Supplement - Structured Variational Learning of Bayesian Neural Networks with Horseshoe Priors

1. Conditional variational pre-activations

Recall from Section 4.2, that the variational pre-activation distribution is given by $q(b \mid a, \nu_l, \phi_{\beta_l}) = \mathcal{N}(b \mid \mu_b, \Sigma_b) = \mathcal{N}(b \mid M^T_{\beta_l|\nu_l} a, (a^T U_{\beta_l|\nu_l} a)V)$, where $U = \Psi + hh'$, and V is diagonal. To equation requires $M_{\beta_l|\nu_l}$ and $U_{\beta_l|\nu_l}$. The expressions for these follow directly from the properties of partitioned Gaussians.

For a particular layer l, we drop the explicit dependency on l from the notation. Recall that $B = \begin{bmatrix} \beta \\ \nu^T \end{bmatrix}$, and let $B \in \mathbb{R}^{m \times n}$, $\beta \in \mathbb{R}^{m-1 \times n}$, and $\nu \in \mathbb{R}^{n \times 1}$ $q(B \mid \phi_B) = \mathcal{MN}(B \mid M, U, V)$. From properties of the Matrix normal distribution, we know that a column-wise vectorization of B, $\vec{B} \sim \mathcal{N}(\vec{M}, V \otimes U)$. From this and Gaussian marginalization properties it follows that the j^{th} column $t_j = [\beta_j; \nu_j]$ of B is distributed as $t_j \sim \mathcal{N}(m_j, V_{jj}U)$, where m_j is the appropriate column of M. Conditioning on ν_j then yields, $q(\beta_j \mid \nu_j) = \mathcal{N}(\beta_j \mid \mu_{\beta_j|\nu_j}, \Sigma_{\beta_j|\nu_j})$, where

$$\begin{aligned}
\Sigma_{\beta_j|\nu_j} &= V_{jj}(\Psi_\beta + \frac{\Psi_\nu}{\Psi_\nu + h_\nu^2} h_\beta h_\beta^T) \\
\mu_{\beta_j|\nu_j} &= \mu_{\beta_j} + \frac{h_\nu(\nu_j - \mu_{\nu_j})}{\Psi_\nu + h_\nu^2} h_\beta
\end{aligned} \tag{1}$$

Rearranging, we can see that, $M_{\beta|\nu}$ is made up of the columns $\mu_{\beta_j|\nu_j}$ and $U_{\beta|\nu} = \Psi_\beta + \frac{\Psi_\nu}{\Psi_\nu + h_\nu^2} h_\beta h_\beta^T$.

2. Algorithmic details

The ELBO corresponding to the non-centered regularized HS model is,

$$\begin{aligned}
\mathcal{L}(\phi) = &\ \mathbb{E}[\ln \text{Inv-Gamma}(c \mid c_a, c_b)] + \mathbb{E}[\ln \text{Inv-Gamma}(\kappa \mid 1/2, 1/\rho_\kappa)] + \mathbb{E}[\ln \text{Inv-Gamma}(\rho_\kappa \mid 1/2, 1/b_\kappa^2)] \\
&+ \sum_n \mathbb{E}[\ln p(y_n \mid \beta, \mathcal{T}, \kappa, x_n)] \\
&+ \sum_{l=1}^{L-1} \sum_{k=1}^{K_L} \mathbb{E}[\ln \text{Inv-Gamma}(\lambda_{kl} \mid 1/2, 1/b_0^2)] \\
&+ \sum_{l=1}^{L-1} \mathbb{E}[\ln \text{Inv-Gamma}(\upsilon_l \mid 1/2, 1/\vartheta_l)] + \mathbb{E}[\ln \text{Inv-Gamma}(\vartheta_l \mid 1/2, 1/b_g^2)] \\
&+ \sum_{l=1}^{L-1} \mathbb{E}_{q(B_l)}[\ln \mathcal{N}(\beta_l \mid 0, \mathbb{I}) + \ln \text{Inv-Gamma}(\tau_l \mid 1/2, 1/\lambda_l)] + \sum_{k=1}^{K_L} \mathbb{E}[\ln \mathcal{N}(\beta_{kL} \mid 0, \mathbb{I})] + \mathbb{H}[q(\theta \mid \phi)].
\end{aligned} \tag{2}$$

We rely on a Monte-Carlo estimates to evaluate the expectation involving the likelihood $\mathbb{E}[\ln p(y_n \mid \beta, \mathcal{T}, \kappa, x_n)]$.

