Scalable uncertainty quantification in neural networks

Suraj Yerramilli

Project report for IEMS 490: Uncertainty Quantification

1 Introduction

Neural networks and deep learning methods have achieved state of the art prediction performance across a wide variety of domains [1]. Ultimately, these models are to be used in decision making, which requires them to have good uncertainty representation. Unfortunately, neural networks are poor at quantifying uncertainty and tend to produce overconfident predictions [2]. This can have potentially harmful consequences when applied to real world settings [e.g. 3]. Thus, there is a need for rigorous uncertainty quantification (UQ) in neural networks.

Bayesian neural networks (BNNs) are a principled method for UQ in neural networks, and have been the gold standard for probabilistic inference in neural networks [4]. They represent uncertainty by placing a prior distribution over the parameters, and then marginalizing them over their posterior distribution given data to obtain predictions. There is no closed-form expression for the posterior distribution. In practice, one draws samples from the posterior through Markov chain Monte Carlo (MCMC) techniques. A commonly used MCMC algorithm is the Hamiltonian Monte Carlo (HMC), which requires gradients of the posterior to be supplied [4]. Unfortunately, MCMC techniques require full batch evaluations at each step, and this can be prohibitively expensive for large datasets. In recent years, several scalable alternatives and approximations for exact BNNs have been proposed [e.g. 5, 6, 7, 8, 9]. These methods, however, tend to sacrifice calibration quality for scalability.

In this work, we study this trade-off for some of these scalable methods using simple feed-forward neural network regression models trained on a few small and medium-sized datasets. The goal is to develop an understand as to which of these methods are likelier to work well in practice for large datasets. For developing potential scalable methods, we would also like to identify certain apsects of the posterior distribution that need to be captured for good UQ, and those that can be ignored with minimal impact of UQ.

The rest of the report is outlined as follows. We discuss some scalable methods and the evaluation metrics in Section 2. We discuss the setup and results of our experiments in Section 3. Finally, we offer concluding remarks in Section 4.

2 Methodology

We begin by specifying a standard regression setup for neural network regression with a point estimate for the parameters. Let y denote the output, and $\mathbf{x}$ denote the predictors. The regression model is as follows:

$$y = f(\mathbf{x}; \boldsymbol{\theta}) + \epsilon, \tag{2.1}$$

where $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$ is a random noise component, and $f(\mathbf{x}; \boldsymbol{\theta})$ is a neural network model with parameters $\boldsymbol{\theta}$. With these assumptions, the probabilistic model for the neural network is

$$y|\mathbf{x}, \boldsymbol{\theta} \sim \mathcal{N}\left(f(\mathbf{x}; \boldsymbol{\theta}), \sigma^2\right). \tag{2.2}$$

Given training data with n observations $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, one can compute the maximum a-posteriori (MAP) estimate of $\boldsymbol{\theta}$ and σ^2 by minimizing their negative log-posterior:

$$-\log p\left(\theta, \sigma^2 | \mathbf{y}_n\right) \propto \frac{1}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i; \boldsymbol{\theta})\right)^2 - \log p\left(\theta, \sigma^2\right), \tag{2.3}$$

where $\mathbf{y}_n = [y_1, \ldots, y_n]^T$ is the vector of outputs, and $p\left(\theta, \sigma^2\right)$ is the prior distribution on the parameters. The predictive distribution at a new input can then be obtained by plugging the MAP estimate in (2.2). This point estimate approach would ignore the estimation uncertainty. Bayesian methods take this uncertainty into account by marginalizing (2.2) over the posterior $p\left(\theta, \sigma^2 | \mathbf{y}_n\right)$. However, exact Bayesian methods require full batch evaluations and hence do not scale with the number of observations.

In the remainder of the section, we discuss a few approximation methods and the evaluation metrics used in this report.

2.1 Approximation methods for BNNs

We consider three different approximation methods for BNNs. Note that these are not the only methods in literature.

Ensembles The negative log-posterior (2.3) is typically non-convex and has multiple local optima. So, a simple baseline method for uncertainty quantification is an ensemble of neural network models, each corresponding to different local optima. This can achieved by a different random initialization of the the parameters for each model. The models can be trained in parallel to speed up the process. Bagging can help in further de-correlating the different models. However, this would come at the expense of increased bias, as each model on average is trained only 63% of training observations. We, therefore, will not use bagging.

