

# Paper presentation: Rethinking Variational Inference for Probabilistic Programs with Stochastic Support

Aleksander Buszydlik, Karol Dobiczek



# About the paper

## Authors:

- Tim Reichelt
- Luke Ong
- Tom Rainforth

Published in:  
NeurIPS 2022

Impact:  
As of now, 0 citations



---

## Rethinking Variational Inference for Probabilistic Programs with Stochastic Support

---

Tim Reichelt<sup>1</sup> Luke Ong<sup>1,2</sup> Tom Rainforth<sup>1</sup>

<sup>1</sup> University of Oxford

<sup>2</sup> Nanyang Technological University, Singapore

{tim.reichelt, lo}@cs.ox.ac.uk rainforth@stats.ox.ac.uk

### Abstract

We introduce *Support Decomposition Variational Inference* (SDVI), a new variational inference (VI) approach for probabilistic programs with stochastic support. Existing approaches to this problem rely on designing a single global variational guide on a variable-by-variable basis, while maintaining the stochastic control flow of the original program. SDVI instead breaks the program down into sub-programs with static support, before automatically building separate sub-guides for each. This decomposition significantly aids in the construction of suitable variational families, enabling, in turn, substantial improvements in inference performance.

### 1 Introduction

Probabilistic programming systems (PPSs) enable users to express probabilistic models with computer programs and provide tools for inference. Many PPS, such as Stan [1] or PyMC3 [2], limit the expressiveness of their language to ensure that the programs in their language always correspond to models with static support—i.e. the number of variables and their support do not vary between program executions. In contrast, *universal PPSs* [3–11] can encode programs where the sequence of variables itself—not just the variable values—changes between executions, leading to models with stochastic support. These models have applications in numerous fields, such as natural language processing [12], Bayesian Nonparametrics [13], and statistical phylogenetics [14]. A wide range of simulator-based models similarly require such stochastic control flow [15–17].

The effectiveness of PPSs is heavily reliant on the underlying inference schemes they support. Variational inference (VI) is one of the most popular such schemes, both in PPSs and more generally [18–20]. This popularity is due to its ability to use derivatives to scale to large datasets and high-dimensional models [21–24], often providing much faster and more scalable inferences compared to Monte Carlo approaches [25]. To provide the required derivatives, a number of modern universal PPSs—such as Pyro [5], ProbTorch [26], PyProb [15], Gen [7], and Turing [6]—have introduced automatic differentiation [27] capabilities for programs with stochastic control flow. One of the core aims behind these developments was to support VI schemes in such settings [5].

However, constructing appropriate variational families, typically known as guides in PPSs, can be very challenging for problems with stochastic support, even for expert users. This is because the stochasticity of the control flow induces discontinuities and complex dependency structures that are difficult to remain faithful to and design parameterized approximations for. Furthermore, while there are a plethora of different automatic guide construction schemes for static support problems [18–20], there is a lack of suitable schemes applicable to models with stochastic support. Consequently, existing methods tend to give unreliable results in such settings, as we demonstrate in Figure 1.

We argue that a significant factor of this shortfall is that standard practice—for both manual and automated methods—is to construct the guide on a variable-by-variable basis [28–31]. Namely,

# Background: static and stochastic support

- In simple terms, *support* is the set of values which can be taken by an RV
- *Static support* programs have:
  - a constant number of variables
  - a constant support for these variables
- *Stochastic support* programs may have:
  - a number of variables that varies between executions
  - a sequence of variables that varies between executions
- Today we are discussing *Support Decomposition Variational Inference*

# What are the challenges of stochastic support?

1. Discontinuities and complex dependency structure of the PDF
2. Difficult design of parameterized approximations for those PDFs
3. Lack of appropriate techniques to construct guides (variational distributions)

# Where do these discontinuities come from?

```
def model():  
    x = sample("x", Normal(0, 1))  
    if x < 0:  
        z = sample("z1", Normal(-3, 1))  
    else:  
        z = sample("z2", Normal(3, 1))  
    sample("a", Normal(z, 2), obs=2.0)
```

## Now imagine...

```
def model2():  
    x = sample("x", Normal(0, 1))  
    if x < 0:  
        z = sample("z1", Poisson(|x|))  
    else:  
        y = sample("y", Normal(0, x))  
        z = sample("z2", Exponential(|y|))  
    sample("a", Normal(x, z), obs=2.0)
```

`y` exists only in the `else` branch...

