Benchmarking the Neural Linear Model for Regression

Sebastian W. Ober
University of Cambridge, Cambridge, UK
swo25@cam.ac.uk

Carl Edward Rasmussen
University of Cambridge, Cambridge, UK
cer54@cam.ac.uk

PROWLER.io, Cambridge, UK

Abstract
The neural linear model is a simple adaptive Bayesian linear regression method that has recently been used in a number of problems ranging from Bayesian optimization to reinforcement learning. Despite its apparent successes in these settings, to the best of our knowledge there has been no systematic exploration of its capabilities on simple regression tasks. In this work we characterize these on the UCI datasets, a popular benchmark for Bayesian regression models, as well as on the recently introduced UCI “gap” datasets, which are better tests of out-of-distribution uncertainty. We demonstrate that the neural linear model is a simple method that shows competitive performance on these tasks.

1. Introduction
Despite the recent successes that neural networks have shown in an impressive range of tasks, they tend to be overconfident in their predictions (Guo et al., 2017). Bayesian neural networks (BNNs; Neal (1995)) attempt to address this by providing a principled framework for uncertainty estimation in predictions. However, inference in BNNs is intractable to compute, requiring approximate inference techniques. Of these, Monte Carlo methods and variational methods, including dropout (Gal and Ghahramani, 2016), are popular; however, the former are difficult to tune, and the latter are often limited in their expressiveness (Foong et al., 2019b; Yao et al., 2019; Foong et al., 2019a).

The neural linear model represents a compromise between tractability and expressiveness for BNNs in regression settings: instead of attempting to perform approximate inference over the entire set of weights, it performs exact inference on only the last layer, where prediction can be done in closed form. It has recently been used in active learning (Pinsler et al., 2019), Bayesian optimization (Snoek et al., 2015), reinforcement learning (Riquelme et al., 2018), and AutoML (Zhou and Precioso, 2019), among others; however, to the best of our knowledge, there has been no systematic attempt to benchmark the model in the simple regression setting. In this work we do so, first demonstrating the model on a toy example, followed by experiments on the popular UCI datasets (as in Hernández-Lobato and Adams (2015)) and the recent UCI gap datasets from Foong et al. (2019b), who identified (along with Yao et al. (2019)) well-calibrated ‘in-between’ uncertainty as a desirable feature of BNNs.
2. Methods

![Figure 1: Predictive distributions for the toy problem. Each shaded region represents one predictive standard deviation.](image)

In this section, we briefly describe the different models we train in this work, which are variations of the neural linear (NL) model, in which a neural network extracts features from the input to be used as basis functions for Bayesian linear regression. The central issue in the neural linear model is how to train the network: in this work, we provide three different models, with a total of four different training methods. For a more complete mathematical description of the models, refer to Appendix A; we summarize the models in Appendix C.

**MAP NL** Following the work of Snoek et al. (2015), we can first train the neural network using maximum a posteriori (MAP) estimation. After this training phase, the outputs of the last hidden layer of the network are used as the features for Bayesian linear regression. To reduce overfitting, the noise variance and prior variance (for the Bayesian linear regression) are subsequently marginalized out by slice sampling (Neal et al., 2003) according to the tractable marginal likelihood, using uniform priors. We refer to this model as the maximum a posteriori neural linear model (which we abbreviate as MAP-L NL, where L is the number of hidden layers in the network). We tune the hyperparameters for the MAP estimation via Bayesian optimization (Snoek et al., 2012).

**Regularized NL** The MAP NL model’s basis functions are learned independently of the final model’s predictions. This is an issue for uncertainty quantification, as MAP training has no incentive to learn features useful for providing uncertainty in out-of-distribution areas. To address this issue, we propose to learn the features by optimizing the (tractable) marginal likelihood with respect to the network weights (previous to the output layer), treating them as hyperparameters of the model in an approach analogous to hyperparameter optimization in Gaussian process (GP) regression (Rasmussen and Williams, 2006). However, unlike in GP regression, the per-iteration computational cost of this method is linear in the size of the data. We additionally regularize the weights to reduce overfitting, resulting in a model we call regularized neural linear (which we abbreviate as Reg-L NL). As in the MAP NL model, we marginalize out the noise and prior variances via slice sampling. We tune the regularization and other hyperparameters via Bayesian optimization.
Bayesian noise NL  Instead of using slice sampling for the noise variance, we can place a normal-inverse-gamma ($N$-$\Gamma^{-1}$) prior on the weights and noise variance. This formulation is still tractable, and integrates the marginalization of the noise variance into the model itself, rather than having it implemented after the features are learned. Additionally, the $N$-$\Gamma^{-1}$ prior can act as a regularizer, meaning that we can avoid using Bayesian optimization to tune the prior parameters by jointly optimizing the marginal likelihood over all hyperparameters. However, this risks overfitting. Therefore, we consider training this model, which we call the Bayesian noise (BN) neural linear model, both by maximizing the marginal likelihood for all parameters (including prior parameters), and by tuning the prior parameters with Bayesian optimization. We abbreviate the first as $BN(ML)$-NL and the second as $BN(BO)$-NL. Finally, in both cases we slice sample the remaining (non-weight) hyperparameters.

