Closed Form Variational Objectives For Bayesian Neural Networks with a Single Hidden Layer

Martin Jankowiak jankowiak@uber.com
Uber AI Labs San Francisco, CA

Abstract

We consider setups in which variational objectives for Bayesian neural networks can be computed in closed form. In particular we focus on single-layer networks in which the activation function is piecewise polynomial (e.g. ReLU). In this case we show that for a Normal likelihood and structured Normal variational distributions one can compute a variational lower bound in closed form. In addition we compute the predictive mean and variance in closed form. Finally, we also show how to compute approximate lower bounds for other likelihoods (e.g. softmax classification). In experiments we show how the resulting variational objectives can help improve training and provide fast test time predictions.

1 Introduction

In recent years significant effort has gone into developing flexible probabilistic models for the supervised setting. These include, among others, deep gaussian processes [5] as well as various approaches to Bayesian neural networks [21, 18, 10, 7, 3, 9]. While neural networks promise considerable flexibility, scalable learning algorithms for Bayesian neural networks that can deliver robust uncertainty estimates remain elusive. While some of the difficulty stems from the inadequate (weight-space) priors that are typically used, much of the challenge can be traced to the difficulty of the inference problem itself. In the variational inference setting, this manifests itself in at least two ways. First, the need to restrict the variational family to a tractable class limits the fidelity of the approximate learned posterior. Second, nested non-linearities necessitate sampling methods during training, which can make for a challenging stochastic optimization problem, especially for wide, deep networks. In this work our goal is to make the stochastic optimization problem (somewhat) easier by integrating out some of the weights analytically. In the next section we focus on the regression case, leaving a discussion of other cases to the appendix.[1]

2 Regression Setup

Consider a dataset $\{\boldsymbol{x}_i, \mathbf{y}_i\}$ of size N with inputs $\boldsymbol{x}_i$ and outputs $\mathbf{y}_i$. To simplify the notation, we consider the case where $\boldsymbol{x}$ is D-dimensional and $\mathbf{y}$ is 1-dimensional. We consider a neural network with a single hidden layer defined by the following computational flow:[2]

$$\boldsymbol{x} \to \mathbf{A}\boldsymbol{x} \to g(\mathbf{A}\boldsymbol{x}) \to \mathbf{B}g(\mathbf{A}\boldsymbol{x}) \tag{1}$$

Here $g(\cdot)$ is the non-linearity, $\mathbf{A}$ is of size $H \times D$ and $\mathbf{B}$ is of size $1 \times H$, where H is the number of hidden units. We choose a Normal likelihood with precision β and standard Normal priors for the weights. Thus the marginal likelihood of the observed data is:

$$p(\mathbf{Y}|\mathbf{X}) = \int d\mathbf{A} d\mathbf{B}\; p(\mathbf{A}) p(\mathbf{B}) \prod_i p(\mathbf{y}_i|\mathbf{B}g(\mathbf{A}\boldsymbol{x}_i), \beta) \tag{2}$$

[1] Please refer to the appendix for a more detailed discussion of related work.

[2] We handle bias terms by augmenting inputs to each neural network layer with an element equal to 1.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

3 Variational Bound

We consider a variational distribution of the form

$$q(\mathbf{A}, \mathbf{B}) = q(\mathbf{B}) \prod_{h=1}^{H} q(\mathbf{a}_h) \quad \text{with} \quad q(\mathbf{a}_h) = \mathcal{N}(\mathbf{a}_h | \mathbf{a}_{0h}, \mathbf{\Sigma}_{\mathbf{a}_h}) \quad \text{and} \quad q(\mathbf{B}) = \mathcal{N}(\mathbf{B} | \mathbf{B}_0, \mathbf{\Sigma}_{\mathbf{B}}) \tag{3}$$

where each component distribution is Normal. Since we treat each row of $\mathbf{A}$ independently, the activations $\{g(\mathbf{A}\boldsymbol{x})_h\}$ are conditionally independent given an input $\boldsymbol{x}$. With these assumptions we can write down the following variational bound:

$$\log p(\mathbf{Y}|\mathbf{X}) \geq \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})} \left[\sum_i \log p(\mathbf{y}_i | \boldsymbol{x}_i, \mathbf{A}, \mathbf{B}) \right] - \mathrm{KL}(q(\mathbf{A})|p(\mathbf{A})) - \mathrm{KL}(q(\mathbf{B})|p(\mathbf{B})) \tag{4}$$

The KL divergences are readily computed. We now show that we can compute closed form expressions for the first term in Eqn. 4 (i.e. the expected log likelihood) for certain non-linearities $g(\cdot)$. For concreteness we consider the ReLU activation function, i.e. $g(x) = \max(0, x) = \frac{1}{2}(x + |x|)$. The expected log likelihood (ELL) for a single datapoint is given by

$$\mathrm{ELL} = -\frac{\beta}{2} \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})} \left[(\mathbf{y} - \mathbf{B}g(\mathbf{A}\boldsymbol{x}))^2 \right] + \frac{1}{2} \left(\log \beta - \log 2\pi \right) \tag{5}$$

