Supplemental Materials for Deep Generative Models with Stick-Breaking Priors

Eric Nalisnick
Department of Computer Science
University of California, Irvine
enalisni@uci.edu

Padhraic Smyth
Department of Computer Science
University of California, Irvine
smyth@ics.uci.edu

KL-Divergence Derivation

$$\begin{aligned} KL(q_{\boldsymbol{\phi}}(\boldsymbol{\pi}_i|\mathbf{x}_i)||p_{\text{SB}}(\boldsymbol{\pi}_i;\boldsymbol{\alpha}_0)) &= \mathbb{E}_q\left[\log \frac{\prod_\infty q(v_{i,k})}{\prod_\infty p(v_{i,k})}\right] \\ &= \sum_{k=1}^{\infty} \mathbb{E}_q[\log q(v_{i,k})] - \mathbb{E}_q[\log p(v_{i,k})] \\ &= \sum_{k=1}^{K-1} \mathbb{E}_q[\log q(v_{i,k})] - \mathbb{E}_q[\log p(v_{i,k})]. \end{aligned} \tag{1}$$

The first term we need is the Kumaraswamy's entropy, which is given in [6] as

$$\mathbb{E}_q\left[\log q(v_k)\right] = \log a_\phi b_\phi + \frac{a_\phi - 1}{a_\phi}(-\gamma - \Psi(b_\phi) - \frac{1}{b_\phi}) - \frac{b_\phi - 1}{b_\phi}$$

where γ is Euler's constant and Ψ is the Digamma function. The second term, the log Beta's expectation under the variational Kumaraswamy, is

$$\begin{aligned} \mathbb{E}_q\left[\log p(v_k)\right] &= \int_0^1 a_\phi b_\phi v_k^{a_\phi - 1}(1 - v_k^{a_\phi})^{b_\phi - 1} \log \frac{1}{B(\alpha,\beta)} v_k^{\alpha-1}(1-v_k)^{\beta-1} dv_k \\ &= a_\phi b_\phi(\alpha - 1)\int_0^1 v_k^{a_\phi - 1}(1 - v_k^{a_\phi})^{b_\phi - 1}\log v_k dv_k \\ &+ a_\phi b_\phi(\beta - 1)\int_0^1 v_k^{a_\phi - 1}(1 - v_k^{a_\phi})^{b_\phi - 1}\log(1 - v_k) dv_k - \log B(\alpha,\beta). \end{aligned} \tag{2}$$

The first integral above is also needed in the derivation of the Kumaraswamy's entropy; [6] shows it can be evaluated using a Gradshteyn & Ryzhik formula (4.253), yielding

$$\begin{aligned} a_\phi b_\phi(\alpha - 1)\int_0^1 v_k^{a_\phi - 1}(1 - v_k^{a_\phi})^{b_\phi - 1}\log v_k dv_k &= a_\phi b_\phi(\alpha - 1)\frac{1}{a_\phi^2} B(1, b_\phi)[\Psi(1) - \Psi(1 + b_\phi)] \\ &= \frac{\alpha - 1}{a_\phi}[-\gamma - \Psi(b_\phi) - \frac{1}{b_\phi}]. \end{aligned} \tag{3}$$

The second integral, unfortunately, requires a Taylor series approximation. We expand $\log(1 - v_k)$ around 1 following [1]:

$$\log(1 - v_k) = -\sum_{m=1}^{\infty} \frac{1}{m} v_k^m.$$

This infinite sum converges since $|v_k| < 1$, and thus by the monotone convergence theorem, we can write the expectation of the sum as the sum of expectations:

$$\begin{aligned}(\beta - 1)\mathbb{E}_q[\log(1 - v_k)] &= (1 - \beta) \sum_{m=1}^{\infty} \frac{1}{m} \mathbb{E}_q[v_k^m] \\ &= (1 - \beta) b_\phi \sum_{m=1}^{\infty} \frac{1}{m} B\left(\frac{m}{a_\phi} + 1, b_\phi\right) \\ &= (1 - \beta) b_\phi \sum_{m=1}^{\infty} \frac{1}{m + a_\phi b_\phi} B\left(\frac{m}{a_\phi}, b_\phi\right)\end{aligned} \tag{4}$$

where $b_\phi B\left(\frac{m}{a_\phi} + 1, b_\phi\right)$ is the formula for the Kumaraswamy's mth moment. The low-order moments should dominate the infinite sum, and thus we need only a few terms for a good approximation. Substituting back into Equation 1 we have our final expression for the KL-divergence:

$$\begin{aligned}\mathbb{E}_q[\log q(v_k)] - \mathbb{E}_q[\log p(v_k)] =& \frac{a_\phi - \alpha}{a_\phi}\left(-\gamma - \Psi(b_\phi) - \frac{1}{b_\phi}\right) + \log a_\phi b_\phi + \log B(\alpha, \beta) \\ &+ (\beta - 1) b_\phi \sum_{m=1}^{\infty} \frac{1}{m + a_\phi b_\phi} B\left(\frac{m}{a_\phi}, b_\phi\right) - \frac{b_\phi - 1}{b_\phi}.\end{aligned} \tag{5}$$

Experiments and Optimization

Below we give more details about our experiments and model optimization. Also we have released Theano implementations available at `https://github.com/enalisnick/stick-breaking_dgms`. All experiments were run on AWS G2.2XL instances.