Quickly becomes a mess!

# Let's consider an individual execution

```
def model2():
```

```
    x = sample("x", Normal(0, 1))
```

→ x = -0.50

```
    if x < 0:
```

```
        z = sample("z1", Poisson(|x|))
```

→ z = 1.00

```
    else:
```

```
        y = sample("y", Normal(0, x))
```

```
        z = sample("z2", Exponential(|y|))
```

```
    sample("a", Normal(x, z), obs=2.0)
```

→ a = -0.80

# This is actually a program with static support!

```
def model2A():  
    x = sample("x", Normal(0, 1))           → x = -0.50  
    z = sample("z1", Poisson(|x|))         → z = 1.00  
    sample("a", Normal(x, z), obs=2.0)     → a = -0.80
```



# And so is this program!

```
def model2B():  
    x = sample("x", Normal(0, 1))           → x = 0.67  
    y = sample("y", Normal(0, x))          → y = 0.20  
    z = sample("z2", Exponential(|y|))     → z = 0.30  
    sample("a", Normal(x, z), obs=2.0)     → a = 0.50
```

# Straight Line Programs (SLPs)

- In simple terms, SLP is program consisting only of a sequence of basic operations
- In other words no loops, branching, ...
- Any program can be represented as a *mixture* of SLP densities:

$$\text{model2} = \alpha * \text{model2A} + \beta * \text{model2B} + \gamma * \text{model2C} + \dots$$

- Which SLPs? We will come back to that question!

# Decomposing a stochastic support PP

- Each encountered sample and observe statement contributes to the density function:

$$\gamma(x_{1:n_x}) := \prod_{i=1}^{n_x} f_{a_i}(x_i | \eta_i) \prod_{j=1}^{n_y} g_{b_j}(y_j | \phi_j)$$

*n<sub>x</sub> samples*                      *n<sub>y</sub> observes*

# Decomposing a stochastic support PP

- Each encountered sample and observe statement contributes to the density function:

$$\gamma(x_{1:n_x}) := \prod_{i=1}^{n_x} f_{a_i}(x_i | \eta_i) \prod_{j=1}^{n_y} g_{b_j}(y_j | \phi_j)$$

$n_x$  samples  $n_y$  observes

- The density of the  $k^{\text{th}}$  SLP is simply:

$$\gamma_k(x_{1:n_k}) = \mathbb{I}[x_{1:n_k} \in \chi_k] \gamma(x_{1:n_k})$$

## Global and local ELBO

$$\mathcal{L}(\phi, \lambda) = \mathbb{E}_{q(x; \phi, \lambda)} [\log \gamma(x) - \log q(x; \phi, \lambda)]$$

## Global and local ELBO

$$\mathcal{L}(\phi, \lambda) = \mathbb{E}_{q(x; \phi, \lambda)} [\log \gamma(x) - \log q(x; \phi, \lambda)]$$

Factorize the distribution for each SLP  $k$

$$q(x; \phi, \lambda) = \sum_{k=1}^K q_k(x; \phi_k) q(k; \lambda)$$

# Global and local ELBO

$$\mathcal{L}(\phi, \lambda) = \mathbb{E}_{q(x; \phi, \lambda)} [\log \gamma(x) - \log q(x; \phi, \lambda)]$$

Factorize the distribution for each SLP  $k$

$$q(x; \phi, \lambda) = \sum_{k=1}^K q_k(x; \phi_k) q(k; \lambda)$$

Derive the global...

$$\mathcal{L}(\phi, \lambda) = \mathbb{E}_{q(k; \lambda)} [\mathcal{L}_k(\phi_k) - \log q(k; \lambda)]$$

and the local ELBO

$$\mathcal{L}_k(\phi_k) := \mathbb{E}_{q_k(x; \phi_k)} \left[ \log \frac{\gamma_k(x)}{q_k(x; \phi_k)} \right]$$

# Global and local ELBO

$$\mathcal{L}(\phi, \lambda) = \mathbb{E}_{q(x; \phi, \lambda)} [\log \gamma(x) - \log q(x; \phi, \lambda)]$$

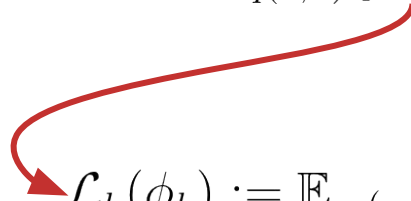
Factorize the distribution for each SLP  $k$

$$q(x; \phi, \lambda) = \sum_{k=1}^K q_k(x; \phi_k) q(k; \lambda)$$

Derive the global...