The expectation in Eqn. 5 becomes

$$\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})} \Big[\mathbf{y}^2 - \mathbf{y}\mathbf{B}_{1h} \left(\mathbf{a}_h^{\mathrm{T}} \boldsymbol{x} + |\mathbf{a}_h^{\mathrm{T}} \boldsymbol{x}| \right) + \tfrac{1}{4} \Sigma_{h,\tilde{h}} \mathbf{B}_{1h} \mathbf{B}_{1\tilde{h}} \left(\mathbf{a}_h^{\mathrm{T}} \boldsymbol{x} + |\mathbf{a}_h^{\mathrm{T}} \boldsymbol{x}| \right) \left(\mathbf{a}_{\tilde{h}}^{\mathrm{T}} \boldsymbol{x} + |\mathbf{a}_{\tilde{h}}^{\mathrm{T}} \boldsymbol{x}| \right) \Big] \tag{6}$$

Massaging terms, the expected log likelihood for the full dataset is given by

$$\mathrm{ELL} = -\frac{\beta}{2} \sum_i \left[(\mathbf{y}_i - \hat{\mathbf{y}}_i(\boldsymbol{x}_i))^2 + \mathrm{var}_{\mathbf{AB}}(\boldsymbol{x}_i) \right] + \frac{N}{2} \left(\log \beta - \log 2\pi \right) \tag{7}$$

Here we have introduced the mean function $\hat{\mathbf{y}}(\boldsymbol{x}) = \mathbf{B}_0 \cdot \mathbf{\Phi}$ as well as the corresponding variance:

$$\mathrm{var}_{\mathbf{AB}}(\boldsymbol{x}) = \mathrm{diag}(\mathbf{\Xi}_{\mathbf{B}}) \cdot (\mathbf{\Upsilon} - \mathbf{\Phi} \odot \mathbf{\Phi}) + \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Sigma}_{\mathbf{B}} \mathbf{\Phi} \tag{8}$$

Note that $\hat{\mathbf{y}}(\boldsymbol{x})$ and $\mathrm{var}_{\mathbf{AB}}(\boldsymbol{x})$ can be used at test time to yield fast predictive means and variances. We have also defined the matrix $\mathbf{\Xi}_{\mathbf{B}} = \mathbf{\Sigma}_{\mathbf{B}} + \mathbf{B}_0 \mathbf{B}_0^{\mathrm{T}}$ and the H-dimensional vectors $\mathbf{\Phi}$ and $\mathbf{\Upsilon}$:

$$\Phi_h = \mathbb{E}_{q(\mathbf{A})} \left[g(\mathbf{A}\boldsymbol{x})_h \right] \qquad \Upsilon_h = \mathbb{E}_{q(\mathbf{A})} \left[g(\mathbf{A}\boldsymbol{x})_h^2 \right] \tag{9}$$

The key quantities are the expectations in Eqn. 9. As we show in the appendix, these can be computed in closed form for piecewise polynomial activation functions. The resulting expressions involve nothing more exotic than the error function.

4 Experiments

We present a few experiments that demonstrate how our approach can be folded into larger probabilistic models. Note that our focus here is on how (partial) analytic control can help training (Sec. 4.1-4.2) and prediction (Sec. 4.3) and *not* the suitability of Bayesian neural networks for particular tasks or datasets. Please refer to the appendix for details on experimental setups.

4.1 Variance Reduction

We train a Bayesian neural network with two hidden layers on a regression task and compute the gradient variance during training. As can be seen from Table 1, Rao-Blackwellizing the two weight matrices closest to the outputs reduces the variance, especially for the covariance parameters. As the weight matrices we integrate out get larger, this variance reduction becomes more pronounced.

	Upon initialization			Late in training		
	First Layer	Second Layer	Final Layer	First Layer	Second Layer	Final Layer
Analytic	$8.6 / 6 \times 10^{-4}$	$3.2 / 2 \times 10^{-6}$	$227 / 1 \times 10^{-5}$	$1.75 / 1 \times 10^{-4}$	$0.2 / 2 \times 10^{-6}$	$259 / 3 \times 10^{-8}$
Sampling	$19.7 / 7 \times 10^{-4}$	$10.1 / 2 \times 10^{-4}$	$704 / 0.03$	$2.6 / 1 \times 10^{-4}$	$0.4 / 1 \times 10^{-4}$	$518 / 3 \times 10^{-3}$

Table 1: Mean gradient variances for the network in Sec. 4.1. The first number in each cell corresponds to gradients w.r.t. weight means and the second to gradients w.r.t. (log root) variances.

