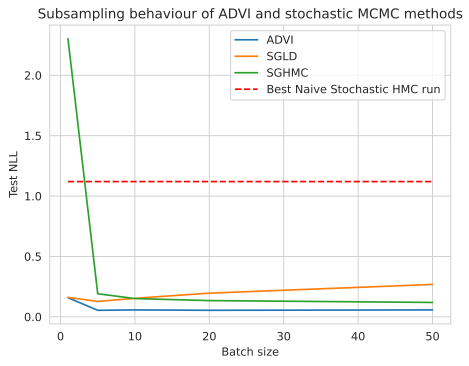


Figure 10: Test NLLs for stochastic ADVI, SGHMC and SGLD for a Bayeisan neural network on MNIST. We also plot the accuracy of the best performing HMC run under naive subsampling

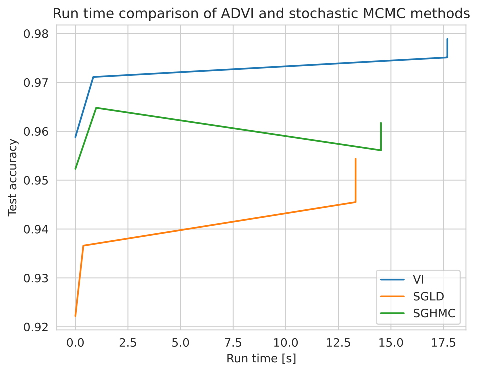


Figure 11: Run time comparison of ADVI to stochastic MCMC methods on MNIST for a fixed number of epochs and a fixed batch size.

5 Conclusion

In conclusion, we find that we are largely able to replicate the results presented by Kucukelbir et al. [2015]. Across a range of experimental settings evaluated in the original paper, we obtain qualitatively similar results. We do however observe that, as compared to the results in Kucukelbir et al. [2015], ADVI slightly underperforms during the beginning of training in the experimental setting of section 3.1.2 and that the NUTS baseline performs better than reported in the paper in section 3.2.

Additionally, we study two study two experimental settings in which we expect that ADVI might underperform MCMC methods and which are only insufficiently discussed in the original paper: model misspecfication and data sub-sampling. We find that, in our experiment, ADVI is more robust to model misspecification than NUTS. Furthermore, we find that ADVI is highly robust to data sub-sampling when training a Bayesian neural network for image classification and matches the performance of specialized MCMC methods which are designed for mini-batch training.

References

Michael Betancourt. The fundamental incompatibility of scalable hamiltonian monte carlo and naive data subsampling. In *International Conference on Machine Learning*, pages 533–540. PMLR, 2015.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming, 2018. URL `https://arxiv.org/abs/1810.09538`.

John Canny. Gap: A factor model for discrete data. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 122–129, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138814. doi: 10.1145/1008992.1009016. URL `https://doi.org/10.1145/1008992.1009016`.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.

Jan Drugowitsch. Variational bayesian inference for linear and logistic regression, 2013. URL `https://arxiv.org/abs/1310.5438`.

Brendan Frey and Sam Roweis. Frey faces dataset, 2008. URL `https://cs.nyu.edu/~roweis/data.html`.

Adrià Garriga-Alonso and Vincent Fortuin. Exact langevin dynamics with stochastic gradients. *arXiv preprint arXiv:2102.01691*, 2021.

Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*, volume Analytical methods for social research. Cambridge University Press, New York, 2007.

Bogdan Ionescu, Henning Müller, Renaud Péteri, Asma Ben Abacha, Vivek Datla, Sadid A. Hasan, Dina Demner-Fushman, Serge Kozlovski, Vitali Liauchuk, Yashin Dicente Cid, Vassili Kovalev, Obioma Pelka, Christoph M. Friedrich, Alba García Seco de Herrera, Van-Tu Ninh, Tu-Khiem Le, Liting Zhou, Luca Piras, Michael Riegler, Pål Halvorsen, Minh-Triet Tran, Mathias Lux, Cathal Gurrin, Duc-Tien Dang-Nguyen, Jon Chamberlain, Adrian Clark, Antonio Campello, Dimitri Fichou, Raul Berari, Paul Brie, Mihai Dogariu, Liviu Daniel Ştefan, and Mihai Gabriel Constantin. Overview of the ImageCLEF 2020: Multimedia retrieval in medical, lifelogging, nature, and internet applications. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, volume 12260 of *Proceedings of the 11th International Conference of the CLEF Association (CLEF 2020)*, Thessaloniki, Greece, September 22-25 2020. LNCS Lecture Notes in Computer Science, Springer.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pages 4629–4640. PMLR, 2021.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic variational inference in stan, 2015. URL `https://arxiv.org/abs/1506.03431`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Stan Development Team. Stan modeling language, version 2.18.0, 2018a. URL `http://mc-stan.org/`.

Stan Development Team. Stan modeling language users guide and reference manual, version 2.18.0, 2018b. URL `https://mc-stan.org/docs/2_18/reference-manual/`.

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Stephen G. Walker. Bayesian inference with misspecified models. *Journal of Statistical Planning and Inference*, 143(10):1621–1633, 2013. ISSN 0378-3758. doi: https://doi.org/10.1016/j.jspi.2013.05.013. URL `https://www.sciencedirect.com/science/article/pii/S037837581300116X`.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.