$$\mathcal{L}(\phi, \lambda) = \mathbb{E}_{q(k; \lambda)} [\mathcal{L}_k(\phi_k) - \log q(k; \lambda)]$$

and the local ELBO


$$\mathcal{L}_k(\phi_k) := \mathbb{E}_{q_k(x; \phi_k)} \left[ \log \frac{\gamma_k(x)}{q_k(x; \phi_k)} \right]$$



# Discussion: Finding SLPs

How would you approach it?

# Finding SLPs the trivial way

- Sample the paths by running the simulation
  - Simple
  - Cheap

# Finding SLPs the trivial way

- Sample the paths by running the simulation
  - Simple
  - Cheap
  
- Although there are drawbacks
  - **What are those?**

# Finding SLPs the trivial way

- Sample the paths by running the simulation
  - Simple
  - Cheap
  
- Although there are drawbacks
  - Not guaranteed to find all SLPs
  - We pay the same amount of resources to all SLPs

# Finding SLPs the proper way

- Some SLPs might be more promising than others – focus on those;
- Frame the problem as cost optimization:
  1. Given some finite resource  $T$
  2. Distribute it over each local ELBO
  3. Find a distribution which maximizes global ELBO
- Take only those SLPs which are the most important

## Last improvement – batching

- When observations are conditionally independent we can load the data in batches
- This leads to a technique authors call Stochastic-SDVI
  - Similar performance
  - Smaller disk requirements

# Discussion: How does SDVI compare to DCC?

- Similarities?
- Differences?
- Anything else?

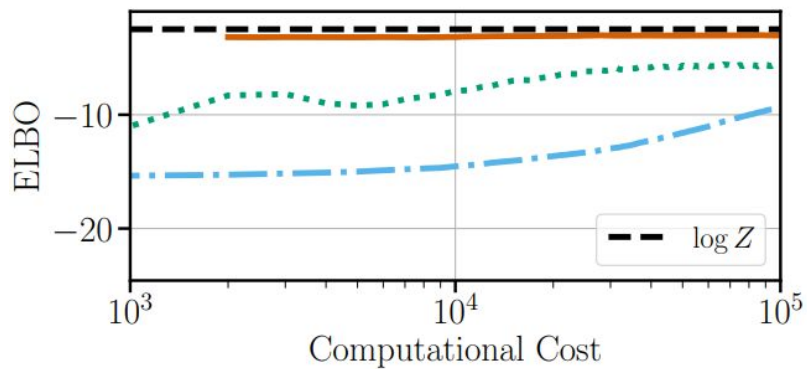
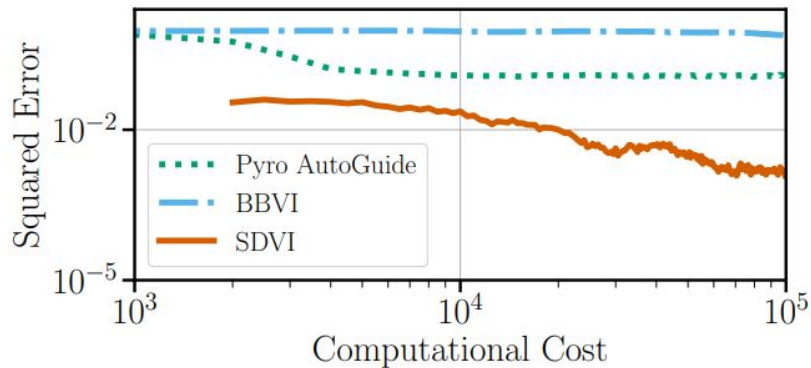
# Experiment 1: Failure Modes 1

- In what ways do various approaches for VI can fail?
- Look at the following approaches for Variational Inference:
  - *SDVI (Support Decomposition Variational Inference, this paper)*
  - *Pyro AutoGuide*
  - *BBVI (guides generated on a variable-by-variable basis <sup>1</sup>)*
- Consider the following model, assume we observed  $y = 2$

$$\begin{aligned} u &\sim \mathcal{N}(0, 5^2), \\ x &\sim \mathcal{N}(z, 1), \\ y &\sim \mathcal{N}(x, 1). \end{aligned} \quad \text{where} \quad z = \begin{cases} 0, & \text{if } u \in (-\infty, -4] \\ K, & \text{if } u \in (-5 + K, -4 + K] \text{ for } K = 1, \dots, 8 \\ 9, & \text{if } u \in (4, \infty) \end{cases}$$