3. Experiments

We compare these models on a toy problem, the UCI datasets, and the UCI “gap” datasets (Foong et al., 2019b). In all experiments, we consider 1- and 2-layer ReLU fully-connected networks with 50 hidden units in each layer (except for the toy problem, where we only consider 2-layer networks). We additionally provide results for simple MAP inference as a baseline. For experimental details, refer to Appendix B. We provide additional experimental results in Appendix D.

Toy problem  We construct a synthetic 1-D dataset comprising 100 train and 100 test pairs $(x, y)$, where $x$ is sampled i.i.d. in the range $[-4, -2] \cup [2, 4]$ and $y$ is generated as $y = x^3 + \epsilon$, $\epsilon \sim \mathcal{N}(0, 9)$. This follows the example from Hernández-Lobato and Adams (2015), with the exception of the “gap” added in the range for $x$, which was motivated by Foong et al. (2019b) and Yao et al. (2019). We plot predictive distributions for each model in Figure 1. Somewhat surprisingly, the MAP-2 NL model seems to struggle more than MAP with uncertainty in the gap, while having better uncertainty quantification at the edges. Of the marginal likelihood-based methods, the BN(BO)-2 NL model qualitatively seems to perform the best.

UCI datasets  We next provide results on the UCI datasets in Hernández-Lobato and Adams (2015) (omitting the ‘year’ dataset due to its size), a popular benchmark for Bayesian regression models in recent years. We report test log likelihoods and RMSEs for all the
models in Appendix D.1, for both 1- and 2-layer architectures. We visualize average test log likelihoods for the models in Figure 2; we tabulate the log likelihoods and RMSEs in Tables 2 and 3 in Appendix D.1, respectively.

From the figure and tables, we see that the BN(ML)-2 NL and BN(BO)-2 NL models have the best performance on these metrics, with values competitive with those in the literature for other BNN-based methods (Hernández-Lobato and Adams, 2015; Gal and Ghahramani, 2016; Bui et al., 2016; Tomczak et al.). In fact, these neural linear methods tend to achieve state-of-the-art or near state-of-the-art neural network performance on the ‘energy’ and ‘naval’ datasets, although on some other datasets their performance is worse. While the performance of the Reg-L NL model is decent, it performs worse than the BN-L NL models, showing the advantage of a Bayesian treatment of the noise variance.

**UCI gap datasets** Finally, we provide results on the UCI “gap” datasets proposed by Foong et al. (2019b), which consists of training and testing splits that artificially contain gaps in the training set, ensuring that the model will only succeed if it can represent uncertainty in-between gaps in the data. We again visualize test log likelihoods in Figure 3 while tabulating log likelihoods and RMSEs in Tables 4 and 5 in Appendix D.1.

![Figure 3: Average test log likelihoods (nats) for the UCI gap datasets](image)

Our results on the MAP-based models in Figure 3 echo those of Foong et al. (2019b), showing catastrophic failure to express in-between uncertainty for some datasets (particularly ‘energy’ and ‘naval’). As they are more prone to overfitting, the Reg-L NL models also seem to show this failure, albeit to a lesser extent. However, the remaining models seem less prone to these failures, with the BN(BO)-2 NL model having by far the best performance.

### 4. Conclusions & Future Work

We have shown benchmark results for different variants of the neural linear model in the regression setting. Our results show that the successes these models have seen in other areas such as reinforcement and active learning are not unmerited, with the models achieving good performance despite their simplicity. Furthermore, they are not as susceptible to the inability to express gap uncertainty as mean-field variational inference or dropout.

While we have shown empirically that these methods can perform well, some questions still remain to be addressed in future work. The main question that remains concerns an issue that is central to determining whether the neural linear model is viable for practical use: is Bayesian optimization necessary for hyperparameter choice in the BN-L NL model,
or can the hyperparameters simply be set to “suitable” values? Additionally, our work suggests that exact inference on a subset of parameters can perform better than approximate inference on the entire set, at least for BNNs. We believe this broader issue is worthy of further investigation.

References


Appendix A. The Neural Linear Model

The neural linear model uses a neural network to parameterize basis functions for Bayesian linear regression by treating the output weights and bias of the network probabilistically, while treating the rest of the network’s parameters $\theta$ as hyperparameters. This can be used as an approximation to full Bayesian inference of the neural network’s parameters, with the main advantage being that this simplified case is tractable (assuming Gaussian prior and likelihood). Given the fact that there are significant redundancies in the weight-space posterior for BNNs, this tradeoff may not be a completely unreasonable approximation.

We now describe the model mathematically. Let $D := (X, Y) = \{(x_n, y_n)\}_{n=1}^N$, where $(x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, be the training data, and let

$$\phi_\theta(x_n) = [\phi_{\theta,1}(x_n), \ldots, \phi_{\theta,N_L}(x_n)]^T$$

$$= [\phi(y_1^L(x_n)), \ldots, \phi(y_{N_L}^L(x_n))]^T$$

represent the outputs (post-activations) of the last hidden layer of the neural network, which will be parameterized by all the weights and biases up to the last layer, $\theta$. We then define a weight vector $w \in \mathbb{R}^M = \mathbb{R}^{N_L+1}$ (this includes a bias term, augmenting $\phi_\theta(x)$ with a 1). If we define a design matrix $\Phi_\theta = [\phi_\theta(x_1), \ldots, \phi_\theta(x_N)]^T$, we can then define our model as

$$Y = \Phi_\theta w + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2),$$

where we treat $Y$ as a column vector of the $y_n$. Given an appropriate $\theta$, Bayesian inference of the weights $w$ is straightforward: given a prior $p(w) = \mathcal{N}(w; 0, \alpha I_M)$ on the weights, the
Benchmarking Neural Linear

posterior is given by

\[ p(w|D, \sigma^2) = \mathcal{N}(w|w_N, V_N) \propto \mathcal{N}(w; 0, \alpha I_M) \mathcal{N}(Y; \Phi_\theta w, \sigma^2), \]

\[ w_N = \frac{1}{\sigma^2} V_N \Phi^T \theta Y, \]

\[ V_N^{-1} = \frac{1}{\alpha} I_M + \frac{1}{\sigma^2} \Phi^T \theta \Phi_\theta. \]