Variational approximations Rather than compute the exact posterior, variational inference (VI) methods [e.g. 10, 5] optimize the parameters of a specified distribution to be closest to the posterior in terms of KL-divergence. The KL divergence objective allows the use of stochastic gradient optimization, and thus VI methods are scalable to large datasets. A commonly used approximating distribution is a multivariate normal with a diagonal covariance matrix - an independent normal distribution is optimized for each parameter. Note that this approach disregards correlations among the different parameters. To assess the impact of ignoring these correlations, we also test a multivariate normal distribution with a low-rank plus diagonal covariance matrix. For convenience, we will refer to the VI methods with these approximating distributions as "VI - Independent" and "VI - Low Rank" respectively.

Scalable MCMC approximations A different direction of work focuses on scaling HMC methods by allowing stochastic gradients computed from batches in place of the full gradient. One such method is the stochastic gradient HMC (SGHMC) [11]. A potential caveat with this approach is that it introduces additional tuning parameters, that have a significant influence on the performance. We will instead use an adaptive variant of the SGHMC proposed in [8], that internally adapts these additional tuning parameters.

2.2 Evaluation metrics

To assess the uncertainty representations produced by the different methods, we use two proper scoring rules [12] defined in a negative orientation - the 95% interval score (IS) and the continuously ranked probability score (CRPS). The 95% IS measures the quality of a 95% credible interval, while the CRPS is the measure of the overall quality of the predicted distribution. We define a 95% credible interval using the 2.5% and the 97.5% quantiles of a distribution. Since the predictive distributions computed by each of the methods in Section 2.1 are uniform mixtures of Gaussians, we need to numerically compute the quantiles and the CRPS through Monte Carlo sampling from this predictive distribution.

While our primary interest is in assessing UQ, we also want accurate predictions - models that produce many inaccurate predictions with large uncertainties are not useful. We use the root mean squared error (RMSE) to determine prediction accuracy.

3 Experiments

We test the different approaches on six regression benchmark datasets from the UCI repository [13], shown in Table 1. We use PyTorch[14] for building the neural network models and the Pyro [15] library for implementing MAP estimation and the variational approximations. For adaptive SGHMC, we use an implementation in the pybnn[1] library.

3.1 Experimental setup

We use an identical network architecture for all the datasets - a single hidden-layer neural network with 50 units and TanH as the non-linear activation function. We place independent $\mathcal{N}(0,1)$ priors on the weights and $\mathcal{N}(0.,10.)$ on the bias terms. We reparameterize the noise variance σ^2 using a log transform and place a $\mathcal{N}(-4,0.1)$ on $\log\sigma^2$.

For evaluation, we use 20 different train-test splits with a 90-10 split percentage. We use the same training procedure and tuning parameters for each approach across all datasets. In our preliminary runs, we have found that results do not vary significantly with different tuning parameters. For the VI and the adaptive SGHMC methods, we draw 50 samples from the posterior. For the ensemble method, we have used only 10 models in the ensemble. This is partly due to limitations in computational capacity at our end. In the VI - Low Rank approach, we set the rank of the low-rank part of the covariance matrix to be approximately the square root of the number of parameters.

3.2 Results and discussion

The results are shown in Table 2. An interesting observation is that the IS and CRPS are not always aligned with each other. In fact, for the smaller datasets, they indicate different

[1]code available at https://github.com/automl/pybnn

Table 1: Datasets used in the experiments

Dataset	Number of observations	Number of predictors
Auto	392	7
Boston	512	13
Concrete	1030	8
Abalone	4177	8
Kin8nm	8192	8
Cpu (small)	8192	12

approaches as the best. The 95% IS measures the quality of a specific 95% credible interval. One can obtain a different IS by changing the lower and upper quantiles defining the 95% interval. Also, the ranking of the methods from a 95% IS can be different from that of a 90% IS or a 99% IS. So, CRPS might be the more appropriate metric for assessing the quality of UQ.

Among the different approaches, we can see from these results that the adaptive SGHMC works the best. Except for the smaller Auto and Boston datasets, it clearly outperforms the other approaches in terms of UQ. Even in terms of prediction accuracy, only the ensemble approach comes close. However, the adaptive SGHMC approach performs much worse than the others on the Auto and Boston datasets. Given that these two datasets have fewer than 500 observations in each training split, the poor performance can be attributed to the noise that is introduced by using stochastic gradients at each HMC step.

