2nd Symposium on Advances in Approximate Bayesian Inference, 2019 1–18

Variationally Inferred Sampling Through a Refined Bound

Víctor Gallego VICTOR.GALLEGO@ICMAT.ES
David Ríos Insua DAVID.RIOS@ICMAT.ES
Institute of Mathematical Sciences (ICMAT), Madrid, SPAIN.
Duke University (Dept. of Statistical Sciences) and SAMSI, Durham, NC, USA.

Abstract

A framework for efficient Bayesian inference in probabilistic programs is introduced by embedding a sampler inside a variational posterior approximation. Its strength lies in both ease of implementation and automatically tuning sampler parameters to speed up mixing time. Several strategies to approximate the *evidence lower bound* (ELBO) computation are introduced, including a rewriting of the ELBO objective. Experimental evidence is shown by performing experiments on an unconditional VAE on density estimation tasks; solving an influence diagram in a high-dimensional space with a conditional variational autoencoder (cVAE) as a deep Bayes classifier; and state-space models for time-series data.

1. Introduction

We consider a probabilistic program (PP) to define a distribution $p(x, z)$, where x are observations and z, both latent variables and parameters, and ask queries involving the posterior $p(z|x)$. This distribution is typically intractable but, conveniently, probabilistic programming languages (PPLs) provide inference engines to approximate it using Monte Carlo methods (e.g. particle Markov Chain Monte Carlo (MCMC) (Andrieu et al., 2010) or Hamiltonian Monte Carlo (HMC) (Neal et al., 2011)) or variational approximations (e.g. Automatic Differentiation Variational Inference (ADVI) (Kucukelbir et al., 2017)). Whereas the latter are biased and underestimate uncertainty, the former may be exceedingly slow depending on the target distribution. For such reason, over the recent years, there has been an increasing interest in developing more efficient posterior approximations (Nalisnick et al., 2016; Salimans et al., 2015; Tran et al., 2015).

It is known that the performance of a sampling method depends on the parameters used (Papaspiliopoulos et al., 2007). Here we propose a framework to automatically adapt the posterior shape and tune the parameters of a posterior sampler with the aim of boosting Bayesian inference in PPs. Our framework constitutes a principled way to enhance the flexibility of the variational posterior approximation, yet can be seen also as a procedure to tune the parameters of an MCMC sampler. Our contributions are a new flexible and unbiased variational approximation to the posterior, which improves an initial variational approximation with a (learnable via automatic differentiation) stochastic process. Appendix A discusses related work.

© V. Gallego & D.R. Insua.

2. The Variationally Inferred Sampling (VIS) framework

In standard VI, the variational approximation $q_\phi(z|x)$ is analytically tractable and typically chosen as a factorized Gaussian distribution. We propose to use a more flexible approximating posterior by embedding a sampler through:

$$q_{\phi,\eta}(z|x) = \int Q_{\eta,T}(z|z_0) q_{0,\phi}(z_0|x) dz_0, \tag{1}$$

where $q_{0,\phi}(z|x)$ is the initial and tractable density (i.e., the starting state for the sampler). We refer to $q_{\phi,\eta}(z|x)$ as the refined variational approximation. The distribution $Q_{\eta,T}(z|z_0)$ refers to a stochastic process parameterized by η used to evolve the original density $q_{0,\phi}(z|x)$ and achieve greater flexibility; we describe below particular forms of it. When $T = 0$, no refinement steps are performed, and the refined variational approximation coincides with the original one, $q_{\phi,\eta}(z|x) = q_{0,\phi}(z|x)$. As T increases, the variational approximation will be closer to the exact posterior, provided that $Q_{\eta,T}$ is a valid MCMC sampler. Next, we maximize a refined ELBO objective,

$$\text{ELBO}(q) = \mathbb{E}_{q_{\phi,\eta}(z|x)} \left[\log p(x,z) - \log q_{\phi,\eta}(z|x)\right] \tag{2}$$

to optimize the divergence $KL(q_{\phi,\eta}(z|x)||p(z|x))$. The first term of ELBO only requires sampling from $q_{\phi,\eta}(z|x)$; however the second term, $-\mathbb{E}_{q_{\phi,\eta}(z|x)} \left[\log q_{\phi,\eta}(z|x)\right]$ requires also evaluating the evolving density. Regarding $Q_{\eta,T}(z|z_0)$, we consider the following families of sampling algorithms.

2.1. The sampler $Q_{\eta,T}(z|z_0)$

When the latent variables z are continuous ($z \in \mathbb{R}^d$), we evolve the original variational density $q_{0,\phi}(z|x)$ through a stochastic diffusion process. To make it tractable, we discretize the Langevin dynamics using the Euler-Maruyama scheme, arriving at the stochastic gradient Langevin dynamics (SGLD) sampler. We then follow the process $Q_{\eta,T}(z|z_0)$ (representing T iterations of an MCMC sampler). As an example, for the SGLD sampler $z_i = z_{i-1} + \eta \nabla \log p(x, z_{i-1}) + \xi_i$, where i iterates from 1 to T; in this case, the only parameter of the SGLD sampler is the learning rate η. The noise for the SGLD is $\xi_i \sim \mathcal{N}(0, 2\eta I)$. The initial variational distribution $q_{0,\phi}(z|x)$ is a Gaussian parameterized by a deep neural network (NN). Then, T iterations of a sampler Q parameterized by η are applied leading to $q_{\phi,\eta}$.