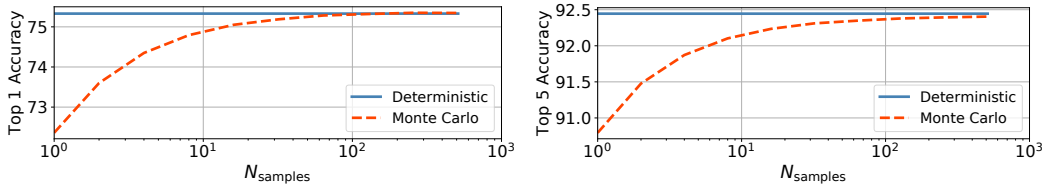

Figure 1: We compare the performance of a Monte Carlo estimate of classification accuracy to a deterministic approximation. **Left:** Top-1 accuracy. **Right:** Top-5 accuracy. See Sec. 4.3 for details.

4.2 VAE with a Bayesian Decoder

We train a VAE [15, 22] with a Normal likelihood on a continuous-valued dataset. For the decoder we use a Bayesian neural network with a single hidden layer.[3] We train three model and inference variants and report test log likelihoods in Table 2. Apart from the first variant (V1), all variants make use of a Bayesian neural network as the decoder. Variants V2 and V3 differ in which weights are sampled during training.[4] In V2 the weights before the non-linearity are sampled, while in V3 no weights are sampled. While the test log likelihoods in Table 2 do not differ dramatically, we see evidence that: i) a Bayesian decoder can be useful in this setting; and ii) integrating out weights can help us train a better model.

	V1	V2	V3 (this work)
Bayesian Decoder	No	Yes	Yes
Sampling	z only	z and some weights	z only
Test LL	-107.16	-107.20	**-107.10**

Table 2: Test log likelihoods for the VAE experiment in Sec. 4.2. Higher is better.

4.3 Fast Prediction

We train a Bayesian neural network on ImageNet [24].[5] Specifically we place a prior on the two weight matrices closest to the softmax output. We then compute classification accuracies on the test set using two methods: i) Monte Carlo; and ii) a deterministic approximation using the analytic results described above (see Appendix for details). As can be seen from Fig. 1, a large number of samples must be drawn before the MC estimator reaches the performance of the deterministic approximation. Indeed even with 512 samples the deterministic approximation outperforms MC on top-5 accuracy.

5 Discussion

The approach developed here is expected to be most useful when integrated into larger Bayesian neural networks setups. It would be of particular interest to combine this approach with the class of priors described in [12], the deterministic approximations in [26], or Normal variational distributions with flexible conditional dependence like those in [17]. Finally, our analytic results could be useful in the context of other classes of non-linear probabilistic models.

[3] Alternatively, we can think of this as the neural network analog of the deep latent variable model in ref. [4].

[4] More precisely, we always use the 'local reparameterization trick' [14] and never sample weights directly.

[5] While our analytic results can be used to form approximate variational objectives (or, alternatively, control variates) in the classification setting (see Appendix) here we sample the weights during training.

References

[1] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[2] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[4] Z. Dai, A. Damianou, J. González, and N. Lawrence. Variational auto-encoded deep gaussian processes. *arXiv preprint arXiv:1511.06455*, 2015.

[5] A. Damianou and N. Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[6] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[7] A. Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] J. M. Hernández-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[10] G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

[11] M. Kandemir, M. Haussmann, and F. A. Hamprecht. Sampling-free variational inference of bayesian neural nets. *arXiv preprint arXiv:1805.07654*, 2018.

[12] T. Karaletsos, P. Dayan, and Z. Ghahramani. Probabilistic meta-representations of neural networks. *arXiv preprint arXiv:1810.00555*, 2018.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[15] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] W. V. Li, A. Wei, et al. Gaussian integrals involving absolute value functions. In *High dimensional probability V: the Luminy volume*, pages 43–59. Institute of Mathematical Statistics, 2009.

[17] C. Louizos and M. Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

[18] D. J. MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[19] B. M. Marlin, M. E. Khan, and K. P. Murphy. Piecewise bounds for estimating bernoulli-logistic latent gaussian models. In *ICML*, pages 633–640, 2011.

[20] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[21] C. Peterson. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.

[22] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[23] S. M. Ross. *Simulation*. Academic Press, San Diego, 2006.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[25] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[26] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt. Fixing variational bayes: Deterministic variational inference for bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

6 Appendix

The main goal of this appendix is to show how to compute the necessary expectations in Eqn. 9 for piecewise polynomial non-linearities. Instead of presenting a (unwieldy) master formula for the general case, we proceed step by step and show how the computation is done in a few cases of increasing complexity. We begin with a basic ReLU integral.

