Facilitating Multimodality in Normalizing Flows

Chin-Wei Huang*   David Krueger*   Aaron Courville
Montreal Institute for Learning Algorithms (MILA)
chin-wei.huang@umontreal.ca david.krueger@umontreal.ca
courvila@iro.umontreal.ca

Abstract

The true Bayesian posterior of a model such as a neural network may be highly multimodal. In principle, normalizing flows can represent such a distribution via compositions of invertible transformations of random noise. In practice, however, existing normalizing flows may fail to capture most of the modes of a distribution. We argue that the conditionally affine structure of the transformations used in [Dinh et al., 2014, 2016, Kingma et al., 2016] is inefficient, and show that flows which instead use (conditional) invertible non-linear transformations naturally enable multimodality in their output distributions. With just two layers of our proposed deep sigmoidal flow, we are able to model complicated 2d energy functions with much higher fidelity than six layers of deep affine flows.

1 Introduction

Normalizing flows [Rezende and Mohamed, 2015] apply a series of parametrized invertible transformations which reshape the probability distribution of their inputs. Due to computational concerns, the authors limit the flows to a scalar bottleneck which allows the use of the matrix determinant lemma. Dinh et al. [2014, 2016], Kingma et al. [2016] instead choose to manipulate the order of conditioning, which also allows for linear time computation of the determinant of jacobian.

In contrast, we note that a flow can be formed via any (arbitrarily conditioned) *invertible* mapping, and propose a non-linear deep **sigmoidal flow**. The transformations in deep sigmoidal flows resemble neural networks with parameter constraints which ensure they are monotonic in their inputs. Such transformations can represent the inverse CDF of a density function if the base distribution is chosen to be uniform, hence resembling the inverse transform sampling method. But conventionally we consider using standard normal as base distribution [1] and transform and random noise sampled from it. While composing affine flows allows for modeling of multi-modal distributions, sigmoidal flows allow for a much more natural expression of multi-modality.

As suggested in [Huang et al., 2017], sigmoidal flows are potential universal approximators for any multivariate density function. This property could be used to establish the asymptotic unbiasedness of deep latent Gaussian models such as Variational Autoencoders [Kingma and Welling, 2013]. It also suggests that sigmoidal flows could provide a better model of the posterior over model parameters in Bayesian treatments of deep neural networks [Krueger et al., 2017].

2 Deep Sigmoidal Flow

We now describe a single transformation (of a single input variable, x_j) by a deep sigmoidal flow, which resembles a deep neural network with $2L$ layers. Carrying forward the analogy, the "pa-

[1] We also tried multimodal base distribution and pass it through the inverse autoregressive transformation with affine functions but did not find it to be better.

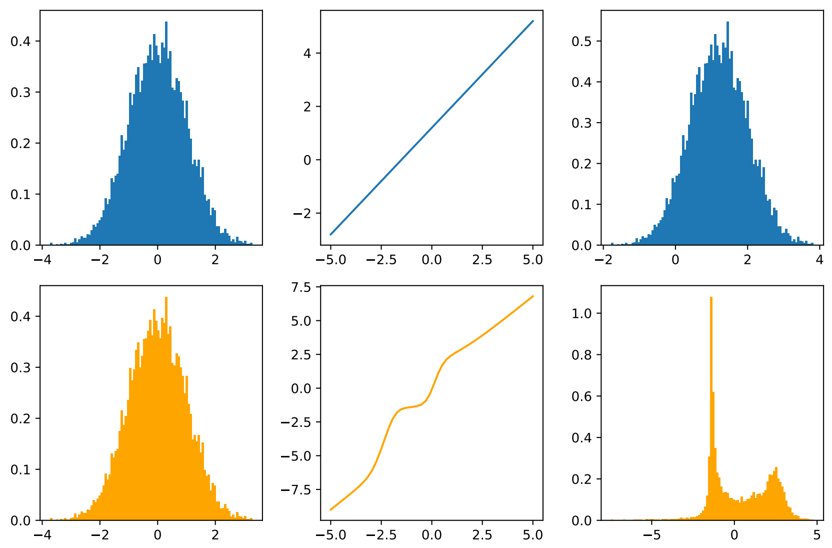

Figure 1: Illustration of invertible mapping's effect on normal distribution (left). If the source normal distribution is passed through an affine map (top center), the output distribution remains a normal distribution (top right). When a deep sigmoidal flow (bottom center) is used, multiple modes can be introduced (bottom right).