An alternative may be given by ignoring the noise vector ξ (Mandt et al., 2017), thus refining the initial variational approximation with just stochastic gradient descent (SGD). Moreover, we can use Stein variational gradient descent (SVGD) (Liu and Wang, 2016) or a stochastic version (Gallego and Insua, 2018) to apply repulsion between particles and promote a more extensive exploration of the latent space.

2.2. Approximating the entropy term

We propose a set of guidelines for the ELBO optimization using the refined variational approximation.

Particle approximation We can consider the flow $Q_{\eta,T}(z|z_0)$ as a mixture of Dirac deltas (i.e., we approximate it with a finite set of particles). That is, we sample $z^1, \dots, z^K \sim Q_{\eta,T}(z|z_0)$ and use $\tilde{Q}_{\eta,T}(z|z_0) = \frac{1}{K}\sum_{i=1}^K \delta(z - z^i)$. Thus, that entropy term is zero so $\mathbb{E}_{q_{\phi,\eta}(z|x)}[\log q_{\phi,\eta}(z|x)] = \mathbb{E}_{q_{0,\phi}(z|x)}[\log q_{0,\phi}(z|x)]$. If using SGD as the sampler, the resulting ELBO is tighter than the one with no refinement (see Appendix D.1). However, discarding the entropy in the sampling process results in variational approximations that are to concentrated around the MAP solution, and this might be undesirable for training generative models.

Gaussian approximation In settings were it could be helpful to have a posterior approximation that places density over the whole latent space. For the particular case of using SGD as the inner kernel, we have

$$z_0 \sim q_{0,\phi}(z_0|x) = \mathcal{N}(z_0|\mu_\phi(x), \sigma_\phi(x))$$
$$z_i = z_{i-1} + \eta \nabla \log p(x, z_{i-1}), \qquad i = 1, \dots, T.$$

By treating the gradient terms as points, we have that the refined variational approximation can be computed as $q_{\phi,\eta}(z|x) = \mathcal{N}(z|z_T, \sigma_\phi(x))$. Note that there is an implicit dependence on η through z_T.

MC approximation Instead of performing the full marginalization in integral (1), we can approximate it as $q_{\phi,\eta}(z_T|x) = \prod_{i=1}^T q_\eta(z_i|z_{i-1}) q_{0,\phi}(z_0|x)$. The entropy for each factor can be straightforwardly computed, i.e. for the case of SGLD, $q_\eta(z_i|z_{i-1}) = \mathcal{N}(z_{i-1} + \eta \nabla \log p(x, z_{i-1}), 2\eta I)$. This approximation keeps track of a better estimate of the entropy than the particle approximation.

Deterministic flows If using a deterministic flow (such as SGD or SVGD), we can keep track of the change in entropy at each iteration using the change of variable formula as done in Duvenaud et al. (2016). However, this requires a costly Jacobian computation, making it unfeasible to combine with our *backpropagation through the sampler* scheme (Sec. 2.3) for moderately complex problems.

2.3. Tuning sampler parameters via Automatic Differentiation

In standard VI, the variational approximation $q(z|x;\phi)$ is parameterized by ϕ. The parameters are learned using SGD or variants such as Adam (Kingma and Ba, 2014), using $\nabla_\phi \text{ELBO}(q)$. Since we have shown how to embed a sampler inside the variational guide, it is also possible to compute a gradient of the objective with respect to the sampler parameters η. For instance, we can compute a gradient with respect to the learning rate η from the SGLD or SGD process from Section 2.1, $\nabla_\eta \text{ELBO}(q)$, to search for an optimal step size at every VI iteration. This is an additional step apart from using the gradient $\nabla_\phi \text{ELBO}(q)$ employed to learn a good initial sampling distribution. See Appendix D.3 for a discussion on two modes of automatic differentiation that can be used.

3. Results

Code is released at https://github.com/vicgalle/vis. The VIS framework was implemented using Pytorch (Paszke et al., 2017), though we also release a notebook for the first

experiment using Jax to highlight its simple implementation. Appendix B contains additional experiments; Appendix C, implementation details.

Funnel density As a preliminary experiment, we test the VIS framework on a synthetic yet complex target distribution. The target, bi-dimensional density is defined through:

$$z_1 \sim \mathcal{N}(0, 1.35), \quad z_2 \sim \mathcal{N}(0, \exp(z_1)).$$

As a variational approximation we take the usual diagonal Gaussian. For the VIS case, we refine it for $T = 1$ steps using SGLD. Results are in Figure 1. Clearly, our refined version achieves a tighter bound, the VIS variant is placed nearer to the mean of the true distribution and is more disperse than the original variational approximation, confirming that the refinement step helps in attaining more flexible posterior approximations.

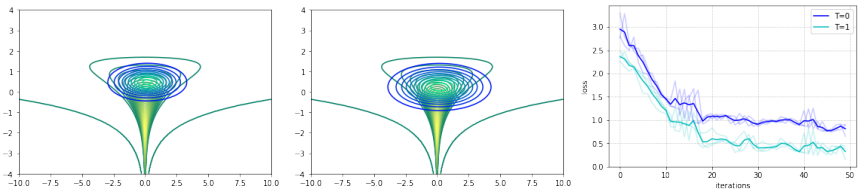

Figure 1: Left: contour curves (turquoise) of variational approximation with no refinement ($T = 0$) at iteration 30 (loss, 1.011). Center: contour curves (turquoise) of refined variational approximation ($T = 1$) at iteration 30 (loss, 0.667). Green-yellow curves denote target density. Right: evolution of -ELBO objective for 50 iterations. Darker lines depict mean along different seeds (lighter lines).

