Supplementary Materials for Latent Normalizing Flows for Discrete Sequences

Zachary M. Ziegler [1] **Alexander M. Rush** [1]

A. Invertible discrete mappings

It is interesting to consider how one might directly apply flows to discrete sequences. We begin with the discrete change of variables formula: for $X \in \Omega_x^D, Y \in \Omega_y^D$ (with Ω_x, Ω_y finite), base density $p_X(\boldsymbol{x})$, and deterministic function $\boldsymbol{y} = f(\boldsymbol{x})$,

$$p_Y(\boldsymbol{y}) = \sum_{\boldsymbol{x} \in f^{-1}(\boldsymbol{y})} p_X(\boldsymbol{x})$$

If f is invertible, as is required for the flow framework, this reduces to

$$p_Y(\boldsymbol{y}) = p_X(f^{-1}(\boldsymbol{y}))$$

First, examine the simplest case where $D = 1$ and $X \in \Omega_x$, $Y \in \Omega_y$ i.e. x, y are just single elements of a set. In this case, invertible functions can only be found if $|\Omega_x| = |\Omega_y|$, so without loss of generality we can rename elements such that $\Omega_x = \Omega_y = \Omega$. Thus we are interested in invertible functions $f : \Omega \to \Omega$. By definition, a permutation of Ω is any invertible mapping from Ω to itself (Nering, 1970). We conclude that when $D = 1$, the only possible invertible functions are permutations. As permutations do not permit a parameterized changing of densities, a normalizing flow cannot be used in the 1D case to define a useful distribution.

This is not just a theoretical result, consider the following example: we are interested in learning the distribution of the first word in a set of documents. In this case Ω would be the vocabulary of possible words, y represents a word, and we would like to use a discrete flow to model $p(y)$. We pick an uninformative base density such as the uniform distribution. According to the result above, a flow cannot learn the distribution $p(y)$, whereas simply counting would model the distribution well.

Even if we were to choose a different distribution, say a geometric distribution in some order, the flow could at best find a permutation of the geometric probabilities that best matches the true distribution. Clearly this is an undesirable optimum.

In the more general case where $D > 1$, non-permutation invertible mappings certainly exist. A common example is the XOR function. Therefore, it is in principle possible to use flows to model data when $D > 1$, and future work should investigate the limits of this approach. Given that the 1D case fails, however, it will be important to understand how this failure relates to the higher dimensional cases. Is this simply an unfortunate edge case? Or is it indicative of a larger limitation?

B. Proposed flow validity

A transformation function $f : \mathcal{R}^D \to \mathcal{R}^D$ represents a valid normalizing flow if f is invertible. A transformation function f represents a *useful* normalizing flow is the Jacobian of f can be computed with linear complexity in dimension of the data. We show that the three proposed flows in this work have both of these properties.

First consider the AF / AF flow, defined by transformation function f_θ:

$$\boldsymbol{z}_t = f_{\text{AF}}(\boldsymbol{\epsilon}_t; \boldsymbol{z}_{<t}, \theta)$$

To prove the mapping is invertible it suffices to find the inverse:

$$\boldsymbol{\epsilon}_t = f_{\text{AF}}^{-1}(\boldsymbol{z}_t; \boldsymbol{z}_{<t}, \theta)$$

f_{AF} is a normalizing flow and therefore an invertible function. Each $\boldsymbol{\epsilon}_t$ can thus be calculated from $\boldsymbol{z}_{1:T}$ giving $\boldsymbol{\epsilon}_{1:T} = f_\theta^{-1}(\boldsymbol{z}_{1:T})$.

For the latent flows considered in the main text $\boldsymbol{z} \in \mathcal{R}^{T \text{x} H}$. Here we equivalently view $\boldsymbol{z}$ as a large $D = T \cdot H$ vector. We write $\boldsymbol{z} = \boldsymbol{z}_{1:T} = \{\boldsymbol{z}_1, ..., \boldsymbol{z}_T\} = \{z_{1,1}, ..., z_{1,H}, z_{2,1} ..., z_{2,H}, ..., z_{T,1}, ..., z_{T,H}\}$. In this case the Jacobian matrix $\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}}$ can be written as a block matrix

$$\begin{bmatrix} \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{\epsilon}_1} & \cdots & \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{\epsilon}_T} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{z}_T}{\partial \boldsymbol{\epsilon}_1} & \cdots & \frac{\partial \boldsymbol{z}_T}{\partial \boldsymbol{\epsilon}_T} \end{bmatrix}$$

where each block $\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{\epsilon}_s}$ is a $H \text{x} H$ Jacobian matrix.