A surprising result here is that the VI models do not do better than the simple ensemble approach with just 10 models in terms of UQ. Except for the two smaller datasets, they do worse in terms of prediction accuracy. This could perhaps be explained by the multi-modal nature of the posterior, and the type of information captured by each approximation, as illustrated in Figure 1. Since the VI methods use a normal approximation, they can only capture uncertainty around a single mode. On the other hand, each model in the ensemble approach potentially corresponds to a different local optima, and hence a different mode of the posterior. Therefore, they tend to produce more robust predictions than the VI approach. However, they pay a price by ignoring local uncertainty around the modes. Due to data shift between the training and validation sets, they do not capture the best solution in the neighborhood of each mode. This is counterbalanced by the variety in the solutions captured by the ensemble approach.

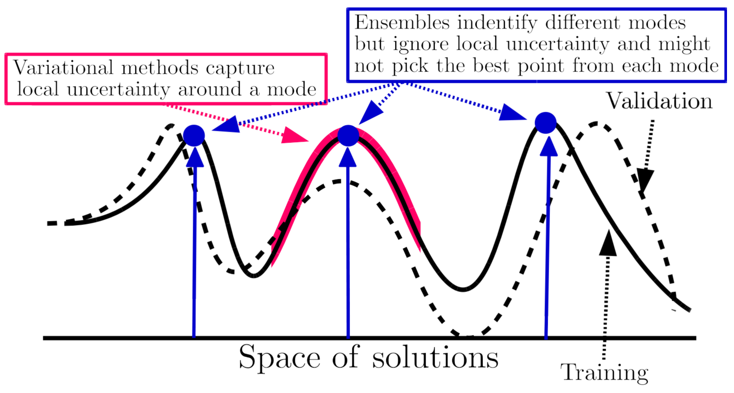
Ensembles indentify different modes but ignore local uncertainty and might not pick the best point from each mode
Variational methods capture local uncertainty around a mode
Validation
Space of solutions
Training

Figure 1: Illustration of the behavior of ensemble and variational approximation methods. The y-axis represents the loss function on training and validation data. Source: [16]

Table 2: Results on the benchmark regression datasets. The datasets are arranged in increasing order of the number of observations. For each dataset and each metric, the values in bold indicate the best performance.

Dataset	Metric	Ensembles	VI - Independent	VI - Low Rank	Adaptive SGHMC
Auto	RMSE	0.371	0.342	**0.338**	0.489
	Mean IS	3.525	**2.937**	3.154	6.649
	Mean CRPS	0.212	0.195	**0.194**	0.375
Boston	RMSE	0.383	0.352	**0.346**	0.424
	Mean IS	**2.422**	2.826	2.902	5.301
	Mean CRPS	0.219	0.185	**0.185**	0.268
Concrete	RMSE	0.269	0.344	0.336	**0.265**
	Mean IS	2.110	2.825	3.038	**1.661**
	Mean CRPS	0.143	0.196	0.194	**0.138**
Abalone	RMSE	**0.648**	0.656	0.653	0.648
	Mean IS	10.908	7.617	8.263	**4.130**
	Mean CRPS	0.397	0.379	0.384	**0.344**
Kin8nm	RMSE	0.293	0.351	0.338	**0.284**
	Mean IS	2.622	2.844	2.834	**1.449**
	Mean CRPS	0.172	0.204	0.198	**0.160**
Cpu (small)	RMSE	0.160	0.175	0.174	**0.157**
	Mean IS	1.545	1.020	1.079	**0.854**
	Mean CRPS	0.091	0.095	0.095	**0.084**

Finally, VI with a multivariate normal distribution with a low-rank plus diagonal covariance matrix does not do much better than one with a diagonal covariance matrix. This suggests that capturing correlations among the parameters is perhaps not as important for prediction quality and UQ as capturing the multiple modes in the exact posterior. Moreover, parameterizing correlations might not be scalable for larger neural networks.

4 Conclusions

In this work, we have tested a few scalable approximations for exact BNNs on a few small and medium sized datasets. We have found that the adaptive SGHMC method to work the best among these approaches. We have found the ensemble and variational approaches to have a few drawbacks, which are quite opposite of each other. While the ensemble approach ignores local uncertainty, the variational approach with normal distributions is able to capture only a single mode of the posterior. So, for a potential scalable UQ method to work well, it needs to capture the multi-modal nature of the posterior and estimate local uncertainty around each mode to be robust to data shift. Our results have also indicated that capturing correlation information among the different parameters is perhaps not as important as capturing the multi-modal nature.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[2] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, 2017.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer New York, 1996.

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1613–1622, 2015.

[6] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[7] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1050–1059, 2016.

[8] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[9] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[10] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

[11] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1683–1691, 2014.

[12] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank

Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[15] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, 20(1):973–978, 2019.

[16] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.