State-space model (DLM) We now test the VIS framework on the Mauna Loa monthly CO_2 time series data (Keeling, 2005). As the training set, we take the first 10 years, and we evaluate over the next 2 years. We use a dynamic linear model (DLM) composed of a local linear trend plus a seasonality block of periodicity 12. Full model specification can be checked in Appendix C.1. As a preprocessing step, we standardize the time series to zero mean and unitary deviation. To guarantee the same computational budget time, the model without refining is run for 10 epochs, whereas the model with refinement is run for 4 epochs. We use the particle approximation from Sec. 2.2. We report mean absolute error (MAE) and predictive entropy in Table 1. In addition, we compute the interval score as defined in (Gneiting and Raftery, 2007), a strictly proper scoring rule. As can be seen, for similar wall-clock times, the refined model not only achieves lower MAE, but also its predictive intervals are narrower than the non-refined counterpart.

Table 1: Prediction metrics for the DLM.

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

Variational Autoencoder We aim to check whether VIS is competitive with respect to other recent algorithms. We test our approach in a Variational Autoencoder (VAE) model (Kingma and Welling, 2013), which is the building block of more complex models and tasks (Chen et al., 2018b; Bouchacourt et al., 2018). The VAE defines a conditional distribution $p_\theta(x|z)$, generating an observation x from a latent variable z. We are interested in modelling two 28×28 image distributions, MNIST and fashion-MNIST. To perform inference (learn parameters θ), the VAE introduces a variational approximation $q_\phi(z|x)$. In the standard setting, this is Gaussian; we instead use the refined variational approximation with various values of T. We used the MC approximation, though achieved similar results using the Gaussian one. As experimental setup, we reproduce the setting from Titsias and Ruiz (2019). Results are reported in Table 2. To guarantee a fair comparison, we trained the VIS-5-10 variant for 10 epochs, whereas all the other variants were trained for 15 (fMNIST) or 20 epochs (MNIST), so that the VAE performance is comparable to that in Titsias and Ruiz (2019). Although VIS is trained for less epochs, by increasing the number of MCMC iterations T, we dramatically improve on test log-likelihood. In terms of computational complexity, the average time per epoch using $T = 5$ is 10.46s, whereas with no refinement ($T = 0$) is 6.10s (hence our decision to train the refined variant for less epochs): a moderate increase in computing time compensates the dramatic increase in log-likelihood while not introducing new parameters, except for the learning rate η. We also compare our results with the contrastive divergence approach (Ruiz and Titsias, 2019). Figure 2 displays ten random samples of reconstructed digit images as visual check.

Table 2: Test log-likelihood on binarized MNIST and fMNIST. VIS-X-Y denotes $T = X$ refinement iterations during training and $T = Y$ refinement iterations during testing.

Method	**MNIST**	**fMNIST**
Results from (Titsias and Ruiz, 2019)		
UIVI	-94.09	-110.72
SIVI	-97.77	-121.53
VAE	-98.29	-126.73
Results from (Ruiz and Titsias, 2019)		
VCD	-95.86	-117.65
HMC-DLGM	-96.23	-117.74
This paper		
VIS-5-10	$\mathbf{-82.74 \pm 0.19}$	$\mathbf{-105.08 \pm 0.34}$
VIS-0-10	-96.16 ± 0.17	-120.53 ± 0.59
VAE (VIS-0-0)	-100.91 ± 0.16	-125.57 ± 0.63

Discussion We have proposed a flexible and efficient framework to perform inference in probabilistic programs defining wide classes of models. Our framework can be seen as a general way of tuning SG-MCMC sampler parameters, adapting the initial distributions and the learning rate. Key to the success and applicability of the VIS framework are the approximations introduced for the intractable parts of the refined variational approximations, which are computationally cheap but convenient.

Acknowledgments

VG acknowledges support from grant FPU16-05034. DRI is grateful to the MINECO MTM2014-56949-C3-1-R project and the AXA-ICMAT Chair in Adversarial Risk Analysis. All authors acknowledge support from the Severo Ochoa Excellence Programme SEV-2015-0554. This material was based upon work partially supported by the National Science Foundation under Grant DMS-1638521 to the Statistical and Applied Mathematical Sciences Institute. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin. Continuous-time flows for efficient inference and density estimation, 2018a. URL https://openreview.net/forum?id=rkcya1ZAW.

Liqun Chen, Shuyang Dai, Yunchen Pu, Erjin Zhou, Chunyuan Li, Qinliang Su, Changyou Chen, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–669, 2018b.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. *arXiv preprint arXiv:1801.03558*, 2018.

David Duvenaud, Dougal Maclaurin, and Ryan Adams. Early stopping as nonparametric variational inference. In *Artificial Intelligence and Statistics*, pages 1070–1077, 2016.

Yihao Feng, Dilin Wang, and Qiang Liu. Learning to draw samples with amortized stein variational gradient descent. *arXiv preprint arXiv:1707.06626*, 2017.

Victor Gallego and David Rios Insua. Stochastic gradient mcmc with repulsive forces. *arXiv preprint arXiv:1812.00071*, 2018.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

James D Hamilton. State-space models. *Handbook of econometrics*, 4:3039–3080, 1994.