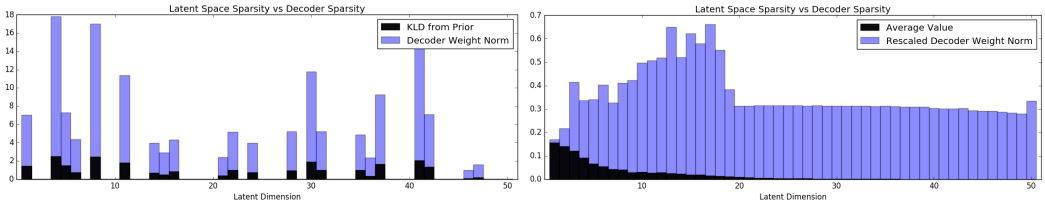
In regards to datasets, we used the following train-valid-test splits. Frey Faces was divided into $\{1500, 100, 300\}$, MNIST into $\{45000, 5000, 10000\}$, MNIST+rot into $\{70000, 10000, 20000\}$, and SVHN into $\{65000, 8257, 26032\}$. SVHN was the only dataset that underwent preprocessing; following [4], we reduced the dimensionality via PCA to 500 dimensions that capture 99.9% of the data's variance. No effort was made to preserve class balance across train-valid splits nor during label removal for the semi-supervised tasks.

Regarding optimization, all models were trained with minibatches of size 100 and using *AdaM* [3] to set the gradient descent step size. For AdaM, we used $\alpha = 0.0003$, $b1 = 0.95$, and $b2 = 0.999$ in all experiments. Early stopping was used during semi-supervised training with a look-ahead threshold of 30 epochs. For the semi-supervised deep generative models, classification loss needs up-weighted in some way. In [4], an extra weight was placed on the label log likelihood term. We attempted this strategy but attained better performance (for all models) by re-weighting the contribution of the supervised data within each mini-batch, i.e. $\lambda \nabla \tilde{\mathcal{J}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i, y_i) + (1 - \lambda)\tilde{\mathcal{J}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}_i)$. We calibrated λ by comparing the log likelihood of the supervised and unsupervised data in preliminary training runs and setting the parameter such that the supervised data had a slightly higher likelihood. For the MNIST datasets, $\lambda = .375$, and for SVHN, $\lambda = .45$.

As for the model architectures, all experiments used ReLUs exclusively for hidden unit activations. The dimensionality / truncation-level of the latent variables was set at 50 for every experiment except Frey Faces. All weights were initialized by drawing from $\mathrm{N}(\mathbf{0}, 0.001 \cdot \mathbb{1})$, and biases were set to zero to start. No regularization (dropout, weight decay, etc) was used, and only one sample was used for each calculation of the Monte Carlo expectations. We used the leading ten terms to compute the infinite sum in the KL divergence between the Beta and Kumaraswamy.

Decoder Capacity

The 'component collapsing' behavior of the variational autoencoder has been well noted [2, 5]: the model will set to zero the outgoing weights of latent variables that remain near the prior. Figure 1 (a) depicts this phenomenon for the Gauss-VAE by plotting the KL divergence from the prior and outgoing decoder weight norm for each latent dimension. We see the weights are only nonzero in



(a) Gauss VAE (b) Stick-Breaking VAE

Figure 1: Sparsity in the latent representation vs sparsity in the decoder network. The Gaussian VAE 'turns off' unused latent dimensions by setting the outgoing weights to zero (in order to dispel the sampled noise). The SB VAE, on the other hand, also has sparse representations but without decay of the associated decoder weights.

the dimensions in which there is posterior deviation. Ostensibly the model receives only sampling noise from the dimensions that remain at the prior, and setting the decoder weights to zero quells this variance. While the behavior of the Gauss VAE is not necessarily improper, all examples are restricted to pass through the same latent variables. A sparse-coded representation—one having few active components per example (like the Gauss-VAE) but diversity of activations across examples–would likely be better.

We compare the activation patterns against the sparsity of the decoder for the SB-VAE in Figure 1 (b). Since KL-divergence doesn't directly correspond to sparsity in stick-breaking latent variables like it does for Gaussian ones, the black lines denote the average activation value per dimension. Similarly to (a), blue lines denoted the decoder weight norms, but they had to be down-scaled by a factor of 100 so they could be visualized on the same plot. The SB-VAE does not seem to have any component collapsing, which is not too surprising since the model can set latent variables to zero to deactivate decoder weights without being in the heart of the prior. We conjecture that this increased capacity is the reason stick-breaking variables demonstrate better discriminative performance in many of our experiments.

References

[1] Lawrence Carin, David M Blei, and John W Paisley. Variational inference for stick-breaking Beta process priors. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 889–896, 2011.

[2] Laurent Dinh and Vincent Dumoulin. Training neural Bayesian nets, 2014. Slides from CIFAR NCAP Summer School, August 12–16, University of Toronto, Toronto, ON.

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

[4] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

[5] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

[6] Joseph Victor Michalowicz, Jonathan M Nichols, and Frank Bucholtz. *Handbook of differential entropy*. CRC Press, 2013.