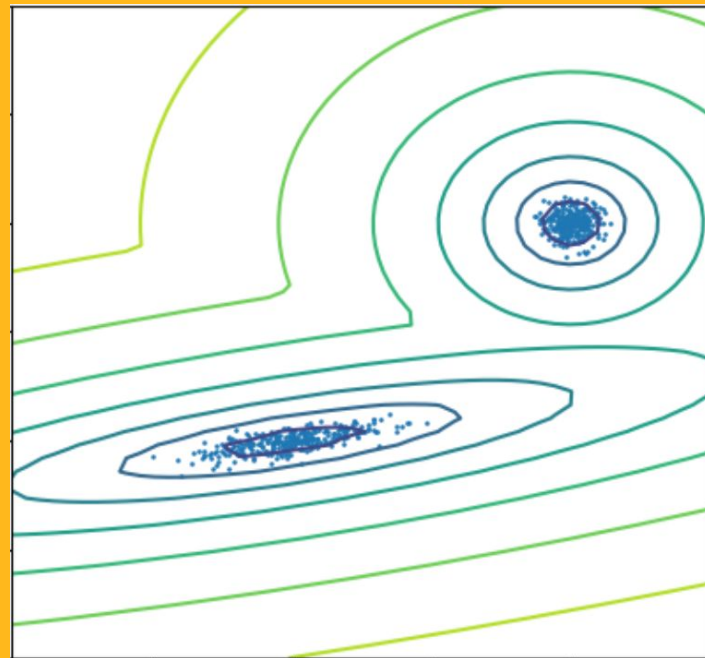


# Experiment 1: Failure Modes 2



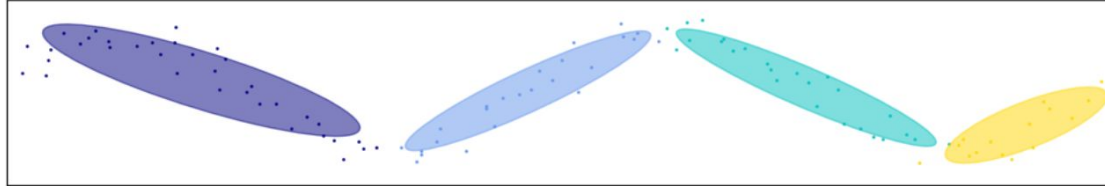
## Detour: Gaussian Mixture Models 1

- Assume the data is a mixture of a (finite) set of Gaussian distributions
- We may not know the parameters and the number of distributions
- Very expressive way to describe distributions, also for clustering

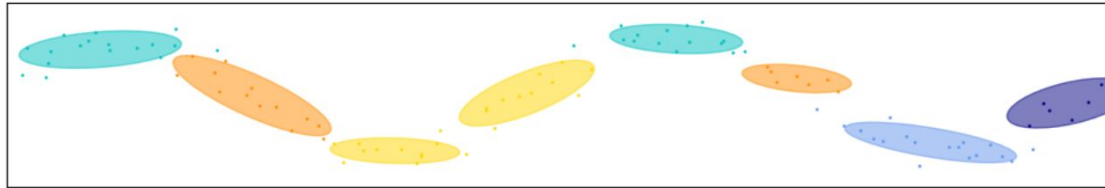


# Detour: Gaussian Mixture Models 2

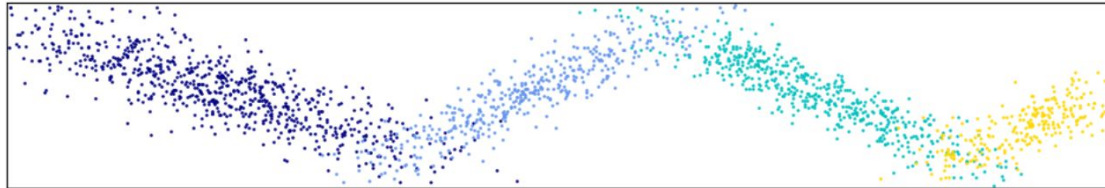
Bayesian Gaussian mixture models with a Dirichlet process prior for  $\gamma_0 = 0.01$ .