The posterior predictive for a test input \(x^*\) is then given by

\[ p(y^*|x^*, D, \sigma^2) = \mathcal{N}(y^*; w_N^T \phi_\theta(x^*), \sigma^2 + \phi_\theta(x^*)^T V_N \phi_\theta(x^*)). \]

It now remains to be determined how to learn \(\theta\).

A.1. MAP Neural Linear

As described in Snoek et al. (2015), we can learn \(\theta\) by simply setting it to the values of the corresponding weights and biases in a maximum a posteriori (MAP)-trained network, maximizing the objective

\[ L_{MAP}(\theta_{Full}) = \log \mathcal{N}(Y; \Phi_\theta w, \sigma^2) - \gamma \|\theta_{Full}\|_2^2, \]

with respect to \(\theta_{Full}\) and \(\sigma^2\), where \(\theta_{Full}\) represents the parameters of the full network (which includes the output weights and bias), and \(\gamma\) is a regularization parameter. As in Snoek et al. (2015), once we have obtained \(\theta\) from \(\theta_{Full}\), we use can use Bayesian linear regression as outlined above. However, the question of setting \(\alpha\) still remains. To address this, we marginalize \(\alpha\) and \(\sigma^2\) out by slice sampling them according to the log marginal likelihood of the data:

\[ L_{\theta, \alpha, \sigma^2}(D) = \log p(Y|X, \sigma^2) = \log \int p(Y|X, w, \sigma^2) p(w) \, dw \]

\[ = -\frac{M}{2} \log \alpha - \frac{N}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \|Y - \Phi_\theta w_N\|^2 - \frac{1}{2\alpha} w^T N w_N \]

\[ - \frac{1}{2} \log |V_N| - \frac{N}{2} \log 2\pi. \]

In order to learn a suitable value of \(\gamma\), along with learning rates and number of epochs, we use Bayesian optimization. For a complete description of the experimental details, see Appendix B.

One key disadvantage of this approach is that it separates the feature learning from prediction: in particular, there is no reason for the network to learn features relevant for out-of-distribution prediction, particularly when it comes to uncertainty estimates.

A.2. Regularized Neural Linear

From a Bayesian perspective, the neural linear model can be interpreted as a Gaussian process model with a covariance kernel determined by a finite number of basis functions \(\phi_{\theta,i}\) with hyperparameters \(\theta\). Therefore, as in Gaussian process regression, we propose to
maximize the log marginal likelihood of the data, \( \mathcal{L}_{\theta, \alpha, \sigma^2}(\mathcal{D}) \), with respect to \( \theta \) and \( \sigma^2 \) as the hyperparameters of the model for an empirical Bayes approach.

Note that the computational complexity of this expression is \( \mathcal{O}(N + M^3) \), as opposed to the \( \mathcal{O}(N^3) \) cost typically seen in GP regression. This is because we are able to apply the Woodbury identity to obtain the determinant in terms of \( V_N \), which is \( M \times M \), due to the fact that there is a finite number of basis functions. Since we typically have that \( N \gg M \), this results in significant computational savings.

One issue with this Type-2 maximum likelihood approach is that it will tend to overfit to the training data due to the large number of hyperparameters \( \theta \). As a result, the noise variance \( \sigma^2 \) will tend to be pushed towards zero. One way of addressing this is by introducing a regularization scheme. There are many potential regularization schemes that could be introduced: we could regularize \( \theta \), \( \alpha \), or \( \sigma \) individually, or using any combination of the three. We found empirically that of these, simply regularizing \( \theta \) alone via \( L^2 \) regularization seemed the most promising approach. This results in a Type-2 MAP approach wherein we maximize

\[
\hat{\mathcal{L}}_{\theta, \alpha, \sigma^2}(\mathcal{D}) = \mathcal{L}_{\theta, \alpha, \sigma^2}(\mathcal{D}) - \gamma_W \| \theta_W \|_2^2 - \gamma_b \| \theta_b \|_2^2,
\]

where we have divided \( \theta \) into weights \( \theta_W \) and biases \( \theta_b \) and introduced regularization hyperparameters \( \gamma_W \) and \( \gamma_b \).