Efficient computation of the Matrix Normal Entropy The entropy of $q(B) = \mathcal{MN}(B \mid M, U, V)$ is given by $\frac{mn}{2} \ln(2\pi e) + \frac{1}{2} \ln |V \otimes U|$. We can exploit the structure of U and V to compute this efficiently. We note that $\ln |V \otimes U| = m \ln |V| + n \ln |U|$. Since V is diagonal $\ln |V| = \sum_j \ln V_{jj}$. Using the matrix determinant lemma we can efficiently compute $|U| = (1 + h'\Psi^{-1}h)|\Psi|$. Owing to the diagonal structure of Ψ, computing it's determinant and inverse is particularly efficient.

Fixed point updates The auxiliary variables ρ_κ, ϑ_l and ϑ_l all follow inverse Gamma distributions. Here we derive for λ_{kl}, the others follow analogously. Consider,

$$\begin{aligned}\ln q(\lambda_{kl}) &\propto \mathbb{E}_{-q_{\lambda_{kl}}}[\ln \text{Inv-Gamma}(\tau_{kl} \mid 1/2, 1/\lambda_{kl})] + \mathbb{E}_{-q_{\lambda_{kl}}}[\ln \text{Inv-Gamma}(\lambda_{kl} \mid 1/2, 1/b_0^2)],\\ &\propto (-1/2 - 1/2 - 1)\ln \lambda_{kl} - (\mathbb{E}[1/\tau_{kl}] + 1/b_0^2)(1/\lambda_{kl}),\end{aligned} \tag{3}$$

from which we see that,

$$\begin{aligned}q(\lambda_{kl}) &= \text{Inv-Gamma}(\lambda_{kl} \mid c, d),\\ c = 1, d &= \mathbb{E}[\frac{1}{\tau_{kl}}] + \frac{1}{b_0^2}.\end{aligned} \tag{4}$$

Since, $q(\tau_{kl}) = \ln \mathcal{N}(\mu_{\tau_{kl}}, \sigma^2_{\tau_{kl}})$, it follows that $\mathbb{E}[\frac{1}{\tau_{kl}}] = \exp\{-\mu_{\tau_{kl}} + 0.5 * \sigma^2_{\tau_{kl}}\}$. We can thus calculate the necessary fixed point updates for λ_{kl} conditioned on $\mu_{\tau_{kl}}$ and $\sigma^2_{\tau_{kl}}$. Our algorithm uses these fixed point updates given estimates of $\mu_{\tau_{kl}}$ and $\sigma^2_{\tau_{kl}}$ after each Adam step.

3. Algorithm

Algorithm 1 provides pseudocode summarizing the overall algorithm for training regularized HSBNN (with strictured variational approximations).

Algorithm 1 Regularized HS-BNN Training

```
1: **Input** Model $p(\mathcal{D}, \theta)$, variational approximation $q(\theta \mid \phi)$, number of iterations T.
2: **Output**: Variational parameters $\phi$
3: Initialize variational parameters $\phi$.
4: for T iterations do
5:     Update $\phi_c$, $\phi_\kappa$, $\phi_\gamma$, $\{\phi_{B_l}\}_l$, $\{\phi_{\upsilon_l}\}_l \leftarrow$ ADAM($\mathcal{L}(\phi)$).
6:     for all hidden layers $l$ do
7:         Conditioned on $\phi_{B_l}$, $\phi_{\upsilon_l}$ update $\phi_{\vartheta_l}$, $\phi_{\lambda_{kl}}$ using fixed point updates (Equation 4).
8:     end for
9:     Conditioned on $\phi_\kappa$ update $\phi_{\rho_\kappa}$ via the corresponding fixed point update.
10: end for
```

4. Experimental details

For regression problems we use Gaussian likelihoods with an unknown precision γ, $p(y_n \mid f(\mathcal{W}, x_n), \gamma) = \mathcal{N}(y_n \mid f(\mathcal{W}, x_n), \gamma^{-1})$. We place a vague prior on the precision,$\gamma \sim \text{Gamma}(6, 6)$ and approximate the posterior over γ using another variational distribution $q(\gamma \mid \phi_\gamma)$. The corresponding variational parameters are learned via a gradient update during learning.

Regression Experiments For comparing the reg-HS and HS models we followed the protocol of (Hernandez-Lobato & Adams, 2015) and trained a single hidden layer network with 50 rectified linear units for all but the larger "Protein" and "Year" datasets for which we train a 100 unit network. For the smaller datasets we train on a randomly subsampled 90% subset and evaluate on the remainder and repeat this process 20 times. For "Protein" we perform 5 replications and for "Year" we evaluate on a single split. For, VMG we used 10 pseudo-inputs, a learning rate of 0.001 and a batch size of 128.