For the AF / AF flow $\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{\epsilon}_s} = \mathbf{0}; s > t$ because $\boldsymbol{z}_t$ depends only on $\boldsymbol{\epsilon}_t$ and $\boldsymbol{z}_{<t}$, which itself only depends on $\boldsymbol{\epsilon}_{<t}$. There-

fore the Jacobian matrix is block triangular with determinant

$$\left|\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{\epsilon}}\right| = \prod_{t=1}^{T} \left|\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{\epsilon}_t}\right| = \prod_{t=1}^{T} \left|\frac{\partial f_{AF}}{\partial \boldsymbol{\epsilon}_t}\right|$$

Thus, the Jacobian determinant is simply the product of the Jacobian determinants of the AF-in-hidden transformations at each time step. (Papamakarios et al., 2017) show that the Jacobian determinant is linear in H for AF, thus the overall complexity for the determinant calculation of AF / AF is $\mathcal{O}(TH) = \mathcal{O}(D)$.

The proof holds when f_{AF} is replaced with f_{SCF}, as (Dinh et al., 2017) show that the Jacobian of f_{SCF} can be computed with linear complexity. This concludes the proof that AF / AF and AF / SCF are valid normalizing flows with Jacobian determinant calculations linear in the data dimension.

For IAF / SCF the transformation function pair is:

$$\boldsymbol{z}_t = f_{\text{SCF}}(\boldsymbol{\epsilon}_t; \boldsymbol{\epsilon}_{<t}, \theta), \quad \boldsymbol{\epsilon}_t = f_{\text{SCF}}^{-1}(\boldsymbol{z}_t; \boldsymbol{\epsilon}_{<t}, \theta)$$

This is invertible because an inverse function is found. $\frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{\epsilon}_s} = \mathbf{0}; s > t$ because $\boldsymbol{z}_t$ depends only on $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\epsilon}_{<t}$. The Jacobian matrix is thus block triangular with determinant $\prod_{t=1}^{T} \left|\frac{\partial f_{SCF}}{\partial \boldsymbol{\epsilon}_t}\right|$. The same argument as for AF / AF gives a Jacobian determinant complexity of $\mathcal{O}(TH) = \mathcal{O}(D)$.

C. NLSq invertibility

The NLSq function is

$$f(\epsilon) = z = a + b\epsilon + \frac{c}{1 + (d\epsilon + g)^2} \tag{1}$$

In the following discussion we assume $b > 0, d > 0$ A real scalar function is invertible if its derivative is positive everywhere.

$$f'(\epsilon) = b - \frac{2cd(d\epsilon + g)}{(1 + (d\epsilon + g)^2)^2}$$

Taking another derivative and setting it equal to 0 gives the critical points $\epsilon^* = (g \pm \sqrt{1/3})/d$. The distinction between maximum and minimum depends on the sign of c. In either case, the minimum slope is

$$f'(\epsilon^*) = b - \frac{9}{8\sqrt{3}}|c|d$$

Thus invertibility is guaranteed if $b > \frac{9}{8\sqrt{3}}|c|d$. In our implementation $a = a$, $g = g$, $b = e^{b'}$, $d = e^{d'}$, and $c = \frac{8\sqrt{3}}{9d} b\alpha \cdot \tanh(c')$, where a, b', c', d', g are unrestricted and output from the model, and $0 < \alpha < 1$ is a constant included for stability. We found $\alpha = 0.95$ allows significant freedom of the perturbation while disallowing "barely invertible" functions.

The inverse of the NLSq function is analytically computable, which is important for efficient generation. Solving for ϵ in Eq. 1 gives the cubic equation

$$\begin{aligned} &-bd^2\epsilon^3 + ((z-a)d^2 - 2dgb)\epsilon^2 \\ &+(2dg(z-a) - b(g^2+1))\epsilon \\ &+((z-a)(g^2+1) - c) = 0 \end{aligned}$$

Under the invertibility condition above this is guaranteed to have one real root which can be found analytically (G. C. Holmes, 2002).