rameters" of this network, $(a^l, b^l, w^l, m^l)_{l=1}^L$, are all produced as a function of the preceding input variables $x_1, ..., x_{j-1}$. Defining $h^0 = x_j$, we have

$$h^l = \text{Logit}(\sum_{k=1}^{c^l} w^{lk} \sigma(a^{lk} \cdot h^{l-1} + b^{lk})) + m^l \tag{1}$$

where $0 < w^{lj} < 1$, $\sum_j w^{lj} = 1$, $a^{lj} > 0$[2], $\text{Logit}(\xi) \doteq \log \frac{\xi}{1-\xi}$, $x'_j = h_L$, and c^l controls the number of modes conditioned on the preceding input variables. Since a^{lj} and w^{lj} are both positive, h^l is a composition of monotonic functions (including σ) and is therefor monotonic as function of h^{l-1} (and hence invertible). Returning to the deep network description of this transformation, we can view the computation of h^l as composing projected onto a layer of c^l sigmoid units (representing different localized contractions, which push down probability mass as depicted in Figure 1), followed by an inner product with w^l, yielding a CDF for a mixture of logistic distributions. Finally, this CDF is "inverted" into a bijection from $[0, 1]$ to $\mathbb{R}$ via the Logit operation.

Efficient training. Having defined an invertible transformation, we can use the change of variables formula to compute the density of the transformed variable (x') in terms of the original variable's density, as typically done with normalizing flows: $p_{x'}(x') = p_x(x)|\det \frac{\partial x'}{\partial x}|^{-1}$. Assume we are given an energy function that we seek to model $U(x) : \mathbb{R}^D \to \mathbb{R}^+$, and that our objective is the exclusive KL-divergence. Then estimation of the loss function can be carried out through the Law of Unconscious Statistician and Monte Carlo Sampling:

$$\mathcal{KL}(p_{x'}||U) = \mathbf{E}_{p_{x'}}[\log p_{x'}(x') - \log U(x')] = \mathbf{E}_{p_x}[\log p_x(x) - \log|\det \frac{\partial x'}{\partial x}| - \log U(x')] \tag{2}$$

Comparison to Mixture Proposals. Interestingly, differentiating the input to the Logit function w.r.t. x_j in (1) yields the pdf of a mixture of logistic distributions. This can be interpreted as parameterizing the CDF of the mixture density function using a deep neural network to fit the inverse integral of any density function. Intuitively, a mixture proposal with a sufficient number of components can asymptotically approximate any target density, but in practice one needs to either marginalize out all the components (which is expensive) or sample from a specific mode and update

[2]Constraints on the variables are enforced via activation functions; w^{lj} and a^{lj} are outputs of a softmax and softplus, respectively.

its sufficient parameters independently (which introduces a lot of variance). Our method, on the other hand, uses a neural network to define a flexible monotone function and updates the parameters through efficient Monte Carlo sampling to carve up the correct density function.

Comparison to Affine Transformations. Unlike the affine transformation constructions used in NICE and IAF, Equation 1 creates "humps" through superpositions of sigmoid functions at different locations. Multimodality can easily be introduced through such a transformation, as shown by Figure 1. It is important to note that through compositions of transformations (suppose we are considering multivariate models and ordering of conditioning is different for each transformation to allow all dimensions to influence each other), conditional affine flows can potentially introduce multimodality. Empirically, we found that it relies heavily on initialization and tuning the optimizer, and still does not always capture multiple modes. Using the non-affine flow allows us to capture multiple modes more easily, even without needing to tune the hyperparameters of optimization carefully.

3 Experiments

3.1 Fitting Multimodal Energy Functions

We visualize the ability of sigmoidal flows to fit complicated energy functions, including those used by Rezende and Mohamed [2015]. We find that sigmoidal, but not affine, flows are able to capture multi-modality, and sigmoidal flows perform better overall, both qualitatively and quantitatively. We set $c_l = 1$ in all experiments, which is essentially equivalent to a one-layer MLP with constraints on weights.