Reinforcement learning Experiments We used a learning rate of $2e - 4$. For the $2D$ map domain we trained for 1500 epochs and for acrobot we trained for 2000 epochs.

5. Additional Experimental results

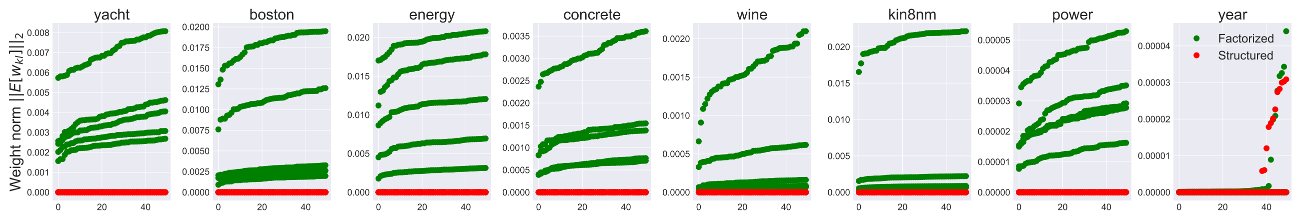

Figure 1. Structured variational approximations consistently recover solutions that exhibit stronger shrinkage. We plot the 50 smallest $||w_{kl}||_2$ recovered by the two approximations on five random trials on a number of UCI datasets.

In Figure 1 we provide further shrinkage results from the experiments described in the main text comparing regularized Horseshoe models utilizing factorized and structured approximations.

Shrinkage provided by fully factorized Horseshoe BNNs on UCI benchmarks Figure 2 illustrates the shrinkage afforded by 50 unit HS-BNNs using fully factorized approximations. Similar to factorized regularized Horseshoe BNNs limited compression is achieved. Figures On some datasets, we do not achieve much compression and all 50 units are used. A consequence of the fully factorized approximations providing weaker shrinkage as well as 50 units not being large enough to model the complexity of the dataset.

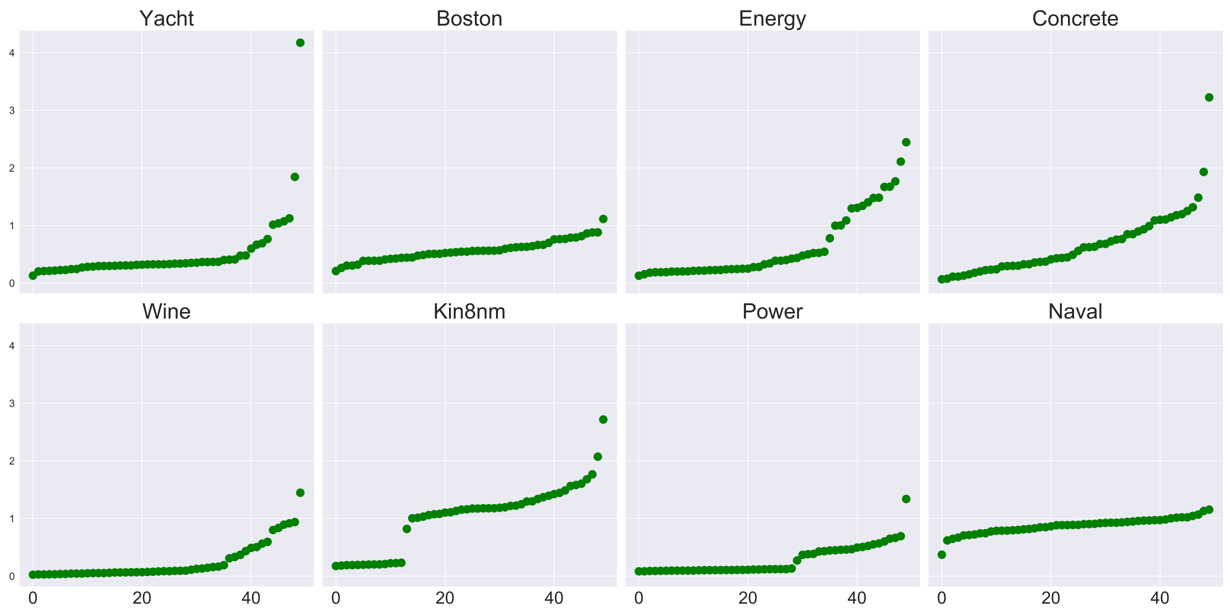

Figure 2. We plot $||w_{kl}||_2$ recovered by the HS-BNN using the fully factorized variational approximation on a number of UCI datasets.