A.3. Bayesian Noise Neural Linear

An alternative to regularization would be to treat the noise variance in a Bayesian manner by integrating it out. Fortunately, for Bayesian linear regression this is still tractable with the use of a normal-inverse-gamma prior on the outputs weights and parameters

\[
p(w, \sigma^2) = N \Gamma^{-1}(w, \sigma^2; 0, \alpha I_M, a_0, b_0) = \mathcal{N}(w; 0, \sigma^2 \alpha I_M) \Gamma^{-1}(\sigma^2; a_0, b_0).
\]

The posterior has the form

\[
p(w, \sigma^2|\mathcal{D}) = N \Gamma^{-1}(w, \sigma^2; w_N, V_N, a_N, b_N),
\]

\[
w_N = V_N \Phi_\theta^T Y,
\]

\[
V_N^{-1} = \frac{1}{\alpha} I_M + \Phi_\theta^T \Phi_\theta,
\]

\[
a_N = a_0 + N/2,
\]

\[
b_N = b_0 + \frac{1}{2} (Y^T Y - w_N^T V_N^{-1} w_N),
\]

with posterior predictive

\[
p(y_s|x_s, \mathcal{D}) = T \left( y_s; w_N^T \phi_\theta(x_s), \frac{b_N}{a_N} (I_M + \phi_\theta(x_s)^T V_N \phi_\theta(x_s)), 2a_N \right),
\]

where \( T(\cdot; \mu, \Sigma, \nu) \) is a Student’s t-distribution with mean \( \mu \), scale \( \Sigma \), and degrees of freedom \( \nu \). As before, we train the network using empirical Bayes, where the marginal
Benchmarking Neural Linear

likelihood is given by

\[ p(Y|X) = T \left( Y; 0, \frac{b_0}{a_0}(I_M + \alpha \Phi \Phi^T), 2a_0 \right). \]

Note that by using the Woodbury identity it is possible to compute this in \( \mathcal{O}(N + M^3) \) computational cost as before.

Appendix B. Experimental Details

All neural networks tested were ReLU networks with one or two 50-unit hidden layers. When using a validation set, we set its size to be one fifth of the size of the training set, except for the toy example, where we used half the training set. We now describe the experimental setup for each model we used.

**MAP** For the MAP baseline, we select a batch size of 32. We subsequently use Bayesian optimization (see section B.1 for a description of the Bayesian optimization algorithm we use) to optimize four hyperparameters using validation log likelihood: the regularization parameter \( \gamma \), a learning rate for the weights, a learning rate for the noise variance, and the number of epochs. The regularization parameter is allowed to vary within the range corresponding to a log prior variance between -5 and 5. The learning rates are also optimized in log space in the range \([\log 1e-4, \log 1e-2]\). Finally, the number of epochs is set to vary between zero and the number required to obtain at least 10000 gradient steps (the number of epochs will thus vary with the size of the dataset given a constant batch size). We initialize the regularization parameter to 0.5, the learning rates at 1e-3, the noise variance at \( e^{-3} \), and the number of epochs at the maximum value. The network itself is optimized using ADAM (Kingma and Ba, 2014).

**MAP NL** For the MAP neural linear, we take the above optimal MAP network and obtain 200 slice samples of \( \alpha_W \) (the output weight prior variance), \( \alpha_b \) (the output bias prior variance), and \( \sigma^2 \) for Bayesian linear regression. We initialize \( \alpha_W = 1/50 \) and \( \alpha_b = 1 \), to match the scaling used in Neal (1995).

**Regularized NL** For the regularized NL model, there are five hyperparameters which we tune via Bayesian optimization: \( \gamma_W \), \( \gamma_b \), a learning rate for \( \theta \), a learning rate for \( \sigma^2 \), and the number of epochs. We allow \( \gamma_W \) and \( \gamma_b \) to vary within a range of log prior variances between -10 and 10, and the number of epochs to be in the range of \([0, 5000]\) (since each epoch corresponds to one gradient step). The ranges for the other parameters remain the same. We initialize \( \gamma_W \) and \( \gamma_b \) to 1, and the remaining parameters the same way as in the MAP model. We again initialize \( \alpha_W = 1/50 \) and \( \alpha_b = 1 \). As before, we use 200 slice samples to marginalize out \( \sigma^2 \), \( \alpha_W \), and \( \alpha_b \) after the Bayesian optimization was completed.

**Bayesian noise NL (ML)** Here we optimize the parameters \( \theta, a_0, b_0, \alpha_W \), and \( \alpha_b \) directly and jointly via the log marginal likelihood. We employ early stopping by tracking the validation log likelihood up to 5000 epochs, and also maximize the validation log likelihood over a grid of 10 learning rates ranging logarithmically from \( \log 1e-4 \) to \( \log 1e-2 \). We also initialize \( a_0 = b_0 = 1 \) and \( \alpha_W = \alpha_b = 1 \). Finally, we use slice sampling to obtain 200 samples to marginalize out these hyperparameters.
Table 1: Summary of the models presented. The first column lists the model; the second shows the optimization objective, while the third shows which parameters were optimized using this objective. Meanwhile, the fourth lists the parameters that were tuned using Bayesian optimization, while the final lists the parameters that slice sampling was performed on.

**Bayesian noise NL (BO)** Instead of optimizing over the hyperparameters jointly as in the BN(ML) model, we keep all except \( \theta \) fixed over each iteration of Bayesian optimization. We retain the same initializations, and allow the following ranges for the hyperparameters: \( a_0 \in [0, 20] \), \( b_0 \in [0, 10] \), \( \log \alpha_W \in [-10, 10] \), \( \log \alpha_b \in [-10, 10] \), with the ranges for the learning rate and number of epochs being the same as before. We retain the same initializations as well. The slice sampling also remains the same.