Bayesian Gaussian mixture models with a Dirichlet process prior for  $\gamma_0 = 100$



Gaussian mixture with a Dirichlet process prior for  $\gamma_0 = 0.01$  sampled with 2000 samples.



## Experiment 2: Challenging clustering 1

- How does SDVI compare to baselines on a challenging clustering task?
- Look at the following approaches for Variational Inference:
  - SDVI (*Support Decomposition Variational Inference*, this paper)
  - S-SDVI (*Stochastic SDVI*, mini-batched SDVI)
  - BBVI (guides generated on a variable-by-variable basis)
  - DCC<sup>2</sup>
- Consider the following model, assume we have  $K = 5$

$$K \sim \text{Poisson}(9) + 1; \quad u_k \sim \mathcal{N}(\mathbf{0}, 10\mathbf{I}) \text{ for } k = 1, \dots, K; \quad y \sim \frac{1}{K} \sum_{k=1}^K \mathcal{N}(\mu_k, 0.1\mathbf{I}),$$

## Experiment 2: Challenging clustering 2

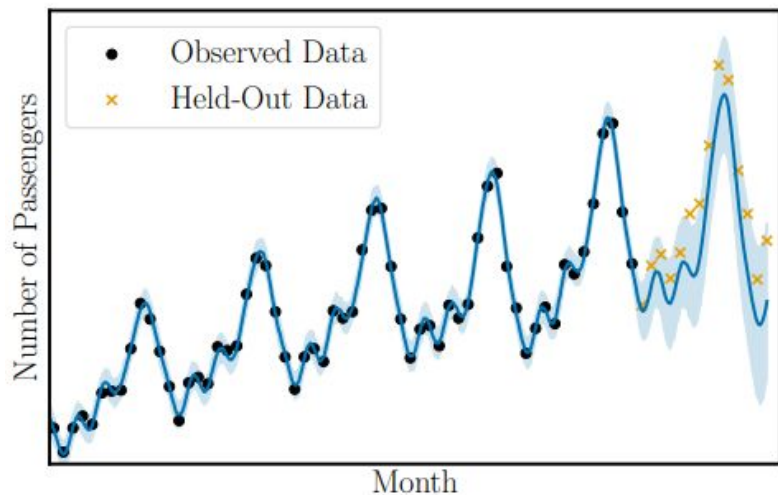
Method	LPPD ( $\uparrow$ , $\times 10^3$ )	ELBO ( $\uparrow$ , $\times 10^3$ )	MAP $K$
DCC	$-9842.90 \pm 3904.57$	N/A	14, 11, 16, 14, 15
BBVI	$-2217.07 \pm 146.31$	$-8770.55 \pm 544.95$	25, 25, 25, 25, 25
SDVI	<b><math>32.84 \pm 0.02</math></b>	<b><math>128.76 \pm 0.17</math></b>	5, 5, 6, 6, 5
S-SDVI	<b><math>32.80 \pm 0.02</math></b>	<b><math>128.63 \pm 0.22</math></b>	5, 5, 6, 5, 6

# Experiment 3: Challenging regression 1

- How does **SDVI** compare to **baselines** on a **challenging regression task**?
- Look at the following approaches for Variational Inference:
  - *SDVI* (*Support Decomposition Variational Inference*, this paper)
  - *BBVI* (guides generated on a variable-by-variable basis)
  - *DCC*
- Consider Gaussian Process Kernels built (probabilistically) according to:

$$\mathcal{K} \rightarrow \begin{array}{c} \text{SE} \mid \text{RQ} \mid \text{PER} \mid \text{LIN} \mid \mathcal{K} \times \mathcal{K} \mid \mathcal{K} + \mathcal{K}. \\ 0.2 \quad 0.2 \quad 0.2 \quad 0.2 \quad 0.1 \quad 0.1 \end{array}$$

## Experiment 3: Challenging regression 2



Method	LPPD ( $\uparrow$ )	ELBO ( $\uparrow$ )
DCC	$-58.92 \pm 32.47$	N/A
BBVI	$-18.82 \pm 1.20$	$-48.48 \pm 0.33$
SDVI	<b><math>2.05 \pm 3.30</math></b>	<b><math>34.53 \pm 21.42</math></b>

Thank you for your attention!

Aleksander & Karol