6.1 ReLU Mean Function

We first consider the mean function for the ReLU activation function $g(x) = \max(0, x) = \frac{1}{2}(x + |x|)$, i.e. we would like to compute the following expectation:

$$\mathbb{E}_{q(\mathbf{a})}\left[\mathbf{a}^{\mathrm{T}}\boldsymbol{x} + |\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] = \mathbb{E}_{q(\mathbf{a})}\left[\mathbf{a}^{\mathrm{T}}\boldsymbol{x}\right] + \mathbb{E}_{q(\mathbf{a})}\left[\,|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] \tag{10}$$

where $q(\mathbf{a}) = \mathcal{N}(\mathbf{a}|\mathbf{a}_0, \boldsymbol{\Sigma}_{\mathbf{a}})$. The first expectation in Eqn. 10 is elementary. For the second expectation note that, since $\mathbf{a}^{\mathrm{T}}\boldsymbol{x} \sim \mathcal{N}(\mathbf{a}_0^{\mathrm{T}}\boldsymbol{x}, \mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{a}}\boldsymbol{x})$, the expectation can be transformed to a one-dimensional integral

$$\mathbb{E}_{q(\mathbf{a})}\left[|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] = \mathbb{E}_{y\sim\mathcal{N}(\mathbf{a}_0^{\mathrm{T}}\boldsymbol{x}, \mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{a}}\boldsymbol{x})}\left[|y|\right] \tag{11}$$

that can readily be computed in terms of the error function. We do not do so here, however, because in subsequent derivations we will find an alternative strategy—namely to make use of a particular integral representation for the absolute value function—to be more convenient. Thus before we give an explicit formula for Eqn. 10 we collect a few useful identities.

6.2 Useful Integrals

First we define the following (scalar) quantities, which we will make extensive use of throughout the appendix:

$$\xi^2 \equiv \mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{a}}\boldsymbol{x} \qquad\qquad \gamma \equiv \mathbf{a}_0^{\mathrm{T}}\boldsymbol{x} \tag{12}$$

We then have

$$\mathbb{E}_{q(\mathbf{a})}\left[\exp(i\mathbf{a}^{\mathrm{T}}\boldsymbol{x}t)\right] = \exp(-\tfrac{1}{2}\xi^2t^2 + i\gamma t) \tag{13}$$

and

$$\mathbb{E}_{q(\mathbf{a})}\left[\mathbf{a}^{\mathrm{T}}\boldsymbol{x}\exp(i\mathbf{a}^{\mathrm{T}}\boldsymbol{x}t)\right] = \left(i\xi^2t + \gamma\right)\exp(-\tfrac{1}{2}\xi^2t^2 + i\gamma t) \tag{14}$$

as well as

$$\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})^2\right] = \xi^2 + \gamma^2 \tag{15}$$

The integral identity we make use of is:[6]

$$|z| = \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\left(1-\cos(zt)\right) = \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\left(1-\frac{\exp(izt)+\exp(-izt)}{2}\right) \tag{16}$$

For reference we note that this identity can easily be derived by integrating by parts and making use of the well-known sine integral:[7]

$$\int_0^\infty \frac{\sin(t)}{t}dt = \tfrac{1}{2}\pi \tag{17}$$

6.3 ReLU Part II

Combining the above identities we get:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{a})}\left[|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\\ &= \sqrt{\frac{2}{\pi}}\xi e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}} + \gamma\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right)\\ &\equiv \Omega(\xi,\gamma)\end{aligned} \tag{18}$$

and

$$\begin{aligned}\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(\gamma-\gamma e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)+\xi^2te^{-\frac{1}{2}\xi^2t^2}\sin(\gamma t)\Big)\\ &= \gamma\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\} + \xi^2\frac{\partial}{\partial\gamma}\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\}\\ &= \gamma\left\{\sqrt{\frac{2}{\pi}}\xi e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}} + \gamma\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right)\right\} + \xi^2\frac{\partial}{\partial\gamma}\left\{\sqrt{\frac{2}{\pi}}\xi e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}} + \gamma\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right)\right\}\\ &= \sqrt{\frac{2}{\pi}}\gamma\xi e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}} + (\xi^2+\gamma^2)\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right)\\ &\equiv \Psi(\xi,\gamma)\end{aligned} \tag{19}$$

Thus we have all the ingredients to compute the expectations in Eqn. 9:

$$\begin{aligned}\Phi_h &= \mathbb{E}_{q(\mathbf{A})}\left[g(\mathbf{A}\boldsymbol{x})_h\right] = \tfrac{1}{2}(\gamma_h+\Omega_h)\\ \Upsilon_h &= \mathbb{E}_{q(\mathbf{A})}\left[g(\mathbf{A}\boldsymbol{x})_h^2\right] = \tfrac{1}{2}(\xi_h^2+\gamma_h^2+\psi_h)\end{aligned} \tag{20}$$

As stated in the main text, these expectations involve nothing more exotic than the error function. Note that as $\gamma_h/\xi_h \to \infty$ only a small portion of probability mass is propagated through the constant portion of the ReLU activation function. As such, in this limit we expect $\Phi_h \to \gamma_h$ and $\Upsilon_h \to \gamma_h^2+\xi_h^2$. It is easy to verify that this is indeed the case. Similarly, as $\gamma_h/\xi_h \to -\infty$, we have $\Phi_h, \Upsilon_h \to 0$.