In practice, because the forward direction as written (applying $f(\epsilon)$) requires fewer operations it is used for the reverse function $f^{-1}(z)$, and the solution to the cubic equation is used for the forward function $f(\epsilon)$.

D. Variable length input

When working with non-autoregressive models we need to additionally deal with the variable length nature of the observed sequences. Unlike autoregressive models, which can emit an end-of-sentence token, non-autoregressive models require the length to be sampled initially. Given a sequence of length T we can write

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|T')p(T')\mathrm{d}T' = p(\boldsymbol{x}|T)p(T)$$

where the second equality comes from the fact that $p(\boldsymbol{x}|T') = 0$ for $T' \neq T$. For unconditional sequence modeling we can use the empirical likelihood for $p(T)$, and then condition all parts of the model itself on T. In this work we implement the conditioning as a two one-hot vectors at every timestep t, indicating the distance from the beginning and end of the sequence. Compared to other popular position encodings in the literature, such as the one commonly used in the Transformer (Vaswani et al., 2017), this primarily encodes the absolute length T instead of the relative position between tokens needed in a self-attention based architecture.

The generative process becomes:

$$\begin{aligned} T &\sim p(T) \\ \boldsymbol{\epsilon} &\sim p_\epsilon(\boldsymbol{\epsilon}) \\ \boldsymbol{z} &= f_\theta(\boldsymbol{\epsilon}; T) \\ \boldsymbol{x} &\sim p(\boldsymbol{x}|\boldsymbol{z}, T) \end{aligned}$$

E. Implementation and optimization details

During optimization, the expectation in the ELBO is approximated with 10 samples. 5 layers of AF-in-hidden or SCF-in-hidden flow are used for the AF / AF and AF / SCF models and 3 layers are used for the IAF / SCF models, for character-level language modeling. 5 layers of SCF-in-hidden are used for all models on the polyphonic datasets. The base density is a standard Gaussian. Adam is used as the optimizer with a learning rate of 1e-3 and a gradient clipping cutoff of 0.25. Dropout is used to regularize the baseline model and the LSTM in the prior of the AF / AF and AF / SCF models. All LSTMs are two layers deep, and all embedding and hidden layers are made up of 500 units. Weight tying between the input embedding of the encoder and output embedding of the decoder is employed.

A latent size of $D = 50$ for each random vector $\boldsymbol{z}_t$ and $\boldsymbol{\epsilon}_t$ is used. During preliminary experiments we found that for character-level language modeling the results were nearly identical for $D = 5 - 80$.

Many recent works have found that it is necessary to bias the variational optimization to prevent posterior collapse, most commonly by using KL annealing or modifying the objective (Bowman et al., 2016; Kingma et al., 2016; Chen et al., 2017), without which it is easy for the model to obtain strong performance by simply ignoring the latent code. In our case we similarly find that KL annealing is essential to learn a strong mapping. We hypothesize that while the decoder is extremely weak, the prior itself is powerful and thus the generative model overall is still powerful enough to require such a bias.

Specifically, for the language modeling task we use KL annealing with an initial period of 0 weight on the KL term for 4 epochs followed by a linear increase to the full ELBO across 10 epochs. This schedule allows the models to first encode the vocabulary in the continuous space with 0 reconstruction loss and then learn the statistical dependencies between tokens. For the polyphonic datasets we extend this to 0 weight for 20 epochs followed by a linear increase over 15 epochs, due to the reduced dataset size.

References

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating Sentences from a Continuous Space. *Proceedings of CoNLL*, 2016. URL http://arxiv.org/abs/1511.06349.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. Variational Lossy Autoencoder. *Proceedings of ICLR*, 2017.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *Proceedings of ICLR*, 2017.

G. C. Holmes. The Use of Hyperbolic Cosines in Solving Cubic Polynomials. *The Mathematical Gazette*, 86(507): 473–477, 2002. URL http://www.jstor.org/stable/3621149.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving Variational Inference with Inverse Autoregressive Flow. *29th Conference on Neural Information Processing Systems*, 2016.

Nering, E. D. *Linear Algebra and Matrix Theory*. Wiley, New York, 2nd edition, 1970.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. *Advances in Neural Information Processing Systems*, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need. *31st Conference on Neural Information Processing Systems*, pp. 5998–6008, 2017.