**B.1. Bayesian Optimization**

Here we describe the Bayesian optimization algorithm that we used throughout. In each case we attempt to maximize the validation log likelihood. We largely follow the formulation set out in Snoek et al. (2012). We use a Gaussian process with a Matérn-5/2 kernel with the model hyperparameters as inputs and the validation log likelihoods as outputs (normalizing the inputs and outputs). We first learn the kernel hyperparameters (including a noise variance) by maximizing the marginal likelihood of the GP, using 5000 iterations of ADAM (Kingma and Ba, 2014) with a learning rate of 1e-2. We then obtain 20 slice samples of the GP hyperparameters, before using the expected improvement acquisition function to find the next set of network hyperparameters to test. In total, we use 50 iterations of Bayesian optimization for each model, initialized with 10 iterations of random search.

**Appendix C. Summary of the Models**

In Table 1, we provide a summary of the models we use, describing which parameters are optimized and how (we exclude learning rates and the number of epochs from this table), and which are then slice sampled.

**Appendix D. Additional Experimental Results**

In this appendix, we provide the full results from the main text, before briefly describing empirically the effect of slice sampling on the models.
D.1. Tabulated Experimental Results

On the next pages, we present tables of average test log likelihoods and test RMSEs for the UCI and UCI gap datasets for all models. These tables also show the average rank of each model across all datasets, and highlight the best performing model for each dataset and overall.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAP-1</th>
<th>MAP-2</th>
<th>MAP-1 NL</th>
<th>MAP-2 NL</th>
<th>Reg-1 NL</th>
<th>Reg-2 NL</th>
<th>BN(ML)-1 NL</th>
<th>BN(ML)-2 NL</th>
<th>BN(BO)-1 NL</th>
<th>BN(BO)-2 NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boston</td>
<td>-2.87 ± 0.17</td>
<td>-2.85 ± 0.16</td>
<td>-2.73 ± 0.11</td>
<td>-2.70 ± 0.10</td>
<td>-2.82 ± 0.14</td>
<td>-2.79 ± 0.14</td>
<td>-2.84 ± 0.10</td>
<td>-2.50 ± 0.09</td>
<td>-2.60 ± 0.08</td>
<td>-2.64 ± 0.11</td>
</tr>
<tr>
<td>concrete</td>
<td>-3.22 ± 0.04</td>
<td>-3.28 ± 0.06</td>
<td>-3.17 ± 0.04</td>
<td>-3.31 ± 0.07</td>
<td>-3.20 ± 0.06</td>
<td>-3.22 ± 0.06</td>
<td>-3.11 ± 0.05</td>
<td>-3.10 ± 0.03</td>
<td>-3.11 ± 0.05</td>
<td>-3.10 ± 0.04</td>
</tr>
<tr>
<td>energy</td>
<td>-0.47 ± 0.07</td>
<td>-0.73 ± 0.05</td>
<td>-0.68 ± 0.06</td>
<td>-0.59 ± 0.06</td>
<td>-0.73 ± 0.05</td>
<td>-0.66 ± 0.05</td>
<td>-0.64 ± 0.03</td>
<td>-0.59 ± 0.04</td>
<td>-0.71 ± 0.03</td>
<td>-0.56 ± 0.04</td>
</tr>
<tr>
<td>kin8nm</td>
<td>1.07 ± 0.01</td>
<td>1.20 ± 0.01</td>
<td>1.11 ± 0.01</td>
<td>1.21 ± 0.01</td>
<td>1.16 ± 0.01</td>
<td>1.24 ± 0.01</td>
<td>1.12 ± 0.01</td>
<td>1.22 ± 0.01</td>
<td>1.22 ± 0.01</td>
<td>1.26 ± 0.01</td>
</tr>
<tr>
<td>naval</td>
<td>4.94 ± 0.30</td>
<td>5.71 ± 0.08</td>
<td>6.38 ± 0.05</td>
<td>6.62 ± 0.06</td>
<td>8.16 ± 0.04</td>
<td>8.57 ± 0.04</td>
<td>7.87 ± 0.04</td>
<td>8.71 ± 0.03</td>
<td>7.38 ± 0.05</td>
<td>8.05 ± 0.06</td>
</tr>
<tr>
<td>power</td>
<td>-2.84 ± 0.01</td>
<td>-2.80 ± 0.01</td>
<td>-2.81 ± 0.01</td>
<td>-2.78 ± 0.01</td>
<td>-2.77 ± 0.01</td>
<td>-2.77 ± 0.02</td>
<td>-2.79 ± 0.01</td>
<td>-2.75 ± 0.01</td>
<td>-2.79 ± 0.01</td>
<td>-2.73 ± 0.01</td>
</tr>
<tr>
<td>protein</td>
<td>-2.97 ± 0.01</td>
<td>-2.89 ± 0.01</td>
<td>-2.94 ± 0.00</td>
<td>-2.87 ± 0.01</td>
<td>-2.86 ± 0.00</td>
<td>-2.80 ± 0.00</td>
<td>-2.87 ± 0.00</td>
<td>-2.80 ± 0.00</td>
<td>-2.87 ± 0.00</td>
<td>-2.78 ± 0.01</td>
</tr>
<tr>
<td>wine</td>
<td>-0.98 ± 0.01</td>
<td>-0.97 ± 0.02</td>
<td>-0.98 ± 0.01</td>
<td>-0.98 ± 0.02</td>
<td>-0.98 ± 0.02</td>
<td>-0.97 ± 0.02</td>
<td>-0.98 ± 0.02</td>
<td>-0.97 ± 0.02</td>
<td>-0.98 ± 0.02</td>
<td>-0.96 ± 0.01</td>
</tr>
<tr>
<td>yacht</td>
<td>-1.57 ± 0.28</td>
<td>-1.62 ± 0.28</td>
<td>-1.17 ± 0.16</td>
<td>-1.45 ± 0.25</td>
<td>-1.28 ± 0.22</td>
<td>-1.53 ± 0.34</td>
<td>-1.14 ± 0.08</td>
<td>-1.01 ± 0.16</td>
<td>-1.11 ± 0.07</td>
<td>-0.98 ± 0.11</td>
</tr>
</tbody>
</table>

