We propose a **refined variational approximation** by embedding a MCMC sampler inside it, accelerating Bayesian inference

Variationally Inferred Sampling for Probabilistic Programs

Victor Gallego and David Rios Insua

Intro & Background

- A **probabilistic program** defines a probabilistic model $p(x, z)$ that factorizes as $p(x,z) = p(x|z) \prod_{i=1}^{n} p(z_i | z_{<i})$, where x are observations and z latent variables or parameters. **Problem**: posterior $p(z|x)$ is hard to compute. **Two solutions**:

Markov Chain Monte Carlo (MCMC):

- Inference as **sampling**: Markov chain towards posterior distribution in the limit.
- **Problem**: scalability, slow mixing.
- SGLD [Welling and Teh, 2011]: $z_{t+1} \leftarrow z_t - \eta_t \nabla \log p(z_t, x) + \mathcal{N}(0, 2\eta_t I)$.
- SGLD+R [Gallego and Rios Insua, 2018]: add interaction term to speed up mixing.

Variational Inference (VI):

- Inference as **optimization**: Minimize divergence $KL(q||p)$ between true posterior $p(z|x)$ and a tractable family $q(z|x)$ (eg: mean field Gaussian).
- SVI: Maximize ELBO(q) $= E_{q_\phi(z|x)} [\log p(x,z) - \log q_\phi(z|x)]$
- **Problem**: bias, underestimation of uncertainty.

Goal: propose a variational approximation, that is flexible enough (i.e., the user can control its accuracy by using more computing time).

The VIS framework

The **refined variational approximation** is given by

$$q_{\phi,\eta}(z|x) = \int Q_\eta(z|z_0) q_{0,\phi}(z_0|x) dz_0$$

- $q_{0,\phi}(z|x)$ is the initial and tractable density (diagonal Gaussian).
- $Q_\eta(z|z_0)$ refers to a stochastic process parameterized by η used to evolve the original density $q_{0,\phi}(z|x)$.
- Think of $Q_\eta(z|z_0)$ as T iterations of an MCMC transition kernel, for example SGLD (see at Background section).

- Since the resulting distribution is implicitly defined by the sampler, its density is not available to us.
- $q_{\phi,\eta}(z|x)$ is approximated using a **finite set of particles** (each treated as a Dirac Delta distribution):
$\tilde{q}_{\phi,\eta}(z|x) = \frac{1}{K} \sum_{i=1}^{K} \delta(z - z^i)$
- Convergence results in the paper.

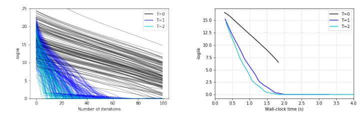
(Hyper)parameter tuning via autodiff

- Since we have embedded the sampler inside a variational approximation, we can **optimize wrt the sampler parameters**:
- Initial distribution of the sampler: ∇_ϕELBO(q), learns good starting points.
- Sampler parameters: ∇_ηELBO(q), learns learning rate.

Experiments

State space models

- **Hidden Markov Model**: $p(z_{1:T}, x_{1:T}, \theta) = \prod_{t=1}^{T} p(x_t|z_t, \theta) p(z_t|z_{t-1}, \theta) p(\theta)$.
- Experiments on a synthetic time-series for 100 different random initializations of θ.



- **Dynamic Linear Model**: same structure as HMM but with Gaussian latents and observations. We use the Mauna Loa monthly CO_2 time series data. As the training set, we take the first 10 years, and we evaluate over the next 2 years:

	$T=0$	$T=1$
MAE	0.270	**0.239**
predictive entropy	2.537	**2.401**
interval score ($\alpha = 0.05$)	15.247	**13.461**

- VIS helps in predicting uncertainty.

Variational Autoencoder (VAE)

- Problem: learn a complex, high-dimensional data distribution $p(x)$.
- Datasets: MNIST and fashion-MNIST: 60000 28 × 28 images each.
- VAE as the model:
  - $p_\theta(x|z)$ is a deep neural network (generates the pixels)
  - $q_\phi(z|x)$ is a diagonal Gaussian whose mean and variance is parameterized by a deep neural network.
- We compare the VIS framework, specifying different values of T.

Table 3: Test log-likelihood on binarized MNIST and fMNIST.

Method	MNIST	fMNIST
Results from (Titsias and Ruiz 2019)		
UIVI	-94.09	-110.72
SIVI	-97.77	-121.53
VAE	-98.29	-126.73
VIS-5-10 (this paper)	$\mathbf{-86.23 \pm 0.80}$	$\mathbf{-105.92 \pm 0.49}$
VIS-0-10 (this paper)	-96.16 ± 0.17	-120.53 ± 0.59
VAE (VIS-0-0)	-100.91 ± 0.16	-125.57 ± 0.63

- Mean times (s) per epoch: 10.30 ($T = 5$), 6.52 ($T = 0$) (on GPU).
- For fair comparisons, VIS-5-10 was run for 10 epochs; all others for 20 epochs.

Conclusions

- **VIS** uses variational inference techniques to **speed up a MCMC sampler**. If you prefer, it uses MCMC to make **VI more accurate**.
- The user can naturally tradeoff compute for better accuracy.
- Autotuning MCMC parameters via autodiff.
- Only requires a standard automatic differentiation library (coded in Pytorch).

ICMAT Instituto de Ciencias Matemáticas

slides & poster at https://vicgalle.github.io