Matthew D Hoffman. Learning deep latent gaussian models with markov chain monte carlo. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1510–1519. JMLR. org, 2017.

Matthew D Hoffman, Pavel Sountsov, Joshua Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. 2018.

Ronald A Howard and James E Matheson. Influence diagrams. *Decision Analysis*, 2(3): 127–143, 2005.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Charles D Keeling. Atmospheric carbon dioxide record from mauna loa. 2005.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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.

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

Shuo-Hui Li and Lei Wang. Neural network renormalization group. *Phys. Rev. Lett.*, 121: 260601, Dec 2018. doi: 10.1103/PhysRevLett.121.260601. URL https://link.aps.org/doi/10.1103/PhysRevLett.121.260601.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1): 4873–4907, 2017.

Lawrence Murray, Daniel Lundén, Jan Kudlicka, David Broman, and Thomas B Schön. Delayed sampling and automatic rao-blackwellization of probabilistic programs. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. 2016.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

Omiros Papaspiliopoulos, Gareth O Roberts, and Martin Sköld. A general framework for the parametrization of hierarchical models. *Statistical Science*, pages 59–73, 2007.

Matthew Parno and Youssef Marzouk. Transport map accelerated markov chain monte carlo. *arXiv preprint arXiv:1412.5492*, 2014.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

Francisco Ruiz and Michalis Titsias. A contrastive divergence for combining variational inference and MCMC. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5537–5545, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/ruiz19a.html.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

Ross D. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 36 (4):589–604, 1988. doi: 10.1287/opre.36.4.589. URL https://doi.org/10.1287/opre.36.4.589.

Michalis K Titsias and Francisco Ruiz. Unbiased implicit variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 167–176, 2019.

Dustin Tran, Rajesh Ranganath, and David M Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. *arXiv preprint arXiv:1805.11183*, 2018.

Paul Zarchan and Howard Musoff. *Fundamentals of Kalman filtering: a practical approach*. American Institute of Aeronautics and Astronautics, Inc., 2013.

Appendix A. Related work

The idea of preconditioning the posterior distribution to speed up the mixing time of an MCMC sampler has recently been explored in (Hoffman et al., 2018) and (Li and Wang, 2018), where a reparameterization is learned before performing the sampling via HMC. Both papers extend seminal work of (Parno and Marzouk, 2014) by learning an efficient and expressive deep, non-linear transformation instead of a polynomial regression. However, they do not account for tuning the parameters of the sampler as we introduce in Section 2, where a fully, end to end differentiable sampling scheme is proposed.

The work of (Rezende and Mohamed, 2015) introduced a general framework for constructing more flexible variational distributions, called normalizing flows. These transformations are one of the main techniques to improve the flexibility of current VI approaches and have recently pervaded the literature of approximate Bayesian inference with current developments such as continuous-time normalizing flows (Chen et al., 2018a) which extend an initial simple variational posterior with a discretization of Langevin dynamics. However, they require a generative adversarial network (GAN) (Goodfellow et al., 2014) to learn the posterior, which can be unstable in high-dimensional spaces. We overcome this issue with the novel formulation stated in Section 2. Our framework is also compatible with different optimizers, not only those derived from Langevin dynamics. Other recent proposals to create more flexible variational posteriors are based on implicit approaches, which typically require a GAN (Huszár, 2017) or implicit schema such as UIVI (Titsias and Ruiz, 2019) or SIVI (Yin and Zhou, 2018). Our variational approximation is also implicit, but we use a sampling algorithm to drive the evolution of the density, combined with a Dirac delta approximation to derive an efficient variational approximation as we report on the extensive experiments in the Section 3.

Closely related to our framework is the work of Hoffman (2017), where a VAE is learned using HMC. We use a similar compound distribution as the variational approximation, though our framework allows for any SG-MCMC sampler (via the entropy approximation strategies introduced) and also the tuning of sampler parameters via gradient descent. Our work is also related to the recent idea of amortization of samplers (Feng et al., 2017). A common problem with these approaches is that they incur in an additional error, the amortization gap (Cremer et al., 2018). We alleviate this by evolving a set of particles z_i with a stochastic process in the latent space after learning a good initial distribution. Hence, the bias generated by the initial approximation is significantly reduced after several iterations of the process. A recent article related to our paper is (Ruiz and Titsias, 2019), who define a compound distribution similar to our framework. However, we focus on an efficient approximation using the reverse KL divergence, the standard and well understood divergence used in variational inference, which allows for tuning sampler parameters and achieving competitive results.

Appendix B. Supplementary results

B.1. Variational Autoencoder as a deep Bayes Classifier

With the final experiments we show that the VIS framework can deal with more general probabilistic graphical models. Influence diagrams (Howard and Matheson, 2005) are one of the most familiar representations of a decision analysis problem. There is a long history

Figure 2: Top row: original images. Bottom row: reconstructed images using VIS-5-10 at 10 epochs.

on bridging the gap between influence diagrams and probabilistic graphical models (see (Shachter, 1988), for instance), so developing better tools for Bayesian inference can be automatically used to solve influence diagrams.