**Table 2:** Average Test Log Likelihoods (nats) on the UCI Datasets (Best Values in Bold, Errors are ±1 Standard Error)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAP-1</th>
<th>MAP-2</th>
<th>MAP-1 NL</th>
<th>MAP-2 NL</th>
<th>Reg-1 NL</th>
<th>Reg-2 NL</th>
<th>BN(ML)-1 NL</th>
<th>BN(ML)-2 NL</th>
<th>BN(BO)-1 NL</th>
<th>BN(BO)-2 NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boston</td>
<td>3.28 ± 0.24</td>
<td>3.27 ± 0.22</td>
<td>3.21 ± 0.21</td>
<td>3.13 ± 0.19</td>
<td>3.19 ± 0.21</td>
<td>3.21 ± 0.26</td>
<td>3.16 ± 0.22</td>
<td>3.07 ± 0.23</td>
<td>3.00 ± 0.17</td>
<td>2.95 ± 0.20</td>
</tr>
<tr>
<td>concrete</td>
<td>5.45 ± 0.13</td>
<td>5.31 ± 0.14</td>
<td>5.22 ± 0.15</td>
<td>5.18 ± 0.14</td>
<td>5.11 ± 0.15</td>
<td>4.91 ± 0.14</td>
<td>5.01 ± 0.12</td>
<td>5.09 ± 0.12</td>
<td>4.94 ± 0.12</td>
<td>4.96 ± 0.15</td>
</tr>
<tr>
<td>energy</td>
<td>0.52 ± 0.02</td>
<td>0.45 ± 0.01</td>
<td>0.44 ± 0.01</td>
<td>0.39 ± 0.01</td>
<td>0.45 ± 0.01</td>
<td>0.42 ± 0.01</td>
<td>0.43 ± 0.01</td>
<td>0.42 ± 0.01</td>
<td>0.29 ± 0.02</td>
<td>0.41 ± 0.02</td>
</tr>
<tr>
<td>kin8nm</td>
<td>0.08 ± 0.00</td>
<td>0.07 ± 0.00</td>
<td>0.06 ± 0.00</td>
<td>0.07 ± 0.00</td>
<td>0.07 ± 0.00</td>
<td>0.07 ± 0.00</td>
<td>0.06 ± 0.00</td>
<td>0.07 ± 0.00</td>
<td>0.06 ± 0.00</td>
<td>0.07 ± 0.00</td>
</tr>
<tr>
<td>naval</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>power</td>
<td>4.13 ± 0.03</td>
<td>3.99 ± 0.05</td>
<td>4.02 ± 0.03</td>
<td>3.90 ± 0.04</td>
<td>3.91 ± 0.04</td>
<td>3.77 ± 0.04</td>
<td>3.94 ± 0.04</td>
<td>3.74 ± 0.04</td>
<td>3.91 ± 0.04</td>
<td>3.70 ± 0.04</td>
</tr>
<tr>
<td>protein</td>
<td>4.70 ± 0.04</td>
<td>4.34 ± 0.02</td>
<td>4.58 ± 0.01</td>
<td>4.29 ± 0.02</td>
<td>4.24 ± 0.02</td>
<td>3.93 ± 0.01</td>
<td>4.25 ± 0.02</td>
<td>3.95 ± 0.01</td>
<td>4.25 ± 0.01</td>
<td>3.90 ± 0.03</td>
</tr>
<tr>
<td>wine</td>
<td>0.64 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.64 ± 0.01</td>
<td>0.64 ± 0.01</td>
<td>0.64 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.64 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.64 ± 0.01</td>
</tr>
<tr>
<td>yacht</td>
<td>0.67 ± 0.06</td>
<td>0.63 ± 0.06</td>
<td>0.60 ± 0.05</td>
<td>0.58 ± 0.05</td>
<td>0.62 ± 0.04</td>
<td>0.60 ± 0.06</td>
<td>0.74 ± 0.06</td>
<td>0.52 ± 0.04</td>
<td>0.71 ± 0.06</td>
<td>0.65 ± 0.07</td>
</tr>
</tbody>
</table>