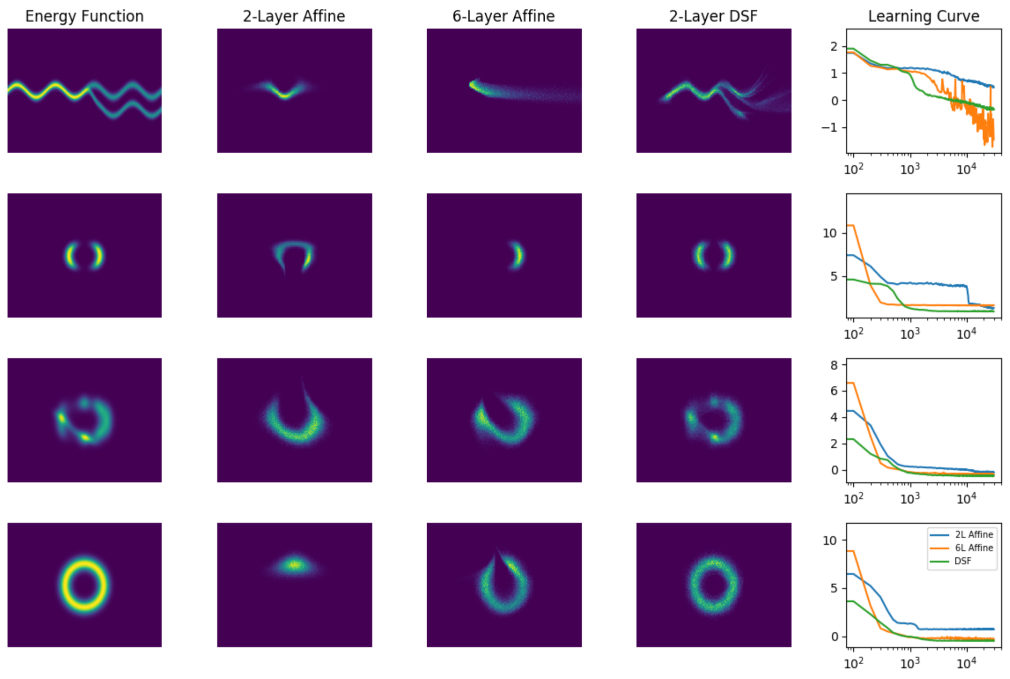

Figure 2: Comparison of affine flows with deep sigmoidal flows (DSF) on four synthetic energy functions (rows). With just 2 layers of transformations, the DSF already captures the energy functions with much higher fidelity than 6 layers of NICE affine flow transformations. The right column shows KL-divergence learning curves. The DSF generally learns somewhat faster and ultimately outperforms the affine flows in terms of KL-divergence in three out of four cases.

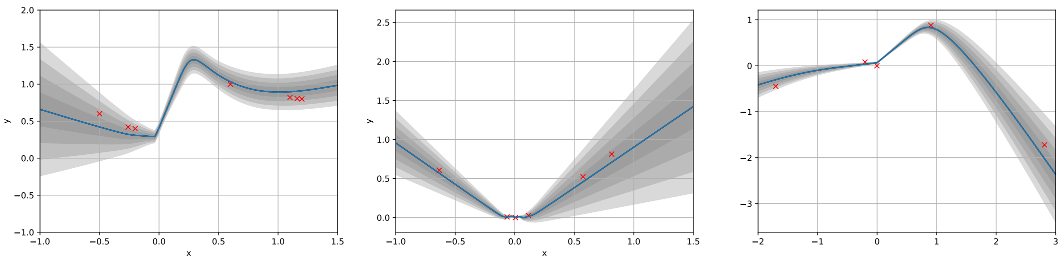

Figure 3: Deep Sigmoidal Flows as approximate posterior of Bayesian DNNs. We show the mean statistic and half standard deviation of the posterior predictive mean. Red crosses are training examples.

3.2 1-D Regression

We follow the Bayesian Hypernetworks [Krueger et al., 2017] and demonstrate using DSF allows us to capture the posterior over the weight parameters of a neural net. For these experiments, we choose ReLU as nonlinear activation. We use one hidden layer of MLP with 64 hidden nodes.

References

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Chin-Wei Huang, Ahmed Touati, Laurent Dinh, Michal Drozdzal, Mohammad Havaei, Laurent Charlin, and Aaron Courville. Learnable explicit density for continuous latent space and variational inference. *arXiv preprint arXiv:1710.02248*, 2017.

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

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.