We showcase the flexibility of the proposed scheme to solve inference problems in an experiment with a classification task in a high-dimensional setting. As dataset, the MNIST (LeCun et al., 1998) handwritten digit classification task is chosen, in which grey-scale 28×28 images have to be classified in one of the ten classes $\mathcal{Y} = \{0, 1, \ldots, 9\}$. More concretely, we extend the VAE model to condition it on a discrete variable y, leading to the conditional VAE (cVAE). A cVAE defines a decoder distribution $p_\theta(x|z, y)$ on an input space $x \in \mathbb{R}^D$ given class label $y \in \mathcal{Y}$ and latent variable $z \in \mathbb{R}^d$. To perform inference, a variational posterior is learned as an encoder $q_\phi(z|x, y)$ from a prior $p(z) \sim \mathcal{N}(0, I)$. Leveraging the conditional structure on y, we use the generative model as a classifier using Bayes rule:

$$\begin{aligned} p(y|x) \propto p(y)p(x|y) &= p(y) \int p_\theta(x|z, y)q_\phi(z|x, y)dz \\ &\approx \frac{1}{K} \sum_{k=1}^{K} p_\theta(x|z^{(k)}, y)p(y) \end{aligned} \tag{3}$$

where we use K Monte Carlo samples $z^{(k)} \sim q_\phi(z|x, y)$. In the experiments we set $K = 5$. Given a test sample x, the label $\hat{y}$ with highest probability $p(y|x)$ is predicted. Figure 5 in Appendix depicts the corresponding influence diagram. Additional details regarding the model architecture and hyperparameters can be found in Appendix C.

For comparison purposes, we perform various experiments changing T for the transition distribution $Q_{\eta,T}$ in the refined variational approximation. Results are in Table 3. We report the test accuracy achieved at the end of training. Note we are comparing different values of T depending on being on the training or testing phases (in the latter, where the model and variational parameters are kept frozen). The model with $T_{tr} = 5$ was trained for 10 epochs, whereas the other settings for 15 epochs, in order to give all settings similar training times. Results are averaged from 3 runs with different random seeds. From the results it is clear that the effect of using the refined variational approximation (the cases when $T > 0$) is crucially beneficial to achieve higher accuracy. The effect of learning a good initial distribution and inner learning rate by using the gradients $\nabla_\phi \text{ELBO}(q)$ and $\nabla_\eta \text{ELBO}(q)$ has a highly positive impact in the accuracy obtained.

On a final note, we have not included the case when only using a SGD or SGLD sampler (i.e., without learning an initial distribution $q_{0,\phi}(z|x)$) since the results were much worse than the ones in Table 3, for a comparable computational budget. This strongly suggests that for

Table 3: Results on digit classification task using a deep Bayes classifier.

T_{tr}	T_{te}	Acc. (test)
0	0	96.5 ± 0.5 %
0	10	97.7 ± 0.7 %
5	10	$\mathbf{99.8 \pm 0.2}$ %

inference in high-dimensional, continuous latent spaces, learning a good initial distribution through VIS can dramatically accelerate mixing time.

B.2. State-space Markov models

We test our variational approximation on two state-space models, one for discrete data and the other for continuous observations. All the experiments in this subsection use the Fast AD version from Section D.3 since it was not necessary to further tune the sampler parameters to have competitive results.

Hidden Markov Model (HMM) The model equations are given by

$$p(z_{1:\tau}, x_{1:\tau}, \theta) = \prod_{t=1}^{\tau} p(x_t|z_t, \theta_{em}) p(x_t|x_{t-1}, \theta_{tr}) p(\theta),$$

where each conditional is a Categorical distribution which takes 5 different classes and the prior $p(\theta) = p(\theta_{em})p(\theta_{tr})$ are two Dirichlet distributions that sample the emission and transition probabilities, respectively. We perform inference on the parameters θ.

Dynamic Linear Model (DLM) The model equations are the same as in the HMM case, though the conditional distributions are now Gaussian and the parameters θ refer to the emission and transition variances. As before, we perform inference over θ.

The full model implementations can be checked in Appendix C.1, based on `funsor`[1], a PPL on top of the `Pytorch` autodiff framework. For each model, we generate a synthetic dataset, and use the refined variational approximation with $T = 0, 1, 2$. As the original variational approximation to the parameters θ we use a Dirac Delta. Performing VI with this approximation corresponds to MAP estimation using the Kalman filter in the DLM case (Zarchan and Musoff, 2013) and the Baum-Welch algorithm in the HMM case (Rabiner, 1989), since we marginalize out the latent variables $z_{1:\tau}$. Model details are given in Appendix C.1.1. Figure 3 shows the results. The first row reports the experiments related to the HMM; the second one to the DLM. While in all graphs we report the evolution of the loglikelihood during inference, in the first column we report the number of ELBO iterations, whereas in the second column we measure wall-clock time as the optimization takes place. We confirm that VIS ($T > 0$) achieve better results than regular optimization with VI ($T = 0$) for a similar amount of time.