**Table 3:** Test RMSEs on the UCI Datasets (Best Values in Bold, Errors are ±1 Standard Error)
Table 4: Average Test Log Likelihoods (nats) on the UCI Gap Datasets (Best Values in Bold, Errors are ±1 Standard Error)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAP-1</th>
<th>MAP-2</th>
<th>MAP-1 NL</th>
<th>MAP-2 NL</th>
<th>Reg-1 NL</th>
<th>Reg-2 NL</th>
<th>BN(ML)-1 NL</th>
<th>BN(ML)-2 NL</th>
<th>BN(BO)-1 NL</th>
<th>BN(BO)-2 NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boston</td>
<td>-2.85 ± 0.08</td>
<td>-2.85 ± 0.13</td>
<td>-2.74 ± 0.04</td>
<td>-2.83 ± 0.13</td>
<td>-2.77 ± 0.06</td>
<td>-2.93 ± 0.11</td>
<td>-2.94 ± 0.04</td>
<td>-2.78 ± 0.05</td>
<td>-2.76 ± 0.06</td>
<td>-2.75 ± 0.05</td>
</tr>
<tr>
<td>concrete</td>
<td>-3.64 ± 0.06</td>
<td>-3.63 ± 0.04</td>
<td>-3.58 ± 0.06</td>
<td>-3.57 ± 0.05</td>
<td>-3.56 ± 0.07</td>
<td>-3.62 ± 0.06</td>
<td>-3.51 ± 0.03</td>
<td>-3.54 ± 0.02</td>
<td>-3.45 ± 0.05</td>
<td>-3.48 ± 0.06</td>
</tr>
<tr>
<td>energy</td>
<td>-4.91 ± 1.65</td>
<td>-4.35 ± 1.65</td>
<td>-3.65 ± 1.01</td>
<td>-3.46 ± 1.04</td>
<td>-3.42 ± 0.97</td>
<td>-4.02 ± 1.18</td>
<td>-2.47 ± 0.61</td>
<td>-2.25 ± 0.47</td>
<td>-2.29 ± 0.51</td>
<td>-2.11 ± 0.53</td>
</tr>
<tr>
<td>kind-00m</td>
<td>0.96 ± 0.06</td>
<td>1.09 ± 0.03</td>
<td>1.03 ± 0.06</td>
<td>1.14 ± 0.03</td>
<td>1.04 ± 0.06</td>
<td>1.12 ± 0.06</td>
<td>1.05 ± 0.02</td>
<td>1.15 ± 0.02</td>
<td>1.02 ± 0.05</td>
<td>1.18 ± 0.02</td>
</tr>
<tr>
<td>naval</td>
<td>-9.33 ± 4.21</td>
<td>-56.77 ± 23.69</td>
<td>-2.82 ± 2.55</td>
<td>-44.71 ± 14.57</td>
<td>-7.20 ± 2.67</td>
<td>-24.31 ± 8.78</td>
<td>1.30 ± 0.67</td>
<td>1.46 ± 0.77</td>
<td>1.61 ± 0.60</td>
<td>1.85 ± 0.67</td>
</tr>
<tr>
<td>power</td>
<td>-2.88 ± 0.02</td>
<td>-2.91 ± 0.04</td>
<td>-2.86 ± 0.02</td>
<td>-2.89 ± 0.02</td>
<td>-2.85 ± 0.03</td>
<td>-2.86 ± 0.02</td>
<td>-2.99 ± 0.08</td>
<td>-2.86 ± 0.03</td>
<td>-2.85 ± 0.02</td>
<td>-2.85 ± 0.02</td>
</tr>
<tr>
<td>protein</td>
<td>-3.07 ± 0.01</td>
<td>-3.06 ± 0.02</td>
<td>-3.07 ± 0.01</td>
<td>-3.07 ± 0.02</td>
<td>-3.05 ± 0.01</td>
<td>-3.06 ± 0.02</td>
<td>-3.09 ± 0.01</td>
<td>-3.09 ± 0.01</td>
<td>-3.04 ± 0.01</td>
<td>-3.05 ± 0.01</td>
</tr>
<tr>
<td>wine</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.95 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
<td>-0.96 ± 0.01</td>
</tr>
<tr>
<td>yacht</td>
<td>-2.96 ± 0.71</td>
<td>-1.96 ± 0.14</td>
<td>-2.26 ± 0.58</td>
<td>-1.71 ± 0.22</td>
<td>-1.95 ± 0.29</td>
<td>-3.24 ± 1.01</td>
<td>-1.93 ± 0.34</td>
<td>-2.22 ± 0.36</td>
<td>-1.79 ± 0.16</td>
<td>-1.71 ± 0.19</td>
</tr>
</tbody>
</table>

Avg. Rank | 7.56 | 7.00 | 5.24 | 5.67 | 4.56 | 6.44 | 4.78 | 4.56 | 2.89 | 1.33 |