6.4 Other Non-linearities

6.4.1 Leaky ReLU

We consider the 'leaky' ReLU, which we define to be given by

$$g_\epsilon(x) = \max(\epsilon x, x) = \tfrac{1}{2}((1+\epsilon)x + (1-\epsilon)|x|) \tag{21}$$

for some $\epsilon > 0$. In this case one finds:

$$\begin{aligned}\Phi_h &= \mathbb{E}_{q(\mathbf{A})}\left[g_\epsilon(\mathbf{A}\boldsymbol{x})_h\right] = \tfrac{1}{2}((1+\epsilon)\gamma_h + (1-\epsilon)\Omega_h)\\ \Upsilon_h &= \mathbb{E}_{q(\mathbf{A})}\left[g_\epsilon(\mathbf{A}\boldsymbol{x})_h^2\right] = \tfrac{1}{2}((1+\epsilon^2)(\xi_h^2+\gamma_h^2) + (1-\epsilon^2)\psi_h)\end{aligned} \tag{22}$$

[6] Note that this identity is also used in [16] to compute a related class of Gaussian integrals.

[7] The absolute value in Eqn. 16 is an immediate consequence of $\cos(zt) = \cos(-zt)$.

6.4.2 Hard Sigmoid

We consider the 'hard sigmoid' non-linearity, which we define to be given by

$$g_\alpha(x) = \tfrac{1}{2}(|x+\alpha| - |x-\alpha|) \tag{23}$$

for a given constant α. The identity in Eqn. 16 can immediately be generalized to

$$\begin{aligned} |z+z_0| &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-\cos(zt)\cos(z_0t)+\sin(zt)\sin(z_0t)\Big) \\ &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-\frac{1}{2}[\exp(izt+iz_0t)+\exp(-izt-iz_0t))]\Big) \end{aligned} \tag{24}$$

Proceeding as before we compute

$$\begin{aligned} \mathbb{E}_{q(\mathbf{a})}\left[|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}+\alpha|\right] &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos((\gamma+\alpha)t)\Big) \\ &= \Omega(\xi,\gamma+\alpha) \end{aligned} \tag{25}$$

and (for $\alpha_+, \alpha_- > 0$)

$$\begin{aligned} &\mathbb{E}_{q(\mathbf{a})}\left[|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}+\alpha_+||\mathbf{a}^{\mathrm{T}}\boldsymbol{x}-\alpha_-|\right] = \\ &\sqrt{\frac{2}{\pi}}\xi\Big\{(\gamma+\alpha_+)e^{-\frac{1}{2}\frac{(\gamma-\alpha_-)^2}{\xi^2}} - (\gamma-\alpha_-)e^{-\frac{1}{2}\frac{(\gamma+\alpha_+)^2}{\xi^2}}\Big\} + \\ &\Big\{\xi^2+(\gamma+\alpha_+)(\gamma-\alpha_-)\Big\}\Big\{1-\mathrm{erf}\left(\frac{\gamma+\alpha_+}{\sqrt{2}\xi}\right)+\mathrm{erf}\left(\frac{\gamma-\alpha_-}{\sqrt{2}\xi}\right)\Big\} \\ &\equiv \chi(\xi,\gamma,\alpha_+,\alpha_-) \end{aligned} \tag{26}$$

Using these identities we find:

$$\begin{aligned} &\Phi_h = \mathbb{E}_{q(\mathbf{A})}\left[g_\alpha(\mathbf{A}\boldsymbol{x})_h\right] = \tfrac{1}{2}(\Omega(\xi_h,\gamma_h+\alpha)-\Omega(\xi_h,\gamma_h-\alpha)) \\ &\Upsilon_h = \mathbb{E}_{q(\mathbf{A})}\left[g_\alpha(\mathbf{A}\boldsymbol{x})_h^2\right] = \tfrac{1}{2}(\xi_h^2+\gamma_h^2-\chi_h+\alpha^2) \qquad \text{with} \qquad \chi_h \equiv \chi(\xi_h,\gamma_h,\alpha,\alpha) \end{aligned} \tag{27}$$

As $\alpha \to \infty$, we have $g_\alpha(x) \to x$, i.e. the hard sigmoid non-linearity approaches the identity function. It is easy to verify that in this limit the expectations in Eqn. 27 approach the correct limit.