1. https://github.com/pyro-ppl/funsor/

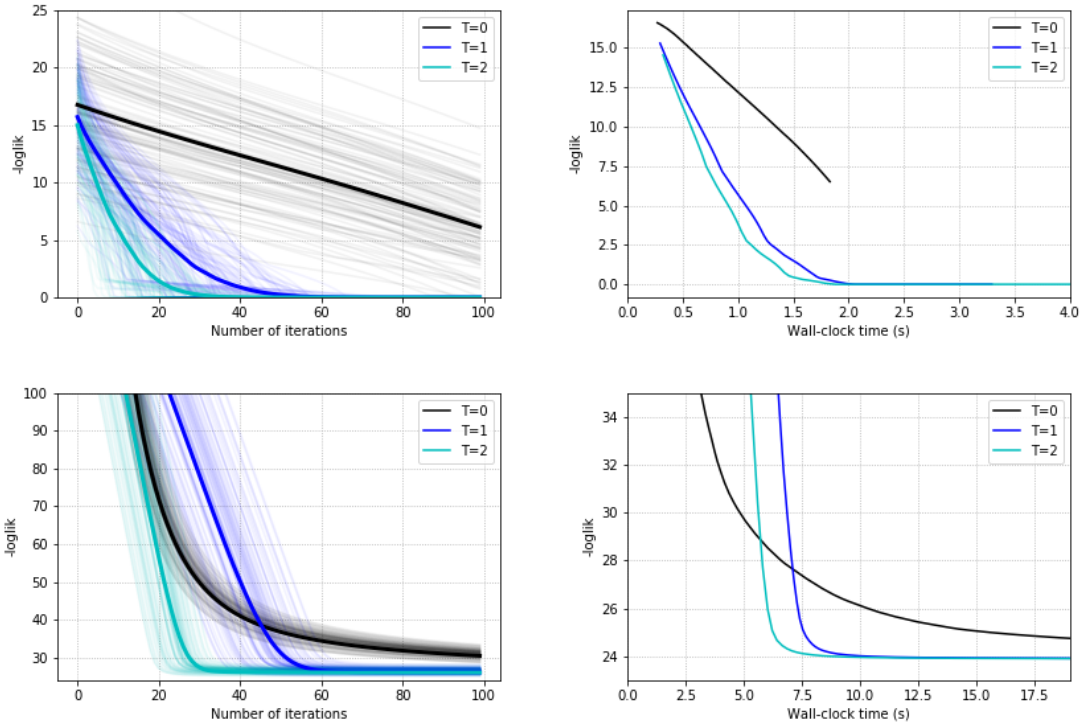

Figure 3: Results of ELBO optimization for state-space models. Top left (HMM): -loglikelihood against number of ELBO gradient iterations. Top right (HMM): -loglikelihood against wall-clock time. Bottom left (DLM): -loglikelihood against number of ELBO gradient iterations. Bottom right (DLM): -loglikelihood against number of ELBO gradient iterations

B.2.1. Prediction tasks in a HMM

With the aim of assessing whether ELBO optimization helps in attaining better auxiliary scores, we also report results on a prediction task. We generate a synthetic time series of alternating 0 and 1 for $\tau = 105$ timesteps. We train the HMM model from before on the first 100 points, and report in Table 4 the accuracy of the predictive distribution $p(y_t)$ averaged over the last 5 time-steps. We also report the predictive entropy since it helps in assessing the confidence of the model in its forecast and is a strictly proper scoring rule (Gneiting and Raftery, 2007). To guarantee the same computational budget time and a fair comparison, the model without refining is run with 50 epochs, whereas the model with refinement is run for 20 epochs. We see that the refined model achieves higher accuracy than its counterpart; in addition it is correctly more confident in its predictions.

Table 4: Prediction metrics for the HMM.

	$T = 0$	$T = 1$
accuracy	0.40	**0.84**
predictive entropy	1.414	**1.056**
logarithmic score	-1.044	$\mathbf{-0.682}$

Appendix C. Experiment details

C.1. State-space models

C.1.1. Initial experiments

For the HMM, both the emission and transition probabilities are Categorical distributions, taking values in the domain $\{0, 1, 2, 3, 4\}$.

The equations of the DLM are given by

$$\begin{aligned} z_{t+1} &\sim \mathcal{N}(0.5z_t + 1.0, \sigma_{tr}) \\ x_t &\sim \mathcal{N}(3.0z_t + 0.5, \sigma_{em}). \end{aligned}$$

with $z_0 = 0.0$.

C.1.2. Prediction task in a DLM

The DLM model is comprised of a linear trend component plus a seasonal block of period 12. The trend is specified as

$$\begin{aligned} x_t &= \mu_t + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_{obs}) \\ \mu_t &= \mu_{t-1} + \delta_{t-1} + \epsilon'_t \qquad \epsilon'_t \sim \mathcal{N}(0, \sigma_{level}) \\ \delta_t &= \delta_{t-1} + \epsilon''_t \qquad \epsilon''_t \sim \mathcal{N}(0, \sigma_{slope}). \end{aligned}$$

With respect to the seasonal component, the main idea is to *cycle the state*: suppose $\theta_t \in \mathbb{R}^p$, with p being the seasonal period. Then, at each timestep, the model focuses on the first component of the state vector:

$$\underset{\uparrow}{(\alpha_1, \alpha_2, \ldots, \alpha_p)} \xrightarrow{\text{next period}} \underset{\uparrow}{(\alpha_2, \alpha_3, \ldots, \alpha_p, \alpha_1)}.$$

Thus, we can specify the seasonal component via:

$$\begin{aligned} x_t &= F\theta_t + v_t \\ \theta_t &= G\theta_{t-1} + w_t \end{aligned}$$

where F is a $p-$dimensional vector and G is a $p \times p$ matrix such that

$$G = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & & 1 & 0 \end{bmatrix}$$

and $F = (1, 0, \ldots, 0, 0)$.