Table 5: Test RMSEs on the UCI Gap Datasets (Best Values in Bold, Errors are ±1 Standard Error)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAP-1</th>
<th>MAP-2</th>
<th>MAP-1 NL</th>
<th>MAP-2 NL</th>
<th>Reg-1 NL</th>
<th>Reg-2 NL</th>
<th>BN(ML)-1 NL</th>
<th>BN(ML)-2 NL</th>
<th>BN(BO)-1 NL</th>
<th>BN(BO)-2 NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>boston</td>
<td>3.91 ± 0.29</td>
<td>3.61 ± 0.16</td>
<td>3.76 ± 0.17</td>
<td>3.60 ± 0.16</td>
<td>3.85 ± 0.20</td>
<td>4.36 ± 0.52</td>
<td>3.76 ± 0.21</td>
<td>3.86 ± 0.21</td>
<td>3.78 ± 0.19</td>
<td>3.74 ± 0.19</td>
</tr>
<tr>
<td>concrete</td>
<td>8.07 ± 0.52</td>
<td>7.91 ± 0.22</td>
<td>7.65 ± 0.29</td>
<td>7.07 ± 0.20</td>
<td>7.99 ± 0.34</td>
<td>7.67 ± 0.17</td>
<td>7.81 ± 0.30</td>
<td>8.09 ± 0.21</td>
<td>7.43 ± 0.32</td>
<td>7.47 ± 0.44</td>
</tr>
<tr>
<td>energy</td>
<td>2.85 ± 0.88</td>
<td>3.34 ± 1.24</td>
<td>2.82 ± 0.89</td>
<td>3.50 ± 1.38</td>
<td>2.91 ± 0.94</td>
<td>3.08 ± 1.06</td>
<td>2.97 ± 1.05</td>
<td>3.12 ± 0.99</td>
<td>2.68 ± 0.87</td>
<td>2.74 ± 0.95</td>
</tr>
<tr>
<td>kind-00m</td>
<td>0.09 ± 0.01</td>
<td>0.08 ± 0.00</td>
<td>0.09 ± 0.01</td>
<td>0.07 ± 0.00</td>
<td>0.09 ± 0.01</td>
<td>0.08 ± 0.01</td>
<td>0.08 ± 0.00</td>
<td>0.08 ± 0.00</td>
<td>0.09 ± 0.00</td>
<td>0.07 ± 0.00</td>
</tr>
<tr>
<td>naval</td>
<td>0.02 ± 0.00</td>
<td>0.03 ± 0.00</td>
<td>0.10 ± 0.02</td>
<td>0.05 ± 0.01</td>
<td>0.06 ± 0.01</td>
<td>0.03 ± 0.01</td>
<td>0.02 ± 0.00</td>
<td>0.02 ± 0.00</td>
<td>0.02 ± 0.00</td>
<td>0.02 ± 0.00</td>
</tr>
<tr>
<td>power</td>
<td>4.32 ± 0.10</td>
<td>4.41 ± 0.15</td>
<td>4.25 ± 0.08</td>
<td>4.33 ± 0.09</td>
<td>4.17 ± 0.12</td>
<td>4.23 ± 0.08</td>
<td>4.73 ± 0.29</td>
<td>4.22 ± 0.14</td>
<td>4.19 ± 0.09</td>
<td>4.18 ± 0.10</td>
</tr>
<tr>
<td>protein</td>
<td>5.16 ± 0.05</td>
<td>5.07 ± 0.07</td>
<td>5.17 ± 0.06</td>
<td>5.09 ± 0.07</td>
<td>5.09 ± 0.06</td>
<td>5.08 ± 0.06</td>
<td>5.26 ± 0.06</td>
<td>5.26 ± 0.06</td>
<td>5.04 ± 0.06</td>
<td>5.08 ± 0.06</td>
</tr>
<tr>
<td>wine</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
<td>0.63 ± 0.01</td>
</tr>
<tr>
<td>yacht</td>
<td>1.16 ± 0.19</td>
<td>0.99 ± 0.07</td>
<td>1.17 ± 0.16</td>
<td>1.05 ± 0.10</td>
<td>1.32 ± 0.14</td>
<td>2.75 ± 1.42</td>
<td>2.57 ± 1.23</td>
<td>3.10 ± 1.48</td>
<td>1.50 ± 0.30</td>
<td>1.25 ± 0.11</td>
</tr>
</tbody>
</table>

Avg. Rank | 5.44 | 4.44 | 5.33 | 4.44 | 3.96 | 5.83 | 5.94 | 6.09 | 3.11 | 2.22 |
D.2. Effect of Slice Sampling

In this section, we briefly investigate the effect of slice sampling on the performance of the models. We first make plots of the predictive posterior distribution for each model trained on the toy problem of Section 3. These plots are visible in Figure 4. Note that the MAP-2 NL model simply becomes MAP inference. The most visible difference between Figure 4 and Figure 1 can be seen in the BN(BO)-2 NL model, which seems to have gained certainty at the edges while perhaps becoming slightly more uncertain in the gap; however, the effect in the gap is almost negligible. We observe the opposite effect in the MAP-2 NL model. Additionally, the Reg-2 NL model becomes slightly smoother. In general, however, it would seem that the effect of slice sampling for the toy problem is small.

![Predictive distributions for the toy problem, without slice sampling. Each shaded region represents one predictive standard deviation. Note that without slice sampling, the MAP-2 NL model simply becomes MAP-2.](image)

We then plot the differences in the log likelihoods and RMSEs between the full models (with slice sampling) and the equivalent models without the final slice sampling step, to observe any quantitative differences. These plots are shown in Figure 5 for the UCI datasets and Figure 6 for the UCI gap datasets. These plots show that in most cases the slice sampling improves the test log likelihoods, while the pattern for the change in RMSEs is not as clear. For both, slice sampling generally seems to improve models that are already struggling, in particular the MAP-based models. However, the improvement seen still seems to be very dependent on the specific model and dataset combination. However, as a general rule, it would seem that the magnitude of the gains made by slice sampling outweigh the potential for loss. Therefore, we contend that slice sampling is appropriate in most cases, although it can be safely avoided if required when using one of the BN-L NL models.
Figure 5: Differences in average test log likelihoods (nats) and RMSEs for the UCI datasets

Figure 6: Differences in average test log likelihoods (nats) and RMSEs for the UCI gap datasets