6.4.3 ReLU Squared

We consider the 'ReLU squared' non-linearity, which we define to be given by

$$g(x) = \mathrm{ReLU}(x)^2 = \Big(\tfrac{1}{2}(x+|x|)\Big)^2 = \tfrac{1}{2}(x^2+x|x|) \tag{28}$$

This is the first non-linearity we have considered that contains a piecewise quadratic portion. Using previous results as well as the higher moments computed in the next section we find:

$$\begin{aligned} &\Phi_h = \mathbb{E}_{q(\mathbf{A})}\left[g(\mathbf{A}\boldsymbol{x})_h\right] = \tfrac{1}{2}(\gamma_h^2+\xi_h^2+\Psi_h) \\ &\Upsilon_h = \mathbb{E}_{q(\mathbf{A})}\left[g(\mathbf{A}\boldsymbol{x})_h^2\right] = \tfrac{1}{2}(\gamma_h^4+6\gamma_h^2\xi_h^2+3\xi_h^4+\rho_h) \end{aligned} \tag{29}$$

6.5 Higher Moments

We can also compute higher moments of activation functions. We start with the integral identities

$$\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})^2\exp(i\mathbf{a}^{\mathrm{T}}\boldsymbol{x}t)\right] = \left(\xi^2+(i\xi^2t+\gamma)^2\right)\exp(-\tfrac{1}{2}\xi^2t^2+i\gamma t) \tag{30}$$

and

$$\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})^3\exp(i\mathbf{a}^{\mathrm{T}}\boldsymbol{x}t)\right] = \left\{3\xi^2(i\xi^2t+\gamma)+(i\xi^2t+\gamma)^3\right\}\exp(-\tfrac{1}{2}\xi^2t^2+i\gamma t) \tag{31}$$

and

$$\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})^3\right] = \gamma^3+3\gamma\xi^2 \tag{32}$$

We then have

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})^2|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(\xi^2+\gamma^2-(\xi^2+\gamma^2-\xi^4t^2)e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)+2\xi^2\gamma t e^{-\frac{1}{2}\xi^2t^2}\sin(\gamma t)\Big) \\
&= (\xi^2+\gamma^2)\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\}+\frac{2\xi^4}{\pi}\int_0^\infty dt e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t) \\
&+ 2\xi^2\gamma\frac{\partial}{\partial\gamma}\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\} \\
&= (\xi^2+\gamma^2)\left\{\sqrt{\frac{2}{\pi}}\xi e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}}+\gamma\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right)\right\}+\sqrt{\frac{2}{\pi}}\xi^3 e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}} \\
&+ 2\xi^2\gamma\frac{\partial}{\partial\gamma}\left\{\sqrt{\frac{2}{\pi}}\xi e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}}+\gamma\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right)\right\} \\
&= \sqrt{\frac{2}{\pi}}(2\xi^3+\gamma^2\xi)e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}}+(3\gamma\xi^2+\gamma^3)\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right) \\
&\equiv \zeta(\xi,\gamma)
\end{aligned} \tag{33}$$

as well as

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{a})}\left[(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})^3|\mathbf{a}^{\mathrm{T}}\boldsymbol{x}|\right] &= \frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(\gamma^3+3\gamma\xi^2-(\gamma^3+3\gamma\xi^2-3\gamma\xi^4t^2)e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t) \\
&+ (3\xi^4t+3\gamma^2\xi^2t+\xi^6t^3)e^{-\frac{1}{2}\xi^2t^2}\sin(\gamma t)\Big) \\
&= (\gamma^3+3\gamma\xi^2)\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\}+\frac{6\gamma\xi^4}{\pi}\int_0^\infty dt e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t) \\
&+ (3\xi^4+3\gamma^2\xi^2)\frac{\partial}{\partial\gamma}\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\} \\
&- 2\xi^6\frac{\partial}{\partial\xi^2}\frac{\partial}{\partial\gamma}\left\{\frac{2}{\pi}\int_0^\infty \frac{dt}{t^2}\Big(1-e^{-\frac{1}{2}\xi^2t^2}\cos(\gamma t)\Big)\right\} \\
&= \sqrt{\frac{2}{\pi}}(\gamma^3\xi+7\gamma\xi^3)e^{-\frac{1}{2}\frac{\gamma^2}{\xi^2}}+(\gamma^4+6\gamma^2\xi^2+3\xi^4)\mathrm{erf}\left(\frac{\gamma}{\sqrt{2}\xi}\right) \\
&\equiv \rho(\xi,\gamma)
\end{aligned} \tag{34}$$

6.6 Piecewise Polynomial Activation Functions

Piecewise polynomial functions in x can be represented by composing polynomials in x with the absolute value function. Thus in order to compute Eqn. 9 for general piecewise polynomial activation functions we need to be able to compute expectations of the form

$$\mathbb{E}_{q(\mathbf{a})}\left[p_0(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})\prod_i |p_i(\mathbf{a}^{\mathrm{T}}\boldsymbol{x})|\right] \tag{35}$$

where the $p_i(\cdot)$ are polynomials. We have shown how this computation can be done in a number of cases. For any specific case the recipe we have used to do the computation remains applicable. In particular one can compute any needed 'base' integrals by doing the computation in one dimension as in Eqn. 11. One can then make use of the integral identity in Eqn. 16 and differentiation to compute higher order moments via purely algebraic operations (c.f. the manipulations in Sec. 6.5).