C.2. VAE

C.2.1. Model details

```
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        self.z_d = 10
        self.h_d = 200
        self.x_d = 28*28

        self.fc1_mu = nn.Linear(self.x_d, self.h_d)
        self.fc1_cov = nn.Linear(self.x_d, self.h_d)
        self.fc12_mu = nn.Linear(self.h_d, self.h_d)
        self.fc12_cov = nn.Linear(self.h_d, self.h_d)
        self.fc2_mu = nn.Linear(self.h_d, self.z_d)
        self.fc2_cov = nn.Linear(self.h_d, self.z_d)

        self.fc3 = nn.Linear(self.z_d, self.h_d)
        self.fc32 = nn.Linear(self.h_d, self.h_d)
        self.fc4 = nn.Linear(self.h_d, self.x_d)

    def encode(self, x):
        h1_mu = F.relu(self.fc1_mu(x))
        h1_cov = F.relu(self.fc1_cov(x))
        h1_mu = F.relu(self.fc12_mu(h1_mu))
        h1_cov = F.relu(self.fc12_cov(h1_cov))
        # we work in the logvar-domain
        return self.fc2_mu(h1_mu),
        torch.log(F.softplus(self.fc2_cov(h1_cov)))

    def decode(self, z):
        h3 = F.relu(self.fc3(z))
        h3 = F.relu(self.fc32(h3))
        return torch.sigmoid(self.fc4(h3))
```

Figure 4: Model architecture for the VAE.

The VAE model is implemented with PyTorch (Paszke et al., 2017). The prior distribution $p(z)$ for the latent variables $z \in \mathbb{R}^{10}$ is a standard factorized Gaussian. The decoder distribution $p_\theta(x|z)$ and the encoder distribution (initial variational approximation) $q_{0,\phi}(z|x)$ are parameterized by two feed-forward neural networks whose details can be checked in Figure 4.

C.2.2. Hyperparameter settings

The optimizer Adam is used in all experiments, with a learning rate $\lambda = 0.001$. We also set $\eta = 0.001$. We train for 15 epochs (fMNIST) and 20 epochs (MNIST), in order to achieve similar performance to the explicit VAE case in (Titsias and Ruiz, 2019). For the VIS-5-10 setting, we train for only 10 epochs, to allow for a fair computational comparison (similar computing times).

C.3. CVAE

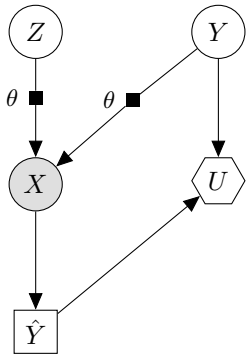

Figure 5: Influence Diagram for the deep Bayes classifier.

C.3.1. Model details

```
class cVAE(nn.Module):
    def __init__(self):
        super(cVAE, self).__init__()

        self.z_d = 10
        self.h_d = 200
        self.x_d = 28*28
        num_classes = 10

        self.fc1_mu = nn.Linear(self.x_d + num_classes, self.h_d)
        self.fc1_cov = nn.Linear(self.x_d + num_classes, self.h_d)
        self.fc12_mu = nn.Linear(self.h_d, self.h_d)
        self.fc12_cov = nn.Linear(self.h_d, self.h_d)
        self.fc2_mu = nn.Linear(self.h_d, self.z_d)
        self.fc2_cov = nn.Linear(self.h_d, self.z_d)

        self.fc3 = nn.Linear(self.z_d + num_classes, self.h_d)
        self.fc32 = nn.Linear(self.h_d, self.h_d)
        self.fc4 = nn.Linear(self.h_d, self.x_d)

    def encode(self, x, y):
        h1_mu = F.relu(self.fc1_mu(torch.cat([x, y], dim=-1)))
        h1_cov = F.relu(self.fc1_cov(torch.cat([x, y], dim=-1)))
        h1_mu = F.relu(self.fc12_mu(h1_mu))
        h1_cov = F.relu(self.fc12_cov(h1_cov))
        # we work in the logvar-domain
        return self.fc2_mu(h1_mu),
        torch.log(F.softplus(self.fc2_cov(h1_cov)))

    def decode(self, z, y):
        h3 = F.relu(self.fc3(torch.cat([z, y], dim=-1)))
        h3 = F.relu(self.fc32(h3))
        return torch.sigmoid(self.fc4(h3))
```

Figure 6: Model architecture for the cVAE.

The cVAE model is implemented with PyTorch (Paszke et al., 2017). The prior distribution $p(z)$ for the latent variables $z \in \mathbb{R}^{10}$ is a standard factorized Gaussian. The decoder distribution $p_\theta(x|y,z)$ and the encoder distribution (initial variational approximation) $q_{0,\phi}(z|x,y)$ are parameterized by two feed-forward neural networks whose details can be checked in Figure 6. The integral (3) is approximated with 1 MC sample from the variational approximation in all experimental settings.

C.3.2. Hyperparameter settings

The optimizer Adam is used in all the experiments, with a learning rate $\lambda = 0.01$. We set the initial $\eta = 5e-5$.

Appendix D. Analysis of VIS

In this Section we study in detail key properties of the proposed VIS framework.