6. Prior samples from networks with HS and regularized Horseshoe priors

To provide further intuition into the behavior of networks with Horseshoe and regularized Horseshoe priors we provide functions drawn from networks endowed with these priors. Figure 3 plots five random functions sampled from one layer networks with varying widths. Observe that the regularized horseshoe distribution leads to smoother functions, thus affording stronger regularization. As demonstrated in the main paper, this stronger regularization leads to improved predictive performance when the amount of training data is limited.

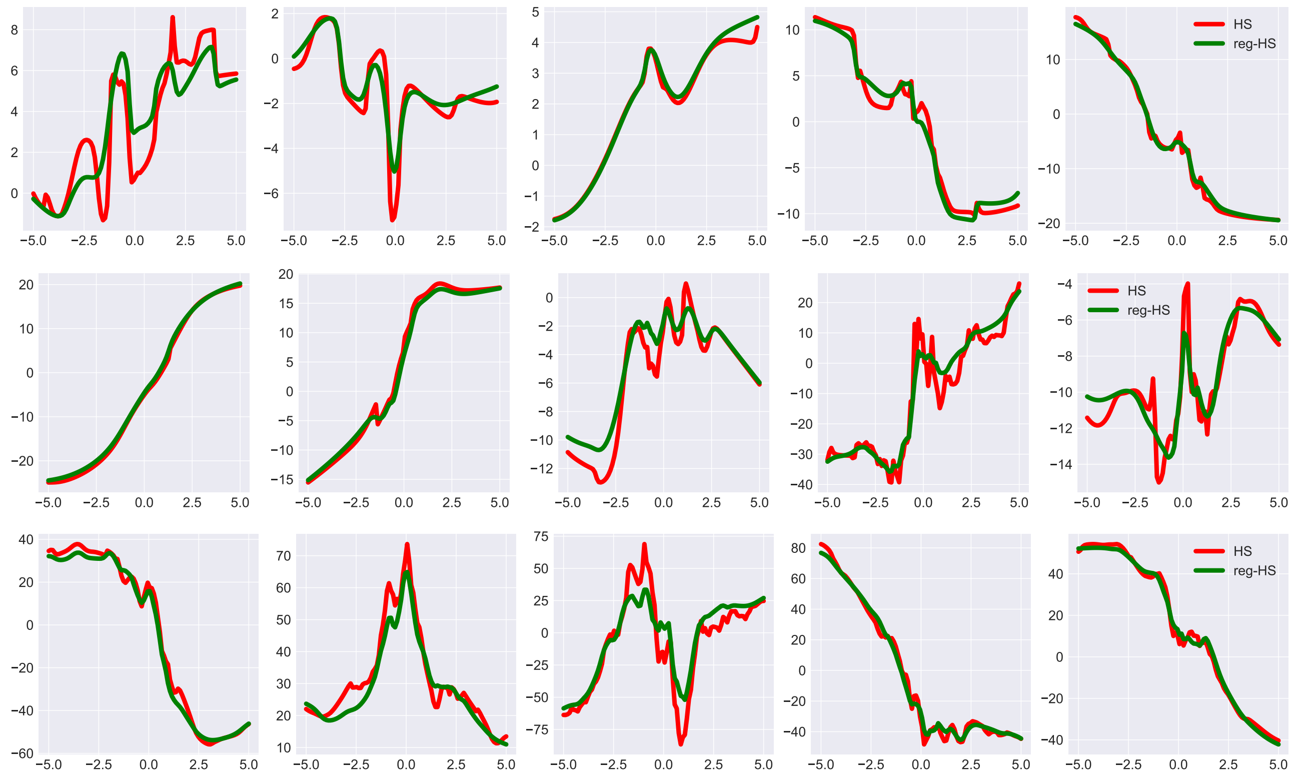

Figure 3. Each row displays five random samples drawn from a single layer network with TANH non-linearities. The top row contains samples from a 50 unit network, the middle row contains samples from a 500 unit network and the bottom row displays samples from a 5000 unit network. Matched samples from the regularized HS and HS priors were generated by sharing β_{kl} samples between the two. The hyper-parameters used were $b_0 = b_g = 1$, $c_a = 2$ and $c_b = 6$.