6.7 Other likelihoods

In the main text we showed how to compute exact closed form expressions for the ELBO variational objective in the regression case. For other likelihoods, the required expectations are generally

intractable. Nevertheless we can still compute closed form variational objectives at the price of some approximation. Alternatively, if we are worried about the bias introduced by our approximations, we can use our approximations as control variates in the Monte Carlo sampling setting. In this section we briefly describe how this goes in the case of softmax classification.

6.7.1 Softmax Categorical Likelihood

Using familiar bounds[8] we have:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log p(y=k|\mathbf{A},\mathbf{B},\boldsymbol{x})\right] &= \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log \frac{e^{\mathbf{y}_k}}{\sum_j e^{\mathbf{y}_j}}\right] \\ &\geq \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\mathbf{y}_k\right] - \log \sum_j \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[e^{\mathbf{y}_j}\right]\end{aligned} \tag{36}$$

We do a second-order Taylor expansion of $\mathbf{y}_j$ around its expectation $\hat{\mathbf{y}}_j$ to obtain

$$\begin{aligned}\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[e^{\mathbf{y}_j}\right] &\approx e^{\hat{\mathbf{y}}_j}\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[1 + (\mathbf{y}_j - \hat{\mathbf{y}}_j) + \frac{1}{2}(\mathbf{y}_j - \hat{\mathbf{y}}_j)^2\right] \\ &= e^{\hat{\mathbf{y}}_j}\left(1 + \tfrac{1}{2}\mathrm{var}(\mathbf{y}_j)\right)\end{aligned} \tag{37}$$

so that our approximate lower bound to the expected log likelihood becomes

$$\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log p(y=k|\mathbf{A},\mathbf{B},\boldsymbol{x})\right] \gtrapprox \hat{\mathbf{y}}_k - \log \sum_j e^{\hat{\mathbf{y}}_j}\left(1 + \tfrac{1}{2}\mathrm{var}(\mathbf{y}_j)\right) \tag{38}$$

We can then use the closed form expressions for the mean function and variance given in the main text to form a deterministic approximation to the expected log likelihood.

6.7.2 Logistic Bernoulli Likelihood

For the case with two classes with $y \in \{0, 1\}$ and where we have a single logit $\hat{y}$ the approximation in Eqn. 38 reduces to

$$\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log p(y|\mathbf{A},\mathbf{B},\boldsymbol{x})\right] \gtrapprox y\hat{y} - \log\left(1 + e^{\hat{y}}\left[1 + \tfrac{1}{2}\mathrm{var}(\hat{y})\right]\right) \tag{39}$$

6.7.3 Control Variates

To reduce bias to zero one can construct a control variate [23] version of the variational objective in Eqn. 38:

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log p(y=k|\mathbf{A},\mathbf{B},\boldsymbol{x})\right] = \\ &\hat{\mathbf{y}}_k - \log \sum_j e^{\hat{\mathbf{y}}_j}\left(1 + \tfrac{1}{2}\mathrm{var}(\mathbf{y}_j)\right) - \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log \sum_j e^{\mathbf{y}_j}\right] + \\ &\log\Big(\sum_j e^{\hat{\mathbf{y}}_j}\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[1 + (\mathbf{y}_j - \hat{\mathbf{y}}_j) + \frac{1}{2}(\mathbf{y}_j - \hat{\mathbf{y}}_j)^2\right]\Big)\end{aligned} \tag{40}$$

The expectations in Eqn. 40 are then estimated with Monte Carlo, while the rest is available in closed form. Alternatively, we can use the following estimator:

$$\mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log p(y=k|\mathbf{A},\mathbf{B},\boldsymbol{x})\right] = \hat{\mathbf{y}}_k - \mathbb{E}_{q(\mathbf{A})q(\mathbf{B})}\left[\log \sum_j e^{\mathbf{y}_j}\right] \tag{41}$$

That is, we use the analytic result for the numerator in the softmax likelihood and sample the troublesome denominator.

[8] See e.g. `http://www.columbia.edu/~jwp2128/Teaching/E6720/Fall2016/papers/twobounds.pdf`

6.7.4 Fast Approximate Prediction

To make fast test time predictions we can simply use the analytic mean function $\hat{\mathbf{y}}$, i.e. use

$$\text{pred}(\boldsymbol{x}) = \text{argmax}_k \hat{\mathbf{y}}_k(\boldsymbol{x}) \tag{42}$$

effectively ignoring the normalizing term in the softmax likelihood. As can be seen in Fig. 1, this approximation can be quite effective in practice.

6.8 Experimental Details

All the experiments described in this work were implemented in the Pyro probabilistic programming language [2], which is built on top of PyTorch [20]. As noted in the main text, whenever sampling a weight matrix we make use of the 'local reparameterization trick' [14], i.e. we sample in pre-activation space and not in weight space. This can lead to substantial variance reduction as compared to sampling in weight space directly.