D.1. Rewriting the ELBO, using particle approximation

Performing variational inference with the refined variational approximation can be regarded as using the original variational guide while optimizing an alternative, tighter ELBO. Note that for a refined guide of the form $q(z|z_0)q(z_0|x)$, the objective function can be written as

$$\mathbb{E}_{q(z|z_0)q(z_0|x)}\left[\log p(x,z) - \log q(z|z_0) - \log q(z_0|x)\right].$$

However, using the Dirac Delta approximation for $q(z|z_0)$ and noting that $z = z_0 + \eta\nabla \log p(x,z_0)$ when using SGD with $T=1$, we arrive at the modified objective:

$$\mathbb{E}_{q(z_0|x)}\left[\log p(x, z_0 + \eta\nabla \log p(x,z_0)) - \log q(z_0|x)\right]$$

which is equivalent to the refined ELBO introduced in (2). Since we are perturbing the latent variables in the steepest ascent direction, it is straightforward to show that, for moderate η, the previous bound is tighter than the one, for the original variational guide $q(z_0|x)$, $\mathbb{E}_{q(z_0|x)}\left[\log p(x,z_0) - \log q(z_0|x)\right]$. This reformulation of ELBO is also convenient since it provides a clear way of implementing our refined variational inference framework in any PPL supporting algorithmic differentiation.

D.2. Taylor expansion

From the result in subsection D.1, we can further restrict to the case when the original variational approximation is also a Dirac point mass. Then, the original ELBO optimization resorts to the standard maximum likelihood estimation, i.e., $\max_z \log p(x,z)$. Within the VIS framework, we optimize instead $\max_z \log p(x, z+\Delta z)$, where Δz is one iteration of the sampler, i.e., $\Delta z = \eta\nabla \log p(x,z)$ in the SGD case. For notational clarity we resort to the case $T=1$, but a similar analysis can be straightforwardly done if more refinement steps are performed.

We may now perform a first-order Taylor expansion of the refined objective as

$$\log p(x, z+\Delta z) \approx \log p(x,z) + (\Delta z)^\intercal \nabla \log p(x,z).$$

Taking gradients of the first order approximation w.r.t. the latent variables z we arrive at

$$\nabla_z \log p(x,z) + \eta \nabla_z \log p(x,z)^\intercal \nabla_z^2 \log p(x,z),$$

where we have not computed the gradient through the Δz term. That is, the *refined gradient* can be deemed as the original gradient plus a second order correction. Instead of being modulated by a constant learning rate, this correction is adapted by the chosen sampler. In the experiments in Section B.1 we show that this is beneficial for the optimization as it can take less iterations to achieve lower losses. By further taking gradients through the Δz term, we may tune the sampler parameters such as the learning rate as described in Section 2.3. Consequently, the next subsection describes both modes of differentiation.

D.3. Two modes of Automatic Differentiation for ELBO optimization

Here we describe how to implement two variants of the ELBO objective. First, we define a *stop gradient* operator[2] $\perp$ that sets the gradient of its operand to zero, i.e., $\nabla_x \perp(x) = 0$ whereas in the forward pass it acts as the identity function, that is, $\perp(x) = x$. Then, the two variants of the ELBO objective are

$$\mathbb{E}_q \left[\log p(x, z + \Delta z) - \log q(z + \Delta z | x)\right] \qquad \text{(Full AD)}$$

and

$$\mathbb{E}_q \left[\log p(x, z + \perp(\Delta z)) - \log q(z + \perp(\Delta z) | x)\right]. \qquad \text{(Fast AD)}$$

The Full AD ELBO makes it possible to further compute a gradient wrt sampler parameters inside Δz at the cost of a slight increase in the computational burden.

Appendix E. State-space model specialization

The previous framework is particularly useful in large families of state-space models (and by extension, models that exhibit hierarchical and/or temporal structure), mainly through two complementary strategies: i) exact marginalization of some particular terms (i.e., Rao-Blackwellization (Murray et al., 2018) to reduce the variance); ii) exact computation in linear cases. Recall that a state-space model (Hamilton, 1994) can be expressed with the following probabilistic model, where the time-step t iterates from 1 to τ:

$$\begin{aligned} z_{t+1} &\sim p(z_{t+1} | z_t, \theta_{tr}), \\ x_{t+1} &\sim p(x_{t+1} | z_{t+1}, \theta_{em}). \end{aligned}$$

This formulation subsumes many models used in Machine Learning such as Hidden Markov Models (HMMs) or Dynamic Linear Models (DLMs). It is often required to perform inference on the $\theta := (\theta_{em}, \theta_{tr})$ parameters from the transition and emission equations, respectively. We propose to use a variational distribution $q(\theta)$, which will be refined by any sampling method (as described in Section 2.1):

$$\theta \leftarrow \theta + \nabla_\theta \log p(x_{1:\tau} | z_{1:\tau}, \theta) + \xi. \qquad (4)$$

2. corresponds to `detach` in Pytorch or `stop_gradient` in tensorflow.

Note that for a large class of models (including HMMs and DLMs) we can marginalize out $z_{1:\tau}$ and have reduced variance iterating with:

$$\theta \leftarrow \theta + \nabla_\theta \log p(x_{1:\tau}|\theta) + \xi, \tag{5}$$

where the latent variables $z_{1:\tau}$ have been marginalized out using the sum-product algorithm. For linear-Gaussian models we can also compute the exact form of the refined posterior, since all terms in Eq. 5 are linear wrt the latent variables θ. However, inference in these linear models is exact by using conjugate distributions, so the proposed framework is more fit to the case of state-space models containing non-linear (or non-conjugate) components. For these families of models, we resort to use just a gradient estimator of the entropy or the Delta approximation in Section 2.1.