6.8.1 Variance Reduction

We use the 90-dimensional 'YearPredictionMSD' dataset from the UCI repository [6]. This dataset is a subset of the Million Song Dataset [1]. The architecture of our neural network is given by $90 - 200 - 200 - 1$, where all layers are fully connected and both non-linearities are ReLU. We use mean field (Normal) variational distributions for all weight matrices. To compute gradient variances of variational parameters with respect to the variational objective, we fix a random mini-batch of training data with 500 elements. We then compute 10^4 samples and report empirical gradient variances averaged over the elements of each tensor. We report gradient variances computed before any training as well as after 50 epochs. For the (partially) analytic result, only the weight matrix closest to the inputs is sampled, while for the sampling result all weight matrices are sampled.

6.8.2 VAE with a Bayesian Decoder

We use the same dataset as for the variance reduction experiment above (with the difference that in this unsupervised setting we only use the input features). This dataset has $N = 515345$ data points. We split the data into training, test, and validation sets in the proportion $7 : 2 : 1$. For the encoder we use a fully connected (non-Bayesian) neural network with 500 hidden units in each of the two hidden layers. For the decoder we use a neural network with a single hidden layer with 500 hidden units. All non-linearities are ReLU. We use the Adam optimizer [13] and mini-batches of size 2000 during training. We use mean field (Normal) variational distributions for all weight matrices in the decoder. We do a grid search over the hyperparameters of the optimizer and use the validation set to choose the number of epochs to train. For all three model variants this procedure resulted in the following choices: default Adam hyperparameters and 1500 epochs of training. We use a latent dimension of 30. We report test log likelihoods that make use of an importance weighted estimator that draws 500 $\times$ 100 samples per data point (100 samples inside the log averaged over 500 trials).

6.8.3 Fast Prediction

We take a ResNet50 [8] that is pre-trained[9] on ImageNet [24] and then lop off the final layer and replace it with the following neural network architecture: $2048 - 1000 - 1000 - 1000$. Here the first two dashes represent ReLU non-linearities and the final layer of outputs represents softmax logits. We learn the first weight matrix (2048 - 1000) using MLE and are Bayesian about the subsequent weight matrices (we use mean field variational distributions). We do not fine-tune the weight matrices inherited from the pre-trained ResNet50. Our test set and validation set consist of 40k and 10k images, respectively. We train[10] for up to 120 epochs and use the validation set to fix optimization hyperparameters and determine how many epochs to train. In contrast to the typical approach taken in deep learning, we used a fixed size/crop for training images, i.e. we do not do any data augmentation.

[9] `https://pytorch.org/docs/stable/torchvision/models.html`

[10] Note that we also trained our network using the approximations and control variates described in Sec. 6.7, but we do not report those results here.

Fig. 1 is generated as follows. For the MC estimate of the predicted class probabilities, we draw a total of 4096 samples per datapoint. These samples are then combined via the following allocation:

$$N_{\text{inner}} \in [1, 2, 4, 8, 16, 32, 64, 128, 256, 512] \qquad \text{and} \qquad N_{\text{outer}} = 4096/N_{\text{inner}} \tag{43}$$

Here N_{inner}, which represents the number of samples inside the log, is the quantity plotted on the horizontal axis of Fig. 1. To form the deterministic approximation we follow Sec. 6.7.4.

6.9 Related Work

The approach most closely related to ours is probably the deterministic approximations in ref. [26] (indeed they compute some of the same ReLU integrals that we do). While we focus on single-layer neural networks, the distinct advantage of their approximation scheme is that it can be applied to networks of arbitrary depth. Thus some of our results are potentially complementary to theirs. Reference [11] also constructs deterministic variational objectives for the specific case of the ReLU activation function. Reference [19] considers quadratic piecewise linear bounds for the logistic-log-partion function in the context of Bernoulli-logistic latent Gaussian models. Finally, approaches for variance reduction in the stochastic variational inference setting include [14] and [25].

6.10 Assorted Remarks

1. The factorization assumption in Eqn. 3 can probably be weakened at the cost of dealing with special functions more exotic than the error function.
2. Note that the expression for the full (predictive) variance, which decomposes into three readily identified components, is given by:

$$\text{var}(\boldsymbol{x}) = \underbrace{\text{diag}(\boldsymbol{\Xi}_{\mathbf{B}}) \cdot (\boldsymbol{\Upsilon} - \boldsymbol{\Phi} \odot \boldsymbol{\Phi})}_{\text{variance from } \mathbf{A}} + \underbrace{\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{B}} \boldsymbol{\Phi}}_{\text{variance from } \mathbf{B}} + \underbrace{\beta^{-1}}_{\text{observation noise}} \tag{44}$$

3. Although we do not do so here, it would probably be straightforward to compute an all orders formula for the moments of the ReLU activation function, for which the main computational ingredient is the expectation

$$\mathbb{E}_{q(\mathbf{a})} \left[(\mathbf{a}^{\mathrm{T}} \boldsymbol{x})^n |\mathbf{a}^{\mathrm{T}} \boldsymbol{x}| \right] \tag{45}$$